UC-7101/7110/7112 User's Manual

Eighth Edition, February 2009

www.moxa.com/product

UC-7101/7110/7112 User's Manual

Technical Support Contact Information
www.moxa.com/support

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

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

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

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

Table of Contents

1 Introduction

The Moxa UC-7101/7110/7112 Series of mini RISC-based ready-to-run embedded computers are ideal for your embedded applications. The UC-7110 and UC-7112 feature dual 10/100 Mbps Ethernet ports and two RS-232/422/485 serial ports in a built-in µClinux ARM9 box. The UC-7101 features one 10/100 Mbps Ethernet port and one RS-232/422/425 serial port. In addition, the UC-7101 and UC-7112 provide an internal SD socket for storage expansion, offer high performance communication and unlimited storage in a super compact, palm-size box. The UC-7101/7110/7112 embedded computers are the right solutions for embedded applications that use a lot of memory, but that must be housed in a small physical space without sacrificing performance.

This chapter covers the following topics:

- ❑ **Overview**
- ❑ **Package Checklist**
- ❑ **Product Features**
- ❑ **Product Specifications**
  - ➢ Hardware Specifications
  - ➢ Software Specifications

Overview

The UC-7101/7110/7112 Series of mini RISC-based communication platforms are ideal for your embedded applications. The UC-7101/7110/7112 come with RS-232/422/485 serial ports and 10/100 Mbps Ethernet LAN ports to provide users with a versatile communication platform.

The UC-7101/7110/7112 use the ARM9 RISC CPU. Unlike the X86 CPU, which uses a CISC design, the ARM9's RISC design architecture and modern semiconductor technology provide the UC-7101/7110/7112 with a powerful computing engine and communication functions, but without generating too much heat. The built-in 8 MB NOR Flash ROM and 16 MB SDRAM give you enough storage capacity, and an additional SD socket provides you with flexible storage expansion to run a wide range of applications. The LAN ports built into the ARM9 make the UC-7101/7110/7112 ideal communication platforms for data acquisition and protocol conversion applications, and the RS-232/422/485 serial ports allow you to connect a variety of serial devices.

The pre-installed µClinux operating system provides an open software operating system for software program development. Software written for desktop PCs is easily ported to the UC-7101/7110/7112 with a GNU cross complier, so that you will not need to spend time modifying existing software code. The operating system, device drivers, and your own software can all be stored in the UC-7101/7110/7112 Flash memory.

Package Checklist

The following models of the UC-7101/7110/7112 Series are currently available:

UC-7101-LX

Mini RISC-based Ready-to-Run Embedded Computer with 1 Serial Port, 1 Ethernet port, SD, and µClinux operating system

UC-7110-LX

Mini RISC-based Ready-to-Run Embedded Computer with 2 Serial Ports, Dual Ethernet, µClinux OS

UC-7112-LX

Mini RISC-based Ready-to-Run Embedded Computer with 2 Serial Ports, Dual Ethernet, SD, µClinux OS

The UC-7101/7110/7112 embedded computers are shipped with the following items:

- 1 UC-7101/7110/7112
- UC-7101/7110/7112 Quick Installation Guide
- Document and Software CD
- Ethernet cross-over cable: RJ45 to RJ45, 100 cm
- Console port cable: CBL-4PINDB9F-100 (4-pin header to female DB9 cable, 100 cm)
- Universal Power Adaptor
- Product Warranty Statement

Optional Accessories

- DK-35A DIN-Rail Mounting Kit (35 mm)

NOTE*: Please notify your sales representative if any of the above items are missing or damaged.*

Product Features

The UC-7101/7110/7112 embedded computers have the following features:

- Mini controller with ready-to-run platform for customized applications
- 32-bit ARM9 RISC microcontroller
- On-board 16 MB RAM, 8 MB Flash ROM
- Two RS-232/422/485 serial ports (one RS-232/422/485 serieal port for UC-7101)
- Dual 10/100 Mbps Ethernet (one 10/100 Mbps ethernet for UC-7101)
- SD expansion slot for storage expansion (UC-7101/7112 only)
- µClinux-ready communication platform
- Wall mounting installation
- Robust fanless design

Product Specifications

Hardware Specifications

CPU	Moxa ARM9-based 32-bit RISC CPU, 192 MHz
RAM	16 MB (12 MB of user programmable space)
Flash	8 MB (4 MB of user programmable space)
Storage Expansion	Internal SD socket x 1 for SD memory card (UC-7101/7112)
LAN	Auto-sensing 10/100 Mbps x 2
LAN Protection	Built-in 1.5 KV magnetic isolation
Serial Ports	RS-232/422/485 ports support: RS-232 signals: TxD, RxD, DTR, DSR, RTS, CTS, DCD, GND RS-422 signals: TxD+, TxD-, RxD+, RxD-, GND 4-wire RS-485 signals: TxD+, TxD-, RxD+, RxD-, GND 2-wire RS-485 signals: Data+, Data-, GND
Serial Protection	15 KV ESD for all signals
Data bits	5, 6, 7, 8
Stop bit(s)	1, 1.5, 2
Parity	None, Even, Odd, Space, Mark
Flow Control	RTC/CTS, XON/XOFF, RS-485 ADDCTM
Speed	50 bps to 921.6 Kbps; Any Baudrate supported
Watchdog Timer	Yes
Real Time Clock	Yes
Buzzer	Yes
Console Port	3-wire RS-232 (Tx, Rx, GND) (19200, n, 8 , 1)
LEDs	Ready Serial Tx, Rx (2 of each) LAN 10/100 (one on each LAN connector)
Dimensions (WxDxH)	77 x 111 x 26 mm (3.03 x 4.37 x 1.02 in)
Gross Weight	190g
Power input	12-48 VDC
Power Consumption	300 mA @ 12 VDC (UC-7101)

	340 mA @ 12 VDC, 4.5W (UC-7110/7112)
Operating temperature	-10 to 60°C, (14 to 140°F), 5 to 95% RH
Storage temperature	-20 to 80°C, (-4 to 176°F), 5 to 95% RH
Regulatory Approvals	EMC: FCC Class A, CE Class A Safety: UL, CUL, TÜV
Warranty	5 years

Software Specifications

Kernel	µClinux Kernel 2.6 Support for dynamic driver module load / unload
Protocol Stack	ARP, ICMP, IPV4, TCP, UDP, FTP, Telnet, SNMP V1/V2c, HTTP, CHAP, PAP, DHCP, NTP, NFS V2/V3, SMTP, Telnet, FTP, PPP, PPPoE
File System	JFFS2 for Kernel, Root File System (Read Only) and User Directory (Read / Write)
Msh	Minix shell command
pppd	Dial in/out over serial port daemon
PPPoE	Point-to-Point over Ethernet daemon
snmpd	SNMP V1/V2c Agent daemon
busybox	Linux normal command utility
Tinylogin	login and user manager utility
Telnetd	Telnet server daemon
telnet	Telnet client program
inetd	TCP server manager program
ftpd	FTP server program
ftp	FTP client program
boa	Web server daemon
ntpdate	Network Time Protocol client utility
Tool Chain	
Linux Tool Chain	Arm-elf-gcc (V2.95.3): C/C++ PC Cross Compiler uClibc (V0.9.26): POSIX standard C library

2 Getting Started

In this chapter, we explain the basic procedure for getting the UC-7101/7110/7112 connected and ready to use.

This chapter covers the following topics:

Powering on the UC-7101/7110/7112

Connect the SG wire to the Shielded Contact located on the upper left corner of the UC-7101/7110/7112, and then power on the UC-7101/7110/7112 by connecting the power adaptor. It takes about 16 seconds for the system to boot up. Once the system is ready, the Ready LED will light up.

ATTENTION

After connecting the UC-7101/7110/7112 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-7101/7110/7112 to a PC

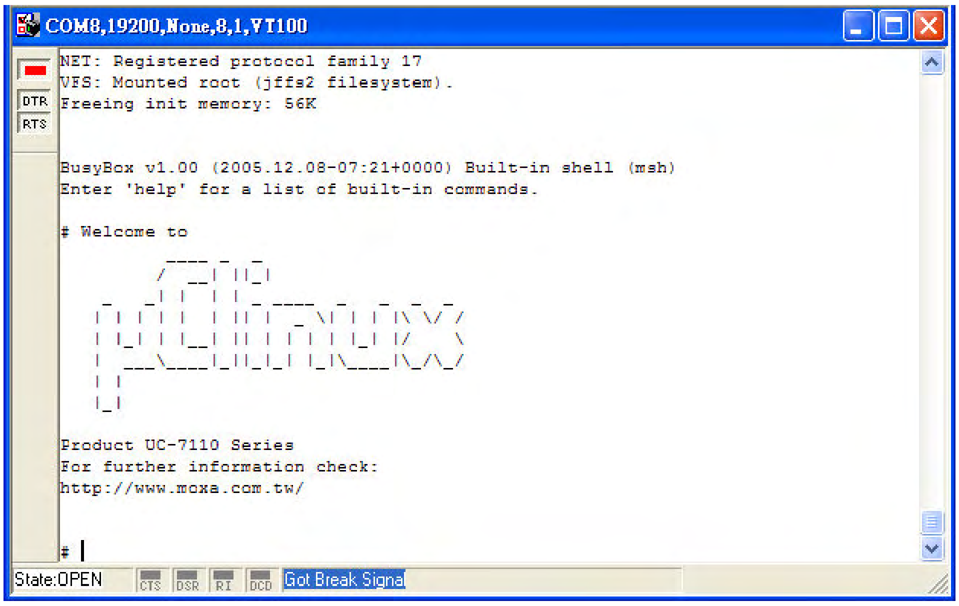
There are two ways to connect the UC-7101/7110/7112 to a PC.

Console Port

The serial console port offers users a convenient means of connecting to the UC-7101/7110/7112. This method is particularly useful when using the UC-7101/7110/7112 for the first time. Since the communication is over a direct serial connection, you do not need to know either of the IP addresses in order to make contact.

Use the serial console port settings shown on the right. Once the connection is established, the window below will open.

Serial Console Port Settings	
Baudrate	19200 bps
Parity	None
Data bits	8
Stop bits	1
Flow Control	None
Terminal	VT100



```
NET: Registered protocol family 17
VFS: Mounted root (jffs2 filesystem).
Freeing init memory: 56K


BusyBox v1.00 (2005.12.08-07:21+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

# Welcome to
          ____ _  _
         /  __| ||_|
    _   _| |  | |  _ ____  _   _  _  _
   | | | | |  | | | |  _ \| | | |\ \/ /
   | |_| | |__| |_| | | | | |_| |/    \
   |  ___\____|_____|_| |_|\____|\_/\_/
   | |
   |_|

Product UC-7110 Series
For further information check:
http://www.moxa.com.tw/


# |
```

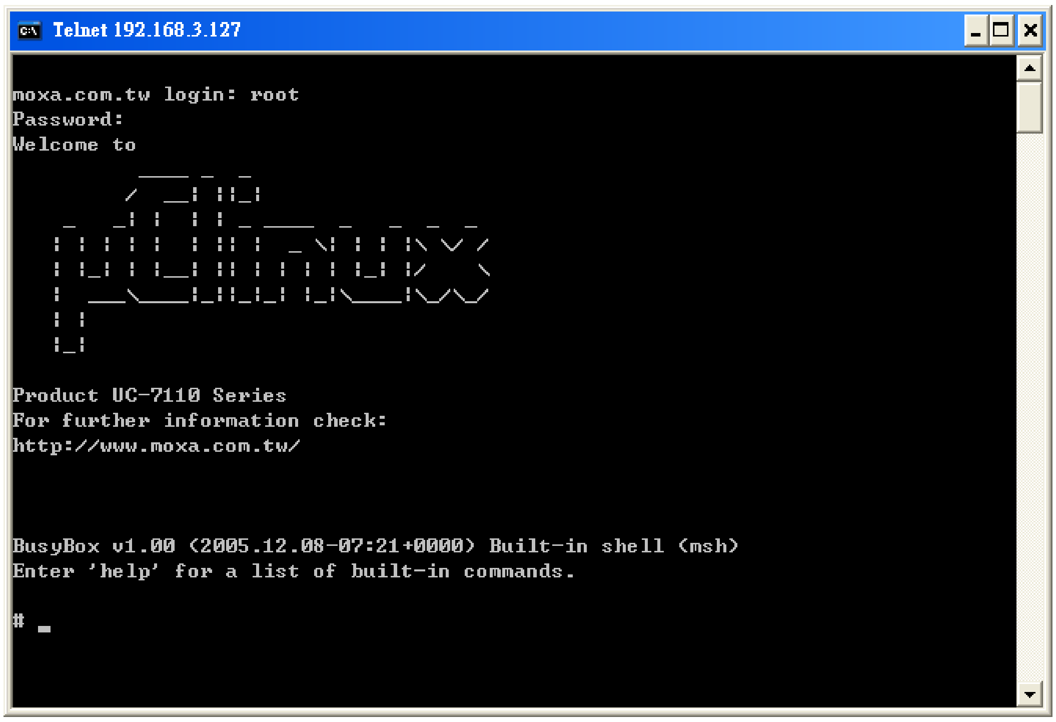
Telnet

If you know at least one of the two IP addresses and netmasks, then you can use Telnet to connect to the UC-7101/7110/7112's console.

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

Telnet can be used locally by using a crossover Ethernet cable to connect your computer to the UC-7101/7110/7112, or over a LAN or the Internet. The default IP addresses and netmasks are shown above. To login, type the Login name and password as requested. The defaults are:

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



```
moxa.com.tw login: root
Password:
Welcome to
          _____ _ _
         /  __| ||_|
    _   _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|

Product UC-7110 Series
For further information check:
http://www.moxa.com.tw/



BusyBox v1.00 (2005.12.08-07:21+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

# _
```

Once you open the "msh command shell" you can proceed to configure the UC-7101/7110/7112's network settings, as described in the next section.

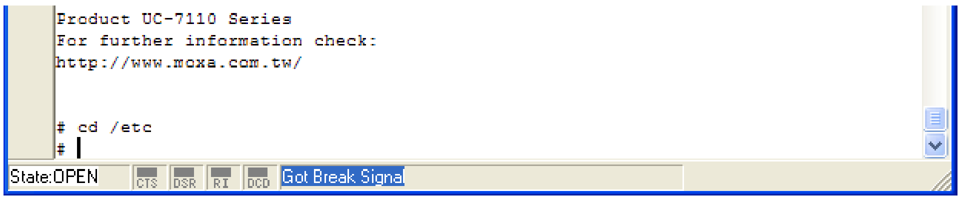
ATTENTION

- **Serial Console Reminder**: Remember to choose VT100 as the terminal type. Use the cable CBL-RJ45F9-150 that comes with the UC-7101/7110/7112 to connect to the serial console port. If you are not able to connect on the first try, unplug and then re-plug the UC-7101/7110/7112's power cord.
- **Telnet Reminder**: When connecting to the UC-7101/7110/7112 over a LAN, configure your PC's Ethernet card to be on the same subnet as the UC-7101/7110/7112 you wish to contact.

Configuring the Ethernet Interface

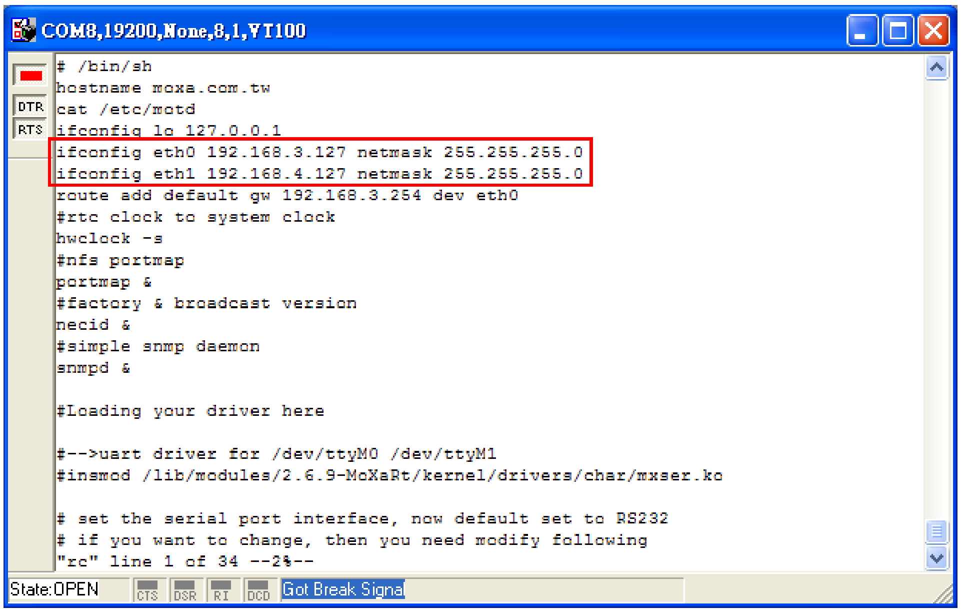
In this section, we use the serial console to explain how to modify the UC-7101/7110/7112's network settings.

1. Change directories by issuing the command `cd /etc`.



```
Product UC-7110 Series
For further information check:
http://www.moxa.com.tw/

# cd /etc
# 
```

2. Type the command `vi rc` to use VI Editor to edit the configuration file. The IP addresses for the UC-7101/7110/7112's LAN1 and LAN2 are given as:
`ifconfig eth0 192.168.3.127`
`ifconfig eth1 192.168.4.127`
as shown in the following figure. Edit these two lines to modify the static IP addresses.

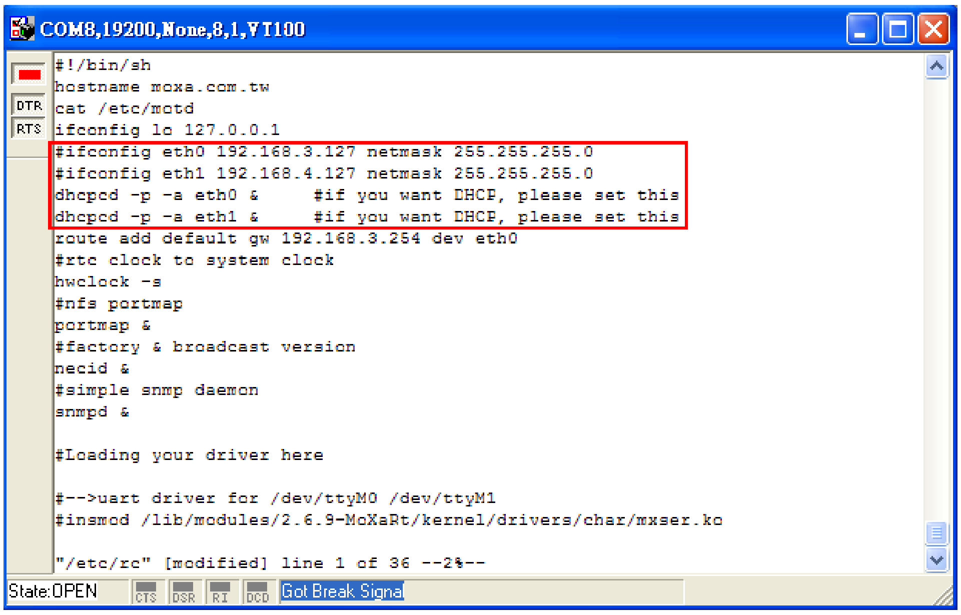


```
# /bin/sh
hostname moxa.com.tw
cat /etc/motd
ifconfig lo 127.0.0.1
ifconfig eth0 192.168.3.127 netmask 255.255.255.0
ifconfig eth1 192.168.4.127 netmask 255.255.255.0
route add default gw 192.168.3.254 dev eth0
#rtc clock to system clock
hwclock -s
#nfs portmap
portmap &
#factory & broadcast version
necid &
#simple snmp daemon
snmpd &

#Loading your driver here

#-->uart driver for /dev/ttyM0 /dev/ttyM1
#insmod /lib/modules/2.6.9-MoXaRt/kernel/drivers/char/mxser.ko

# set the serial port interface, now default set to RS232
# if you want to change, then you need modify following
"rc" line 1 of 34 --2%--
```

3. You may also configure the UC-7101/7110/7112 to request IP addresses from a DHCP server. In this case, use the sharp sign (#) to comment out one or both "ifconfig" lines, and then add the setting about the "dhcpcd" into the rc file as below:

   dhcpcd -p -a eth0 &
   dhcpcd -p -a eth1 &

   Note that the UC-7101/7110/7112 will send out DHCP broadcast packets, and then get the IP addresses from the first DHCP server that responds.



```
#!/bin/sh
hostname moxa.com.tw
cat /etc/motd
ifconfig lo 127.0.0.1
#ifconfig eth0 192.168.3.127 netmask 255.255.255.0
#ifconfig eth1 192.168.4.127 netmask 255.255.255.0
dhcpcd -p -a eth0 &      #if you want DHCP, please set this
dhcpcd -p -a eth1 &      #if you want DHCP, please set this
route add default gw 192.168.3.254 dev eth0
#rtc clock to system clock
hwclock -s
#nfs portmap
portmap &
#factory & broadcast version
necid &
#simple snmp daemon
snmpd &

#Loading your driver here

#-->uart driver for /dev/ttyM0 /dev/ttyM1
#insmod /lib/modules/2.6.9-MoXaRt/kernel/drivers/char/mxser.ko

"/etc/rc" [modified] line 1 of 36 --2%--
```

4. Issue the vi "write" command to save the file, and then reboot. Since the UC-7101/7110/7112 only reads the "rc" file when booting up, you must reboot (e.g., by issuing the vi **`reboot`** command) for the changes to take affect.

ATTENTION

You may reset the IP address immediately by issuing the command:

ifconfig eth0 192.168.5.127

(This will change the IP address of LAN1.)Issuing this command will however NOT update the "rc" file in the UC-7101/7110/7112's flash memory, so the next time you reboot, the IP address will revert to its previous value.

Developing Your Applications

Step 1:
Connect the UC-7101/7110/7112 to a Linux PC.

Step 2:
Install the Tool Chain (GNU Cross Compiler & uClibc).

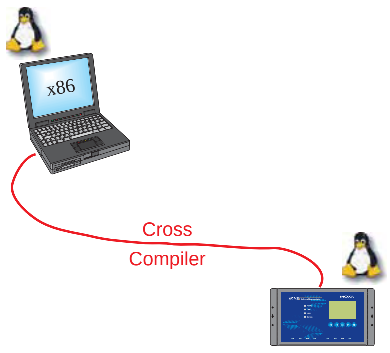
Step 3:
Configure the cross compiler and uClibc environment variables.

Step 4:
Code and compile your program.

Step 5:
Download the program to the UC-7101/7110/7112 by FTP or NFS.

Step 6:
Debug the program. If the program is OK, proceed to Step 7. If the program needs to be modified, go back to Step 4.

Step 7:
Back up the user directory, and then if needed, distribute the code to additional UC-7101/7110/7112 units.



Installing the UC-7101/7110/7112 Tool Chain

Linux

The PC must have the Linux operating system pre-installed to install the UC-7101/7110/7112 Linux GNU Tool Chain. Debian 3.0R-Woody, Redhat 7.3/8.0, and compatible versions are recommended. The Tool Chain requires about 100 MB of hard disk space on your PC. The UC-7101/7110/7112 Tool Chain can be found on the UC-7101/7110/7112 CD. To install the Tool Chain, insert the CD into your PC and then issue the following command:

```
#mount –t iso9660 /dev/cdrom /mnt/cdrom
```

Next, run the following script as root to install the compilers, linkers, and libraries in the **/usr/local** directory:

```
#sh /mnt/cdrom/tool-chain/linux/installer/arm-elf-moxa-toolchain-1.1.sh
```

The Tool Chain installation will take a few minutes to complete.

ATTENTION

You can download the Tool Chain software from Moxa's website. Go to the UC-7101/7110/7112 product page, click the Documentation & Drivers link, and then click **Go** under Driver & Software Downloads.

Compiling Hello.c

The Tool Chain path is:

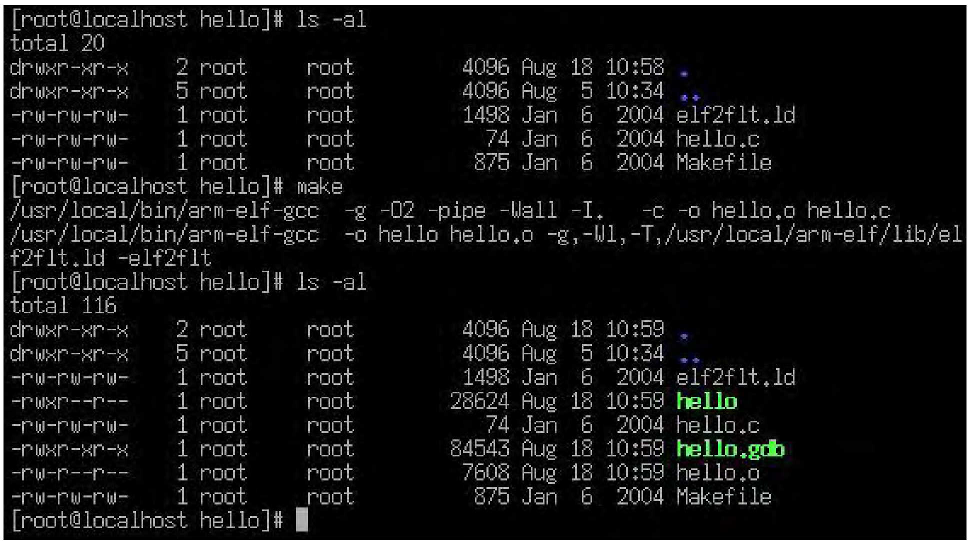
PATH=/usr/local/bin:$PATH

The UC-7101/7110/7112 CD includes several example programs. We use **Hello.c** to illustrate how to compile and run applications.

Issue the following commands from your PC to compile **Hello.c**:

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

Go to the Hello subdirectory, and issue the command **#make** to compile Hello.c. Finally, execute the program to generate **hello** and **hello.gdb**.

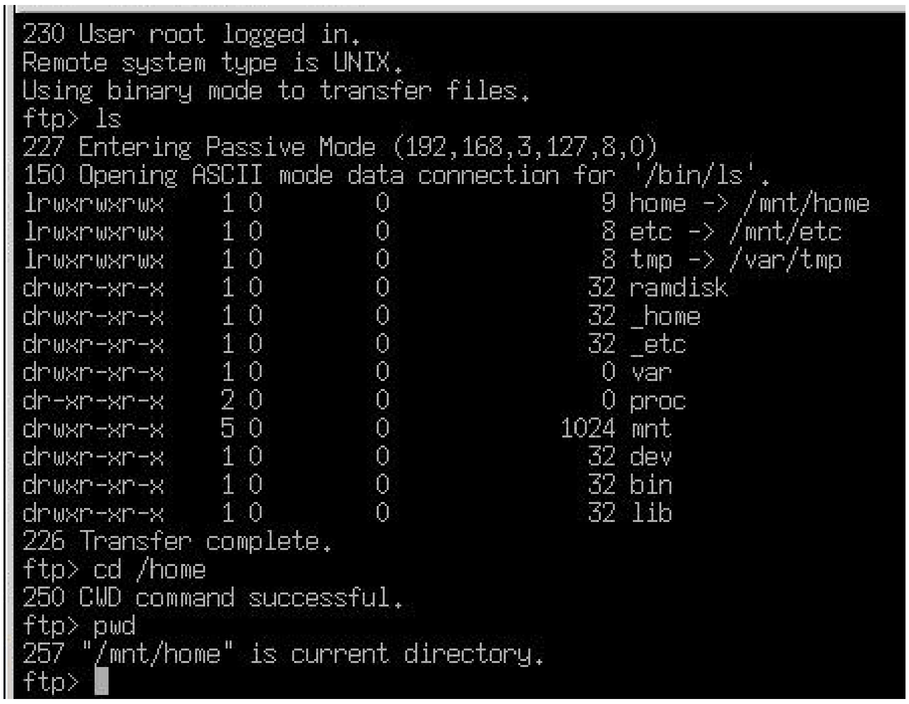


```
[root@localhost hello]# ls -al
total 20
drwxr-xr-x    2 root     root         4096 Aug 18 10:58 .
drwxr-xr-x    5 root     root         4096 Aug  5 10:34 ..
-rw-rw-rw-    1 root     root         1498 Jan  6  2004 elf2flt.ld
-rw-rw-rw-    1 root     root           74 Jan  6  2004 hello.c
-rw-rw-rw-    1 root     root          875 Jan  6  2004 Makefile
[root@localhost hello]# make
/usr/local/bin/arm-elf-gcc  -g -O2 -pipe -Wall -I.   -c -o hello.o hello.c
/usr/local/bin/arm-elf-gcc  -o hello hello.o -g,-Wl,-T,/usr/local/arm-elf/lib/el
f2flt.ld -elf2flt
[root@localhost hello]# ls -al
total 116
drwxr-xr-x    2 root     root         4096 Aug 18 10:59 .
drwxr-xr-x    5 root     root         4096 Aug  5 10:34 ..
-rw-rw-rw-    1 root     root         1498 Jan  6  2004 elf2flt.ld
-rwxr--r--    1 root     root        28624 Aug 18 10:59 hello
-rw-rw-rw-    1 root     root           74 Jan  6  2004 hello.c
-rwxr-xr-x    1 root     root        84543 Aug 18 10:59 hello.gdb
-rw-r--r--    1 root     root         7608 Aug 18 10:59 hello.o
-rw-rw-rw-    1 root     root          875 Jan  6  2004 Makefile
[root@localhost hello]#
```

Uploading "Hello" to the UC-7101/7110/7112

To use FTP to upload **hello** to UC-7101/7110/7112, issue the following commands on the PC:

```
#ftp 192.168.3.127
ftp> cd /home
ftp> bin
ftp> put ./hello
ftp> quit
#telnet 192.168.3.127
```



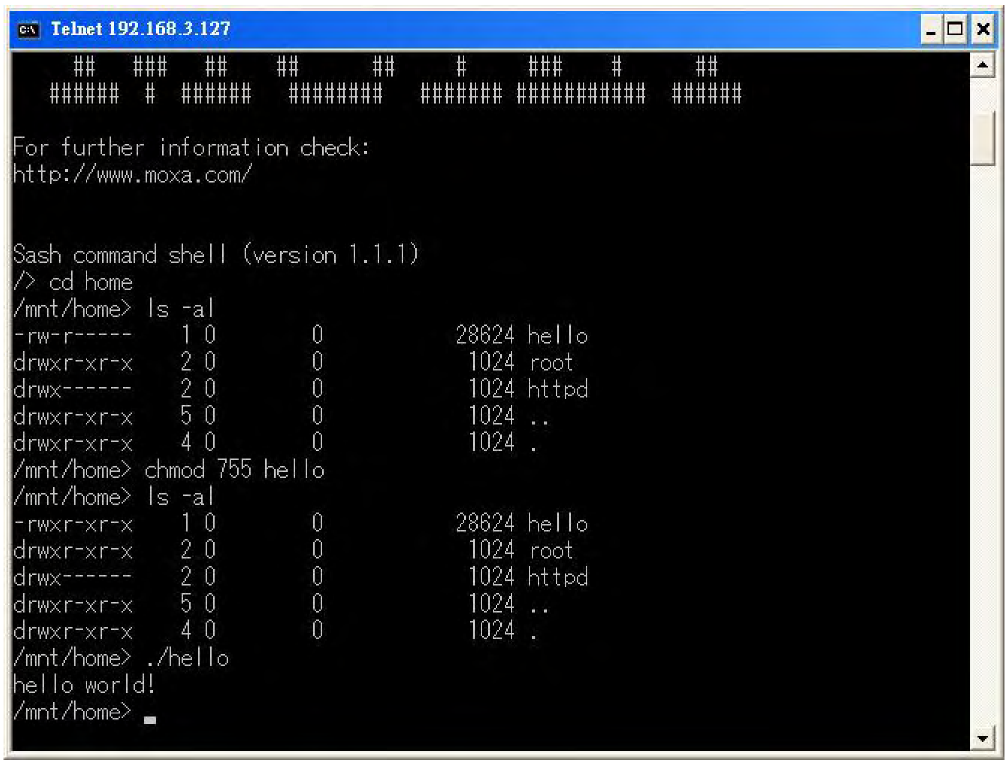
```
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,3,127,8,0)
150 Opening ASCII mode data connection for '/bin/ls'.
lrwxrwxrwx    1 0        0               9 home -> /mnt/home
lrwxrwxrwx    1 0        0               8 etc -> /mnt/etc
lrwxrwxrwx    1 0        0               8 tmp -> /var/tmp
drwxr-xr-x    1 0        0              32 ramdisk
drwxr-xr-x    1 0        0              32 _home
drwxr-xr-x    1 0        0              32 _etc
drwxr-xr-x    1 0        0               0 var
dr-xr-xr-x    2 0        0               0 proc
drwxr-xr-x    5 0        0            1024 mnt
drwxr-xr-x    1 0        0              32 dev
drwxr-xr-x    1 0        0              32 bin
drwxr-xr-x    1 0        0              32 lib
226 Transfer complete.
ftp> cd /home
250 CWD command successful.
ftp> pwd
257 "/mnt/home" is current directory.
ftp>
```

Running "Hello" on the UC-7101/7110/7112

To run the "Hello" program issue the following commands on the UC-7101/7110/7112:

# chmod 755 hello
#./hello

The words "hello world" will be printed on the screen.



```
      ##   ###   ##    ##       ##      #     ###    #      ##
   ######  #  ######   ########   ####### ###########  ######

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


Sash command shell (version 1.1.1)
/> cd home
/mnt/home> ls -al
-rw-r-----    1 0        0           28624 hello
drwxr-xr-x    2 0        0            1024 root
drwx------    2 0        0            1024 httpd
drwxr-xr-x    5 0        0            1024 ..
drwxr-xr-x    4 0        0            1024 .
/mnt/home> chmod 755 hello
/mnt/home> ls -al
-rwxr-xr-x    1 0        0           28624 hello
drwxr-xr-x    2 0        0            1024 root
drwx------    2 0        0            1024 httpd
drwxr-xr-x    5 0        0            1024 ..
drwxr-xr-x    4 0        0            1024 .
/mnt/home> ./hello
hello world!
/mnt/home> _
```

ATTENTION

Be sure to calculate the amount of Flash Memory used by the User File System in the Flash ROM. Use one of the following two commands to determine the amount of memory being used:

`# df -k` or **`# df`**

```
# df
Filesystem           1k-blocks      Used Available Use% Mounted on
rootfs                    1525      1525         0 100% /
/dev/rom0                 1525      1525         0 100% /
/dev/mtdblock2            4096       688      3408  17% /mnt
```

If the flash memory is full, you will no longer be able to save data in Flash ROM. To free up some memory, use the console cable to connect to the UC-7101/7110/7112’s serial console terminal, and then delete files from the Flash ROM.

Sample Makefile Code

The following Makefile example codes are copied from the Hello example on the UC-7101/7110/7112’s CD-ROM.

```
srcdir = .
LDFLAGS =  -Wl,-elf2flt
```

```
LIBS =
CFLAGS =

# Change these if necessary

CC = arm-elf-gcc
CPP = arm-elf-gcc -E

all:	hello

hello:
	$(CC) -o $@ $(CFLAGS) $(LDFLAGS) $(LIBS) $@.c

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

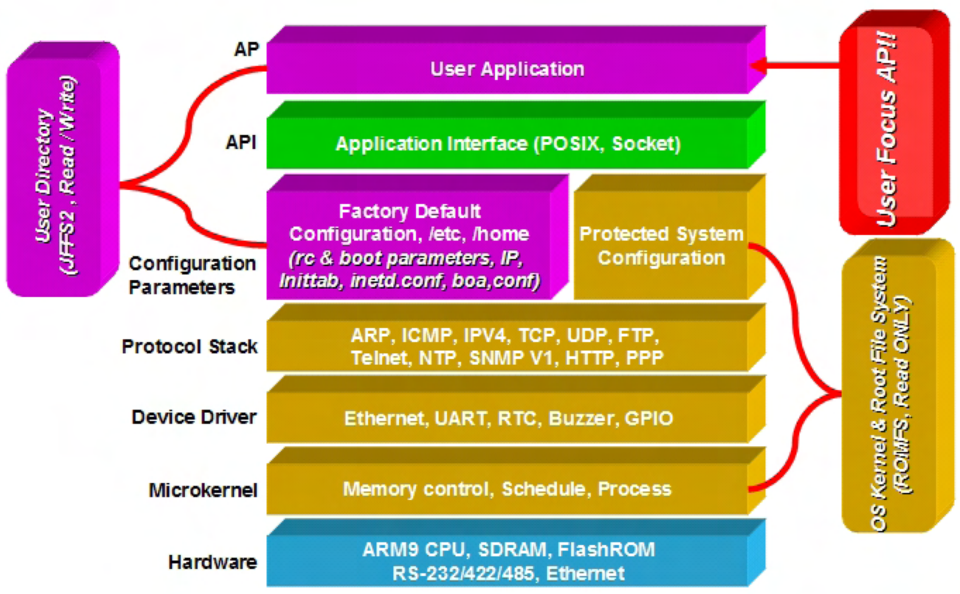
3
Software Package

This chapter includes information about the software that is used with the UC-7101/7110/7112 Series of embedded computers.

This chapter covers the following topics:

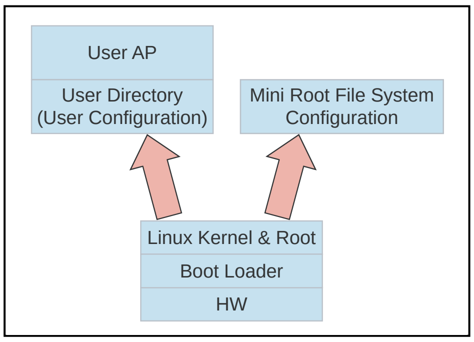
UC-7101/7110/7112 Software Architecture

The pre-installed µClinux operating system used by the UC-7101/7110/7112 follows the standard µClinux architecture, making programs that follow the POSIX standard easily ported to the UC-7101/7110/7112 by using the GNU Tool Chain provided by www.uClinux.org. In addition to the Standard POSIX API, device drivers for the buzzer, and UART for the serial ports are also included.


User Directory (JFFS2 , Read /Write)
AP
User Application
User Focus API!
API
Application Interface (POSIX, Socket)
Factory Default Configuration, /etc, /home (rc & boot parameters, IP, Inittab, inetd.conf, boa,conf)
Protected System Configuration
Configuration Parameters
Protocol Stack
ARP, ICMP, IPV4, TCP, UDP, FTP, Telnet, NTP, SNMP V1, HTTP, PPP
Device Driver
Ethernet, UART, RTC, Buzzer, GPIO
Microkernel
Memory control, Schedule, Process
OS Kernel & Root File System (ROMFS, Read ONLY)
Hardware
ARM9 CPU, SDRAM, FlashROM RS-232/422/485, Ethernet

The UC-7101/7110/7112's Flash ROM has multiple smaller partitions for the Boot Loader, Linux Kernel & Root (/) File System Image, and User Directory.

For most applications, users need to spend a lot time maintaining the operating system and modifying the system configuration. In order to save on the total cost of development and maintenance, the UC-7101/7110/7112 is specially design to partition a "User Directory" for storing the user's system configuration parameters.


User AP
User Directory (User Configuration)
Mini Root File System Configuration
Linux Kernel & Root
Boot Loader
HW

The UC-7101/7110/7112 have a built-in mechanism that prevents system crashes and improves system reliability. The procedure is described below.

When the Linux kernel boots up, the kernel mounts the root file system and then enables services and daemons. The kernel also looks for the system configuration parameters using rc or inittab.

Normally, the kernel uses the User Directory to boot up the system. The kernel will only use the default configuration _etc & _home when the User Directory crashes.

The UC-7101/7110/7112 uses ROMFS for the Linux kernel image, Root File System, and Protected configuration, and uses JFFS2 for the User Directory.

The partition sizes are hard coded into the kernel binary. You must rebuild the kernel to change the partition sizes. The flash memory map is shown in the following table.

Flash Context	Flash Address	Size	Access control
Boot loader	0 – 0x3ffff	256 K	Read ONLY
Kernet & Root File System	0x40000– 0x3fffff	4 M	Read ONLY **JFFS2**
User Directory	0x400000 – 0x7fffff	4 M – 256 K	Read / Write **JFFS2**

Developers write their own programs only on partitions **/etc**, **/home**, **/tmp**, and **/usr/bin**. It is advised the executed file be put in **/usr/bin** as this will allow developers to use hotkeys.

In addition to the flash file systems, a RAM based file system is mounted on **/var/**.

Journaling Flash File System (JFFS2)

The flash User Directory is formatted by the Journaling Flash File System (JFFS2), which places a compressed file system on the flash, transparent to the user.

Axis Communications in Sweden developed the Journaling Flash File System (JFFS2).

JFFS2 provides a file system directly on flash, rather than emulating a block device designed for use on flash-ROM chips. It recognizes flash-ROM chips' special write requirements, does wear-leveling to extend flash life, keeps the flash directory structure in the RAM at all times, and implements a log-structured file system that is always consistent—even if the system crashes or unexpectedly powers down. It does not require fsck on boot up.

JFFS2, a newer version of JFFS, provides improved wear-leveling and garbage-collection performance, an 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 (to enhance the write-life of the devices), native data compression inside the file system design, and support for hard links.

Key features of JFFS2 are:

- Directly targeted to Flash ROM
- Robust
- Consistent across power failure
- 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 ensure that data will not be lost. The file system will remain in a consistent state across power failures, and will always be mountable. However, if the board is powered down during a write, then the incomplete write will be rolled back on the next boot. Any writes that were already completed will not be affected.

Additional information about JFFS2 is available on the following websites:
http://sources.redhat.com/jffs2/jffs2.pdf
http://developer.axis.com/software/jffs/
http://www.linux-mtd.infradead.org/

UC-7101/7110/7112 Software Package

bin	dev	
upkernel	mtdblock1	ptype
passwd -> tinylogin	mtdr1	ptypd
login -> tinylogin	mtd1	ptypc
tinylogin	mtdblock0	ptypb
telnetd	mtdr0	ptypa
snmpd	mtd0	ptyp9
mail	cum1	ptyp8
sh	cum0	ptyp7
routed	ttyM1	ptyp6
netstat	ttyM0	ptyp5
arp	urandom	ptyp4
chat	random	ptyp3
pppd	zero	ptyp2
portmap	ttypf	ptyp1
ntpdate	ttype	ptyp0
necid	ttypd	ppp
eraseall	ttypc	pio
kversion	ttypb	rtc
init	ttypa	ram1
expand	ttyp9	ram0
inetd	ttyp8	null
hwclock	ttyp7	kmem
ftpd	ttyp6	mem
ftp	ttyp5	cua0
mke2fs	ttyp4	console
e2fsck	ttyp3	tty
discard	ttyp2	
dhcpcd	ttyp1	
cpu	ttyp0	
busybox	ttyS0	
boa	tty3	
downramdisk	tty0	
upramdisk	rom1	
	rom0	
	ptypf	

4
Configuring UC-7101/7110/7112

In this chapter, we describe how to configure the UC-7101/7110/7112 embedded computers.

The following topics are covered in this chapter:

Enabling and Disabling Daemons

The following daemons are enabled when the UC-7101/7110/7112 boot up for the first time.

• SNMP Agent daemon:	**snmpd**
• Telnet Server / Client daemon:	**telnetd**
• Internet Daemons:	**inetd**
• FTP Server / Client daemon:	**ftpd**
• WWW Server daemon:	**boa**

ATTENTION

How to enable/disable telnet/ftp server

a. Edit the file '/etc/inetd.conf'
Example (default enable):
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

b. Disable the daemon by typing '#' in front of the first character of the row.

How to enable/disable /etc/inittab www server

a. Edit the file '/etc/inittab'

b. Disable the www service by typing "#" in front of the first character of the row.

How to enable Network Time Protocol

ntpdate is a time adjusting client utility. The UC-7101/7110/7112 play the role of Time client, and sends requests to the Network Time Server to request the correct time.

Set the time server address for adjusting the system time with the command:
```
/>ntpdate ntp_server_ip
```

Save the system time to the hardware's real time clock with the command:
```
/>hwclock -w
```

Visit http://www.ntp/org for a list of recommended public NTP servers.

How to update the system time periodically with Network Time Protocol

1. Create a shell script file that includes the following description.
```
#!/bin/sh
ntpdate ntp_server_ip
hwclock –w
sleep 100          ← The min time is 100ms.
```
2. Save and make this shell script executable by typing
```
chmod 755 <shell-script_name>
```

Edit the file '/etc/inittab' by adding the following line:
```
ntp: unknown: /directory/<shell_script_name>
```

Adding a Web Page

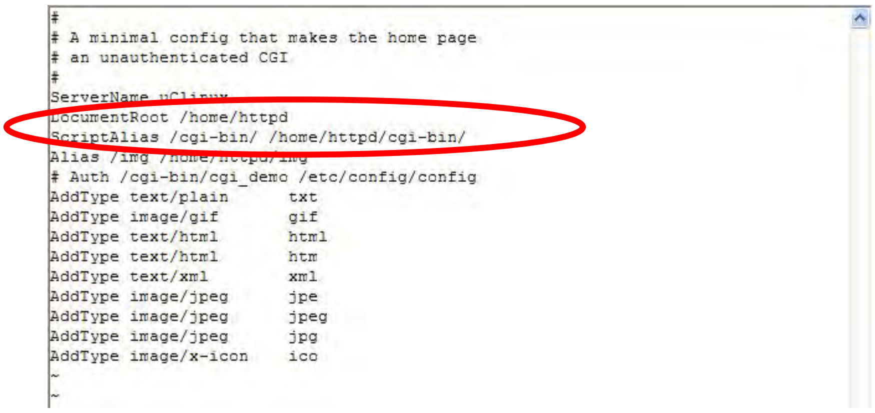
Default Home Page address:

/home/httpd/index.html

You may change the default home page directory by editing the web server's configuration file, located at: **/etc/boa.conf**.

Type the following command to edit the boa.conf file:

/etc>vi boa.conf



```
#
# A minimal config that makes the home page
# an unauthenticated CGI
#
ServerName uClinux
DocumentRoot /home/httpd
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/
Alias /img /home/httpd/img
# Auth /cgi-bin/cgi_demo /etc/config/config
AddType text/plain      txt
AddType image/gif       gif
AddType text/html       html
AddType text/html       htm
AddType text/xml        xml
AddType image/jpeg      jpe
AddType image/jpeg      jpeg
AddType image/jpeg      jpg
AddType image/x-icon    ico
~
~
```

To add your web page, place your home page in the following directory:

/home/httpd/

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 the action to be taken 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-7101/7110/7112 support three types of IPTABLES tables: 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 the Internet, then other computers that connect to this host can connect to the Internet when the computer does not have an actual IP address.

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

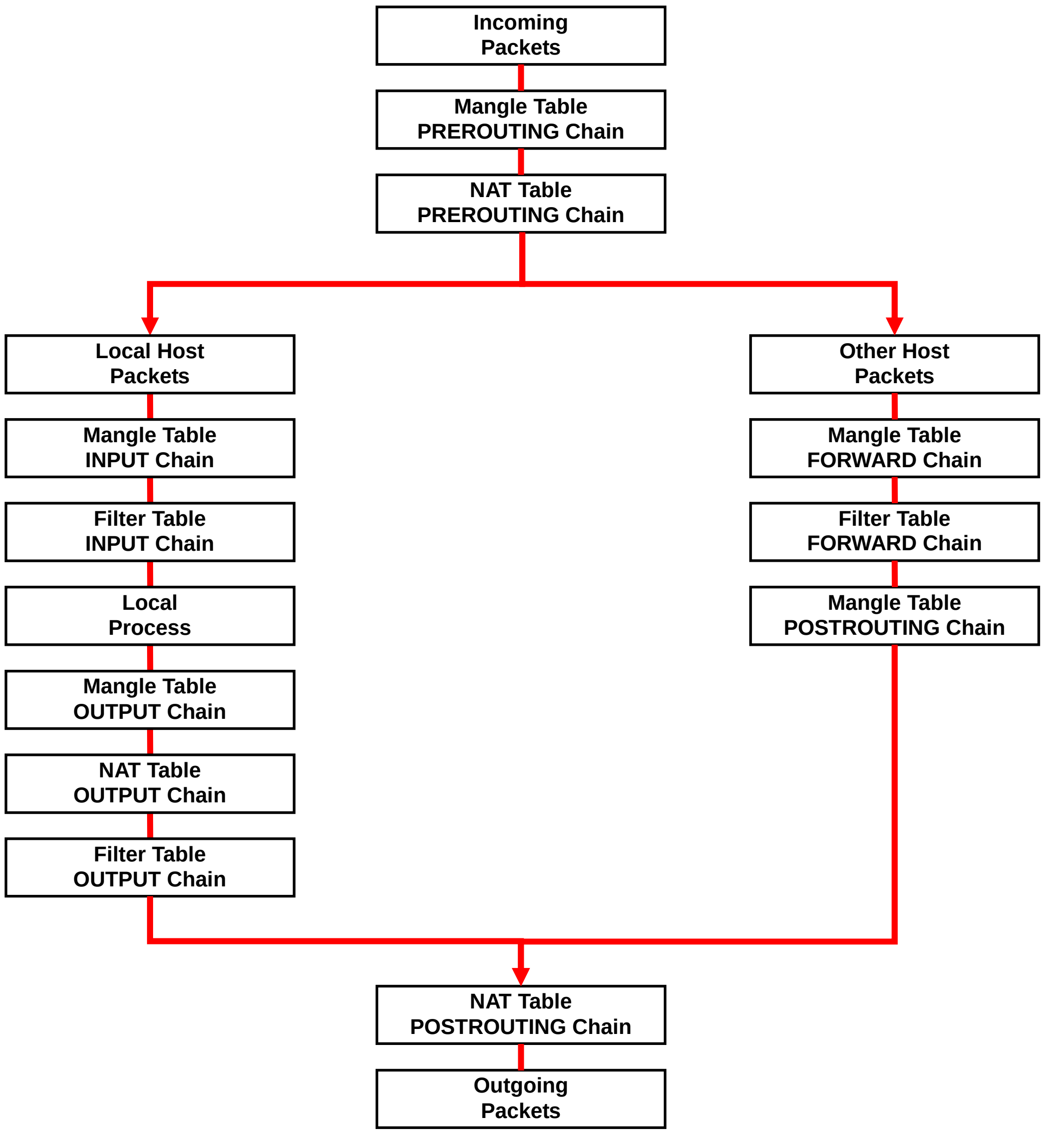
C. **Mangle Table**—includes two chains

PREROUTING chain—pre-processes packets before the routing process.

OUTPUT chain—processes packets after the routing process.

It has three extensions—TTL, MARK, TOS.

The following figure shows the IPTABLES hierarchy.


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

NOTE The UC-7101/7110/7112 do NOT support IPV6 and ipchains.
IPTABLES supports 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 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-7101/7110/7112.
OUTPUT:	For locally-generated packets.
FORWARD:	For packets routed out through the UC-7101/7110/7112.
PREROUTING:	To alter packets as soon as they come in.
POSTROUTING:	To alter packets as they are about to be sent out.

Examples:

```
#iptables –P INPUT DROP
#iptables –P OUTPUT ACCEPT
#iptables –P FORWARD ACCEPT
```

```
#iptables –t nat –P PREROUTING ACCEPT
#iptables –t nat –P OUTPUT ACCEPT
#iptables -t nat –P POSTROUTING ACCEPT
```

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

Append or delete rules:

Usage:

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

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

Examples:

Example 1: Accept all packets from lo interface.
```
# iptables –A INPUT –i lo –j ACCEPT
```

Example 2: Accept TCP packets from 192.168.0.1.
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.0.1 –j ACCEPT
```

Example 3: Accept TCP packets from Class C network 192.168.1.0/24.
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.1.0/24 –j ACCEPT
```

Example 4: Drop TCP packets from 192.168.1.25.
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.1.25 –j DROP
```

Example 5: Drop TCP packets addressed for port 21.
```
# iptables –A INPUT –i eth0 –p tcp --dport 21 –j DROP
```

Example 6: Accept TCP packets from 192.168.0.24 to UC-7101/7110/7112's port 137, 138, 139
```
# iptables –A INPUT –i eth0 –p tcp –s 192.168.0.24 --dport 137:139 –j ACCEPT
```

Example 7: Log TCP packets that visit UC-7101/7110/7112's port 25.
```
# iptables –A INPUT –i eth0 –p tcp --dport 25 –j LOG
```

Example 8: Drop all packets from MAC address 01:02:03:04:05:06.
```
# iptables –A INPUT –i eth0 –p all –m mac –mac-source 01:02:03:04:05:06 –j DROP
```

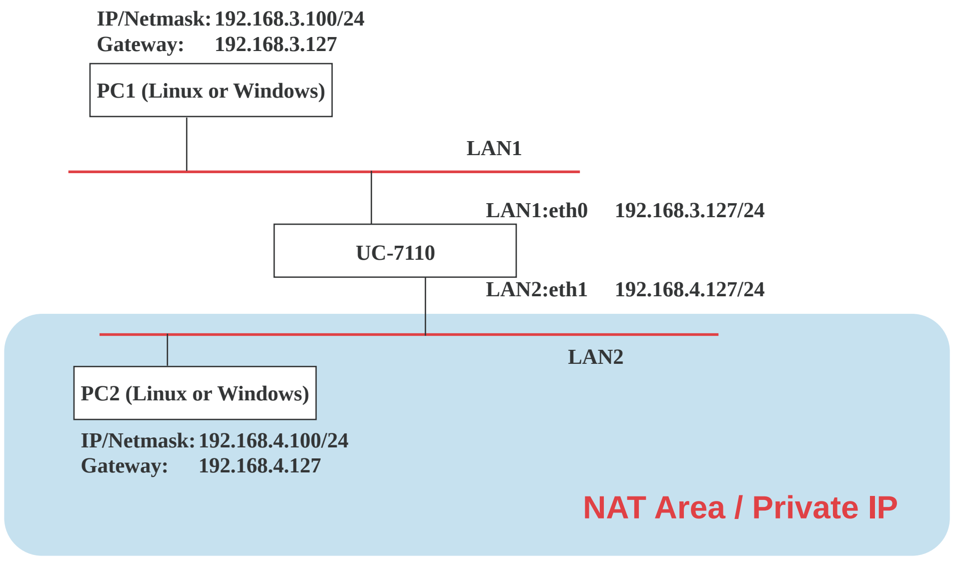
NAT

NAT (Network Address Translation) protocol translates IP addresses used on one network into different IP addresses used on another network. One network is designated the inside network and the other is the outside network. Typically, the UC-7101/7110/7112 connects several devices on a network and maps local inside network addresses to one or more global outside IP addresses, and remaps 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 addresses of all packets leaving LAN1 are changed to 192.168.3.127 (you will need to load the module ipt_MASQUERADE):



```
1.  #echo 1 > /proc/sys/net/ipv4/ip_forward
2.  insmod /lib/modules/2.6.19-uc1MoXaRt/kernel/net/netfilter/x_tables.ko
3.  insmod
    /lib/modules/2.6.19-uc1MoXaRt/kernel/net/netfilter/xt_multiport.ko
4.  insmod /lib/modules/2.6.19-uc1MoXaRt/kernel/net/netfilter/xt_MARK.ko
5.  insmod /lib/modules/2.6.19-uc1MoXaRt/kernel/net/netfilter/xt_tcpudp.ko
6.  insmod
    /lib/modules/2.6.19-uc1MoXaRt/kernel/net/ipv4/netfilter/ip_tables.ko
7.  insmod
    /lib/modules/2.6.19-uc1MoXaRt/kernel/net/ipv4/netfilter/ip_nat.ko
8.  insmod
    /lib/modules/2.6.19-uc1MoXaRt/kernel/net/ipv4/netfilter/iptable_nat.ko
9.  insmod
    /lib/modules/2.6.19-uc1MoXaRt/kernel/net/ipv4/netfilter/ipt_MASQUERADE
    .ko
10. insmod
    /lib/modules/2.6.19-uc1MoXaRt/kernel/net/ipv4/netfilter/ip_nat_ftp.ko
11. #iptables -t nat –A POSTROUTING –o eth0 –j SNAT --to-source 192.168.3.127
```

or

```
12. #iptables –t nat –A POSTROUTING –o eth0 –j MASQUERADE
```

Enabling NAT at Bootup

In most real world situations, you should use a simple shell script to enable NAT when the UC-7101/7110/7112 boot up, as indicated by the following:

1. setting iptables
2. iptables-save > /home/xxx.file (xxx.file is the user defined file name)
3. vi /etc/rc
4. Append echo 1 > /proc/sys/net/ipv4/ip_forward
5. Append iptables-restore /home/xxx.file (xxx.file is the user defined file name)

Configuring Dial-in/Dial-out Service

Dial-out Service

Direct cable connection:

- *Without* username and password, use:
```
/>pppd connect 'chat -v' /dev/ttyM0 38400 crtscts&
```
- *With* username and password, use:
```
/>pppd connect 'chat -v' user xxxxx password xxxxx /dev/ttyM0 38400 crtscts&
```

Connect Using a Modem:

- Use:
```
/>pppd connect 'chat -v ATDT<phone_number> CONNECT' user xxxxx password xxxxx
/dev/ttyM0 38400 crtscts&
```

ATTENTION

If dial out fails, the pppd connection will be blocked, and the users will need to shut down pppd, and re-dial. Since the return value is always OK (regardless of whether or not the connection is blocked), the API must be set up to check the network status to determine if the connection is complete.

Dial-in Service

Direct cable connection:

- Use either of the following:
```
/>pppd <Local_IP_Address>:<Remote_IP_Address> /dev/ttyM1 38400 local crtscts
```
or
```
/>pppd <Local_IP_Address>:<Remote_IP_Address> /dev/ttyM0 38400 local crtscts login
auth
```

Connect Using a Modem:

- Use:
```
/>pppd connect 'chat -v AT CONNECT' <local_IP_Address>:<Remote_IP_Address>
/dev/ttyM0 38400 crtscts login auth
```

Configuring PPPoE

PPPoE relies on two widely accepted standards: PPP and Ethernet, which permit the use of PPPoE(Point-to-Point Over Ethernet).

PPPoE is a specification for connecting users on an Ethernet to the Internet through a common broadband medium, such as a single DSL line, wireless device or cable modem, used by many ADSL service providers. All users on the Ethernet share a common connection, so the Ethernet principles that support multiple users on a LAN combine with the PPP principles, which apply to serial connections.

- Create the Connection:

  ```
  />pppd pty "pppoe -I <ETHERNET_INTERFACE> -m 1412" user <USER_NAME> password
  <USER_PASSWORD>&
  ```

  <ETHERNET_INTERFACE>: Ethernet card connected to ADSL modem, for example, eth0
  <USER_NAME>: User account, for example, moxa@adsl.net
  <USER_PASSWORD>: Password for user account

To check if PPPOE is successfully connected, use the command:

- `/>ifconfig ppp0`

How to Mount a Remote NFS Server

Currently, the UC-7101/7110/7112 only supports NFS (Network File System) clients. Users can open NFS service on a Linux PC to enable the UC-7101/7110/7112 to push data to it. The UC-7101/7110/7112 can use NFS to mount a remote disk as a local disk for data or log purposes.

1. First, the NFS server must open an export directory and allow access to the IP address. Edit the file "/etc/exports" on your Linux PC, and then run the NFS daemon. The following example gives one possibility (refer to the NFS-HOWTO document at http://nfs.sourceforge.net/nfs-howto/server.html):

   ```
   /home/usr 192.168.3.1 (rw,no_root_squash,no_all_squash)
   ```

2. The UC-7101/7110/7112 must run the "portmap" utility. This program is enabled by default in the "/etc/rc" file. Use the following command to mount the remote NFS server:

   ```
   />mount –t nfs <remote-ip>:<remote-export-directory> <local-directory>
   ```

Dynamic Driver Module Load / Unload

In addition to supporting traditional static drivers, the UC-7101/7110/7112 also support the dynamic driver module load / unload mechanism. It allows user to load a special driver into the kernel to enable hardware features for specific applications. To load / unload a dynamic driver module, use the following commands.

Load module:
```
/>insmod <module-directory>/<module file name>
```

For example, to load the UART driver, type the following command:
```
/>insmod /lib/modules/2.6.9-MoXaRt/kernel/drivers/char/mxser.ko
```

Show module list:
```
/>lsmod
```

Unload module:
```
/>rmmod <module-name listed by lsmod command>
```

For example, to unload the UART driver, type the following command:
`/>rmmod mxser`

For the UC-7110, the factory default is to load the UART driver "mxser.ko". The additional driver module to control the SD/MMC memory card is loaded for the UC-7112. Please see the information below for the locations and file names of these driver modules.

UART:

`/lib/modules/2.6.9-MoXaRt/kernel/drivers/char/mxser.ko`

SD/MMC:

`/lib/modules/2.6.9-MoXaRt/kernel/drivers/mmc/mmc_core.ko`

`/lib/modules/2.6.9-MoXaRt/kernel/drivers/mmc/mmc_block.ko`

`/lib/modules/2.6.9-MoXaRt/kernel/drivers/mmc/moxasd.ko`

Upgrading the Kernel

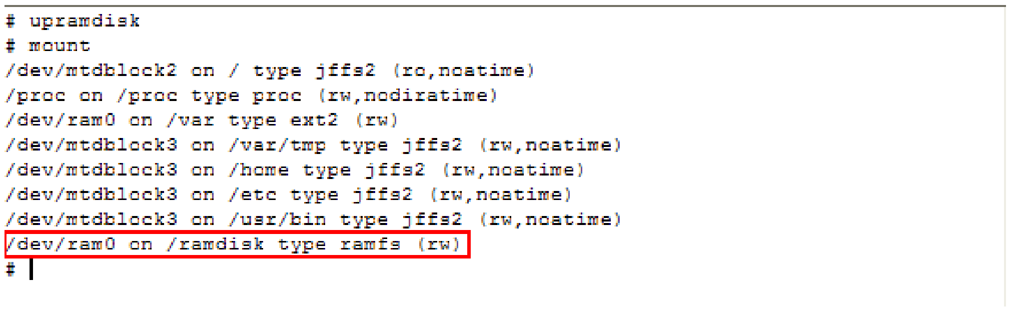
The UC-7101/7110/7112 kernel is ***uc7110-3.x..bin (uc7112-1.x.bin for UC-7112)***, which can be downloaded from www.moxa.com. You must first download this file to your PC, and then use the Console Terminal or Telnet Console to copy the file to the UC-7101/7110/7112.

You can save this file to the UC-7101/7110/7112's RAM disk, and then upgrade the kernel. The following is a step-by-step example.

To enable the RAM disk, use the following command:

/>upramdisk

After executing "upramdisk", you may use "mount" to find out if the new ramdisk has been created successfully:



```
# upramdisk
# mount
/dev/mtdblock2 on / type jffs2 (ro,noatime)
/proc on /proc type proc (rw,nodiratime)
/dev/ram0 on /var type ext2 (rw)
/dev/mtdblock3 on /var/tmp type jffs2 (rw,noatime)
/dev/mtdblock3 on /home type jffs2 (rw,noatime)
/dev/mtdblock3 on /etc type jffs2 (rw,noatime)
/dev/mtdblock3 on /usr/bin type jffs2 (rw,noatime)
/dev/ram0 on /ramdisk type ramfs (rw)
#
```

To navigate to the device node, use the following command:

/>cd ramdisk

Use the built-in FTP client to download the uc7110-3.x.bin file from the PC.

/ramdisk>ftp <destination PC's IP>

Login Name: xxxx

Login Password: xxxx

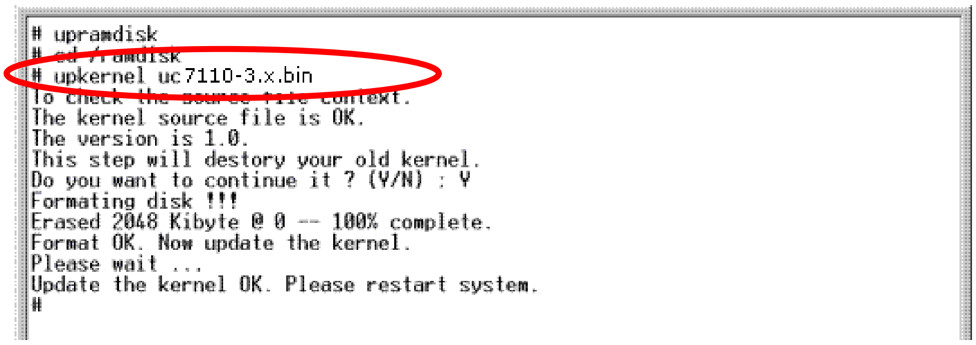
ftp> bin

ftp> get uc7110-3.x.bin

Use the **upkernel** command to upgrade the kernel and root file system.

/ramdisk>upkernel uc7110-3.x.bin

/ramdisk>reboot



```
# upramdisk
# cd /ramdisk
# upkernel uc7110-3.x.bin
To check the source file context.
The kernel source file is OK.
The version is 1.0.
This step will destory your old kernel.
Do you want to continue it ? (Y/N) : Y
Formating disk !!!
Erased 2048 Kibyte @ 0 -- 100% complete.
Format OK. Now update the kernel.
Please wait ...
Update the kernel OK. Please restart system.
#
```

Upgrading the Root File System & User Directory

The UC-7101/7110/7112 uses JFFS2 for the root file system and user directory. By default, the root file system is pre-set to READ only. The UC-7101/7110/7112 provides a read/write user's directory in the JFFS2 file system. Use this user's directory to store the system configuration file and user's programs on the disk.

You can search the UC-7101/7110/7112's CD-ROM for the latest user directory file, or download the file from www.moxa.com. The format of the file is uc7110-3.x.dsk (uc7112-1.x.dsk for UC-7112). You must download this file to a PC first, and then use the console terminal or Telnet console to copy the file to the UC-7101/7110/7112.

You can save this file to the UC-7101/7110/7112's RAM disk, and then upgrade the user directory. A step-by-step example is shown below.

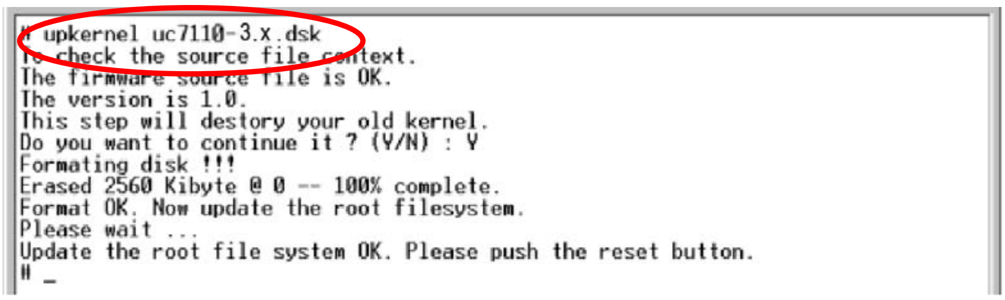
Use the following commands to enable the RAM disk:

/>upramdisk
/>cd ramdisk

Use the built-in FTP client to download the uc7110-3.x.dsk file from the PC:

/ramdisk>ftp <destination PC's IP>
Login Name: xxxx
Login Password: xxxx
ftp> bin
ftp> get uc7110-3.x.dsk
ftp>quit
/ramdisk>upkernel /ramdisk/uc7110-3.x.dsk
/reboot

You will also need to restore factory defaults to load the new settings. To do this, either press the RESET button for more than 5 seconds, or input the command "stdef" from the Telnet console.



```
# upkernel uc7110-3.x.dsk
To check the source file context.
The firmware source file is OK.
The version is 1.0.
This step will destory your old kernel.
Do you want to continue it ? (Y/N) : Y
Formating disk !!!
Erased 2560 Kibyte @ 0 -- 100% complete.
Format OK. Now update the root filesystem.
Please wait ...
Update the root file system OK. Please push the reset button.
# _
```

Loading Factory Defaults

The easiest way to "Load Factory Defaults" is with the "Upgrade User directory" operation.

Refer to the previous section **Upgrading the Root File Sysem & User Directory** for an introduction.

You may also press the RESET button for more than 5 seconds to load the factory default configuration or input the command "stdef" from the Telnet console to restore the factory defaults.

Autostarting User Applications on Bootup

To autostart user applications on bootup, edit the **/etc/rc** file by adding your application program. For example, you might add the following line to the file:

/ap-directory/ap-program &

Checking the Kernel and Root File System Versions

Use the following commands to check the version of the kernel and root file system:

Use the following command to check the kernel version:

/>kversion

Use the following command to check the root file system (firmware) version of the UC-7101/7110/7112:

/>fsversion

Use the following command to check the user directory version of the UC-7101/7110/7112:

/>cat /etc/version

5

UC-7101/7110/7112 Device API

In this chapter, we discuss the Device API for the UC-7101/7110/7112 Series. We introduce the APIs for the following functions:

- ❑ **RTC (Real Time Clock)**
- ❑ **Buzzer**
- ❑ **UART Interface**
- ❑ **WDT (Watch Dog Timer)**

RTC (Real Time Clock)

The device node is located at **`/dev/rtc`**. The UC-7101/7110/7112 support µClinux standard simple RTC control. You must include **`<linux/rtc.h>`** to use these functions.

1. Function: `RTC_RD_TIME`

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

   Description: Reads time information from RTC.

2. Function: `RTC_SET_TIME`

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

   Description: Sets RTC time.

Buzzer

The device node is located at **`/dev/console`**. The UC-7101/7110/7112 support µClinux standard buzzer control. The UC-7101/7110/7112's buzzer runs at a fixed frequency of 100 Hz. You must include **`<sys/kd.h>`** to use these functions.

1. Function: `KDMKTONE`

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

   Description: Buzzer will beep, as stipulated by the function arguments.

UART Interface

The normal tty device node is located at **/dev/ttyM0…ttyM1**, and modem tty device node is located at **/dev/com0 … com1**. The UC-7101/7110/7112 Series support µClinux standard termios control. The Moxa UART Device API supports the configuration of ttyM0 to ttyM1 as RS-232/422/485. To use these functions, after the Tool Chain package is installed, then include <moxadevice.h> in your application.

```
#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 mode;
   mode=which mode you want to set;
   int ioctl(fd, MOXA_SET_OP_MODE, &mode)
   ```

   Description: Sets the interface mode.

2. Function: `MOXA_GET_OP_MODE`

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

   Description: Gets the interface mode.

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 disabled when the system boots up. The user application can also enable ack. When the user does not ack, it will let the system reboot.

Kernel boot

…..

….

User application running and enable user ack

….

….

3. The user API

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

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

int swtd_open (void)

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

Input
None

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

Use errno() to retrieve errors.

int swtd_enable (int fd, unsigned long time)

Description
Enable 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 be reboot automatically. The minimum time is 50 msec, the maximum time is 60 seconds. The time unit is msec.

Output

0 (zero) for no error. Any other number indicates an error. You can get the error code from errno().

int swtd_disable (int fd)

Description
Disable the application to ack sWatchDog. The kernel will auto ack it. Users does not to do it periodically.

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

Output
0 (zero) for no error. Any other number indicates an error. You can get the error code from errno().

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

Description
Get current setting values.

Mode—1 for user application enable sWatchDog: need to do ack.
0 for user application disable sWatchdog: does not need to do ack.

time – The time period to ack sWatchDog.

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

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

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

Output
0 (zero) for no error. Any other number indicates an error. You can get the error code from errno().

int swtd_ack (int fd)

Description
Acknowledge sWatchDog. When the user application has enabled sWatchDog, it will call this function periodically with a user-predefined time in the application program.

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

Output
0 (zero) for no error. Any other number indicates an error. You can get the error code from errno().

int swtd_close (int fd)

Description
Close the file handle.

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

Output
0 (zero) for no error. Any other number indicates an error. You can get the error code from errno().

4. Special Note

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

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

5. User application example

```
Example 1:
#include  <stdio.h>
#include  <stdlib.h>
#include  <string.h>
#include  <moxadevice.h>

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

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

The makefile is shown below:

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

```
Example 2:
#include  <stdio.h>
#include  <stdlib.h>
#include  <signal.h>
#include  <string.h>
#include  <sys/stat.h>
#include  <sys/ioctl.h>
#include  <sys/select.h>
#include  <sys/time.h>
#include  <moxadevice.h>

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

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

static int swtdfd;
static int stopflag=0;

static void stop_swatchdog()
{
        stopflag = 1;
}

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

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

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

The makefile is shown below:

All:

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

A
System Commands

µClinux normal command utility collection

File manager

cp	copy file
ls	list file
ln	make symbolic link file
mount	mount and check file system
rm	delete file
chmod	change file owner & group & user
chown	change file owner
chgrp	change file group
sync	sync file system; save system file buffer to hardware
mv	move file
pwd	display active file directly
df	list active file system space
du	estimate file space usage
mkdir	make new directory
rmdir	delete directory
head	print the first 10 lines of each file to standard output
tail	print the last 10 lines of each file to standard output
touch	update the access and modification times of each file to the current time

Editor

vi	text editor
cat	dump file context
grep	print lines matching a pattern
cut	remove sections from each line of files
find	search for files in a directory hierarchy
more	dump file by one page
test	test if file exists or not
echo	echo string

Network

ping	ping to test network
route	routing table manager
netstat	display network status
ifconfig	set network IP address
tftp	tftp protocol
telnet	user interface to TELNET protocol
ftp	file transfer protocol
iptables	iptables command

Process

kill	kill process
killall	kill process by name
ps	report process status
sleep	suspend command on time

Other

dmesg	dump kernel log message
stty	set serial port
mknod	make device node
free	display system memory usage
date	print or set the system date and time
env	run a program in a modified environment
clear	clear the terminal screen
reboot	reboot / power off/on the server
halt	halt the server
gzip, gunzip, zcat	compress or expand files
hostname	show system's host name
tar	tar archiving utility

Moxa Special Utilities

cat /etc/version	show user directory version
upramdisk	mount ramdisk
downramdisk	unmount ramdisk
kversion	show kernel version
setinterface	set UART interfaces program

B
SNMP Agent with MIB II & RS-232 Like Group

The UC-7101/7110/7112 has a built-in SNMP (Simple Network Management Protocol) agent that supports RFC1317 RS-232 like group and RFC 1213 MIB-II. The following table lists the variable implementation for the UC-7101/7110/7112.

The full SNMP object ID of the UC-7101/7110/7112 is **.iso.3.6.1.4.1.8691.12.7112** and **.iso.3.6.1.4.1.8691.12.7110**.

Note that the UC-7101/7110/7112 does not support SNMP trap.

RFC1213 MIB-II supported SNMP variables:

system MIB	**interface MIB**	**at MIB**	**icmp MIB**
sysDescr	ifNumber	atTable	icmpInMsgs
sysObjectID	ifTable	atIfIndex	icmpInErrors
sysUpTime	ifIndex		icmpInDestUnreachs
sysContact	ifDescr	atPhysAddress	icmpInTimeExcds
sysName	ifType	atNetAddress	icmpInParmProbs
sysLocation	ifMtu		icmpInSrcQuenchs
sysServices	ifSpeed		icmpInRedirects
	ifPhysAddress		icmpInEchos
	ifAdminStatus		icmpInEchoReps
	ifOperStatus		icmpInTimestamps
	ifLastChange		icmpInAddrMasks
	ifInOctets		icmpInAddrMaskReps
	ifInUcastPkts		icmpOutMsgs
	ifInNUcastPkts		icmpOutErrors
	ifInDiscards		icmpOutDestUnreachs
	ifInErrors		icmpOutTimeExcds
	ifInUnknownProtos		icmpOutParmProbs
	ifOutOctets		icmpOutSrcQuenchs
	ifOutUcastPkts		icmpOutRedirects
	ifOutNUcastPkts		icmpOutEchos
	ifOutDiscards		icmpOutEchoReps
	ifOutErrors		icmpOutTimestamps
	ifOutQLen		icmpOutAddrMasks
	ifSpecific		icmpOutAddrmaskReps

ip MIB	**tcp MIB**	**udp MIB**
ipForwarding	tcpRtoAlgorithm	udpInDatagrams
ipDefaultTTL	tcpRtoMin	udpNoPorts
ipInReceives	tcpRtoMax	udpInErrors
ipInHdrErrors	tcpMaxConn	udpOutDatagrams
ipInAddrErrors	tcpActiveOpens	udpTable
ipForwDatagrams	tcpPassiveOpens	udpLocalAddress
ipInUnknownProtos	tcpAttemptFails	udpLocalPort
ipInDiscards	tcpEstabResets	
ipInDelivers	tcpCurrEstab	
ipOutRequests	tcpInSegs	
ipOutDiscards	tcpOutSegs	
ipOutNoRoutes	tcpRetransSegs	
ipReasmTimeout	tcpConnTable	
ipReasmReqds	tcpConnState	
ipReasmFails	tcpConnLocalAddress	
ipFragOKs	tcpConnLocalPort	
ipFragFails	tcpConnRemAddress	
ipFragCreates	tcpConnRemPort	
ipAddrTable	tcpInErrs	
ipAdEntAddr	tcpOutRsts	
ipAdEntIfIndex		
ipAdEntNetMask		
ipAdEntBcastAddr		
ipAdEntReasmMaxSize		
ipRouteTable		
ipRouteDest		
ipRouteIfIndex		
ipRouteMetric1		
ipRouteMetric2		
ipRouteMetric3		
ipRouteMetric4		
ipRouteNextHop		
ipRouteType		
ipRouteProto		
ipRouteAge		
ipRouteMask		
ipRouteMetric5		
ipRouteInfo		
ipNetToMediaTable		
ipNetToMediaIfIndex		
ipNetToMediaPhysAddress		
ipNetToMediaNetAddress		
ipNetToMediaType		
ipRoutingDiscards		

snmp MIB
snmpInPkts
snmpOutPkts
snmpInBadVersions
snmpInBadCommunityNames
snmpInBadCommunityUses
snmpInASNParseErrs
snmpInTooBigs
snmpInNoSuchNames
snmpInBadValues
snmpInReadOnlys
snmpInGenErrs
snmpInTotalReqVars
snmpInTotalSetVars
snmpInGetRequests
snmpInGetNexts
snmpInSetRequests
snmpInGetResponses
snmpInTraps
snmpOutTooBigs
snmpOutNoSuchNames
snmpOutBadValues
snmpOutGenErrs
snmpOutGetRequests
snmpOutGetNexts
snmpOutSetRequests
snmpOutTraps
snmpEnableAuthenTraps

RFC1317 RS-232 like group supported variables

rs232 MIB
rs232Number
rs232PortTable
rs232PortIndex
rs232PortType
rs232PortInSigNumber
rs232PortOutSigNumber
rs232PortInSpeed
rs232PortOutSpeed
rs232AsyncPortTable
rs232AsyncPortIndex
rs232AsyncPortBits
rs232AsyncPortStopBits
rs232AsyncPortParity
rs232InSigTable
rs232InSigPortIndex
rs232InSigName
rs232InSigState
rs232OutSigTable
rs232OutSigPortIndex
rs232OutSigName
rs232OutSigState

C
FAQ

FAQ 1 Why am I only able to use vfork ( ),and cannot use fork ( )?

Answer 1 µClinux only supports vfork ( ). It does not support fork ( ). Note that when using vfork ( ), the parent process will hang until the child process calls an exec group API, or exits.

FAQ 2 When using a pthread group API, why can I not use SIGUSR1 and SIGUSR2?

Answer 2 We cannot use the SIGUSR1 and SIGUSR2 signals since a pthread group API uses SIGUSR1 and SIGUSR2 to do a pthread control suspend, restart exit function. You will get the same result if you link the pthread. This means that you cannot use **`-1pthread`** to add an option to the linker.

FAQ 3 What is the correct format for linking to an API?

Answer 3 **`arm-elf-gcc –W1, -elf2flt`**
(In this example, the API converts elf format to flat format.)