Moxa C Programmable RTU Controllers User's Manual

Third Edition, June 2012

www.moxa.com/product

Moxa C Programmable RTU Controllers User's Manual

Copyright Notice

Trademarks

Disclaimer

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

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

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

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

Technical Support Contact Information

www.moxa.com/support

Moxa Americas

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

Moxa Europe

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

Moxa China (Shanghai office)

Toll-free:	800-820-5036
Tel:	+86-21-5258-9955
Fax:	+86-21-5258-5505

Moxa Asia-Pacific

Tel:	+886-2-8919-1230
Fax:	+886-2-8919-1231

Table of Contents

1

Introduction

The Moxa C programmable RTU controller is a system with 1 or 2 10/100 Mbps Ethernet ports, an internal SD socket, 1 or 2 RS-232/422/485 serial ports, built-in or user-selectable modular I/Os, and pre-installed operating system, depending on the specific model. The Moxa C programmable RTU controller offers high-performance communication capability, and high storage capacity in one compact and rugged box. It is the ideal solution for stand-alone, remote monitoring applications in hard-to-wire environments, and applications that require a great deal of memory space, advanced processing power, integration with physical I/O channels, and other peripherals.

The following topics are covered in this chapter:

- **Overview**
- **Software Architecture**
  - Journaling Flash File System (JFFS2)
  - Software Package
- **Available Products**
  - ioPAC 8020-C Series
  - ioLogik W5348-C Series
- **Product Specifications**
  - ioPAC 8020 Series
  - ioLogik W5348 Series

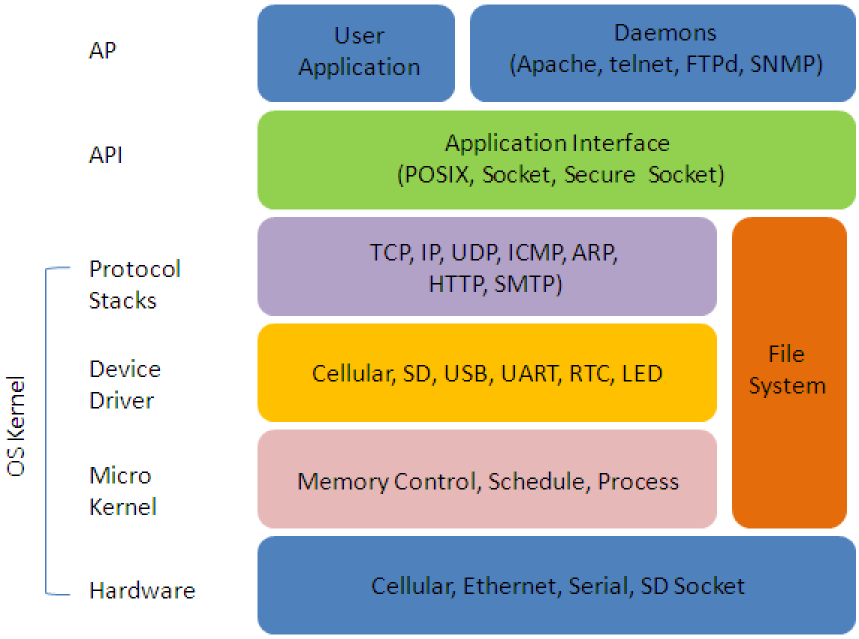
Overview

The Moxa C programmable RTU controller uses the Moxa ART RISC CPU. The RISC architecture and advanced semiconductor technology provide the RTU controller with a powerful computing engine and communication functions without generating a lot of heat. A 32 MB NOR Flash ROM, 64 MB on-board SDRAM, and an SD socket provide enough memory for you to install application software and store data directly on the RTU controller. In addition, the cellular modem, the Ethernet switch ports, and 3-in-1 serial interfaces create the best flexibility to communicate with field devices and with the central host, making the Moxa C programmable RTU controller ideal for remote data acquisition and industrial control applications.

The pre-installed operating system (OS) provides an open operating system for your software program development. Software that runs on desktop PCs can be easily exported to the RTU controller with a cross compiler. The software development package also provides versatile API functions, such as I/Os and communication control, SCADA connection, alarms, and Modbus communication.

Software Architecture

The operating system that is pre-installed in the Moxa C programmable RTU controller follows a standard Linux-based architecture; the program porting can be done with the Tool Chain provided by Moxa.


AP
User Application
Daemons (Apache, telnet, FTPd, SNMP)
API
Application Interface (POSIX, Socket, Secure Socket)
OS Kernel
Protocol Stacks
TCP, IP, UDP, ICMP, ARP, HTTP, SMTP)
Device Driver
Cellular, SD, USB, UART, RTC, LED
Micro Kernel
Memory Control, Schedule, Process
File System
Hardware
Cellular, Ethernet, Serial, SD Socket

The built-in Flash ROM is partitioned into **Boot Loader, Kernel, Root File System,** and **User directory** partitions.

In order to prevent user applications from crashing the **Root File System**, the Moxa C programmable RTU controller uses a unique **Root File System with Protected Configuration** for emergency use. This **Root File System** comes with serial and Ethernet communication capability for users to load the **Factory Default Image** file. User settings and applications are saved in the user directory.

To improve system reliability, the Moxa C programmable RTU controller has a built-in mechanism that prevents the system from crashing. When the kernel boots up, the RTU will mount the root file system in read only mode, and then enable services and daemons. At the same time, the kernel will start searching for system configuration parameters via ***rc*** or ***inittab***.

Normally, the kernel uses the Root File System to boot up the system. The Root File System is protected, and cannot be changed by users, which creates a safe zone for users.

For more information about the memory map and programming, refer to Chapter 6, *Programmer's Guide*.

Journaling Flash File System (JFFS2)

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

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

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

The key features of JFFS2 are:

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

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

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

Software Package

Boot Loader	Moxa Boot Loader (v1.2)
Kernel	Linux 2.6.9
Protocol Stack	ARP, PPP, CHAP, PAP, IPv4, ICMP, TCP, UDP, DHCP, FTP, HTTP, NTP, NFS, SMTP, SSH 1.0/2.0, SSL, Telnet, PPPoE, OpenVPN
File System	JFFS2, NFS, Ext2, Ext3, VFAT/FAT
OS shell command	Bash
Busybox	Linux normal command utility collection
Utilities	
tinylogin	Login and user manager utility
telnet	Telnet client program
ftp	FTP client program
smtpclient	Email utility
scp	Secure file transfer Client Program
Daemons	
pppd	Dial in/out over serial port daemon
telnetd	Telnet server daemon

inetd	TCP server manager program
ftpd	FTP server daemon
apache	Web server daemon
sshd	Secure shell server
openvpn	Virtual private network
openssl	Open SSL
Linux Tool Chain	
Gcc (V3.3.2)	C/C++ PC Cross Compiler
GDB (V5.3)	Source Level Debug Server
Glibc(V2.2.5)	POSIX standard C library

Available Products

ioPAC 8020-C Series

ioPAC 8020-5-M12-C-T: ioPAC 8020 modular RTU controller with dual M12 Ethernet LAN ports and 5 I/O slots, C programming capability, -40 to 75°C operating temperature

ioPAC 8020-5-RJ45-C-T: ioPAC 8020 modular RTU controller with dual RJ45 Ethernet LAN ports and 5 I/O slots, C programming capability, -40 to 75°C operating temperature

ioPAC 8020-9-M12-C-T: ioPAC 8020 modular RTU controller with dual M12 Ethernet LAN ports and 9 I/O slots, C programming capability, -40 to 75°C operating temperature

ioPAC 8020-9-RJ45-C-T: ioPAC 8020 modular RTU controller with dual RJ45 Ethernet LAN ports and 9 I/O slots, C programming capability, -40 to 75°C operating temperature

RM-1602-T: ioPAC I/O module with 16 digital inputs, 24 VDC sink/source type, -40 to 75°C operating temperature

RM-1050-T: ioPAC I/O module with 10 digital inputs, 110 VDC, -40 to 75°C operating temperature

RM-2600-T: ioPAC I/O module with 16 digital outputs, 24 VDC sink type, -40 to 75°C operating temperature

RM-3802-T: ioPAC I/O module with 8 analog inputs, 4 to 20 mA, -40 to 75°C operating temperature

RM-3810-T: ioPAC I/O module with 8 analog inputs, 0 to 10 V, -40 to 75°C operating temperature

KM-2430-T: ioPAC 4-port unmanaged Ethernet swtich module with M12 connector, -40 to 75°C operating temperature

Conformal Coating: Available on request

NOTE The 9th slot of the ioPAC 8020-9 series is reserved for future expansion. All I/O modules may only be installed in slot 1 through slot 8.

ioLogik W5348-C Series

ioLogik W5348-HSDPA-C: HSDPA micro RTU controller with 4 AIs, 8 DIOs, 2 relay outs, C programming capability, -10 to 55°C operating temperature

ioLogik W5348-HSDPA-C-T: HSDPA micro RTU controller with 4AIs, 8 DIOs, 2 relay outs, C programming capability, -20 to 70°C operating temperature

Product Specifications

ioPAC 8020 Series

Computer

CPU: ARM9 based CPU, 32-bit/160 MHz

OS: Linux

Clock: Real-time clock with battery backup

SDRAM: 64 MB

Flash: 32 MB

SD™ Slot: Up to 32 GB (SD 2.0 compatible)

Note: For units operating in extreme temperatures, industrial grade, wide-temperature SD cards are required.

Ethernet Interface

LAN: 2 auto-sensing 10/100 Mbps switch ports (M12 or RJ45)

Ethernet Relay Function: Hardware Normal Close

Protection: 1.5 KV magnetic isolation

Serial Interface

Serial COM1: RS-232/422/485 (DB9 male)

Serial Debug Port: RS-232 (4-pin connector)

Serial COM1 Signals

RS-232: TxD, RxD, DTR, DSR, RTS, CTS, DCD, GND

RS-422: TxD+, TxD-, RxD+, RxD-, GND

RS-485-4w: TxD+, TxD-, RxD+, RxD-, GND

RS-485-2w: Data+, Data-, GND

Power Requirements

Input Voltage: 12 to 36 VDC

Note: Compliant with EN 50155 at 24 VDC

Power Consumption: R[25]C184 mA @ 24 VDC (without I/O modules)

Physical Characteristics

Housing: Aluminum

Dimensions:

5-slot Version: 190.9 x 135 x 100 mm (7.52 x 5.31 x 3.94 in)

9-slot Version: 292.5 x 135 x 100 mm (11.52 x 5.31 x 3.94 in)

I/O Module Slots: 5 or 9 slots (the 9th slot is reserved)

Weight:

5-slot Version: 2,000 g

9-slot Version: 2,575 g

Mounting: DIN rail (standard), wall (with optional kit)

Environmental Limits

Operating Temperature: -40 to 75°C (-40 to 167°F)

Storage Temperature: -40 to 85°C (-40 to 185°F)

Ambient Relative Humidity: 5 to 95% (non-condensing)

Standards and Certifications

Safety: UL 508

EMI:

EN 61000-3-2; EN 61000-3-3; EN 61000-6-4;
FCC Part 15, Subpart B, Class A

EMS:

EN 55024, EN 61000-4-2, EN 61000-4-3,
EN 61000-4-4, EN 61000-4-5, EN 61000-4-6,
EN 61000-4-8, EN 61000-4-11, EN 61000-6-2

Shock: IEC 60068-2-27

Freefall: IEC 60068-2-32

Vibration: IEC 60068-2-6

Rail Traffic: EN 50155, EN 50121-3-2, EN 50121-4

Green Product: RoHS, CRoHS, WEEE

Note: Please check Moxa’s website for the most up-to-date certification status.

MTBF (mean time between failure)

Time: 690,214 hrs

Database: Telcordia (Bellcore)

Warranty

Warranty Period: 5 years

Details: See www.moxa.com/warranty

ioLogik W5348 Series

Computer

CPU: ARM9 based CPU, 32-bit/160 MHz

SDRAM/Flash: 64 MB / 32 MB

Storage

Expansion Slot: Up to 32 GB SD™ memory card (SD 2.0 compatible)

Note: For units operating in extreme temperatures, industrial grade, wide-temperature SD cards are required.

Cellular

Network:

ioLogik W5348-HSDPA-C:

- Tri-band UMTS/HSDPA 850/1900/2100 MHz
- Quad-band GSM/GPRS/EDGE 850/900/1800/1900 MHz

Internet:

HSDPA:

- Up to 3.6M bps upload speed.
- Up to 384k bps download speed.

UMTS:

- Up to 384k bps upload/download speed.

GPRS/EDGE:

- Multi-slot class: Class 10
- Coding schemes: CS 1-4, MCS 1-9
- Terminal device class: Class B

SMS: Point-to-Point Text/PDU mode

SIM Control Voltage: 3 V

LAN

Ethernet: 1 x 10/100 Mbps, RJ45

Protection: 1.5 KV magnetic isolation

Protocols: Modbus/TCP, TCP/IP, UDP, DHCP, Bootp, SNMP, SNTP

Serial Communication

Interface: 2 x RS-232/422/485, software selectable (9-pin D-Sub male)

Baudrate:

300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps

Inputs and Outputs

Analog Inputs: 4 channels

Configurable DIOs: 8 channels

Relay Outputs: 2 channels

Isolation: 3K VDC or 2K Vrms

Analog Input

Type: Differential input

Resolution: 16 bits

I/O Mode: Voltage / Current

Input Range: 0 to 10 V, ±10 V, ±5 V, 0 to 20 mA, 4 to 20 mA

Accuracy:

- ±0.1% FSR @ 25°C
- ±0.3% FSR @ -30 and 70°C

Sampling Rate (all channels):

- W5340: 100 samples/sec
- W5348: 5 samples/sec

Input Impedance: 200K ohms (min.)

Built-in Resistor for Current Input: 102 ohms

Digital Input

Sensor Type: Wet Contact (NPN or PNP) and Dry Contact

I/O Mode: DI or Event Counter

Dry Contact:

- On: short to GND
- Off: open

Wet Contact (DI to GND):

- On: 0 to 3 VDC
- Off: 10 to 30 VDC

Common Type: 4 points per COM

Counter Frequency: 250 Hz

Digital Filtering Time Interval: Software selectable/Programmable

Digital Output

Type: Sink

I/O Mode: DO or Pulse Output

Pulse Output Frequency: 500 Hz

Over-voltage Protection: 45 VDC

Over-current Protection: 2.6 A (4 channels @ 650 mA)

Over-temperature Shutdown: 160°C (min.)

Current Rating: 200 mA per channel

DIO Output Leakage Current: 3.6 mA @ 24 VDC

Relay Output

Type: Form A (N.O.) power relay

Contact Current Rating:

- Resistive Load: 1 A @ 30 VDC, 250 VAC, 110 VAC

Initial Insulation Resistance: 1000 m ohms (min.) @ 500 VDC

Mechanical endurance: 5,000,000 operations

Electrical endurance: 600,000 operations @ 1 A resistive load

Contact Resistance: 100 m ohms (max.)

Pulse Output: 0.3 Hz at rated load

Power Requirements

Power Input: 24 VDC nominal, 12 to 36 VDC

Power Consumption:

ioLogik W5348-HSDPA-C:

- Always on: 196 mA @ 24 VDC
- On demand: 189 mA @ 24 VDC

Physical Characteristics

Dimensions: 46.8 x 135 x 105 mm (1.84 x 5.31 x 4.13 in)

Weight: 495 g

Mounting: DIN-rail (standard), wall (optional)

Environmental Limits

Operating Temperature:

Standard Models: -10 to 55°C (14 to 131°F)

Wide Temp. Models:

ioLogik W5348-HSDPA-C: -20 to 70°C (-4 to 158°F)

Storage Temperature: -40 to 85°C (-40 to 185°F)

Ambient Relative Humidity: 5 to 95% (non-condensing)

Standards and Certifications

Safety: UL 508, EN 60950-1, NCC

EMI:

EN 55022; EN 61000-3-2; EN 61000-3-3;

FCC Part 15, Subpart B, Class A

EMS:

EN 55024, EN 61000-4-2, EN 61000-4-3,

EN 61000-4-4, EN 61000-4-5, EN 61000-4-6,

EN 61000-4-8, EN 61000-4-11, EN 61000-6-2

Mobile Network: PTCRB

Shock: IEC 60068-2-27

Freefall: IEC 60068-2-32

Vibration: IEC 60068-2-6

Green Product: RoHS, CRoHS, WEEE

Note: Please check Moxa’s website for the most up-to-date certification status.

MTBF (mean time between failure)

Time:

ioLogik W5348-HSDPA-C: 280,739 hrs

Database: Telcordia (Bellcore)

Warranty

Warranty Period: 2 years*

*Because of the limited lifetime of power relays, products that use that component are covered by a 2-year warranty.

Details: See www.moxa.com/warranty

2

Getting Started

In this chapter, it explains how to connect the Moxa RTU C Programmable controller, and how to get started on the programming.

The following topics are covered in this chapter:

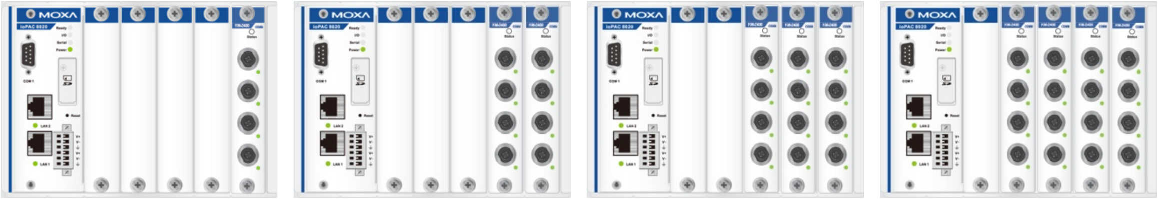
I/O and Communication Module Installation

The 5 I/O modules and single communication module may be selected for installation on the ioPAC 8020 system; all modules may be hot-swapped, allowing for convenient installation/removal at anytime.

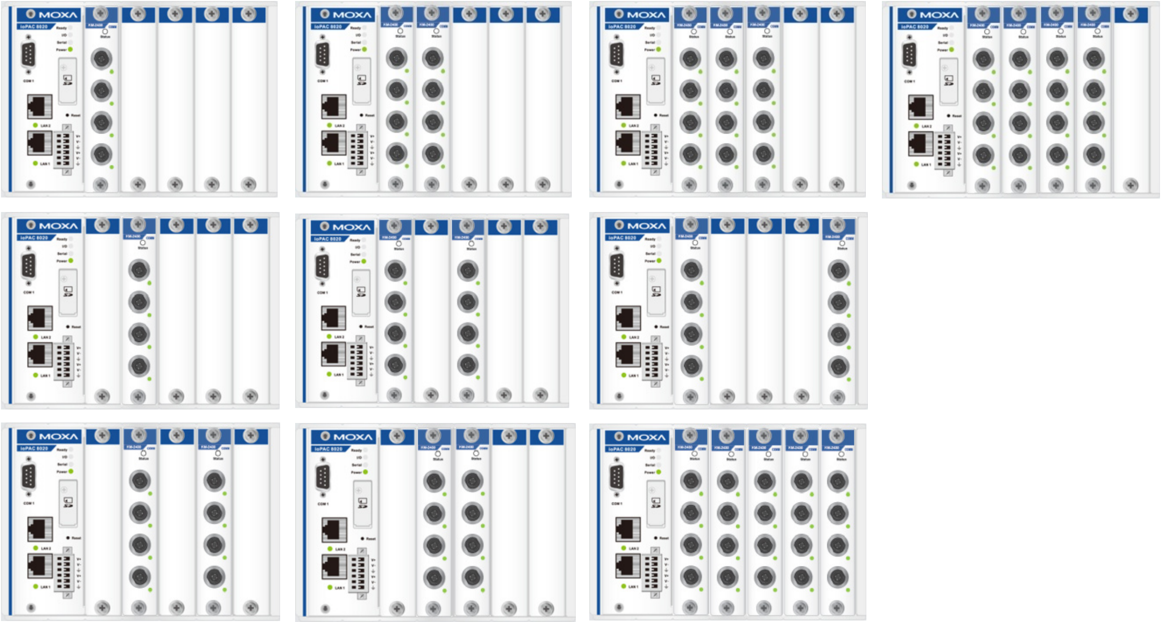
Care should be taken before installing the modules:

1. I/O modules (RM-1050-T, RM-1602-T, RM-2600-T, RM-3802-T, and RM-3810-T) may be installed on the ioPAC 8020-C system in any order. However, the 9th slot of the ioPAC 8020-9 series is reserved for future expansion, so I/O modules may only be installed in slots 1 through 8.
2. If multiple kM-2430-T Ethernet communication modules are to be installed on the ioPAC 8020-C system, the installation order must start from the last slot and continue consecutively, "downwards" (9,8,7…). Here are some examples:

Correct Installation (module installation in steps, from right to left)



Incorrect Installation (in steps, from right to left)



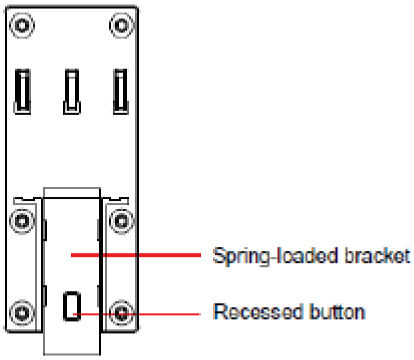
Installing the RTU Controller on a DIN-Rail

ioPAC 8020-C Series

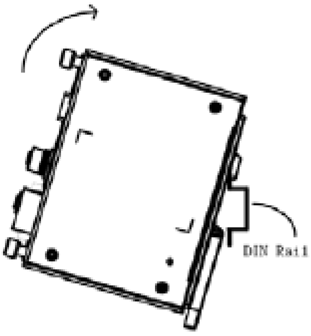
The aluminum DIN-rail attachment plate should already be fixed to the back panel of the ioPAC 8020-C when you take it out of the box. If you need to reattach the DIN-rail attachment plate to the ioPAC 8020-C, make sure the spring-loaded bracket is situated towards the bottom, as shown in the following figures.

NOTE **Users can purchase a wall-mounting (WK-75) kit separately.**

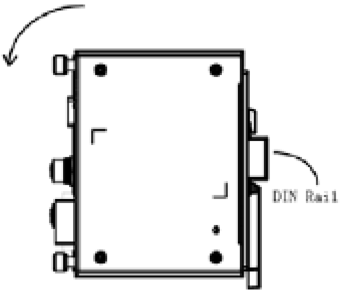
STEP 1: If the spring-loaded bracket is locked in place, push the recessed button to release it. Once it is released, you should feel some resistance from the spring as you slide the bracket up and down a few millimeters in each direction.



STEP 2: Insert the top of the DIN-rail into the top slots on the DIN-rail attachment plate.

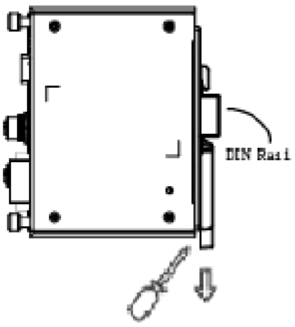


STEP 3: The DIN-rail attachment unit will snap into place as shown below.



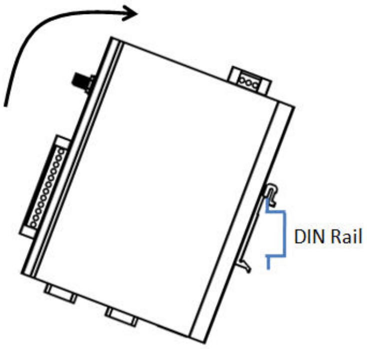
Removing ioPAC 8020-C Series from the DIN-Rail

To remove the ioPAC 8020-C from the DIN-rail, use a screwdriver to push down the spring-loaded bracket until it locks in place, as shown in the diagram to the right. Next, rotate the bottom of the switch upwards and then remove the switch from the DIN-rail.

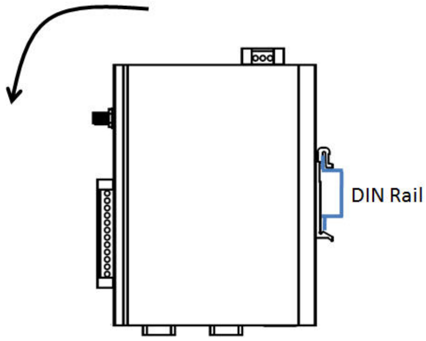


ioLogik W5348-C Series

STEP 1: Insert the top of the DIN-rail into the slot just below the stiff metal spring.



STEP 2: The DIN-rail attachment unit will snap into place as shown below.

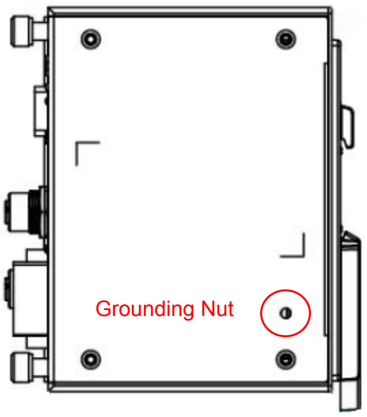


To remove the ioLogik unit from the DIN-rail, simply reverse Steps 1 and 2 above.

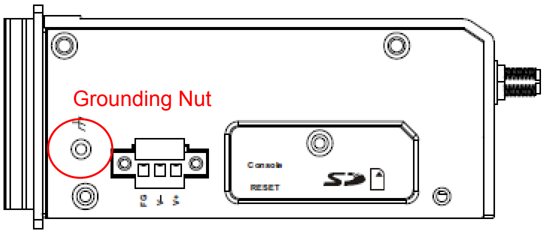
Grounding the RTU Controller

The Moxa C programming RTU Controller is grounded to enhance EMS performance. The RTU controller comes with a metal DIN-Rail bracket for grounding the system. For optimal EMS performance, connect the chassis ground nut on the RTU controller to the grounding point.

ioPAC 8020-C Series



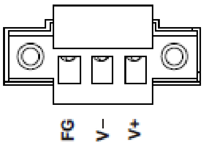
ioLogik W5348 Series:



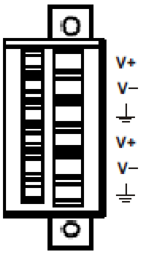
Powering on the RTU Controller

Connect the 12 to 36 VDC power lines from the power supply to the Moxa C programming RTU controller’s power terminal block, and then power on the power supply attached to it. It takes about 30 to 60 seconds for the system to boot up. Once the system is ready, the Ready LED will light up.

Power Terminal Block for ioLogik W5348-C Series



Power Terminal Block for ioPAC 8020-C Series



NOTE After connecting the RTU controller to the power supply, it will take about 30 to 60 seconds for the operating system to boot up. The green Ready LED will not turn on until the operating system is ready.

ATTENTION

This product is intended to be supplied by a Listed Power Unit with output marked "LPS" and rated for 12 to 36 VDC, 1.2A (minimum requirements)

LED Indicators

ioPAC 8020-C Series

Mark	Function	Description
Power	Power input	OFF: No system power available
		Green: Power on
Serial	Serial communication activity	OFF: No serial communication
		Green: Serial Tx/Rx
I/O	Tool chain API control	Green/Red/Off: Controlled by API
Ready	System status	Green: System ready
	System status	Red: System error
	System status	Green Flashing: System booting
	Tool chain API control	Green/Red/Off: After booting up, the API is able to control this LED
Port1/2	Ethernet communication activity	Off: Ethernet disconnected
		Green/Flashing: Ethernet Tx/Rx

ioLogik W5348-C Series

Mark	Function	Description
PWR	Power input	OFF: No system power available
		Green: Power on
LINK	Tool chain API control	Green/Off
READY	System status	Green: System ready
		Off: System boot up error
		Green Flashing: System booting
	Tool chain API control	Green/Off: After booting up, the API is able to control this LED.
DATA	Serial communication activity	OFF: No serial communication
		Green: Serial Tx/Rx
FAULT	Tool chain API control	Red/Off
SIGNAL	Tool chain API control	Green/Off

Connecting the RTU Controller to a PC

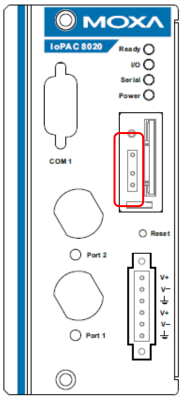
There are three ways to connect the Moxa C programming RTU controller to a PC: through the serial console, Telnet/SSH console, or the RTUAdmin utility over the network.

Serial Console

The serial console gives users a convenient way of connecting to the RTU controller. This method is particularly useful when using the computer for the first time. The serial console is also effective for connecting the RTU controller when users do not know the target network settings and IP addresses.

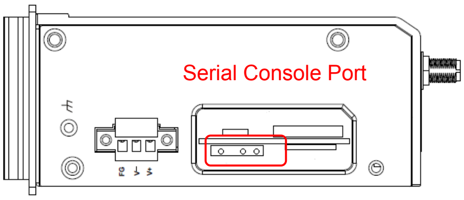
To use the serial console, remove the cover from the front and top panel first, and attach the 4-pin serial console cable to the console port.

Console Port for the ioPAC 8020-C Series



Serial Console Port

Console Port for the ioLogik W5348-C Series



Serial Console Port

Pin Assignment for the Serial Console Port

No.	Pin Assignments
1	Tx
2	Rx
3	N/A
4	GND

Serial Console

Baudrate	115200 bps
Parity	None
Data bits	8
Stop bits:	1
Flow Control	None
Terminal	VT100

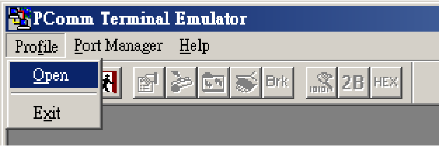
To connect to the RTU's serial console, Moxa PComm Terminal Emulator is recommended to be used as the console terminal. In the following steps, we describe how to connect the console.

STEP 1: Find the Moxa PComm Lite from the Moxa website (www.moxa.com) or Document and Software CD\ Software\utility\PComm\

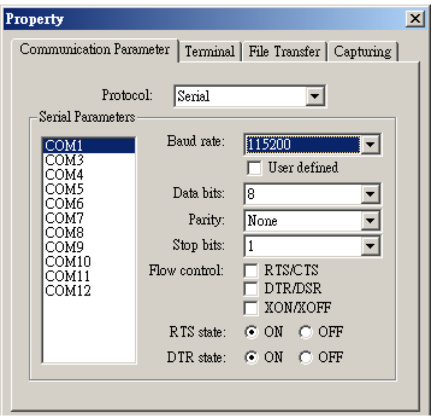
STEP 2: Install the Moxa PComm Lite to the host Windows PC.

STEP 3: Run the PComm Lite Terminal Emulator from Start\Programs\PComm Lite Ver 1.x\Terminal Emulator.

STEP 4: Click on Profile\Open.



STEP 5: Specify the COM port that is connecting the RTU controller, and configure the settings to 115,200, 8, none, and 1.



Protocol: Serial

Baud rate: 115200

□ User defined

Data bits: 8

Parity: None

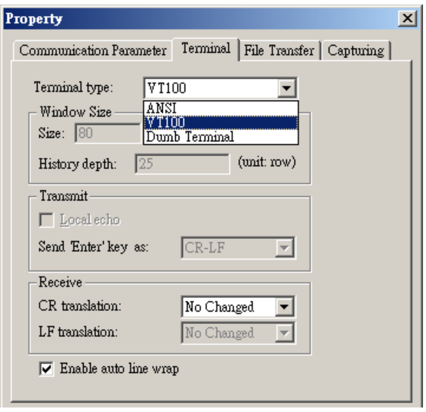
Stop bits: 1

Flow control: □ RTS/CTS □ DTR/DSR □ XON/XOFF

RTS state: ● ON ○ OFF

DTR state: ● ON ○ OFF

STEP 6: Click on the Terminal tab, and configure the Terminal Type to VT100. Click OK to proceed.



Terminal type: VT100

Size: 80

History depth: 25 (unit: row)

□ Local echo

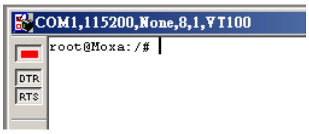
Send 'Enter' key as: CR-LF

CR translation: No Changed

LF translation: No Changed

☑ Enable auto line wrap

STEP 7: Serial console will be displayed on the terminal screen.



```
COM1,115200,None,8,1,VT100
root@Moxa:/#
```

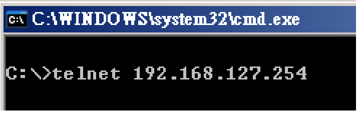
Telnet Console

It will be easy to use the system embedded command "telnet" to connect the RTU controller via network connection. The default IP address and Netmask are given below:

	Default IP Address	Netmask
LAN 1	192.168.127.254	255.255.255.0

Use a cross-over Ethernet cable to connect directly from the host PC to the RTU controller. User must first modify the host PC's IP address and netmask to be in the same subnet as the target RTU controller. For example, users can set the host PC's IP address to 192.168.127.253 and netmask to 255.255.255.0 to meet the default settings of the RTU controller.

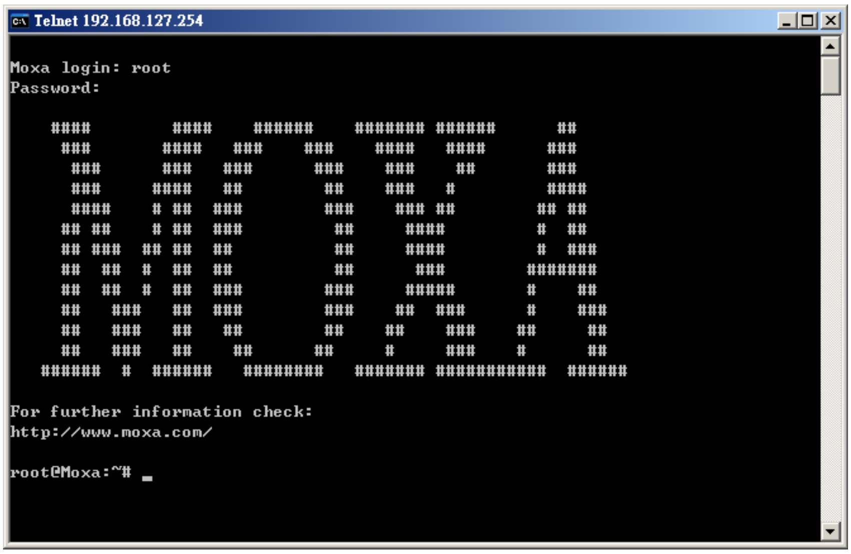
To start the telnet console, launch the Windows Command prompt and use the following command:



```
C:\>telnet 192.168.127.254
```

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

Login: root
Password: root



```
Moxa login: root
Password:

    ####        ####    ######    ####### ######      ##
     ###       ####   ###    ###    ####   ####      ###
      ###      ###   ###      ###    ###    ##       ###
      ###     ####   ##        ##    ###   #         ####
      ####    # ##  ###        ###    ### ##        ## ##
     ## ##    # ##  ###         ##     ####         #  ##
     ## ###  ## ##  ##          ##     ####         #  ###
     ##  ##  #  ##  ##          ##      ###        #######
     ##  ##  #  ##  ###        ###     #####       #    ##
     ##   ###   ##  ###        ###    ##  ###      #    ###
     ##   ###   ##   ##        ##    ##    ###    ##     ##
     ##   ###   ##    ##      ##     #     ###    #      ##
   ######  #  ######   ########   ####### ###########  ######

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

root@Moxa:~# _
```

You can proceed with configuring the network settings of the target RTU controller when you reach the bash command shell. Configuration instructions are given in the next section.

Users can perform the "logout" command to terminate the console.

ATTENTION

Serial Console Reminder

Remember to choose VT100 as the terminal type. Use the cable CBL-4PINDB9F-100, which comes with the RTU controller, to connect to the serial console port.

Telnet Reminder

When connecting a PC to the RTU controller over a LAN, users must configure the PC's Ethernet IP address to be on the same subnet as the RTU controller.

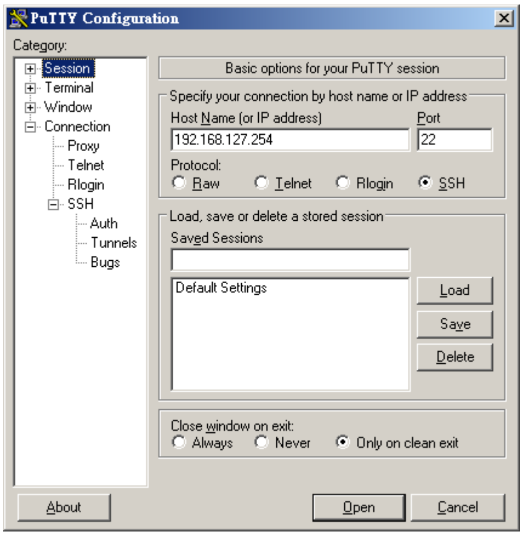
For operating system that does not have "telnet" commands, such as Windows 7, intall the RTUAdmin utility from the Document and Software CD to perform network access to the RTU Controller.

SSH Console

The RTU controller supports an SSH console to provide users with better security options.

Windows Users

Click on the link http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html to download PuTTY (free software) to set up an SSH console for the Moxa RTU in a Windows environment. The following figure shows a simple example of the configuration that is required.



Specify your connection by host name or IP address

Host Name (or IP address): 192.168.127.254

Port: 22

Protocol: ○ Raw ○ Telnet ○ Rlogin ● SSH

Load, save or delete a stored session

Saved Sessions: Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Click on "Yes" to accept the security key exchange from the RTU controller's SSH console.

Linux Users

For a Linux-based system, users can use the "ssh" command to access the console of the RTU controller via SSH.

#ssh 192.168.127.254

Select **yes** to complete the connection.

```
[root@bee_notebook root]# ssh 192.168.127.254
The authenticity of host '192.168.127.254 (192.168.127.254)' can't be established.
RSA key fingerprint is 8b:ee:ff:84:41:25:fc:cd:2a:f2:92:8f:cb:1f:6b:2f.
Are you sure you want to continue connection (yes/no)? yes_
```

NOTE SSH provides better security compared to Telnet for accessing the RTU controller's console over the network.

RTUAdmin Utility

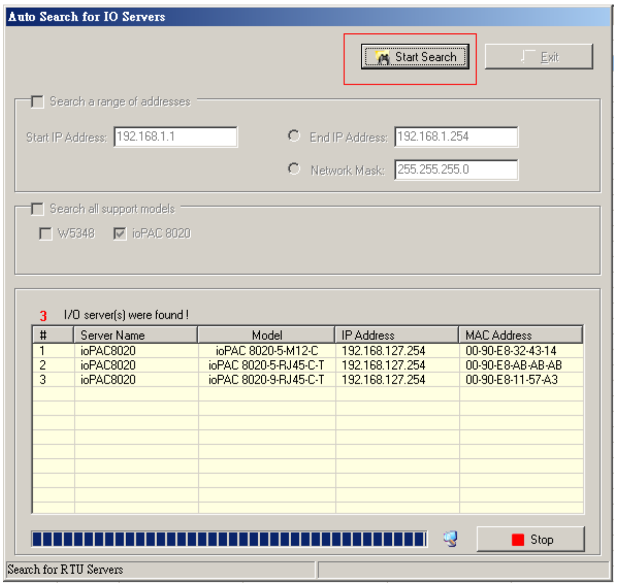
The Moxa C programming RTU controller can be managed and configured over the Ethernet or Cellular network with RTUAdmin, a Windows utility provided with Moxa RTU products. RTUAdmin's graphical user interface gives you advanced search function to find a Moxa RTU controller in the local network, and provide easy access to the telnet console.

Installing RTUAdmin

Insert the Document and Software CD into the host computer. In the **Software\utility\RTUAdmin\RTUAdminSetup** directory of the CD, locate and run **SETUP.EXE**. The installation program will guide you through the installation process and install the ioAdmin utility. After the installation is finished, run ioAdmin from the Windows Start menu.

Broadcast Search

Select the model and click the "Start Search" button to proceed.



□ Search a range of addresses

Start IP Address: 192.168.1.1

○ End IP Address: 192.168.1.254

○ Network Mask: 255.255.255.0

□ Search all support models

□ W5348 ☑ ioPAC 8020

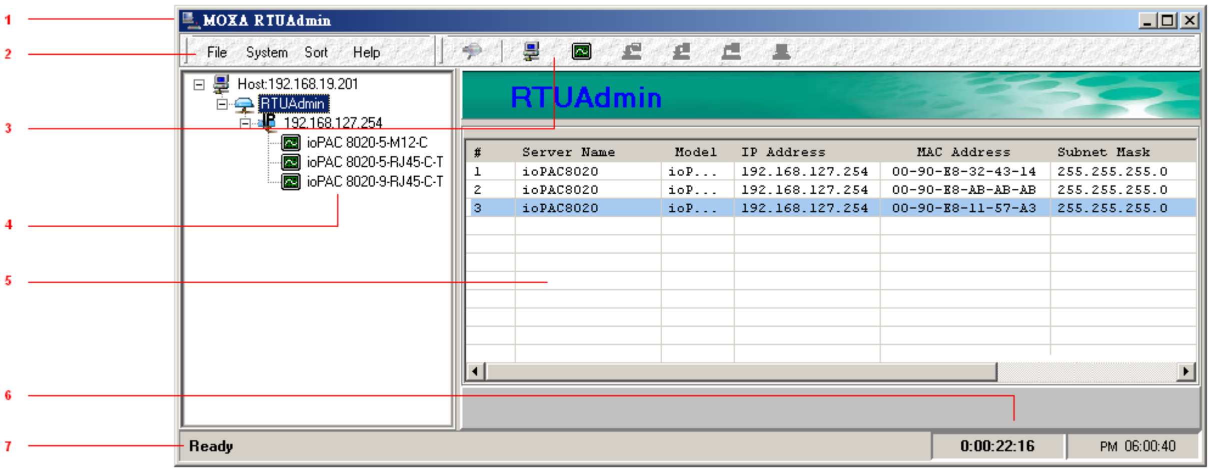
3 I/O server(s) were found !

#	Server Name	Model	IP Address	MAC Address
1	ioPAC8020	ioPAC 8020-5-M12-C	192.168.127.254	00-90-E8-32-43-14
2	ioPAC8020	ioPAC 8020-5-RJ45-C-T	192.168.127.254	00-90-E8-AB-AB-AB
3	ioPAC8020	ioPAC 8020-9-RJ45-C-T	192.168.127.254	00-90-E8-11-57-A3

Search for RTU Servers

Main Screen Overview

The main screen displays all the results of broadcast search.



Host:192.168.19.201
- RTUAdmin
  - 192.168.127.254
    - ioPAC 8020-5-M12-C
    - ioPAC 8020-5-RJ45-C-T
    - ioPAC 8020-9-RJ45-C-T

#	Server Name	Model	IP Address	MAC Address	Subnet Mask
1	ioPAC8020	ioP...	192.168.127.254	00-90-E8-32-43-14	255.255.255.0
2	ioPAC8020	ioP...	192.168.127.254	00-90-E8-AB-AB-AB	255.255.255.0
3	ioPAC8020	ioP...	192.168.127.254	00-90-E8-11-57-A3	255.255.255.0

Ready 0:00:22:16 PM 06:00:40

1. Title	2. Menu Bar	3. Quick Link	4. Navigation Panel
5. Main Window	6. Sync. Rate Status	7. Status Bar	

Title

It displays the program that is running. In this case, it is Moxa RTUAdmin.

Menu Bar

The Menu bar has four items: File, System, Sort, and Help.

File

Use the File\Exit to close the RTUAdmin.

System

a. **Auto Search:** Searches for Moxa RTU controllers on the local network.

b. **Network Interface:** Selects a network interface to use the RTUAdmin.

c. **Auto Search Timeout:** Selects the preferred timeout value for broadcast search.

Sort

a. **By Connection:** Sorts by the target Moxa RTU controller's IP address.

b. **By Server:** Sorts by the target Moxa RTU controller's model name.

Help

Displays the software information of the RTUAdmin.

Quick Link

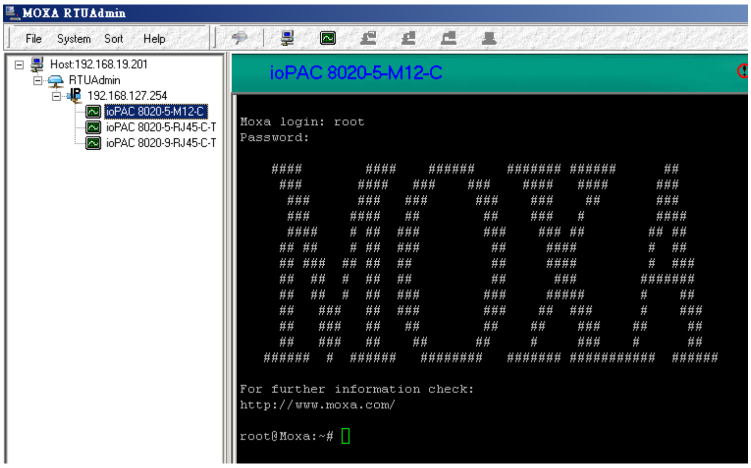
Icon	Function
🔍	Auto Search
🖥	Sort by Connection
▣	Sort by Server

Navigation Panel

Lists the current search result.

Main Window

Displays the detailed information about the Moxa RTU controllers that are found. Click on the specific product in the Navigation Panel, the Main Window will connects to its telnet console automatically.



```
Moxa login: root
Password:

    ####        ####    ######    ####### ######      ##
     ###       ####   ###    ###    ####   ####      ###
      ###      ###   ###      ###    ###    ##       ###
      ###     ####   ##        ##    ###   #         ####
      ####    # ##  ###        ###    ### ##        ## ##
     ## ##    # ##  ###         ##     ####         #  ##
     ## ###  ## ##  ##          ##     ####         #  ###
     ##  ##  #  ##  ##          ##      ###        #######
     ##  ##  #  ##  ###        ###     #####       #    ##
     ##   ###   ##  ###        ###    ##  ###      #    ###
     ##   ###   ##   ##        ##    ##    ###    ##     ##
     ##   ###   ##    ##      ##     #     ###    #      ##
   ######  #  ######   ########   ####### ###########  ######

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

root@Moxa:~# 
```

Sync Rate Status

Indicates the period that RTUAdmin has been launched.

Status Bar

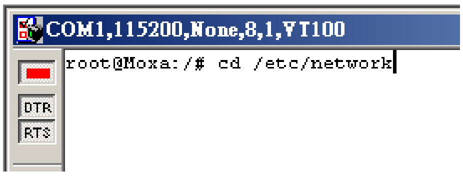
Displays the current time.

Configuring the Ethernet Interface

The network settings of the Moxa C programming RTU controller can be modified with the serial console port, or online over the network.

Modifying Network Settings via Serial Console

Follow the instructions given in the previous section to access the serial console of the target RTU controller, and then type **#cd /etc/network** to change directories.

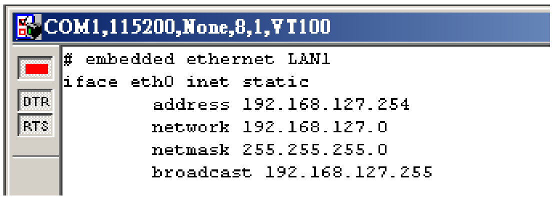


```
root@Moxa:/# cd /etc/network
```

Type **#vi interfaces** to edit the network configuration file with vi editor. Users can configure the Ethernet ports of the RTU controller for static or dynamic (DHCP) IP addresses.

Static IP addresses

As shown in the table below, 4 network addresses must be modified: **address, network, netmask,** and **broadcast**. The default IP address for LAN1 is 192.168.127.254, with default netmask of 255.255.255.0.



```
# embedded ethernet LAN1
iface eth0 inet static
        address 192.168.127.254
        network 192.168.127.0
        netmask 255.255.255.0
        broadcast 192.168.127.255
```

Dynamic IP addresses

By default, the RTU controller is configured for "static" IP addresses. To configure the LAN port to request an IP address dynamically, replace **static** with **dhcp** and then delete the address, network, netmask, and broadcast lines.

Default Setting for LAN1	Dynamic Setting using DHCP
iface eth0 inet **static** address 192.168.127.254 network: 192.168.127.0 netmask 255.255.255.0 broadcast 192.168.127.255	iface eth0 inet **dhcp**

After the LAN interface settings have been modified and saved, perform the following command to activate the LAN settings immediately:

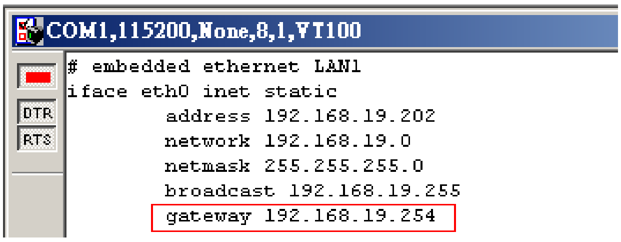
#/etc/init.d/networking restart

NOTE After changing the IP settings, use the networking restart command to activate the new IP address. Users must modify the **/etc/network/interfaces** to store the new settings. Using commands such as **#ifconfig eth0 192.168.1.1** WILL NOT save the settings to the flash memory.

Adding a Default Gateway

Follow the instructions given in a previous section to access the serial console of the target Moxa RTU Controller, and then type **#cd /etc/network** to change directories.

1. Type **#vi interfaces** to edit the network configuration file with vi editor.
2. Add the gateway IP to the last entry of the interface settings.



```
# embedded ethernet LAN1
iface eth0 inet static
        address 192.168.19.202
        network 192.168.19.0
        netmask 255.255.255.0
        broadcast 192.168.19.255
        gateway 192.168.19.254
```

After the LAN interface settings have been modified and saved, perform the following command to activate the LAN settings immediately:

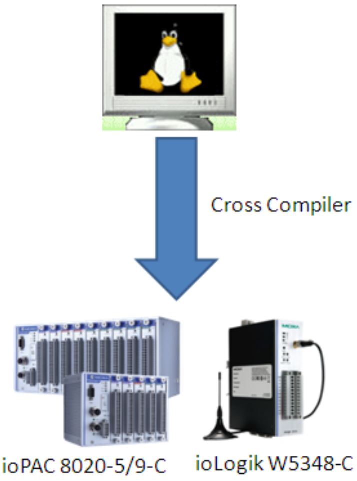
#/etc/init.d/networking restart

Adding DNS Settings

For details, refer to *Chapter 4, Managing Communications .*

Developing Procedures

In general, program development involves the following seven steps.



Step 1:
Connect the RTU controller to a Linux PC.
Step 2:
Install Tool Chain (GNU Cross Compiler & glibc) from the Document and Software CD.
Step 3:
Set the cross compiler and glibc environment variables.
Step 4:
Code and compile the program.
Step 5:
Download the program to the target RTU controller using FTP or NFS.
Step 6:
Debug the program
→ If bugs are found, return to Step 4.
→ If no bugs are found, continue with Step 7.
Step 7:
Back up the user directory (distribute the program to additional Moxa RTU controller if needed).

Installing the Tool Chain (Linux)

The Linux Operating System must be pre-installed in the host PC before installing the RTU controller's GNU Tool Chain. Fedora Core or compatible versions are recommended. The Tool Chain requires approximately 150 MB of hard disk space on your PC. The RTU controller's Tool Chain software is located on the attached Document and Software CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount/dev/cdrom /mnt/cdrom
#sh /mnt/cdrom/Software/toolchain/arm-linux_3.3.2_V1.X_BuildXXXXXXXX.sh
```

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

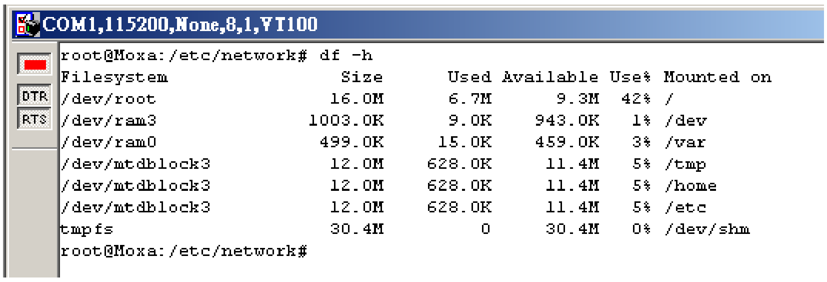
```
PATH=/usr/local/arm-linux/bin:$PATH
```

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

Checking the Flash Memory Space

If the flash memory is full, you will not be able to save data to the Flash ROM. Use the following command to calculate the amount of "Available" flash memory:

```
/>df –h
```



```
root@Moxa:/etc/network# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/root                16.0M      6.7M      9.3M  42% /
/dev/ram3              1003.0K      9.0K    943.0K   1% /dev
/dev/ram0               499.0K     15.0K    459.0K   3% /var
/dev/mtdblock3           12.0M    628.0K     11.4M   5% /tmp
/dev/mtdblock3           12.0M    628.0K     11.4M   5% /home
/dev/mtdblock3           12.0M    628.0K     11.4M   5% /etc
tmpfs                    30.4M         0     30.4M   0% /dev/shm
root@Moxa:/etc/network#
```

If there isn't enough "Available" space for user's program, users must delete some existing files. To do this, connect the host PC to the RTU controller with the console cable, and then use the console utility to remove the none-using files from the flash memory. To check the amount of available space, look at the directories in the read/write directory **/dev/mtdblock3**. Note that the directories **/home and /etc** are both mounted on the directory **/dev/mtdblock3**.

NOTE If the flash memory is full, users must release some memory space before saving files to the Flash ROM.

Compiling Hello.c

The Software and Document CD contains an example **Hello.c** program allowing users to run pilot tests to the RTU controller. Type the following commands from the host PC to copy the files used for this example.

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

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

```
# cd example/hello
# make
```

Users should see the following response:

```
[root@localhost hello]# make
/usr/local/arm-linux/bin/arm-linux-gcc -o hello-release hello.c
/usr/local/arm-linux/bin/arm-linux-strip -s hello-release
/usr/local/arm-linux/bin/arm-linux-gcc -ggdb -o hello-debug hello.c
[root@localhost hello]# _
```

The **hello-release** and **hello-debug** are described as below:

hello-release—An ARM platform executable file (created specifically to run on the Moxa RTU Controllers)

hello-debug—An ARM platform GDB debug server execution file (see Chapter 5 for details)

NOTE Since the Moxa's tool chain places a specially designed Makefile in the directory **/tmp/example/ hello**, be sure to type the **#make** command from within that directory. If users type the **#make** command from any other directory, the host Linux PC may use other system compilers (for example, cc or gcc) and resulting errors.

Uploading and Running the "Hello" Program

1. Use the following commands to upload hello-release to the RTU controller by FTP.

   From the PC, type:
   #ftp 192.168.127.254

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

   ftp> bin
   ftp> cd /home
   ftp> put hello-release

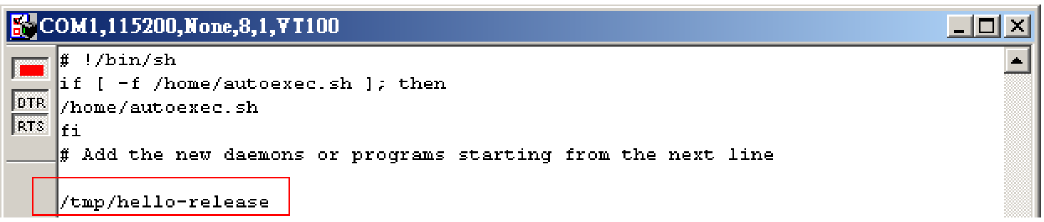
3. From the Moxa RTU, type:

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

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

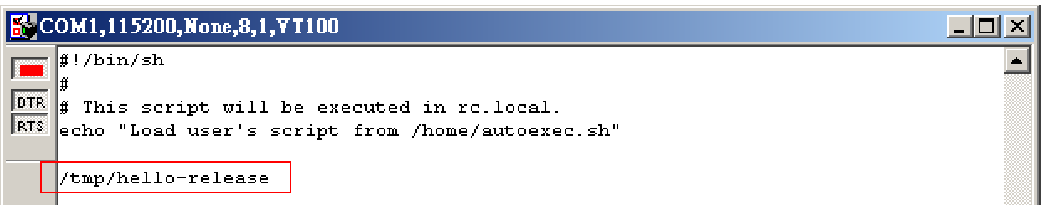
```
root@Moxa:~# ./hello-release
Hello
```

To run the program automatically after system starts, modify the **rc.local file at /etc/rc.d** as follows,



```
# !/bin/sh
if [ -f /home/autoexec.sh ]; then
/home/autoexec.sh
fi
# Add the new daemons or programs starting from the next line

/tmp/hello-release
```

Modify the **/home/autoexec.sh** will also be OK.



```
#!/bin/sh
#
# This script will be executed in rc.local.
echo "Load user's script from /home/autoexec.sh"

/tmp/hello-release
```

3

Managing the RTU Controllers

This chapter includes information for version control, deployment, updates, and peripherals. The information in this chapter will be useful when users need to run the same application on several Moxa RTU controllers.

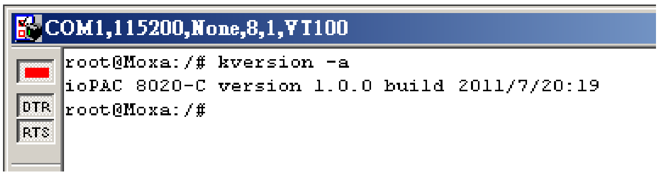
The following topics are covered in this chapter:

System Version Information

To verify the hardware capability of the target RTU controller, and the supported software, check the version numbers of your Moxa RTU controller's hardware, kernel, and user file system. Contact Moxa to verify the hardware version. You will need the Production S/N (Serial number), which is labeled on the RTU controller's bottom.

To check the kernel version, type:

#kversion -a



```
root@Moxa:/# kversion -a
ioPAC 8020-C version 1.0.0 build 2011/7/20:19
root@Moxa:/#
```

NOTE The kernel version number is for factory default configuration. You may download the latest firmware version from Moxa's website and then upgrade the RTU controller's hardware.

Firmware Upgrade and Default Settings

Upgrading the Firmware

The Moxa RTU controller's kernel and root file system are combined into one firmware file, which can be found in the Software and Document CD or be downloaded from Moxa's website (www.moxa.com). The name of the file has the form **xxxx.hfm**. To upgrade the firmware, the firmware file should be placed in the target RTU controller's Ramdisu, using SD card or FTP.

ATTENTION

Upgrading the firmware will erase all data on the Flash ROM
Firmware upgrade requires the RTU controller's RAM disk. If the application data is stored in the RAM disk, back up or remove the data before upgrading the firmware.

Since different Flash disks have different sizes, it is recommended to check the size of the current Flash disk before upgrading the firmware. You can do so by using the **#df –h** command to list the size of each memory block and see how much free space is available in each block.

Follow the steps to transfer the firmware file to the RTU controller's RAM disk, and to upgrade the firmware.

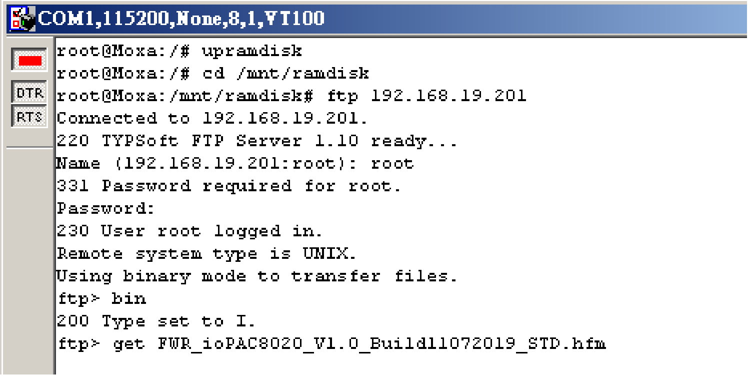
1. Type the following commands to enable the RAM disk:

   #upramdisk
   #cd /mnt/ramdisk

2. Activate the PC's FTP server, and put the firmware file to the FTP folder.

3. In Moxa RTU Controller's console, perform the following commands to use the built-in FTP client to retrieve the firmware file (xxxx.hfm) from the host PC to the RTU controller:

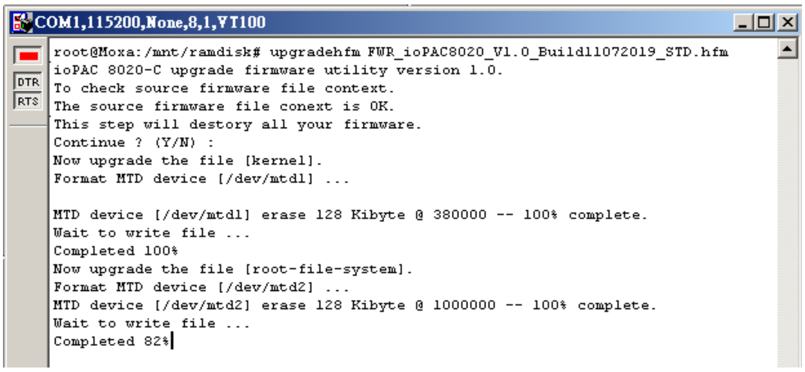
   /mnt/ramdisk# ftp <remote FTP Server's IP>
   Login Name: xxxx
   Login Password: xxxx
   ftp> bin
   ftp> get xxxx.hfm



```
root@Moxa:/# upramdisk
root@Moxa:/# cd /mnt/ramdisk
root@Moxa:/mnt/ramdisk# ftp 192.168.19.201
Connected to 192.168.19.201.
220 TYPSoft FTP Server 1.10 ready...
Name (192.168.19.201:root): root
331 Password required for root.
Password:
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bin
200 Type set to I.
ftp> get FWR_ioPAC8020_V1.0_Build11072019_STD.hfm
```

4. After the firmware file is transferred to the RAM disk, perform **upgradehfm** command to upgrade the kernel and root file system.

   #upgradehfm xxxx.hfm
   Press "Y" to complete the upgrade.



```
root@Moxa:/mnt/ramdisk# upgradehfm FWR_ioPAC8020_V1.0_Build11072019_STD.hfm
ioPAC 8020-C upgrade firmware utility version 1.0.
To check source firmware file context.
The source firmware file conext is OK.
This step will destory all your firmware.
Continue ? (Y/N) :
Now upgrade the file [kernel].
Format MTD device [/dev/mtd1] ...

MTD device [/dev/mtd1] erase 128 Kibyte @ 380000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [root-file-system].
Format MTD device [/dev/mtd2] ...
MTD device [/dev/mtd2] erase 128 Kibyte @ 1000000 -- 100% complete.
Wait to write file ...
Completed 82%
```

ATTENTION

The upfirm utility will reboot your target RTU controller after the upgrade is done.

Loading Factory Defaults

To load the factory default settings, press and hold the RESET button for more than 5 seconds. All files in the **/home &/etc** directory will be removed.

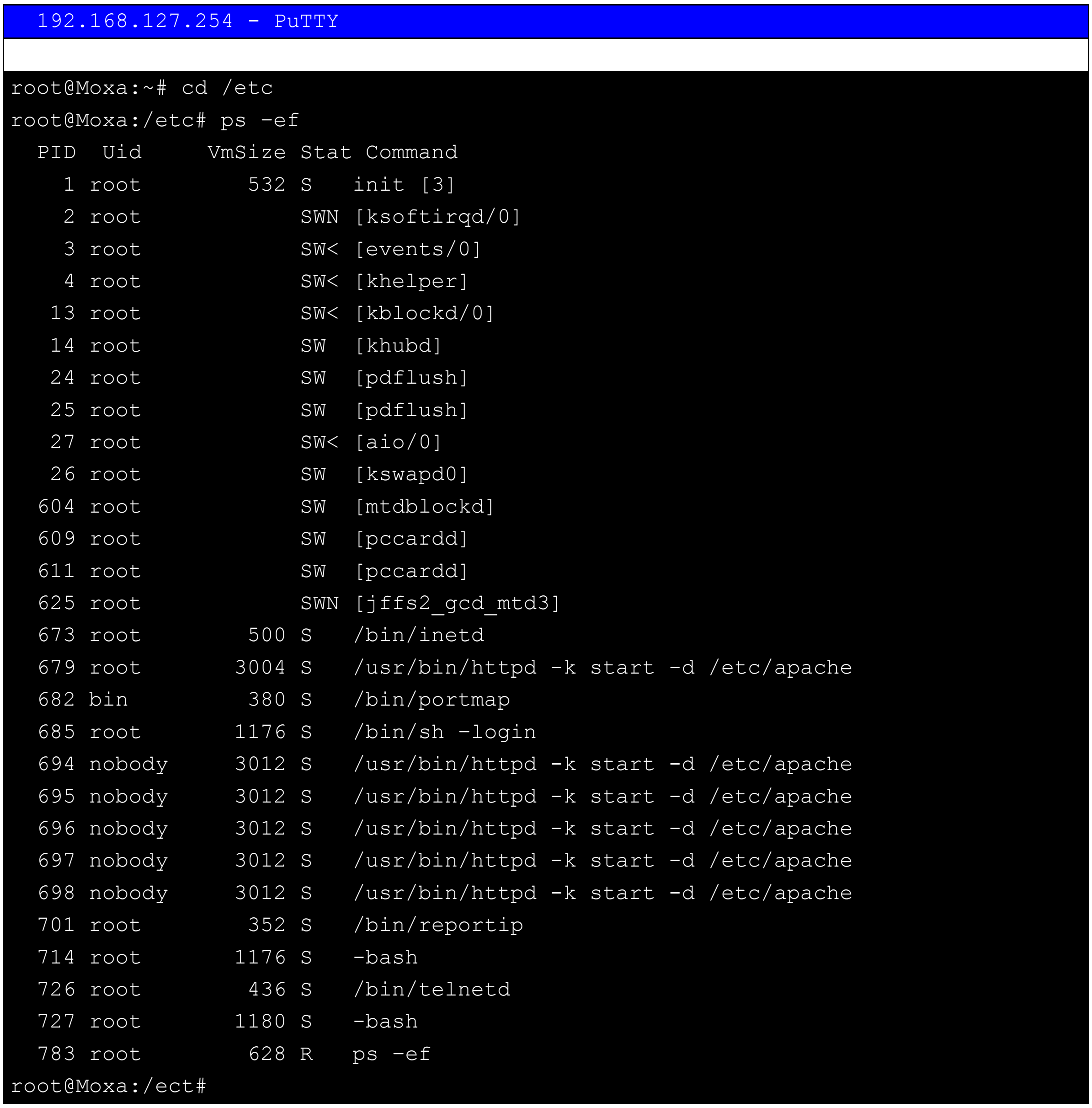
Enabling and Disabling Daemons

The following daemons are enabled when the RTU controller is boot up.

telnetd	Telnet Server / Client daemon
inetd	Internet Daemons
ftpd	FTP Server / Client daemon
sshd	Secure Shell Server daemon

httpd Apache WWW Server daemon

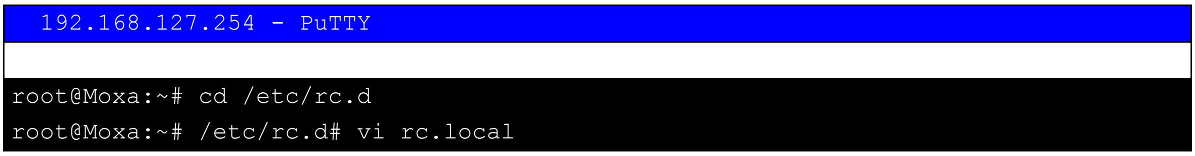
Perform the command "**ps -ef**" to list all the running processes.



```
root@Moxa:~# cd /etc
root@Moxa:/etc# ps -ef
  PID  Uid     VmSize Stat Command
    1 root        532 S    init [3]
    2 root            SWN  [ksoftirqd/0]
    3 root            SW<  [events/0]
    4 root            SW<  [khelper]
   13 root            SW<  [kblockd/0]
   14 root            SW   [khubd]
   24 root            SW   [pdflush]
   25 root            SW   [pdflush]
   27 root            SW<  [aio/0]
   26 root            SW   [kswapd0]
  604 root            SW   [mtdblockd]
  609 root            SW   [pccardd]
  611 root            SW   [pccardd]
  625 root            SWN  [jffs2_gcd_mtd3]
  673 root        500 S    /bin/inetd
  679 root       3004 S    /usr/bin/httpd -k start -d /etc/apache
  682 bin         380 S    /bin/portmap
  685 root       1176 S    /bin/sh -login
  694 nobody     3012 S    /usr/bin/httpd -k start -d /etc/apache
  695 nobody     3012 S    /usr/bin/httpd -k start -d /etc/apache
  696 nobody     3012 S    /usr/bin/httpd -k start -d /etc/apache
  697 nobody     3012 S    /usr/bin/httpd -k start -d /etc/apache
  698 nobody     3012 S    /usr/bin/httpd -k start -d /etc/apache
  701 root        352 S    /bin/reportip
  714 root       1176 S    -bash
  726 root        436 S    /bin/telnetd
  727 root       1180 S    -bash
  783 root        628 R    ps -ef
root@Moxa:/ect#
```

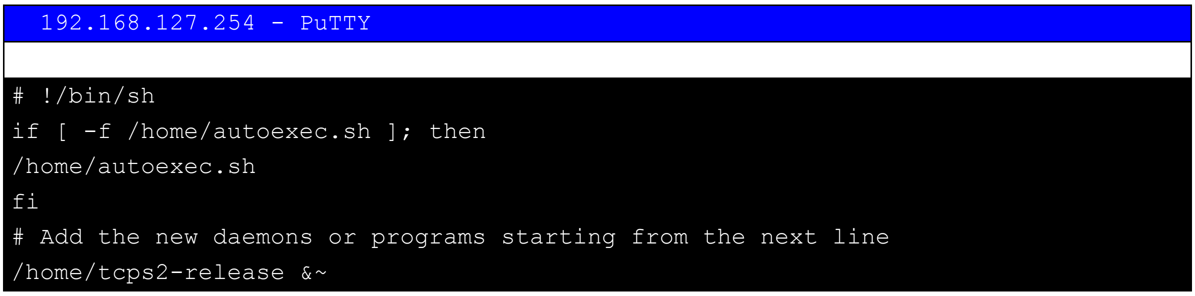
To run a private daemon, users need to edit the file **rc.local**, as follows:

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



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

The following diagram shows how to edit the last line of the **rc.local** to activate the complied example "**tcps2-release**", and run in the background.



```
# !/bin/sh
if [ -f /home/autoexec.sh ]; then
/home/autoexec.sh
fi
# Add the new daemons or programs starting from the next line
/home/tcps2-release &~
```

The enabled daemons will be available after rebooting the RTU controller.

An alternative is the **autoexec.sh** located in the **/home** directory to wake up and activate those daemons and programs. By editing and updating it to the RTU controller system, users will no longer need to modify the **rc.loca** file.

Setting the Run-Level

Perform the following commands to add or delete the settings, such as system service or programs, to the run levels.

#cd /etc/rc.d/init.d
If there is a program **tcps2** at **/home**, link it to the run level,

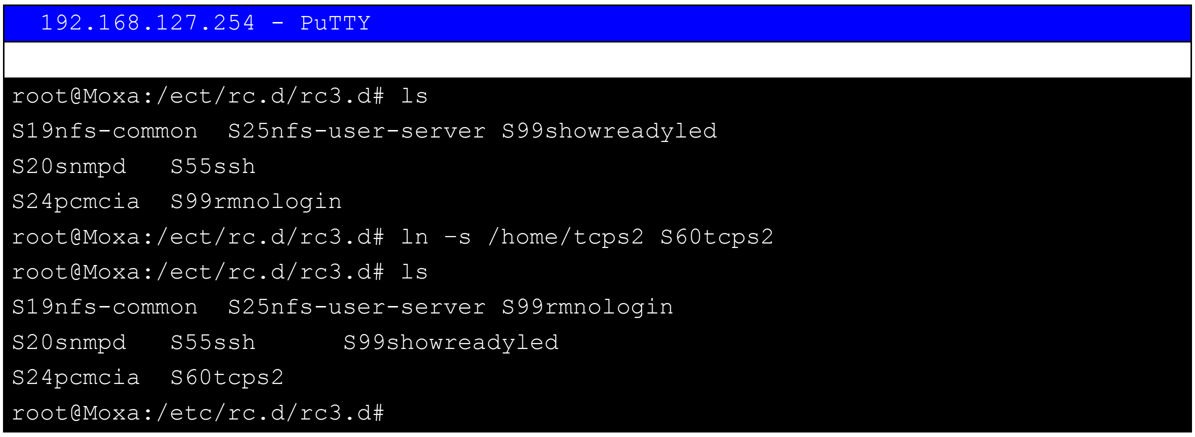
#cd /etc/rc.d/rc3.d
#ln –s /home/tcps2 S60tcps2

SxxRUNFILE stands for

S:Starts the run file while linux boots up.

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

RUNFILE: The file name.



```
root@Moxa:/ect/rc.d/rc3.d# ls
S19nfs-common  S25nfs-user-server S99showreadyled
S20snmpd   S55ssh
S24pcmcia  S99rmnologin
root@Moxa:/ect/rc.d/rc3.d# ln -s /home/tcps2 S60tcps2
root@Moxa:/ect/rc.d/rc3.d# ls
S19nfs-common  S25nfs-user-server S99rmnologin
S20snmpd   S55ssh      S99showreadyled
S24pcmcia  S60tcps2
root@Moxa:/etc/rc.d/rc3.d#
```

Remove the link by performing the following command:

#rm –f /etc/rc.d/rc3.d/S60tcps2

Adjusting the System Time

Setting the Time Manually

The Moxa C programming RTU controller has two time settings. One is the system time, and the other is the RTC (Real Time Clock) time kept by the RTU controller's hardware. Use the **#date** command to query the current system time or set a new system time. Use **#hwclock** to query the current RTC time or set a new RTC time.

Use the following command to query the system time:

#date

Use the following command to query the RTC time:

#hwclock

Use the following command to set the system time:

#date MMDDhhmmYYYY

MM = Month

DD = Date

hhmm = hour and minute

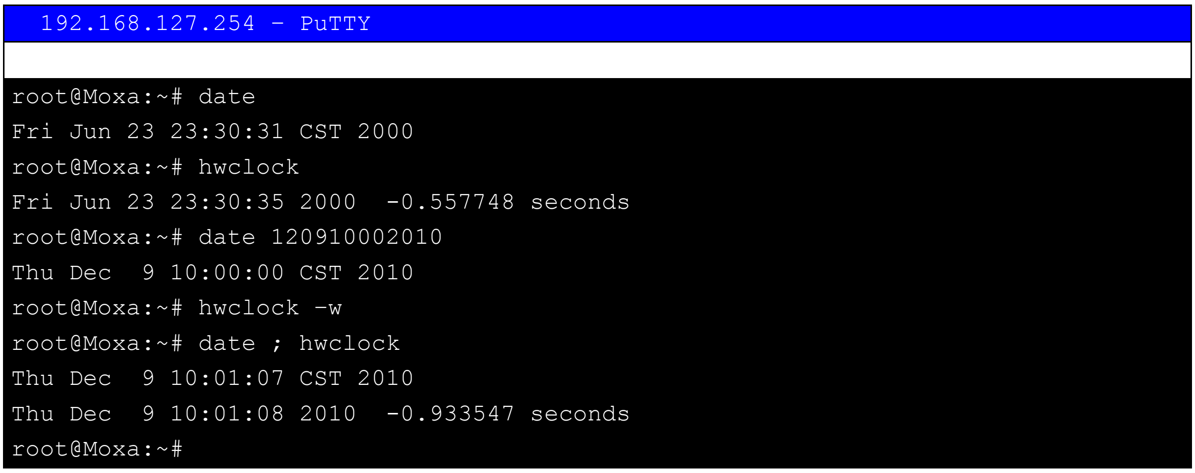
YYYY = Year

Use the following command to set the RTC time:

#hwclock –w

Write current system time to RTC.

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



```
root@Moxa:~# date
Fri Jun 23 23:30:31 CST 2000
root@Moxa:~# hwclock
Fri Jun 23 23:30:35 2000  -0.557748 seconds
root@Moxa:~# date 120910002010
Thu Dec  9 10:00:00 CST 2010
root@Moxa:~# hwclock -w
root@Moxa:~# date ; hwclock
Thu Dec  9 10:01:07 CST 2010
Thu Dec  9 10:01:08 2010  -0.933547 seconds
root@Moxa:~#
```

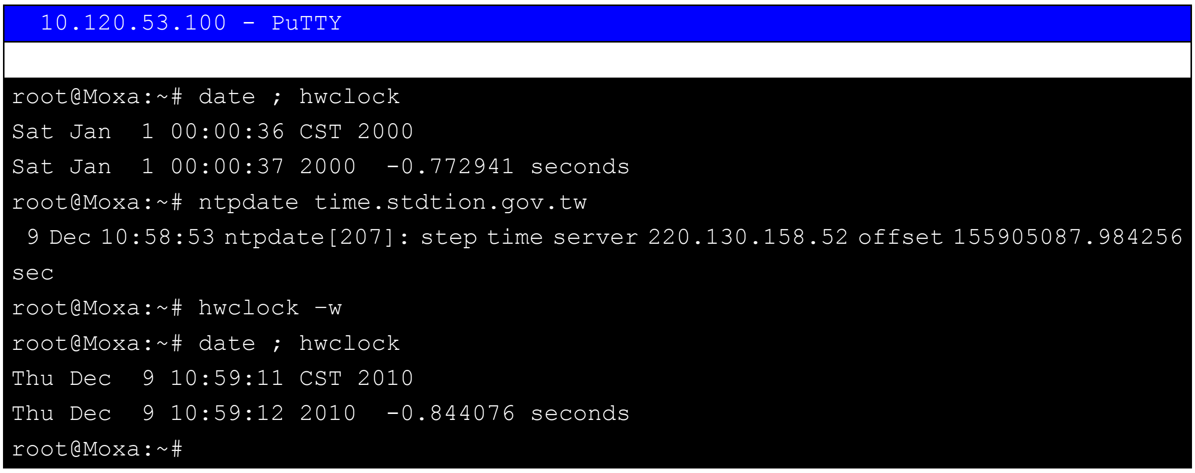
Updating the Time with NTP Client

The Moxa C programming RTU controller has a built-in NTP (Network Time Protocol) client that is used to initialize a time request to a remote NTP server. Use **#ntpdate <Time Server>** to update the system time, and save it to the RTC.

#ntpdate time.stdtime.gov.tw

#hwclock –w

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



10.120.53.100 - PuTTY

```
root@Moxa:~# date ; hwclock
Sat Jan  1 00:00:36 CST 2000
Sat Jan  1 00:00:37 2000  -0.772941 seconds
root@Moxa:~# ntpdate time.stdtion.gov.tw
 9 Dec 10:58:53 ntpdate[207]: step time server 220.130.158.52 offset 155905087.984256
sec
root@Moxa:~# hwclock -w
root@Moxa:~# date ; hwclock
Thu Dec  9 10:59:11 CST 2010
Thu Dec  9 10:59:12 2010  -0.844076 seconds
root@Moxa:~#
```

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

Updating the Time Automatically

In this section, it shows how to use a shell script to update the time automatically.

How to update the system time periodically

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

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

How to update the time automatically when the kernel boots up

Copy the example shell script **fixtime** to directory /etc/init.d, and then perform the command **#chmod 755 fixtime** to change the shell script mode. Next, use vi editor to edit the file /etc/inittab. Add the following line to the bottom of the file:

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

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

Executing Scheduled Commands with Cron Daemon

Start Cron from the directory /etc/rc.d/rc. local. It will return immediately without adding a '&' to run in the background.

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

Cron wakes up every minute, and checks each command to see if it should be run in that minute. Modify the file **/etc/cron.d/crontab** to set up the scheduled applications. Crontab files are in the following formats:

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

The following example demonstrates how to use Cron.

<u>How to use Cron to update the system time and RTC time every day at 8:00.</u>

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

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

STEP2: Change mode of fixtime.sh

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

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

Add the following line to the end of crontab:

```
* 8 * * * root/homefixtime.sh
```

STEP4: Enable the cron daemon manually.

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

STEP5: Enable Cron when the system boots up.

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

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

4

Managing Communications

In this chapter, it explains how to configure the RTU controller’s communication functions.

The following topics are covered in this chapter:

Telnet/FTP

The Telnet and FTP Server service is enabled by default on the RTU controller. To enable or disable the Telnet/ftp server, users need to edit the file /etc/inetd.conf.

Enabling the Telnet/FTP Server

The following example shows the default content of the file /etc/inetd.conf.

telnet stream tcp nowait root /bin/telnetd
ftp stream tcp nowait root /bin/ftpd -l

Disabling the Telnet/FTP Server

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

DNS

To set up DNS client for the RTU controller, users need to edit three configuration files: /etc/hosts, /etc/resolv.conf, and /etc/nsswitch.conf.

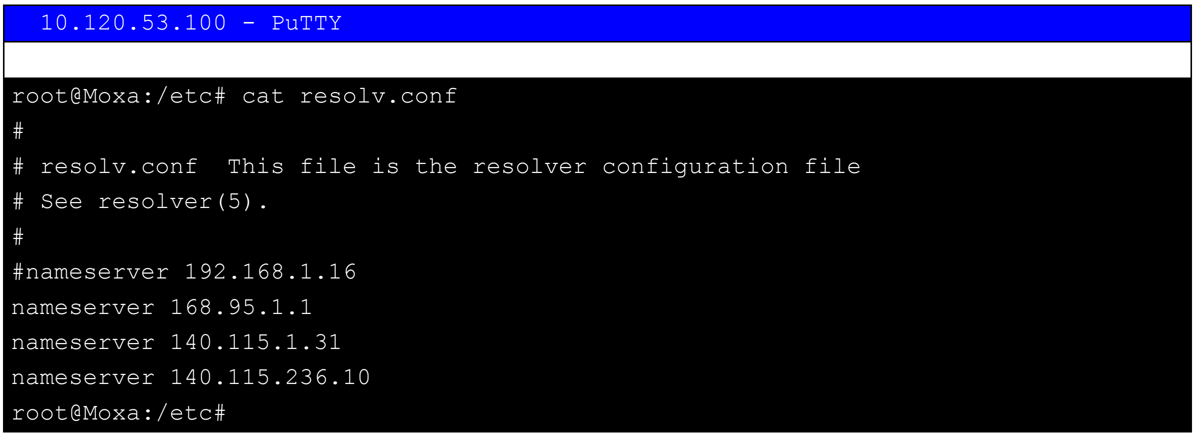
/etc/hosts

The **hosts** is the first file that the RTU controller system reads to resolve the host name of the remote IP address.

/etc/resolv.conf

The **resolv.conf** contains the remote DNS server's address in it. Ask the network administrator or service provider which DNS server address should be configured to this file. The DNS server's IP address is specified with the "nameserver" command. For example, add the following line to /etc/resolv.conf if the DNS server's IP address is 168.95.1.1:

nameserver 168.95.1.1



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

/etc/nsswitch.conf

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

Web Service—Apache

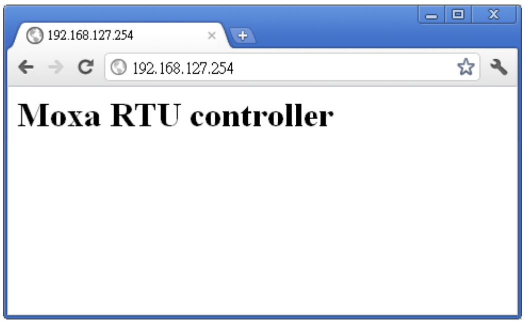
The Apache web server's main configuration file is **/etc/apache/conf/httpd.conf**, with the default homepage located at **/home/httpd/htdocs/index.html**. Save the customized homepage to the following directory:

/home/httpd/htdocs/

Save the customized CGI page to the following directory:

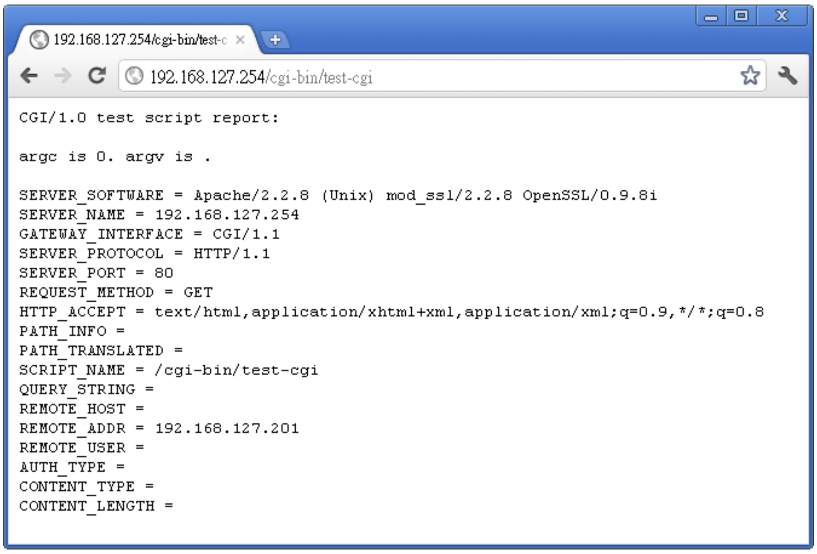
/home/httpd/cgi-bin/

Before modifying the homepage, use a browser from the host PC to perform a test to see if the Apache Web Server is working. Type the LAN IP address of the target RTU controller in the browser's address box to open the homepage. E.g., **http://192.168.127.254**



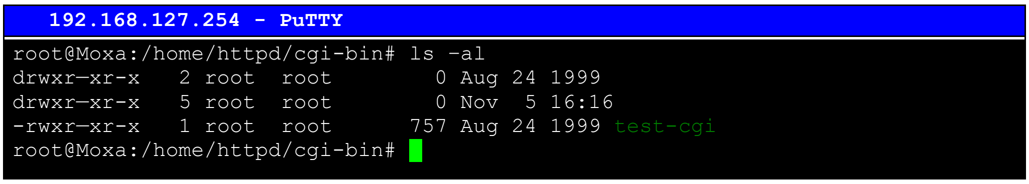
Moxa RTU controller

To open the default CGI page, type **http://192.168.127.254/cgi-bin/test-cgi** in the browser's address box.


192.168.127.254/cgi-bin/test-c
192.168.127.254/cgi-bin/test-cgi

```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = Apache/2.2.8 (Unix) mod_ssl/2.2.8 OpenSSL/0.9.8i
SERVER_NAME = 192.168.127.254
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.1
SERVER_PORT = 80
REQUEST_METHOD = GET
HTTP_ACCEPT = text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST =
REMOTE_ADDR = 192.168.127.201
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE =
CONTENT_LENGTH =
```

NOTE The CGI function is enabled by default. Modify the file **/etc/apache/conf/httpd.conf t** to disable it. While developing the customized CGI application, make sure the CGI file is executable.

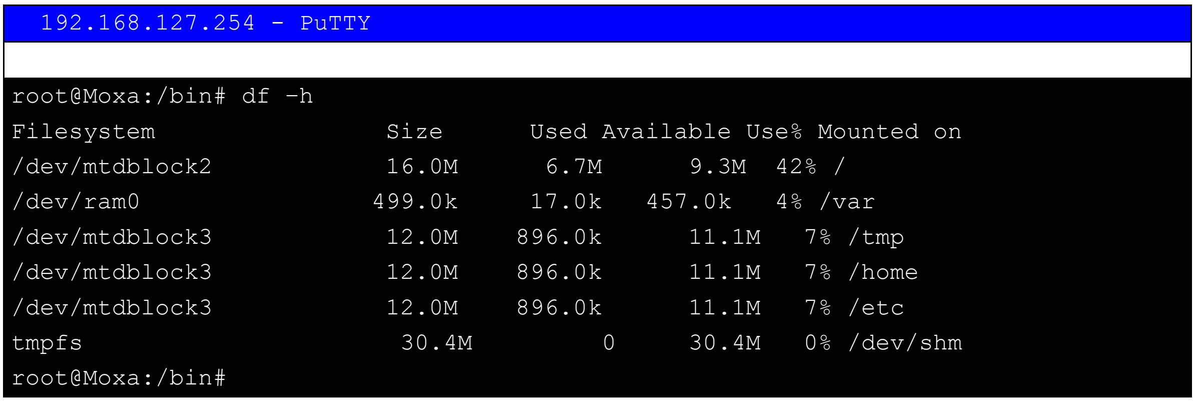


```
root@Moxa:/home/httpd/cgi-bin# ls -al
drwxr-xr-x    2 root   root         0 Aug 24 1999
drwxr-xr-x    5 root   root         0 Nov  5 16:16
-rwxr-xr-x    1 root   root       757 Aug 24 1999 test-cgi
root@Moxa:/home/httpd/cgi-bin#
```

Installing PHP for Apache Web Server

The Moxa C programming RTU controller supports the PHP option. However, since the PHP file is 3 MB, it is not installed by default. To install it, first make sure there is enough free space (at least 3 MB) on the target RTU controller's flash ROM.

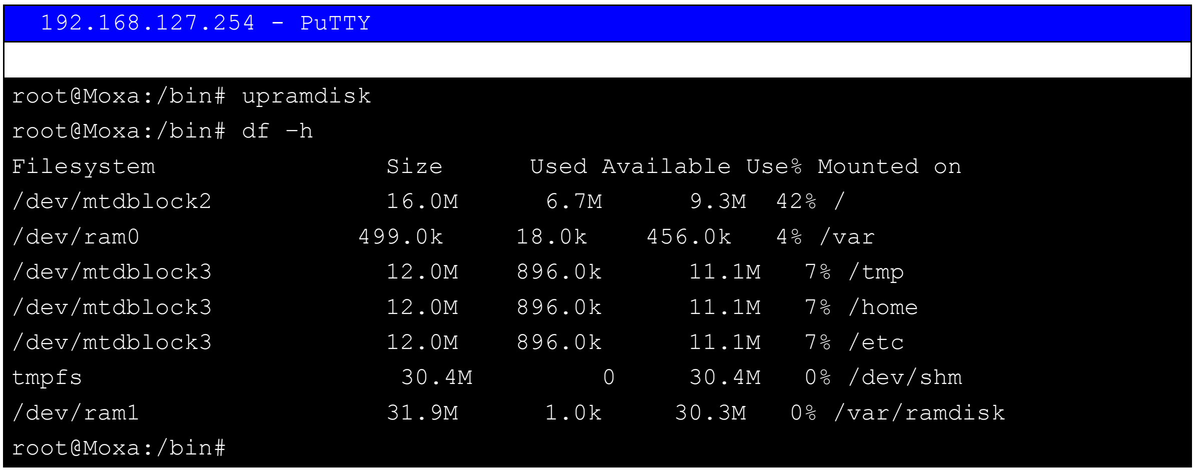
Step 1: Check if the system contains enough free space by performing the **df -h** command.



```
root@Moxa:/bin# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/mtdblock2            16.0M      6.7M      9.3M  42% /
/dev/ram0                499.0k     17.0k   457.0k   4% /var
/dev/mtdblock3            12.0M    896.0k      11.1M   7% /tmp
/dev/mtdblock3            12.0M    896.0k      11.1M   7% /home
/dev/mtdblock3            12.0M    896.0k      11.1M   7% /etc
tmpfs                      30.4M         0      30.4M   0% /dev/shm
root@Moxa:/bin#
```

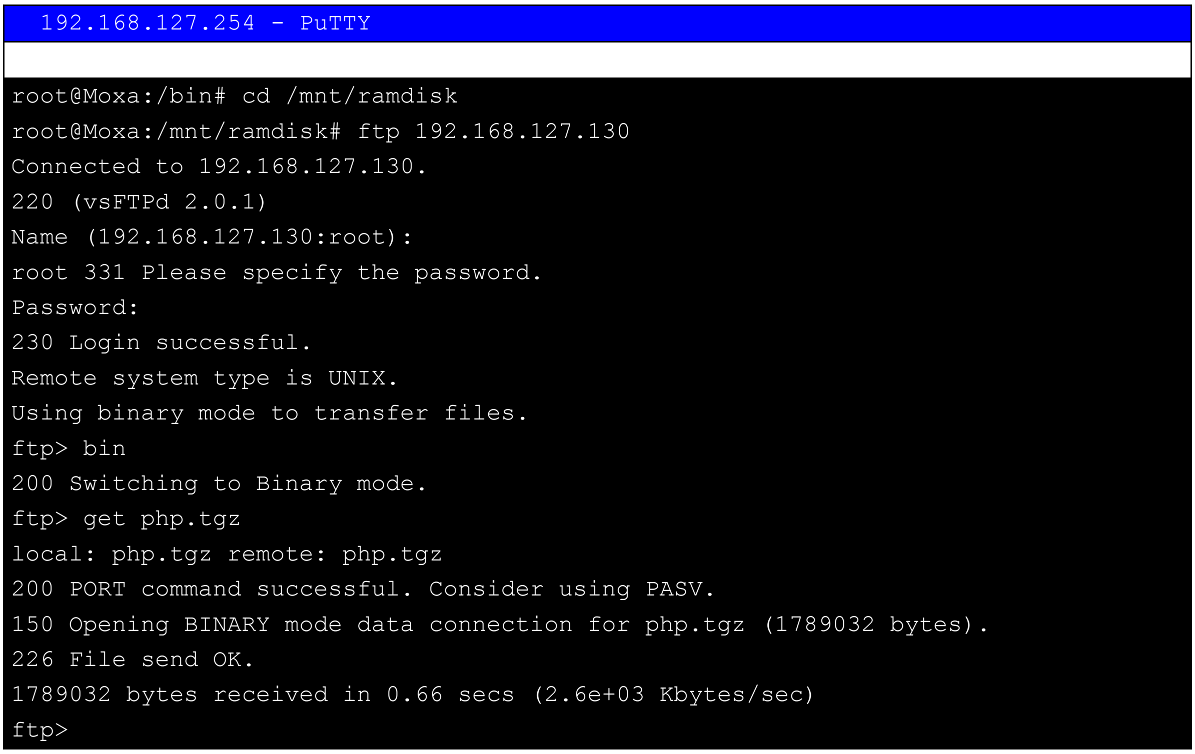
Checking the **/dev/mtdblock3** to see if the available space is over 3 MB.

Step 2: Perfrom **upramdisk** command to create a temporary RAM disk to download and save the package.



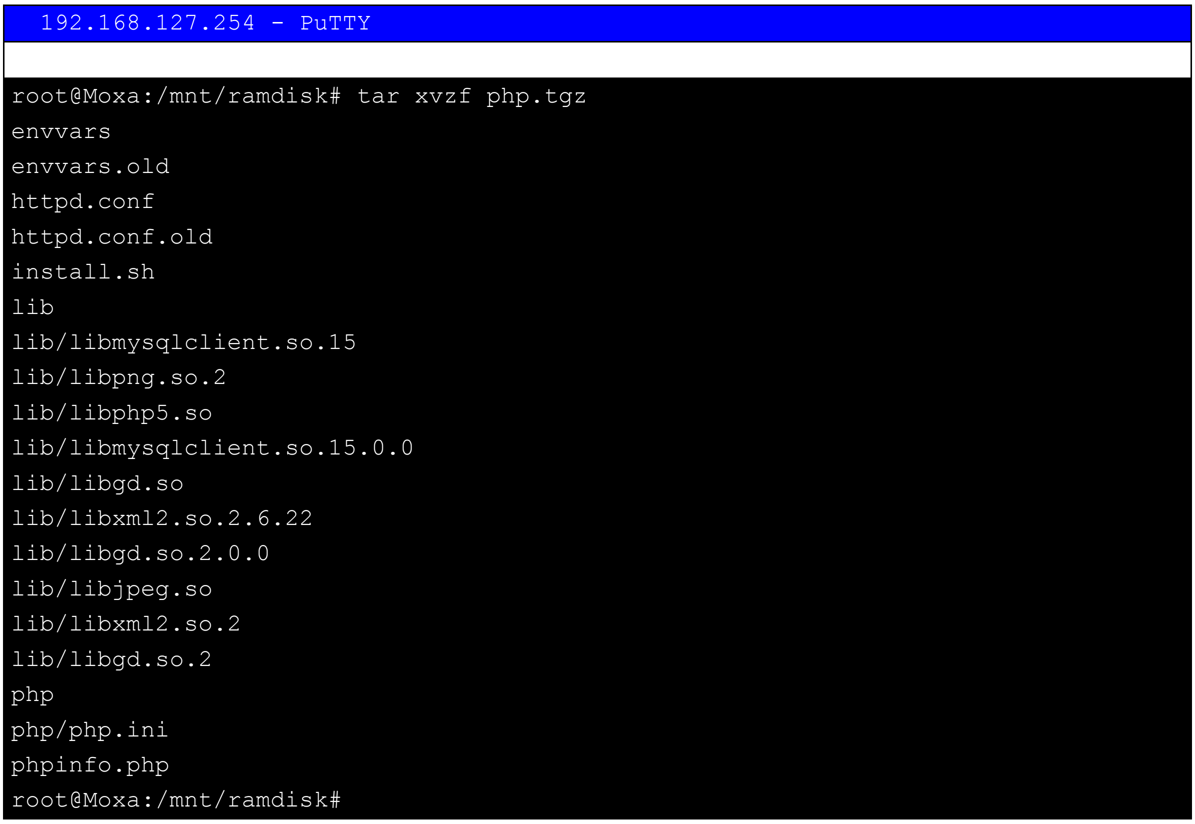
```
root@Moxa:/bin# upramdisk
root@Moxa:/bin# df -h
Filesystem                Size      Used Available Use% Mounted on
/dev/mtdblock2            16.0M      6.7M      9.3M  42% /
/dev/ram0               499.0k     18.0k    456.0k   4% /var
/dev/mtdblock3            12.0M    896.0k      11.1M   7% /tmp
/dev/mtdblock3            12.0M    896.0k      11.1M   7% /home
/dev/mtdblock3            12.0M    896.0k      11.1M   7% /etc
tmpfs                      30.4M         0      30.4M   0% /dev/shm
/dev/ram1                 31.9M      1.0k     30.3M   0% /var/ramdisk
root@Moxa:/bin#
```

Step 3: Download the PHP package. Users can locate the package in Software and Document CD/Software/php/php.tgz



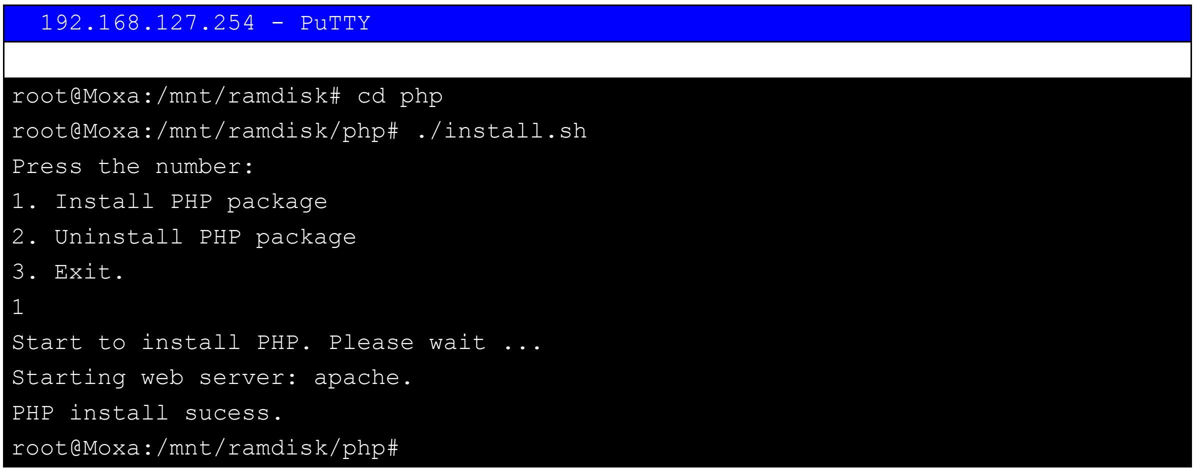
```
root@Moxa:/bin# cd /mnt/ramdisk
root@Moxa:/mnt/ramdisk# ftp 192.168.127.130
Connected to 192.168.127.130.
220 (vsFTPd 2.0.1)
Name (192.168.127.130:root):
root 331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> bin
200 Switching to Binary mode.
ftp> get php.tgz
local: php.tgz remote: php.tgz
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for php.tgz (1789032 bytes).
226 File send OK.
1789032 bytes received in 0.66 secs (2.6e+03 Kbytes/sec)
ftp>
```

Step 4: Unzip the package. Perform the command "tar xvzf php.tar.gz".



```
root@Moxa:/mnt/ramdisk# tar xvzf php.tgz
envvars
envvars.old
httpd.conf
httpd.conf.old
install.sh
lib
lib/libmysqlclient.so.15
lib/libpng.so.2
lib/libphp5.so
lib/libmysqlclient.so.15.0.0
lib/libgd.so
lib/libxml2.so.2.6.22
lib/libgd.so.2.0.0
lib/libjpeg.so
lib/libxml2.so.2
lib/libgd.so.2
php
php/php.ini
phpinfo.php
root@Moxa:/mnt/ramdisk#
```

Step 5: Change directory to "php". Run "install.sh" and select "Install PHP Package".



```
root@Moxa:/mnt/ramdisk# cd php
root@Moxa:/mnt/ramdisk/php# ./install.sh
Press the number:
1. Install PHP package
2. Uninstall PHP package
3. Exit.
1
Start to install PHP. Please wait ...
Starting web server: apache.
PHP install sucess.
root@Moxa:/mnt/ramdisk/php#
```

Step 6: Open the page with the host PC's browser (e.g. http://192.168.127.254/phpinfo.php).

If you want to uninstall PHP, follow step 2 to 5, and select the Uninstall option.

IPTABLES

IPTABLES is an administrative tool for setting up, maintaining, and inspecting the RTU controller'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 specific type of packet. Each rule specifies what to do with a matching packet. A rule (such as a jump to a user-defined chain in the same table) is called a "target."

The Moxa C programming RTU controller supports 3 types of IPTABLES table: **Filter** tables, **NAT** tables, and **Mangle** tables:

A. **Filter Table—**includes three chains:

INPUT chain

OUTPUT chain

FORWARD chain

B. **NAT Table**—includes three chains:

PREROUTING chain—transfers the destination IP address (DNAT)

POSTROUTING chain—works after the routing process and before the Ethernet device process to transfer the source IP address (SNAT)

OUTPUT chain—produces local packets

sub-tables

Source NAT (SNAT)—changes the first source packet IP address
Destination NAT (DNAT)—changes the first destination packet IP address
MASQUERADE—a special form for SNAT. If one host can connect to Internet, then other computers that connect to this host can connect to the Internet when the computer does not have an actual IP address.
REDIRECT—a special form of DNAT that re-sends packets to a local host independent of the destination IP address.

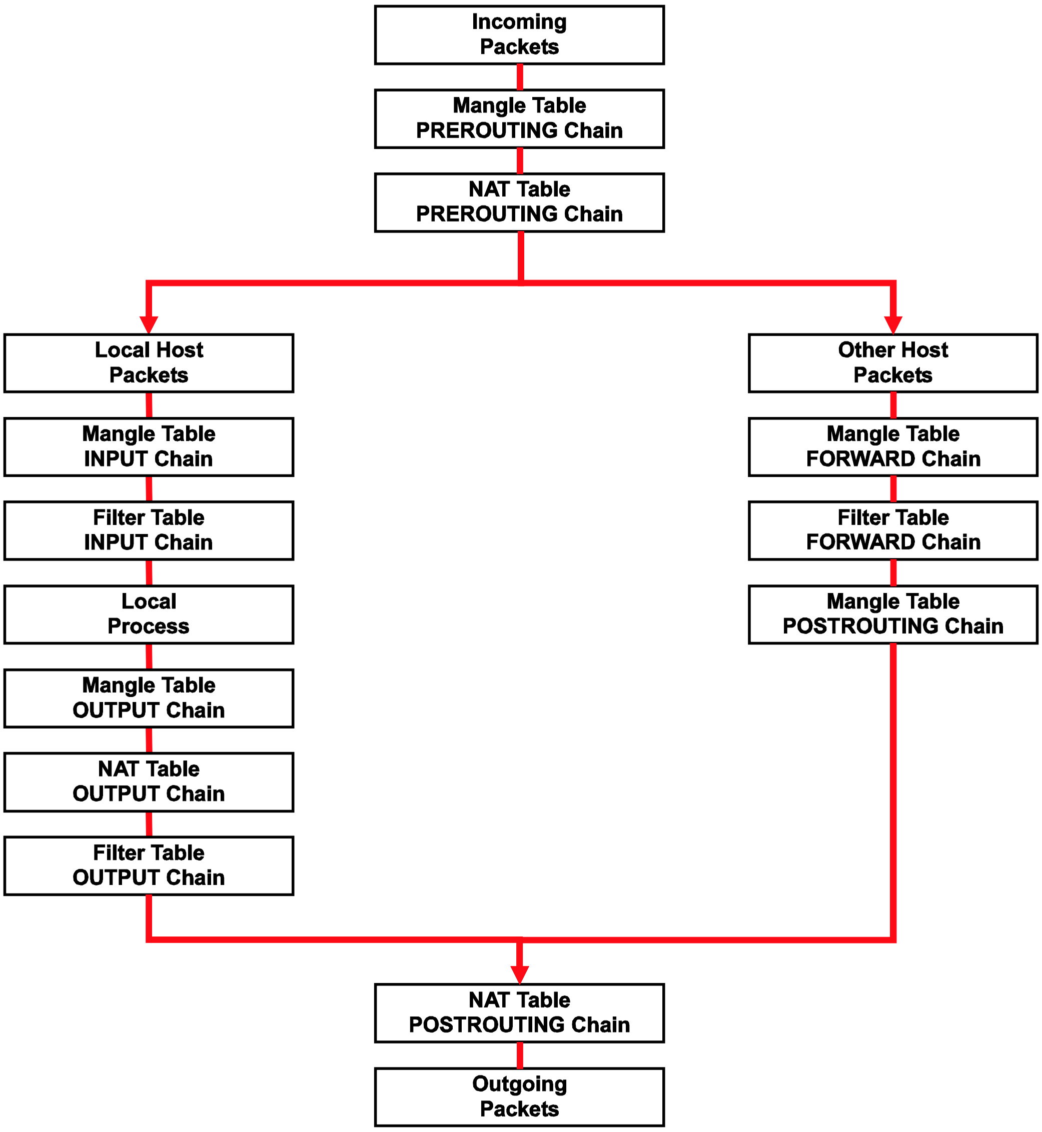
C. **Mangle Table—includes two chains**

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

OUTPUT chain—processes packets after the routing process.

It has three extensions—TTL, MARK, TOS.

The following figure shows the IPTABLES hierarchy.


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

The Moxa C programming RTU controller supports the following sub-modules. Be sure to use the module that matches the real application.

ip_conntrack	ipt_MARK	ipt_ah	ipt_state
ip_conntrack_ftp	ipt_MASQUERADE	ipt_esp	ipt_tcpmss
ipt_conntrack_irc	ipt_MIRROT	ipt_length	ipt_tos
ip_nat_ftp	ipt_REDIRECT	ipt_limit	ipt_ttl
ip_nat_irc	ipt_REJECT	ipt_mac	ipt_unclean
ip_nat_snmp_basic	ipt_TCPMSS	ipt_mark	
ip_queue	ipt_TOS	ipt_multiport	
ipt_LOG	ipt_ULOG	ipt_owner	

NOTE The Moxa C programming RTU controller Do NOT support IPV6 and ipchains.

The basic syntax to enable and load an IPTABLES module is as follows:

```
#lsmod
#modprobe ip_tables
#modprobe iptable_filter
```

Use the **lsmod** command to check if the ip_tables module has already been loaded in the Moxa RTU unit. Use the **modprobe** command to insert and enable the module.

Use the following command to load the modules (iptable_filter, iptable_mangle, iptable_nat):

#modprobe iptable_filter

NOTE IPTABLES plays the role of packet filtering or NAT. Be careful when setting up the IPTABLES rules. If the rules are not correct, remote hosts that are connected via a LAN or PPP may deny access. It is strongly recommended to use the serial console to set up the IPTABLES.
Click on the following links for more information about iptables.
http://www.linuxguruz.com/iptables/
http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html

Since the IPTABLES command is very complex, illustrating the IPTABLES syntax is divided 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 Moxa RTU.
OUTPUT: For locally-generated packets.

FORWARD: For packets routed out through the Moxa RTU.

PREROUTING: To alter packets as soon as they come in.

POSTROUTING: To alter packets as they are about to be sent out.

Examples:

#iptables –P INPUT DROP

#iptables –P OUTPUT ACCEPT

#iptables –P FORWARD ACCEPT

#iptables –t nat –P PREROUTING ACCEPT

#iptables –t nat –P OUTPUT ACCEPT

#iptables -t nat –P POSTROUTING ACCEPT

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

Append or Delete Rules

Usage:

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

-A: Append one or more rules to the end of the selected chain.

-I: Insert one or more rules in the selected chain as the given rule number.

-i: Name of an interface via which a packet is going to be received.

-o: Name of an interface via which a packet is going to be sent.

-p: The protocol of the rule or of the packet to check.

-s: Source address (network name, host name, network IP address, or plain IP address).

--sport: Source port number.

-d: Destination address.

--dport: Destination port number.

-j: Jump target. Specifies the target of the rules; i.e., how to handle matched packets. For example, ACCEPT the packet, DROP the packet, or LOG the packet.

Examples:

Example 1: Accept all packets from lo interface.

# iptables –A INPUT –i lo –j ACCEPT

Example 2: Accept TCP packets from 192.168.0.1.

# iptables –A INPUT –i 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 Moxa RTU's port 137, 138, 139

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

Example 7: 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
```

NOTE: In Example 7, remember to issue the command **#modprobe ipt_mac** first to load module ipt_mac.

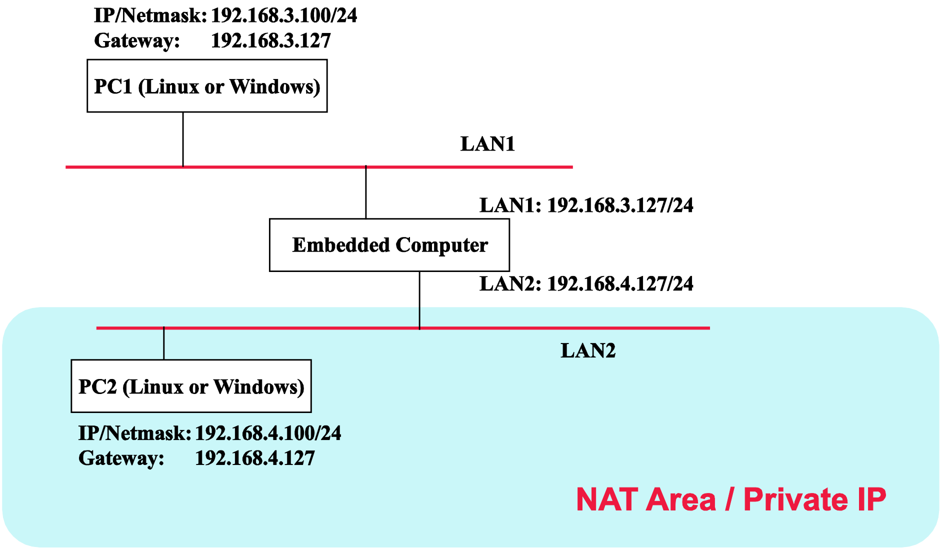
NAT

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

NOTE Click on the following links for more information about iptables and NAT:
http://www.netfilter.org/documentation/HOWTO/NAT-HOWTO.html

NAT Example

The IP address of LAN1 is changed to 192.168.127.254 (you will need to load the module ipt_MASQUERADE):



1. #echo 1 > /proc/sys/net/ipv4/ip_forward
2. #modprobe ip_tables
3. #modprobe iptable_filter
4. #modprobe ip_conntrack
5. #modprobe iptable_nat
6. #modprobe ipt_MASQUERADE
7. #iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 192.168.127.254
8. #iptables -t nat -A POSTROUTING -o eth0 -s 192.168.127.0/24 -j MASQUERADE

Enabling NAT at Bootup

In most of the situations, you will want to use a simple shell script to enable NAT when the RTU controller boots up. The following script is an example.

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

Dial-up Service—PPP

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

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

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

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

Example 1: Connecting to a PPP Server over a Simple Dial-up Connection

The following command is used to connect to a PPP server by modem. Use this command for old ppp servers that prompt for a login name and password. Note that debug and default route 192.1.1.17 are optional.

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

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

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

The pppd options are described below:

connect ‘chat etc...’

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

-v

verbose mode; log what we do to syslog

“ “

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

ATDT5551212

Dial the modem, and then ...

CONNECT

Wait for an answer.

“ “

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

login: username word: password

Log in with the username and password.

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

/dev/

Specify the callout serial port.

115200

The baudrate.

debug

Log status in syslog.

crtscts

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

modem

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

defaultroute

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

192.1.1.17

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

Example 2: Connecting to a PPP Server over a Hard-wired Link

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

```
#pppd connect ‘chat –v “ “ “ “ ‘ noipdefault /dev/ttyM0 19200 crtscts “
```

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

```
#pppd connect ‘chat –v “ “ “ “ ‘ user root password root noipdefault
/dev/ttyM0 19200 crtscts
```

How to Check the Connection

Once a PPP connection is set up, you can follow the steps to test the connection. First, type:

```
/sbin/ifconfig
```

You should be able to see all the network interfaces that are UP. Check “ppp0” interface, and you should recognize the first IP address as your own, and the “P-t-P address” (or point-to-point address) the address of your server. Here’s what it looks like on the RTU controller:

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

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

Now, type:

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

where z.z.z.z is the address of your name server. The response may look like:

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

Try typing:

netstat –nr

It should show three routes, similar as the following:

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

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

Setting up a Machine for Incoming PPP Connections

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

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

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

***   *   “”   ***

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

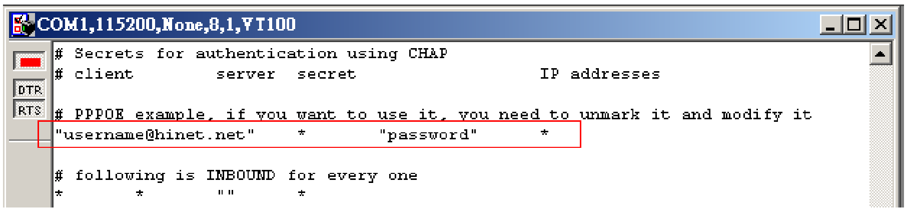
The following example does not check the username and password:

pppd/dev/ttyM0 115200 crtscts modem 192.168.16.1:192.168.16.2

PPPoE

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

   “username@hinet.net”*“password”*

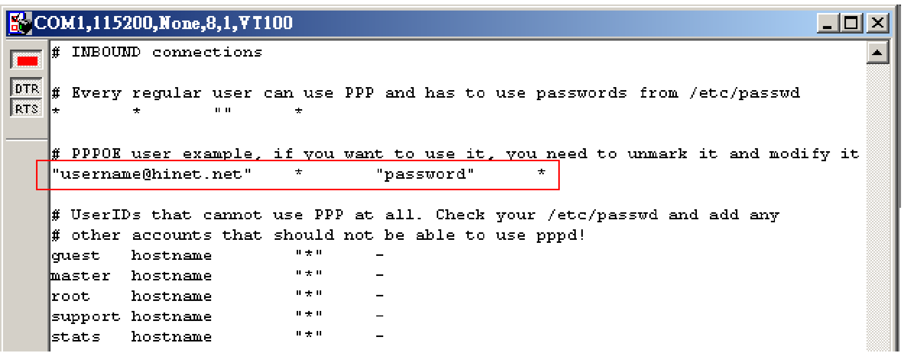


```
# Secrets for authentication using CHAP
# client        server   secret                  IP addresses

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

# following is INBOUND for every one
*       *       ""      *
```

“username@hinet.net” is the username obtained from the ISP to log in to the ISP account. **“password”** is the corresponding password for the account.

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

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

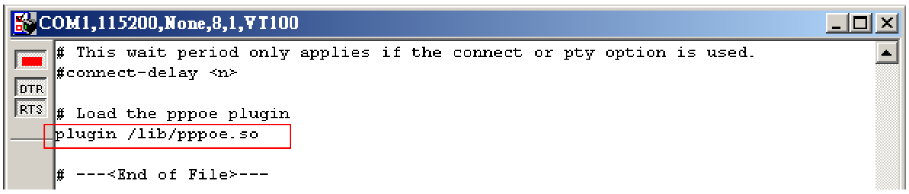


```
# INBOUND connections

# Every regular user can use PPP and has to use passwords from /etc/passwd
*         *         ""        *

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

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

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

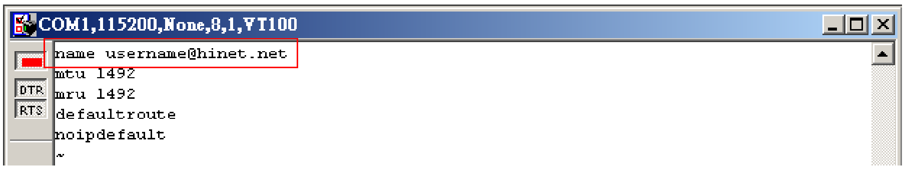
5. Edit the file **/etc/ppp/options** and add the following line:

   plugin pppoe



```
# This wait period only applies if the connect or pty option is used.
#connect-delay <n>

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

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

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



```
name username@hinet.net
mtu 1492
mru 1492
defaultroute
noipdefault
~
```

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

7. Set up DNS

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

   nameserver ip_addr_of_first_dns_server

   nameserver ip_addr_of_second_dns_server

   For example:

   nameserver 168.95.1.1

   nameserver 139.175.10.20

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

   pppd eth0

   The eth0 is what is connected to the ADSL modem LAN port. The example above uses LAN1.

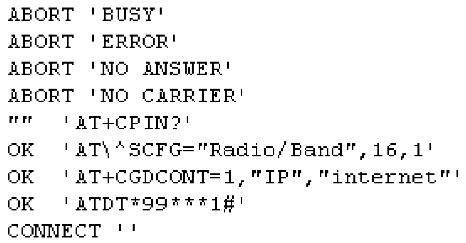
9. Type **ifconfig ppp0** to check if the connection is OK or has failed. If the connection is OK, you will see information about the ppp0 setting for the IP address. Use ping to test the IP.

10. If you want to disconnect it, use the kill command to kill the pppd process.

PPP over Cellular

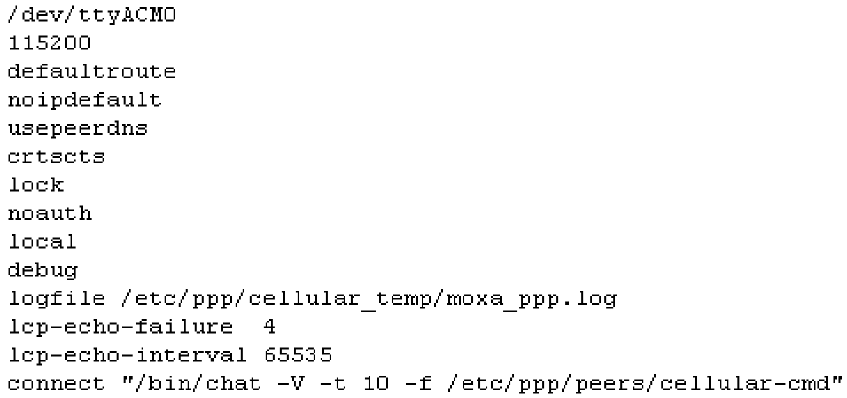
It is recommend to use the callular API in the RTU controller's Toolchain; however, the pppd also supports cellular network with following examples,

Create a chat configuration "**cellular-cmd"** at **/etc/ppp/peers**



```
ABORT 'BUSY'
ABORT 'ERROR'
ABORT 'NO ANSWER'
ABORT 'NO CARRIER'
""   'AT+CPIN?'
OK   'AT\^SCFG="Radio/Band",16,1'
OK   'AT+CGDCONT=1,"IP","internet"'
OK   'ATDT*99***1#'
CONNECT ''
```

Create a pppd configuration "**cellular-connect**" at **/etc/ppp/peers**



```
/dev/ttyACM0
115200
defaultroute
noipdefault
usepeerdns
crtscts
lock
noauth
local
debug
logfile /etc/ppp/cellular_temp/moxa_ppp.log
lcp-echo-failure  4
lcp-echo-interval 65535
connect "/bin/chat -V -t 10 -f /etc/ppp/peers/cellular-cmd"
```

Perform the command

#pppd call cellular-connect

NFS (Network File System)

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

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

Setting up the RTU Controller as an NFS Client

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

1. To know the NFS Server's shared directory.
2. Establish a mount point on the NFS Client site.
3. Mount the remote directory to a local directory.

#mkdir –p /home/nfs/public

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

Example:

#mount –t nfs 192.168.127.100:/home/public /home/nfs/public

Mail

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

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

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

Example:

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

-s: The mail subject.

-f: Sender's mail address

-S: SMTP server IP address

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

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

OpenVPN

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

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

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

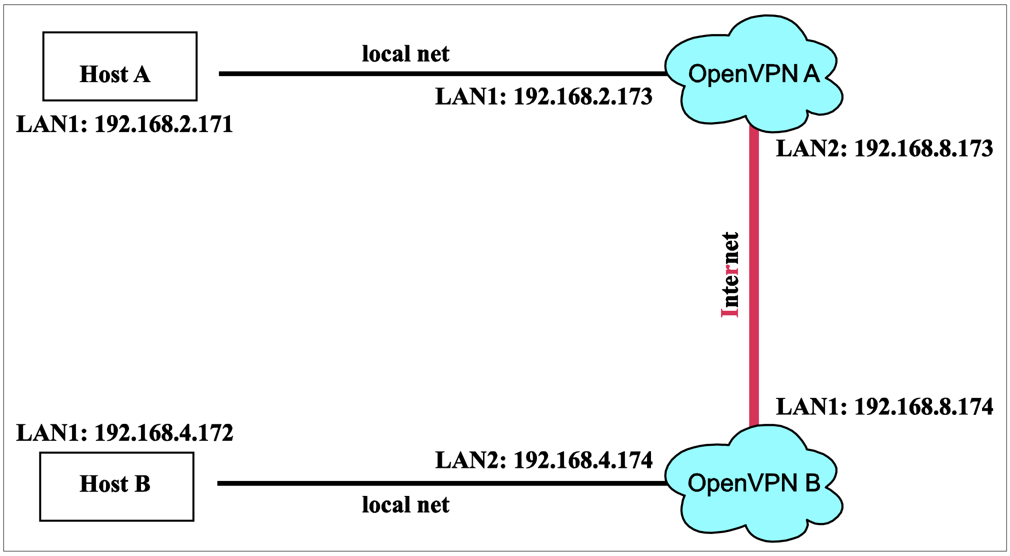
On each OpenVPN machine, you should generate a working directory, such as **/etc/openvpn**, where script files and key files reside. Once established, all operations will be performed in that directory.

NOTE Network interface definition may be different in in different product lines. For ioLogik W5348 series, eth0 = ppp0(WAN)、eth1 = eth0(LAN)

In the following examples, the OpenVPN A and B represent two Moxa RTU controllers. Note the network interface definition may be different in different product lines. For ioLogik W5348 series, "eth0" in the example will be ppp0(cellular WAN), and the "eth1" in the example will be the eth0(LAN)

Setup 1: Ethernet Bridging for Private Networks on Different Subnets

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



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

# openvpn --genkey --secret secrouter.key

Copy the file that is generated to the OpenVPN machine.

2. Generate a script file named **openvpn-bridge** on each OpenVPN machine. This script reconfigures interface "eth1" as IP-less, creates logical bridge(s) and TAP interfaces, loads modules, enables IP forwarding, etc.

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

#!/bin/sh
iface=eth1 # defines the internal interface
maxtap=`expr 1` # defines the number of tap devices. I.e., # of tunnels
IPADDR=
NETMASK=
BROADCAST=
# it is not a great idea but this system doesn't support
# /etc/sysconfig/network-scripts/ifcfg-eth1
ifcfg_vpn()
{
while read f1 f2 f3 f4 r3
do
  if [ "$f1" = "iface" -a "$f2" = "$iface" -a "$f3" = "inet" -a "$f4" = "static" ];then
    i=`expr 0`
    while :
    do
      if [ $i -gt 5 ]; then
        break
      fi
      i=`expr $i + 1`
      read f1 f2
      case "$f1" in
        address ) IPADDR=$f2
```

```
            ;;
          netmask ) NETMASK=$f2
            ;;
          broadcast ) BROADCAST=$f2
            ;;
        esac
      done
            break
    fi
done < /etc/network/interfaces
}
# get the ip address of the specified interface
mname=
module_up()
{
  oIFS=$IFS
  IFS='
'
  FOUND="no"
  for LINE in `lsmod`
  do
    TOK=`echo $LINE | cut -d' ' -f1`
    if [ "$TOK" = "$mname" ]; then
      FOUND="yes";
      break;
    fi
  done
  IFS=$oIFS
  if [ "$FOUND" = "no" ]; then
    modprobe $mname
  fi
}
start()
{
ifcfg_vpn
if [ ! \( -d "/dev/net" \) ]; then
  mkdir /dev/net
fi
if [ ! \( -r "/dev/net/tun" \) ]; then
  # create a device file if there is none
  mknod /dev/net/tun c 10 200
fi
# load modules "tun" and "bridge"
mname=tun
module_up
mname=bridge
module_up
# create an ethernet bridge to connect tap devices, internal interface
brctl addbr br0
brctl addif br0 $iface
# the bridge receives data from any port and forwards it to other ports.
i=`expr 0`
while :
do
  # generate a tap0 interface on tun
```

```
   openvpn --mktun --dev tap${i}
   # connect tap device to the bridge
   brctl addif br0 tap${i}
   # null ip address of tap device
   ifconfig tap${i} 0.0.0.0 promisc up
   i=`expr $i + 1`
   if [ $i -ge $maxtap ]; then
      break
   fi
done
# null ip address of internal interface
ifconfig $iface 0.0.0.0 promisc up
# enable bridge ip
ifconfig br0 $IPADDR netmask $NETMASK broadcast $BROADCAST
ipf=/proc/sys/net/ipv4/ip_forward
# enable IP forwarding
echo 1 > $ipf
echo “ip forwarding enabled to”
cat $ipf
}
stop() {
echo “shutdown openvpn bridge.”
ifcfg_vpn
i=`expr 0`
while :
do
   # disconnect tap device from the bridge
   brctl delif br0 tap${i}
   openvpn --rmtun --dev tap${i}
   i=`expr $i + 1`
   if [ $i -ge $maxtap ]; then
      break
   fi
done
brctl delif br0 $iface
brctl delbr br0
ifconfig br0 down
ifconfig $iface $IPADDR netmask $NETMASK broadcast $BROADCAST
killall -TERM openvpn
}
case “$1” in
   start)
      start
      ;;
   stop)
      stop
      ;;
   restart)
      stop
      start
      ;;
   *)
      echo “Usage: $0 [start|stop|restart]”
      exit 1
esac
```

```
exit 0
#--------------------------------- end ----------------------------
Run the shell command to start, stop, or restart the OpenVPN
# sh /etc/openvpn/openvpn-bridge start
# sh /etc/openvpn/openvpn-bridge stop
# sh /etc/openvpn/openvpn-bridge restart
```

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

```
# point to the peer
remote 192.168.8.174
dev tap0
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
up /etc/openvpn/A-tap0-br.sh
#----------------------------------Start------------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.4.0 netmask 255.255.255.0 dev br0
#---------------------------------- end ------------------------------
```

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

```
# point to the peer
remote 192.168.8.173
dev tap0
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5 tun-mtu 1500
tun-mtu-extra 64
ping 40
up /etc/openvpn/B-tap0-br.sh
#----------------------------------Start------------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 dev br0
#---------------------------------- end ------------------------------
```

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

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

4. Start both of OpenVPN peers,

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

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

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

```
# route
```

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

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

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

route add –net 192.168.4.0 netmask 255.255.255.0 dev eth0

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

route add –net 192.168.2.0 netmask 255.255.255.0 dev eth0

Now ping Host B from Host A by typing:

ping 192.168.4.174

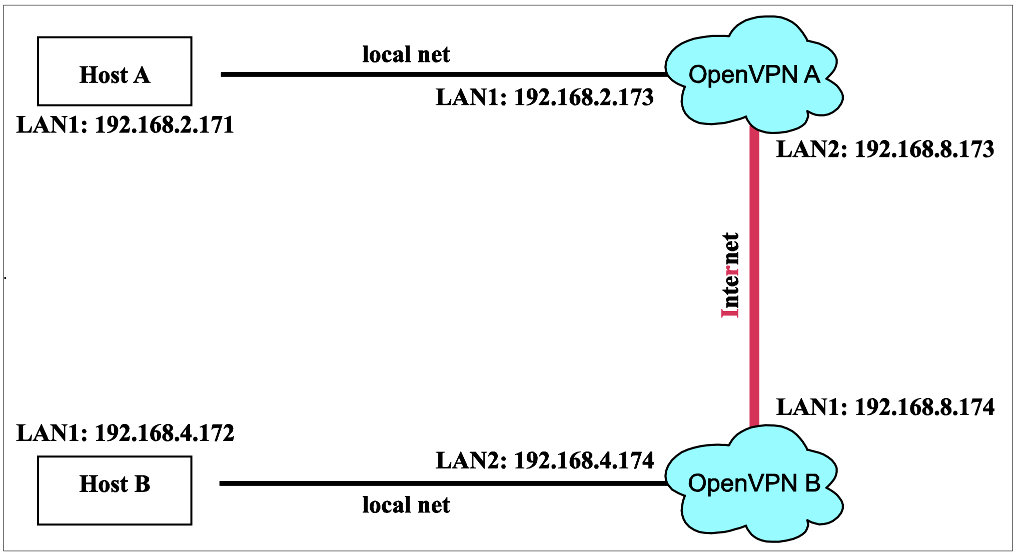
A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 5000 between OpenVPN peers.

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

# killall -TERM openvpn

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

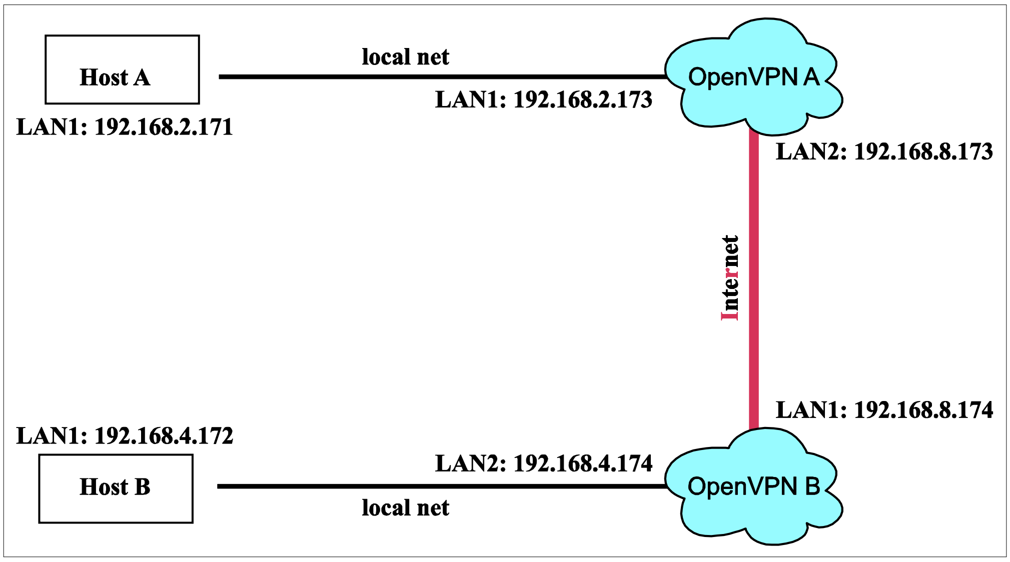
1. Set up four machines as shown in the following diagram:



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

Setup 3: Routed IP

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



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

```
# point to the peer
remote 192.168.8.174
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
ifconfig 192.168.2.173 192.168.4.174
up /etc/openvpn/A-tun.sh
#----------------------------------Start------------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.4.0 netmask 255.255.255.0 gw $5
#---------------------------------- end ------------------------------
```

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

```
remote 192.168.8.173
dev tun
secret /etc/openvpn/secrouter.key
cipher DES-EDE3-CBC
auth MD5
tun-mtu 1500
tun-mtu-extra 64
ping 40
ifconfig 192.168.4.174 192.168.2.173
up /etc/openvpn/B-tun.sh
#----------------------------------Start------------------------------
#!/bin/sh
# value after "-net" is the subnet behind the remote peer
route add -net 192.168.2.0 netmask 255.255.255.0 gw $5
#---------------------------------- end ------------------------------
```

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

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

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

# route

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

5

Tool Chains for Application Development

In this chapter we describe how to install a tool chain on the host computer to develop user applications. In addition, the process for performing cross-platform development and debugging is also introduced. For clarity, the MOXA C Programming RTU controller is called a target system.

The following topics are covered in this chapter:

- ❒ **Linux Tool Chain**
    - ➢ Installing the Linux Tool Chain
    - ➢ Compiling Applications
    - ➢ On-Line Debugging with GDB

Linux Tool Chain

The Linux tool chain contains a suite of cross compilers and other tools, as well as the libraries and header files that are necessary to compile your applications. These tool chain components must be installed on a Linux-based host computer (PC). The following Linux distributions can be used to install the tool chain.

- Fedora Core 6 (on x86)
- Mandrake 8.1 (on x86)
- Red Hat 7.3, 8.0, 9.0 (on x86)
- SuSE 7.3 (on x86)
- YellowDog 2.1 (on PowerPC)
- Solaris 7 and 8 (on Sparc)
- Debian 3.1, 4.0 (on x86)
- Ubuntu 9.04. (see note)

NOTE Ubuntu users will need to prepare their system by entering the following commands:

```
apt-get install libncurse5-dev
mkdir /mnt/ramdisk
```

Disregard the "[==: unexpected operator" warning when installing the tool chain.

Installing the Linux Tool Chain

The tool chain requires approximately 1 GB of hard disk space. To install the tool chain, follow the steps.

1. Insert the Documentation and Software CD into your PC, and then enter the following commands:
   ```
   #mount /dev/cdrom /mnt/cdrom
   #sh /mnt/cdrom/Software/toolchain/arm-linux_3.3.2_V1.X_BuildXXXXXXXX.sh
   ```
2. Wait for the installation process to complete. This should take a few minutes.
3. Add the directory **/usr/local/arm-linux/bin** to your path. You can do this in the current login by issuing the following commands:
   ```
   #export PATH="/usr/local/arm-linux/bin:$PATH"
   ```

Alternatively, the same commands can be added to **$HOME/.bash_profile** to make it effective for all login sessions.

Compiling Applications

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

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

Most of the cross compiler tools are the same as their native compiler counterparts, only with an additional prefix that specifies the target system. The prefix is "i386-linux-" for x86 environments and "arm-linux-" for MOXA RTU controllers. For example, "gcc" is the native C compiler, whereas "arm-linux-gcc" is the cross C compiler for the ARM-based ioLogik W5348-C/ioPAC 8020-C series.

Moxa provides cross compiler tools for the following native compilers. Simply add the "arm-linux-" prefix.

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

On-Line Debugging with GDB

The tool chain also provides an on-line debugging mechanism to help you develop your program. Before starting a debugging session, add the option -**ggdb** when you compile the program. A debugging session runs on a client-server architecture on which the server **gdbserver** is installed on the target system and the client **ddd** is installed on the host computer. In the following instructions, we assume that you have uploaded a program named **hello-debug** to the target system and wish to debug this program.

1. Log on to the target system and run the debugging server program.

   ```
   #gdbserver 192.168.4.142:2000 hello-debug
   Process hello-debug created; pid=38
   ```

   This tells the debugging server to listen for connections on network port 2000 of the network interface 192.168.4.142 of the target system. The name of the program to be debugged is indicated after the network port. Additional arguments can be added after the program name as needed.
2. In the host computer, switch to the directory that contains the program source.

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

3. Execute the client program.

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

4. Enter the following command at the GDB, ddd command prompt.

   ```
   >> target remote 192.168.4.142:2000
   ```

   The command produces a line of output on the target system console, similar to the following.

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

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

5. Set a break point in the main function by double clicking or entering **b main** on the command line.
6. Click the **cont** button.

6

Programmer's Guide

This chapter includes important information for programmers.

The following topics are covered in this chapter:

❒ **Flash Memory**

❒ **C Library**

- APIs

Flash Memory

Partition sizes are hard coded into the kernel binary. The flash memory map is shown in the following table.

	Total Size	Contents	Location	Access
System Space	20 MB	Boot Loader	0x80000000 to 0x80080000	Read Only
		Linux Kernel	0x80080000 to 0x80400000	
		Root File System (JFFS2)	0x80400000 to 0x81400000	
User Space	12 MB	User directory (JFFS2)	0x81400000 to 0x82000000	Read/Write

If the user file system is incorrect, the kernel will change the root file system to the kernel and use the default Moxa file system. To finish the boot process, run the init program.

NOTE

1. The user file system is a complete file system. Users can create and delete directories and files (including source code and executable files) as needed.
2. Users can create the user file system on the host PC or the target platform and copy it to the ioLogik W5348-C series or the ioPAC 8020-C series.

C Library

The ioLogik W5348-C series and ioPAC 8020-C series both support control devices with Moxa APIs. Users will need to include **libmoxa_pgm.h** to use the following Moxa APIs.

APIs

For details on APIs, refer to Chapter 2 of “Developer’s Guide for Moxa RTU Controllers.”

7

Software Lock

“Software Lock” is an innovative technology developed by Moxa. It can be adopted by a system integrator or developer to protect his applications from being copied. An application is compiled into a binary format bound to the embedded computer and the operating system (OS) that the application runs on. As long as one obtains it from the computer, he/she can install it into the same hardware and the same operating system. The add-on value created by the developer is thus lost.

Moxa used data encryption to develop this protection mechanism for your applications. The binary file associated with each of your applications needs to undergo an additional encryption process after you have developed it. The process requires you to install an encryption key in the target computer.

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

   #setkey ABigKey

   NOTE: set an empty string to clear the encryption key in the target computer by:

   #setkey “”
2. Develop and compile your program in the development PC.
3. In the development PC, run the utility program ‘binencryptor’ to encrypt your program with an encryption key.

   #binencryptor yourProgram ABigKey
4. Upload the encrypted program file to the target computerby FTP or NFS and test the program.

   The encryption key is a computer-wise key. That is to say, a computer has only one key installed. Running the program ‘setkey’ multiple times overrides the existing key.

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

   This mechanism also allows the computer with an encryption key to bypass programs that are not encrypted. Therefore, in the development phase, you can develop your programs and test them in the target computer cleanly.

A

System Commands

Linux normal command utility collection:

File manager

1.	cp	copy file
2.	ls	list file
3.	ln	make symbolic link file
4.	mount	mount and check file system
5.	rm	delete file
6.	chmod	change file owner & group & user
7.	chown	change file owner
8.	chgrp	change file group
9.	sync	sync file system, let system file buffer be saved to hardware
10.	mv	move file
11.	pwd	display now file directly
12.	df	list now file system space
13.	mkdir	make new directory
14.	rmdir	delete directory

Editor

1.	vi	text editor
2.	cat	dump file context
3.	zcat	compress or expand files
4.	grep	search string on file
5.	cut	get string on file
6.	find	find file where are there
7.	more	dump file by one page
8.	test	test file exist or not
9.	sleep	sleep (seconds)
10.	echo	echo string

Network

1.	ping	ping to test network
2.	route	routing table manager
3.	netstat	display network status

4.	ifconfig	set network ip address
5.	tracerout	trace route
6.	tftp	
7.	telnet	
8.	ftp	

Process

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

Other

1.	3dmesg	dump kernel log message
2.	sty	to set serial port
3.	zcat	dump .gz file context
4.	mknod	make device node
5.	free	display system memory usage
6.	date	print or set the system date and time
7.	env	run a program in a modified environment
8.	clear	clear the terminal screen
9.	reboot	reboot / power off/on the server
10.	halt	halt the server
11.	du	estimate file space usage
12.	gzip, gunzip	compress or expand files
13.	hostname	show system's host name

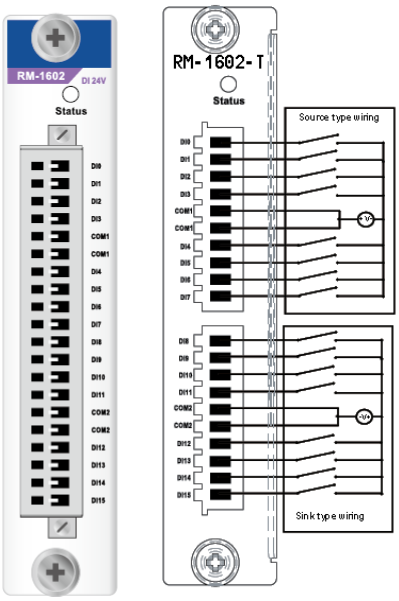
Moxa Special Utilities

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

B

Module Specifications and Wiring

16-channel 24 VDC Digital Input Module



RM-1602-T: 16 digital inputs, 24 VDC, sink/source type

Inputs per Module: 16 channels, sink/source type

On-state Voltage: 24 VDC nominal, 10 VDC min.

OFF-state Voltage: 0 to 3 VDC, 3 VDC max.

Input Impedance: 3K ohms (typical)

Common Type: 16 channels / 2 DI_COMs

Response Time: 100 ms

Over Current Protection: 200 mA per channel

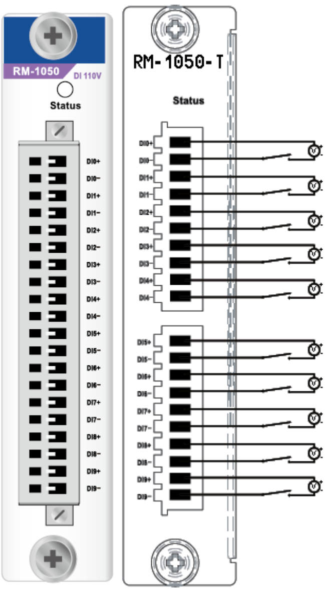
Isolation: I/O to logic (photocoupler isolation)

Operating Temperature: -40 to 75°C

Power Consumption: 20 mA @ 24 VDC (typical)

I/O Cable Wire: AWG 14 (2.0 mm x mm) max.

10-channel 110 VDC Digital Input Module



RM-1050-T: 10 digital inputs, 110 VDC, isolated

Inputs per Module: 10 channels, 110 VDC, channel-to-channel isolated

On-state Voltage: 72 VDC nominal, 50 VDC (min.) to 175 VDC (max.)

Off-state Voltage: 0 to 15 VDC, 15 VDC max.

Input Impedance: 120K ohms (typical)

Common Type: None

Response Time: 100 ms

Over Current Protection: 200 mA per channel

Isolation: I/O to logic (photocoupler isolation)

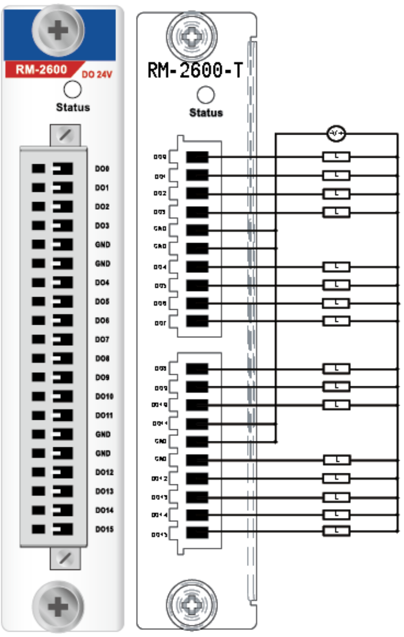
Channel-to-Channel Isolation: 2.5K VDC

Operating Temperature: -40 to 75°C

Power Consumption: 24 mA @ 24 VDC (typical)

I/O Cable Wire: AWG 14 (2.0 mm x mm) max.

16-channel Digital Output Module



RM-2600-T: 16 digital outputs, 24 VDC, sink type, 0.2 A

Outputs er Module: 16 channels, 24 VDC, sink type

Output Impedance: 120m ohms (typical)

Off-state Resistance: 500K ohms (typical)

Response Time: 100 ms

Over Current Protection: 200 mA per channel

Isolation: I/O to logic (photocoupler isolation)

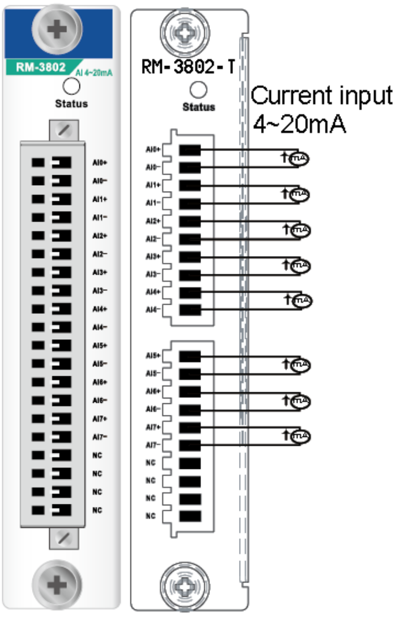
Channel-to-Channel Isolation: 2.5K VDC

Operating Temperature: -40 to 75°C

Power Consumption: 24 mA @ 24 VDC (typical)

I/O Cable Wire: AWG 14 (2.0 mm x mm) max.

8-channel Analog Input Module, 16-bit Resolution



RM-3802-T: 8 analog inputs, 4 to 20 mA, 16 bits

Inputs per Module: 8 channels, differential

Input Current Range: 4 to 20 mA

Input Impendence: 120 ohms

Resolution Range: 16 bits, 0.24 μA/bit

Accuracy:

±0.1%, FSR @ 25°C

±0.3%, FSR @ -40°C, 75°C

Response Time: 100 ms for all channels

Over Current Protection: 200 mA per channel

Isolation: I/O to logic (photocoupler isolation)

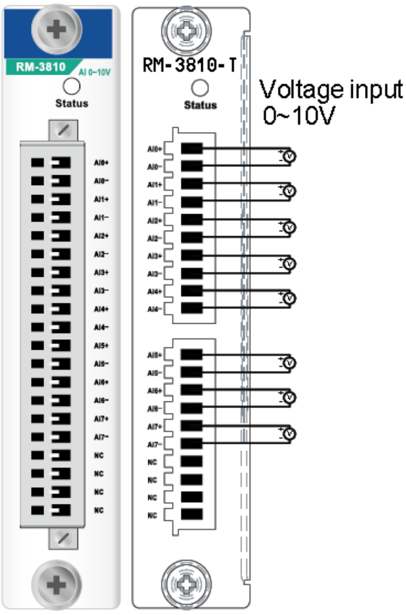
Channel-to-Channel Isolation: 2.5K VDC

Operating Temperature: -40 to 75°C

Power Consumption: 30 mA @ 24 VDC (typical)

I/O Cable Wire: AWG 14 (2.0 mm x mm) max.

8-channel Analog Input Module, 16-bit Resolution



RM-3810-T: 8 analog inputs, 0 to 10 V, 16 bits

Inputs per Module: 8 channels, differential

Input Current Range: 0 to 10 VDC

Input Impendence: > 10M ohms

Resolution Range: 16 bits, 0.15 μA/bit

Data Format: 16-bit integer (2's complement)

Accuracy:

±0.1%, FSR @ 25°C

±0.3%, FSR @ -40°C, 75°C

Response Time: 100 ms for all channels

Over Current Protection: 200 mA per channel

Isolation: I/O to logic (photocoupler isolation)

Channel-to-Channel Isolation: 2.5K VDC

Operating Temperature: -40 to 75°C

Power Consumption: 30 mA @ 24 VDC (typical)

I/O Cable Wire: AWG 14 (2.0 mm x mm) max.

NOTE The 9th slot of the ioPAC 8020-9 series is reserved for future expansion. All I/O modules may only be installed in slot 1 through slot 8.

4-port unmanaged Ethernet Switch Module

16-channel 24 VDC Digital Input Module

KM-2430-T: 4-port unmanaged Ethernet switch module, with M12 connector

Standards:

IEEE 802.3 for 10BaseT

IEEE 802.3u for 100BaseT(X)

IEEE 802.3ab for 1000BaseT(X)

IEEE 802.3x for Flow Control

Processing Type: Store and Forward

Interface: Front cabling, M12 connector, 10/100BaseT(X) auto negotiation speed

Operating Temperature: -40 to 75°C

Power Consumption: 20 mA @ 24 VDC (typical)