

# EM-1220-LX User's Manual

---

First Edition, August 2007

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



Moxa Systems Co., Ltd.

Tel: +886-2-2910-1230

Fax: +886-2-2910-1231

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

**MOXA Technical Support**

Worldwide: [support@moxa.com](mailto:support@moxa.com)

The Americas: [support@usa.moxa.com](mailto:support@usa.moxa.com)

# **EM-1220-LX User's Manual**

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

## **Copyright Notice**

Copyright © 2007 Moxa Systems Co., Ltd.  
All rights reserved.  
Reproduction without permission is prohibited.

## **Trademarks**

MOXA is a registered trademark of the Moxa Group.  
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.

# Table of Contents

<b>Chapter 1</b>	<b>Introduction .....</b>	<b>1-1</b>
	Overview .....	1-2
	Package Checklist .....	1-2
	Product Features .....	1-2
	Product Specifications .....	1-3
	Hardware Specifications .....	1-3
	Software Specifications .....	1-4
	Hardware Block Diagram .....	1-5
	Appearance .....	1-5
	Dimensions .....	1-6
	Installing the EM-1220-LX .....	1-7
	LED Indicators .....	1-7
	Wiring Requirements .....	1-7
	Connecting the Power .....	1-8
	Grounding the EM-1220 Development Kit .....	1-8
	Connecting Data Transmission Cables .....	1-9
	Connecting to the Network .....	1-9
	Connecting to a Serial Device .....	1-9
	Serial Console Port .....	1-10
	Internal SD Socket .....	1-10
	Additional Functions .....	1-10
	Reset Button .....	1-10
	Real-time Clock .....	1-11
<b>Chapter 2</b>	<b>Getting Started .....</b>	<b>2-1</b>
	Powering on the EM-1220-LX .....	2-2
	Connecting the EM-1220-LX to a PC .....	2-2
	Console Port .....	2-2
	Telnet .....	2-3
	Configuring the Ethernet Interface .....	2-4
	Installing a Secure Digital (SD) Memory Card .....	2-6
	Developing Your Applications .....	2-6
	Installing the EM-1220-LX Tool Chain .....	2-7
	Compiling Hello.c .....	2-9
	Uploading “Hello” to the EM-1220-LX .....	2-10
	Running “Hello” on the EM-1220-LX .....	2-10
	Make File Example Code .....	2-11
<b>Chapter 3</b>	<b>Software Package .....</b>	<b>3-1</b>
	EM-1220-LX Software Architecture .....	3-2
	Journaling Flash File System (JFFS2) .....	3-3
	EM-1220-LX Software Package .....	3-4
<b>Chapter 4</b>	<b>Configuring the EM-1220-LX .....</b>	<b>4-1</b>
	Enabling and Disabling Daemons .....	4-2
	Adding a Web Page .....	4-3
	IPTABLES .....	4-3
	NAT .....	4-7
	NAT Example .....	4-7

	Enabling NAT at Bootup .....	4-7
	Configuring Dial-in/Dial-out Service .....	4-8
	Dial-out Service .....	4-8
	Dial-in Service .....	4-8
	Configuring PPPoE .....	4-8
	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
	User Directory Backup—EM-1220-LX to PC .....	4-12
	Loading Factory Defaults .....	4-12
	Mirroring the Application Program and Configuration .....	4-13
	Autostarting User Applications on Bootup .....	4-13
	Checking the Kernel and Root File System Versions .....	4-13
<b>Chapter 5</b>	<b>EM-1220-LX Device API .....</b>	<b>5-1</b>
	RTC (Real-time Clock) .....	5-2
	Buzzer .....	5-2
	UART Interface .....	5-2
<b>Chapter 6</b>	<b>UC Finder .....</b>	<b>6-1</b>
	Windows UC Finder .....	6-2
	Linux UC Finder .....	6-5
<b>Appendix A</b>	<b>System Commands .....</b>	<b>A-1</b>
	busybox: $\mu$ Clinux 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</b>	<b>SNMP Agent with MIB II &amp; RS-232 Like Group .....</b>	<b>B-1</b>
<b>Appendix C</b>	<b>EM-1220-LX FAQ .....</b>	<b>C-1</b>

The Moxa EM-1220-LX Series of Mini RISC-based Ready-to-Run Embedded Computer features dual 10/100 Mbps Ethernet ports and two RS-232/422/485 serial ports in a built-in  $\mu$ Clinux ARM9 module. In addition, the EM-1220-LX supports an external SD socket to install SD memory card for storage expansion, offers high performance communication and unlimited storage in a super compact, palm-sized module. The EM-1220-LX is an ideal solution for embedded applications that use a lot of memory and must be housed in a small physical space without sacrificing performance.

In this chapter we cover the following topics:

- ☐ **Overview**
- ☐ **Package Checklist**
- ☐ **Product Features****Product Specifications**
  - Hardware Specifications
  - Software Specifications
- ☐ **Hardware Block Diagram**
- ☐ **Appearance**
- ☐ **Dimensions**
- ☐ **Installing the EM-1220-LX**
- ☐ **LED Indicators**
- ☐ **Wiring Requirements**
  - Connecting the Power
  - Grounding the EM-1220 Development Kit
- ☐ **Connecting Data Transmission Cables**
  - Connecting to the Network
  - Connecting to a Serial Device
  - Serial Console Port
- ☐ **Internal SD Socket**
- ☐ **Additional Functions**
  - Reset Button
  - Real-time Clock

## Overview

The EM-1220-LX Series of mini RISC-based communication platforms are ideal for your embedded applications. The EM-1220-LX comes with 2 RS-232/422/485 serial ports and dual 10/100 Mbps Ethernet LAN ports to provide users with a versatile communication platform.

The EM-1220-LX uses the MOXA ART ARM9 RISC CPU. Unlike the X86 CPU, which uses a CISC design, the ARM9's RISC design architecture and modern semiconductor technology provide the EM-1220-LX 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 applications. The dual LAN ports built into the ARM9 make the EM-1220-LX an ideal communication platform for simple data acquisition and protocol conversion applications, and the two 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 EM-1220-LX 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 EM-1220-LX's Flash memory.

## Package Checklist

### EM-1220-LX

Mini RISC-based ready-to-run embedded module with 2 serial ports, dual Ethernet, SD,  $\mu$ Clinux OS

The EM-1220 package includes the EM-1220 embedded module only. The EM-1220 Development Kit is available for evaluation purposes. The EM-1220 Development Kit package contains the following items:

- 1 EM-1220-LX embedded module
- 1 EM-1220-DK, the carrier board of the EM-1240 Development Kit
- Quick Installation Guide
- Document & Software CD
- Cross-over Ethernet cable
- Console port cable  
CBL-4PINDB9F-100: 4-pin header to DB9 (Female) cable, 100 cm
- Universal Power Adapter
- Product Warranty Statement

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

## Product Features

EM-1220-LX Series products have the following features:

- MOXA ART ARM9 32-bit 192 MHz processor
- On-board 16 MB RAM, 8 MB flash disk
- 2 software-selectable RS-232/422/485 serial ports
- Dual 10/100 Mbps Ethernet for network redundancy
- Ready-to-run  $\mu$ Clinux Kernel 2.6 platform
- SD signals supported for external SD socket connection

- Built-in RTC, buzzer
- 10 GPIOs reserved for system integration
- Credit card size design for easy integration at any field
- Full-function development kit for quick evaluation and application development
- -40 to 75°C wide temperature model available

## Product Specifications

### Hardware Specifications

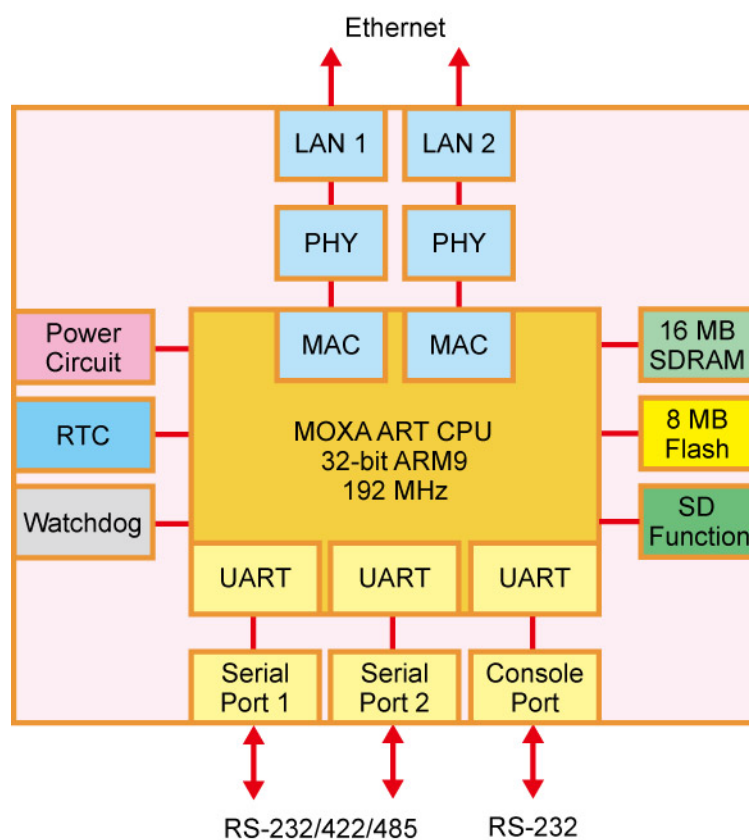
<b>Model</b>	<b>EM-1220 Embedded Module</b>
<b>CPU</b>	MOXA ART ARM9 32-bit 192 MHz processor
<b>RAM</b>	16 MB
<b>Flash</b>	8 MB
<b>LAN</b>	Auto-sensing 10/100 Mbps × 2
<b>LAN Protection</b>	Built-in 1.5 KV magnetic isolation
<b>Serial Ports</b>	<b>Two serial ports supporting RS-232/422/485 signals</b>
	<b>RS-232 signals:</b> TxD, RxD, DTR, DSR, RTS, CTS, DCD, GND
	<b>RS-422 signals:</b> TxD+, TxD-, RxD+, RxD-, GND
	<b>4 wire RS-485 signals:</b> TxD+, TxD-, RxD+, RxD-, GND
	<b>2 wire RS-485 signals:</b> Data+, Data-, GND
<b>Serial Console</b>	<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
<b>Storage Expansion</b>	<b>Speed:</b> 50 bps to 921.6 Kbps, supports Any Baudrate
	RS-232 × 1, TxD, RxD, GND
	SD signals for external Secure Digital (SD) socket connection
	GPIO × 10 (to enable GPIO, SD must be disabled.)
	GPIO × 10 (to enable GPIO, SD must be disabled.)
<b>GPIO</b>	
<b>Real-time Clock</b>	Yes
<b>Watchdog Timer</b>	Yes
<b>Buzzer Signals</b>	Buzzer signals reserved for external buzzer connection
<b>LED Signals</b>	LAN 10/100 × 2 on LAN Connector
	Reserve signals for the following LED connections:
	System Ready × 1
<b>Reset Signal</b>	Serial Port Status × 2 pairs (TxD, RxD, 2 for each pair)
<b>Power Input</b>	Reserve signal for external “Reset to Default” button connection
<b>Dimensions (W × L)</b>	Accepts external 3.3 VDC through pin header
<b>Operating temperature</b>	80 × 50 mm
<b>Storage temperature</b>	-10 to 60°C (14 to 141°F), 5 to 95% RH
	-40 to 75°C (-40 to 167°F) is optional for EM-1220-T model
	-20 to 80°C (-4 to 176°F), 5 to 95% RH
<b>Module Interface</b>	-40 to 85°C (-40 to 185°F) is optional for EM-1220-T model
	Two 2 × 17 pin-headers; pitch: 2.5 × 2.5 (mm)

## Software Specifications

<b>Kernel</b>	μClinux Kernel 2.6.9 Supports 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>ineted</b>	TCP server manager program
<b>ftpd</b>	FTP server program
<b>ftp</b>	FTP client program
<b>boa</b>	Web server daemon
<b>ntpdate</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
<b>Windows Tool Chain</b>	Arm-elf-gcc (V2.95.3): C/C++ PC Cross Compiler uClibc (V0.9.26): POSIX standard C library
<b>UC Finder</b>	UC's LAN IP broadcast searching utility for Windows and Linux

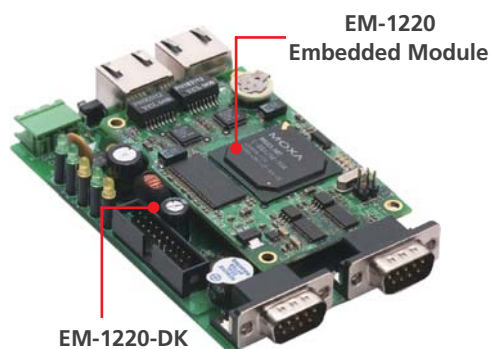


## Hardware Block Diagram



## Appearance

### EM-1220 Embedded Module + Development Kit

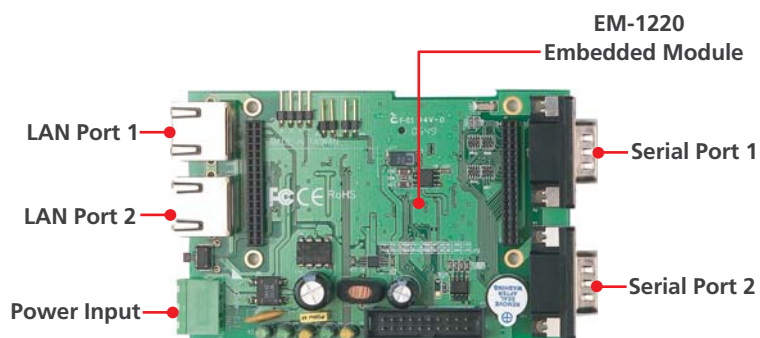
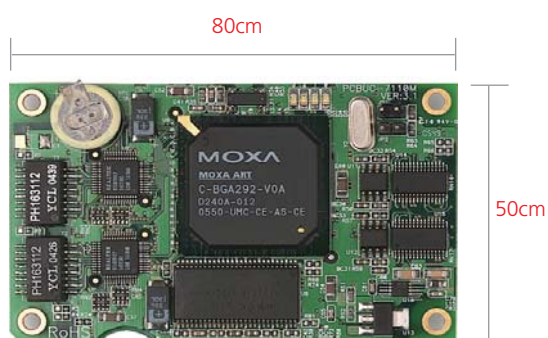


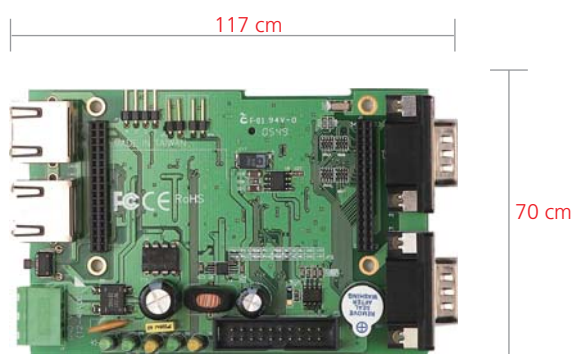
**EM-1220 Embedded Module**

MOXA ART ARM9 32-bit  
Communication Processor



onboard 16 MB RAM

**EM-1220 Development Kit****Dimensions****EM-1220 Embedded Module**

**EM-1220 Development Kit****Installing the EM-1220-LX**

If you would like to use the EM-1220 Embedded Module and the EM-1240 Development Kit, insert the EM-1220 Embedded Module vertically onto the Development Kit. Note that the pin marked “J2” on the Embedded Module must be matched with the pin marked “J2” on the Development Kit; and the Pin marked “J1” on the Embedded Module must be matched with the Pin marked “J1” on the Development Kit. Be careful when inserting the module to avoid damaging the product.

**LED Indicators**

The following table explains the function of the five LED indicators located on the EM-1220-LX's Development Kit.

LED Name	LED Color	LED Function
Ready	Green	Power is on and functioning normally.
P1/P2 (Tx)	Green	Serial port 1 or 2 is transmitting data.
	Off	Serial port 1 or 2 is not transmitting data.
P1/P2 (Rx)	Yellow	Serial port 1 or 2 is receiving data.
	Off	Serial port 1 or 2 is not receiving data.

**Wiring Requirements**

This section describes how to connect the EM-1220-LX to serial devices.

You should pay attention to the following common safety precautions before proceeding with the installation of any electronic device:

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

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

- Use the type of signal transmitted through a wire to determine which wires should be kept separate. The rule of thumb is that wiring that shares similar electrical characteristics can be bundled together.
- Keep input wiring and output wiring separate.

- It is advisable to label the wiring to all devices in the system.

**ATTENTION****Safety First!**

Be sure to disconnect the power cord before installing and/or wiring your EM-1220-LX.

**Wiring Caution!**

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

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

**Temperature Caution!**

Be careful when handling the EM-1220-LX. When plugged in, the EM-1220-LX's internal components generate heat, and consequently the outer casing may feel hot to the touch.

## Connecting the Power

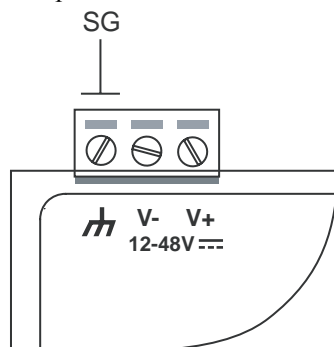
Connect the “live-wire” end of the 12-48 VDC power adaptor to the terminal block of EM-1220-LX Development Kit. If the power is properly supplied, the “Ready” LED will glow a solid green after a 25 to 30 second delay.

## Grounding the EM-1220 Development Kit

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

**ATTENTION**

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



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

## Connecting Data Transmission Cables

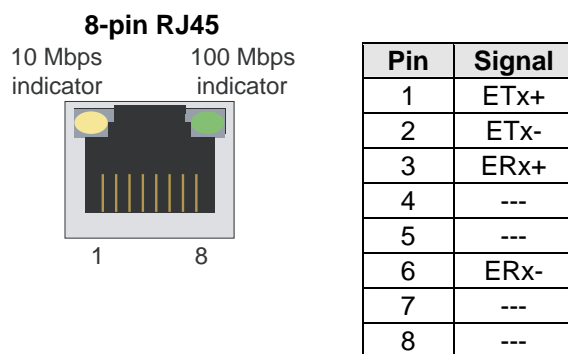
This section describes how to connect the EM-1220 Development Kit to the network, serial devices, and serial COM terminal.

### Connecting to the Network

Connect one end of the Ethernet cable to the 10/100M Ethernet port of EM-1220 Development Kit and the other end of the cable to the Ethernet network. If the cable is properly connected, the EM-1220 Development Kit will indicate a valid connection to the Ethernet in the following ways:

- The top-right LED on the connector glows a solid green when connected to a 100 Mbps Ethernet network.
- The top-left LED on the connector glows a solid orange when connected to a 10 Mbps Ethernet network.
- The LEDs will flash when Ethernet packets are being transmitted or received.

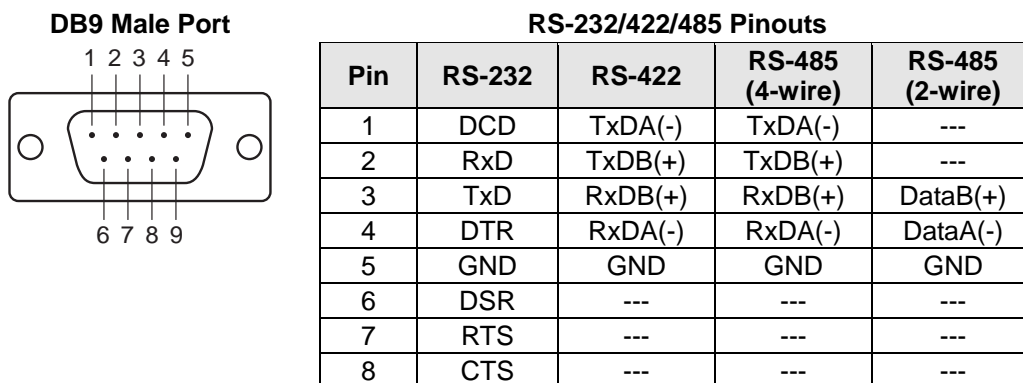
The 10/100 Mbps Ethernet LAN 1 and LAN 2 ports use 8-pin RJ45 connectors. Pinouts for these ports are given in the following diagram.



### Connecting to a Serial Device

Connect the serial cable between the EM-1220 Development Kit and the serial device(s).





Serial ports P1 and P2 on EM-1220-DK use male DB9 connectors, and can be configured for RS-232/422/485 by software. The pin assignments are shown in the following table:



## Serial Console Port

The serial console port on the EM-1220-DK is a 4-pin pin-header RS-232 port. It is designed for serial console terminals, which are useful for identifying the EM-1220-LX boot up message.

**Serial Console Port & Pinouts**

		Pin	Signal
4		1	TxD
3		2	RxD
2		3	NC
1		4	GND

**Serial Console Cable**



## Internal SD Socket

The EM-1220 Development Kit provides an internal SD socket for storage expansion. It allows users to plug in a Secure Digital (SD) memory card compliant with the SD 1.0 standard for up to 1 GB of additional memory space. The internal SD socket is located on the backside of the EM-1220 Development Kit's bottom board; at the right of the EM-1220 Development Kit, Plug the SD card into the socket directly and remember to press the SD card first if you want to take it out. Please note that the SD function shares the same chipset with the DIO. If you would like to enable the SD function, the DIO must be disabled. If you would like to enable the DIO, the SD function must be disabled.

## Additional Functions

### Reset Button

Press the **Reset** button on the EM-1220-DK continuously for at least 5 seconds to load the factory default configuration. After the factory default configuration has been loaded, the system will reboot automatically. We recommend that you only use this function if the software is not working properly and you want to load factory default settings. To reset an embedded Linux system, always use the software reboot command `/>reboot` to protect the integrity of data being transmitted or processed. The Reset button is not designed to hard reboot the EM-1220 Development Kit.



#### ATTENTION

Resetting to factory defaults will not format the user directory and erase all of the user's data. Loading factory defaults will only load the configuration file. The files in the EM-1220-LX that will be replaced include:

- /etc/boa.conf
- /etc/hosts
- /etc/inittab
- /etc/password
- /etc/ramfs.img
- /etc/resolv.conf
- /etc/version
- /etc/group
- /etc/inetd.conf
- /etc/motd

- k. /etc/protocols
- l. /etc/rc
- m. /etc/services
- n. /home/httpd/index.html

**ATTENTION**

This function only takes effect when the user directory is working correctly. If the user directory has crashed, the kernel will automatically load the factory defaults.

## Real-time Clock

The EM-1220-LX's real time clock is powered by a lithium battery. We strongly recommend that you do not replace the lithium battery without the help of Moxa's support team. If the battery needs to be changed, contact the Moxa RMA service team for RMA service.

**ATTENTION**

The battery may explode if replaced by an incorrect type. To avoid this potential danger, always be sure to use the correct type of battery.

# 2

## Getting Started

---

In this chapter, we explain the basic procedure for getting the EM-1220-LX connected and ready for your needs.

In this chapter we cover the following topics:

- ❑ **Powering on the EM-1220-LX**
- ❑ **Connecting the EM-1220-LX to a PC**
  - Console Port
  - Telnet
- ❑ **Configuring the Ethernet Interface**
- ❑ **Installing a Secure Digital (SD) Memory Card**
- ❑ **Developing Your Applications**
  - Installing the EM-1220-LX Tool Chain
  - Compiling Hello.c
  - Uploading “Hello” to the EM-1220-LX
  - Running “Hello” on the EM-1220-LX
  - Make File Example Code



## Powering on the EM-1220-LX

Connect the SG wire to the Shielded Contact located on the upper left corner of the EM-1220-LX, and then power on the EM-1220-LX 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 EM-1220-LX 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 EM-1220-LX to a PC

There are two ways to connect the EM-1220-LX to a PC.

### Console Port

The serial console port offers users a convenient means of connecting to the EM-1220-LX. This method is particularly useful when using the EM-1220-LX 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 following window 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

```

aaa - 超級終端機
檔案(F) 編輯(E) 檢視(V) 呼叫(C) 轉送(T) 說明(H)
[Icons]
NET: Registered protocol family 1
NET: Registered protocol family 17
VFS: Mounted root (jffs2 filesystem).
Freeing init memory: 56K

BusyBox v1.00 (2006.03.31-09:54+0000) Built-in shell (msh)
Enter 'help' for a list of built-in commands.

# Welcome to

      _ _ _ _ _
     / / / / /
    / / / / /
   / / / / /
  / / / / /
 / / / / /
/_/_/_/_/_

Product EM-1220 Serial
For further information check:
http://www.moxa.com.tw/

連線 00:00:21 自動偵測 19200 8-N-1 SCROLL CAPS NUM 關 F1-F12

```



## Configuring the Ethernet Interface

In this section, we use the serial console to explain how to modify the EM-1220-LX's network settings.

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

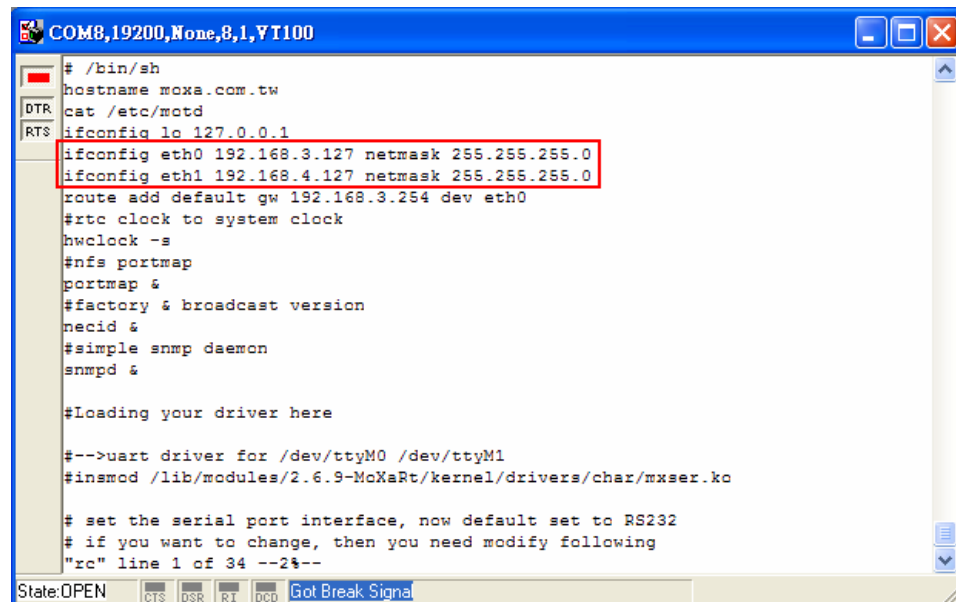


Type the command `vi rc` to use the VI Editor to edit the configuration file. The IP addresses for the EM-1220-LX's LAN1 and LAN2 are given as

```
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.



- You may also configure the EM-1220-LX 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 EM-1220-LX 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
ncid &
#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 --24--
```

- Issue the vi “write” command to save the file, and then reboot. Since the EM-1220-LX 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 NOT however update the “rc” file in the EM-1220-LX’s flash memory, so the next time you reboot, the IP address will revert to its previous value.

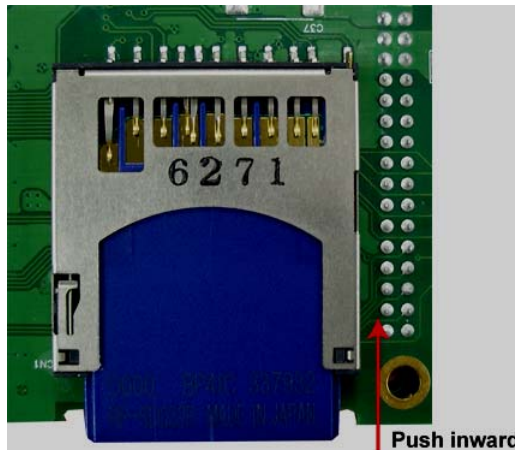
## Installing a Secure Digital (SD) Memory Card

The EM-1220 Development Kit provides an internal SD socket for storage expansion. To access this socket, perform the following steps to install the SD memory card.

**Step 1:** Find the exact location of the SD socket.

**Step 2:** Insert the SD card into the socket. Make sure the card is situated correctly.

**Step 3:** Push the SD card inward.



**Step 4:** Before using the SD card, check the /etc/rc file to ensure that the driver module for the SD card control is loaded. The loading sequence should be as follows:

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

**Step 5:** To take out the SD memory card, press the SD card again. The card will pop out part of the way, after which you can pull it out directly.

## Developing Your Applications

**Step 1:**

Connect the EM-1220 Development Kit to a Linux PC.

**Step 2:**

Install Tool Chain (GNU Cross Compiler & uClibc).

**Step 3:**

Configure cross compiler and uClibc environment variables.

**Step 4:**

Code & compile your program.

**Step 5:**

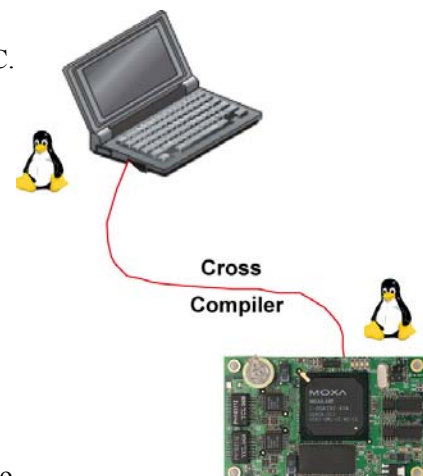
Download program to the EM-1220-LX via FTP or NFS.

**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 distribute the code to additional EM-1220-LX units.



## Installing the EM-1220-LX Tool Chain

### Linux

The PC must have the Linux Operating System pre-installed to install the EM-1220-LX 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 EM-1220-LX Tool Chain can be found on the EM-1220-LX Document & Software 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 from the root to install the compilers, linkers, and libraries in the `/usr/local` directory:

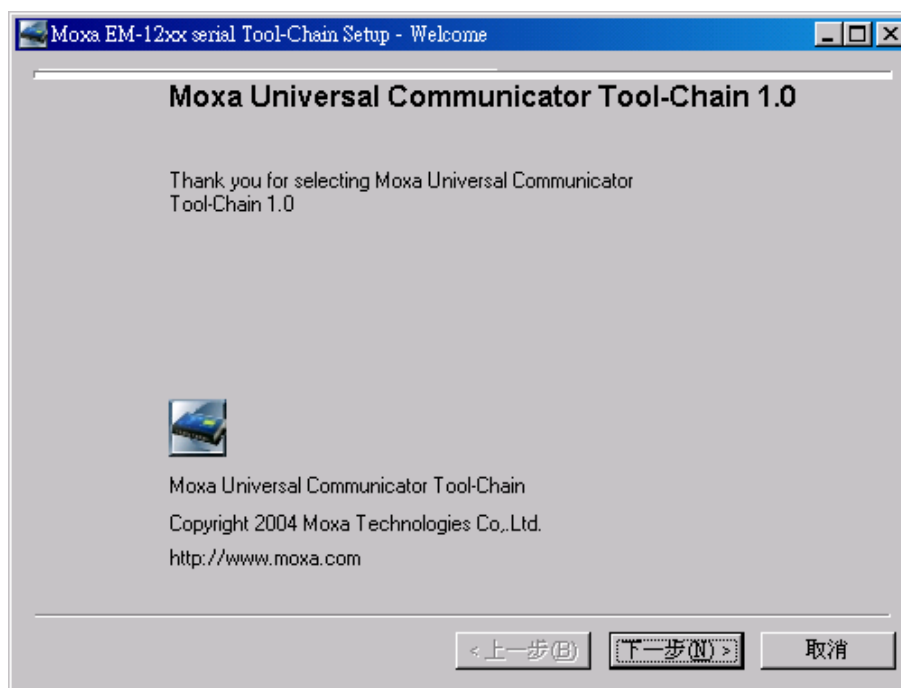
```
#sh /mnt/cdrom/tool-chain/linux/installer/arm-elf-moxa-toolchain-1.1.sh
```

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

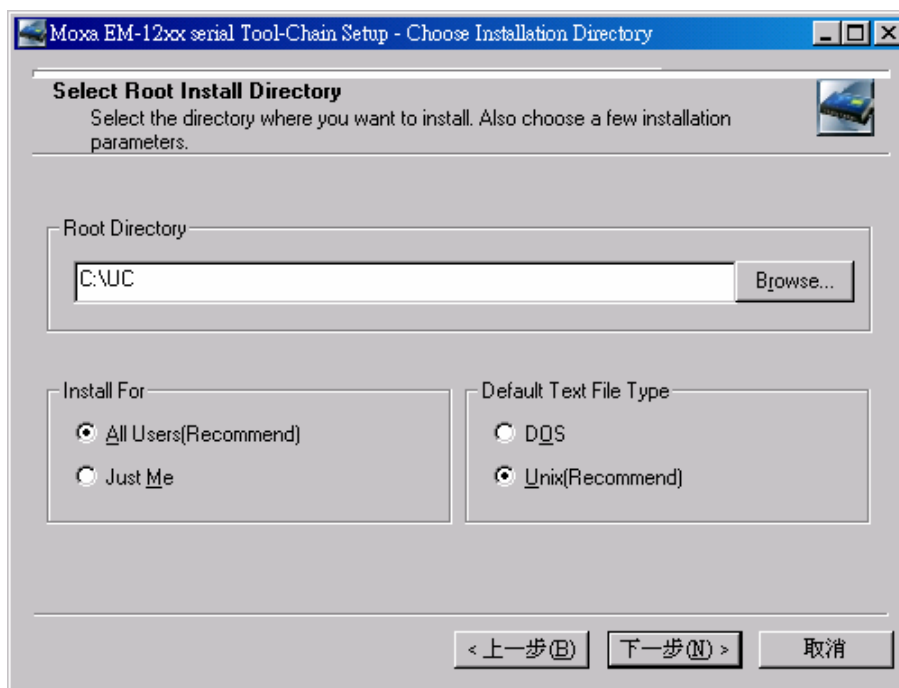
### Windows

In addition to the Linux Tool Chain, the Windows Tool Chain for the EM-1220-LX is on the official EM-1220-LX Document & Software CD. Please refer to the following installation procedure to install the EM-1220-LX Windows Tool Chain.

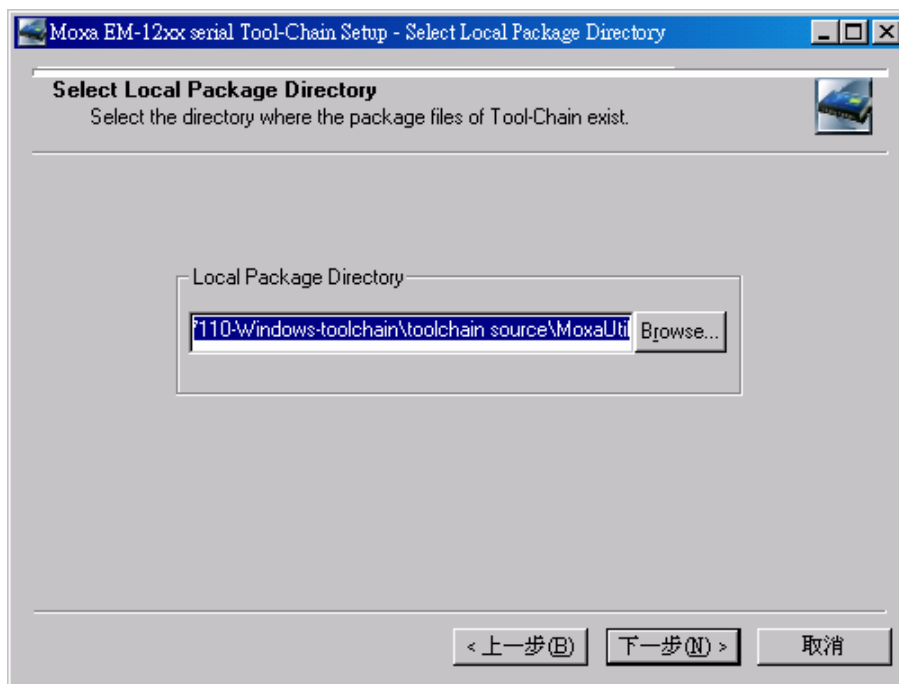
**Step 1:** Double click `tool-chain\windows\setup.exe` on the EM-1220-LX CD to begin the installation, and then click **Next**.



**Step 2:** Click **Browse...** to select your installation location. The default location is under C:\UC.



**Step 3:** Click **Next** to select the local package files directory, and then click **Browse...** to select where your installation source is. The default path is the location of the **setup.exe** file.



**Step 4:** Click **Next** to begin the package installation. A progress bar will indicate the MD5 status of each software package. Click **Next** to complete the installation.

**ATTENTION**

The Tool Chain can be downloaded from Moxa's website. To do this, navigate to the EM-1220-LX 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/arm-elf/bin:$PATH
```

The EM-1220-LX 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 then issue the command

```
#make
```

to compile Hello.c. Finally, execute the program to generate **hello** and **hello.gdb**.

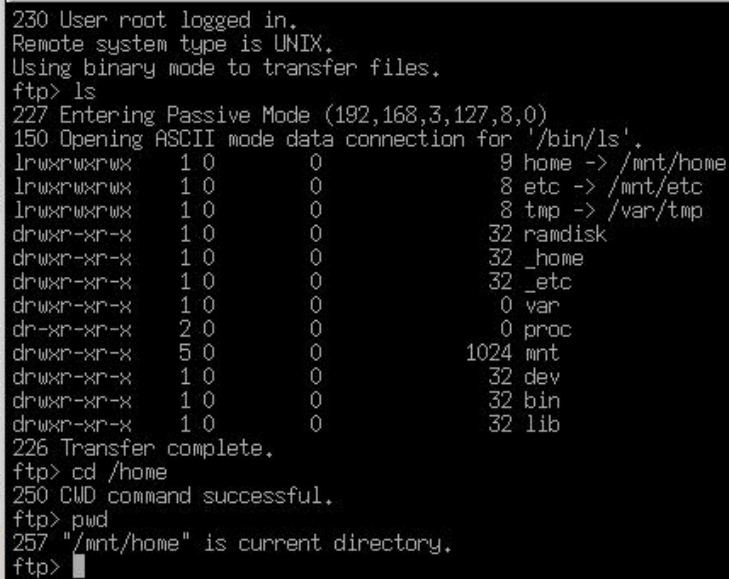
```
[root@localhost hello]# ls -al
total 20
drwxr-xr-x  2 root  root    4096 Aug 18 10:58 .
drwxr-xr-x  5 root  root    4096 Aug  5 10:34 ..
-rw-rw-rw-  1 root  root    1498 Jan  6 2004 elf2flt.ld
-rw-rw-rw-  1 root  root     74 Jan  6 2004 hello.c
-rw-rw-rw-  1 root  root     875 Jan  6 2004 Makefile
[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 10:59 .
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 EM-1220-LX

To use FTP to upload **hello** to the EM-1220-LX, 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
```

A screenshot of a terminal window showing an FTP session. The text is as follows:

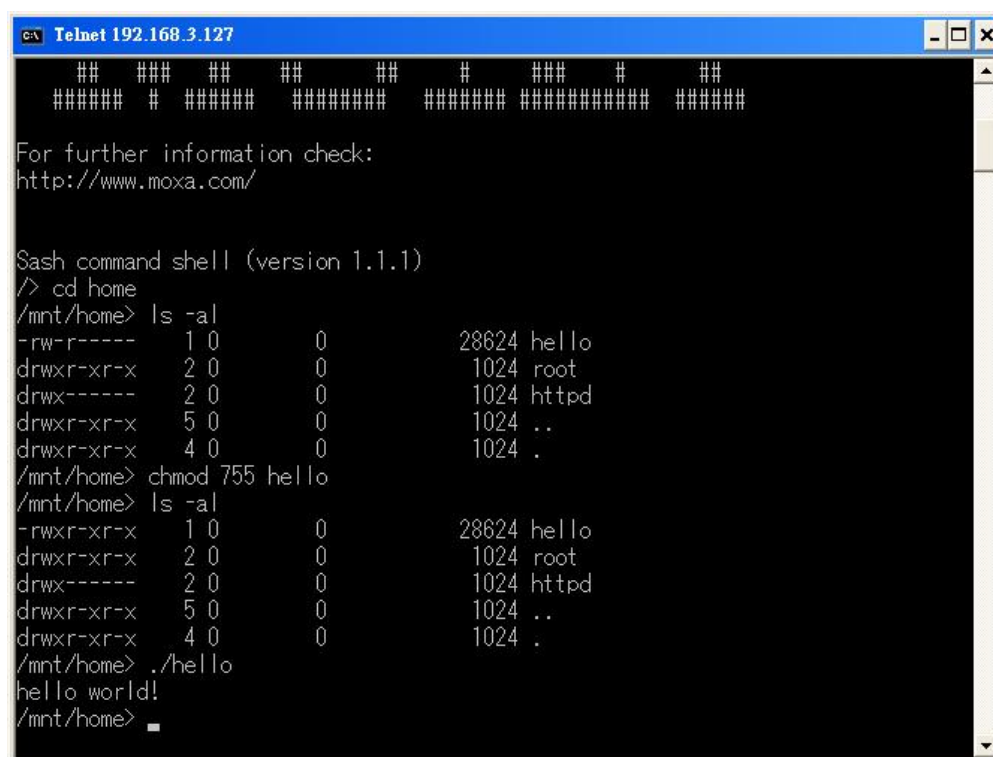
```
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,3,127,8,0)
150 Opening ASCII mode data connection for '/bin/ls'.
lrwxrwxrwx 1 0 0 9 home -> /mnt/home
lrwxrwxrwx 1 0 0 8 etc -> /mnt/etc
lrwxrwxrwx 1 0 0 8 tmp -> /var/tmp
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 EM-1220-LX

To run the **hello** program issue the following commands on the EM-1220-LX:

```
# chmod 755 hello
# ./hello
```

The words “hello world” are 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 in use:

# **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 to the Flash ROM. To free up some memory, use the console cable to connect to the EM-1220-LX's serial console terminal, and then delete files from the Flash ROM.

## Make File Example Code

The following Make File example codes are copied from the Hello example on the EM-1220-LX'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

---

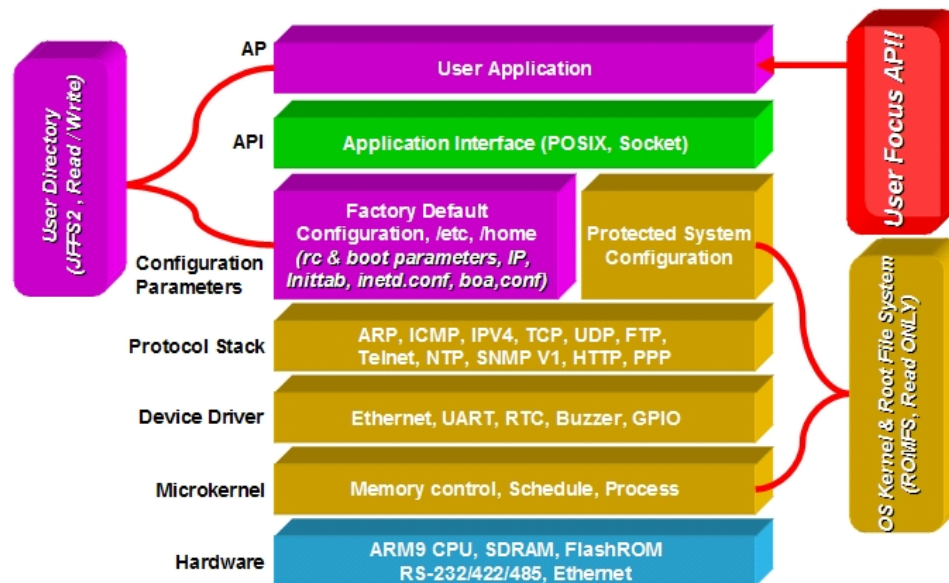
This chapter includes information about the software that is used with EM-1220-LX Series products.

In this chapter, we cover the following topics:

- ❑ **EM-1220-LX Software Architecture**
  - Journaling Flash File System (JFFS2)
- ❑ **EM-1220-LX Software Package**

## EM-1220-LX Software Architecture

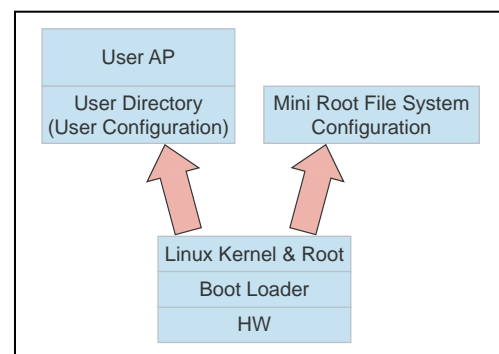
The pre-installed  $\mu$ Clinux Operating System used by the EM-1220-LX follows the standard  $\mu$ Clinux architecture. This means that programs following the POSIX standard are easily ported to the EM-1220-LX with the GNU Tool Chain provided by [www.uClinux.org](http://www.uClinux.org). In addition to the Standard POSIX API, device drivers for the buzzer and UART for the serial ports are also included.



The EM-1220-LX's Flash ROM is divided into smaller partitions, including the Boot Loader, Linux Kernel & Root (/) File System Image, and User Directory partitions.

For most applications, users often spend a lot of time maintaining the operating system and modifying the system configuration. In order to save on the total cost of development and maintenance, the EM-1220-LX is specially designed to partition a "User Directory" for storing the user's system configuration parameters.

The EM-1220-LX has a built-in mechanism that prevents system crashes to help preserve 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 system configuration parameters via 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 EM-1220-LX 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– 0x3ffff	4 M	Read ONLY <b>JFFS2</b>
User Directory	0x400000 – 0x7ffff	4 M – 256 K	Read / Write <b>JFFS2</b>

Developers should store their own programs only to partitions `/etc`, `/home`, `/tmp`, and `/usr/bin`. In addition, executable files should be stored in `/usr/bin`, as doing so 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 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 later version of JFFS, provides improved wear-leveling and garbage-collection performance, improved RAM footprint and response to system-memory pressure, improved concurrency and support for suspending flash erases, marking of bad sectors with continued use of the remaining good sectors (thus enhancing 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 preclude the loss of data. The file system will remain in a consistent state across power failures, and will always be mountable. However, if the board is powered down during a write, then the incomplete write will be rolled back on the next boot. Any writes that were already completed will not be affected.

**Additional information about JFFS2 is available at**

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

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

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

## EM-1220-LX Software Package

Bin	dev
upkernel	mtdblock1
passwd -> tinylogin	mtdr1
login -> tinylogin	mtd1
tinylogin	mtdblock0
telnetd	mtdr0
snmpd	mtd0
mail	cum1
sh	cum0
routed	ttyM1
netstat	ttyM0
arp	urandom
chat	random
pppd	zero
portmap	ttypf
ntpdate	ttype
necid	ttypd
eraseall	ttypec
kversion	ttypb
init	ttypa
expand	ttyp9
inetd	ttyp8
hwclock	ttyp7
ftpd	ttyp6
ftp	ttyp5
mke2fs	ttyp4
e2fsck	ttyp3
discard	ttyp2
dhcpcd	ttyp1
cpu	ttyp0
busybox	ttyS0
boa	tty3
bf	tty2
backupfs	tty1
downramdisk	tty0
upramdisk	rom1
	rom0
	ptypf
	ptype
	ptypd
	ptypc
	ptypb
	ptypa
	ptyp9
	ptyp8
	ptyp7
	ptyp6
	ptyp5
	ptyp4
	ptyp3
	ptyp2
	ptyp1

Bin	dev
	ptyp0 ppp pio rtc ram1 ram0 null kmem mem cua0 console tty



## Configuring the EM-1220-LX

---

In this chapter, we describe how to configure the EM-1220-LX Series products.

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**
- ☐ **User Directory Backup—EM-1220-LX to PC**
- ☐ **Loading Factory Defaults**
- ☐ **Mirroring the Application Program and Configuration**
- ☐ **Autostarting User Applications on Bootup**
- ☐ **Checking the Kernel and Root File System Versions**

## Enabling and Disabling Daemons

The following daemons are enabled when the EM-1220-LX boots 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

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

#### How to enable/disable /etc/inittab www server

- Edit the file '/etc/inittab'
- 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 EM-1220-LX plays 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 recommended public NTP server list.

#### How to update the system time periodically with Network Time Protocol

- 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.
- 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

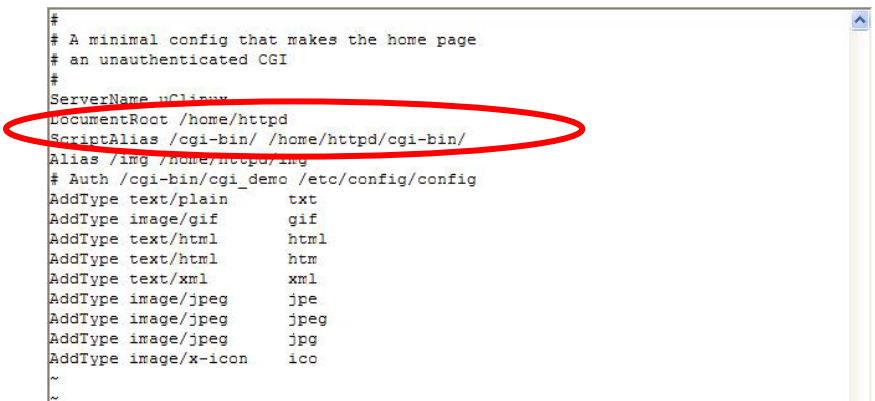
**Default Home Page address:**

`/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 EM-1220-LX supports 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.

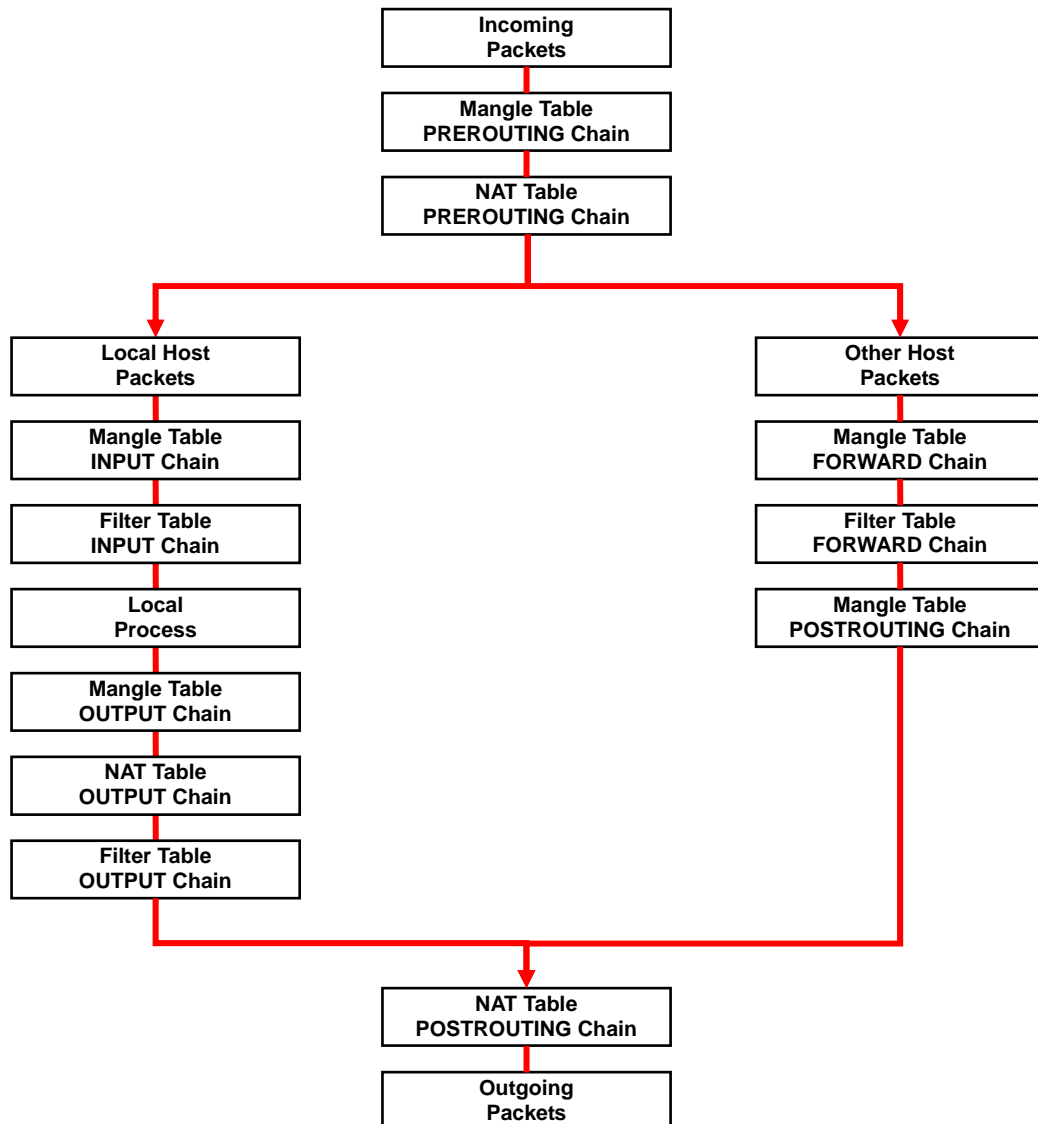
C. **Mangle Table**—includes two chains

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.



The EM-1240-LX supports the following sub-modules. Be sure to use the module that matches your application. You must load a module before you can use it. Use the **insmod** command to load a module.

x_tables	xt_conntrack	xt_helper	xt_mark
xt_pkttype	xt_state	xt_tcpudp	xt_CLASSIFY
xt_dccp	xt_length	xt_MARK	xt_quota
xt_statistic	xt_comment	xt_dscp	xt_multiport
xt_realm	xt_string	xt_connbytes	xt_esp
xt_mac	xt_NFQUEUE	xt_sctp	xt_tcpmss
xt_limit			
arptable_filter	ip_nat	iptable_raw	ipt_hashlimit
ipt_owner	ipt_time	arp_tables	ip_nat_snmp_basic
ip_tables	ipt_iprange	ipt_recent	ipt_tos
arpt_mangle	ip_nat_tftp	ipt_addrtype	ipt_layer7
ipt_REDIRECT	ipt_TOS	ip_nat_amanda	iptable_filter
ipt_ah	ipt_LOG	ipt_REJECT	ipt_ttl
ip_nat_ftp	iptable_mangle	ipt_ecn	ipt_MASQUERADE
ipt_SAME	ipt_TTL	ip_nat_irc	iptable_nat
ipt_ECN	ipt_NETMAP	ipt_TCPMSS	ipt_ULONG

**NOTE** The EM-1220-LX does NOT support IPV6 and ipchains.

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

**NOTE** 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 EM-1240-LX.  
`OUTPUT:` For locally-generated packets.  
`FORWARD:` For packets routed out through the EM-1240-LX.  
`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] [-A] [INPUT, OUTPUT, FORWARD] [-i interface] [-p tcp, udp, icmp,
all] [-s IP/network] [--sport ports] [-d IP/network] [--dport ports] -j [ACCEPT, DROP]
```

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

**Examples:**

Example 1: Accept all packets from lo interface.

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

Example 2: Accept TCP packets from 192.168.0.1.

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

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

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

Example 4: Drop TCP packets from 192.168.1.25.

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

Example 5: Drop TCP packets addressed for port 21.

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

Example 6: Accept TCP packets from 192.168.0.24 to the EM-1240-LX'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 EM-1240-LX's port 25.

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

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 EM-1220-LX 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: <a href="http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html">http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html</a>
------	--

## 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. First load the following device drivers:
  - `x_tables.ko`
  - `xt_multiport.ko`
  - `xt_MARK.ko`
  - `xt_tcpudp.ko`
  - `ip_tables.ko`
  - `ip_nat.ko`
  - `iptables_nat.ko`
  - `ipt_MASQUERADE.ko`
2. `#echo 1 > /proc/sys/net/ipv4/ip_forward`
3. `#iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.3.127`  
or
4. `#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 EM-1220-LX boots 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 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 permits 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 EM-1220-LX only supports NFS (Network File System) clients. Users can open NFS service on a Linux PC to enable the EM-1220-LX to push data to it. The EM-1220-LX 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 EM-1220-LX 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

Besides supporting traditional static drivers, the EM-1220-LX also supports the dynamic driver module load / unload mechanism. It allows users 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 EM-1240-LX, the factory default is to load the UART driver **mxser.ko**. An additional driver module for controlling the SD/MMC memory card is loaded for the EM-1240-LX. The location and file name for these driver modules is given below.

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 EM-1220-LX kernel is **em1220-1.x.bin**, which can be downloaded from [www.moxa.com](http://www.moxa.com). You must first download this file to your PC, and then use Console Terminal or Telnet Console to copy the file to the EM-1220-LX.

You can save this file to the EM-1220-LX'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
```

As illustrated below, after executing “upramdisk”, you may use “mount” to determine if the new ramdisk was created successfully or not.

```
# 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)
# |
```

Use the following command to navigate to the device node:

```
/>cd ramdisk
```

Use the built-in FTP client to download the file **em1220-1.x.bin** from the PC.

```
/ramdisk>ftp <destination PC's IP>
Login Name: xxxx
Login Password: xxxx
ftp> bin
ftp> get em1220-1.x.bin
```

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

```
/ramdisk>upkernel em1220-1.x.bin
/ramdisk>reboot
```

```
# upramdisk
# cd /ramdisk
# upkernel em1220-1.0.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 EM-1220-LX uses JFFS2 for the root file system and user directory. By default, the root file system is pre-set to READ only. The EM-1220-LX provides a read/write user's directory in the JFFS2 file system. By using this user's directory, the system configuration file and user's program can be stored on this disk.

Search the EM-1220-LX's CD-ROM for the latest user directory file, or download the file from [www.moxa.com](http://www.moxa.com). The format is **em1220-1.x.dsk**. You must download this file to a PC first, and then use Console Terminal or Telnet Console to copy the file to the EM-1220-LX.

You can save this file to the EM-1220-LX's RAM disk, and then upgrade the user directory. The following is a step-by-step example.

To enable the RAM disk, use the following commands.

```
/>upramdisk
/>cd ramdisk
```

Use the built-in FTP client to download the em1220-1.x.dsk file from the PC.

```
/ramdisk>ftp <destination PC's IP>
Login Name: xxxx
Login Password: xxxx
ftp> bin
ftp> get em1220-1.x.dsk
ftp>quit
/ramdisk>upkernel /ramdisk/em1220-1.x.dsk
/reboot
```

```
# upkernel em1220-1.0.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.
# -
```

## User Directory Backup—EM-1220-LX to PC

To enable the RAM disk, use the following commands:

```
/>upramdisk
/>cd ramdisk
```

Use the **backupfs** command to back up the file system.

```
/ramdisk>backupfs /ramdisk/usrdisk-backup
/> backupfs /ramdisk/usrdisk-backup
Sync the file system.
Now backup the user directory. Please wait ...
Backup the user directory OK.
/>
```

The file system will be backed up. Use ftp commands to transfer the usrdisk-backup to the FTP server on the PC.

```
/> cd /ramdisk
/ramdisk> ls -al
----- 1 0 0 4194304 usrdisk-backup
drwxr-xr-x 1 0 0 32 ..
drwxr-xr-x 2 0 0 1024 .
/ramdisk> ftp 192.168.3.11
Connected to 192.168.3.11.
220 TYPSoft FTP Server 1.10 ready...
Name (192.168.3.11:root): root
331 Password required for root.
Password:
230 User root logged in.
+bin
ftp> put usrdisk-backup
local: usrdisk-backup remote: usrdisk-backup
200 Port command successful.
150 Opening data connection for usrdisk-backup.
226 File received complete
4195224 bytes sent in 37 secs (113 Kbytes/sec)
ftp>
```

## 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 System & 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 “ldfactory” from the Telnet console to restore the factory defaults.

## Mirroring the Application Program and Configuration

For some applications, you may need to “Mirror” (or sometimes “Ghost”) one EM-1220-LX’s user directory, and duplicate it to other EM-1220-LX units.

To do this, use the following recommended procedure:

1. Back up the user directory to a PC.  
(Refer to the previous topic **User Directory Backup—EM-1220-LX to a PC.**)

**Hint:**

```
/ramdisk>backupfs /ramdisk/<user defined file name>
```

2. Download the backed up user directory to the other EM-1220-LX.  
(Refer to the previous topic **Upgrading the User Directory**)

**Hint:**

```
/ramdisk>bf /ramdisk/<User directory file name>
```

## Autostarting User Applications on Bootup

Edit the `/etc/rc` file by adding your application program. E.g.,

```
/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:

To check the kernel version:

```
/>kversion
```

To check the root file system (firmware) version of the EM-1220-LX, type:

```
/>fsversion
```

You may also check the user directory version of the EM-1220-LX by using the following command:

```
/>cat /etc/version
```

## EM-1220-LX Device API

---

In this chapter, we discuss the Device API for the EM-1220-LX Series. We introduce the APIs for the following functions:

- ❑ **RTC (Real-time Clock)**
- ❑ **Buzzer**
- ❑ **UART Interface**

## RTC (Real-time Clock)

The device node is located at `/dev/rtc`. The EM-1220-LX supports  $\mu$ 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 EM-1220-LX supports  $\mu$ Clinux standard buzzer control. The EM-1220-LX'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 the modem tty device node is located at `/dev/com0 ... com1`. The EM-1220-LX Series supports  $\mu$ Clinux standard termios control. The Moxa UART Device API supports configurations ttyM0 to ttyM1, as RS-232/422/485. To use these functions, after the Tool Chain package is installed, 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.

The EM-1220-LX comes with a UC Finder utility, which has the sole purpose of searching the LAN or intranet for EM-1220-LX units.

For most applications, it is not easy to remember the IP addresses of embedded computers connected to the LAN. This is especially true for problem solving and testing in the field. The UC Finder utility broadcasts messages over the LAN to search for IP addresses of embedded computers connected to the LAN. UC Finder searches for the class of MAC addresses assigned to MOXA's embedded computers. The EM-1220-LX supports the GUI-style Windows UC Finder, and a command line utility for Linux environments.

In this chapter, we discuss the following UC Finder topics:

- ❑ **Windows UC Finder**
- ❑ **Linux UC Finder**



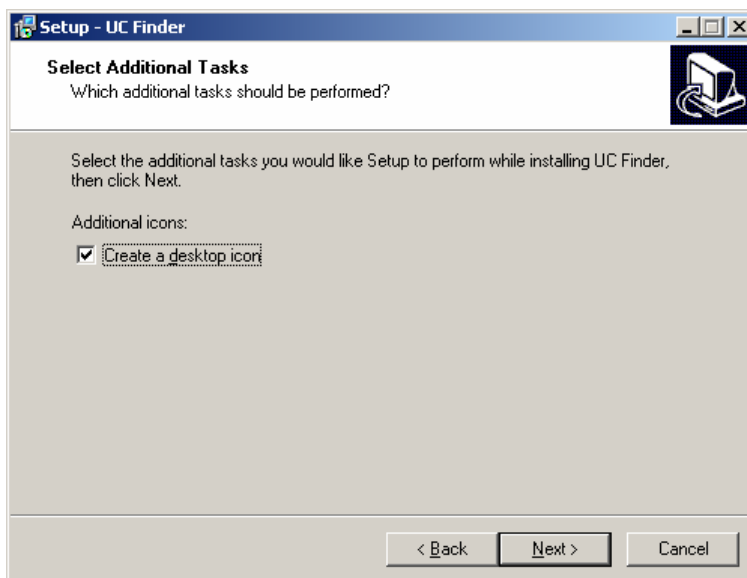
## Windows UC Finder

The following steps describe how to install UC Finder on a Windows PC.

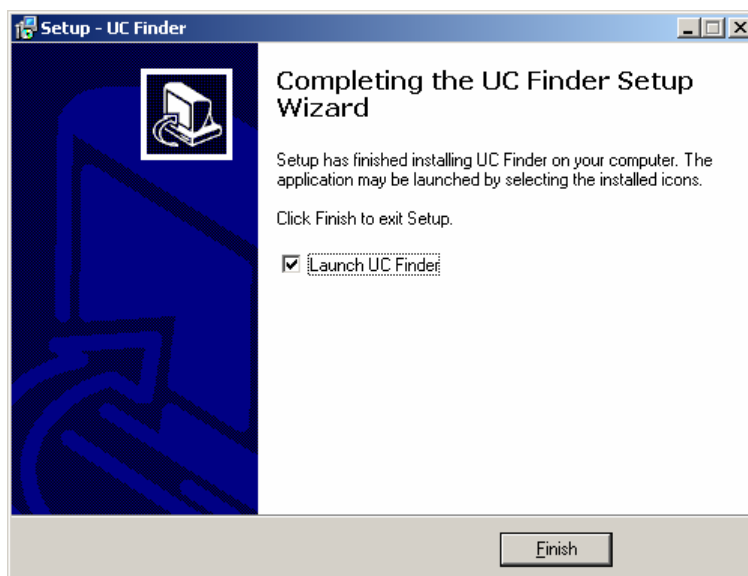
1. Double click the UC Finder installation program, **Setup.exe**, to start the installation.
2. When the **Welcome to the UC Finder Setup Wizard** window opens, click **Next** to continue.



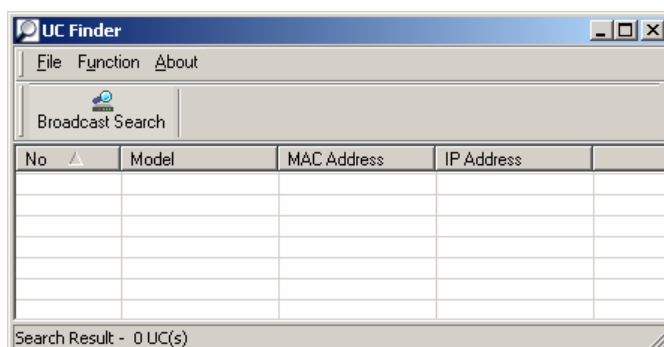
3. Select the **Create a desktop icon** option, and then click **Next** to continue.



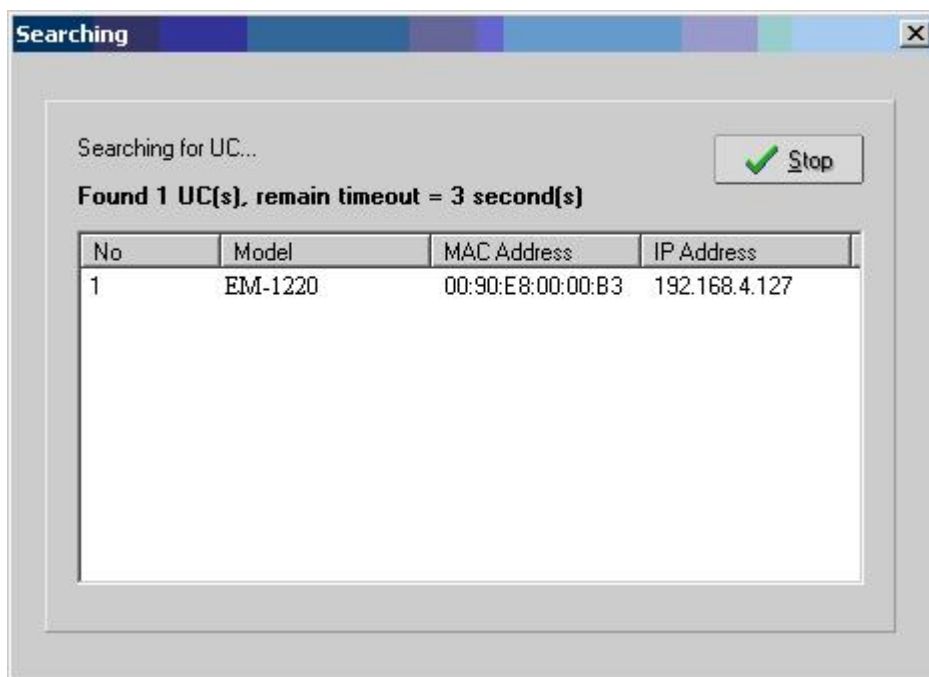
4. Select the **Launch UC Finder** option, to use UC Finder immediately after the installation has finished, and then click **Next** to complete the installation.



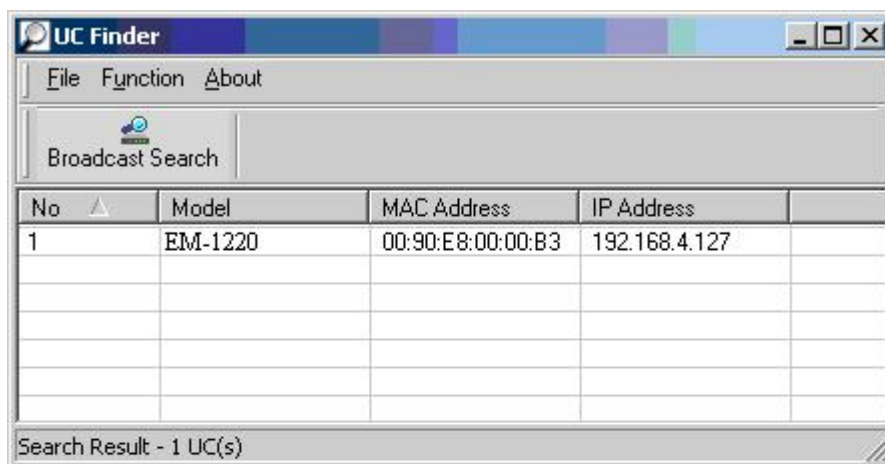
5. When the UC Finder window opens, click **Broadcast Search** to search for all MOXA embedded computers connected to the LAN.



6. The **Searching** window will show the MOXA embedded computers that have been located. You can click **Stop** as soon as the embedded computer you are looking for is listed.



7. When the search is complete, the Broadcast Search window closes, and the **Model**, **MAC Address**, and **IP Address** of all embedded computers that were located will be listed in the UC Finder window.



**ATTENTION**

UC Finder is designed solely to find IP addresses of networked MOXA embedded computers. UC Finder cannot be used to configure embedded computers over the network. If you need to configure an embedded computer's IP address or other parameters, connect to the embedded computer's console utility by Telnet (over the network) or serial console (using the serial console cable that came with the product).

The next time you need to run UC Finder, double click the UC Finder icon located on your PC's desktop to launch this utility.



## Linux UC Finder

To use the Linux **ucfinder** utility, copy **ucfinder** from the CD-ROM to your Linux PC, and then use the following command to start **ucfinder**. The **ucfinder** utility will automatically broadcast a message over your LAN network to find the IP addresses of all UC's connected to the LAN.

**#./ucfinder**

# A

## System Commands

---

### busybox: $\mu$ Clinux 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>tracerout</b>        | trace route                                    |
| <b>tftp</b>             | tftp protocol                                  |
| <b>telnet</b>           | user interface to TELNET protocol              |
| <b>ftp</b>              | file transfer protocol                         |
| <b>iptables-restore</b> | restore iptables configuration file to network |
| <b>iptables</b>         | iptables command                               |
| <b>iptables-save</b>    | save recent iptables configuration to file     |

## 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>backupfs</b>         | backup file system (user directory) |
| <b>bf</b>               | build file system (user directory)  |
| <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 EM-1220-LX 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 EM-1220-LX.

The full SNMP object ID of EM-1220-LX is **.iso.3.6.1.4.1.8691.12.1220**.

Note: The EM-1220-LX does not support SNMP trap.

### RFC1213 MIB-II supported SNMP variables:

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

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



| <b>snmp MIB</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| snmpInPkts<br>snmpOutPkts<br>snmpInBadVersions<br>snmpInBadCommunityNames<br>snmpInBadCommunityUses<br>snmpInASNParsingErrors<br>snmpInTooBigs<br>snmpInNoSuchNames<br>snmpInBadValues<br>snmpInReadOnly<br>snmpInGenErrors<br>snmpInTotalReqVars<br>snmpInTotalSetVars<br>snmpInGetRequests<br>snmpInGetNexts<br>snmpInSetRequests<br>snmpInGetResponses<br>snmpInTraps<br>snmpOutTooBigs<br>snmpOutNoSuchNames<br>snmpOutBadValues<br>snmpOutGenErrors<br>snmpOutGetRequests<br>snmpOutGetNexts<br>snmpOutSetRequests<br>snmpOutTraps<br>snmpEnableAuthenTraps |

**RFC1317 RS-232 like group supported variables**

| <b>rs232 MIB</b>                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rs232Number<br>rs232PortTable<br>rs232PortIndex<br>rs232PortType<br>rs232PortInSigNumber<br>rs232PortOutSigNumber<br>rs232PortInSpeed<br>rs232PortOutSpeed<br>rs232AsyncPortTable<br>rs232AsyncPortIndex<br>rs232AsyncPortBits<br>rs232AsyncPortStopBits<br>rs232AsyncPortParity<br>rs232InSigTable<br>rs232InSigPortIndex<br>rs232InSigName<br>rs232InSigState<br>rs232OutSigTable<br>rs232OutSigPortIndex<br>rs232OutSigName<br>rs232OutSigState |



## EM-1220-LX FAQ

---

**FAQ 1** Why can I only use `vfork()`, and am not able to 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't I use `SIGUSR1` and `SIGUSR2`?

**Answer 2** Since a pthread group API uses `SIGUSR1` and `SIGUSR2` to do a pthread control suspend and restart the exit function, we cannot use the `SIGUSR1` and `SIGUSR2` signals. 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.)