

# UC-8410-LX User's Manual

---

First Edition, October 2008

[www.moxa.com/product](http://www.moxa.com/product)

**MOXA**<sup>®</sup>

© 2008 Moxa Inc. All rights reserved.  
Reproduction without permission is prohibited.

# UC-8410-LX User's Manual

The software described in this manual is furnished under a license agreement and may be used only in accordance with the terms of that agreement.

## Copyright Notice

Copyright © 2008 Moxa Inc.  
All rights reserved.  
Reproduction without permission is prohibited.

## Trademarks

MOXA is a registered trademark of Moxa Inc.  
All other trademarks or registered marks in this manual belong to their respective manufacturers.

## 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](http://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

<b>Chapter 1</b>	<b>Introduction .....</b>	<b>1-1</b>
	Overview.....	1-2
	Software Architecture .....	1-2
	Journaling Flash File System (JFFS2).....	1-3
	Software Features .....	1-4
<b>Chapter 2</b>	<b>Getting Started .....</b>	<b>2-1</b>
	Powering on the UC-8410 .....	2-2
	Connecting the UC-8410 to a PC.....	2-2
	Serial Console.....	2-2
	Telnet Console.....	2-3
	SSH Console.....	2-5
	Configuring the Ethernet Interface .....	2-6
	Modifying Network Settings with the Serial Console .....	2-6
	Modifying Network Settings over the Network.....	2-7
	Test Program—Developing Hello.c.....	2-8
	Installing the Tool Chain (Linux) .....	2-8
	Checking the Flash Memory Space .....	2-8
	Compiling Hello.c .....	2-9
	Uploading and Running the “Hello” Program .....	2-10
<b>Chapter 3</b>	<b>Managing Embedded Linux .....</b>	<b>3-1</b>
	System Version Information.....	3-2
	Firmware Upgrade .....	3-2
	Upgrading the Firmware.....	3-2
	Loading Factory Defaults .....	3-5
	Enabling and Disabling Daemons.....	3-5
	Setting the Run-Level .....	3-8
	Setting the System Time .....	3-9
	TZ variable .....	3-9
	/etc/timezone.....	3-11
	Adjusting the System Time .....	3-11
	Setting the Time Manually .....	3-11
	NTP Client.....	3-12
	Updating the Time Automatically .....	3-13
	Cron—Daemon to Execute Scheduled Commands .....	3-13
	Connecting Peripherals .....	3-14
	USB Mass Storage.....	3-14
	CF Mass Storage.....	3-14
<b>Chapter 4</b>	<b>Managing Communication .....</b>	<b>4-1</b>
	Telnet/FTP .....	4-2
	DNS .....	4-2
	Web Service—Apache .....	4-3
	IPTABLES .....	4-5
	NAT.....	4-10
	NAT Example.....	4-10
	Enabling NAT at Bootup.....	4-11
	Dial-up Service—PPP.....	4-11

	PPPoE .....	4-15
	NFS (Network File System) Client .....	4-17
	Setting up the UC-8410 as an NFS Client .....	4-17
	Mail .....	4-17
	SNMP .....	4-18
	OpenVPN .....	4-18
	Package Management—ipkg .....	4-26
<b>Chapter 5</b>	<b>Programmer's Guide.....</b>	<b>5-1</b>
	Flash Memory Map .....	5-2
	Linux Tool Chain Introduction.....	5-2
	Debugging with GDB .....	5-4
	Device API .....	5-4
	RTC (Real Time Clock) .....	5-4
	Buzzer .....	5-5
	WDT (Watch Dog Timer) .....	5-5
	Digital I/O .....	5-9
	UART .....	5-13
	SRAM .....	5-15
	Make File Example .....	5-17
	Software Lock .....	5-17
<b>Appendix A</b>	<b>System Commands.....</b>	<b>A-1</b>
	Busybox (V1.10.4): Linux normal command utility collection .....	A-1
	File manager .....	A-1
	Editor .....	A-2
	Network .....	A-2
	Process .....	A-2
	Modules .....	A-3
	Other .....	A-3
	Special Moxa Utilities .....	A-4

Welcome to the Moxa UC-8400 Series of RISC-based communication platforms.

The UC-8410 embedded computer comes with 8 RS-232/422/485 serial ports, 3 Ethernet ports, 4 digital input channels, 4 digital output channels, a CompactFlash socket, and 2 USB 2.0 ports.

The UC-8410 computer uses the Intel XScale IXP-435 533 MHz RISC CPU. This powerful computing engine supports several useful communication functions, but will not generate too much heat. The built-in 16 MB NORFlash ROM can be used to store the operating system, and 256 MB SDRAM gives you enough memory to run your application software directly on the UC-8410. Moreover, the 32 MB NAND Flash provides the capacity for data storage.

The UC-8410 computer supports RS-232/422/485 serial ports, digital I/O, and has three LAN ports, making it ideal as a communication platform for industrial applications that require network redundancy.

Pre-installed with the Linux 2.6 platform, the UC-8410 provides an open software operating system for software program development. Software written for a desktop PC can be easily ported to the UC-8410 platform by using a common compiler, without needing to modify the code. This makes the UC-8410 an optimal solution for your industrial applications with the minimal cost and effort.

In addition to the standard model, a wide temperature version (-40 to 75°C) of the UC-8410 is also available for use in harsh industrial environments.

In this chapter, we cover the following topics:

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

## Overview

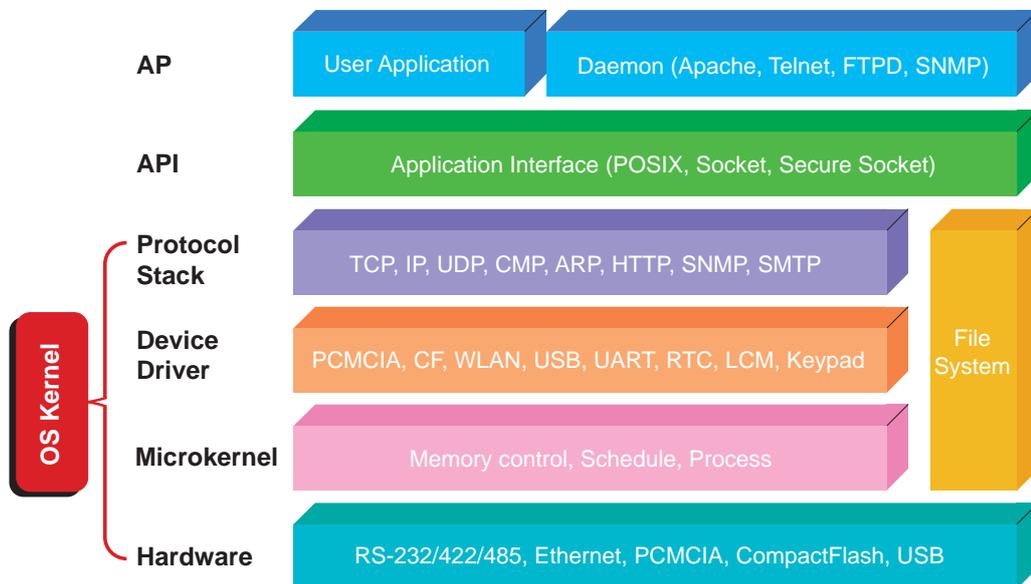
The UC-8410 computer, which features a RISC CPU, RAM memory, serial ports for connecting RS-232/422/485 devices, and 3 10/100 Mbps Ethernet ports, is designed for embedded applications.

The UC-8410 uses an Intel XScale IXP-435 533 Mhz RISC CPU. Unlike the X86 CPU, which uses a CISC design, the RISC architecture and modern semiconductor technology provide this computer with a powerful computing engine and communication functions, but without generating a lot of heat. The built-in 16 MB NOR Flash ROM can be used to store the operating system, and 256 MB of SDRAM gives you enough memory to run your application software directly on the UC-8410. Moreover, the 32 MB NAND Flash provides the capacity for data storage. In addition, the network capability provided by the 3 LAN ports built into the RISC CPU combined with the ability to control connected serial devices makes the UC-8410 an ideal communication platform for data acquisition and industrial control applications.

The UC-8410's pre-installed open source Linux operating system can run software written for desktop PCs, provided the software is ported to the computer with a GNU cross compiler. The process is easy, straightforward, and your programmer will not need to modify the source code. The OS, device drivers (e.g., Ethernet, SRAM, watchdog, and Buzzer control) and your own applications, can all be stored in the NOR Flash memory.

## Software Architecture

The Linux operating system that comes pre-installed on the UC-8410 follows the standard Linux architecture, making it easy to run programs that follow the POSIX standard. Program porting is done with the GNU Tool Chain provided by Moxa. In addition to Standard POSIX APIs, device drivers for the buzzer, SRAM and watchdog controls, and UART are also included in the Linux OS.



The UC-8410's built-in Flash ROM is divided into **Boot Loader**, **Linux Kernel**, **Root File System**, and **User Root File System** partitions.

In order to prevent user applications from crashing the Root File System, the UC-8410 uses a specially designed **Root File System with Protected Configuration** for emergency use. This **Root File System** comes with serial and Ethernet communication capability for users to load the **Factory Default Image** file. The user directory saves the user's settings and application.

To improve system reliability, the UC-8410 has a built-in mechanism that prevents the system from crashing. When the Linux kernel boots up, the kernel will mount the root file system for read only, and then enable services and daemons. During this time, the kernel will start searching for system configuration parameters via *rc* or *inittab*.

Normally, the kernel uses the Root File System to boot up the system. The Root File System is protected, and cannot be changed by the user, providing a "safe" zone.

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

## Journaling Flash File System (JFFS2)

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

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

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

The key features of JFFS2 are:

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

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

**Additional information about JFFS2 is available at:**

<http://sources.redhat.com/jffs2/jffs2.pdf>  
<http://developer.axis.com/software/jffs/>  
<http://www.linux-mtd.infradead.org/>

## Software Features

<b>Operating System</b>	
Boot Loader	Redboot
Kernel	Linux 2.6.23
Protocol Stack	ARP, PPP, CHAP, PAP, IPv4, ICMP, TCP, UDP, DHCP, FTP, SNMP V1/V2, 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
<b>Utilities</b>	
tinylogin	login and user manager utility
telnet	telnet client program
ftp	FTP client program
smtpclient	email utility
scp	Secure file transfer Client Program
<b>Daemons</b>	
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
openvpn	virtual private network
<b>Linux Tool Chain</b>	
Gcc (V4.2.1)	C/C++ Cross Compiler
GDB (V6.3)	Source Level Debug Server
Glibc (V2.2.5)	POSIX standard C library

# 2

## Getting Started

---

In this chapter, we explain how to connect the UC-8410, turn on the power, and then get started with programming and using other functions.

The following topics are covered:

- ❑ **Powering on the UC-8410**
- ❑ **Connecting the UC-8410 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
- ❑ **Test Program—Developing Hello.c**
  - Installing the Tool Chain (Linux)
  - Checking the Flash Memory Space
  - Compiling Hello.c
  - Uploading and Running the “Hello” Program

## Powering on the UC-8410

Connect the SG wire to the Shielded Contact located in the upper left corner of the UC-8410, and then power on the computer by connecting it to the power adaptor. It takes about 30 to 60 seconds for the system to boot up. Once the system is ready, the Ready LED will light up, and the model name of the computer will appear on the LCM display.



### ATTENTION

After connecting the UC-8410 to the power supply, it will take about 16 seconds for the operating system to boot up. The green Ready LED will not turn on until the operating system is ready.

## Connecting the UC-8410 to a PC

There are two ways to connect the UC-8410 to a PC: directly through the serial console port, or by Telnet over the network.

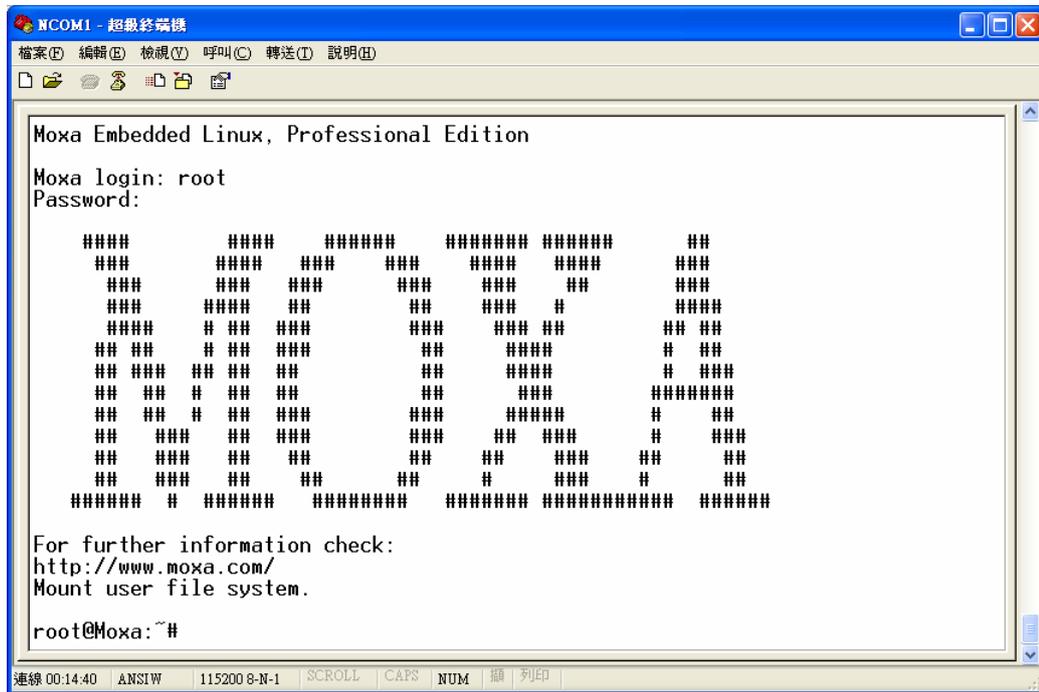
### Serial Console

The serial console port gives users a convenient way of connecting to the UC-8410's console utility. This method is particularly useful when using the computer for the first time. The signal is transmitted over a direct serial connection, so you do not need to know either of its 3 IP addresses in order to connect to the UC-8410.

Use the serial console port settings shown below.

<b>Baudrate</b>	115200 bps
<b>Parity</b>	None
<b>Data bits</b>	8
<b>Stop bits</b>	1
<b>Flow Control</b>	None
<b>Terminal</b>	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 IP addresses and netmasks, you can use Telnet to connect to the UC-8410's console utility. The default IP address and Netmask for each of the ports are given below:

	Default IP Address	Netmask
<b>LAN 1</b>	192.168.3.127	255.255.255.0
<b>LAN 2</b>	192.168.4.127	255.255.255.0
<b>LAN 3</b>	192.168.5.127	255.255.255.0

Use a cross-over Ethernet cable to connect directly from your PC to the UC-8410. You should first modify your PC's IP address and netmask so that your PC is on the same subnet as one of the UC-8410's 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 log in, type the Login name and password as requested. The default values are both root:

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

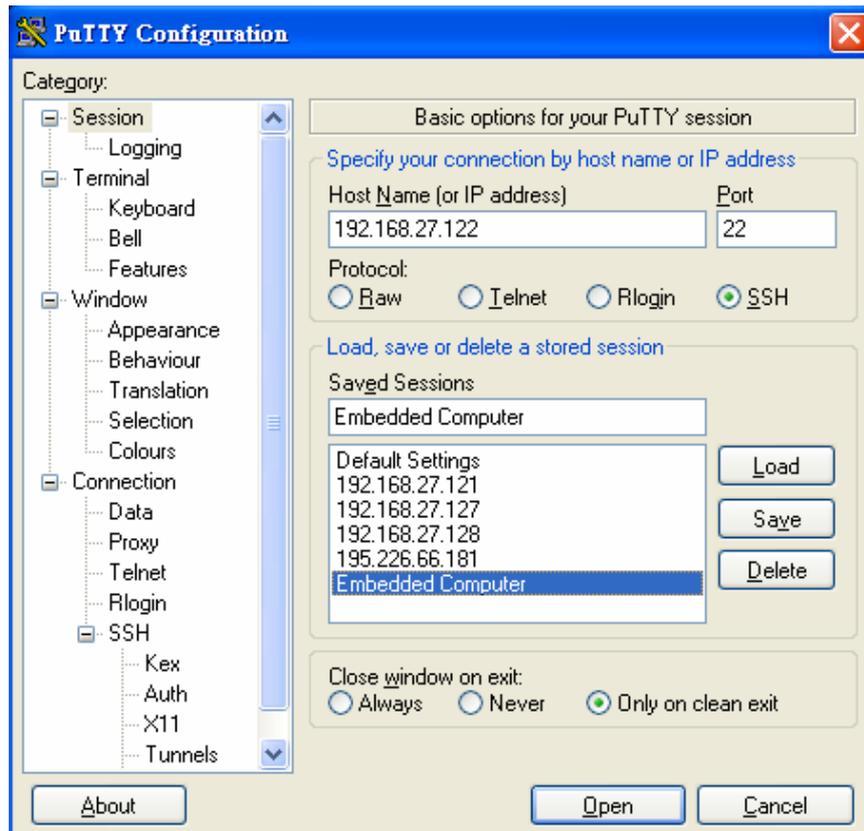


## SSH Console

The UC-8410 supports an SSH console to provide users with better security options.

### Windows Users

Click on the link <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html> to download PuTTY (free software) to set up an SSH console for the UC-8410 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 the UC-8410’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 the UC-8410’s console utility over the network.

## Configuring the Ethernet Interface

The network settings of the UC-8410 can be modified with the serial console, or online over the network.

### Modifying Network Settings with the Serial Console

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

1. Follow the instructions given in a previous section to access the console utility of the target computer via the serial console port, and then type `#cd /etc/network` to change directory.

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

2. Type `#vi interfaces` to edit the network configuration file with vi editor. You can configure the UC-8410's Ethernet ports for **static** or **dynamic** (DHCP) IP addresses.

#### Static IP addresses:

As shown below, 3 network addresses need to be modified: **address**, **network**, **netmask**, and **broadcast**. The default IP addresses are 192.168.3.127 for LAN1, 192.168.4.127 for LAN2, and 192.168.5.127 for LAN3, with default netmasks of 255.255.255.0.

```
# We always want the loopback interface.  
  
auto eth0 eth1 eth2 eth3 eth4 lo  
iface lo inet loopback  
  
# embedded ethernet LAN1  
iface eth0 inet static  
    address 192.168.3.127  
    network 192.168.3.0  
    netmask 255.255.255.0  
    broadcast 192.168.3.255  
  
# embedded ethernet LAN2  
iface eth1 inet static  
    address 192.168.4.127  
    network 192.168.4.0  
    netmask 255.255.255.0  
    broadcast 192.168.4.255  
  
# embedded ethernet LAN3  
iface eth2 inet static  
    address 192.168.5.127  
    network 192.168.5.0
```

**Dynamic IP Addresses:**

By default, the UC-8410 is configured for “static” IP addresses. To configure LAN ports to request an IP address dynamically, replace **static** with **dhcp** and then delete the address, network, netmask, and broadcast lines.

Default Setting for LAN1	Dynamic Setting using DHCP
<pre>iface eth0 inet <b>static</b>   address 192.168.3.127   network: 192.168.3.0   netmask 255.255.255.0   broadcast 192.168.3.255</pre>	<pre>iface eth0 inet <b>dhcp</b></pre>

```
Auto eth0 eth1 lo
iface lo inet loopback

iface eth0 inet dhcp

iface eth1 inet dhcp
```

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

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

**ATTENTION**

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

## Modifying Network Settings over the Network

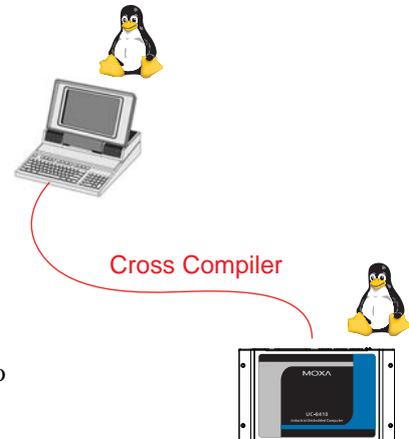
IP settings can be activated over the network, but the new settings will not be saved to the flash ROM without modifying the file `/etc/network/interfaces`.

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

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

## Test Program—Developing Hello.c

- Step 1:** Connect the UC-8410 to a Linux PC.
- Step 2:** Install the 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 the UC-8410 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-8410 units if needed).



## Installing the Tool Chain (Linux)

The PC must have the Linux operating system pre-installed before installing the UC-8410 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-8410 Tool Chain software is located on the UC-8410 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom
#/mnt/cdrom/tool-chain/Linux/arm-linux_2.0.sh
```

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

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

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

## Checking the Flash Memory Space

The UC-8410 uses a specially designed root file system. Only /tmp, /etc, /home, /usr/local/bin, /usr/local/sbin, /usr/local/libexec, and /usr/local/lib directories are writable. Others are read-only. The writable directories are mounted on /dev/mtdblock4. If the /dev/mtdblock4 is full, you will not be able to save data to the Flash ROM. Use the following command to calculate the amount of “Available” flash memory:

```
/>df -h
```

```

root@Moxa:/# df -h
Filesystem      Size      Used Available Use% Mounted on
rootfs          13.4M     9.8M      3.5M   74% /
/dev/root       13.4M     9.8M      3.5M   74% /
/dev/ram15      1.7M     19.0k      1.6M    1% /dev
/dev/ram0       499.0k    18.0k     456.0k   4% /var
/dev/mtdblock4 32.0M     1.9M     30.1M    6% /tmp
/dev/mtdblock4 32.0M     1.9M     30.1M    6% /home
/dev/mtdblock4 32.0M     1.9M     30.1M    6% /etc
tmpfs           252.5M    0         252.5M   0% /dev/shm
/dev/sdb1       483.4M    57.9M     425.5M  12% /var/sdb
root@Moxa:/#

```

If there isn't enough "Available" space for your application, you will need to delete some existing files. To do this, connect your PC to the UC-8410 with the console cable, and then use the console utility to delete the files from the UC-8410's flash memory.

## Compiling Hello.c

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

```

# cd /tmp/
# mkdir example
# cp -r /mnt/cdrom/example/* /tmp/example

```

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

```

#cd example/hello
#make

```

You should receive the following response:

```

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

```

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

**hello-release**—an IXP platform execution file (created specifically to run on the UC-8410)

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



### ATTENTION

Be sure to type the **#make** command from within the **/tmp/example/hello** directory, since the UC-8410's tool chain puts a specially designed **Makefile** in that directory. This special Makefile uses the **xscale-linux-gcc** compiler to compile the **hello.c** source code for the Xscale environment. If you type the **#make** command from any other directory, Linux will use the **x86** compiler (for example, **cc** or **gcc**). Refer to Chapter 5 to see a Make file example.

## Uploading and Running the “Hello” Program

Use the following command to upload **hello-release** to the UC-8410 via FTP.

1. From the PC, type:

```
#ftp 192.168.3.127
```

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

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

3. From the UC-8410, type:

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

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

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

## Managing Embedded Linux

---

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

In this chapter, we cover the following topics:

- ❑ **System Version Information**
- ❑ **Firmware Upgrade**
  - Upgrading the Firmware
  - Loading Factory Defaults
- ❑ **Enabling and Disabling Daemons**
- ❑ **Setting the Run-Level**
- ❑ **Setting the System Time**
  - TZ variable
  - /etc/timezone
- ❑ **Adjusting the System Time**
  - Setting the Time Manually
  - NTP Client
  - Updating the Time Automatically
- ❑ **Cron—Daemon to Execute Scheduled Commands**
- ❑ **Connecting Peripherals**
  - USB Mass Storage
  - CF Mass Storage

## System Version Information

To determine the hardware capability of your UC-8410 and what kind of software functions are supported, check which version of the firmware your UC-8410 is running. Contact Moxa to determine the hardware version. You will need the **Production S/N** (Serial number), which is located on the UC-8410's bottom label.

To check the kernel version, type:

```
#kversion
```

```
192.168.3.127 - PuTTY
root@moxa:~# kversion
UC8410 version 1.0
root@moxa:~# █
root@moxa:/# kversion -a
UC8410 version 1.0 Build 08091716
root@moxa:~# █
```

## Firmware Upgrade

### Upgrading the Firmware

The UC-8410's bootloader, kernel, and root file system are combined into one firmware file, which can be downloaded from Moxa's website ([www.moxa.com](http://www.moxa.com)). The name of the file has the form **FWR\_uc8400\_Va.b.c\_Build\_YYMMDDHH.hfm**, with "a.b.c" indicating the firmware version and YYMMDDHH indicating the build date. To upgrade the firmware, download the firmware file to a PC, and then transfer the file to the UC-8410 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
rootfs              13.4M     9.8M      3.5M   74% /
/dev/root           13.4M     9.8M      3.5M   74% /
/dev/ram15          1.7M      19.0k     1.6M    1% /dev
/dev/ram0           499.0k    18.0k    456.0k   4% /var
/dev/mtdblock4     32.0M     1.9M     30.1M   6% /tmp
/dev/mtdblock4     32.0M     1.9M     30.1M   6% /home
/dev/mtdblock4     32.0M     1.9M     30.1M   6% /etc
tmpfs               252.5M     0      252.5M   0% /dev/shm
/dev/sdb1           483.4M    57.9M    425.5M  12% /var/sdb
root@Moxa: /#
root@Moxa:/# upramdisk
root@Moxa:/# df -h
Filesystem          Size      Used Available Use% Mounted on
/dev/mtdblock2     14.0M    11.2M     2.8M   80% /
/dev/ram15          1.7M      18.0k     1.6M    1% /dev
/dev/ram0           499.0k    34.0k    440.0k   7% /var
/dev/mtdblock3     15.8M     2.6M     13.1M  17% /tmp
/dev/mtdblock3     15.8M     2.6M     13.1M  17% /home
/dev/mtdblock3     15.8M     2.6M     13.1M  17% /etc
/dev/ram1           38.7M    13.0k     36.7M   0% /mnt/ramdisk
root@Moxa:/# cd /mnt/ramdisk/
root@Moxa: /mnt/ramdisk#

```

The following instructions give the steps required to save the firmware file to the UC-8410'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 the UC-8410's built-in FTP client to transfer the firmware file (`FWR_uc8400_Va.b.c_Build_YYMMDDHH.hfm`) from the PC to the UC-8410:

```
/mnt/ramdisk> ftp <destination PC's IP>
Login Name: xxxx
Login Password: xxxx
ftp> bin
ftp> get FWR_uc8400_Va.b.c_Build_YYMMDDHH.hfm
```

```

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

```

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

```
#upgradefm FWR_uc8400_Va.b.c_Build_YMMDH.hfm
```

```

192.168.3.127 - PuTTY
root@Moxa:/mnt/ramdisk# upgradefm FWR_uc8400_Va.b.c_Build_YMMDH.hfm
Moxa UC-8400 Upgrade firmware utility version 1.0.
To check source firmware file context.
The source firmware file context is OK.
This step will upgrade firmware. All the data on flash will be destroyed.
Do you want to continue? (Y/N) :
Now upgrade the file [redboot].
Format MTD device [/dev/mtd0] ...
MTD device [/dev/mtd0] erase 128 Kibyte @ 60000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [kernel].
Format MTD device [/dev/mtd1] ...
MTD device [/dev/mtd1] erase 128 Kibyte @ 1a0000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [root-file-system].
Format MTD device [/dev/mtd2] ...
MTD device [/dev/mtd2] erase 128 Kibyte @ e00000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [directory].
Format MTD device [/dev/mtd5] ...
MTD device [/dev/mtd5] erase 128 Kibyte @ 20000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the new configuration file.
Upgrade the firmware is OK. Rebooting
█

```

## Loading Factory Defaults

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



### ATTENTION

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

If you have stored data in the writable partition, you will need to back up these files before resetting the system to default. On the UC-8410, the directories /tmp, /etc, /usr/local/bin, /usr/local/sbin, /usr/local/lib, /usr/local/libexec, and /home are mounted on /dev/mtdblock4. This means that all of the data stored in these directories will be destroyed after resetting to default.

## Enabling and Disabling Daemons

The following daemons are enabled when the UC-8410 boots up for the first time.

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

Type the command “ps” to list all processes currently running.

```

192.168.3.127 - PuTTY
root@Moxa:~# cd /etc
root@Moxa:/etc# ps
?K  PID USER      VSZ STAT COMMAND
  1 root        1316 S    init [3]
  2 root          0 SW<  [kthreadd]
  3 root          0 SW<  [ksoftirqd/0]
  4 root          0 SW<  [events/0]
  5 root          0 SW<  [khelper]
 30 root          0 SW<  [kblockd/0]
 33 root          0 SW<  [kseriod]
 52 root          0 SW   [pdflush]
 53 root          0 SW   [pdflush]
 54 root          0 SW<  [kswapd0]
 55 root          0 SW<  [aio/0]
613 root          0 SW<  [mtdblockd]
652 root          0 SW<  [ixp400_eth time]
655 root          0 SW<  [ixp400_eth time]
657 root          0 DW<  [EthMac Recovery]
667 root          0 SW<  [rpciod/0]
728 root          0 SW<  [khubd]
773 root          0 SW<  [scsi_eh_0]
774 root          0 SW<  [usb-storage]
788 root          0 SWN  [jffs2_gcd_mtd4]
814 root          0 SW   [ixp400_eth0]
820 root          0 SW   [ixp400_eth1]
834 root        1360 S    /bin/inetd
858 root       12536 S    /usr/bin/httpd -k start -d /etc/apache
861 bin         1300 S    /bin/portmap
867 root       2412 S    /bin/sh --login
872 root       1360 S    /bin/snmpd -c public
878 root       3508 S    /bin/sshd -f /etc/ssh/sshd_config
881 root       1292 S    /bin/reportip
883 nobody    12560 S    /usr/bin/httpd -k start -d /etc/apache
884 nobody    12560 S    /usr/bin/httpd -k start -d /etc/apache
885 nobody    12560 S    /usr/bin/httpd -k start -d /etc/apache
886 nobody    12560 S    /usr/bin/httpd -k start -d /etc/apache
887 nobody    12560 S    /usr/bin/httpd -k start -d /etc/apache

```

To run a private daemon, 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 vi editor to edit your application program. We use the sample program **tcps2-release**, and set it to run in the background.

```

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

```

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

```
192.168.3.127 - PuTTY
root@Moxa:~# ps -ef
PID  USER  VSZ  STAT  COMMAND
  1  root   1316  S     init [3]
  2  root    0  SW<   [kthreadd]
  3  root    0  SW<   [ksoftirqd/0]
  4  root    0  SW<   [events/0]
  5  root    0  SW<   [khelper]
 30  root    0  SW<   [kblockd/0]
 33  root    0  SW<   [kseriod]
 52  root    0  SW    [pdflush]
 53  root    0  SW    [pdflush]
 54  root    0  SW<   [kswapd0]
 55  root    0  SW    [aio/0]
613  root    0  SW    [mtdblockd]
652  root    0  SW    [ixp400_eth time]
655  root    0  SW    [ixp400_eth time]
657  root    0  DW    [Ethmac Recovery]
666  root    0  SW    [rpciod/0]
727  root    0  SW<   [khubd]
772  root    0  SW<   [scsi_eh_0]
773  root    0  SW<   [usb-storage]
788  root    0  SWN<  [jffs2_gcd_mtd4]
809  root    0  SW    [ixp400_eth0]
815  root    0  SW    [ixp400_eth1]
829  root   1360  S     /bin/inetd
832  bin    1300  S     /bin/portmap
838  root   2428  S     /bin/sh --login
843  root   1360  S     /bin/snmpd -c public
867  root   3508  S     /bin/sshd -f/etc/ssh/sshd_config
873  root   7284  S     /usr/bin/httpd -k start -d/etc/apache
876  root   1292  S     /bin/reportip
878  nobody 7308  S     /usr/bin/httpd -k start -d/etc/apache
879  nobody 7308  S     /usr/bin/httpd -k start -d/etc/apache
880  nobody 7308  S     /usr/bin/httpd -k start -d/etc/apache
881  nobody 7308  S     /usr/bin/httpd -k start -d/etc/apache
882  nobody 7380  S     /usr/bin/httpd -k start -d/etc/apache
883  root   1264  S     /root/tcps2-release
896  root   2156  R     ps

root@Moxa:~# █
```

## Setting the Run-Level

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

```
192.168.3.127 - PuTTY
root@Moxa:/etc/rc.d/rc3.d# ls
S20snmpd      S55ssh       S99showreadyled
S25nfs-server S99rnmologin
root@Moxa:/etc/rc.d/rc3.d# █
```

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

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

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

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

SxxRUNFILE stands for

S: start the run file while linux boots up.

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

RUNFILE: the file name.

```
192.168.3.127 - PuTTY
root@Moxa:/etc/rc.d/rc3.d# ls
S20snmpd      S55ssh       S99showreadyled
S25nfs-server S99rnmologin
root@Moxa:/etc/rc.d/rc3.d# ln -s /root/tcps2-release S60tcps2
root@Moxa:/etc/rc.d/rc3.d# ls
S20snmpd      S55ssh       S99showreadyled
S25nfs-server S99rnmologin S60tcps2
root@Moxa:/etc/rc.d/rc3.d# █
```

KxxRUNFILE stands for

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

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

RUNFILE: the file name.

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

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

## Setting the System Time

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

### TZ variable

TZ environment variable format

```
TZ=standardHH[:MM[:SS]][daylight[HH[:MM[:SS]]][,startdate[/start  
time],enddate[/endtime]]]
```

The time kept by the local machine should be a universal standard representation, such as Greenwich Mean Time (GMT) or Universal Time Coordinated (UTC), hereafter referred to as the universal reference time. For personal computers that are not sharing data across time zones, the local time is an adequate standard. To support a universal standard, all MKS utilities assume that times stored in the file system and returned by the operating system are stored in the universal reference time, and then translated to local times. The mapping from the universal reference time to local time is specified by the TZ (time zone) environment variable. If left undefined, the TZ variable defaults to the current time zone setting of your operating system.

```
Here are some possible settings for the North American Eastern time  
zone:  
    TZ=EST5EDT  
    TZ=EST0EDT  
    TZ=EST0
```

In the first case, the reference time is GMT and the stored time values are correct worldwide. A simple change of the TZ variable prints local time correctly, anywhere. In the second case, the reference time is Eastern Standard Time and the only conversion performed is for Daylight Savings Time. Therefore, there is no need to adjust the hardware clock for Daylight Savings Time twice per year. In the third case, the reference time is always the time reported. This is suggested if the hardware clock on your machine automatically adjusts for Daylight Savings Time, or you insist on manually resetting the hardware time twice a year.

```
Other examples include:  
    TZ=NST3:30NDT2:00  
    TZ=MSEZ-1
```

The first applies to Newfoundland, whereas the second works in most of Western Europe.

```
Here are some time zone scenarios that involve Daylight Savings Time  
specification:  
    TZ=PST0PDT-1  
    TZ=ACST-09:30ACDT-10:30,M10.5.0/2:00,M3.5.0/2:00
```

The first scenario shows the TZ of a person in Seattle who stores local time on a PC, but does not adjust the clock to agree with Daylight Savings Time. The stated time zone precedes the machine clock time by one hour when Daylight Savings Time is in effect. The second scenario shows the TZ set by a person in Australia who sets a PC clock to UTC and never adjusts it. The machine clock precedes UTC by 9.5 hours when Daylight Savings Time is not in effect, and by 10.5 hours when in effect. Daylight Savings Time is in effect from 2:00 am on the last Sunday in October until 2:00 am on the last Sunday in March. Adding in the file `/etc/rc.d/rc.local`, the timezone setting will be active after the Moxa embedded computer reboots.

```
export TZ=your_timezone_setting
```

Possible values for the TZ environment variables are listed below:

Hours From Greenwich Mean Time (GMT)	Value	Description
0	GMT	Greenwich Mean Time
+1	ECT	European Central Time
+2	EET	European Eastern Time
+2	ART	
+3	EAT	Saudi Arabia
+3.5	MET	Iran
+4	NET	
+5	PLT	West Asia
+5.5	IST	India
+6	BST	Central Asia
+7	VST	Bangkok
+8	CTT	China
+9	JST	Japan
+9.5	ACT	Central Australia
+10	AET	Eastern Australia
+11	SST	Central Pacific
+12	NST	New Zealand
-11	MIT	Samoa
-10	HST	Hawaii
-9	AST	Alaska
-8	PST	Pacific Standard Time
-7	PNT	Arizona
-7	MST	Mountain Standard Time
-6	CST	Central Standard Time
-5	EST	Eastern Standard Time
-5	IET	Indiana East
-4	PRT	Atlantic Standard Time
-3.5	CNT	Newfoundland

-3	AGT	Eastern South America
-3	BET	Eastern South America
-1	CAT	Azores

## /etc/timezone

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

The UC-8410 does not provide `/usr/share/zoneinfo/` files, so you need to copy a time zone information file to the UC-8410 and write over the original local time file.

1. The `/usr/share/zoneinfo` folder on a PC that is running standard Linux contains several time zone information files.
2. Copy the time zone file that you want to use to the UC-8410 to overwrite the original `/etc/localtime` file.
3. Type a date to check if the new time zone appears.

## Adjusting the System Time

### Setting the Time Manually

The UC-8410 has two time settings. One is the system time, and the other is the RTC (Real Time Clock) time kept by the UC-8410's hardware. Use the `#date` command to query the current system time or set a new system time. Use `#hwclock` to query the current RTC time or set a new RTC time.

Use the following command to query the system time:

```
#date
```

Use the following command to query the RTC time:

```
#hwclock
```

Use the following command to set the system time:

```
#date MMDDhhmmYYYY
```

MM = Month

DD = Date

hhmm = hour and minute

YYYY = Year

Use the following command to set the RTC time:

```
#hwclock -w
```

Write current system time to RTC

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

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

## NTP Client

The UC-8410 has a built-in NTP (Network Time Protocol) client that is used to initialize a time request to a remote NTP server. Use **#ntpdate <NTP server>** to update the system time.

```
#ntpdate time.stdtime.gov.tw
```

```
#hwclock -w
```

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

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



### ATTENTION

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

## Updating the Time Automatically

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

### Example shell script to update the system time periodically

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

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

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

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

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

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

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

## Cron—Daemon to Execute Scheduled Commands

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

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

Cron wakes up every minute, and checks each command to see if it should be run in the current minute. When executing commands, output is mailed to the owner of the crontab (or to the user named in the MAILTO environment variable in the crontab, if such a user exists).

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

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

The following example demonstrates how to use Cron.

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

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

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

**STEP2: Change mode of fixtime.sh**

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

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

Add the following line to the end of crontab:

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

**STEP4: Enable the cron daemon manually.**

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

**STEP5: Enable cron when the system boots up.**

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

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

## Connecting Peripherals

### USB Mass Storage

The UC-8410 supports PNP (plug-n-play), and hot pluggability for connecting USB mass storage devices. The UC-8410 has a built-in auto mount utility that eases the mount procedure. The connected USB mass storage device will be mounted automatically. You can check the location of the USB disk by mount command. The UC-8410 will be un-mounted automatically with **umount** when the device is disconnected.

**ATTENTION**

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

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

The UC-8410 only supports certain types of flash disk USB Mass Storage devices. Some USB flash disks and hard disks may not be compatible with the UC-8410. Check compatibility issues before you purchase a USB device to connect to the UC-8410.

### CF Mass Storage

The UC-8410 Embedded Computer does not support CompactFlash hot swap and PnP (Plug and Play) function. It is necessary to remove power source first before inserting or removing the CompactFlash card. The UC-8410 CF card doesn't support PNP. This means that user need to mount the CF card manually after inserting the CF mass storage.

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

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

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

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

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

# 4

## Managing Communication

---

In this chapter, we explain how to configure the UC-8410's various communication functions.

The following topics are covered:

- Telnet/FTP**
- DNS**
- Web Service—Apache**
- IPTABLES**
- NAT**
  - NAT Example
  - Enabling NAT at Bootup
- Dial-up Service—PPP**
- PPPoE**
- NFS (Network File System) Client**
  - Setting up the UC-8410 as an NFS Client
- Mail**
- SNMP**
- OpenVPN**
- Package Management—ipkg**

## Telnet/FTP

In addition to supporting Telnet client/server and FTP client/server, the UC-8410 also supports SSH and sftp client/server. To enable or disable the Telnet/ftp server, you first need to edit the file `/etc/inetd.conf`.

### Enabling the Telnet/ftp server

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

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

### Disabling the Telnet/ftp server

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

## DNS

The UC-8410 supports DNS client (but not DNS server). To set up DNS client, you need to edit three configuration files: `/etc/hosts`, `/etc/resolv.conf`, and `/etc/nsswitch.conf`.

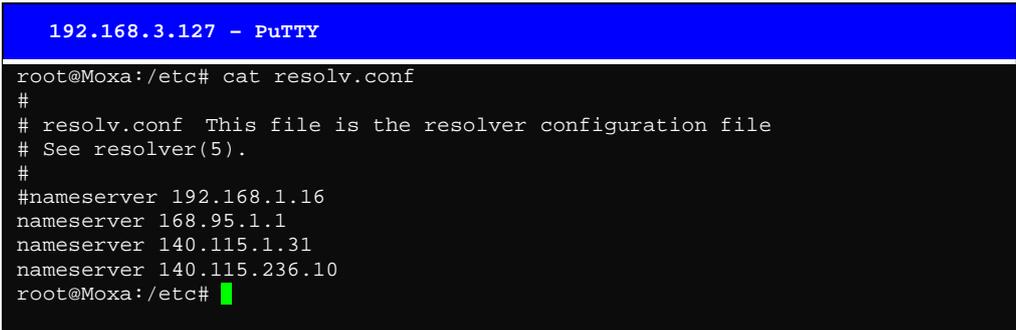
### `/etc/hosts`

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

### `/etc/resolv.conf`

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

```
nameserver 168.95.1.1
```



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

### `/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 `/home/httpd/index.html`. Save your own homepage to the following directory:

`/home/httpd/htdocs/`

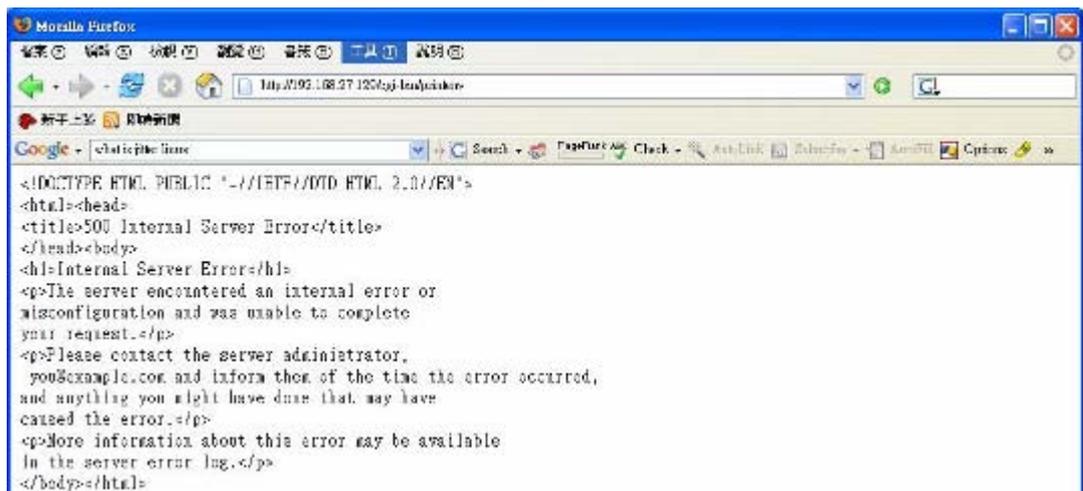
Save your CGI page to the following directory:

`/home/httpd/cgi-bin/`

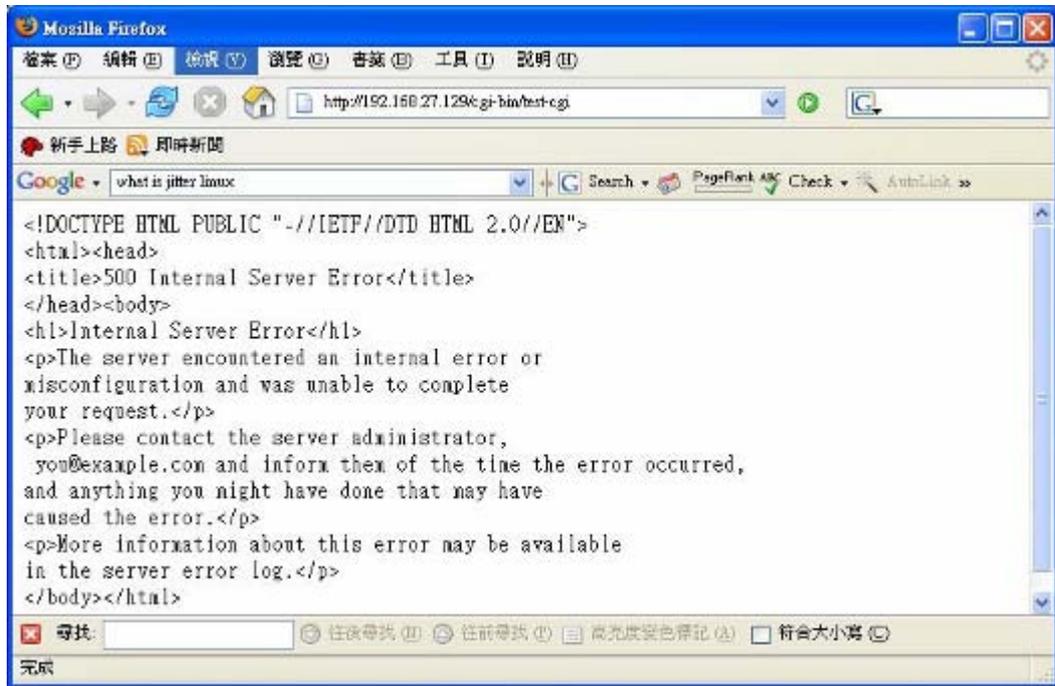
Before you modify the homepage, use a browser (such as Microsoft Internet Explorer or Mozilla Firefox) from your PC to test if the Apache Web Server is working. Type the LAN1 IP address in the browser's address box to open the homepage. E.g., if the default IP address is still active, type `http://host-ip-address`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



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



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



#### ATTENTION

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

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

## IPTABLES

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

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

The UC-8410 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.

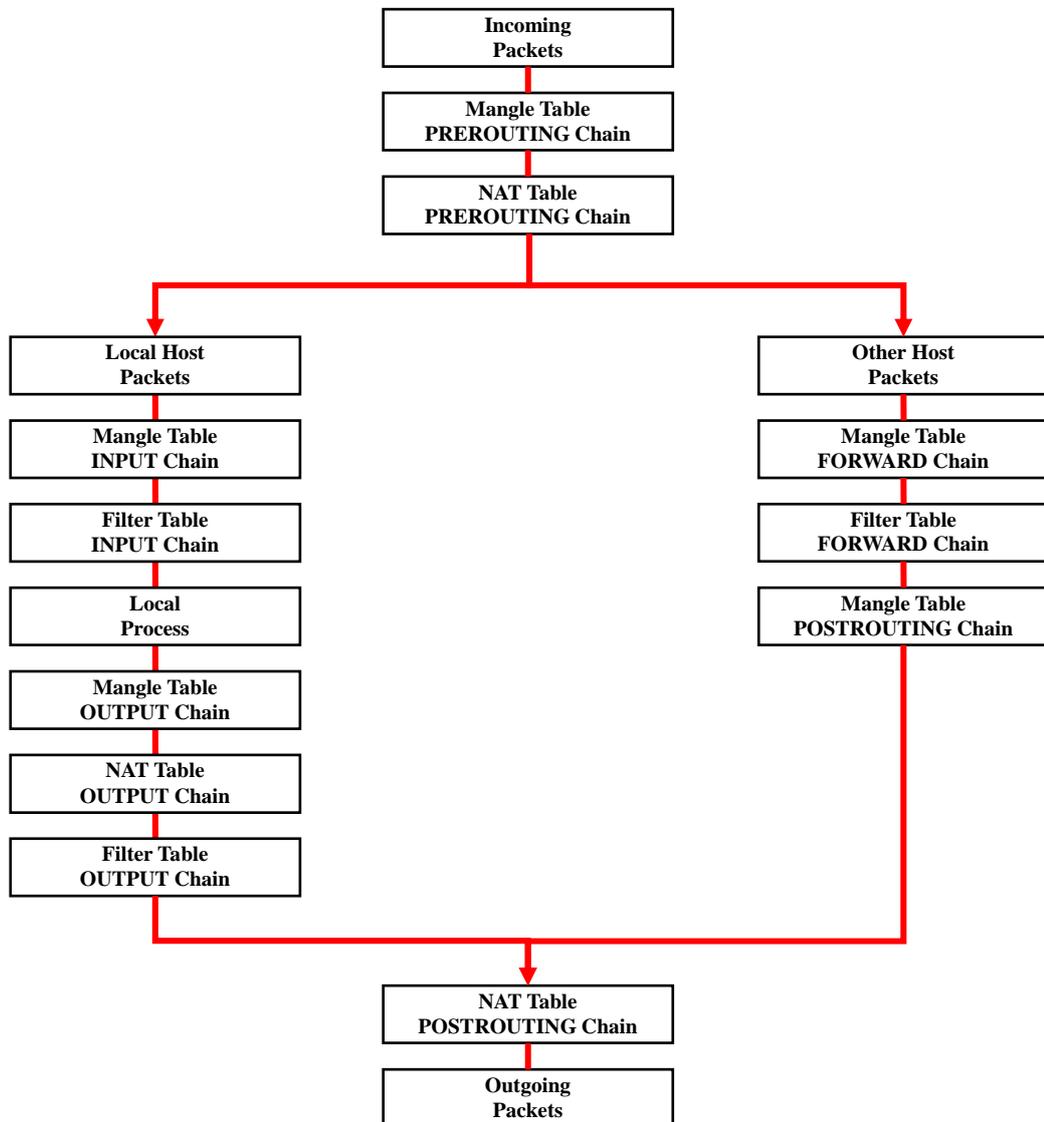
REDIRECT—a special form of DNAT that re-sends packets to a local host independent of the destination IP address.

**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.



The UC-8410 supports the following sub-modules. Be sure to use the module that matches your application.

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

**NOTE** The UC-8410 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-8410. 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
```

Use **iptables**, **iptables-restore**, **iptables-save** to maintain the database.

**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-8410.
OUTPUT:      For locally-generated packets.
FORWARD:     For packets routed out through the UC-8410.
PREROUTING: To alter packets as soon as they come in.
POSTROUTING: To alter packets as they are about to be sent out.
```

### Examples:

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

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

**Append or delete rules:****Usage:**

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

**Examples:**

Example 1: Accept all packets from lo interface.

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

Example 2: Accept TCP packets from 192.168.0.1.

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

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

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

Example 4: Drop TCP packets from 192.168.1.25.

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

Example 5: Drop TCP packets addressed for port 21.

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

Example 6: Accept TCP packets from 192.168.0.24 to UC-8410's port 137, 138, 139

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

Example 7: Log TCP packets that visit the UC-8410's port 25.

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

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

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

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

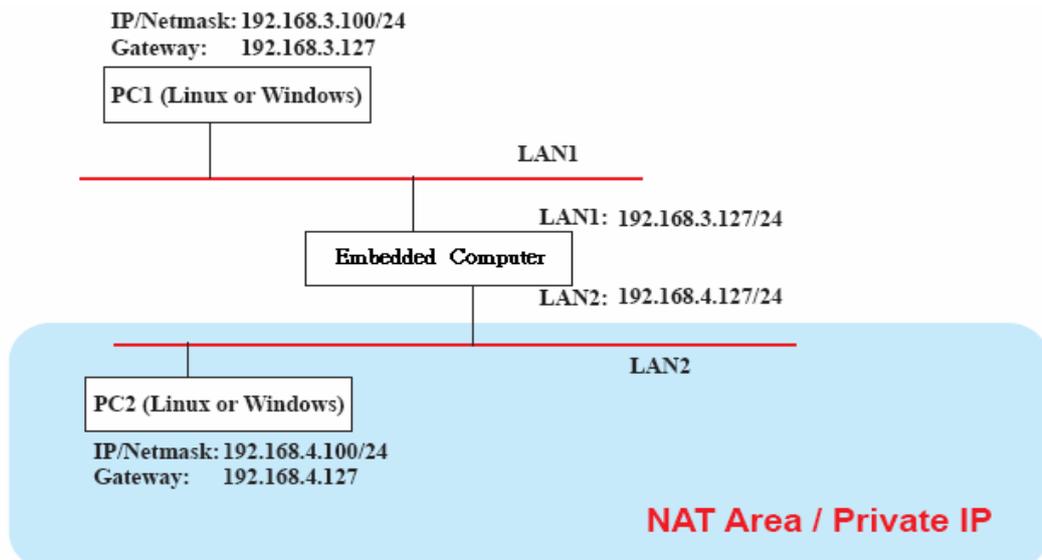
## NAT

NAT (Network Address Translation) protocol translates IP addresses used on one network to different IP addresses used on another network. One network is designated the inside network and the other is the outside network. Typically, the UC-8410 connects several devices on a network and maps local inside network addresses to one or more global outside IP addresses, and un-maps the global IP addresses on incoming packets back into local IP addresses.

**NOTE** Click the following link for more information about iptables and NAT:  
<http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html>

## NAT Example

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



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

## Enabling NAT at Bootup

In most real world situations, you will want to use a simple shell script to enable NAT when the UC-8410 boots up. The following script is an example.

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

## Dial-up Service—PPP

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

Modem/PPP access is almost identical to connecting directly to a network through the UC-8410's Ethernet port. Since PPP is a peer-to-peer system, the UC-8410 can also use PPP to link two networks (or a local network to the Internet) to create a Wide Area Network (WAN).

NOTE Click on the following links for more information about ppp:  
<http://tldp.org/HOWTO/PPP-HOWTO/index.html>  
<http://axion.physics.ubc.ca/ppp-linux.html>

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

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

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

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

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

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

The pppd options are described below:

**connect 'chat etc...'**

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

**-v**

verbose mode; log what we do to syslog

**" "**

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

**ATDT5551212**

Dial the modem, and then ...

**CONNECT**

Wait for an answer.

**" "**

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

**ogin: username word: password**

Log in with username and password.

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

**/dev/**

Specify the callout serial port.

**115200**

The baudrate.

**Debug**

Log status in syslog.

**Crtscts**

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

**Modem**

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

**Defaultroute**

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

**192.1.1.17**

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

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

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

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

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

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

**How to check the connection**

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

First, type:

```
/sbin/ifconfig
```

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

```
lo                Link encap Local Loopback
                  inet addr 127.0.0.1  Bcast 127.255.255.255  Mask 255.0.0.0
                  UP LOOPBACK RUNNING  MTU 2000    Metric 1
                  RX packets 0 errors 0 dropped 0 overrun 0

ppp0              Link encap Point-to-Point Protocol
                  inet addr 192.76.32.3  P-t-P 129.67.1.165  Mask 255.255.255.0
                  UP POINTOPOINT RUNNING  MTU 1500    Metric 1
                  RX packets 33 errors 0 dropped 0 overrun 0
                  TX packets 42 errors 0 dropped 0 overrun 0
```

Now, type:

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

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

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

Try typing:

```
netstat -nr
```

You should see three routes, similar to the following:

Kernel routing table

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

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

## Setting up a Machine for Incoming PPP Connections

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

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

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

```
* * "" *
```

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

The following example does not check the username and password:

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

## PPPoE

1. Connect the UC-8410's LAN port to an ADSL modem with a cross-over cable, HUB, or switch.
2. Log in to the UC-8410 as the root user.
3. Edit the file `/etc/ppp/chap-secrets` and add the following:

```
"username@hinet.net" * "password" *
```

```
192.168.3.127 - PuTTY
# Secrets for authentication using CHAP
# client      server secret          IP addresses
#
# PPPOE example, if you want to use it, you need to unmark it and modify it
"username@hinet.net" * "password" *
```

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

4. Edit the file `/etc/ppp/pap-secrets` and add the following:

```
"username@hinet.net" * "password" *
```

```
192.168.3.127 - PuTTY
support hostname      "*" -
stats hostname       "*" -

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

# INBOUND connections

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

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

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

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

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

```
plugin pppoe
```

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

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

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

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

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

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

7. Set up DNS.

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

```
nameserver ip_addr_of_first_dns_server
nameserver ip_addr_of_second_dns_server
```

For example:

```
nameserver 168.95.1.1
nameserver 139.175.10.20
```

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

```
pppd eth0
```

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

```
pppd eth1
```

9. Type `ifconfig ppp0` to check if the connection is OK or has failed. If the connection is OK, you will see information about the ppp0 setting for the IP address. Use ping to test the IP.
10. If you want to disconnect it, use the kill command to kill the pppd process.

## NFS (Network File System) Client

The Network File System (NFS) is used to mount a disk partition on a remote machine, as if it were on a local hard drive, allowing fast, seamless sharing of files across a network. NFS allows users to develop applications for the UC-8410, without worrying about the amount of disk space that will be available. The UC-8410 supports NFS protocol for client.

NOTE Click on the following links for more information about NFS:  
<http://www.tldp.org/HOWTO/NFS-HOWTO/index.html>  
<http://nfs.sourceforge.net/nfs-howto/client.html>  
<http://nfs.sourceforge.net/nfs-howto/server.html>

## Setting up the UC-8410 as an NFS Client

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

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

Steps 1:

```
#mkdir -p /home/nfs/public
```

Step 2:

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

Example

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

## Mail

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

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

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

**Example:**

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

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

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

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

## SNMP

The UC-8410 has the SNMP V1 (Simple Network Management Protocol) agent software built in. It supports RFC1317 RS-232 like groups and RFC 1213 MIB-II.

The following simple example allows you to use an SNMP browser on the host site to query the UC-8410, which is the SNMP agent. The UC-8410 will respond.

```
debian:~# snmpwalk -v 1 -c public -Cc 192.168.30.127
SNMPv2-MIB::sysDescr.0 = STRING: Linux version 2.6.23.1 (root@UC8400) (gcc version 4.2.1) #1137 Wed Sep 17
16:17:45 EDT 2008
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.8691.12.8410
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (239600) 0:39:56.00
SNMPv2-MIB::sysContact.0 = STRING: Moxa Systems Co., LDT.
SNMPv2-MIB::sysName.0 = STRING: (none)
SNMPv2-MIB::sysLocation.0 = STRING: Unknown
SNMPv2-MIB::sysServices.0 = INTEGER: 6
IF-MIB::ifNumber.0 = INTEGER: 11
IF-MIB::ifIndex.1 = INTEGER: 1
IF-MIB::ifIndex.2 = INTEGER: 2
IF-MIB::ifIndex.3 = INTEGER: 3
IF-MIB::ifIndex.4 = INTEGER: 4
IF-MIB::ifIndex.5 = INTEGER: 5
IF-MIB::ifIndex.6 = INTEGER: 6
IF-MIB::ifIndex.7 = INTEGER: 7
IF-MIB::ifIndex.8 = INTEGER: 8
IF-MIB::ifIndex.9 = INTEGER: 9
IF-MIB::ifIndex.10 = INTEGER: 10
IF-MIB::ifIndex.11 = INTEGER: 11
IF-MIB::ifDescr.1 = STRING: eth0
IF-MIB::ifDescr.2 = STRING: eth1
IF-MIB::ifDescr.3 = STRING: eth2***** SNMP QUERY FINISHED *****
```

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

→ The UC-8410 does NOT support SNMP trap.

The following tables list the variables supported by the UC-8410.

## OpenVPN

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

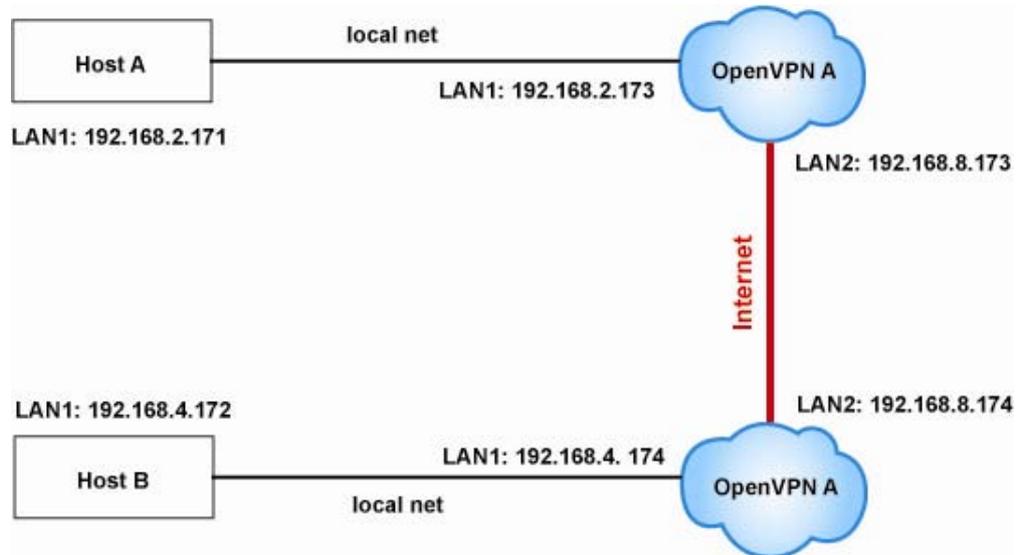
```
# mknod /dev/net/tun c 10 200
```

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

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

## Setup 1: Ethernet Bridging for Private Networks on Different Subnets

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



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

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

Copy the file that is generated to the OpenVPN machine.

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

```
#-----Start-----
#!/bin/sh

iface=eth1      # defines the internal interface
maxtap=`expr 1` # defines the number of tap devices. I.e., # of tunnels

IPADDR=
NETMASK=
BROADCAST=

# it is not a great idea but this system doesn't support
# /etc/sysconfig/network-scripts/ifcfg-eth1
ifcfg_vpn()
{
    while read f1 f2 f3 f4 r3
    do
        if [ "$f1" = "iface" -a "$f2" = "$iface" -a "$f3" = "inet" -a "$f4" =
"static" ];then
            i=`expr 0`
            while :
            do
                if [ $i -gt 5 ]; then
                    break
                fi
                i=`expr $i + 1`
                read f1 f2
            done
        fi
    done
}
```

```

        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}
    done
}

```

```

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

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

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

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

    ipf=/proc/sys/net/ipv4/ip_forward
    # enable IP forwarding
    echo 1 > $ipf
    echo "ip forwarding enabled to"
    cat $ipf
}

stop() {
    echo "shutdown openvpn bridge."
    ifcfg_vpn
    i=`expr 0`
    while :
    do
        # disconnect tap device from the bridge
        brctl delif br0 tap${i}
        openvpn --rmtun --dev tap${i}

        i=`expr $i + 1`
        if [ $i -ge $maxtap ]; then
            break
        fi
    done
    brctl delif br0 $iface
    brctl delbr br0
    ifconfig br0 down
    ifconfig $iface $IPADDR netmask $NETMASK broadcast $BROADCAST
    killall -TERM openvpn
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    *)
        echo "Usage: $0 [start|stop|restart]"
        exit 1
esac
exit 0
#----- end -----

```

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

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

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

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

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

```
# /etc/openvpn/tap0-br.conf
# point to the peer
remote 192.168.8.173
dev tap0
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
up /etc/openvpn/tap0-br.sh
```

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

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

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

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

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

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

Next, start both OpenVPN peers,

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

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

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

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

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

```
# route
```

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

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

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

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

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

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

Now ping Host B from Host A by typing:

```
ping 192.168.4.174
```

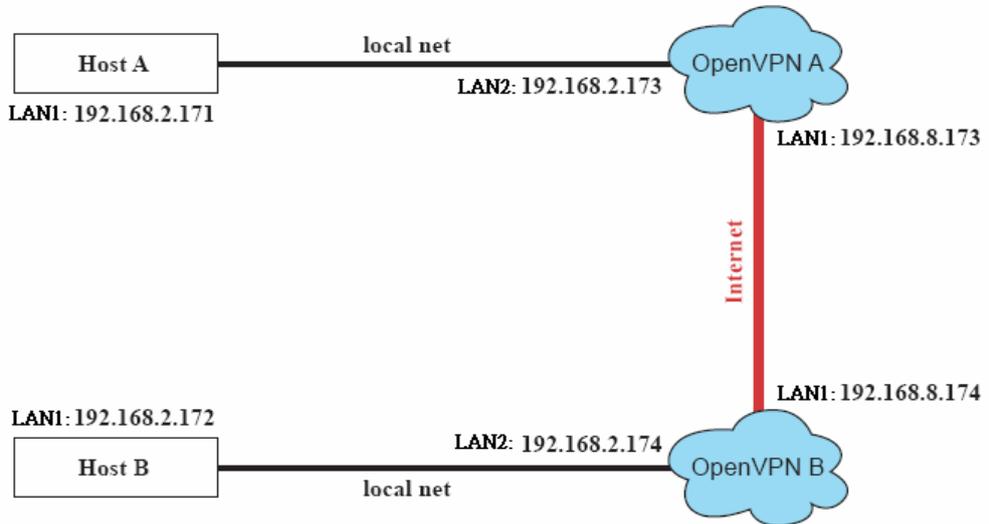
A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 5000 between OpenVPN peers.

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

```
# /etc/openvpn/openvpn-bridge stop
```

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

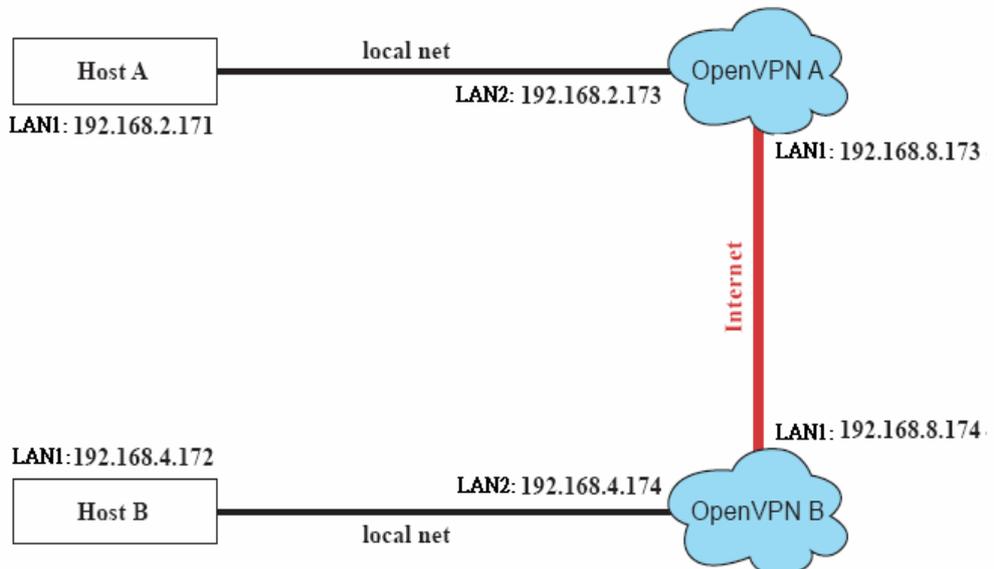
1. Set up four machines as shown in the following diagram:



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

### Setup 3: Routed IP

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



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

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

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

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

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

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

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

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

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

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

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

```
# route
```

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

## Package Management—ipkg

ipkg is a very lightweight package management system. It also allows for dynamic installation/removal of packages on a running system. Because the disk space is limited, we provide the software as extension packages. You can use ipkg-cl to install or remove .ipk packages on the UC-8410-LX.

### Install an .ipk package via an .ipk file

Upload the .ipk package to the Moxa embedded computer:

```
192.168.3.127 - Putty
Moxa:~# scp /mnt/cdrom/utility_tools/ipkg_packages/libphp5_1.0_xscale.ipk
192.168.3.127:/tmp/
```

Install the uploaded package:

```
192.168.3.127 - Putty
Moxa:~# ipkg-cl list-install /tmp/libphp5_1.0_xscale.ipk
```

### List the installed packages

```
192.168.3.127 - Putty
Moxa:~# ipkg-cl list-installed
```

### Remove a package

```
192.168.3.127 - Putty
Moxa:~# ipkg-cl remove libphp5
```

# 5

## Programmer's Guide

---

This chapter includes important information for programmers.

The following functions are covered:

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

## Flash Memory Map

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

Address	Size	Contents
0x00000000 – 0x0005FFFF	640 KB	Boot Loader—Read ONLY
0x00060000 – 0x001FFFFFFF	1.875 MB	Kernel object code—Read ONLY
0x00200000 – 0x00DFFFFFFF	13.375 MB	Root file system (JFFS2) —Read ONLY
0x00E00000 – 0x01FCFFFF	32 MB	User root file system (JFFS2) —Read/Write
0x01FC0000 – 0x01FDFFFF	128 KB	Boot Loader configuration and directory—Read ONLY
	256 KB	SRAM—Read./Write

- 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-8410.

## Linux Tool Chain Introduction

To ensure that an application will be able to run correctly when installed on the UC-8410, you must ensure that it is compiled and linked to the same libraries that will be present on the UC-8410. This is particularly true when the RISC Xscale processor architecture of the UC-8410 differs from the CISC x86 processor architecture of the host system, but it is also true if the processor architecture is the same.

The host tool chain that comes with the UC-8410 contains a suite of cross compilers and other tools, as well as the libraries and headers that are necessary to compile applications for the UC-8410. The host environment must be running Linux to install the UC-8410 GNU Tool Chain. We have confirmed that the following Linux distributions can be used to install the tool chain:

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

The Tool Chain will need about 836 MB of hard disk space on your PC. The UC-8410 Tool Chain is located on the UC-8410 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount /dev/cdrom /mnt/cdrom
#sh /mnt/cdrom/tool-chain/linux/arm-linux_2.0.sh
```

Wait for a few minutes while the Tool Chain is installed automatically on your Linux PC. Once the host environment has been installed, add the directory `/opt/montavista/pro/devkit/arm/xscale_be/bin/` to your path and the directory `/opt/montavista/pro/devkit/arm/xscale_be/man/` to your manual path. You can do this temporarily for the current login session by issuing the following commands:

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

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

## Obtaining help

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

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

## Cross Compiling Applications and Libraries

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

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

## Tools Available in the Host Environment

Most of the cross compiler tools are the same as their native compiler counterparts, but with an additional prefix that specifies the target system. In the case of x86 environments, the prefix is **i386-linux-** and in the case of the UC-8410 Xscale boards, it is **xscale-linux-**.

For example, the native C compiler is **gcc** and the cross C compiler for Xscale in the UC-8410 is **xscale-linux-gcc**.

The following cross compiler tools are provided:

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

## Debugging with GDB

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

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

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

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

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

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

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

3. Enter the following command:

```
#ddd --debugger xscale-linux-gdb hello-debug &
```

4. Enter the following command at the GDB, DDD command prompt:

```
Target remote 192.168.4.99:2000
```

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

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

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

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

## Device API

The UC-8410 supports control devices with the **ioctl** system API. You will need to include `<moxadevice.h>`, and use the following **ioctl** function.

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

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

```
#man ioctl
```

## RTC (Real Time Clock)

The device node is located at `/dev/rtc`. The UC-8410 supports Linux standard simple RTC control. You must include `<linux/rtc.h>`.

1. Function: `RTC_RD_TIME`

```
int ioctl(fd, RTC_RD_TIME, struct rtc_time *time);
```

Description: read time information from the RTC. It will return the value on argument 3.

2. Function: `RTC_SET_TIME`

```
int ioctl(fd, RTC_SET_TIME, struct rtc_time *time);
```

Description: set RTC time. Argument 3 will be passed to the RTC.

## Buzzer

The device node is located at `/dev/console`. The UC-8410 supports Linux standard buzzer control, with the UC-8410's buzzer running at a fixed frequency of 100 Hz. You must include `<sys/kd.h>`.

1. Function: KDMKTONE

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

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

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

## WDT (Watch Dog Timer)

1. Introduction

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

2. How the WDT works

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

Kernel boot

```
....
....
```

User application running and enable user ack

```
....
....
```

3. The user API

The user application must use `include <moxadef.h>`, and `link libmoxalib.a`. A makefile example is shown below:

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

```
int swtd_open(void)
```

### Description

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

### Input

None

### Output

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

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

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

### Description

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

**Input**

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

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

**Output**

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

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

**Description:**

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

**Input :**

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

**Output:**

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

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

**Description:**

Get current setting values.

mode –

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

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

time – The time period to ack sWatchDog.

**Input :**

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

int \*mode - the function will return the status: enable or disable.

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

**Output:**

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

```
int swtd_ack(int fd)
```

**Description:**

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

**Input :**

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

**Output:**

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

```
int swtd_close(int fd)
```

**Description:**

Close the file handle.

**Input :**

int fd - the file handle from swtd\_open() return value.

**Output:**

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

**4. Special Note**

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

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

**5. User application example****Example 1:**

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

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

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

The makefile is shown below:

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

**Example 2:**

```

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

static void mydelay(unsigned long msec)
{
    struct timeval time;

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

static int    swtdfd;
static int    stopflag=0;

static void stop_swatcdog()
{
    stopflag = 1;
}

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

```

The makefile is shown below:

```

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

```

## Digital I/O

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

### Application Programming Interface

Return error code definitions:

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

int state - to set high or low state; DIO\_HIGH (1) for high, DIO\_LOW (0) for low.

Output: none.

Return: reference the error code.

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

Description: To get the DIN port state.

Input: int diport - get the current state of which DIN port. Port numbering is from 0 to 3.

int \*state - save the current state.

Output: state - DIO\_HIGH (1) for high, DIO\_LOW (0) for low.

Return: reference the error code.

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

Description: To get the DOUT port state.

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

int \*state - save the current state.

Output: state - DIO\_HIGH (1) for high, DIO\_LOW (0) for low.

Return: reference the error code.

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

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

Input: int diport - the port that will be used to detect the DIN event.

Port numbering is from 0 to 3.

void (\*func) (int diport) - Not NULL

> Returns the call back function. When the event occurs, the call back function will be invoked.

NULL

> Clears this event

```
int mode DIN_EVENT_HIGH_TO_LOW
```

(1): from high to low

```
DIN_EVENT_LOW_TO_HIGH
```

(0): from low to high

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

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

Description: To retrieve the DIN event configuration, including mode  
 (DIN\_EVENT\_HIGH\_TO\_LOW or DIN\_EVENT\_LOW\_TO\_HIGH), and the value of  
 "duration."  
 Input: **int diport** - which DIN port you want to retrieve.  
 - The port whose din event setting we wish to retrieve  
**int \*mode** - save which event is set.  
**unsigned long \*duration** - the duration of the DIN port is kept in high or low state.  
 - return to the current duration value of diport  
 Output: **mode** DIN\_EVENT\_HIGH\_TO\_LOW  
 (1): from high to low  
 DIN\_EVENT\_LOW\_TO\_HIGH(0): from low to high  
 DIN\_EVENT\_CLEAR(-1): clear this event  
**duration** The value of duration should be 0 or 40 <= duration  
 <= 3600000 milliseconds.  
 Return: reference the error code.

## Special Note

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

## Examples

### Example 1

File Name: tdio.c

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

```
#include <stdio.h>
#include <stdlib.h>
#include <moxadevice.h>
#include <fcntl.h>
#ifdef DEBUG
#define dbg_printf(x...) printf(x)
#else
#define dbg_printf(x...)
#endif
#define MIN_DURATION 40
```

```

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

```

```

printf("Event:\n");
for ( j=0; j<MAX_DOUT_PORT; j++ ) {
retval=get_din_event(j, &i, &duration);
switch ( i ) {
case DIN_EVENT_HIGH_TO_LOW :
printf("(htl,%lu)", duration);
break;
case DIN_EVENT_LOW_TO_HIGH :
printf("(lth,%lu)", duration);
break;
case DIN_EVENT_CLEAR :
printf("(clr,%lu)", duration);
break;
default :
printf("err ");
break;
}
}
printf("\n");
} else {
printf("Select error, please select again !\n");
}
switch(retval) {
case DIO_ERROR_PORT:
printf("DIO error port\n");
break;
case DIO_ERROR_MODE:
printf("DIO error mode\n");
break;
case DIO_ERROR_CONTROL:
printf("DIO error control\n");
break;
case DIO_ERROR_DURATION:
printf("DIO error duratoin\n");
case DIO_ERROR_DURATION_20MS:
printf("DIO error! The duratoin is not a multiple of 20 ms\n");
break;
}
}
return 0;
}

```

## DIO Program Make File Example

```

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

```

## UART

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

The UC-8410 supports Linux standard termios control. The Moxa UART Device API allows you to configure ttyM0 to ttyM7 as RS-232, RS-422, 4-wire RS-485, or 2-wire RS-485. The UC-8410 supports RS-232, RS-422, 2-wire RS-485, and 4-wire RS485.

You must include `<moxadevice.h>`.

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

1. Function: MOXA\_SET\_OP\_MODE

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

### Description

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

2. Function: MOXA\_GET\_OP\_MODE

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

### Description

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

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

Function: MOXA\_SET\_SPECIAL\_BAUD\_RATE

Function: MOXA\_GET\_SPECIAL\_BAUD\_RATE

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

## Setinterface

The Serial Port Expansion Module has 8 serial ports, labeled ttyM0 to ttyM7. The ports support RS-232, RS-422, and RS-485 2-wire and 4-wire operation modes with baudrate settings up to 921600 bps.

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

Usage: `setinterface device-node [interface-no]`

device-node -- /dev/ttyMn; n = 0,1,2,...

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

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

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

```
DA682:~# setinterface /dev/ttyM0 2
DA682:~# setinterface /dev/ttyM0
Now setting is RS422 interface.
DA682:~#
```

### Example for setting the baudrate

```
#include <moxadevice.h>
#include <termios.h>
struct termios term;
int fd, speed;
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
term.c_cflag &= ~(CBAUD | CBAUDEX);
term.c_cflag |= B400000;
tcsetattr(fd, TCSANOW, &term);
speed = 500000;
ioctl(fd, MOXA_SET_SPECIAL_BAUD_RATE, &speed);
```

### Example for getting the baudrate

```
#include <moxadevice.h>
#include <termios.h>
struct termios term;
int fd, speed;
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
if ( (term.c_cflag & (CBAUD|CBAUDEX)) != B400000 )
{ // follow the standard termios baud rate define } else
{ ioctl(fd, MOXA_GET_SPECIAL_BAUD_RATE, &speed); }
```

### Baudrate inaccuracy

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

\*The Inaccuracy should less than 2% for the device to work reliably.

## Special Note

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

## SRAM

### 1. Introduction

The UC-8410 provides 256 KB of embedded SRAM. As there is a system battery inside the computer, the SRAM can work and be used to keep data even when the system is crashed. This means that the data stored on the SRAM will not be lost after the UC-8410 is powered off.

### 2. How the SRAM works

The SRAM device can be programmed through the file `/dev/sram`. This means that you can read from or write to `/dev/sram` to store data on the embedded SRAM. The following example illustrates how to do this:

```

/*****
History :
    Versoin      Author      Date      Comment
    1.0          Jared Wu.   09-11-2008 Write a pattern to SRAM.
*****/
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <linux/kd.h>

#define SRAM_SIZE    0x00040000 // 256 Kbytes
static char sram_buf1[SRAM_SIZE], sram_buf2[SRAM_SIZE];

int main(int argc, char * argv[])
{
    int      fd, len;
    unsigned long  ms=0;
    char pattern='9';

    if ( argc > 2 ) {
        printf("Usage: %s [pattern]\n");
        exit(0);
    }

    if ( argc == 2 ) {
        pattern=argv[1][0];
        printf("pattern:%c\n", pattern);
    }

    fd = open("/dev/sram", O_RDWR);

```

```

if( fd < 0 ) {
    printf("Open /dev/sram fail");
    exit(0);
}

// Write the sram with patern
memset(sram_buf1, pattern, sizeof(sram_buf1));
len=write(fd, sram_buf1, sizeof(sram_buf1));
if( len < 0 ) {
    printf("Write /dev/sram fail");
    exit(0);
}

printf("The content is written\n");

close(fd);
}
/*****
History :
    Versoin      Author      Date      Comment
    1.0          Jared Wu.  09-11-2008 Read from the SRAM and compare with some pattern.
*****/
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <linux/kd.h>

#define SRAM_SIZE    0x00040000 // 256 Kbytes
static char sram_buf1[SRAM_SIZE], sram_buf2[SRAM_SIZE];

int main(int argc, char * argv[])
{
    int      fd, len;
    unsigned long  ms=0;
    char pattern='9';

    if ( argc > 2 ) {
        printf("Usage: %s [pattern]\n");
        exit(0);
    }

    if ( argc == 2 ) {
        pattern=argv[1][0];
        printf("pattern:%c\n", pattern);
    }

    fd = open("/dev/sram", O_RDWR);
    if( fd < 0 ) {
        printf("Open /dev/sram fail");
        exit(0);
    }

```

```
// Write the sram with pattern
memset(sram_buf1, pattern, sizeof(sram_buf1));

// Read from sram and compare with pattern
len=read(fd, sram_buf2, sizeof(sram_buf2));
if( len < 0 ) {
    printf("Read from /dev/sram fail\n");
    exit(0);
}

if ( memcmp(sram_buf1, sram_buf2, SRAM_SIZE) != 0 ) {
    printf("Memory compared fail\n");
    exit(0);
}

printf("The content is identical\n");

close(fd);
}
```

## Make File Example

The following Makefile file sample code is copied from the Hello example on the UC-8410's CD-ROM.

```
CC = xscale-linux-gcc
CPP = xscale-linux-gcc
SOURCES = hello.c

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

all:    hello

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

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

## Software Lock

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

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

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

```
#setkey ABigKey
```

NOTE: If you would like to clear the encryption key on the target computer, use the following command.

```
#setkey ""
```

2. Develop and compile your program on the development PC.
3. On the development PC, run the utility program 'binencryptor' to encrypt your program with an encryption key.

```
#binencryptor yourProgram ABigKey
```

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

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

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

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

# A

## System Commands

---

### Busybox (V1.10.4): Linux normal command utility collection

#### File manager

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

## Editor

<b>vi</b>	text editor
<b>cat</b>	dump file context
<b>grep</b>	Search string on file
<b>egrep</b>	search string on file of Extended regular expressions
<b>fgrep</b>	Search file(s) for lines that match a fixed string
<b>cut</b>	Get string on file
<b>find</b>	Find file where are there
<b>more</b>	dump file by one page
<b>test</b>	test if file exists or not
<b>sleep</b>	Sleep (seconds)
<b>usleep</b>	suspend execution for microsecond intervals
<b>echo</b>	echo string
<b>sed</b>	Stream editor
<b>awk</b>	pattern-directed scanning and processing language
<b>expand</b>	Converts all tabs to spaces
<b>tail</b>	Print the last 10 lines of each FILE to standard output.
<b>tar</b>	The GNU version of the tar archiving utility
<b>tr</b>	Translate, squeeze, and/or delete characters
<b>wc</b>	Print byte, word, and line counts, count the number of bytes, whitespace-separated words, and newlines in each given FILE, or standard input if non are given or for a FILE of '-'. .

## Network

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

## Process

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

## Modules

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

## Other

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

## Special Moxa Utilities

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