

# UC-7101/7110/7112 User's Manual

---

Ninth Edition, April 2015

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



© 2015 Moxa Inc. All rights reserved.  
Reproduction without permission is prohibited.

# UC-7101/7110/7112 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 © 2015 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
	Package Checklist.....	1-2
	Product Features .....	1-3
	Product Specifications .....	1-3
	Hardware Specifications.....	1-3
	Software Specifications .....	1-4
<b>Chapter 2</b>	<b>Getting Started.....</b>	<b>2-1</b>
	Powering on the UC-7101/7110/7112.....	2-2
	Connecting the UC-7101/7110/7112 to a PC.....	2-2
	Console Port .....	2-2
	Telnet.....	2-3
	Configuring the Ethernet Interface .....	2-4
	Developing Your Applications .....	2-6
	Installing the UC-7101/7110/7112 Tool Chain .....	2-6
	Compiling Hello.c .....	2-7
	Uploading “Hello” to the UC-7101/7110/7112 .....	2-7
	Running “Hello” on the UC-7101/7110/7112 .....	2-8
	Sample Makefile Code .....	2-9
<b>Chapter 3</b>	<b>Software Package.....</b>	<b>3-1</b>
	UC-7101/7110/7112 Software Architecture.....	3-2
	Journaling Flash File System (JFFS2).....	3-3
	UC-7101/7110/7112 Software Package .....	3-4
<b>Chapter 4</b>	<b>Configuring UC-7101/7110/7112 .....</b>	<b>4-1</b>
	Enabling and Disabling Daemons.....	4-2
	Adding a Web Page.....	4-3
	IPTABLES .....	4-3
	NAT.....	4-6
	NAT Example.....	4-7
	Enabling NAT at Bootup.....	4-8
	Configuring Dial-in/Dial-out Service .....	4-8
	Dial-out Service.....	4-8
	Dial-in Service.....	4-8
	Configuring PPPoE.....	4-9
	How to Mount a Remote NFS Server .....	4-9
	Dynamic Driver Module Load / Unload .....	4-9
	Upgrading the Kernel.....	4-10
	Upgrading the Root File System & User Directory .....	4-11
	Loading Factory Defaults .....	4-12
	Autostarting User Applications on Bootup .....	4-12
	Checking the Kernel and Root File System Versions.....	4-12
<b>Chapter 5</b>	<b>UC-7101/7110/7112 Device API .....</b>	<b>5-1</b>
	RTC (Real Time Clock).....	5-2
	Buzzer .....	5-2
	UART Interface.....	5-2

WDT (Watch Dog Timer) .....	5-3
<b>Appendix A System Commands .....</b>	<b>A-1</b>
μClinix normal command utility collection .....	A-1
File manager .....	A-1
Editor .....	A-1
Network .....	A-2
Process .....	A-2
Other .....	A-2
Moxa Special Utilities .....	A-2
<b>Appendix B SNMP Agent with MIB II &amp; RS-232 Like Group .....</b>	<b>B-1</b>
<b>Appendix C FAQ.....</b>	<b>C-1</b>

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  $\mu$ Clinux 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  $\mu$ Clinux 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 compiler, 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  $\mu$ Clinux operating system

### UC-7110-LX

Mini RISC-based Ready-to-Run Embedded Computer with 2 Serial Ports, Dual Ethernet,  $\mu$ Clinux OS

### UC-7112-LX

Mini RISC-based Ready-to-Run Embedded Computer with 2 Serial Ports, Dual Ethernet, SD,  $\mu$ Clinux 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 serial 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)
- $\mu$ Clinix-ready communication platform
- Wall mounting installation
- Robust fanless design

## Product Specifications

### Hardware Specifications

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

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

## Software Specifications

<b>Kernel</b>	µClinux Kernel 2.6 Support for dynamic driver module load / unload
<b>Protocol Stack</b>	ARP, ICMP, IPV4, TCP, UDP, FTP, Telnet, SNMP V1/V2c, HTTP, CHAP, PAP, DHCP, NTP, NFS V2/V3, SMTP, Telnet, FTP, PPP, PPPoE
<b>File System</b>	JFFS2 for Kernel, Root File System (Read Only) and User Directory (Read / Write)
<b>Msh</b>	Minix shell command
<b>pppd</b>	Dial in/out over serial port daemon
<b>PPPoE</b>	Point-to-Point over Ethernet daemon
<b>snmpd</b>	SNMP V1/V2c Agent daemon
<b>busybox</b>	Linux normal command utility
<b>Tinylogin</b>	login and user manager utility
<b>Telnetd</b>	Telnet server daemon
<b>telnet</b>	Telnet client program
<b>inetd</b>	TCP server manager program
<b>ftpd</b>	FTP server program
<b>ftp</b>	FTP client program
<b>boa</b>	Web server daemon
<b>ntpd</b>	Network Time Protocol client utility
<b>Tool Chain</b>	
<b>Linux Tool Chain</b>	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**
- ❑ **Connecting the UC-7101/7110/7112 to a PC**
  - Console Port
  - Telnet
- ❑ **Configuring the Ethernet Interface**
- ❑ **Developing Your Applications**
  - Installing the UC-7101/7110/7112 Tool Chain
  - Compiling Hello.c
  - Uploading “Hello” to the UC-7101/7110/7112
  - Running “Hello” on the UC-7101/7110/7112
  - Sample Makefile Code

## 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

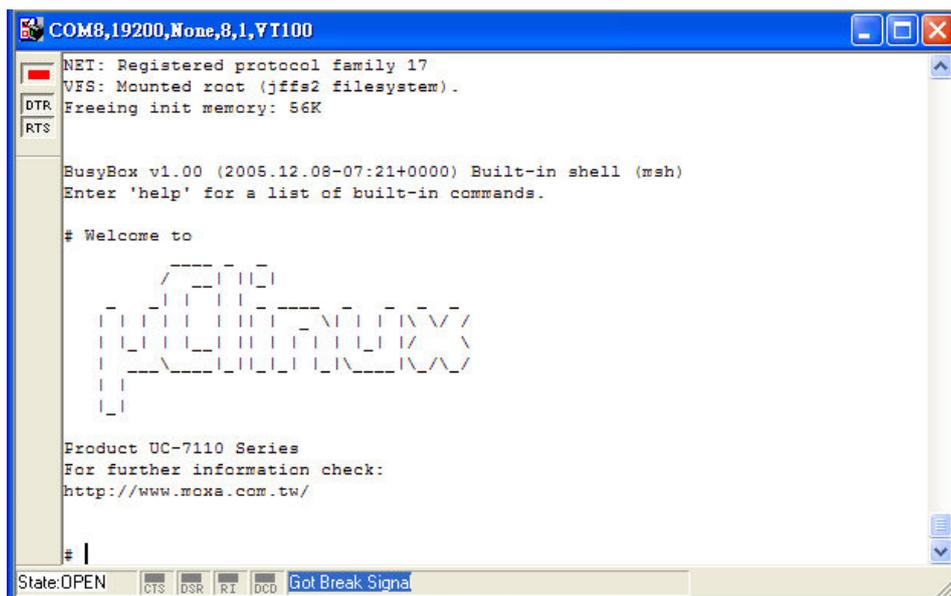
There are two ways to connect the UC-7101/7110/7112 to a PC.

### 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	
<b>Baudrate</b>	19200 bps
<b>Parity</b>	None
<b>Data bits</b>	8
<b>Stop bits</b>	1
<b>Flow Control</b>	None
<b>Terminal</b>	VT100

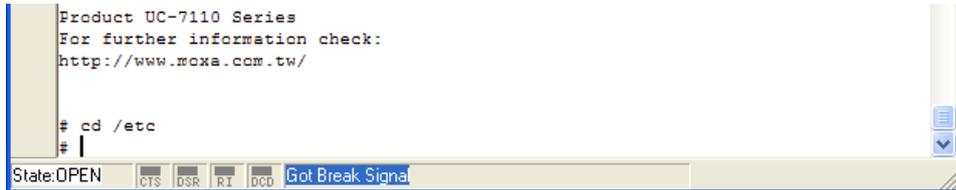




## Configuring the Ethernet Interface

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
# |
```

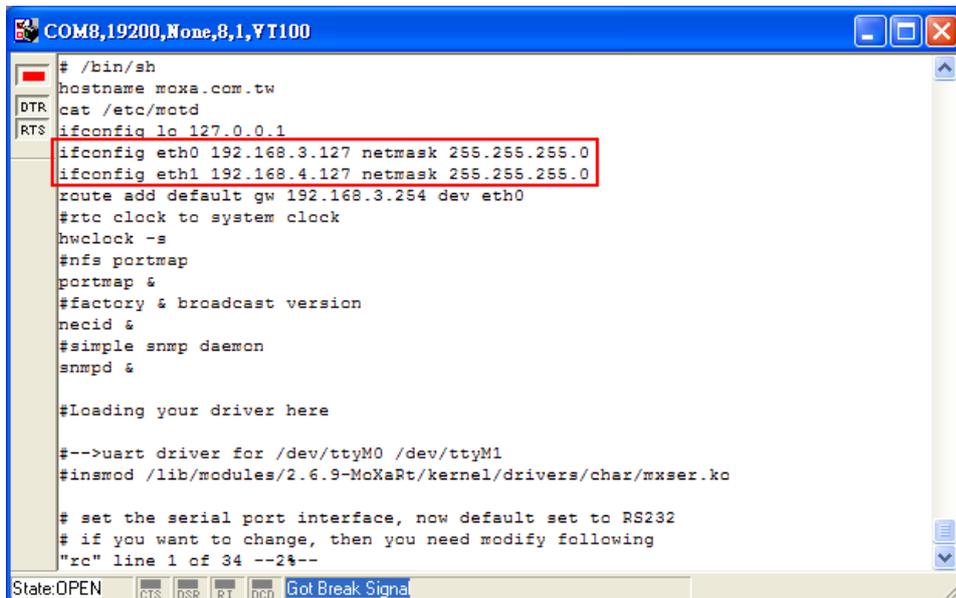
State:OPEN CTS DSR RI DCD Got Break Signal

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.



```
COM8,19200,None,8,1,YT100
# /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
mccid &
#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 --24--
```

State:OPEN CTS DSR RI DCD Got Break Signal

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.

```
COM8,19200,None,8,1,YT100
#!/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%--
State:OPEN  CTS  DSR  RI  DCD  Got Break Signal
```

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:**

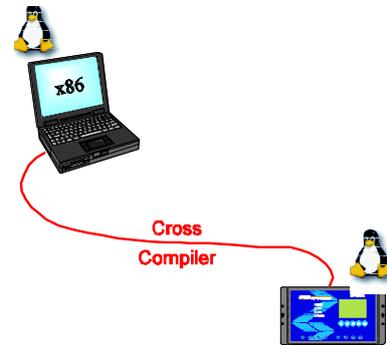
Download the program to the UC-7101/7110/7112 by FTP or NFS.

**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/elf2flt.ld -elf2flt  
[root@localhost hello]# ls -al  
total 116  
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  
-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.

```

c:\ Telnet 192.168.3.127
##### # ##### ##### # ##### #
##### # ##### ##### ##### ##### ##### #####

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          lk-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

---

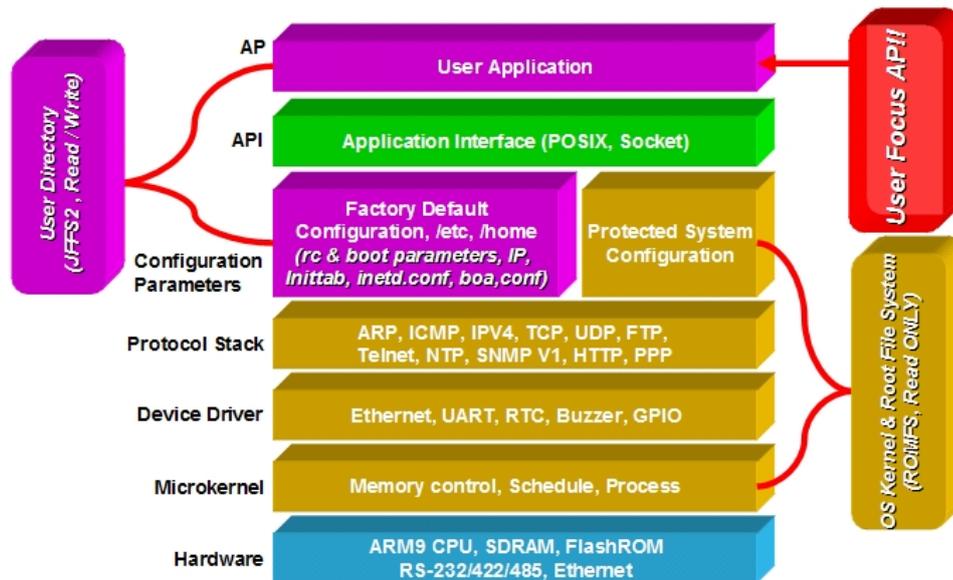
This chapter includes information about the software that is used with the UC-7101/7110/7112 Series of embedded computers.

This chapter covers the following topics:

- ❑ **UC-7101/7110/7112 Software Architecture**
  - Journaling Flash File System (JFFS2)
- ❑ **UC-7101/7110/7112 Software Package**

## UC-7101/7110/7112 Software Architecture

The pre-installed  $\mu$ Clinux operating system used by the UC-7101/7110/7112 follows the standard  $\mu$ Clinux 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](http://www.uClinux.org). In addition to the Standard POSIX API, device drivers for the buzzer, and UART for the serial ports are also included.



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.

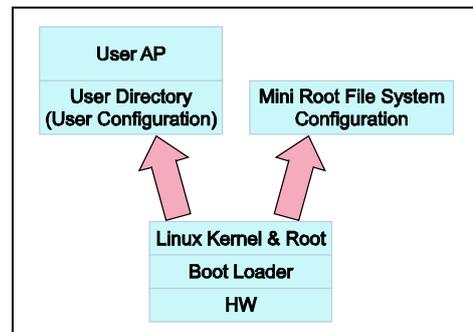
For most applications, users need to spend a lot time maintaining the operating system and modifying the system configuration. In order to save on the total cost of development and maintenance, the UC-7101/7110/7112 is specially design to partition a "User Directory" for storing the user's system configuration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

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
Kernel & Root File System	0x40000– 0x3ffff	4 M	Read ONLY <b>JFFS2</b>
User Directory	0x400000 – 0x7ffff	4 M – 256 K	Read / Write <b>JFFS2</b>

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	ttyf	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
discard	ttyp3	tty□
dhcpcd	ttyp2	
cpu	ttyp1	
busybox	ttyp0	
boa	ttyS0	
downramdisk	tty3	
upramdisk	tty0	
	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**
- Adding a Web Page**
- IPTABLES**
- NAT**
  - NAT Example
  - Enabling NAT at Bootup
- Configuring Dial-in/Dial-out Service**
  - Dial-out Service
  - Dial-in Service
- Configuring PPPoE**
- How to Mount a Remote NFS Server**
- Dynamic Driver Module Load / Unload**
- Upgrading the Kernel**
- Upgrading the Root File System & User Directory**
- Loading Factory Defaults**
- Autostarting User Applications on Bootup**
- Checking the Kernel and Root File System Versions**

## 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

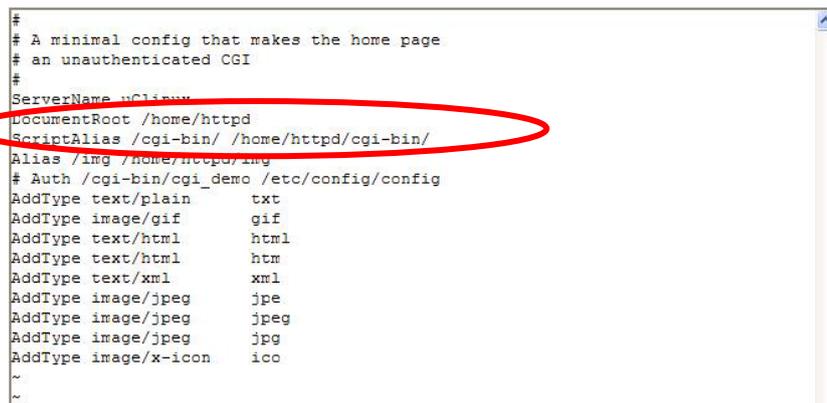
**Default Home Page address:**

*/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.

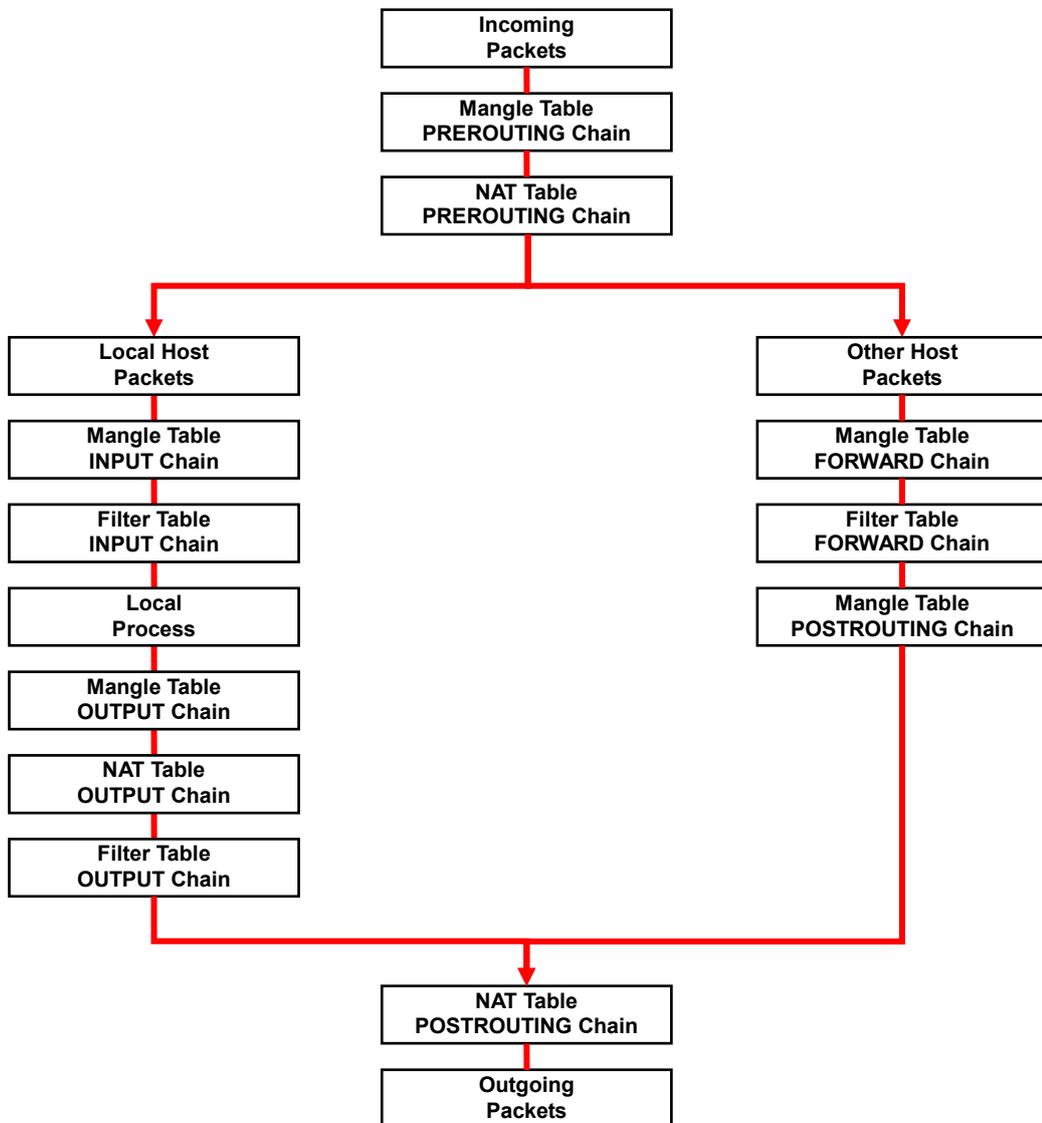
C. **Mangle Table**—includes two chains

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.



**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

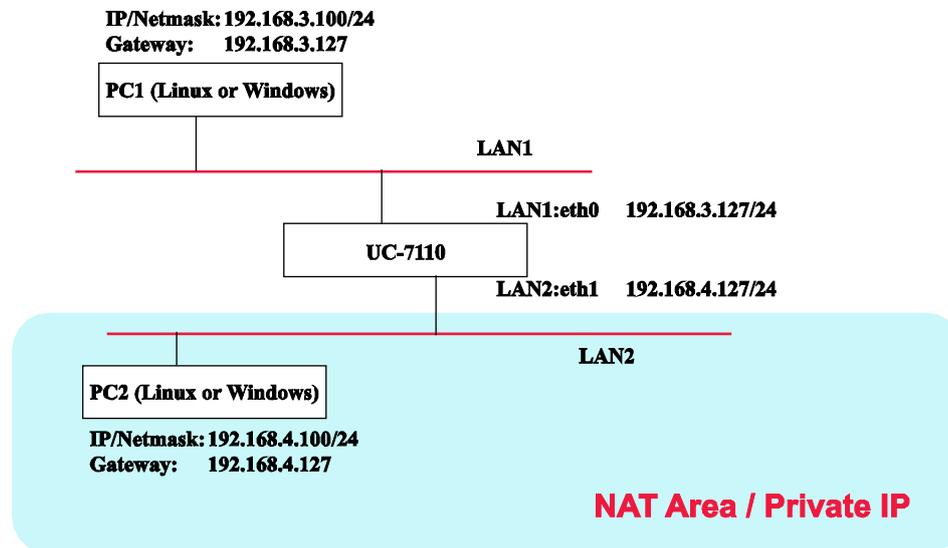
NAT (Network Address Translation) protocol translates IP addresses used on one network into different IP addresses used on another network. One network is designated the inside network and the other is the outside network. Typically, the UC-7101/7110/7112 connects several devices on a network and maps 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:  
 >/>rmmmod <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/2.6.19-uc1MoXaRt/kernel/drivers/mmc/core/mmc_core.ko
/lib/modules/2.6.19-uc1MoXaRt/kernel/drivers/mmc/card/mmc_block.ko
/lib/modules/2.6.19-uc1MoXaRt/kernel/drivers/mmc/host/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](http://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
```

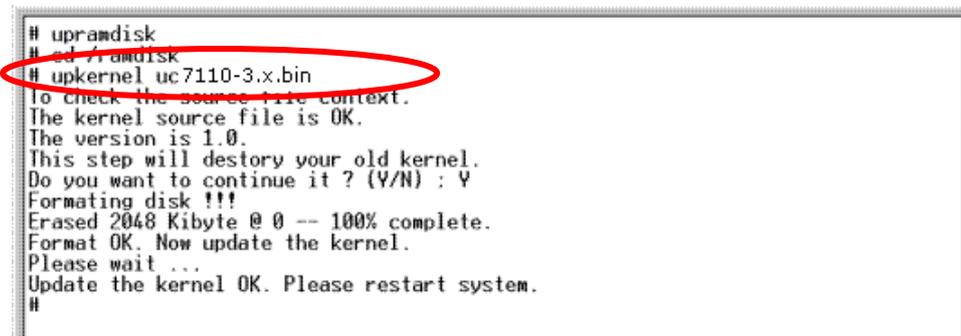
```
ftp> bin
```

```
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 destroy 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](http://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 destroy 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  $\mu$ Clinux 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  $\mu$ Clinux 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  $\mu$ Clinux 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 <moxadec.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_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:
    arm-elf-gcc -Wl, -elf2flt -o xxxx xxxx.c -lmoxalib

```

# A

## System Commands

---

### µClinux 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>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>du</b>	estimate file space usage
<b>mkdir</b>	make new directory
<b>rmdir</b>	delete directory
<b>head</b>	print the first 10 lines of each file to standard output
<b>tail</b>	print the last 10 lines of each file to standard output
<b>touch</b>	update the access and modification times of each file to the current time

#### Editor

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

## Network

<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>	tftp protocol
<b>telnet</b>	user interface to TELNET protocol
<b>ftp</b>	file transfer protocol
<b>iptables</b>	iptables command

## Process

<b>kill</b>	kill process
<b>killall</b>	kill process by name
<b>ps</b>	report process status
<b>sleep</b>	suspend command on time

## Other

<b>dmesg</b>	dump kernel log message
<b>stty</b>	set serial port
<b>mknod</b>	make device node
<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 / power off/on the server
<b>halt</b>	halt the server
<b>gzip, gunzip, zcat</b>	compress or expand files
<b>hostname</b>	show system's host name
<b>tar</b>	tar archiving utility

## Moxa Special Utilities

<b>cat /etc/version</b>	show user directory version
<b>upramdisk</b>	mount ramdisk
<b>downramdisk</b>	unmount ramdisk
<b>kversion</b>	show kernel version
<b>setinterface</b>	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 ipDefaultTTL ipInReceives ipInHdrErrors ipInAddrErrors ipForwDatagrams ipInUnknownProtos ipInDiscards ipInDelivers ipOutRequests ipOutDiscards ipOutNoRoutes ipReasmTimeout ipReasmReqds ipReasmFails ipFragOKs ipFragFails ipFragCreates ipAddrTable ipAdEntAddr ipAdEntIfIndex ipAdEntNetMask ipAdEntBcastAddr ipAdEntReasmMaxSize ipRouteTable ipRouteDest ipRouteIfIndex ipRouteMetric1 ipRouteMetric2 ipRouteMetric3 ipRouteMetric4 ipRouteNextHop ipRouteType ipRouteProto ipRouteAge ipRouteMask ipRouteMetric5 ipRouteInfo ipNetToMediaTable ipNetToMediaIfIndex ipNetToMediaPhysAddress ipNetToMediaNetAddress ipNetToMediaType ipRoutingDiscards	tcpRtoAlgorithm tcpRtoMin tcpRtoMax tcpMaxConn tcpActiveOpens tcpPassiveOpens tcpAttemptFails tcpEstabResets tcpCurrEstab tcpInSegs tcpOutSegs tcpRetransSegs tcpConnTable tcpConnState tcpConnLocalAddress tcpConnLocalPort tcpConnRemAddress tcpConnRemPort tcpInErrs tcpOutRsts	udpInDatagrams udpNoPorts udpInErrors udpOutDatagrams udpTable udpLocalAddress udpLocalPort

**snmp MIB**

snmpInPkts  
 snmpOutPkts  
 snmpInBadVersions  
 snmpInBadCommunityNames  
 snmpInBadCommunityUses  
 snmpInASNParseErrs  
 snmpInTooBig  
 snmpInNoSuchNames  
 snmpInBadValues  
 snmpInReadOnlys  
 snmpInGenErrs  
 snmpInTotalReqVars  
 snmpInTotalSetVars  
 snmpInGetRequests  
 snmpInGetNexts  
 snmpInSetRequests  
 snmpInGetResponses  
 snmpInTraps  
 snmpOutTooBig  
 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



## 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 `-lpthread` to add an option to the linker.

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

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