W321/341 Linux Software Manual

Eleventh Edition, May 2013

www.moxa.com/product

W321/341 Linux Software Manual

Technical Support Contact Information

www.moxa.com/support

Moxa Americas

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

Moxa 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 W321/341 devices are RISC-based 802.11b/g/n (wireless) ready-to-run embedded computers with one 10/100 Mbps Ethernet port, an internal SD socket, two or four RS-232/422/485 serial ports, two USB 2.0 hosts, one relay output channel,four DIs, four DOs, and a pre-installed Linux operating system. W321/341 computers offer high performance communication and unlimited storage in a super compact, palm-size ARM9 box. Moxa's W321/341 series of embedded computers are the right solution for high performance embedded applications requiring a large amount of memory that must be deployed in hard-to-wire and/or small spaces.

The following topics are covered in this chapter:

Overview

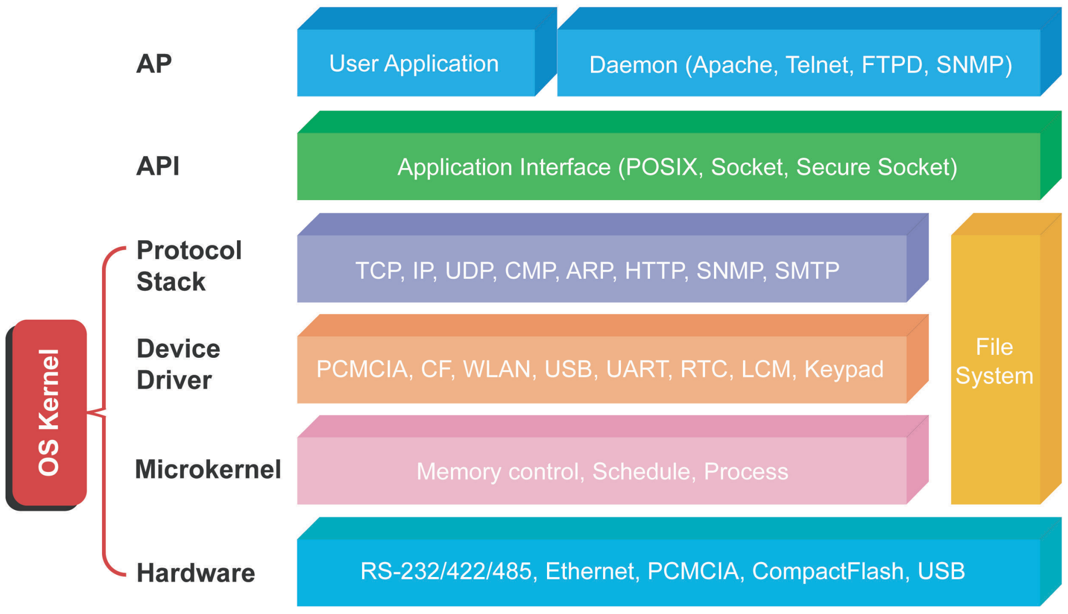
Moxa's W321/341 wireless embedded computers support 802.11b/g/n wireless LAN with WPA2 data encryption for secure transmission tunnels over a WLAN.

Moxa's W321/341 embedded computers use our proprietary ART 192 Mhz RISC CPU. Unlike the X86 CISC CPU, the RISC architecture provides our embedded computers with powerful computing and communication functions without generating a lot of heat. A 16 MB NOR Flash ROM and on-board SDRAM (64 MB for W341 and 32 MB for W321) give enough memory to install application software directly on the embedded computer. In addition, a LAN port and WLAN port are built right into the RISC CPU. This onboard networking, in combination with our embedded serial interfaces, make the W321/341 series an ideal communication platform for data acquisition and industrial control applications.

Moxa's pre-installed Linux operating system (OS) provides an open platform for easier application development. Software written for desktop PCs can be easily ported to the computer with the GNU cross compiler. The OS, device drivers (e.g., serial and buzzer control), and custom applications can all be stored in the onboard NOR flash memory.

Software Architecture

The pre-installed W321/341 Linux operating system follows the standard POSIX architecture. Program porting is done with the GNU tool chain (provided by Moxa). In addition to standard POSIX APIs, device drivers for USB storage, onboard buzzers, network controls, and UART are also included.


AP
User Application
Daemon (Apache, Telnet, FTPD, SNMP)
API
Application Interface (POSIX, Socket, Secure Socket)
Protocol Stack
TCP, IP, UDP, CMP, ARP, HTTP, SNMP, SMTP
OS Kernel
Device Driver
PCMCIA, CF, WLAN, USB, UART, RTC, LCM, Keypad
File System
Microkernel
Memory control, Schedule, Process
Hardware
RS-232/422/485, Ethernet, PCMCIA, CompactFlash, USB

The W321/341's built-in flash ROM is divided into **Boot Loader, Linux Kernel, Root File System,** and **User** partitions.

In order to prevent user applications from crashing the root file system, the W321/341 uses a specially designed **root file tree with protected configuration**. This file system comes with serial and Ethernet communication pre-enabled so that users may conveniently load the **factory default drive image**. All user data, settings, and third-party applications are stored on the user directory partition.

To improve system reliability, the W321/341 has a built-in mechanism that prevents the system from crashing. When the Linux kernel boots up, the kernel will mount the root file system as **read only**, and then enable services and daemons. At the same time, the kernel will start searching for system configuration parameters in the ***rc***/***inittab*** directories.

Because the root file system is protected and cannot be changed by the user, a "safe" zone is created which makes it very difficult for the base OS to become corrupted by anything other than hardware faults. Generally speaking, user behavior or normal read-write operations will not affect the root OS.

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

Journaling Flash File System (JFFS2)

The root file system and user directories in the flash memory are formatted with the **Journaling Flash File System (JFFS2)**. This file system creates a compressed file tree in the flash memory that is is completely transparent to the user, and behaves as if it were an ordinary file sytem installed on a hard disk.

The Journaling Flash File System (JFFS2) was developed by Axis Communications in Sweden; it creates 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, so that the system is always consistent, even if it crashes or other improper power-downs. Because of these featuers, JFFS2 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.

The key features of JFFS2 are:

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

Although JFFS2 is a journaling file system, this does not preclude the loss of data. The file system will remain in a consistent state across power failures and will always be mountable. However, if the board is powered down during a write then the incomplete write will be rolled back on the next boot; only completed writes 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 (v1.0.0.0)
Kernel	Linux 2.6.38
Protocol Stack	ARP, PPP, CHAP, PAP, IPv4, ICMP, TCP, UDP, DHCP, FTP, SNMP V1, NTP, NFS, 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
snmpd	snmpd agent daemon
inetd	TCP server manager program
ftpd	ftp server daemon
sshd	secure shell server
sftp	Secure File Transfer Protocol,
openvpn	virtual private network
openssl	open SSL
Linux Tool Chain	
Gcc (V4.4.2)	C/C++ PC Cross Compiler
Glibc(V2.10.1)	POSIX standard C library

2
Getting Started

In this chapter, we explain how to connect the W321/341, how to turn on the power, how to get started programming, and how to use the W321/341's other functions.

The following topics are covered in this chapter:

Accessing the W321/341 Using a PC

There are two ways to connect the W321/341 to a PC: through the serial console and over the network, by SSH.

Serial Console

The serial console gives users a convenient way of connecting to the W321/341. This method is particularly useful when configuring the computer for the first time, or when an operator does not know either of the two IP addresses.

To access the default console, use the serial port settings shown below:

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

Once the connection is established, you can start using W321/341.

ATTENTION

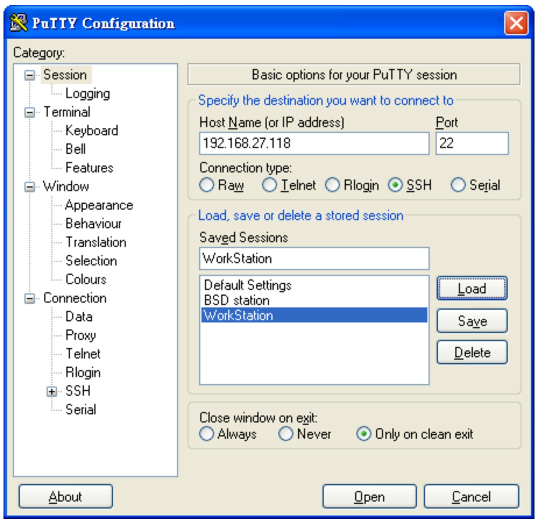
Serial Console Reminder
Remember to choose VT100 as the terminal type. Use the cable CBL-4PINDB9F-100, which comes with the W321/341, to connect to the serial console port.

SSH Console

Moxa W321/341 computers also feature an SSH-accessible console, secure network accessibility. To access the SSH console using a Microsoft Windows interface, Moxa recommends using the open source PuTTY

Windows Users

To download PuTTY Click on the link http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html (free software) to set up an SSH console for the W321/341 in a Windows environment. The following figure shows a simple example of the configuration that is required.



Host Name (or IP address): 192.168.27.118
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions: WorkStation
Default Settings
BSD station
WorkStation
Close window on exit: Always Never [✓ Only on clean exit]

Linux Users

Linux users can connect directly to the console using the "ssh" command over the address below.

#ssh 192.168.3.127

Select **yes** to complete the connection.

```
[root@bee_notebook root]# ssh 192.168.3.127
The authenticity of host '192.168.3.127 (192.168.3.127)' can't be established.
RSA key fingerprint is 8b:ee:ff:84:41:25:fc:cd:2a:f2:92:8f:cb:1f:6b:2f.
Are you sure you want to continue connection (yes/no)? yes_
```

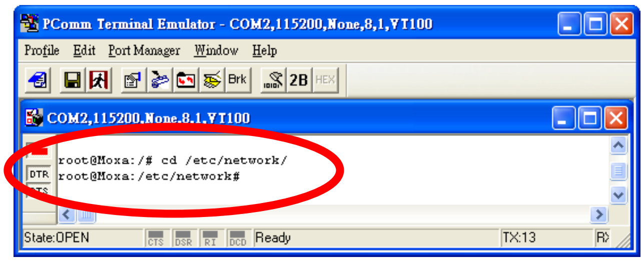
Configuring the Ethernet Interface

The network settings of the W321/341 can be modified with the serial console port or over the network using SSH.

Modifying Network Settings with the Serial Console

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

1. Follow the instructions given in the previous section to access the console using SSH, and then type
   #cd /etc/network
   to change to the networking configuration directory


PComm Terminal Emulator - COM2,115200,None,8,1,VT100
Profile
Edit
Port Manager
Window
Help
COM2,115200,None,8,1,VT100
DTR
State:OPEN
Ready
TX:13

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

2. **#vi interfaces**
   will call up the interfaces configuration file in the VI editor, allowing you to edit the network interface settings. You can configure the Ethernet ports of the W321/341 for **static** or **dynamic** (DHCP) IP addresses.

Default Setting for LAN1	Dynamic Setting using DHCP

LAN is ethernet
WLAN is Wi-Fi interface
 	
# embedded ethernet LAN1 	
iface eth0 inet static
        address 192.168.3.127
        network 192.168.3.0
        netmask 255.255.255.0
        broadcast 192.168.3.255
 	
# embedded Wi-Fi interface 	
iface wlan0 inet static
        address 192.168.4.127
        network 192.168.4.0
        netmask 255.255.255.0
        broadcast 192.168.4.255
	LAN is ethernet
	WLAN is Wi-Fi interface
	
	# embedded ethernet LAN1
	iface eth0 inet dhcp
	
	# embedded Wi-Fi interface
	iface wlan0 inet dhcp

Please note that eth0 stands for the LAN port, while wlan0 stands for the wireless LAN port.

Static IP addresses

As shown in the table above, for static IP configurations 4 settings must be modified: **address, network, netmask,** and **broadcast**. The default IP address for LAN1 is 192.168.3.127, with default netmask of 255.255.255.0.

Dynamic IP addresses

The W321/341 is configured by default for static IP addresses. To configure one or both LAN ports to request an IP address dynamically, replace **static** with **dhcp** and then comment out the address, network, netmask, and broadcast lines by adding a hash mark (#) in the very first space on each line

3. After the boot settings of the LAN interface have been modified, save the file by hitting **`esc + w + enter`** (in succession), close VI, and then issue 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.

Temporarily Modifying Networking Configurations

IP settings can be changed on-the-fly, but the new settings will not available on boot-up without modifying the **`/etc/network/interfaces`** configuration file.

For example, by typing the command
`#ifconfig eth0 192.168.27.115`
the IP address of LAN1 may be changed to 192.168.27.115. However, the change will only take effect after a re-start of the interface, and will not remain in effect after the next system boot.

```
root@Moxa:~# ifconfig eth0 192.168.27.115
root@Moxa:~#
```

Configuring the WLAN

The W321/341 Wi-Fi connection can be configured using the **configuration file** or the **wpa_supplicant** command.We strongly recommend you use **wpa_supplicant** to configure the wireless interface. Other commands might have compatilibty issues.

You can list the available wireless network IDs by issuing the following command:

`#iwpriv wlan0 get_site_survey`

```
root@Moxa:# iwpriv wlan0 get_site_survey
wlan0    get_site_survey:
Ch  SSID       BSSID            Security            Siganl(%)W-Mode  ExtCH  NT
1   MIS-WAP-1  50:67:f0:61:2d:7a   WEP                  55      11b/g   NONE   In
```

If you want to get more detailed info, type the following command:

#iwlist wlan0 scanning

```
root@Moxa:# iwlist wlan0 scanning
wlan0    Scan completed :
        Cell 01 - Address: 50:67:F0:61:2D:7A
                Protocol:802.11b/g
                ESSID:"MIS-WAP-1"
                Mode:Managed
                Frequency:2.412 GHz (Channel 1)
                Quality=81/100  Signal level=-58 dBm  Noise level=-92 dBm
                Encryption key:on
                Bit Rates:54 Mb/s
```

NOTE W321/341 only supports 2.4GHz Wi-Fi, which means only 802.11 b/g/n with 2.4GHz is available.

Using WPA_SUPPLICANT to configure WPA2

This embedded computer supports WPA2 security using the **/bin/wpa_supplicant** program.

Refer to the following table to determine the configuration options. The column labeled "Key required before joining network?" describes whether an encryption and/or authentication key must be configured before associating with a network.

Infrastructure mode	Authentication mode	Encryption status	Manual Key required?	IEEE 802.1X enabled?	Key required before joining network?
ESS	Open	None	No	No	No
ESS	Open	WEP	Optional	Optional	Yes
ESS	Shared	None	Yes	No	Yes
ESS	Shared	WEP	Optional	Optional	Yes
ESS	WPA	WEP	No	Yes	No
ESS	WPA	TKIP	No	Yes	No
ESS	WPA	AES	No	Yes	No
ESS	WPA-PSK	WEP	Yes	Yes	No
ESS	WPA-PSK	TKIP	Yes	Yes	No
ESS	WPA-PSK	AES	Yes	Yes	No
IBSS	Open	None	No	No	No
IBSS	Open	WEP	Yes	No	Yes
IBSS	Shared	None	Yes	No	Yes
IBSS	Shared	WEP	Yes	No	Yes
IBSS	WPA-None	WEP	Yes	No	Yes
IBSS	WPA-None	TKIP	Yes	No	Yes
IBSS	WPA-None	AES	Yes	No	Yes

See the following section for the shell script to use this function.

Connecto to an AP via WEP shared key authentication

Step 1: Edit **/etc/Wireless/wpa_supplicant.conf.**

```
##### WEP #####
network={
            ssid="MIS-WAP-1"
            bssid=50:67:F0:61:2D:7A
            key_mgmt=NONE
            wep_key0=CFEE46EED3FA94FAEB92348922
}
###############
```

The following table describes each parameter:

Parameter	Usage	Function
ssid	{Acess Point Name}	Network name (as announced by the access point). An ASCII or hex string enclosed in quotation marks.
bssid	{MAC address of the AP}	Set network bssid, (typically the MAC address of the access point).
key_mgmt	{NONE,WEP,TKIP,AES}	List of acceptable key management protocols;
wep_key0	{wep key}	WEP key in hexadecimal format

Step 2: Type **/etc/init.d/wireless.sh start** to enable this function. To stop the function, type **/etc/init.d/wireless.sh stop.**

NOTE Click on the following links for more information about wpa_supplicant.conf.

http://www.daemon-systems.org/man/wpa_supplicant.conf.5.html
http://linux.die.net/man/5/wpa_supplicant.conf

NOTE If you need to get an IP address by DHCP, call the "dhcpcd wlan0" command **after** associating with an AP.

Connecting to an AP via WPA/WPA2 PSK authentication

Follow these steps to connect an AP via WPA/WPAs PSK authentication.

Step 1: Edit the related parameter in the file **/etc/Wireless/wpa_supplicant.conf.**

```
##### WPA/WPA2 PSK #####
network={
            ssid="5566"
            proto=WPA WPA2 RSN
            key_mgmt=WPA-PSK
            pairwise=TKIP CCMP
            group=TKIP CCMP
            psk="01234567890"
}
#######################
```

Step 2: Type `/etc/init.d/wireless.sh start` to start (or re-start) the wireless interface. To stop the wireless interface, type:
`/etc/init.d/wireless.sh stop`**.**

The following table describes each parameter:

Parameter	Usage	Function
ssid	{Acess Point Name}	Network name (as announced by the access point). An ASCII or hex string enclosed in quotation marks.
proto	{WPA WPA2 RSN}	List of acceptable protocols; one or more of: WPA

		(IEEE802.11i/D3.0) and RSN (IEEE 802.11i). WPA2 is another name for RSN. If you do not configure, the default value is "WPA RSN".
key_mgmt	{WPA-PSK or WPA-EAP}	List of acceptable key management protocols; one or more of: WPA-PSK (WPA pre-shared key), WPA-EAP (WPA using EAP authentica-tion), IEEE8021X (IEEE 802.1x using EAP authentication and, optionally, dynamically generated WEP keys). If you do not configure, the default value is "WPA-PSK WPA-EAP".
pairwise	{TKIP CCMP, or NONE}	List of acceptable pairwise (unicast) ciphers for WPA; one or more of: CCMP (AES in Counter mode with CBC-MAC, RFC 3610, IEEE802.11i/D7.0), TKIP (Temporal Key Integrity Protocol, IEEE802.11i/D7.0), NONE (deprecated). If you do not configure, the default value is "CCMP TKIP".
group	{CCMP, TKIP, WEP104, WEP40}	List of acceptable group (multicast) ciphers for WPA; one or more of: CCMP (AES in Counter mode with CBC-MAC, RFC 3610, IEEE802.11i/D7.0), TKIP (Temporal Key Integrity Protocol, IEEE802.11i/D7.0), WEP104 (WEP with 104-bit key), EP40 (WEP with 40-bit key). If you do not configure, the default value is "CCMP TKIP WEP104 WEP40".
psk	{preshared key}	WPA preshared key used in WPA-PSK mode. The key is specified as 64 hex digits or as an 8-63 character ASCII passphrase.
mode	# 0 = infrastructure (Managed) mode, i.e., associate with an AP (default) # 1 = IBSS (ad-hoc, peer-to-peer)	IEEE 802.11 operation mode

Configuring STA for Ad-Hoc Mode Using WEP

Use the following commands to configure STA to create a link as adhoc mode with **WEP** encryption.

```
ap_scan=2
network={
      ssid="test adhoc"
      mode=1
      frequency=2412
      key_mgmt=NONE
      wep_key0="01234567890"
}
```

See the following descriptions for each parameter.

Parameter	Usage	Function
ssid	{Acess Point Name}	Network name (as announced by the access point). An ASCII or hex string enclosed in quotation marks.
mode	# 0 = infrastructure (Managed) mode, i.e., associate with an AP (default) # 1 = IBSS (ad-hoc, peer-to-peer)	IEEE 802.11 operation mode

frequency	Channel | Frequency (GHz) | Range 1 | 2.412 | 2.401 -2.423 2 | 2.417 | 2.406 - 2.428 3 | 2.422 | 2.411 - 2.433 4 | 2.427 | 2.416 - 2.438 5 | 2.432 | 2.421 - 2.443 6 | 2.437 | 2.426 - 2.448 7 | 2.442 | 2.431 - 2.453 8 | 2.447 | 2.436 - 2.458 9 | 2.452 | 2.441 - 2.463 10 | 2.457 | 2.446 - 2.468 11 | 2.462 | 2.451 - 2.473 12 | 2.467 | 2.456 - 2.478 13 | 2.472 | 2.461 - 2.483 14 | 2.484 | 2.473 - 2.495	Countries apply their own regulations to both the allowable channels, allowed users and maximum power levels within these frequency ranges. Consult your local authorities as these regulations may be out of date as they are subject to change at any time. Most of the world will allow the first thirteen channels in the spectrum.
bssid	{MAC address of the AP}	Set network bssid, (typically the MAC address of the access point).
key_mgmt	{NONE,WEP,TKIP,AES}	List of acceptable key management protocols;
wep_key0	{wep key}	WEP key in hexadecimal format

Step 2: Type `/etc/init.d/wireless.sh start` to start the wireless daemon. To stop the wireless daemon, type `/etc/init.d/wireless.sh stop`.

Configuring STA for Ad-Hoc Mode Using WPA-none PSK Authentication

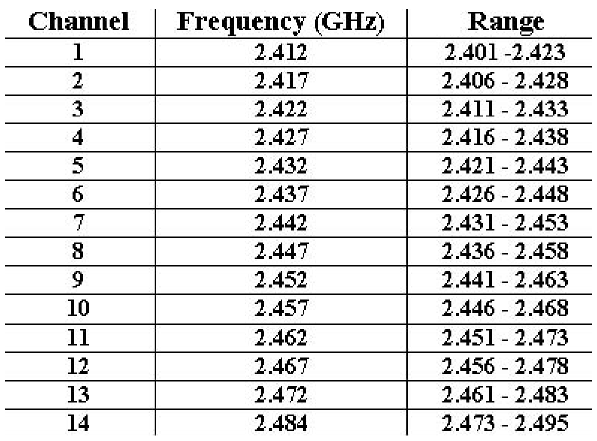
Use the following commands to configure STA to create a link as adhoc mode with **WPANONE/CCMP** (**authentication/encryption)**.

```
ap_scan=2
network={
      ssid="test adhoc"
      mode=1
      frequency=2412
      proto=WPA
      key_mgmt=WPA-NONE
      pairwise=CCMP
      group=CCMP
      psk="01234567890"
}
```

Step 2: Type `/etc/init.d/wireless.sh start` to enable this function. To stop the function, type `/etc/init.d/wireless.sh stop`**.**

See the following descriptions for each parameter.

Parameter	Usage	Function
ssid	{Acess Point Name}	Network name (as announced by the access point). An ASCII or hex string enclosed in quotation marks.

frequency		Countries apply their own regulations to both the allowable channels, allowed users and maximum power levels within these frequency ranges. Consult your local authorities as these regulations may be out of date as they are subject to change at any time. Most of the world will allow the first thirteen channels in the spectrum.
proto	{WPA}	List of acceptable protocols: WPA
key_mgmt	{ WPA-NONE }	List of acceptable key management protocols: WPA-NONE
pairwise	{CCMP}	List of acceptable pairwise (unicast) ciphers for WPA; one or more of: CCMP (AES in Counter mode with CBC-MAC, RFC 3610, IEEE802.11i/D7.0), TKIP (Temporal Key Integrity Protocol, IEEE802.11i/D7.0), NONE (deprecated). If you do not configure, the default value is "CCMP TKIP".
group	{CCMP, TKIP, WEP104, WEP40}	List of acceptable group (multicast) ciphers for WPA; one or more of: CCMP (AES in Counter mode with CBC-MAC, RFC 3610, IEEE802.11i/D7.0), TKIP (Temporal Key Integrity Protocol, IEEE802.11i/D7.0), WEP104 (WEP with 104-bit key), EP40 (WEP with 40-bit key). If you do not configure, the default value is "CCMP TKIP WEP104 WEP40".
psk	{preshared key}	WPA preshared key used in WPA-PSK mode. The key is specified as 64 hex digits or as an 8-63 character ASCII passphrase.
mode	# 0 = infrastructure (Managed) mode, i.e., associate with an AP (default) # 1 = IBSS (ad-hoc, peer-to-peer)	IEEE 802.11 operation mode

Channel	Frequency (GHz)	Range
1	2.412	2.401 -2.423
2	2.417	2.406 - 2.428
3	2.422	2.411 - 2.433
4	2.427	2.416 - 2.438
5	2.432	2.421 - 2.443
6	2.437	2.426 - 2.448
7	2.442	2.431 - 2.453
8	2.447	2.436 - 2.458
9	2.452	2.441 - 2.463
10	2.457	2.446 - 2.468
11	2.462	2.451 - 2.473
12	2.467	2.456 - 2.478
13	2.472	2.461 - 2.483
14	2.484	2.473 - 2.495

NOTE Click on the following links for more information about wpa_supplicant.conf.

http://www.daemon-systems.org/man/wpa_supplicant.conf.5.html
http://linux.die.net/man/5/wpa_supplicant.conf

NOTE Most of time, you need to get the IP from AP by using “dhcpcd wlan0” command after connecting to AP

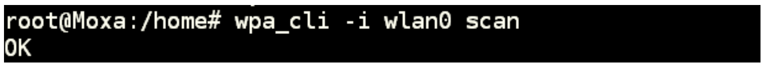
Enabling wpa_cli to interact with wpa_supplicant

wpa_cli is a text-based frontend program for interacting with **wpa_supplicant**. It is used to query current status, change configuration, trigger events, and request interactive user input.

NOTE **wpa_supplicant must be executed before using wpa_cli command.** Click on the following links for more information on wpa_cli.
http://linux.die.net/man/8/wpa_cli

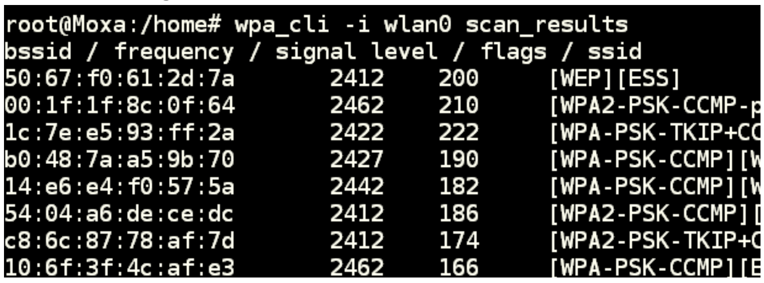
Scanning AP and checking results

Use the following command to scan local access points:



```
root@Moxa:/home# wpa_cli -i wlan0 scan
OK
```

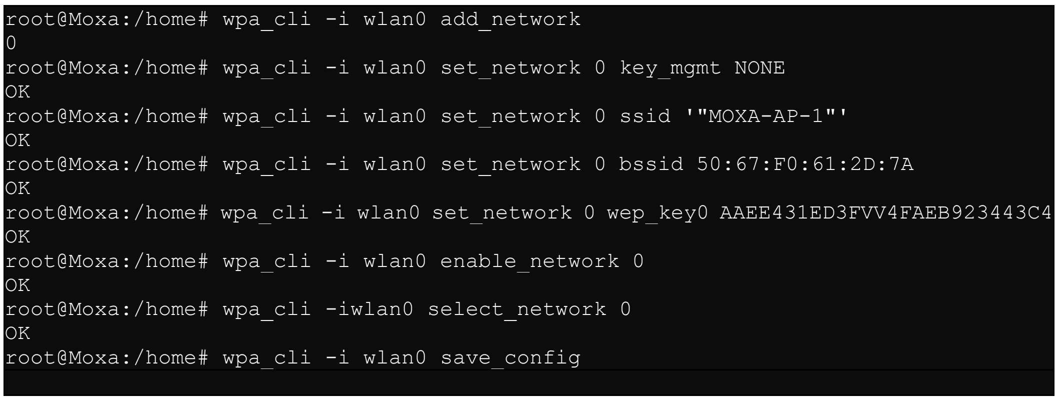
Use the following command to check the results:



```
root@Moxa:/home# wpa_cli -i wlan0 scan_results
bssid / frequency / signal level / flags / ssid
50:67:f0:61:2d:7a       2412    200     [WEP][ESS]
00:1f:1f:8c:0f:64       2462    210     [WPA2-PSK-CCMP-p
1c:7e:e5:93:ff:2a       2422    222     [WPA-PSK-TKIP+CC
b0:48:7a:a5:9b:70       2427    190     [WPA-PSK-CCMP][W
14:e6:e4:f0:57:5a       2442    182     [WPA-PSK-CCMP][W
54:04:a6:de:ce:dc       2412    186     [WPA2-PSK-CCMP][
c8:6c:87:78:af:7d       2412    174     [WPA2-PSK-TKIP+C
10:6f:3f:4c:af:e3       2462    166     [WPA-PSK-CCMP][E
```

Adding WEP setting into configuration file

Use the following commands to add WEP setting into **/etc/Wireless/wpa_supplicant.conf**.



```
root@Moxa:/home# wpa_cli -i wlan0 add_network
0
root@Moxa:/home# wpa_cli -i wlan0 set_network 0 key_mgmt NONE
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 0 ssid '"MOXA-AP-1"'
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 0 bssid 50:67:F0:61:2D:7A
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 0 wep_key0 AAEE431ED3FVV4FAEB923443C4
OK
root@Moxa:/home# wpa_cli -i wlan0 enable_network 0
OK
root@Moxa:/home# wpa_cli -iwlan0 select_network 0
OK
root@Moxa:/home# wpa_cli -i wlan0 save_config
```

Adding WPA/WPA2 Settings into the Configuration File

Use the following commands to add WPA/WPA2 setting into **/etc/Wireless/wpa_supplicant.conf**.

```
root@Moxa:/home# wpa_cli -i wlan0 add_network
1
root@Moxa:/home# wpa_cli -i wlan0 set_network 1 ssid '"MOXA-AP"'
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 1 proto 'WPA WPA2 RSN'
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 1 key_mgmt 'WPA-PSK'
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 1 pairwise 'TKIP CCMP'
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 1 group 'TKIP CCMP'
OK
root@Moxa:/home# wpa_cli -i wlan0 set_network 1 psk '"01234567890"'
'SET_NETWORK 1 psk "01234567890"' command timed out.
root@Moxa:/home# wpa_cli -i wlan0 enable_network 1
OK
root@Moxa:/home# wpa_cli -iwlan0 select_network 1
OK
root@Moxa:/home# wpa_cli -i wlan0 save_config
OK
```

The following **wpa_cli** commands are available:

Command	**Function**
wpa_cli -i wlan0 status	get current WEP/WPA/EAPOL/EAP status
wpa_cli -i wlan0 help	show this usage help
wpa_cli -i wlan0 terminate	terminate wpa_supplicant
wpa_cli -i wlan0 interface	show interfaces/select interface
wpa_cli -i wlan0 list_networks	list configured networks in wpa_supplicant.conf
wpa_cli -i wlan0 select_network	Set network variables. Network id can be received from the LIST_NETWORKS command output. This command uses the same variables and data formats as the configuration file.
wpa_cli -i wlan0 enable_network	Enable a network. Network id can be received from the LIST_NETWORKS command output.
wpa_cli -i wlan0 disable_network	Disable a network. Network id can be received from the LIST_NETWORKS command output. Special network id all can be used to disable all networks.
wpa_cli -i wlan0 remove_network	Remove a network. Network id can be received from the LIST_NETWORKS command output. Special network id all can be used to remove all networks.
wpa_cli -i wlan0 reconfigure	Force wpa_supplicant to re-read its configuration file
wpa_cli -i wlan0 save_config	Save the current configuration. (Replace original /etc/Wireless/wpa_supplicant.conf file)
wpa_cli -i wlan0 scan wpa_cli -i wlan0 scan_results	Scan available networks Get scanning results

SD Slot and USB for Storage Expansion

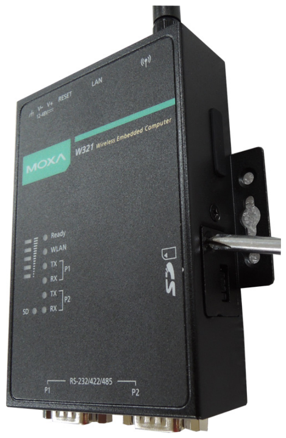
The W321/341 provides an SD slot for storage expansion. Moxa provides an SD flash disk for plug & play expansion that allows users to plug in a Secure Digital (SD) memory card compliant with the SD 1.0 standard for up to 1 GB of additional memory space, or a Secure Digital High Capacity (SDHC) memory card compliant with the SD 2.0 standard for up to 16 GB of additional memory space. The following steps show you how to install SD card into the W321/341.

W321

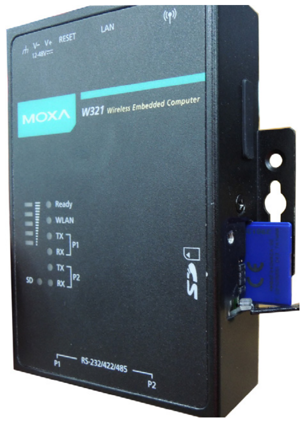
The SD slot is located on the right side of the W321 enclosure. To install an SD memory card, you must first remove the SD slot's protective cover to access the slot, and then plug the SD card directly into the slot.

The SD memory card will be mounted at **`/mnt/sd`**. Detailed installation instructions are shown below:

Step 1: Use a screwdriver to remove the screws holding the SD slot's outer cover.



Step 2: After removing the cover, insert the SD memory card as shown.

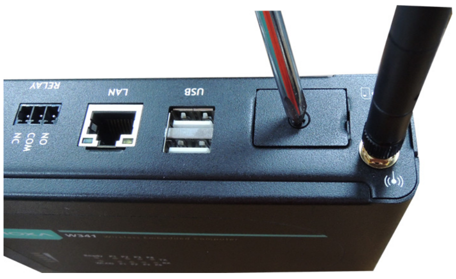


W341

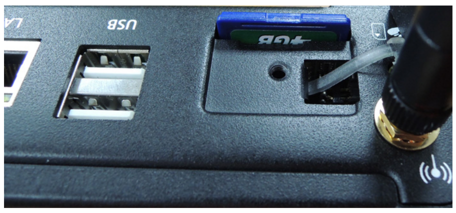
The SD slot is located on the front panel of the W341. To install an SD memory card, you must first remove the SD slot's protective cover to access the slot, and then plug the SD memory card directly into the slot.

The SD memory card will be mounted at **`/mnt/sd`**. Detailed installation instructions are shown below:

Step 1: Use a screwdriver to remove the screws holding the SD slot's outer cover, and then remove the cover.



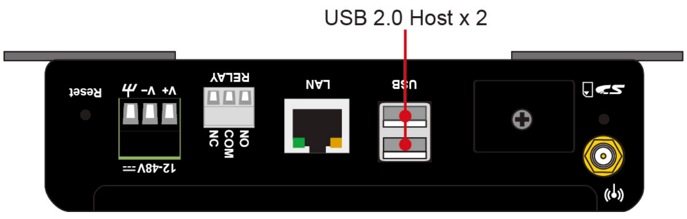
Step 2: Insert the SD memory card as shown.



NOTE To remove the SD memory card from the slot, press the SD memory card in slightly with your finger, and then remove your finger to cause the card to spring out partially. You may now grasp the top of the card with two fingers and pull it out.

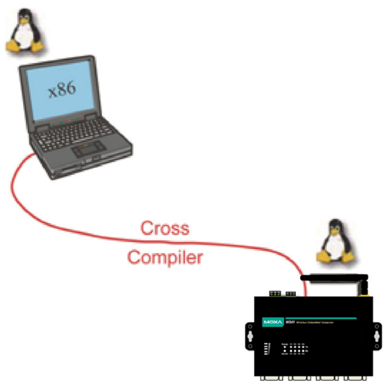
Before removing the SD memory card, remember to type **/sync** to ensure that your data has been written.

In addition to the SD socket, two USB 2.0 ports are located on the W341's upper panel. The USB host is also designed for storage expansion. To use a USB flash disk to expand the storage space, plug the USB flash disk into the USB port. The flash disk will be detected automatically, and its file partition will be mounted into the OS. For the W341, the USB storage will be mounted in one of the following directories: /mnt/usbstorage1, /mnt/usbstorage2.



Test Program—Developing Hello.c

In this section, we use the standard "Hello" programming example to illustrate how to develop a program for the W321/341. In general, program development involves the following seven steps.



Step 1:
Connect the W321/341 to a Linux PC.

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

Step 3:
Set the cross compiler and glibc environment variables.

Step 4:
Code and compile the program.

Step 5:
Download the program to the W321/341 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 W321/341 units if needed).

Installing the Tool Chain (Linux)

The Linux Operating System must be pre-installed in the PC before installing the W321/341 GNU Tool Chain. Fedora core or compatible versions are recommended. The Tool Chain requires approximately 100 MB of hard disk space on your PC. The W321/341 Tool Chain software is located on the W321/341 CD. To install the Tool Chain, insert the CD into your PC and then issue the following commands:

```
#mount/dev/cdrom /mnt/cdrom
/mnt/cdrom/tool-chain/arm-linux_3.1_Build_11111411.sh
```

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

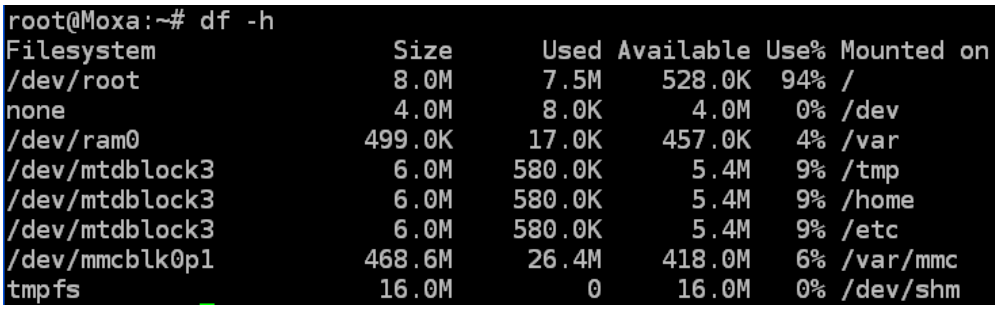
```
PATH=/usr/local/arm-linux-4.4.2/bin:$PATH
```

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:~# df -h
Filesystem                  Size      Used Available Use% Mounted on
/dev/root                   8.0M      7.5M    528.0K  94% /
none                        4.0M      8.0K      4.0M   0% /dev
/dev/ram0                 499.0K     17.0K    457.0K   4% /var
/dev/mtdblock3              6.0M    580.0K      5.4M   9% /tmp
/dev/mtdblock3              6.0M    580.0K      5.4M   9% /home
/dev/mtdblock3              6.0M    580.0K      5.4M   9% /etc
/dev/mmcblk0p1            468.6M     26.4M    418.0M   6% /var/mmc
tmpfs                      16.0M         0     16.0M   0% /dev/shm
```

If there isn't enough "Available" space for your application, you will need to delete some existing files. To do this, connect your PC to the W321/341 with the console cable, and then use the console utility to delete the files from the W321/341's flash memory. To check the amount of free space available, 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, you will need to free up some memory space before saving files to the Flash ROM.

Compiling Hello.c

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

```
# cd /tmp/
# mkdir example
# cp –r /mnt/cdrom/example/W321/* /tmp/example
```

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

```
#cd example/W321/hello #make
```

You should receive 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]# _
```

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

hello-release—an ARM platform execution file (created specifically to run on the W321/341)

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

NOTE Since Moxa's tool chain places a specially designed **Makefile** in the directory **/tmp/example/W321/hello**, be sure to type the **#make** command from within that directory. This special Makefile uses the mxscale-gcc compiler to compile the hello.c source code for the Xscale environment. If you type the **#make** command from within any other directory, Linux will use the x86 compiler (for example, cc or gcc).

Refer to Chapter 5 to see a Makefile example.

Uploading and Running the "Hello" Program

Use the following commands to upload **hello-release** to the W321/341 by FTP.

1. From the PC, type:
   ```
   #ftp 192.168.3.127
   ```
2. Use the bin command to set the transfer mode to Binary mode, and then use the put command to initiate the file transfer:
   ```
   ftp> bin
   ftp> cd /home
   ftp> put hello-release
   ```
3. From the W321/341, type:
   ```
   # chmod +x hello-release
   # ./hello-release
   ```

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

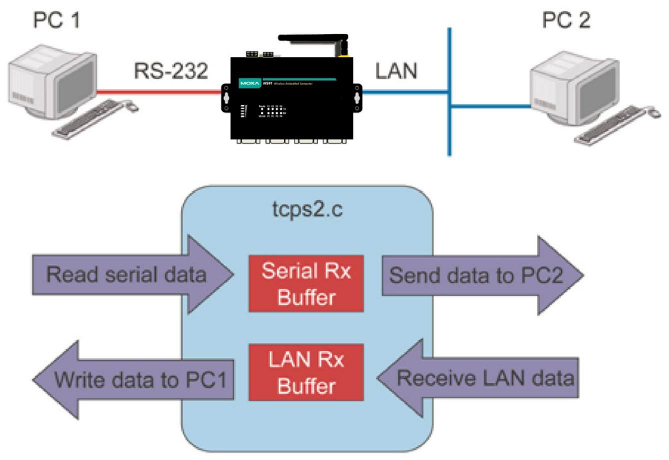
```
root@Moxa:~# ./hello-release
Hello
```

Developing Your First Application

We use the tcps2 example to illustrate how to build an application. The procedure outlined in the following subsections will show you how to build a TCP server program plus serial port communication that runs on the W321/341.

Testing Environment

The tcps2 example demonstrates a simple application program that delivers transparent, bi-directional data transmission between the W321/341's serial and Ethernet ports. As illustrated in the following figure, the purpose of this application is to transfer data between PC 1 and the W321/341 through an RS-232 connection. At the remote site, data can be transferred between the W321/341's Ethernet port and PC 2 over an Ethernet connection.



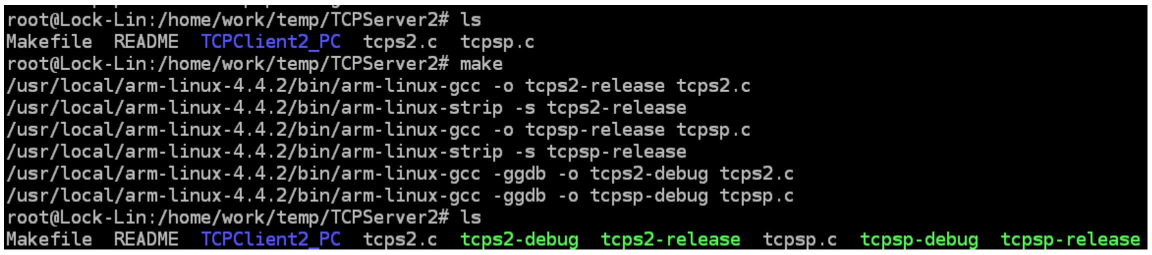
Compiling tcps2.c

The source code for the tcps2 example is located on the CD-ROM at **CD-ROM://example/W321/TCPServer2/tcps2.c.** Use the following commands to copy the file to a specific directory on your PC. We use the direrctory **/home/work/temp/**. Note that you need to copy 3 files—Makefile, tcps2.c, tcpsp.c—from the CD-ROM to the target directory.

```
#mount -t iso9660 /dev/cdrom /mnt/cdrom
#cp /mnt/cdrom/example/W321/TCPServer2/tcps2.c /home/w341/1st_application/tcps2.c
#cp /mnt/cdrom/example/W321/TCPServer2/tcpsp.c /home/w341/1st_application/tcpsp.c
#cp /mnt/cdrom/example/W321/TCPServer2/Makefile.c /home/w341/1st_application/Makefile
```

Type **`#make`** to compile the example code:

You will get the following response, indicating that the example program was compiled successfully.



```
root@Lock-Lin:/home/work/temp/TCPServer2# ls
Makefile  README  TCPClient2_PC  tcps2.c  tcpsp.c
root@Lock-Lin:/home/work/temp/TCPServer2# make
/usr/local/arm-linux-4.4.2/bin/arm-linux-gcc -o tcps2-release tcps2.c
/usr/local/arm-linux-4.4.2/bin/arm-linux-strip -s tcps2-release
/usr/local/arm-linux-4.4.2/bin/arm-linux-gcc -o tcpsp-release tcpsp.c
/usr/local/arm-linux-4.4.2/bin/arm-linux-strip -s tcpsp-release
/usr/local/arm-linux-4.4.2/bin/arm-linux-gcc -ggdb -o tcps2-debug tcps2.c
/usr/local/arm-linux-4.4.2/bin/arm-linux-gcc -ggdb -o tcpsp-debug tcpsp.c
root@Lock-Lin:/home/work/temp/TCPServer2# ls
Makefile  README  TCPClient2_PC  tcps2.c  tcps2-debug  tcps2-release  tcpsp.c  tcpsp-debug  tcpsp-release
```

Two executable files, **tcps2-release** and **tcps2-debug**, are created.

tcps2-release—an ARM platform execution file (created specifically to run on the W321/341).

tcps2-debug—an ARM platform GDB debug server execution file (see Chapter 5 for details about the GDB debug tool).

NOTE If you get an error message at this point, it could be because you neglected to put tcps2.c and tcpsp.c in the same directory. The example Makefile we provide is set up to compile both tcps2 and tcpsp into the same project Makefile. Alternatively, you could modify the Makefile to suit your particular requirements.

Uploading and Running the "tcps2-release" Program

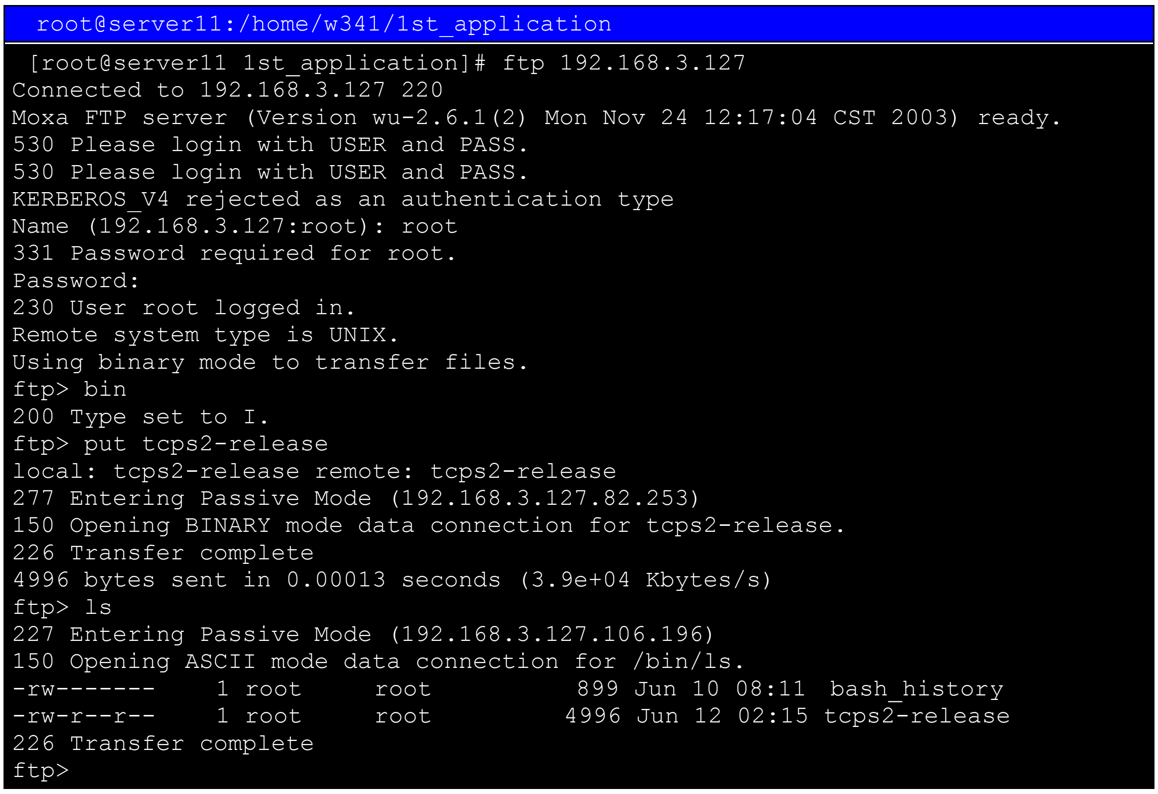
Use the following commands to upload **tcps2-release** to the W321/341 through an FTP connection.

1. From the PC, type:

   ```
   #ftp 192.168.3.127
   ```

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

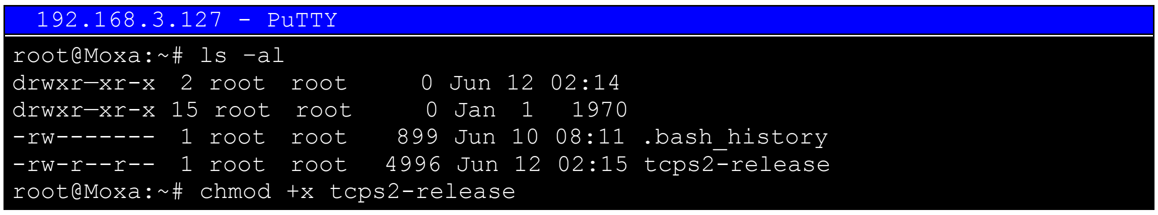
   ```
   ftp> bin
   ftp> cd /home
   ftp> put tcps2-release
   ```



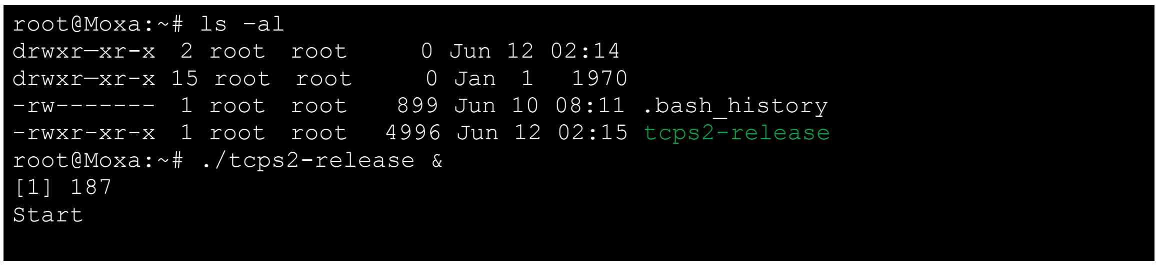
```
  root@server11:/home/w341/1st_application
 [root@server11 1st_application]# ftp 192.168.3.127
Connected to 192.168.3.127 220
Moxa FTP server (Version wu-2.6.1(2) Mon Nov 24 12:17:04 CST 2003) ready.
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.3.127: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> put tcps2-release
local: tcps2-release remote: tcps2-release
277 Entering Passive Mode (192.168.3.127.82.253)
150 Opening BINARY mode data connection for tcps2-release.
226 Transfer complete
4996 bytes sent in 0.00013 seconds (3.9e+04 Kbytes/s)
ftp> ls
227 Entering Passive Mode (192.168.3.127.106.196)
150 Opening ASCII mode data connection for /bin/ls.
-rw-------    1 root     root          899 Jun 10 08:11  bash_history
-rw-r--r--    1 root     root         4996 Jun 12 02:15 tcps2-release
226 Transfer complete
ftp>
```

3. From the W321/341, type:

   ```
   # chmod +x tcps2-release
   # ./tcps2-release &
   ```

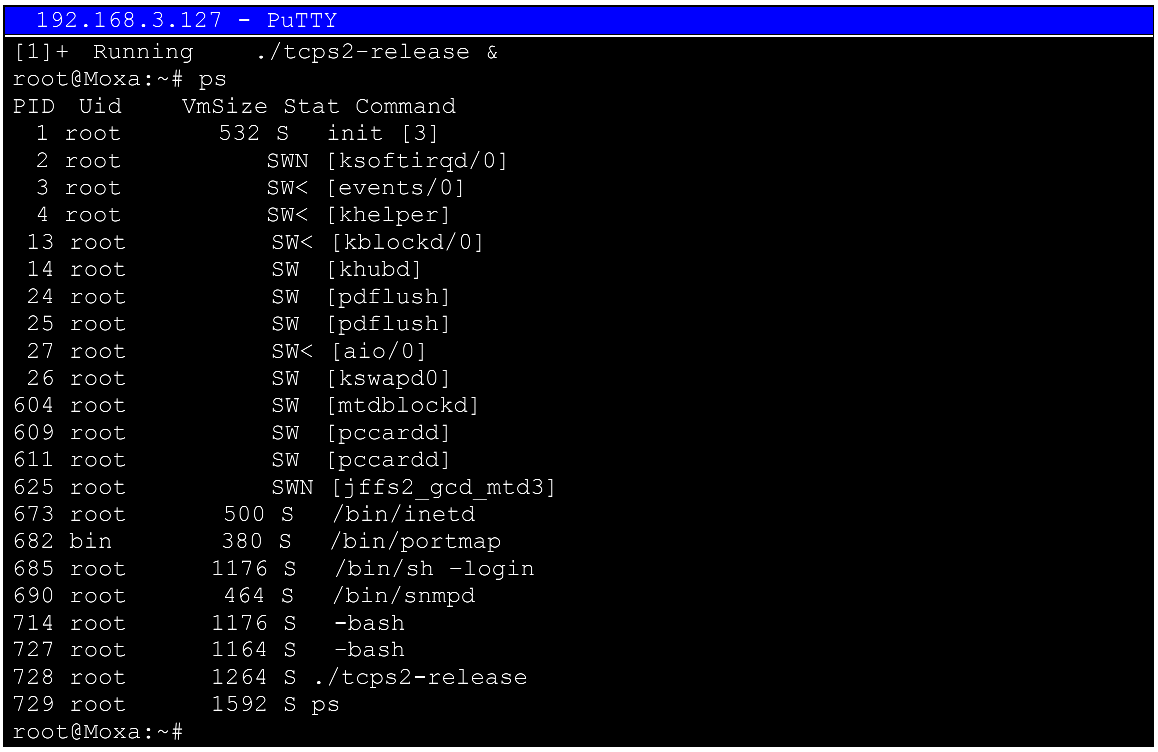


```
  192.168.3.127 - PuTTY
root@Moxa:~# ls -al
drwxr-xr-x  2 root  root     0 Jun 12 02:14
drwxr-xr-x 15 root  root     0 Jan  1   1970
-rw-------  1 root  root   899 Jun 10 08:11 .bash_history
-rw-r--r--  1 root  root  4996 Jun 12 02:15 tcps2-release
root@Moxa:~# chmod +x tcps2-release
```



```
root@Moxa:~# ls -al
drwxr-xr-x  2 root  root      0 Jun 12 02:14
drwxr-xr-x 15 root  root      0 Jan  1   1970
-rw-------  1 root  root    899 Jun 10 08:11 .bash_history
-rwxr-xr-x  1 root  root   4996 Jun 12 02:15 tcps2-release
root@Moxa:~# ./tcps2-release &
[1] 187
Start
```

4. The program should start running in the background. Use the **#ps** command to check if the tcps2 program is actually running in the background.



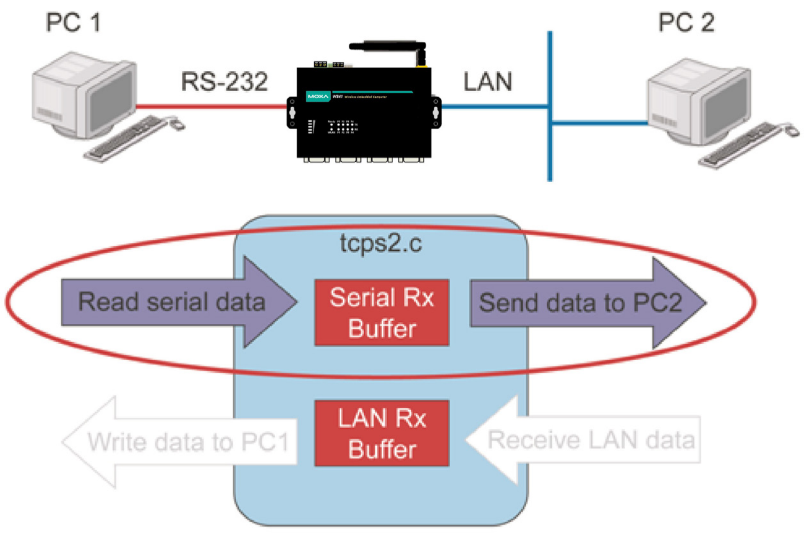
```
  192.168.3.127 - PuTTY
[1]+  Running     ./tcps2-release &
root@Moxa:~# ps
PID  Uid     VmSize Stat Command
  1 root       532 S   init [3]
  2 root           SWN [ksoftirqd/0]
  3 root           SW< [events/0]
  4 root           SW< [khelper]
 13 root           SW< [kblockd/0]
 14 root           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
682 bin        380 S   /bin/portmap
685 root      1176 S   /bin/sh -login
690 root       464 S   /bin/snmpd
714 root      1176 S   -bash
727 root      1164 S   -bash
728 root      1264 S ./tcps2-release
729 root      1592 S ps
root@Moxa:~#
```

NOTE Use the **kill -9** command for PID 728 to terminate this program: **#kill -9 %728**

Testing Procedure Summary

1. Compile **tcps2.c** (**#make**).
2. Upload and run **tcps2-release** in the background **(#./tcps2-release &)**.
3. Check that the process is running **(#ps)**.
4. Use a serial cable to connect PC1 to the W321/341's serial port 1.
5. Use an Ethernet cable to connect PC2 to the W321/341.
6. On PC1: If running Windows, use HyperTerminal (**115200, n, 8, 1**) to open COMn.
7. On PC2: Type **#telnet 192.168.3.127 4001**.
8. On PC1: Type some text on the keyboard and then press **Enter**.
9. On PC2: The text you typed on PC1 will appear on PC2's screen.

The testing environment is illustrated in the following figure. However, note that there are limitations to the example program **tcps2.c**.



NOTE The tcps2.c application is a simple example designed to give users a basic understanding of the concepts involved in combining Ethernet communication and serial port communication. However, the example program has some limitations that make it unsuitable for real-life applications.

1. The serial port is in canonical mode and block mode, making it impossible to send data from the Ethernet side to the serial side (i.e., from PC 2 to PC 1 in the above example).
2. The Ethernet side will not accept multiple connections.

3

Managing Embedded Linux

This chapter includes information about version control, deployment, updates, and peripherals. The information in this chapter will be particularly useful when you need to run the same application on several W321/341 units.

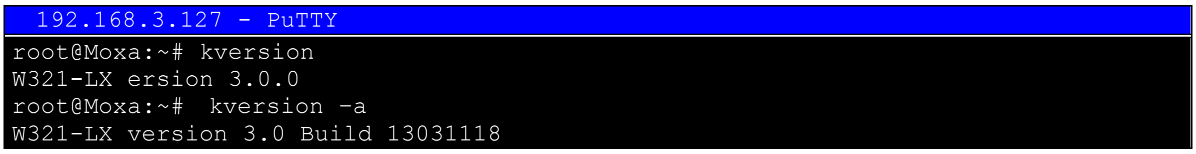
The following topics are covered in this chapter:

System Version Information

To determine the hardware capability of your W321/341, and what kind of software functions are supported, check the version numbers of your W321/341's hardware, kernel, and user file system. Contact Moxa to determine the hardware version. You will need the Production S/N (Serial number), which is located on the W321/341's bottom label.

To check the kernel version, type:

#kversion



```
 192.168.3.127 - PuTTY
root@Moxa:~# kversion
W321-LX ersion 3.0.0
root@Moxa:~#  kversion -a
W321-LX version 3.0 Build 13031118
```

NOTE The kernel version number is for the factory default configuration. You may download the latest firmware version from Moxa's website and then upgrade the W321/341's hardware.

System Image Backup

Upgrading the Firmware

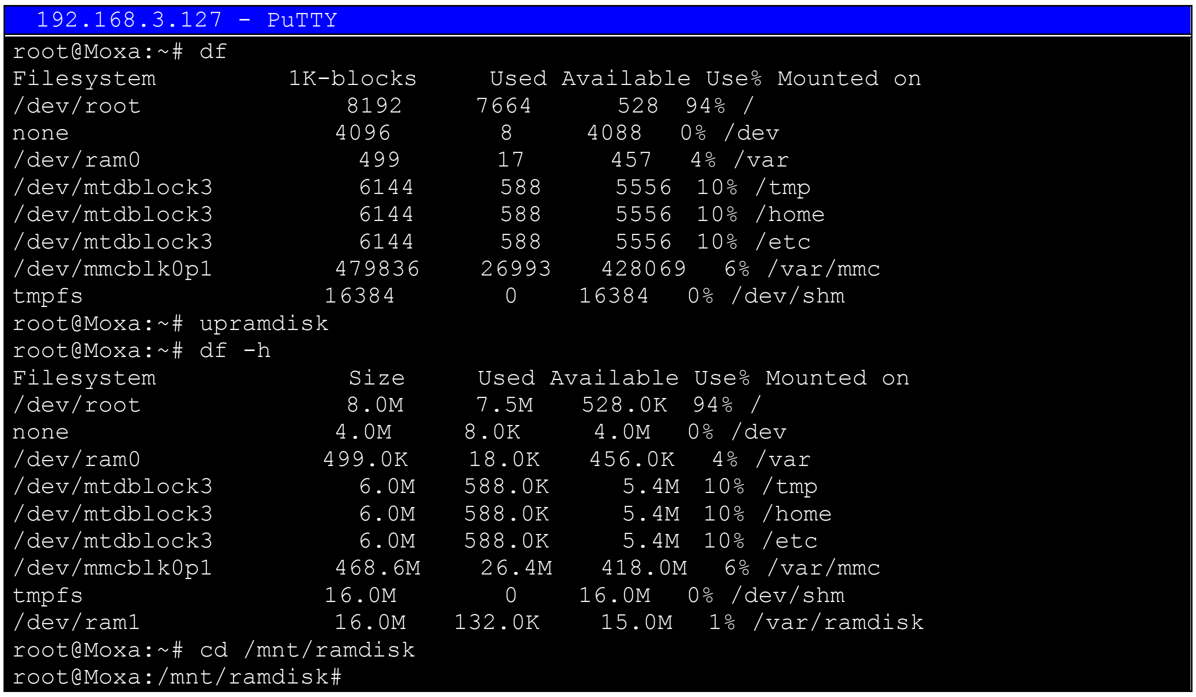
The W321/341's bios, kernel, and root file system are combined into one firmware file, which can be downloaded from Moxa's website www.moxa.com). The name of the file has the form **w321-x.x.x.frm** or **w341-x.x.x.frm**, with "x.x.x" indicating the firmware version. To upgrade the firmware, download the firmware file to a PC, and then transfer the file to the W321/341 using a console port or Telnet console connection.

ATTENTION

Upgrading the firmware will erase all data on the Flash ROM

If you are using the ramdisk to store code for your applications, beware that updating the firmware will erase all of the data on the Flash ROM. You should back up your application files and data before updating the firmware.

Since different Flash disks have different sizes, it is a good idea to check the size of your Flash disk before upgrading the firmware, or before using the disk to store your application and data files. Use the #df –h command to list the size of each memory block and how much free space is available in each block (see next page for commands)



```
192.168.3.127 - PuTTY
root@Moxa:~# df
Filesystem          1K-blocks     Used Available Use% Mounted on
/dev/root               8192      7664       528  94% /
none                   4096         8      4088   0% /dev
/dev/ram0               499        17       457   4% /var
/dev/mtdblock3          6144       588      5556  10% /tmp
/dev/mtdblock3          6144       588      5556  10% /home
/dev/mtdblock3          6144       588      5556  10% /etc
/dev/mmcblk0p1        479836     26993    428069   6% /var/mmc
tmpfs                  16384         0     16384   0% /dev/shm
root@Moxa:~# upramdisk
root@Moxa:~# df -h
Filesystem              Size      Used Available Use% Mounted on
/dev/root               8.0M      7.5M    528.0K  94% /
none                    4.0M      8.0K      4.0M   0% /dev
/dev/ram0             499.0K     18.0K    456.0K   4% /var
/dev/mtdblock3          6.0M    588.0K      5.4M  10% /tmp
/dev/mtdblock3          6.0M    588.0K      5.4M  10% /home
/dev/mtdblock3          6.0M    588.0K      5.4M  10% /etc
/dev/mmcblk0p1        468.6M     26.4M    418.0M   6% /var/mmc
tmpfs                  16.0M         0     16.0M   0% /dev/shm
/dev/ram1              16.0M    132.0K     15.0M   1% /var/ramdisk
root@Moxa:~# cd /mnt/ramdisk
root@Moxa:/mnt/ramdisk#
```

The following instructions give the steps required to save the firmware file to the W321/341's RAM disk and how to upgrade the firmware.

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

   #upramdisk

   #cd /mnt/ramdisk

2. Type the following commands to use the W321/341's built-in FTP client to transfer the firmware file **(FWR_W321_Va.b.c_Build_YYMMDDHH.hfm** or **FWR_W341_Va.b.c_Build_YYMMDDHH.hfm)** from the PC to the W321/341:

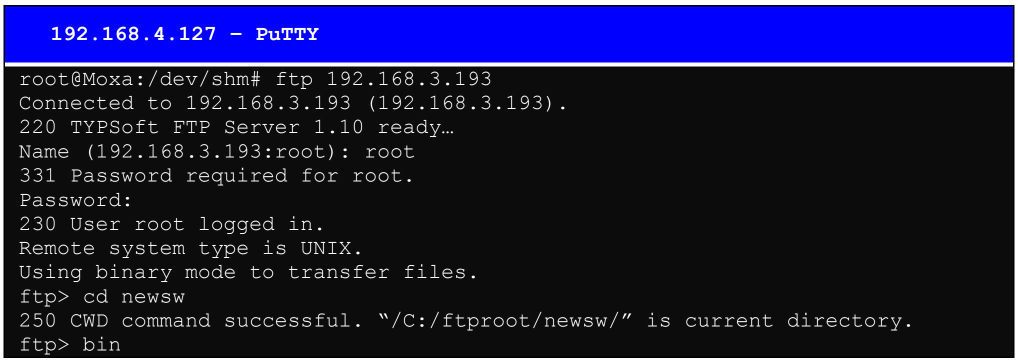
   /dev/shm> ftp <destination PC's IP>

   Login Name: xxxx

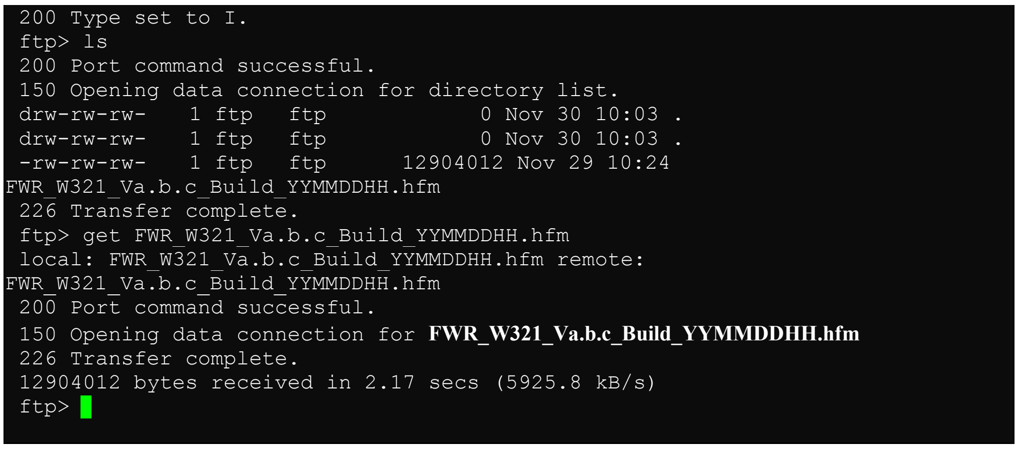
   Login Password: xxxx

   ftp> bin

   ftp> get FWR_W321_Va.b.c_Build_YYMMDDHH.hfm



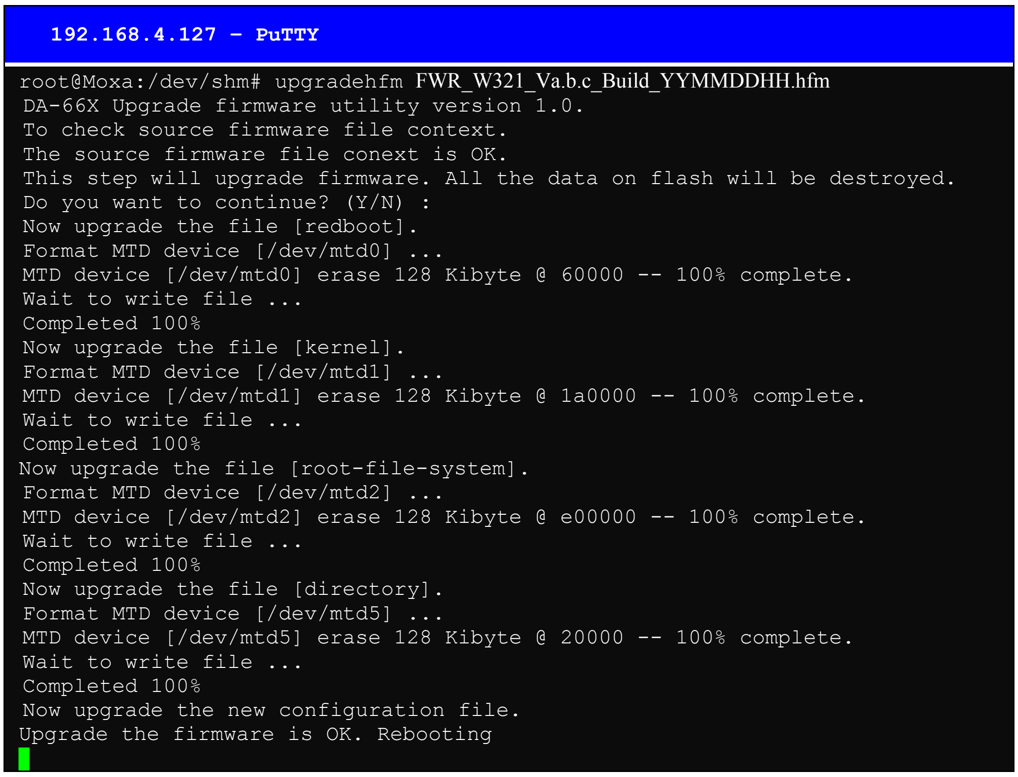
```
192.168.4.127 - PuTTY
root@Moxa:/dev/shm# ftp 192.168.3.193
Connected to 192.168.3.193 (192.168.3.193).
220 TYPSoft FTP Server 1.10 ready…
Name (192.168.3.193:root): root
331 Password required for root.
Password:
230 User root logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd newsw
250 CWD command successful. "/C:/ftproot/newsw/" is current directory.
ftp> bin
```



```
200 Type set to I.
ftp> ls
200 Port command successful.
150 Opening data connection for directory list.
drw-rw-rw-   1 ftp   ftp            0 Nov 30 10:03 .
drw-rw-rw-   1 ftp   ftp            0 Nov 30 10:03 .
-rw-rw-rw-   1 ftp   ftp      12904012 Nov 29 10:24
FWR_W321_Va.b.c_Build_YYMMDDHH.hfm
226 Transfer complete.
ftp> get FWR_W321_Va.b.c_Build_YYMMDDHH.hfm
local: FWR_W321_Va.b.c_Build_YYMMDDHH.hfm remote:
FWR_W321_Va.b.c_Build_YYMMDDHH.hfm
200 Port command successful.
150 Opening data connection for FWR_W321_Va.b.c_Build_YYMMDDHH.hfm
226 Transfer complete.
12904012 bytes received in 2.17 secs (5925.8 kB/s)
ftp>
```

3. Next, use the **`upgradehfm`** command to upgrade the kernel and root file system.

#upgradehfm FWR_W321_Va.b.c_Build_YYMMDDHH.hfm



```
192.168.4.127 - PuTTY

root@Moxa:/dev/shm# upgradehfm FWR_W321_Va.b.c_Build_YYMMDDHH.hfm
DA-66X Upgrade firmware utility version 1.0.
To check source firmware file context.
The source firmware file conext is OK.
This step will upgrade firmware. All the data on flash will be destroyed.
Do you want to continue? (Y/N) :
Now upgrade the file [redboot].
Format MTD device [/dev/mtd0] ...
MTD device [/dev/mtd0] erase 128 Kibyte @ 60000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [kernel].
Format MTD device [/dev/mtd1] ...
MTD device [/dev/mtd1] erase 128 Kibyte @ 1a0000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [root-file-system].
Format MTD device [/dev/mtd2] ...
MTD device [/dev/mtd2] erase 128 Kibyte @ e00000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the file [directory].
Format MTD device [/dev/mtd5] ...
MTD device [/dev/mtd5] erase 128 Kibyte @ 20000 -- 100% complete.
Wait to write file ...
Completed 100%
Now upgrade the new configuration file.
Upgrade the firmware is OK. Rebooting
```

ATTENTION

The upfirm utility will reboot your target after the upgrade is OK.

Loading Factory Defaults

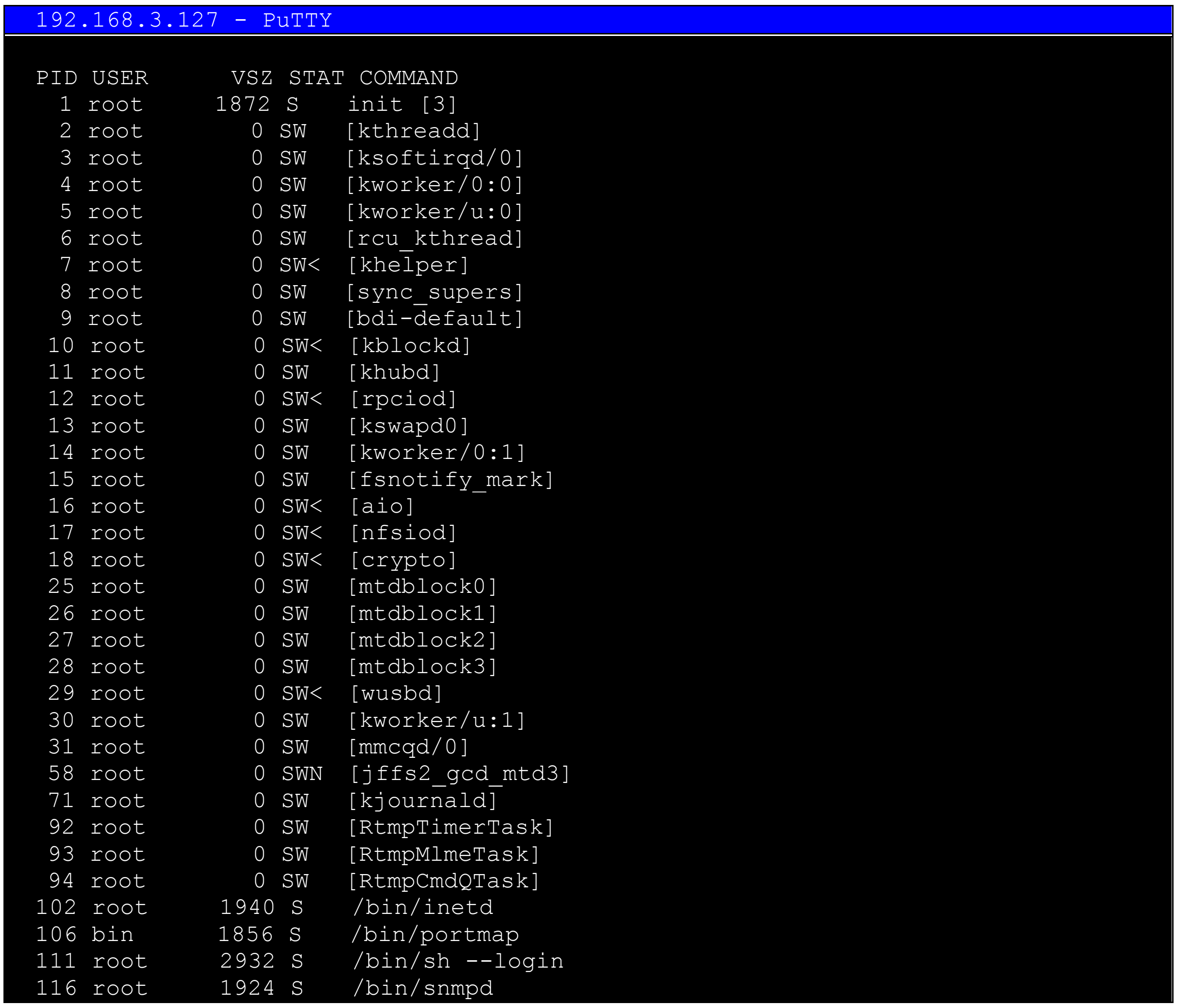
You can press the reset-to-default button over 5 seconds. The system will load the factory setting. And will destroy all on /home & /etc & /usr/local/bin & /usr/local/sbin & /usr/local/lib & /usr/local/libexec & /tmp directory files. The first 5 seconds the ready-light will be toggled for each one second. If you continue to press over 5 second, the read-light will be off and load the factory defaults.

Enabling and Disabling Daemons

The following daemons are enabled when the W321/341 unit boots up for the first time.

snmpd	SNMP Agent daemon
inetd	Internet Daemons
ftpd	FTP Server / Client daemon
sshd	Secure Shell Server daemon

Type the command **ps** to list all processes currently running.



```
192.168.3.127 - PuTTY

PID USER       VSZ STAT COMMAND
  1 root      1872 S    init [3]
  2 root         0 SW   [kthreadd]
  3 root         0 SW   [ksoftirqd/0]
  4 root         0 SW   [kworker/0:0]
  5 root         0 SW   [kworker/u:0]
  6 root         0 SW   [rcu_kthread]
  7 root         0 SW<  [khelper]
  8 root         0 SW   [sync_supers]
  9 root         0 SW   [bdi-default]
 10 root         0 SW<  [kblockd]
 11 root         0 SW   [khubd]
 12 root         0 SW<  [rpciod]
 13 root         0 SW   [kswapd0]
 14 root         0 SW   [kworker/0:1]
 15 root         0 SW   [fsnotify_mark]
 16 root         0 SW<  [aio]
 17 root         0 SW<  [nfsiod]
 18 root         0 SW<  [crypto]
 25 root         0 SW   [mtdblock0]
 26 root         0 SW   [mtdblock1]
 27 root         0 SW   [mtdblock2]
 28 root         0 SW   [mtdblock3]
 29 root         0 SW<  [wusbd]
 30 root         0 SW   [kworker/u:1]
 31 root         0 SW   [mmcqd/0]
 58 root         0 SWN  [jffs2_gcd_mtd3]
 71 root         0 SW   [kjournald]
 92 root         0 SW   [RtmpTimerTask]
 93 root         0 SW   [RtmpMlmeTask]
 94 root         0 SW   [RtmpCmdQTask]
102 root      1940 S    /bin/inetd
106 bin       1856 S    /bin/portmap
111 root      2932 S    /bin/sh --login
116 root      1924 S    /bin/snmpd
```

```
  122 root      4316 S    /bin/sshd -6 -f /etc/ssh/sshd_config
  132 root      4316 S    /bin/sshd -4 -f /etc/ssh/sshd_config
  161 root      6928 S    sshd: root@pts/0
  163 root      2864 S    -bash
  214 root         0 SW   [flush-1:0]
  215 root         0 SW   [flush-1:1]
  217 root      3096 R    ps
```

To run a private daemon, you can edit the file rc.local, as follows:

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

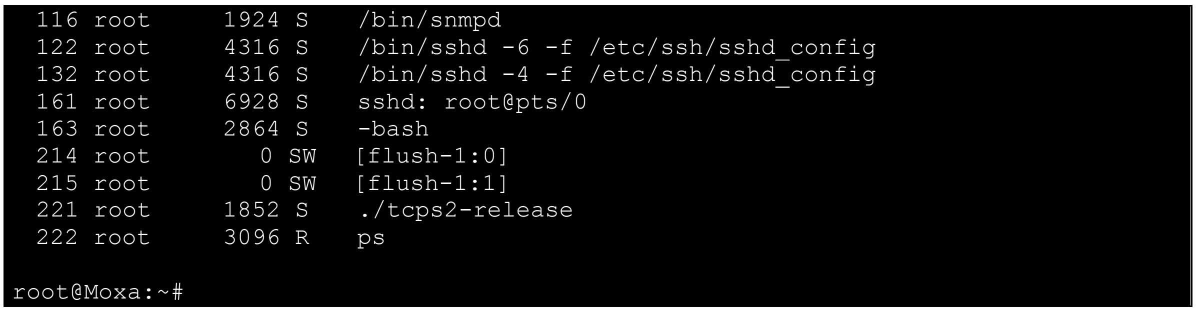
```
  192.168.3.127 - PuTTY
root@Moxa:~# cd /etc/rc.d
root@Moxa:~# /etc/rc.d# vi rc.local
```

Next, use **vi** to open your application program. We use the example program tcps2-release, and put it to run in the background.

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

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

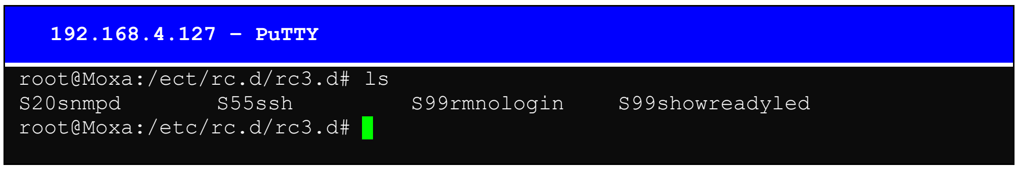
```
  192.168.3.127 - PuTTY

  PID USER       VSZ STAT COMMAND
    1 root      1872 S    init [3]
    2 root         0 SW   [kthreadd]
    3 root         0 SW   [ksoftirqd/0]
    4 root         0 SW   [kworker/0:0]
    5 root         0 SW   [kworker/u:0]
    6 root         0 SW   [rcu_kthread]
    7 root         0 SW<  [khelper]
    8 root         0 SW   [sync_supers]
    9 root         0 SW   [bdi-default]
   10 root         0 SW<  [kblockd]
   11 root         0 SW   [khubd]
   12 root         0 SW<  [rpciod]
   13 root         0 SW   [kswapd0]
   14 root         0 SW   [kworker/0:1]
   15 root         0 SW   [fsnotify_mark]
   16 root         0 SW<  [aio]
   17 root         0 SW<  [nfsiod]
   18 root         0 SW<  [crypto]
   25 root         0 SW   [mtdblock0]
   26 root         0 SW   [mtdblock1]
   27 root         0 SW   [mtdblock2]
   28 root         0 SW   [mtdblock3]
   29 root         0 SW<  [wusbd]
   30 root         0 SW   [kworker/u:1]
   31 root         0 SW   [mmcqd/0]
   58 root         0 SWN  [jffs2_gcd_mtd3]
   71 root         0 SW   [kjournald]
   92 root         0 SW   [RtmpTimerTask]
   93 root         0 SW   [RtmpMlmeTask]
   94 root         0 SW   [RtmpCmdQTask]
  102 root      1940 S    /bin/inetd
  106 bin       1856 S    /bin/portmap
  111 root      2932 S    /bin/sh --login
```



```
  116 root      1924 S    /bin/snmpd
  122 root      4316 S    /bin/sshd -6 -f /etc/ssh/sshd_config
  132 root      4316 S    /bin/sshd -4 -f /etc/ssh/sshd_config
  161 root      6928 S    sshd: root@pts/0
  163 root      2864 S    -bash
  214 root         0 SW   [flush-1:0]
  215 root         0 SW   [flush-1:1]
  221 root      1852 S    ./tcps2-release
  222 root      3096 R    ps

root@Moxa:~#
```

Setting the Run-Level

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



```
192.168.4.127 – PuTTY
root@Moxa:/ect/rc.d/rc3.d# ls
S20snmpd        S55ssh         S99rmnologin     S99showreadyled
root@Moxa:/etc/rc.d/rc3.d#
```

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

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

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

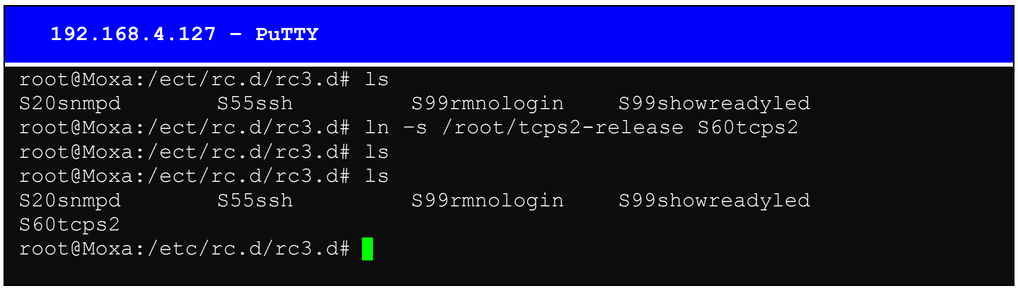
#ln –s /etc/rc.d/init.d/tcps2 S60tcps2

SxxRUNFILE stands for

S: start the run file while linux boots up.

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

RUNFILE: the file name.



```
192.168.4.127 – PuTTY
root@Moxa:/ect/rc.d/rc3.d# ls
S20snmpd        S55ssh         S99rmnologin     S99showreadyled
root@Moxa:/ect/rc.d/rc3.d# ln –s /root/tcps2-release S60tcps2
root@Moxa:/ect/rc.d/rc3.d# ls
root@Moxa:/ect/rc.d/rc3.d# ls
S20snmpd        S55ssh         S99rmnologin     S99showreadyled
S60tcps2
root@Moxa:/etc/rc.d/rc3.d#
```

KxxRUNFILE stands for

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

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

RUNFILE: the file name.

To remove the daemon, remove the run file from the **/etc/rc.d/rc3.d** directory by using the following command:

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

Adjusting the System Time

Setting the Time Manually

The W321/341 have two time settings. One is the system time, and the other is the RTC (Real Time Clock) time kept by the W321/341'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
```

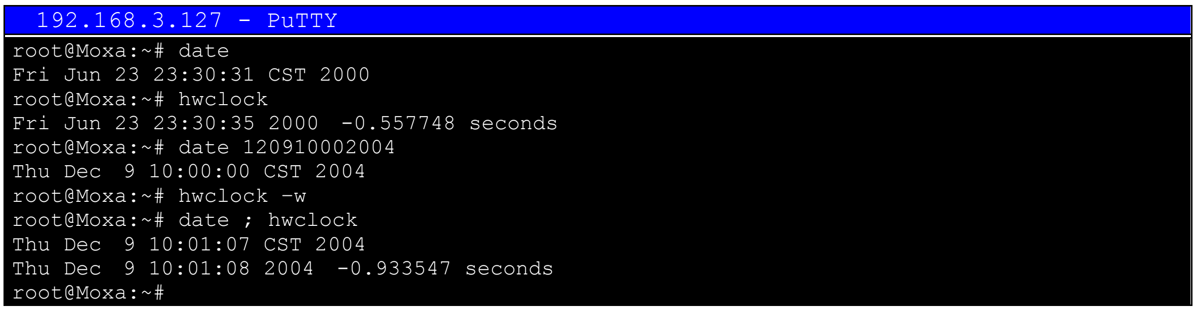
MM = Month
DD = Date
hhmm = hour and minute
YYYY = Year

Use the following command to set the RTC time:

```
#hwclock -w
```

Write current system time to RTC

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


192.168.3.127 - PuTTY

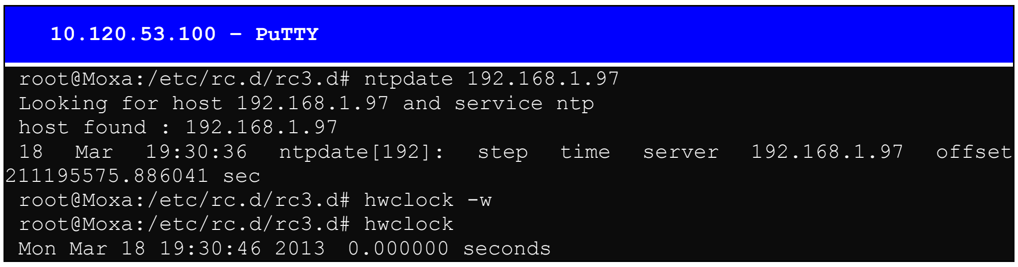
```
root@Moxa:~# date
Fri Jun 23 23:30:31 CST 2000
root@Moxa:~# hwclock
Fri Jun 23 23:30:35 2000  -0.557748 seconds
root@Moxa:~# date 120910002004
Thu Dec  9 10:00:00 CST 2004
root@Moxa:~# hwclock -w
root@Moxa:~# date ; hwclock
Thu Dec  9 10:01:07 CST 2004
Thu Dec  9 10:01:08 2004  -0.933547 seconds
root@Moxa:~#
```

NTP Client

The W321/341 have a built-in NTP (Network Time Protocol) client that is used to initialize a time request to a remote NTP server. Use **#ntpdate <this client utility>** to update the system time.

```
#ntpdate 192.168.1.97
#hwclock -w
```

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


10.120.53.100 - PuTTY

```
 root@Moxa:/etc/rc.d/rc3.d# ntpdate 192.168.1.97
 Looking for host 192.168.1.97 and service ntp
 host found : 192.168.1.97
 18  Mar  19:30:36  ntpdate[192]:  step  time  server  192.168.1.97  offset
211195575.886041 sec
 root@Moxa:/etc/rc.d/rc3.d# hwclock -w
 root@Moxa:/etc/rc.d/rc3.d# hwclock
 Mon Mar 18 19:30:46 2013  0.000000 seconds
```

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

Updating the Time Automatically

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

Example shell script to update the system time periodically

```
#!/bin/sh
ntpdate time.nist.gov
# You can use the time server's ip address or domain
# name directly. If you use domain name, you must
# enable the domain client on the system by updating
# /etc/resolv.conf file.
hwclock --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 run the shell script automatically when the kernel boots up

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

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

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

Cron—Daemon to Execute Scheduled Commands

Start Cron from the directory /etc/rc.d/rc. local. It will return immediately, so you don't need to start it with '&' 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 your scheduled applications. Crontab files have the following format:

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

The following example demonstrates how to use Cron.

How to use cron to update the system time and RTC time every day at 8:00.

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

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

Connecting Storage Peripherals

The W321/341 supports PNP (plug-n-play), and hot pluggability for connecting USB mass storage devices. W321/341 has an mdev auto mount utility that eases the mount procedure. The mdev auto mount utility default only supports mount one partition automatically. For using SD memory card storage, you should create one partition on a host computer. Here we create one partition on the Linux host computer by the fdisk utility.

```
192.168.3.120 - Putty
```

```
root@Moxa:/# fdisk /dev/mmcblk0

The number of cylinders for this disk is set to 15632.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of LILO)
2) booting and partitioning software from other OSs
   (e.g., DOS FDISK, OS/2 FDISK)

Command (m for help): p

Disk /dev/mmcblk0: 512 MB, 512229376 bytes
4 heads, 16 sectors/track, 15632 cylinders
Units = cylinders of 64 * 512 = 32768 bytes

       Device Boot     Start        End     Blocks  Id System

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-15632, default 1):
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-15632, default 15632):
Using default value 15632

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table
root@Moxa:/#
```

Next, you can format the partition.

```
192.168.3.120 - Putty
```

```
root@Moxa:/# mke2fs -t ext3 /dev/mmcblk0p1
```

After that you can remove the external storage and connected it to W321/341. The first connected SD card storage device will be mounted automatically by mount to **/mnt/sd**. W321/341 will be un-mounted automatically with umount when the device is disconnected.

ATTENTION

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

Remember to exit the **/mnt/sd** directory when you disconnect the SD memory card storage device. If you stay in **/mnt/sd**, the auto un-mount process will fail. If that happens, type **#umount /mnt/sd** to un-mount the SD card device manually.

ATTENTION

W321/341 only supports certain types of flash disk SD memory card storage device. Some the U SD card storage may not be compatible with W321/341. Check compatibility issues before you purchase a SD card storage device to connect to W321/341. **/etc/mdev.conf** is the mdev configuration file.

```
mmcblk0p[1-9] 0:0 0660 *(/sbin/automount_sd.sh $MDEV)
```

It automatically run **/sbin/automount_sd.sh** to mount the external storage.

```
#!/bin/sh

DISKNAME=`echo $1|cut -c1-3`
MPATH=/var/$DISKNAME
MPATHLOG=/dev/shm/$1

echo $1 > /dev/shm/mountpath.log

if [ "$1" == "" ]; then
        echo .automount.sh parameter is none.
        exit 1
fi

if [ -e "$MPATH" ]; then
        echo "$MPATH" > /dev/shm/umount.log
        umount $MPATH
        rm -rf $MPATH
else
        echo $MPATH > /dev/shm/mount.log
        if [ -b /dev/$1 ]; then
                mkdir -p $MPATH
                mount /dev/$1 $MPATH
        fi
fi
exit 0
```

If you need to customize the mounting script, you should change root file system partition to writable mode for editing **/sbin/automount.sh** script.

```
root@Moxa:/# mount -o remount,rw /
root@Moxa:/# vi /sbin/automount_sd.sh
root@Moxa:/# umount /
```

4 Managing Communications

In this chapter, we explain how to configure the W321/341’s various communication functions.

The following topics are covered in this chapter:

FTP

In addition to supporting Telnet client/server and FTP client/server, the W321/341 also support SSH and sftp client/server. To enable or disable the Telnet/ftp server, you first need to edit the file /etc/inetd.conf.

Enabling the ftp server

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

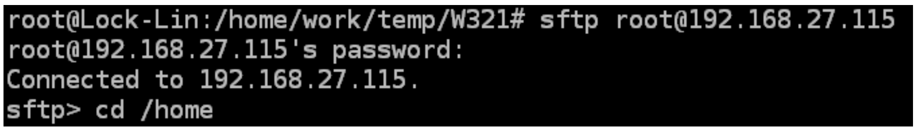
```
discard dgram udp wait root /bin/discard
discard stream tcp nowait root /bin/discard
#telnet stream tcp nowait root /bin/telnetd
ftp stream tcp nowait root /bin/ftpd -l
```

Disabling the Telnet/ftp server

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

SFTP

File Transfer Protocol (FTP) was once the most widely used protocol for transferring files between computers. However, because FTP sends authentication information and file contents over the wire unencrypted, it's not a secure way to communicate. SSH File Transfer Protocol (SFTP) addresses this security concern by providing data transfer over a fully encrypted channel. You can use these alternatives for transferring files securely over the Internet or any other untrusted network.



```
root@Lock-Lin:/home/work/temp/W321# sftp root@192.168.27.115
root@192.168.27.115's password:
Connected to 192.168.27.115.
sftp> cd /home
```

DNS

The W321/341 support DNS client (but not DNS server). To set up DNS client, you need to edit three configuration files: /etc/hosts, /etc/resolv.conf, and /etc/nsswitch.conf.

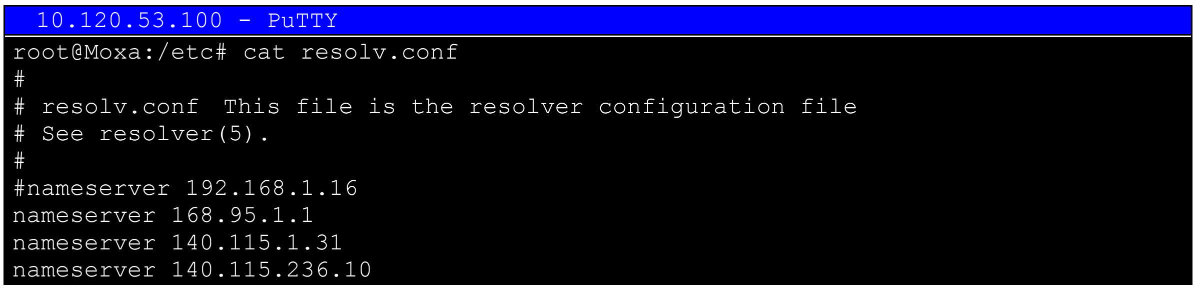
/etc/hosts

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

/etc/resolv.conf

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

```
nameserver 168.95.1.1
```



```
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
```

`/etc/nsswitch.conf`

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

IPTABLES

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

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

The W321/341 support 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 even if these computers 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.

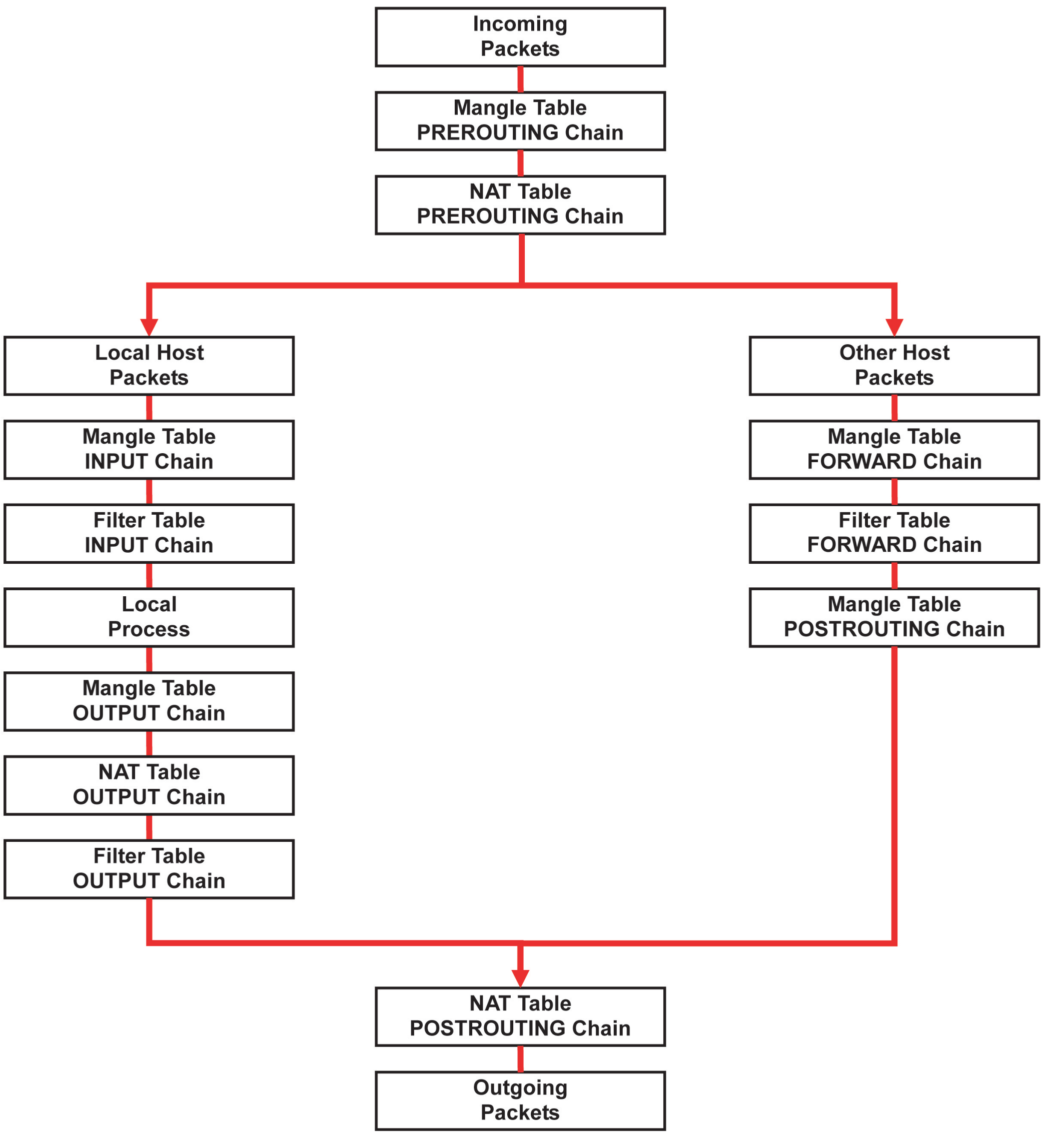
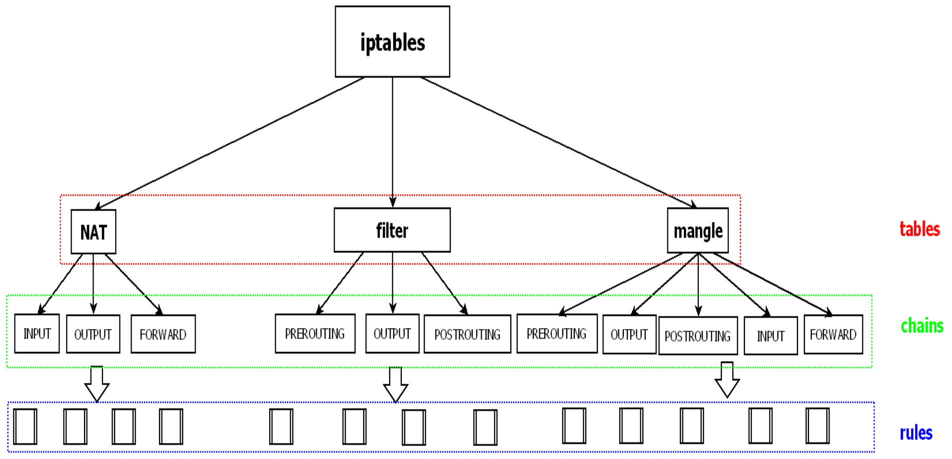

Incoming Packets
Mangle Table PREROUTING Chain
NAT Table PREROUTING Chain
Local Host Packets
Mangle Table INPUT Chain
Filter Table INPUT Chain
Local Process
Mangle Table OUTPUT Chain
NAT Table OUTPUT Chain
Filter Table OUTPUT Chain
Other Host Packets
Mangle Table FORWARD Chain
Filter Table FORWARD Chain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

Table	**Chain**	**Rule**
NAT Network translation translation)	REROUTING	**Types of rule** ■ **Policy** ■ **Self-defined** **Targets of rule** ■ **ACCEPT** ■ **DROP** ■ **REJECT** ■ **LOG** ■ SNAT ■ DNAT ■ MASQUERADE …
	POSTROUTING	
	OUTPUT	
Filter (Default) (Packet filtering)	INPUT	
	OUTPUT	
	FORWARD	
Mangle (Packet header	PREROUTING	

modification)	INPUT	
	FORWARD	
	OUTPUT	
	POSTROUTING	



The W321/341 supports the following sub-modules. Be sure to use the module that matches your application.

x_tables.ko	xt_hl.ko	nfnetlink.ko	xt_DSCP.ko
xt_mark.ko	nfnetlink_queue.ko	xt_HL.ko	xt_tcpudp.ko
arp_tables.ko	ipt_MASQUERADE.ko	iptable_nat.ko	arpt_mangle.ko
ipt_REJECT.ko	iptable_raw.ko	arptable_filter.ko	ipt_ULOG..ko
nf_conntrack_ipv4.ko	ip_queue.ko	ipt_addrtype.ko	nf_defrag_ipv4.ko
ip_tables.ko	ipt_ecn.ko	nf_nat.ko	ipt_ECN.ko
iptable_filter.ko	ipt_LOG..ko	iptable_mangle.ko	

NOTE The W321/341 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 **lsmod** to check if the ip_tables module has already been loaded in the W321/341 unit. Use **modprobe** to insert and enable the module.

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

```
#modprobe iptable_filter
```

Use the following commands to maintain the database:

```
#iptables, #iptables-restore #iptables-save
```

NOTE IPTABLES plays the role of packet filtering or NAT. Take care when setting up the IPTABLES rules. If the rules are not correct, remote hosts that connect via a LAN or PPP may be denied access. We recommend using the serial console to set up the IPTABLES.

Click on the following links for more information about iptables.
http://www.linuxguruz.com/iptables/
http://www.netfilter.org/documentation/HOWTO//packet-filtering-HOWTO.html

Since the IPTABLES command is very complex, to illustrate the IPTABLES syntax we have divided our discussion of the various rules into three categories: **Observe and erase chain rules, Define policy rules,** and **Append or delete rules**.

Observe and erase chain rules

Usage:

```
# iptables [-t tables] [-L] [-n]
```

-t tables: Table to manipulate (default: 'filter'); example: nat or filter.
-L [chain]: List List all rules in selected chains. If no chain is selected, all chains are listed.
-n: Numeric output of addresses and ports.

```
# iptables [-t tables] [-FXZ]
```

-F: Flush the selected chain (all the chains in the table if none is listed).
-X: Delete the specified user-defined chain.
-Z: Set the packet and byte counters in all chains to zero.

Examples:

```
# iptables -L -n
```

In this example, since we do not use the -t parameter, the system uses the default 'filter' table. Three chains are included: INPUT, OUTPUT, and FORWARD. INPUT chains are accepted automatically, and all connections are accepted without being filtered.

```
#iptables –F
#iptables –X
#iptables –Z
```

Define policy for chain rules

Usage:

```
# iptables [-t tables] [-P] [INPUT, OUTPUT, FORWARD, PREROUTING, OUTPUT, POSTROUTING]
[ACCEPT, DROP]
```

-P: Set the policy for the chain to the given target.
INPUT: For packets coming into the W321/341.
OUTPUT: For locally-generated packets.
FORWARD: For packets routed out through the W321/341.
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 W341'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`.**

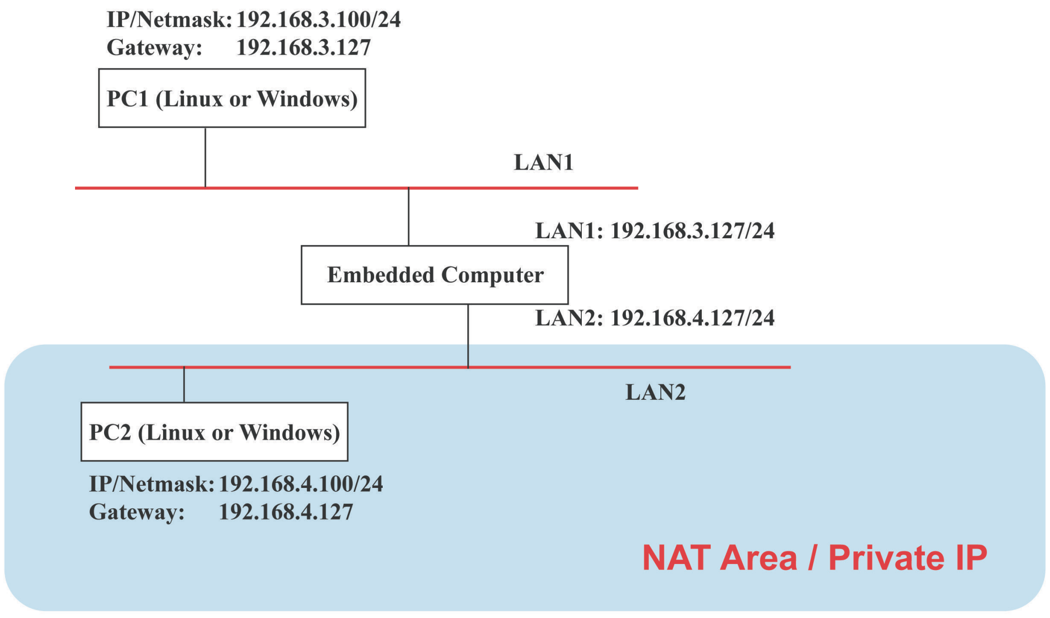
NAT

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 W321/341 connect several devices on a network and maps local inside network addresses to one or more global outside IP addresses, and un-maps the global IP addresses on incoming packets back into local IP addresses.

NOTE Click 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 192.168.3.127 (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 192.168.3.127`
8. `#iptables -t nat -A POSTROUTING -o eth0 -s 192.168.3.0/24 -j MASQUERADE`

Enabling NAT at Bootup

In most real world situations, you will want to use a simple shell script to enable NAT when the W341 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 a serial link. PPP can be used for direct serial connections (using a null-modem cable) over a Telnet link, and links established using a modem over a telephone line.

Modem / PPP access is almost identical to connecting directly to a network through the W321/341's Ethernet port. Since PPP is a peer-to-peer system, the W321/341 can also use PPP to link two networks (or a local network to the Internet) to create a Wide Area Network (WAN).

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

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

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

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

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

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

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

The pppd options are described below:

connect 'chat etc...'

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

-v
verbose mode; log what we do to syslog

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

ATDT5551212
Dial the modem, and then ...

CONNECT
Wait for an answer.

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

ogin: username word: password
Log in with username and password.

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

/dev/
Specify the callout serial port.

115200
The baudrate.

debug
Log status in syslog.

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

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

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

192.1.1.17
This is a degenerate case of a general option of the form x.x.x.x:y.y.y.y. Here x.x.x.x is the local IP address and y.y.y.y is the IP address of the remote end of the PPP connection. If this option is not specified, or if just one side is specified, then x.x.x.x defaults to the IP address associated with the local machine's hostname (located in /etc/hosts), and y.y.y.y is determined by the remote machine.

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

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

#pppd connect 'chat –v " " " " ' *noipdefault* /dev/ttyM0 19200 crtscts "

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

#pppd connect 'chat –v " " " " ' user *root* password *root noipdefault*
/dev/ttyM0 19200 crtscts

How to check the connection

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

/sbin/ifconfig

(The folder **ifconfig** may be located elsewhere, depending on your distribution.) You should be able to see all the network interfaces that are UP. ppp0 should be one of them, 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 one machine:

```
lo          Link encap Local Loopback
            inet addr 127.0.0.1   Bcast 127.255.255.255       Mask 255.0.0.0
            UP LOOPBACK RUNNING         MTU 2000              Metric 1
            RX packets 0 errors 0 dropped 0 overrun 0
ppp0        Link encap Point-to-Point Protocol
            inet addr 192.76.32.3     P-t-P 129.67.1.165      Mask 255.255.255.0
            UP POINTOPOINT RUNNING      MTU 1500              Metric 1
            RX packets 33 errors 0 dropped 0 overrun 0
            TX packets 42 errors 0 dropped 0 overrun 0
```

Now, type:

`ping z.z.z.z`

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

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

Try typing:

`netstat –nr`

This should show three routes, something like this:

```
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**:

`*    *    ""   *`

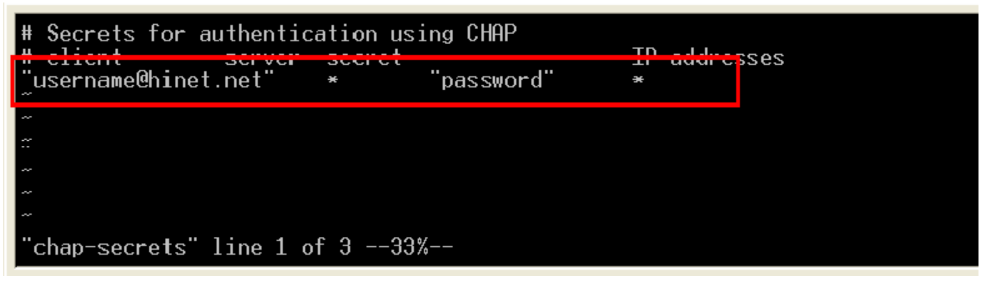
The first star (*) lets everyone login. The second star (*) lets every host connect. The pair of double quotation marks ("") is to use the file /etc/passwd to check the password. The last star (*) is to let any IP connect.

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

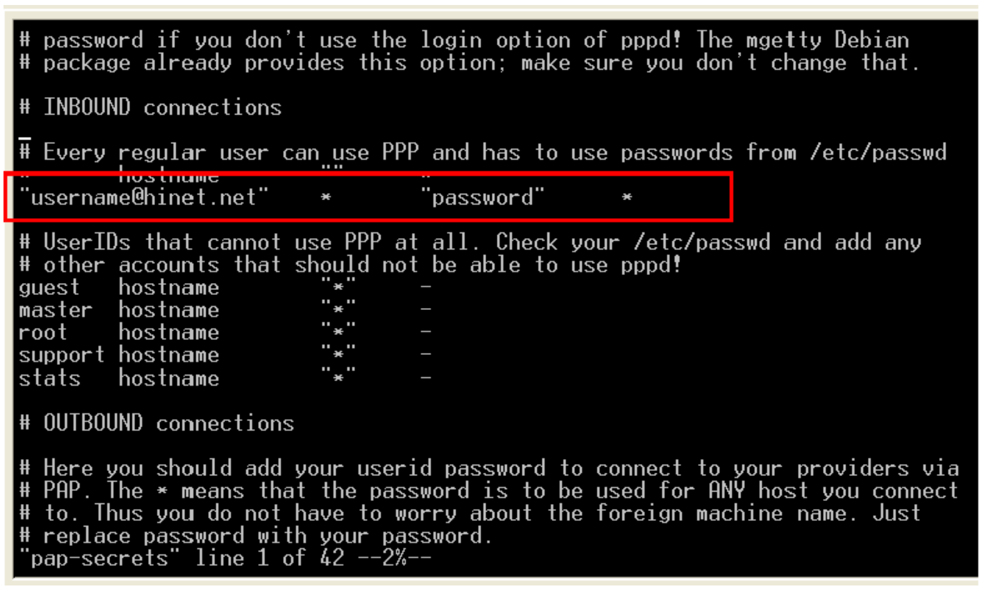
1. Connect the W321/341's LAN port to an ADSL modem with a cross-over cable, HUB, or switch.
2. Log in to the W321/341 as the root user.
3. Edit the file **/etc/ppp/chap-secrets** and add the following: **"username@hinet.net"*"password"***



```
# Secrets for authentication using CHAP
# client        server  secret                  IP addresses
"username@hinet.net"    *       "password"      *
~
~
~
~
~
~
"chap-secrets" line 1 of 3 --33%--
```

"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.

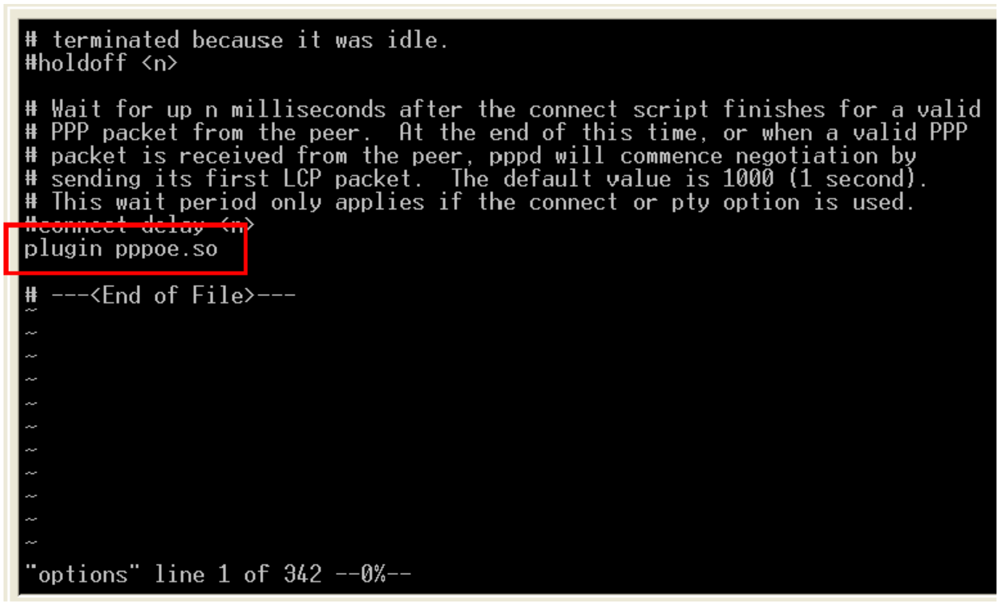
4. Edit the file **/etc/ppp/pap-secrets** and add the following:
   "username@hinet.net"*"password"*



```
# password if you don't use the login option of pppd! The mgetty Debian
# package already provides this option; make sure you don't change that.

# INBOUND connections

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

# 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        "*"     -

# OUTBOUND connections

# Here you should add your userid password to connect to your providers via
# PAP. The * means that the password is to be used for ANY host you connect
# to. Thus you do not have to worry about the foreign machine name. Just
# replace password with your password.
"pap-secrets" line 1 of 42 --2%--
```

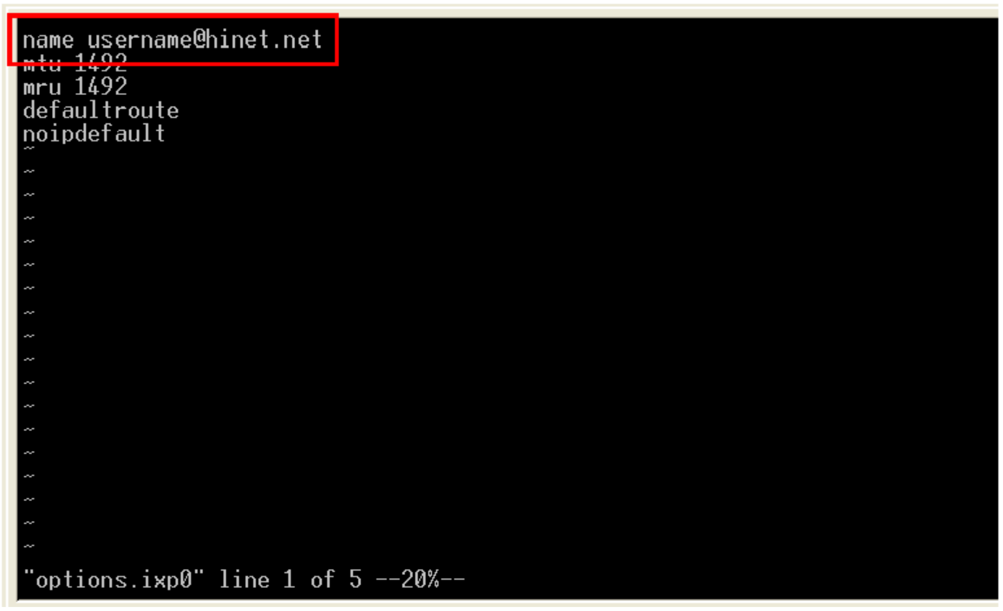
"username@hinet.net" is the username obtained from the ISP to log in to the ISP account. **"password"** is the corresponding password for the account.

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



```
# terminated because it was idle.
#holdoff <n>

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

# ---<End of File>---
~
~
~
~
~
~
~
~
~
~
~
"options" line 1 of 342 --0%--
```

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



```
name username@hinet.net
mtu 1492
mru 1492
defaultroute
noipdefault
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"options.ixp0" line 1 of 5 --20%--
```

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

7. Set up DNS

   If you are using DNS servers supplied by your ISP, edit the file

   /etc/resolv.conf by adding the following lines of code:

   nameserver ip_addr_of_first_dns_server

   nameserver ip_addr_of_second_dns_server

   For example:

   nameserver 168..95.1.1

   nameserver 139.175.10.20

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

   pppd eth0

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

pppd eth1

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

NFS (Network File System)

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 W321/341, without worrying about the amount of disk space that will be available. The W321/341 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 W321/341 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 192.168.3.100:/home/public /home/nfs/public
```

SNMP

The W321/341 have built-in SNMP V1 (Simple Network Management Protocol) agent software. It supports RFC1317 RS-232 like group and RFC 1213 MIB-II.

The following simple example allows you to use an SNMP browser on the host site to query the W321/341, which is the SNMP agent. The W321/341 will respond.

```
debian:~# snmpwalk -v 1 -c public -Cc 192.168.27.115
iso.3.6.1.2.1.1.1.0 = STRING: "Linux version 2.6.38.8 (root@Lock-Lin) (gcc version 4.4.2 (GCC) ) #46 Thu Mar 14 09:36:46 CST
2013
"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8691.12.321
iso.3.6.1.2.1.1.3.0 = Timeticks: (901200) 2:30:12.00
iso.3.6.1.2.1.1.4.0 = STRING: "Moxa Inc., Embedded Computing Business."
iso.3.6.1.2.1.1.5.0 = STRING: "Moxa"
iso.3.6.1.2.1.1.6.0 = STRING: "Fl.8 No.6, Alley 6, Lane 235, Pao-Chiao Rd., Shing Tien City, Taipei, Taiwan, R.O.C."
iso.3.6.1.2.1.1.7.0 = INTEGER: 6
iso.3.6.1.2.1.2.1.0 = INTEGER: 2
iso.3.6.1.2.1.2.2.1.1.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.1.2 = INTEGER: 2
iso.3.6.1.2.1.2.2.1.2.1 = STRING: "eth0"
iso.3.6.1.2.1.2.2.1.2.2 = STRING: "wlan0"
iso.3.6.1.2.1.2.2.1.3.1 = INTEGER: 6
iso.3.6.1.2.1.2.2.1.3.2 = INTEGER: 6
iso.3.6.1.2.1.2.2.1.4.1 = INTEGER: 1500
iso.3.6.1.2.1.2.2.1.4.2 = INTEGER: 1500
iso.3.6.1.2.1.2.2.1.5.1 = Gauge32: 100000000
iso.3.6.1.2.1.2.2.1.5.2 = Gauge32: 100000000
iso.3.6.1.2.1.2.2.1.6.1 = Hex-STRING: 00 90 E8 77 10 0B
iso.3.6.1.2.1.2.2.1.6.2 = Hex-STRING: 00 0E 8E 41 0A 42
iso.3.6.1.2.1.2.2.1.7.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.7.2 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.8.1 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.8.2 = INTEGER: 1
iso.3.6.1.2.1.2.2.1.9.1 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.2.2.1.9.2 = Timeticks: (0) 0:00:00.00
iso.3.6.1.2.1.2.2.1.10.1 = Counter32: 5785056
iso.3.6.1.2.1.2.2.1.10.2 = Counter32: 63760787
iso.3.6.1.2.1.2.2.1.11.1 = Counter32: 56041
iso.3.6.1.2.1.2.2.1.11.2 = Counter32: 396459
iso.3.6.1.2.1.2.2.1.12.1 = Counter32: 0
iso.3.6.1.2.1.2.2.1.12.2 = Counter32: 0
iso.3.6.1.2.1.2.2.1.13.1 = Counter32: 9174
iso.3.6.1.2.1.2.2.1.13.2 = Counter32: 0
…
iso.3.6.1.2.1.7.5.1.1.192.168.27.115.111 = IpAddress: 192.168.27.115
iso.3.6.1.2.1.7.5.1.1.192.168.27.115.161 = IpAddress: 192.168.27.115
iso.3.6.1.2.1.7.5.1.2.192.168.27.115.111 = INTEGER: 111
iso.3.6.1.2.1.7.5.1.2.192.168.27.115.161 = INTEGER: 161
iso.3.6.1.2.1.11.1.0 = Counter32: 191
iso.3.6.1.2.1.11.2.0 = Counter32: 191
iso.3.6.1.2.1.11.3.0 = Counter32: 0
iso.3.6.1.2.1.11.4.0 = Counter32: 0
iso.3.6.1.2.1.11.5.0 = Counter32: 0
iso.3.6.1.2.1.11.6.0 = Counter32: 0
iso.3.6.1.2.1.11.8.0 = Counter32: 0
iso.3.6.1.2.1.11.9.0 = Counter32: 0
iso.3.6.1.2.1.11.10.0 = Counter32: 0
iso.3.6.1.2.1.11.11.0 = Counter32: 0
iso.3.6.1.2.1.11.12.0 = Counter32: 0
iso.3.6.1.2.1.11.13.0 = Counter32: 201
iso.3.6.1.2.1.11.14.0 = Counter32: 0
iso.3.6.1.2.1.11.15.0 = Counter32: 0
iso.3.6.1.2.1.11.16.0 = Counter32: 204
iso.3.6.1.2.1.11.17.0 = Counter32: 0
iso.3.6.1.2.1.11.18.0 = Counter32: 0
iso.3.6.1.2.1.11.19.0 = Counter32: 0
iso.3.6.1.2.1.11.20.0 = Counter32: 0
iso.3.6.1.2.1.11.21.0 = Counter32: 0
iso.3.6.1.2.1.11.22.0 = Counter32: 0
iso.3.6.1.2.1.11.24.0 = Counter32: 0
iso.3.6.1.2.1.11.25.0 = Counter32: 0
iso.3.6.1.2.1.11.26.0 = Counter32: 0
iso.3.6.1.2.1.11.27.0 = Counter32: 0
iso.3.6.1.2.1.11.28.0 = Counter32: 215
iso.3.6.1.2.1.11.29.0 = Counter32: 0
iso.3.6.1.2.1.11.30.0 = INTEGER: 2
End of MIB
```

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

→ W321/341 do NOT support SNMP trap.

OpenVPN

OpenVPN provides two types of tunnels for users to implement VPNS: **Routed IP Tunnels** and **Bridged Ethernet Tunnels**. To begin with

1. check to make sure that the system has a virtual device /dev/net/tun. If not, issue the following command:

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

2. Enable OpenVPN driver

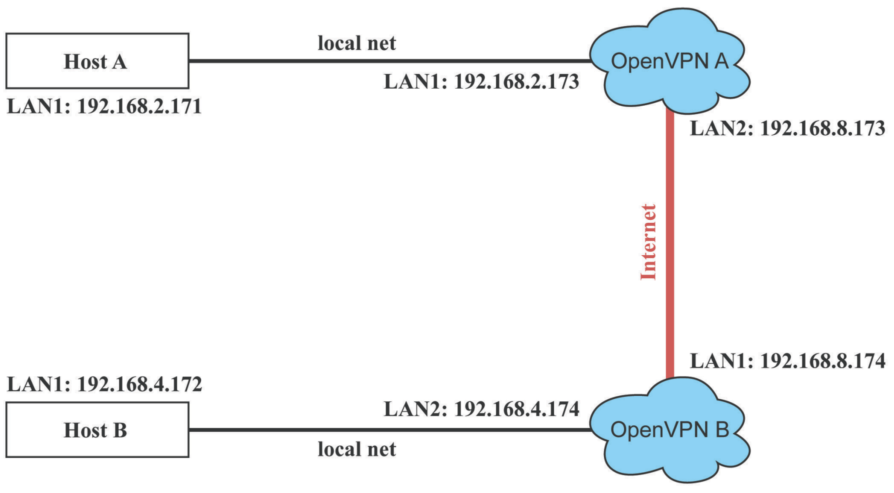
```
root@Moxa:/# modprobe tun
```

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

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

Setup 1: Ethernet Bridging for Private Networks on Different Subnets

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



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

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

Copy the file that is generated to the OpenVPN machine. **Be sure that both two of the devices have the same key.**

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 -----------------------------
```

Create link symbols to enable this script at boot time:

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

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
```

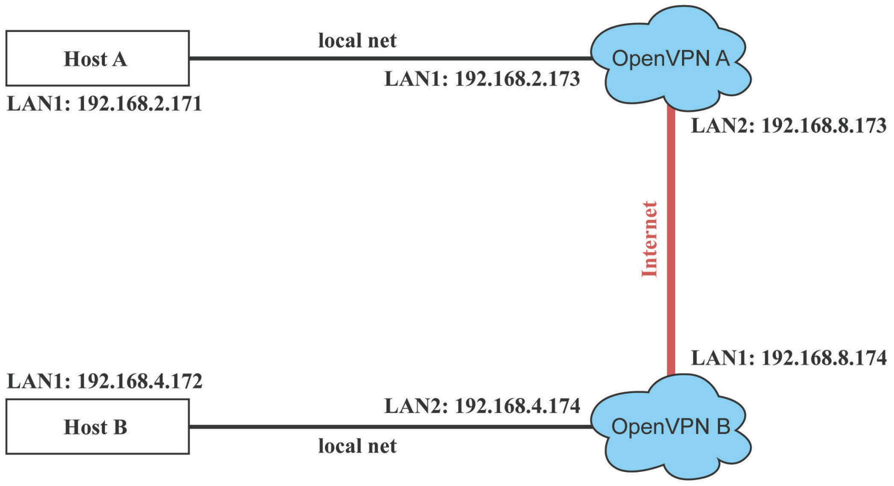
A successful ping indicates that you have created a VPN system that only allows authorized users from one internal network to access users at the remote site. For this system, all data is transmitted by UDP packets on port 5000 between OpenVPN peers.

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

```
# killall -TERM openvpn
```

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

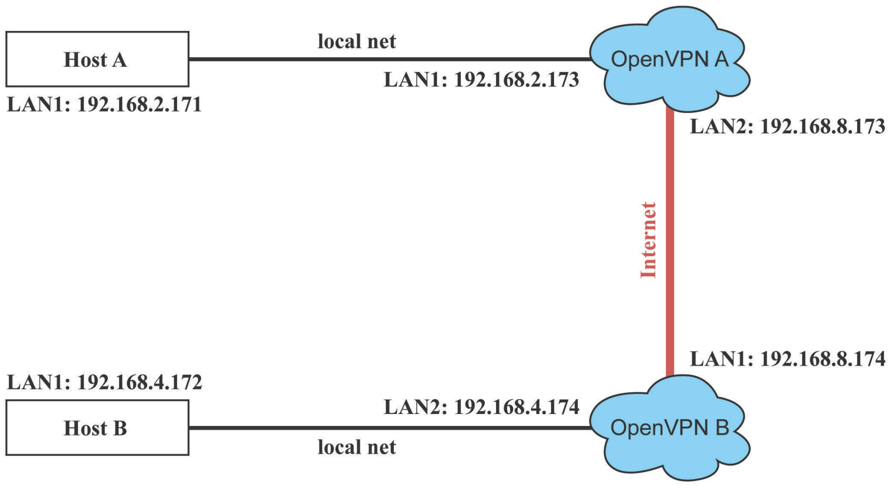
1. Set up four machines as shown in the following diagram:



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

This chapter describes how to install a tool chain in the host computer that you use to develop your applications. In addition, the process of performing cross-platform development and debugging are also introduced. For clarity, the W321/341 embedded computer is called a target computer.

The following topics are covered in this chapter:

- **Linux Tool Chain**
  - Steps for Installing the Linux Tool Chain
  - Compilation for Applications

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 in your host computer (PC) running Linux. We have confirmed that the following Linux distributions can be used to install the tool chain.

Steps for Installing the Linux Tool Chain

The tool chain needs about 485 MB of hard disk space. To install it, follow the steps.

1. Insert the package CD into your PC and then issue the following commands:
   ```
   #mount/dev/cdrom /mnt/cdrom
   #sh/mnt/cdrom/tool-chain/W321/arm-linux_3.1_Build_11111411.sh
   ```
2. Wait for the installation process to complete. This should take a few minutes.
3. Add the directory **/usr/local/arm-linux-4.4.2/bin** to your path. You can do this for the current login by issuing the following commands:
   ```
   #export PATH="/usr/local/arm-linux-4.4.2/bin:$PATH"
   ```
   Alternatively, you can add the same commands to **$HOME/.bash_profile** to make it effective for all login sessions.

Compilation for 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, but with an additional prefix that specifies the target system. In the case of x86 environments, the prefix is `i386-linux-` and in the case of W321/341 ARM boards, it is `arm-linux-4.4.2`.

For example, the native C compiler is `gcc` and the cross C compiler for ARM in the W321/341 is `arm-linux-gcc`.

The following cross compiler tools are provided:

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

6

Programmer's Guide

This chapter includes important information for programmers.

The following topics are covered in this chapter:

Flash Memory Map

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

Address	Size	Contents
0x80000000 – 0x80040000	256KB	Boot Loader—Read ONLY
0x80040000– 0x80200000	1.75 MB	Kernel object code—Read ONLY
0x80200000– 0x80a00000	8 MB	Root file system (JFFS2) —Read ONLY
0x80a00000– 0x81000000	6MB	User root file system (JFFS2) —Read/Write

NOTE

1. The default Moxa file system only enables the network. It lets users recover the user file system when it fails.
2. The user file system is a complete file system. Users can create and delete directories and files (including source code and executable files) as needed.
3. Users can create the user file system on the PC host or target platform, and then copy it to the W321/341.
4. To improve system performance, we strongly recommend that you install your application programs on the on-board flash. However, since the on-board flash has a fixed amount of free memory space, you must not over-write it, and instead use an external storage card, such as an SD for the data log.

Device API

The W321/341 support control devices with the **ioctl** system API. You will need to **include <moxadevice.h>**, and use the following **ioctl** function.

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

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

```
#man ioctl
```

RTC (Real Time Clock)

The device node is located at **/dev/rtc**. The W321/341 support Linux standard simple RTC control. You must include **<linux/rtc.h>**.

1. Function: RTC_RD_TIME
   ```
   int ioctl(fd, RTC_RD_TIME, struct rtc_time *time);
   ```
   Description: read time information from RTC. It will return the value on argument 3.
2. Function: RTC_SET_TIME
   ```
   int ioctl(fd, RTC_SET_TIME, struct rtc_time *time);
   ```
   Description: set RTC time. Argument 3 will be passed to RTC.

Buzzer

The device node is located at **/dev/console**. The W321/341 support Linux standard buzzer control, with the W321/341's buzzer running at a fixed frequency of 100 Hz. You must **include <sys/kd. h>**.

Function: KDMKTONE

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

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

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

WDT (Watch Dog Timer)

1. **Introduction**

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

2. **How the WDT works**

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

   Kernel boot
   .....
   ....

   User application running and enable user ack
   ....
   ....

3. **The user API**
   1. Pinging the watchdog using an ioctl:

      All drivers that have an ioctl interface support at least one ioctl, KEEPALIVE. This ioctl does exactly the same thing as a write to the watchdog device, so the main loop in the above program could be replaced with:

      ```
      while (1) {
              ioctl(fd, WDIOC_KEEPALIVE, 0);
              sleep(10);
      }
      ```

      the argument to the ioctl is ignored.
   2. For some drivers it is possible to modify the watchdog timeout on the fly with the SETTIMEOUT ioctl, those drivers have the WDIOF_SETTIMEOUT flag set in their option field. The argument is an integerrepresenting the timeout in seconds. The driver returns the real timeout used in the same variable, and this timeout might differ from the requested one due to limitation of the hardware.

      ```
      int timeout = 45;
      ioctl(fd, WDIOC_SETTIMEOUT, &timeout);
      printf("The timeout was set to %d seconds\n", timeout);
      ```

This example might actually print "The timeout was set to 60 seconds" if the device has a granularity of minutes for its timeout.

3. Query the current timeout using the GETTIMEOUT ioctl.

```
ioctl(fd, WDIOC_GETTIMEOUT, &timeout);

printf("The timeout was is %d seconds\n", timeout);
```

Example 1:

```
/*
 * Watchdog Driver Test Program
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/watchdog.h>

int fd;

/*
 * This function simply sends an IOCTL to the driver, which in turn ticks
 * the PC Watchdog card to reset its internal timer so it doesn't trigger
 * a computer reset.
 */
static void keep_alive(void)
{
   int dummy;

   ioctl(fd, WDIOC_KEEPALIVE, &dummy);
}

/*
 * The main program.  Run the program with "-d" to disable the card,
 * or "-e" to enable the card.
 */
int main(int argc, char *argv[])
{

   int flags, get_timeout = 0;
   int set_timeout = 60;
   fd = open("/dev/watchdog", O_WRONLY);

   if (fd == -1) {
   fprintf(stderr, "Watchdog device not enabled.\n");
   fflush(stderr);
   exit(-1);
```

```
    }

    if (argc > 1) {
    if (!strncasecmp(argv[1], "-d", 2)) {
       flags = WDIOS_DISABLECARD;
       ioctl(fd, WDIOC_SETOPTIONS, &flags);
       fprintf(stderr, "Watchdog card disabled.\n");
       fflush(stderr);
       exit(0);
    } else if (!strncasecmp(argv[1], "-e", 2)) {
       flags = WDIOS_ENABLECARD;
       ioctl(fd, WDIOC_SETOPTIONS, &flags);
       fprintf(stderr, "Watchdog card enabled.\n");
       fflush(stderr);
       exit(0);
    } else {
       fprintf(stderr, "-d to disable, -e to enable.\n");
       fprintf(stderr, "run by itself to tick the card.\n");
       fflush(stderr);
       exit(0);
    }
    } else {
    fprintf(stderr, "Watchdog Ticking Away!\n");
    fflush(stderr);
    }

    ioctl(fd, WDIOC_SETTIMEOUT, &set_timeout);
    printf("The timeout was set to %d seconds\n", set_timeout);
    ioctl(fd, WDIOC_GETTIMEOUT, &get_timeout);
    printf("The timeout was is %d seconds\n", get_timeout);

    while(1) {
    keep_alive();
    sleep(1);
    }
}
```

The makefile is shown below:

```
all:
      arm-linux-gcc –o xxxx xxxx.c
```

Example 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(void)
{
    int fd = open("/dev/watchdog", O_WRONLY);
    int ret = 0;
```

```
        if (fd == -1) {
            perror("watchdog");
            exit(EXIT_FAILURE);
        }
        while (1) {
            ret = write(fd, "\0", 1);
            if (ret != 1) {
                ret = -1;
                break;
            }
            sleep(10);
        }
        close(fd);
        return ret;
}
```

The makefile is shown below:

```
all:
        arm-linux-gcc -o xxxx xxxx.c
```

Digital Input/Output(W321 only)

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

Application Programming Interface

Return error code definitions:

#define DIO_ERROR_PORT -1 // no such port

#define DIO_ERROR_MODE -2 // no such mode or state

#define DIO_ERROR_CONTROL -3 // open or ioctl fail

#define DIO_ERROR_DURATION -4 // The value of duration is not 0 or not in the range,
40 <= duration <= 3600000 milliseconds (1 hour)

#define DIO_ERROR_DURATION_20MS -5 // The value of duration must be a multiple of 20 ms

#define DIO_OK 0

The definition for DIN and DOUT:

#define DIO_HIGH 1

#define DIO_LOW 0

int set_dout_state(int doport, int state)

Description: To set the DOUT port to high or low state.

Input: int doport - which DOUT port you want to set. Port starts from 0 to 3.

int state - to set high or low state; DIO_HIGH (1) for high, DIO_LOW (0) for low.

Output: none.

Return: reference the error code.

int get_din_state(int diport, int *state)

Description: To get the DIN port state.

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

int *state - save the current state.

Output: state - DIO_HIGH (1) for high, DIO_LOW (0) for low.

Return: reference the error code.

int get_dout_state(int doport, int *state)

Description: To get the DOUT port state.

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

int *state - save the current state.

Output: state - DIO_HIGH (1) for high, DIO_LOW (0) for low.

Return: reference the error code.

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

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

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

Port numbering is from 0 to 3.

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

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

NULL

> Clears this event

int mode DIN_EVENT_HIGH_TO_LOW

(1): from high to low

DIN_EVENT_LOW_TO_HIGH

(0): from low to high

DIN_EVENT_CLEAR

(-1): clear this event

unsigned long duration - 0: detect the din event > DIN_EVENT_HIGH_TO_LOW or DIN_EVENT_LOW_TO_HIGH> without duration

- Not 0

> detect the din event

DIN_EVENT_HIGH_TO_LOW or

DIN_EVENT_LOW_TO_HIGH with

duration. The value of "duration" must be a

multiple of 20 milliseconds. The range of

"duration" is 0, or 40 <= duration <= 3600000

milliseconds. The error of the measurement is

24 ms. For example, if the DIN duration is

200 ms, this event will be generated when the

DIN pin stays in the same state for a time

between 176 ms and 200 ms.

Output: none.

Return: reference the error code.

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

Description: To retrieve the DIN event configuration, including mode

(DIN_EVENT_HIGH_TO_LOW or DIN_EVENT_LOW_TO_HIGH), and the value of "duration."

Input: `int diport` - which DIN port you want to retrieve.

- The port whose din event setting we wish to retrieve

`int *mode` - save which event is set.

`unsigned long *duration` - the duration of the DIN port is kept in high or low state.

- return to the current duration value of diport

Output: `mode` DIN_EVENT_HIGH_TO_LOW

(1): from high to low

DIN_EVENT_LOW_TO_HIGH(0): from low to high

DIN_EVENT_CLEAR(-1): clear this event

`duration` The value of duration should be 0 or 40 <= duration

<= 3600000 milliseconds.

Return: reference the error code.

Special Note

Don’t forget to link the library **libmoxalib which is under /usr/local/arm-linux-4.4.2/lib** for DI/DO programming, and also include the header file **moxadevice.h which is under/usr/local/arm-linux-4.4.2/include**. The DI/DO library only can be used by one program at a time.

Examples

Example 1

File Name: tdio.c

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

```
#include <stdio.h>
#include <stdlib.h>
#include <moxadevice.h>
#include <fcntl.h>
#ifdef DEBUG
#define dbg_printf(x...) printf(x)
#else
#define dbg_printf(x...)
#endif
#define MIN_DURATION 40
static char *DataString[2]={"Low ", "High "};
static void hightolowevent(int diport)
{
printf("\nDIN port %d high to low.\n", diport);
}
static void lowtohighevent(int diport)
{
printf("\nDIN port %d low to high.\n", diport);
}
int main(int argc, char * argv[])
{
int i, j, state, retval;
unsigned long duration;
  while( 1 ) {
    printf("\nSelect a number of menu, other key to exit. \n\
    1. set high to low event \n\
    2. get now data. \n\
    3. set low to high event \n\
    4. clear event \n\
    5. set high data. \n\
    6. set low data. \n\
    7. quit \n\
    8. show event and duration \n\
    Choose : ");
    retval =0;
    scanf("%d", &i);
    if ( i == 1 ) { // set high to low event
      printf("Please keyin the DIN number : ");
      scanf("%d", &i);
      printf("Please input the DIN duration, this minimun value must be over %d : ", MIN_DURATION);
      scanf("%lu", &duration);
      retval=set_din_event(i, hightolowevent, DIN_EVENT_HIGH_TO_LOW, duration);
    } else if ( i == 2 ) { // get now data
      printf("DIN data : ");
      for ( j=0; j<4; j++ ) {
        get_din_state(j, &state);
        printf("%s", DataString[state]);
      }
      printf("\n");
      printf("DOUT data : ");
      for ( j=0; j<MAX_DOUT_PORT; j++ ) {
        get_dout_state(j, &state);
        printf("%s", DataString[state]);
      }
      printf("\n");
      } else if ( i == 3 ) { // set low to high event
        printf("Please keyin the DIN number : ");
        scanf("%d", &i);
        printf("Please input the DIN duration, this minimun value must be over %d :",
MIN_DURATION);
        scanf("%lu", &duration);
        retval = set_din_event(i, lowtohighevent, DIN_EVENT_LOW_TO_HIGH, duration);
      } else if ( i == 4 ) { // clear event
        printf("Please keyin the DIN number : ");
        scanf("%d", &i);
        retval=set_din_event(i, NULL, DIN_EVENT_CLEAR, 0);
      } else if ( i == 5 ) { // set high data
        printf("Please keyin the DOUT number : ");
        scanf("%d", &i);
        retval=set_dout_state(i, 1);
      } else if ( i == 6 ) { // set low data
        printf("Please keyin the DOUT number : ");
           scanf("%d", &i);
           retval=set_dout_state(i, 0);
         } else if ( i == 7 ) { // quit
           break;
         } else if ( i == 8 ) { // show event and duration
           printf("Event:\n");
         for ( j=0; j<MAX_DOUT_PORT; j++ ) {
           retval=get_din_event(j, &i, &duration);
```

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

DI/DO Program Makefile Example

```
FNAME=tdio
CC= arm-linux-gcc
STRIP=arm-linux-strip
LIBS += -L/usr/local/arm-linux-4.4.2/lib
CFLAGS += -I/usr/local/arm-linux-4.4.2/include
release:
   $(CC) $(LDFLAGS) $(CFLAGS) -o $(FNAME) $(FNAME).c -lmoxalib -lpthread
   $(STRIP) -s $(FNAME)
debug:
   $(CC) -DDEBUG -o $(FNAME)-dbg $(FNAME).cxx -lmoxalib -lpthread
clean:
   /bin/rm -f $(FNAME) $(FNAME)-dbg *.o
```

UART

The normal tty device node is located at **/dev/ttyM0 … ttyM3**.

The W321/341 support Linux standard termios control. The Moxa UART Device API allows you to configure ttyM0 to ttyM3 as RS-232, RS-422, 4-wire RS-485, or 2-wire RS-485. The W321/341 support RS-232, RS-422, 2-wire RS-485, and 4-wire RS485.

You must **include <moxadevice.h>.**

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

1. Function: MOXA_SET_OP_MODE

   int ioctl(fd, MOXA_SET_OP_MODE, &mode)

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

2. Function: MOXA_GET_OP_MODE

   int ioctl(fd, MOXA_GET_OP_MODE, &mode)

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

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

Function: MOXA_SET_SPECIAL_BAUD_RATE
Function: MOXA_GET_SPECIAL_BAUD_RATE

If you use this ioctl to set a special baudrate, the termios cflag will be B4000000, in which case the B4000000 define will be different. If the baudrate you get from termios (or from calling tcgetattr()) is B4000000, you must call ioctl with MOXA_GET_SPECIAL_BAUD_RATE to get the actual baudrate.

Example to set the baudrate

```
#include <moxadevice.h>
#include <termios.h>
struct termios term;
int fd, speed;
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
term. c_cflag &= ~(CBAUD | CBAUDEX);
term.c_cflag |= B4000000;
tcsetattr(fd, TCSANOW, &term);
speed = 500000;
ioctl(fd, MOXA_SET_SPECIAL_BAUD_RATE, &speed);
```

Example to get the baudrate

```
#include <moxadevice.h>
#include <termios.h>
struct termios term;
int fd, speed;
fd = open("/dev/ttyM0", O_RDWR);
tcgetattr(fd, &term);
if ( (term.c_cflag & (CBAUD|CBAUDEX)) != B4000000 ) {
  // follow the standard termios baud rate define
} else {
  ioctl(fd, MOXA_GET_SPECIAL_BAUD_RATE, &speed);
}
```

Baudrate inaccuracy

Divisor = 921600/Target Baud Rate. (Only Integer part)
ENUM = 8 * (921600/Targer - Divisor) (Round up or down)
Inaccuracy = (Target Baud Rate – 921600/(Divisor + (ENUM/8))) / Target Baud Rate * 100%
E.g.,
To calculate 500000 bps
Divisor = 1, ENUM = 7,
Inaccuracy = 1.7%
*The Inaccuracy should less than 2% for work reliably.

Special Note

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

Relay Output (W341 only)

The W341 uses a DO (digital ouput) for relay output. Programming with the following API allows you to change the state of the digital output and to get the current state of the digital output.

Example

Description: The program indicates how to control DO.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#define IOCTL_SET_DOUT   15

typedef struct _DGTIO
{
    int port;
    int data;
} DGTIO;

int main(void)
{
    int fd = open("/dev/relay_do", O_WRONLY);
    int ret = 0;
    DGTIO dio;

    if (fd == -1) {
        perror("open relay DO device failed");
```

```
            exit(EXIT_FAILURE);
        }

        dio.port = 0;
        dio.data = 1;

        /* set Relay DO high */
        ioctl(fd, IOCTL_SET_DOUT, &dio);
        sleep (5);

        dio.data = 0;

        /* set Relay DO low*/
        ioctl(fd, IOCTL_SET_DOUT, &dio);
        close(fd);
        return ret;
    }
```

7 Software Lock

“Software Lock” is an innovative technology developed by the Moxa engineering team. 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’s engineerings 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’ (under /usr/local/arm-linux-4.4.2/bin/ after installing Tool Chain) 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, and 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**	displays the current working directly
12.	**df**	displays the amount of free space on the device
13.	**mkdir**	make new directory
14.	**rmdir**	delete directory
15.	**find**	search for files in a directory hierarchy
16.	**head**	output the first part of files
17.	**mkfifo**	make a FIFO special file (a named pipe)
18.	**mknod**	creates a FIFO, character special file, or block special file with the specified name
19.	**touch**	change file timestamps
20.	**which**	Locate a program file in the user's path.

Editor

1.	**vi**	text editor
2.	**cat**	dump file context
3.	**zcat**	compress or expand files
4.	**grep**	search file for a specific pattern
5.	**egrep**	search string on file of Extended regular expressions
6.	**grep**	Search file(s) for lines that match a fixed string
7.	**cut**	get string on file
8.	**find**	find files
9.	**more**	dump file page by page
10.	**test**	test if file exists or not
11.	**sleep**	sleep (seconds)

12.	**usleep**	suspend execution for microsecond intervals
13.	**echo**	echo string
14.	**sed**	stream editor
15.	**awk**	pattern-directed scanning and processing language
16.	**expand**	converts all tabs to spaces
17.	**tail**	print the last 10 lines of each FILE to standard output.
18.	**tar**	the GNU version of the tar archiving utility
19.	**tr**	translate, squeeze, and/or delete characters
20.	**wc**	print byte, word, and line counts, count the number of bytes, whitespace-separated words, and newlines in each given FILE, or standard input if non are given or for a FILE of '-'.

Network

1.	**arp**	manipulate the system ARP cache
2.	**ping**	ping to test network
3.	**route**	routing table manager
4.	**netstat**	display network status
5.	**ifconfig**	set network IP address
6.	**tftp**	IPV4 Trivial File Transfer Protocol client
7.	**telnet**	Connects the local host with a remote host, using the Telnet interface.
8.	**ftp**	file transfer protocol
9.	**ifdown, ifup**	bring a network interface up, or take a network interface down
10.	**ip**	show / manipulate routing, devices, policy routing and tunnels
11.	**tcpsvd**	TCP/IP service daemon
12.	**wget**	the non-interactive network downloader

Process

1.	**kill**	kill process
2.	**ps**	display now running process
3.	**fuser**	identify processes using files or sockets
4.	**killall**	sends a signal to all processes running any of the specified commands
5.	**nice, renice**	run a program with modified scheduling priority / alter priority of running processes
6.	**pidof**	find the process ID of a running program
7.	**run-parts**	run scripts or programs in a directory
8.	**start-stop daemon**	start and stop system daemon programs
9.	**top**	display Linux tasks

Modules

1.	**insmod**	insert a module into the kernel
2.	**lsmod**	icely formats the contents of the /proc/ modules, showing what kernel modules are currently loaded
3.	**modprobe**	intelligently adds or removes a module from the Linux kernel
4.	**rmmod**	remove module from kernel

Other

1.	**dmesg**	dump kernel log message
2.	**stty**	to set serial port
3.	**zcat**	dump .gz file context
4.	**free**	display system memory usage
5.	**date**	print or set the system date and time
6.	**env**	run a program in a modified environment
7.	**clear**	clear the terminal screen
8.	**reboot**	reboot / power off/on the server
9.	**halt**	halt the server
10.	**du**	estimate file space usage
11.	**gzip, gunzip**	compress or expand files
12.	**hostname**	show system's host name
13.	**dirname**	convert a full pathname to just a path
14.	**expr**	evaluate arguments as an expression
15.	**false**	do nothing, returning a non-zero (false) exit status
16.	**true**	do nothing, successfully
17.	**fdisk**	partition table manipulator for Linux
18.	**hwclock**	a tool for accessing the Hardware Clock.
19.	**id**	print the user identity
20.	**klogd**	kernel log daemon
21.	**logger**	a shell command interface to the syslog system log module
22.	**md5sum**	compute and check MD5 message digest
23.	**mesg**	control write access to your terminal
24.	**mktemp**	make temporary file name
25.	**nohup**	no Hang Up
26.	**reset**	terminal initialization
27.	**sty**	change and print terminal line settings
28.	**syslogd**	Linux system logging utilities.
29.	**Uname**	print system information, print information about the machine and operating system it is run on.
30.	**Uptime**	Tell how long the system has been running.
31.	**Xargs**	build and execute command lines from standard input
32.	**yes**	print the command line arguments, separated by spaces and followed by a newline, forever until it is killed.
33.	**tee**	copy standard input to each FILE, and also to standard output.

Moxa Special Utilities

1.	**kversion**	show kernel version
2.	**upramdisk**	mount ramdisk
3.	**downramdisk**	unmount ramdisk
4.	**setinterface**	set /dev/ttyMn to RS232/RS485-2WIRES/RS422/ RS485-4WIRES
5.	**setkey**	set the software encryption key
6.	**upgradehfm**	upgrade firmware utility