Copyright © 2000-2001 ARM Limited. All rights reserved.

Release Information

The following changes have been made to this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sept 2000</td>
<td>A</td>
<td>First release</td>
</tr>
<tr>
<td>June 2001</td>
<td>B</td>
<td>Second release</td>
</tr>
</tbody>
</table>

Proprietary Notice

ARM, the ARM Powered logo, Thumb, and StrongARM are registered trademarks of ARM Limited.

The ARM logo, AMBA, PrimeCell, Angel, ARMulator, EmbeddedICE, ModelGen, MultiICE, ARM7TDMI, ARM7TDMI-S, ARM9TDMI, TDMI, and STRONG are trademarks of ARM Limited.

Portions of source code are provided under the copyright of the respective owners, and are acknowledged in the appropriate source files:

Copyright © 1998-1999 by Interniche Technologies Inc.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole or any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with prior written permission of the copyright holder.

The product described in this document is subject to continuous development and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Ltd shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.
Contents

Command-line Interface Reference Guide

Preface
   About this book ................................................................. vi
   Feedback .............................................................................. ix

Chapter 1 Overview of the Command-Line Interface
   1.1 About the Command-Line Interface ............................... 1-2
   1.2 The console ............................................................... 1-3
   1.3 Runtime commands .................................................... 1-4
   1.4 Sample session .......................................................... 1-5

Chapter 2 General Commands
   2.1 General commands ................................................... 2-2
   2.2 UDP Echo commands ................................................. 2-11
   2.3 TCP Echo commands ................................................ 2-16
   2.4 PPP commands ........................................................ 2-21
   2.5 SNMP commands ...................................................... 2-23
   2.6 NV parameters .......................................................... 2-24

Chapter 3 Diagnostic Commands
   3.1 General diagnostic commands .................................... 3-2
   3.2 Statistics commands .................................................. 3-14
   3.3 DNS commands ........................................................ 3-18
3.4 TCP commands ................................................................. 3-20
3.5 Modem commands ............................................................ 3-26
3.6 HTTP commands .............................................................. 3-28
3.7 PPP commands ............................................................... 3-31
3.8 Memory command ........................................................... 3-34
3.9 IP commands ................................................................. 3-35
3.10 SNMP command ............................................................. 3-38

Chapter 4 Protocol-specific Commands
4.1 DHCP server commands ......................................................... 4-2
4.2 Email Alerter commands ..................................................... 4-8
4.3 FTP client commands ......................................................... 4-15
4.4 Ping commands ............................................................... 4-21
4.5 NAT Router commands ....................................................... 4-28
4.6 Routing Information Protocol (RIP) commands ......................... 4-34
4.7 TELNET commands ......................................................... 4-39
This preface introduces the Command-line Interface Reference Guide. It contains the following sections:

- *About this book* on page vi
- *Feedback* on page ix.
About this book

This guide is provided with the ARM Portable TCP/IP stack sources.

It is assumed that the ARM TCP/IP sources are available as a reference. It is also assumed that the reader has access to a C language programmer’s guide and the ARM Architectural Reference Manual.

Intended audience

This Reference Guide is written for a moderately-experienced C programmer, with a general understanding of TCP/IP, who wants to port the stack to a new environment.

Using this book

This book is organized into the following chapters:

Chapter 1 Overview of the Command-Line Interface

Read this chapter for introductory information on the Command-line Interface (CLI).

Chapter 2 General Commands

Read this chapter for syntax and examples of general commands for the protocols.

Chapter 3 Diagnostic Commands

Read this chapter for syntax and examples of diagnostic and statistics commands for the protocols.

Chapter 4 Protocol-specific Commands

Read this chapter for syntax and examples of commands for additional protocols. The Protocol-specific commands are available only if you have licensed and included the additional protocols as a part of your system.

Examples of optional protocols are NATRouter, RIP, FTP, TELNET, SNMP, Emailer, WebPort, and DHCP. Their related commands are only available when they have been built into your executable.
Typographical conventions

The following typographical conventions are used in this book:

**typewriter** Denotes text that may be entered at the keyboard, such as commands, file and program names, and source code.

**typewriter** Denotes a permitted abbreviation for a command or option. The underlined text may be entered instead of the full command or option name.

**typewriter italic**

Denotes arguments to commands and functions where the argument is to be replaced by a specific value.

**italic** Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

**bold** Highlights interface elements, such as menu names and buttons. Also used for terms in descriptive lists, where appropriate.

**typewriter bold**

Denotes language keywords when used outside example code and ARM processor signal names.
Further reading

This section lists publications from both ARM Limited and third parties that provide additional information on porting ARM Network Protocols.

ARM publications

This book contains reference information that is specific to ARM Network Protocols. For additional information, refer to the following ARM publications:

- **ARM Architecture Reference Manual** (ARM DDI 0100)
- **ARM Developer Suite (ADS)** documentation set.

Other publications

For other reference information, please refer to the following:

Feedback

ARM Limited welcomes feedback on both ARM Network Protocols and its documentation.

Feedback on ARM Network Protocols

If you have any problems with ARM Network Protocols, please contact your supplier. To help us provide a rapid and useful response, please give:

• details of the release you are using
• details of the platform you are running on, such as the hardware platform, operating system type and version
• a small stand-alone sample of code that reproduces the problem
• a clear explanation of what you expected to happen, and what actually happened
• the commands you used, including any command-line options
• sample output illustrating the problem.

Feedback on this book

If you have any comments on this book, please send email to errata@arm.com giving:

• the document title
• the document number
• the page number(s) to which your comments apply
• a concise explanation of your comments.

General suggestions for additions and improvements are also welcome.
Chapter 1
Overview of the Command-Line Interface

This chapter gives introductory information on the Command-Line Interface (CLI) console, commands, and shows a short sample session. It contains the following sections:

• About the Command-Line Interface on page 1-2
• The console on page 1-3
• Runtime commands on page 1-4
• Sample session on page 1-5.
1.1 About the Command-Line Interface

This document describes how to use the ARM Network Protocols Command-Line Interface (CLI). It provides debugging, monitoring, and configuration commands for the TCP/IP stack and the rest of the ARM Network Protocols Suite.

The CLI consists of:

- **Instrumentation**: This can be compiled into all ARM Network Protocols products.
- **Console**: This is the computer on which webport.exe is executing.
- **Menu system**: This provides a list of commands and usage information at the console.

You can use the CLI for development only, include it in your final product, or not use it at all.
1.2 The console

When an executable image runs, if it is built with support for the CLI, it initializes and gives the INET> prompt.

The device where this prompt is displayed is determined by the porting engineer and varies depending on the nature of the target system.

In the Menus demo, the prompt is output on a serial port and user input is accepted from the serial port.

In the Telnet demonstration program, the Telnet server outputs the prompt to a connected Telnet client, and input is accepted from his client.

For purpose of this documentation, this input/output device is called the target system console.

More than one console may be active at the same time.
1.3 Runtime commands

Commands are available in an executable incorporating the ARM TCP/IP stack when the option IN_MENUS is defined in the file ipport.h.

After initialization messages the console shows the following prompt:

INET>_

At this point, you can enter any command processing.

1.3.1 Entering commands and parameters

The command interface only requires that you enter enough of the command name to uniquely identify the command. Many commands can also take one or more parameters typed on the command line. Parameters can be abbreviated to as few as a single letter when the meaning is unambiguous.
1.4 Sample session

As an introduction, a short annotated session transcript is reproduced here:

- The `host` command sets a default remote host IP address for those commands that require a remote host to be specified.
  
  INET> **host 10.0.0.5**
  
  active host number set to 10.0.0.5

- The `ping` command checks if a host is reachable or not.
  
  INET> **ping**
  
  ping 0 to 10.0.0.5: ping 1 sent...
  
  got ping reply; len:62 seq 0 from 10.0.0.5

- The `quit` (or `qui`) command is invoked and the execution ends.
  
  INET> **q**
Overview of the Command-Line Interface
Chapter 2
General Commands

This chapter describes general commands. It contains the following sections:
- General commands on page 2-2
- UDP Echo commands on page 2-11
- TCP Echo commands on page 2-16
- PPP commands on page 2-21
- SNMP commands on page 2-23
- NV parameters on page 2-24.
2.1 General commands

The general commands in this section are:

- help (or ?) on page 2-2
- state on page 2-4
- quit on page 2-4
- history on page 2-5
- obey on page 2-6
- logfile on page 2-7
- sleep on page 2-8
- setup on page 2-8
- version on page 2-9
- ! on page 2-9.

2.1.1 help (or ?)

The help (or ?) command lists all the other commands.

Syntax

```
help commandlist
```

where:

- `commandlist` is an optional parameter that can take the following settings:
  - `general` Displays the most commonly used commands.
  - `diagnostic` Displays the commands used for reporting statistics.
  - `module_name` Displays the appropriate command set for the named module. For example, `help ping` displays the commands for ping.

If this parameter is omitted, the general commands are listed.

The value of `commandlist` can be abbreviated when the meaning is unambiguous.
Example

INET> help
general commands:
  help     - help with menus
  state    - show current station setup
  uesend   - open UDP echo client, send UDP echo packet
  uesinit  - start UDP echo server
  uechalt  - close UDP echo client
  ueshalt  - close UDP echo server
  uestats  - UDP echo statistics
  tesend   - open TCP echo client, send TCP echo packet
  tesinit  - start TCP echo server
  teshalt  - close TCP echo server
  techalt  - close TCP echo client
  testats  - TCP echo statistics
  quit     - quit station program
  nvset    - set non-volatile parameters
  history  - show history
  obey     - run commands from a script
  logfile  - log output to a file
  sleep    - sleep for a while
  setup    - set interface IP address
  version  - display version information
  !command - pass command to OS shell
Also try 'help [general|diagnostic|ping|test]'
INET> _

The Also try line at the bottom of the Help menu lists the other modules that are installed and for which diagnostic help is available.
2.1.2 state

The state command displays the current state information, interfaces, and default settings.

Syntax

state

Example

INET> state
Station IP address for iface 0: 192.9.200.1
Station IP address for iface 1: 127.0.0.1
Active remote host 127.0.0.1
Community string "public"
Object Id: 1.3.6.1.2.1.1.3.0
retry/ping delay time: 1008 ms.
session open to 127.0.0.1, ports: local-1205, remote-161(snmp)
session->timeout: 6 ticks. (tick is 1/18th sec)
No MIBs loaded from numbers files
INET> _

2.1.3 quit

The quit command quits from the station program.

Syntax

quit

Usage

The quit command causes the application to do a clean exit.
2.1.4 history

The history command displays a list of the last commands to be entered, together with the total number of commands in the list.

Syntax

history

Example

INET> history
Command history:
  help
  history
2 entries.
INET> _

Usage

In the Menus demonstration program, the maximum number of commands that can be displayed is defined by MENU_HISTORY in ipport.h. If MENU_HISTORY is undefined, the history command is no longer listed in the help and is not available for use.
2.1.5 obey

The obey command executes console commands from a file.

**Syntax**

`obey filename`

where:

*filename* is the (path and) filename of the obey file.

**Example**

In this example, the file `test.oby` contains the text:

```
help ping
```

```
INET> obey c:\test.oby
OBEY> help ping
  ping commands:
    ping     - Ping [host] [#times]
    delay    - set milliseconds to wait between pings
    host     - set default active IP host
    length   - set default ping packet length
    endping  - terminate the current ping session
    pstats   - display statistics about ping
```

**Usage**

In the demonstration program, the obey file is a text file containing commands separated by carriage returns. Each command executed from the obey file is displayed at an `OBEY` prompt.
2.1.6 logfile

The logfile command switches on and off the copying of the console output to a log file.

Syntax

logfile filename

where:

filename Is the (path and) filename of the log file.

If you do not enter a filename, logging is switched off.

Example

INET> logfile c:\test.txt
INET> help ping
ping commands:
ping - Ping [host] [#times]
delay - set milliseconds to wait between pings
host - set default active IP host
length - set default ping packet length
endping - terminate the current ping session
pstats - display statistics about ping
INET> logfile
Closing logfile
INET> _

The logfile contains:

INET> help ping
ping commands:
ping - Ping [host] [#times]
delay - set milliseconds to wait between pings
host - set default active IP host
length - set default ping packet length
endping - terminate the current ping session
pstats - display statistics about ping
INET> logfile
Closing logfile

Usage

When logging is switched on, the specified logfile is emptied.
2.1.7 sleep

The `sleep` command instructs the command-line interface to implement a delay before processing the next command. This is particularly useful when running scripts, so that a delay is enforced between commands.

**Syntax**

```
sleep delay
```

where:

- `delay` Is the delay in seconds before the next command is processed.

**Example**

```
INET> ping 10.0.0.1
ping 0 to 10.0.0.1: Arping for host...
got ping reply; len :62 seq 0 from 10.0.0.1
INET> sleep 1
INET> ping 10.0.0.1
ping 0 to 10.0.0.1: Arping for host...
got ping reply; len :62 seq 0 from 10.0.0.1
INET> _
```

2.1.8 setip

The `setip` command sets the IP address of a specified network interface.

**Syntax**

```
setip address [interface]
```

where:

- `address` Is the dot notation IP address to be set, for example, `x.x.x.x`.
- `interface` Is the index of the network interface (1 to `n`). If this parameter is omitted, the interface defaults to 1.

**Example**

```
INET> setip 10.0.2.3
WARNING: 'setip' will kill all current net connections!!!!
replacing net[0] IP address 192.168.5.34 with 10.0.2.3
INET> _
```
2.1.9 version

The version command displays the current version of the TCP/IP stack. Version numbers do not appear in the CLI output in the demonstration programs.

Syntax

version

Example

INET> version
ARM's portable TCP/IP demo
INET> _

2.1.10 !

The ! command passes a command to the operating system shell.

Syntax

! command

where:

command Is any valid operating system command. Do not enter a space between ! and the command name.

Example

INET> !dir
Volume in drive C is MAIN DISK
  Volume Serial Number is 1E4D-17D0
  Directory of C:\
  AUTOEXEC DOS 211 11-27-98 12:56p AUTOEXEC.DOS
  COMMAND COM 93,812 08-24-96 11:11a COMMAND.COM
  CONFIG DOS 188 11-27-98 12:56p CONFIG.DOS
  AUTOEXEC BAT 301 12-04-98 12:25p AUTOEXEC.BAT
  WINDOWS <DIR> 11-23-98 4:15p WINDOWS
  INFOS <DIR> 12-02-98 7:55p INFOS
  AUTOEXEC VIA 54 11-23-98 4:21p AUTOEXEC.VIA
  PROGRA~1 <DIR> 11-23-98 4:15p Program Files
  AUTOEXEC VIA 54 11-23-98 4:21p AUTOEXEC.VIA
  SCANDISK LOG 518 01-11-99 10:34a SCANDISK.LOG
  TEMP <DIR> 12-29-98 12:26p TEMP
  CONFIG WIN 188 11-27-98 1:27p CONFIG.WIN
  WINUTILS <DIR> 11-27-98 1:26p WINUTILS
**Usage**

Any command preceded by an exclamation mark `!` is treated as an operating system command, and is executed as if entered at an operating system prompt.

In the case of the Menus demonstration program, the menu target forwards the operating system command to the host system rather than the target system. For example, where Menus is being run on the target using ADS running on Windows NT, the command is forwarded to the Windows NT command line.

---

**Note**

The command does not execute if there is insufficient memory available.
2.2 UDP Echo commands

The following are the UDP Echo commands:

- `uesend` on page 2-12
- `uesinit` on page 2-13
- `uechalt` on page 2-13
- `ueshalt` on page 2-14
- `uestats` on page 2-15.

These commands are included in the menu list if `UDPSTEST` is defined in `ipport.h`.
2.2.1 uesend

The uesend command opens a UDP echo client and sends a UDP echo packet.

This command is included in the menu list if UDPTEST is defined in ipport.h.

Syntax

uesend

Example

INET> uesend
echo socket not open. Opening....
udp echo client is starting.
sending UDP echo 0 to 10.0.0.22
INET> host 10.0.0.20
INET> uesend
host changed, restarting client socket
udp echo client is starting.
sending UDP echo 0 to 10.0.0.20
INET> UDP echo reply; len:64, reply:0, Our send#:0
INET> Deleting idle UDP Echo Client.
INET> _

Usage

The uesend command opens a UDP Echo Client and sends a UDP packet to the UDP Echo Server. The IP address of the UDP Echo Server is specified using the host command (see host on page 4-24). If a UDP Echo Client socket connection is already open for this Server, it is used.

By default, an idle UDP Echo Client is deleted after 10 minutes. This is definable in UDP_IDLE_TIMEOUT.

See also

An associated command is delay on page 4-23.
2.2.2 uesinit

The uesinit command starts the UDP Echo Server on the console, if it is not already running.

This command is included in the menu list if UDPSTEST was defined in ipport.h.

Syntax

uesinit

Example

INET> uesinit
udp echo server is starting.
INET> _

2.2.3 uechalt

The uechalt command closes the UDP Echo Client socket connection.

This command is included in the menu list if UDPSTEST was defined in ipport.h.

Syntax

uechalt

Example

INET> uechalt
udp echo - closing client socket
INET> _

Usage

A UDP Echo Client sends packets to a UDP Echo Server, and the server sends them back. This mechanism tests the functionality of UDP protocol.

Note

Multiple Client socket connections can be open on the console, with one Client socket connection through the console interface. If a TELNET Server is implemented in the console, another computer can make a TELNET connection to the console and open a Client socket connection. So, although only one UDP Echo Server can be running on the console, there can be several UDP Echo Clients.
2.2.4 ueshalt

The ueshalt command closes the UDP Echo Server (running on the console), if it is running.

This command is included in the menu list if UDPSTEST was defined in ipport.h.

**Syntax**

ueshalt

**Example**

INET> ueshalt
udp echo - closing server socket
INET> _
2.2.5 uestats

The uestats command shows the statistics for UDP Echoes completed.

This command is included in the menu list if UDPSTEST was defined in ipport.h.

Syntax

uestats

Example

INET> uestats
Showing UDP Echo statistics.
    There are no Server connections.
    There are no Client connections.
INET> uesend
echo socket not open. Opening....
udp echo client is starting.
sending UDP echo 0 to 10.0.0.20
INET> UDP echo reply; len:128, reply:0, Our send#:0
INET> uesinit
udp echo server is starting.
INET> uestats
Showing UDP Echo statistics.
    There is one Server connection.
    Total pkts for Client 1: sent=1,rcvd=1
    Total Client connections=1.
INET> _
2.3 TCP Echo commands

The following are the TCP Echo commands:

- `tesend` on page 2-17
- `tesinit` on page 2-18
- `teshalt` on page 2-18
- `techalt` on page 2-19
- `testats` on page 2-20.

These commands are included in the menu list if `TCP_ECHOTEST` was defined in `ipport.h`. 
2.3.1 tesend

The tesend command opens a TCP Echo Client for the server (if one is not already open) and sends a TCP packet to the TCP Echo Server.

This command is included in the menu list if TCP_ECHOTEST was defined in ipport.h.

Syntax

tesend

Example

INET> host 10.0.0.20
INET> tesend
All TCP Echo Client connections are in use. Please try at a later time.
INET> tesend
sending TCP echo 0 to 10.0.0.20
INET> TCP echo reply from:10.0.0.20, len:64, reply:0,Our send#:0
INET> Deleting idle TCP Echo Client.
INET> _

Usage

The IP address of the TCP Echo Server is specified using the host command (see host on page 4-24).

The maximum number of open TCP connections on which select() can be used is defined in FD_SETSIZE in tcpport.h. TCP Echo Client uses select() which has a default value of 2. Therefore, by default, only one TCP Echo Client can be open because the other connection is used for TCP Echo Server.

By default, an idle TCP Echo Client is deleted after 10 minutes. This is defined in TCP_IDLE_TIMEOUT.

See also

An associated command is delay on page 4-23.
2.3.2  tesinit

The `tesinit` command starts the TCP Echo Server on the console, if it not already running.

This command is included in the menu list if `TCP_ECHOTEST` was defined in `ipport.h`.

**Syntax**

```plaintext
tesinit
```

**Example**

```plaintext
INET> tesinit
tcp echo srv - starting.
INET> 
```

**Usage**

Because TCP Echo Server and UDP Echo Server usually start up with the application and close when the application quits, the `tesinit` command is not often used.

2.3.3  teshalt

The `teshalt` command closes the TCP Echo Server (on the console) if it is running.

This command is included in the menu list if `TCP_ECHOTEST` was defined in `ipport.h`.

**Syntax**

```plaintext
teshalt
```

**Example**

```plaintext
INET> host 10.0.0.22
INET> teshalt
tcp echo srv - closing.
INET> 
```
2.3.4 techalt

The techalt command closes the TCP Echo Client connection.

This command is included in the menu list if TCP_ECHOTEST was defined in ipport.h.

Syntax

techalt

Example

INET> techalt
Closing TCP Echo Client.
INET> _

Usage

A TCP Echo Client sends packets to a TCP Echo Server, then the server sends them back. This mechanism is used to test the functionality of TCP protocol.

Note

Multiple Client connections can be open on the console, with one Client connection through the console interface. If a Telnet server is implemented in the console, another computer can make a Telnet connection to the console, and open a Client connection.

So, although only one TCP Echo Server can be running on the console, there can be multiple TCP Echo Clients.
2.3.5 testats

The testats command shows the statistics for the TCP Echo connections.
This command is included in the menu list if TCP_ECHOTEST was defined in ipport.h.

Syntax

testats

Example

INET> testats
Showing TCP Echo statistics.
   There are no Server connections.
   There are no Client connections.
INET> tesend
sending TCP echo 0 to 10.0.0.20
INET> TCP echo reply from:10.0.0.20, len:128,
       reply:0,Our send#:0
INET> tesinit
tcp echo srv - starting.
INET> testats
Showing TCP Echo statistics.
   There is one Server connection.
   Total pkts for Client 1: sent=1,rcvd=1
   Total Client connections=1.
INET> Deleting idle TCP Echo Client.
INET> _
2.4 PPP commands

The following are the PPP commands:

- **pppup** on page 2-21
- **pppdonw** on page 2-22.

These commands are included in the menu list if **MANUAL_PPP** is defined in **ipport.h**.

2.4.1 pppup

The **pppup** command manually establishes the PPP link.

**Syntax**

```
pppup
```

**Example**

```
INET> pppup
ppp_establish returned 0 [Established]
INET> _
```

**Usage**

The command is included in the menu list if **MANUAL_PPP** is defined in **ipport.h**. If **MANUAL_PPP** is not defined, the PPP link is automatically established and dropped according to line activity.
2.4.2 pppdown

The pppdown command manually drops the PPP link.

Syntax

pppdown

Example

INET> pppdown
ppp_quit returned 0
INET> _

Usage

The command is included in the menu list if MANUAL_PPP is defined in ipport.h. When MANUAL_PPP is not defined, the PPP link is automatically established and dropped according to line activity.
2.5 SNMP commands

There is one SNMP command, listed below.

2.5.1 trap

The `trap` command sends a SNMP (version 1) trap to the trap host.

This command is included in the menu list if `INCLUDE_SNMP` was defined in `ipport.h`.

**Syntax**

```
trap
```

**Example**

```
INET> trap
trap sent
INET> _
```

2.6 NV parameters

There is one NV Parameters command, listed below.

2.6.1 nvset

The nvset command sets nonvolatile parameters. On execution, nvset saves the current configuration to webport.nv.

This command is included in the menu list if INCLUDE_NVPARAMS was defined in ipport.h.

Syntax

nvset

Example

INET> nvset
INET> _

Usage

The nvset command calls nv_writeflash, as described in the Porting TCP Programmer's Guide.
Chapter 3
Diagnostic Commands

This chapter lists the commands that are used diagnostics and debugging. It contains the following sections:

- General diagnostic commands on page 3-2
- Statistics commands on page 3-14
- DNS commands on page 3-18
- TCP commands on page 3-20
- Modem commands on page 3-26
- HTTP commands on page 3-28
- PPP commands on page 3-31
- Memory command on page 3-34
- IP commands on page 3-35
- SNMP command on page 3-38.
3.1 General diagnostic commands

The commands in this section are:

- `help diagnostic` on page 3-3
- `buffers` on page 3-4
- `queues` on page 3-5
- `dbytes` on page 3-6
- `debug` on page 3-7
- `dtrap` on page 3-9
- `dump` on page 3-9
- `linkstats` on page 3-10
- `allocsize` on page 3-11
- `upcall` on page 3-12
- `clash` on page 3-12
- `swirl` on page 3-13.

When `IN_MENUS` is defined in `ipport.h`, the CLI menu is available. The set of available commands varies according to which other options have been defined in `ipport.h`. For example, if `NET_STATS` is defined, the CLI includes the ability to display certain statistics.
3.1.1 help diagnostic

The `help` (or `?`) `diagnostic` command displays a list of the diagnostic commands that display current statistical data or perform other functions such as manipulating the routing table.

Syntax

help diagnostic

Example

INET> ? diag
SNMP Station: diagnostic commands:
  arps     - display ARP stats and table
  debug    - set IP stack debug tracing
  dtrap    - try to hook debugger
  dump     - hexdump incoming packets
  iface    - display net interface stats
  linkstats - display link layer specific stats
  memory   - list currently allocated memory
  trapsize - set size for alloc() trap
  udp      - display UDP layer stats
  snmpstats - display SNMP MIB counters
  upcall   - trace received packets
INET> _
3.1.2 buffers

The `buffers` command displays statistics for the allocated packet buffers.

**Syntax**

buffers

**Example**

INET> buffers
PACKET    len  buffer    que data offset 0
000566E0,11000,00056724,big:FF FF FF FF FF 00 20 AF CA AE ............
00059228,11000,0005926C,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
0005BD70,11000,0005BD84,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
0005E888,11000,0005E8FC,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
00061400,11000,00061444,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
00063F48,11000,00063F8C,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
00066A90,11000,00066AD4,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
000695D8,11000,0006961C,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
0006C120,11000,0006C164,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
0006EC68,11000,0006ECAC,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
000717B0,11000,000717F4,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
000742F8,11000,0007433C,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
00076E40,11000,00076E84,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
00079988,11000,000799CC,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
0007C4D0,11000,0007C514,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
0007F018,11000,0007F05C,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
00081B60,11000,00081BA4,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
000846A8,11000,000846EC,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
000871F0,11000,00087234,big:00 00 00 00 00 00 00 00 00 00 00 00 ............
....press any key for more (ESC to break)....
### 3.1.3 queues

The queues command dumps packet buffer queues.

#### Syntax

```bash
queues
```

#### Example

```
INET> queues
bigfreq: head:00059228, tail:000566E0, len:50, min:49, max:50
lilfreq: head:000DE4B0, tail:000DE3E0, len:50, min:47, max:50
rcvdq: head:00000000, tail:00000000, len:0, min:0, max:1
INET>
```

The first two lines provide tally information about the big and little packet buffer free queues:

- **head** is a pointer to the start of the queue.
- **tail** is a pointer to the end of the queue.
- **len** gives a snapshot of the number of packet buffers of each type in the queues.
- **min** displays how low **len** has dropped since boot time. This gives you some indication of whether you are running out of packet buffers. When **min** is 0, it means that there were no packet buffers in the listed queue type at least once since you booted the stack.
- **max** displays how many packets were allocated on this queue.

The **rcvdq** line displays information on the packet receive queue:

- **head** is a pointer to the start of the queue.
- **tail** is a pointer to the end of the queue.
- **len** displays how many packets are in the receive queue that have not yet been processed by the IP layer.
- **min** is always zero for **rcvd**, as it starts empty.
- **max** displays how high **len** has risen since boot time. A high value indicates that the stack is not processing the receive queue in a timely manner.
3.1.4 dbytes

The dbytes command dumps a block of memory for use in debugging.

**Syntax**

\[
dbytes\ text\ memory\_location,\ length
\]

where:

\[
\text{memory\_location} \\
\text{length}
\]

- **memory_location**
  Gives the location of the memory block.
- **length**
  Gives the length of the memory block (optional).

**Example**

INET> dbytes 0x8000, 0x40
90 B5 07 1C 2D F0 86 F8 04 06 24 0E 08 48 09 49 .........$.H.I
09 68 44 54 07 48 00 68 80 00 05 49 0C 31 0F 50 .hDT.H.h...I.1.P
04 48 00 68 01 30 03 49 08 60 90 BC 08 BC 18 47 .H.h.0.I.`.....G
B8 64 04 00 78 43 04 00 80 85 07 1C 0E 48 00 68 .d..xC.......H.h
INET>
3.1.5 debug

The debug command starts or stops IP stack debug tracing.

Syntax

d Debug bitmask

where:

bitmask (optional) Is a number that represents a bit mask where each bit specifies whether a particular type of IP stack tracing occurs. The mapping of these bits is given in Table 3-1 on page 3-8.

If you enter the debug command without a parameter, it turns off debug tracing.

Example

INET> debug 7
NDEBUG is now 0x07
INET> debug
IP stack debug tracing off
INET>

Usage

This command sets an internal flag which results in the application printing out status messages as packets are received, or sent up and down the protocol stack. This can be helpful for finding exactly where a bad packet is detected in a protocol stack layer. Often, the nature of the error is reported, for example:

bad cksum

See also

Refer also to the upcall command. Debug tracing will not occur on packets being processed in ISR context unless both these options are enabled.

Table 3-1 gives the definitions of the bits in the bit mask.
### Table 3-1 Bit mask definitions

<table>
<thead>
<tr>
<th>Trace type</th>
<th>Bit mask (Hex)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUGHALT</td>
<td>0x01</td>
<td>Halts on a gross applications level error that is detected in the network code</td>
</tr>
<tr>
<td>DUMP</td>
<td>0x02</td>
<td>Works in conjunction with other options:</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>BUGHALT Dumps all arriving packets.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>PROTERR Dumps header for level of error.</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>NETTRACE (and other trace options). Dumps headers at trace level.</td>
</tr>
<tr>
<td>INFOMSG</td>
<td>0x04</td>
<td>Print general informational messages</td>
</tr>
<tr>
<td>NETERR</td>
<td>0x08</td>
<td>Display net interface error messages</td>
</tr>
<tr>
<td>PROTERR</td>
<td>0x10</td>
<td>Display protocol level error messages</td>
</tr>
<tr>
<td>NETTRACE</td>
<td>0x20</td>
<td>Trace packet in link level net layer</td>
</tr>
<tr>
<td>TMO</td>
<td>0x40</td>
<td>Print message on timeout</td>
</tr>
<tr>
<td>APTRACE</td>
<td>0x80</td>
<td>Trace packet through application</td>
</tr>
<tr>
<td>TPTRACE</td>
<td>0x100</td>
<td>Transport protocol (UDP/TCP/RVD) trace</td>
</tr>
<tr>
<td>IPTRACE</td>
<td>0x200</td>
<td>Trace packet in internet layer</td>
</tr>
<tr>
<td>UPCTRACE</td>
<td>0x400</td>
<td>Trace upcall progress</td>
</tr>
<tr>
<td>MEMTRACE</td>
<td>0x8000</td>
<td>Trace memory allocs and frees - requires MONITOR_ALLOCS to be defined in ipport.h</td>
</tr>
<tr>
<td>DHCPTTRACE</td>
<td>0x1000</td>
<td>Trace DHCP client options</td>
</tr>
<tr>
<td>ARPTRACE</td>
<td>0x2000</td>
<td>Trace ARP transactions</td>
</tr>
<tr>
<td>TFTTPTRACE</td>
<td>0x4000</td>
<td>Trace TFTP operation</td>
</tr>
<tr>
<td>CLANGTRACE</td>
<td>0x8000</td>
<td>Trace CLANGs (packets dropped deliberately, for testing)</td>
</tr>
</tbody>
</table>
3.1.6  **dtrap**

The `dtrap` command causes a call to the `dtrap()` function. Typically, this stops the execution of the program in the debugger, as a breakpoint will have been set on `dtrap()`.

**Syntax**

dtrap

**Usage**

As supplied, `dtrap()` is implemented as a function that does nothing. To stop execution on `dtrap()`, you must set a breakpoint on it in your debugger.

The method of resuming execution varies from debugger to debugger.

3.1.7  **dump**

The `dump` command dumps the first 48 bytes, starting at the IP number.

**Syntax**

dump

**Example**

INET> dump
Packet hex dumping enabled
INET> ping 10.0.0.1
ping 0 to 10.0.0.1: ping 0 sent...
Demux Packet 0000000DE720 length 64
  45 00 00 30 63 00 00 FF 01 42 8E 0A 00 00 01 E..0c<....B....
  0A 00 02 02 00 00 51 A0 00 00 00 00 50 69 6E 67 ......Q.....Ping
  20 66 72 6F 6D 20 4E 65 74 50 6F 72 74 20 49 50  from NetPort IP
got ping reply; len: 62 seq: 0 from 10.0.0.1
INET>
3.1.8  linkstats

The linkstats command displays statistics for the link layer. These are the statistics for the hardware associated with the given interface. The format, content, and accuracy of these statistics varies from link driver to link driver.

Syntax

linkstats  interface_number

where:

interface_number

Is the interface that displays the statements. If no interface is specified, 0 is used.

Example

INET> linkstat 0
PPP unit: 0, iface: 0, mtu:1500  mru:1500  timer:0
packets: In: 0  Out: 0
bytes : In: 0  Out: 0
errors : In: 0  Out: 0
FSM states; LCP:0, IPCP:0
LCP options:
ppptimers: created:0, deleted:0, fired:0
fastq: head:00000000, tail:00000000, len:0, min:0, max:0
IPque: head:00000000, tail:00000000, len:0, min:0, max:0
inq: head:00000000, tail:00000000, len:0, min:0, max:0
outq: head:00000000, tail:00000000, len:0, min:0, max:0
logging: file off, console off
VJC: compressed pkts in:0, out:0; missed:0, in-errors:0
INET>

Usage

This command differs from the iface command in that these counters are generally read from the hardware drivers.

In general, packet counts should be obtained from the iface command, because its counters are well defined (by MIB-2) and are uniform across all devices.

See also

An associated command is iface on page 3-33.
3.1.9 allocsize

The allocsize command sets the number of bytes for alloc() breakpoint.

Syntax

allocsize num_bytes

where:

dnum_bytes Is the number of bytes to be allocated.

Example

INET> allocsize 128
malloc trap size set to 128
INET> _

Usage

The stack code expects the porting engineer to provide an implementation of a function called npalloc() that the stack uses for memory allocation. The function npalloc() take a single parameter which specifies the number of bytes to be allocated by the call.

The supplied code provides an implementation of npalloc() where there is an option to have the code break into a debugger if the value of the passed parameter equals a particular value. The value that causes the code to trap is specified by the allocsize command. This can be useful during debugging.
3.1.10 upcall

The upcall command traces received packets.

Syntax
upcall

Example
INET> upcall
Upcall debugging enabled
INET> tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23
tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23
tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23
tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23
tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23
tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23
tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23
tcp_rcv: TCP packet from 10.0.0.1:1082 to 10.0.2.2:23

INET> _

Usage
This command enables protocol stack trace reporting on incoming packets. The upcall command toggles the UPCTRACE bit in the tracing bit mask that is affected by the debug command. It is a quick way of toggling one particular bit.

See also
An associated command is debug on page 3-7.

3.1.11 clash

The clash command checks the menu structures for consistency and reports if any command is a substring of another. This is useful for testing menus where many new items have been added during porting.

Syntax
clash
3.1.12 swirl

The `swirl` command produces a pattern on the console. It is useful while testing serial drivers or the telnet server code.

**Syntax**

```
swirl num_lines
```

where:

- `num_lines` Is the number of lines of output to produce.

**Example**

INET> swirl 15

```
1: !
2: "# 
3: #$% 
4: $%&'
5: %&'()
6: &'()++ 
7: '('++,- 
8: ()++,-/ 
9: )++,-/01 
10: ++,-/0123 
11: +,-/012345 
12: +,-/01234567 
13: +,-/0123456789 
14: /0123456789; 
15: /0123456789;<=
```

INET>
3.2 Statistics commands

The commands in this section are:

- **arps** on page 3-14
- **ipstat** on page 3-15
- **icmpstat** on page 3-16
- **udp** on page 3-16
- **dcstats** on page 3-17.

3.2.1 arps

The **arps** command displays some ARP statistics and dumps data from the entries in the current ARP table.

This command is included in the menu list if `NET_STATS` was defined in `ipport.h`.

**Syntax**

arps

**Example**

INET> arps
arp Requests In: 1, out: 1
arp Replys In: 1, out: 1
X)  MAC Address  iface  pend  IP  ctime  ltime
0) 009027-DEBC72  1  N  10.0.0.1  817  3602
INET>

where:

- **X)** Shows the position of this entry in the ARP table.
- **MAC address** Gives the MAC address.
- **iface** Is the logical interface that the arp was resolved on.
- **pend** Indicates whether there is an outgoing packet awaiting the results of an arp reply. The value is usually N (no).
- **IP** Is the IP address associated with the MAC address.
- **ctime/ltime** Is the internal timestamp when the ARP reply was last referenced.
3.2.2 ipstat

The `ipstat` command displays the standard IP SNMP MIB statistics.

This command is included in the menu list if `NET_STATS` was defined in `ipport.h`.

**Syntax**

`ipstat`

**Example**

```
INET> ipstat
IP MIB statistics:
Gateway: NO   default TTL: 30
rcv: total: 39002   header err: 0   address err: 0
rcv: unknown Protocols: 0  delivered: 39002
send: total: 2360  discarded: 0  No routes: 0
Routing; forwarded: 0  discarded: 0
Recvd fragments: 0, Frames reassembled: 0
Pkts fragmented: 0, Fragments sent: 0, dropped: 0
Reasm.Timeout: 0, Reasm.Errors: 0
INET> _
```
3.2.3  icmpstat

The `icmpstat` command displays the standard ICMP SNMP MIB statistics.

This command is included in the menu list if `NET_STATS` was defined in `ipport.h`.

**Syntax**

`icmpstat`

**Example**

INET> `icmpstat`

**In counters:**
- `DestUnreach`: 0
- `TimeExceed`: 0
- `ParmProb`: 0
- `SrcQuench`: 0
- `Redirect`: 0
- `Echo(ping)`: 0
- `EchoReps`: 1
- `Timestmp`: 0
- `TStmpRep`: 0
- `AddrMasks`: 0
- `AddrMaskRep`: 0

**Out counts:**
- `DestUnreach`: 2
- `TimeExceed`: 0
- `ParmProb`: 0
- `SrcQuench`: 0
- `Redirect`: 0
- `Echo(ping)`: 0
- `EchoReps`: 0
- `Timestmp`: 0
- `TStmpRep`: 0
- `AddrMasks`: 0
- `AddrMaskRep`: 0

INET> _

3.2.4  udp

The `udp` command displays the standard UDP SNMP MIB statistics.

This command is included in the menu list if `NET_STATS` was defined in `ipport.h`.

**Parameters**

None

**Example**

INET> `udp`

**UDP MIB dump:**
- **In:**
  - Good: 26
  - No Port: 2
  - Bad: 0
- **Out:** 26

INET> _
3.2.5 dcstats

The dcstats command displays statistics about the DHCP client.

This command is included in the menu list if DHCP_CLIENT and NET_STATS were defined in ipport.h.

Syntax

dhcpstat

Example

INET> dcstats
dhcp client stats:
    all errors:  0
    discover sent:  1
    offers rcvd:  1
    requests sent:  1
    acks received: 13
    bootp replys:  0
    declines sent:  0
    releases sent:  0
    naks received:  0
    renew req sent: 14
    rebind req sent: 0
    Interface 1 state = bound
    Interface 2 state = unused
    Interface 3 state = unused
INET>
3.3 DNS commands

The commands in this section are:

- `dnsstats` on page 3-18
- `nslookup` on page 3-19.

3.3.1 dnsstats

The `dnsstats` command shows statistics about the DNS Client, and is included in the menu list if `DNS_CLIENT` and `NET_STATS` were defined in `ipport.h`.

Syntax

dnsstats

Example

INET> dnsstats
DNS servers:10.0.0.1 0.0.0.0 0.0.0.0
DNS cache:
  name: fred.nowhere.com, IP: 10.16.100.31, retry:0, ID:4660, rcode:0, err:0
  name: bob.nowhere.com, IP: 172.16.100.23, retry:0, ID:4661, rcode:0, err:0
  name: jim.nowhere.com, IP: 0.0.0.0, retry:3, ID:4662, rcode:0, err:0
  name: , IP: 0.0.0.0, retry:0, ID:0, rcode:0, err:0
  name: , IP: 0.0.0.0, retry:0, ID:0, rcode:0, err:0
  name: , IP: 0.0.0.0, retry:0, ID:0, rcode:0, err:0
protocol/implementation runtime errors:4
requests sent:3
replies received:6
usable replies:2
total retries:3
timeouts:0
INET>

Usage

The statistics shown are:

- a list of DNS servers that the client is configured to recognize
- a listing of the client-side DNS cache (a history of DNS resolutions that the client has performed)
- general DNS statistics.
3.3.2 nslookup

The nslookup command performs a query against the known DNS servers for the specified host name.

This command is included in the menu list if DNS_CLIENT was defined in ipport.h.

Syntax

nslookup hostname

where:

hostname Is looked for in the DNS database, and its IP addresses (if any) returned.

Example

INET> nslookup pc4.nowhere.com
gethostbyname() succeeded
h_name is pc4.nowhere.com
h_addrtype = 2
h_length = 4
h_addr_list are
10.16.100.31
INET>
3.4 TCP commands

This section lists the TCP diagnostic commands:

- mbuf on page 3-20
- mlist on page 3-21
- tcp on page 3-22
- sockets on page 3-22
- tbconn on page 3-23
- tbsend on page 3-24
- tbrcv on page 3-25.

These commands are included in the menu list if INCLUDE_TCP and NET_STATS are defined in ipport.h.

3.4.1 mbuf

The mbuf command displays information about the queues that are used to hold freed and in-use mbufs.

This command is included in the menu list if INCLUDE_TCP and NET_STATS were defined in ipport.h.

The BSD implementation of TCP uses mbuf for dynamic memory requirements.

Syntax

mbuf

Example

INET> mbuf
mfreeq: head:000E1D80, tail:000E1EA0, len:203, min:196, max:203
mbufq: head:00000000, tail:00000000, len:0, min:0, max:7
mbuf allocs: 2349, frees: 2349
m_copy copies: 0, copied bytes: 0
m_copy clones: 458, cloned bytes: 43717
ip_output appends: 34, prepends: 47, copies: 721
INET>

See also

The command mlist on page 3-21 displays detailed information about each mbuf that is in use.
3.4.2 mlist

The mlist command displays information about the mbufs that are in use.

This command is included in the menu list if INCLUDE_TCP and NET_STATS were defined in ipport.h.

Syntax

mlist

Example

INET> mlist
mbufs in use:
type 1, pkt:000D08C0, data:000DDC3C, len:3
type 1, pkt:000D07C0, data:000DFBEC, len:12
type 1, pkt:000D350, data:000DF3CC, len:1
...
3.4.3 tcp

The tcp command displays the standard TBP SNMP MIB statistics.

This command is included in the menu list if INCLUDE_TCP and NET_STATS were defined in ipport.h.

Syntax

tcp

Example

INET> tcp
tcpRtoAlgorithm 0, tcpRtoMin 0
tcpRtoMax 0, tcpMaxConn 0
tcpActiveOpens 11, tcpPassiveOpens 6
tcpAttemptFails 0, tcpEstabResets 3
tcpCurrEstab 0, tcpInSegs 2344
tcpOutSegs 2567, tcpRetransSegs 16
tcpInErrs 0, tcpOutRsts 0
INET> _

3.4.4 sockets

The sockets command displays the socket list.

This command is included in the menu list if INCLUDE_TCP and NET_STATS were defined in ipport.h.

Syntax

sockets

Example

INET> sockets
TCP sock, fhost, ports, opts, rxbytes, txbytes, snd_una, snd_nxt, state:
000E34D0, 10.0.0.1, 23->1084, 0x0100, 0, 0, 4424857, 4424857, ESTABLISHED
000E335C, 0.0.0.0, 23->0, 0x0102, 0, 0, 0, 0, LISTEN
000E3130, 0.0.0.0, 7->0, 0x0102, 0, 0, 0, 0, LISTEN
000E2EF0, 0.0.0.0, 21->0, 0x0102, 0, 0, 0, 0, LISTEN
000E2D0A0, 0.0.0.0, 80->0, 0x0102, 0, 0, 0, 0, LISTEN
INET>
3.4.5  tbconn

The tbconn command displays the TCP BSD connection statistics.

This command is included in the menu list if INCLUDE_TCP and NET_STATS were defined in ipport.h.

Syntax

tbconn

Example

INET> tbconn
connections initiated: 11,  connections accepted: 6
connections established: 12,  connections dropped: 3
embryonic connections dropped: 5, conn. closed(includes drops):22
segs where we tried to get rtt: 1382,  times we succeeded: 1369
delayed acks sent: 0,  conn. dropped in rxmt timeout: 0
retransmit timeouts: 16,  persist timeouts: 0
keepalive timeouts: 101,  keepalive probes sent: 0
connections dropped in keepalive: 5
INET> _
The `tbsend` command shows statistics about TCP packets sent.

This command is included in the menu list if `INCLUDE_TCP` and `NET_STATS` were defined in `ipport.h`.

**Syntax**

tbsend

**Example**

```
INET> tbsend
total packets sent: 2612,     data packets sent: 1366
data bytes sent: 68491,      data packets retransmitted: 1
data bytes retransmitted: 3,  ack-only packets sent: 1211
window probes sent: 0,        packets sent with URG only: 0
window update-only packets sent:0,   control (SYN|FIN|RST) packets sent:34
INET> _
```
3.4.7  tbrcv

The tbrcv command shows statistics about TCP packets received.

This command is included in the menu list if INCLUDE_TCP and NET_STATS were defined in ipport.h.

**Syntax**

tbrcv

**Example**

INET> tbrcv
total packets received: 2401, packets received in sequence: 1149
bytes received in sequence: 2435, packets received with ccksum errs: 0
packets received with bad offset: 0, packets received too short: 0
duplicate-only packets received: 62, duplicate-only bytes received: 62
packets with some duplicate data: 0, dup. bytes in part-dup. packets: 0
out-of-order packets received: 8, out-of-order bytes received: 0
packets with data after window: 0, bytes rcvd after window: 0
packets rcvd after close: 0, rcvd duplicate acks: 9,
rcvd duplicate acks: 1384, rcvd window probe packets: 0
rcvd window update packets: 0, rcvd acks for unsent data: 0
bytes acked by rcvd acks: 68811
INET> _
3.5 Modem commands

This section lists the modem diagnostic commands:

- hangup on page 3-26
- modem on page 3-27.

3.5.1 hangup

The hangup command hangs up (and resets) the modem. This also shuts down all protocols (for example, PPP) running over that interface.

This command is included in the menu list if USE_MODEM was defined in ipport.h.

Syntax

hangup
3.5.2 modem

The `modem` command displays various statistics related to modem usage, for example, dialer and UART information.

This command is included in the menu list if `USE_MODEM` and `NET_STATS` were defined in `ipport.h`.

**Syntax**

```
modem
```

**Example**

```
INET> modem
unit 0, dialer state: AUTOANS
last baud rate: 33600
Not implemented for this UART

Stats for uart 1
number of putchar calls : 984
tx interrupts : 0
tx drops (buffer full) : 0
tx char (to uart) : 984
number of getchar calls : 1652043
rx interrupts : 63
rx timeout interrupts : 32
rx char (from uart) : 1554
rx drops : 0
rx overrun events : 0
rx break events : 0
rx parity errors : 0
rx framing errors : 0
modem interrupts : 2
tx fc stop (CTS lowered) : 0
tx fc go (CTS raised) : 0
rx fc stop (lowered RTS) : 0
rx fc go (raised RTS) : 0
INET>
```
3.6 HTTP commands

This section lists the HTTP diagnostic commands:

- `hstat` on page 3-28
- `dir` on page 3-29.

3.6.1 hstat

The `hstat` command displays HTTP statistics.

This command is included in the menu list if WEBPORT was defined in ipport.h.

**Syntax**

`hstat`

**Example**

```
INET> hstat
HTTP stats: requests:119 gets:118 errors:0, ssi:117 cgi:0
No http connections currently open.
INET> _
```
3.6.2  dir

The `dir` command lists the directory of VFS files. This command is included in the menu list if `WEBPORT` was defined in `ipport.h`.

Syntax

dir

Example

INET> dir
btmmap.map   --M--------- 00048a94        0        0 0
setip.htm    HB---------- 000471f4      528       214 0
hlpmask.htm  H---------- 00047408      2b6       159 0
helphlp.htm  H---------- 00047561       24c       130 0
hlpipadd.htm H---------- 00047691       5cc       38e 0
ptstats.htm  H---------- 00047a1f       268       ca 0
ptstat.htm   H---------- 00047ae9       a2        5b 0
pdetail.htm  H---------- 00047b44       6ec       285 0
stopmen.htm  H---------- 00047dc9       4cb       1b5 0
tcpstats.htm H---------- 00047f7e       8a         32 0
index.htm    H---------- 00047fb0       5ca       23e 0
body.htm     H---------- 000481ee       41        7 0
inworks.htm  H---------- 000481f5       a8        4b 0
setport.htm  H---------- 00048240      6a3       1d3 0
helpbtn.gif  H---------- 00048d74       4ad       4ad 0
nplogot.gif  H---------- 00049221       6f5       6f5 0
hub4907.gif  H---------- 00049916      1ad0       1ad0 0
btmmap.gif   H---------- 0004b3e6       f67       f67 0
ipaddr       ----------s-- 00000000        0        0 0
ipmask       ----------s-- 00000000        0        0 0
defgw        ----------s-- 00000000        0        0 0
hpport       ----------s-- 00000000        0        0 0
frmsent      ----------s-- 00000000        0        0 0
frmcvd       ----------s-- 00000000        0        0 0
bytercvd     ----------s-- 00000000        0        0 0
ptcolls      ----------s-- 00000000        0        0 0
pterrors     ----------s-- 00000000        0        0 0
portstat     ----------s-- 00000000        0        0 0
netstats     ----------s-- 00000000        0        0 0
setip.cgi    H---------- 00000000        0        0 0
setport.cgi  H---------- 00000000        0        0 0
total files = 32
INET>
In the example, the columns have the following meanings:

**Column 1: name**  Is the name of the file in the VFS.

**Column 2: flags**  Is a combination of:

- **H**  *name* is a compressed HTML file.
- **B**  Basic authentication is required to access *name*.
- **S**  MD5 authentication is required to access *name*.
- **M**  *name* is an image map.
- **V**  *name* is a program variable.
- **W**  *name* is writable.
- **I**  File structure for *name* is dynamically allocated.
- **D**  Data for *name* is dynamically allocated.
- **N**  *name* is in nonvolatile storage.
- **S**  *name* is stale (for example, needs syncing).
- **s**  *name* is a Server-Side Include (SSI) function.
- **c**  *name* is a Common Gateway Interface (CGI) function.
- **m**  (method) *name* is handled by an external file system, not VFS.

**Column 3:**  Memory address of the file data (for VFS files only).

**Column 4:**  Length of the file when uncompressed.

**Column 5:**  Length of the file when compressed.

**Column 6:**  Length of the file buffer.
3.7 PPP commands

This section lists the PPP diagnostic commands:

- `pcons` on page 3-31
- `pfile` on page 3-32
- `chap` on page 3-32
- `iface` on page 3-33.

3.7.1 pcons

The `pcons` command turns PPP trace information on or off. If it is on, the information is logged to the target system console.

This command is included in the menu list if `USE_PPP` was defined in `ipport.h`.

**Syntax**

`pcons`

**Example**

```
INET> pcons
ppp console logging turned ON
INET> modem_gets:
pppu
dialing 2818...
modem_cmd: ATDT2818
modem_gets: ATDT2818
ppp_establish returned 1 [Pending]
INET> modem_gets:
CONNECT 31200/ARQ/V34/LAPM/V42BIS

Connected to 2818
pppstart; unit 0;pppstart: ok
pppinput: got PPP_FLAG pppinput: got PPP_FLAG ppp_infrm: unit 0 Got a Packet lcp packet =pppstart; unit 0;pppstart: ok
pppinput: got PPP_FLAG pppinput: got PPP_FLAG ppp_infrm: unit 0 Got a Packet lcp packet =ChapAuthWithPeer: unit:0, our_name:ppp, digest(type):128
pppinput: got PPP_FLAG pppinput: got PPP_FLAG ppp_infrm: unit 0 Got a Packet Got a CHAP packet
INET> _
```
3.7.2 pfile

The pfile command turns PPP trace information on or off. If it is on, the information is
sent to a log file called ppp.log.

This command is included in the menu list if USE_PPP was defined in ipport.h.

Syntax

pfile

Example

INET> pfile
ppp file logging to ppp.log ON
INET> pfile
ppp file logging turned OFF
INET> _

3.7.3 chap

The chap command displays statistics for the Challenge Handshake Authentication
Protocol (CHAP).

This command is included in the menu list if CHAP_SUPPORT, NET_STATS and USE_PPP were
defined in ipport.h (and/or ppp_port.h).

Syntax

chap

Example

INET> chap
Chap stats for unit 0, iface 0.
  client state: OPEN
  server state: OPEN
  challenge xmits: 0
  challenge replys: 1
INET>
3.7.4 iface

The iface command displays statistics for the given interface.

This command is included in the menu list if NET_STATS was defined in ipport.h.

Syntax

iface iface_number

where:

iface_number  (optional) Selects the interface. If no interface is specified, the default is 0.

Example

INET> iface
Interface number 0, type: PPP link
IP address: 10.1.1.3, subnet mask: 255.0.0.0, gateway: 10.0.0.1
Has been up for: 0 minutes, 26 sec.
rcvd: errors:0   dropped:0   station:17   bcast:0   bytes:1020
sent: errors:0   dropped:0   station:23   bcast:0   bytes:1380
MAC address: FF 00 00 00 21 C0 ....!

INET> _

Usage

The numbering of the interfaces does not correspond to the interface indexing described in MIB-2. This indexing scheme starts numbering at one (that is, three interfaces are numbered 1, 2, 3), whereas the ARM protocol stack internal mechanisms maintain the interface indexes numbered from 0 (that is, 0, 1, 2). Programmers working on the ARM source code must keep this in mind.

See also

An associated command is linkstats on page 3-10.
3.8 Memory command

There is one memory diagnostic command, listed below.

3.8.1 memory

The `memory` command lists currently allocated memory.

This command is included in the menu list if `MEM_BLOCKS` was defined in `ipport.h`.

Syntax

```
memory
```

Example

```
INET> memory
0: 4772 @ 0x0005E54C
1: 52 @ 0x0005F7FC
2: 11005 @ 0x0005F83C
3: 52 @ 0x00062344
4: 11005 @ 0x00062384
5: 52 @ 0x00064E8C
6: 11005 @ 0x00064ECC
...
...
...
INET>
```

Usage

This command dumps the address and size of all allocated but un-freed memory since the application started. This is useful in detecting memory leaks when no other tools are available.
3.9 IP commands

This section lists the IP diagnostic commands:

- `routes` on page 3-35
- `rtadd` on page 3-36
- `rtdel` on page 3-37.

3.9.1 routes

The `routes` command displays the IP route table.

This command is included in the menu list if IP_ROUTING was defined in `ipport.h`.

**Syntax**

```
routes
```

**Example**

```
INET> routes
..IPaddr........mask............nexthop...iface..type
10.2.2.0   255.255.255.0   10.0.0.2   0   LOCAL
0.0.0.0   0.0.0.0   10.0.0.1   0   LOCAL
          empty slot
          empty slot
          empty slot
          empty slot
          empty slot
          empty slot
          empty slot
          empty slot
          empty slot
          empty slot

 cached address is: 10.0.0.1 next hop: 10.0.0.1 iface: 0
INET>
```
3.9.2  rtadd

The rtadd command manually adds an IP route to the routing table.

This command is included in the menu list if IP_ROUTING was defined in ipport.h.

Syntax

rtadd target_ip target_mask next_hop iface_number

where:

target_ip Is the target IP address in dot notation form, for example 10.0.2.0.
target_mask Is the target network mask in dot notation form, for example 255.255.255.0.
next_hop Is the router IP address in dot notation form, for example 10.0.0.20.
iface_number Is the interface number.

Example

INET> rtadd
usage: target.ip target.mask next.hop iface
   where 1st 3 params are in IP dot notation, last is digit 0-2
INET> rtadd 10.0.2.0 255.255.255.0 10.0.0.25 1
INET> _
3.9.3  rtdel

The `rtdel` command manually deletes an IP route from the routing table.

This command is included in the menu list if IP_ROUTING was defined in `ipport.h`.

**Syntax**

```
rtdel target_ip target_mask iface_number
```

where:

- **target_ip**: Must match the target IP address of the route(s) you want to delete.
- **target_mask**: Must match the netmask of the route(s) you want to delete.
- **iface_number**: Must match the interface number of the route(s) you want to delete, or be -1 (a wildcard).

**Example**

```
INET> rtdel 10.0.2.2 255.255.255.0 -1
Deleted 1 route(s)
INET>
```
3.10 SNMP command

This section lists the SNMP diagnostic commands:
- `snmpstat` on page 3-38
- `snmpinfo` on page 3-39.

3.10.1 `snmpstat`

The `snmpstat` command displays the counters associated with the SNMP protocol layer. These counters are those described for SNMP in MIB-2 (RFC1213).

This command is included in the menu list if `INCLUDE_SNMP` was defined in `ipport.h`.

Syntax

`snmpstat`

Example

INET> `snmpstats`

```
snmpInPkts: 0          snmpOutPkts: 0
snmpInBadVersions: 0          snmpInBadCommunityNames: 0
snmpInBadCommunityUses: 0          snmpInASNParseErrs: 0
snmpInTooBigs: 0          snmpInNoSuchNames: 0
snmpInBadValues: 0          snmpInReadOnlys: 0
snmpInGenErrs: 0          snmpInTotalReqVars: 0
snmpInTotalSetVars: 0        snmpInGetRequests: 0
snmpInGetNexts: 0          snmpInSetRequests: 0
snmpInGetResponses: 0        snmpInTraps: 0
snmpOutTooBigs: 0          snmpOutNoSuchNames: 0
snmpOutBadValues: 0          snmpOutGenErrs: 0
snmpOutGetRequests: 0        snmpOutGetNexts: 0
snmpOutSetRequests: 0        snmpOutGetResponses: 0
snmpOutTraps: 0          snmpEnableAuthenTraps: 2
```

INET> _
### 3.10.2 snmpinfo

The `snmpinfo` command displays SNMP agent information.

This command is included in the menu list if `INCLUDE_SNMP` was defined in `ipport.h`.

#### Syntax

```plaintext
snmpinfo
```

#### Example

```
INET> snmpinfo
2 SNMP communities:
   0) public READ-ONLY
   1) private READ-WRITE
system.sysDescr.0: ARM SNMP Evaluation System
system.sysObjectID.0: .1.3.6.1.4.1.4128.1.1
system.sysUpTime.0: 62526
system.sysContact.0: John Smith
system.sysName.0: Test-34
system.sysLocation.0: 123 Acacia Avenue
INET>
```
Chapter 4
Protocol-specific Commands

This chapter lists the commands for the following protocols. It contains the following sections:

- **DHCP server commands** on page 4-2
- **Email Alerter commands** on page 4-8
- **FTP client commands** on page 4-15
- **Ping commands** on page 4-21
- **NAT Router commands** on page 4-28
- **Routing Information Protocol (RIP) commands** on page 4-34
- **TELNET commands** on page 4-39
4.1 DHCP server commands

The commands in this section are:
- `help dhcpsrv` on page 4-2
- `dhsrv` on page 4-3
- `dhlist` on page 4-4
- `dhentry` on page 4-5
- `dhdelete` on page 4-6
- `dhpools` on page 4-7.

4.1.1 help dhcpsrv

The `help dhcpsrv` command displays a list of DHCP server commands.

**Syntax**

```
dhcpsrv
```

**Example**

```
INET> help dhcpsrv
dhcpsrv commands:
    dhsrv  - DHCP server statistics
    dhlist - DHCP server assigned addresses
    dhentry - list specific entry details
    dhdelete - delete a DHCP entry
    dhpools - list free address pools
INET>
```
4.1.2 dhsrv

The dhserv command displays the DHCP server statistics.

**Syntax**

dhsrv

**Example**

INET> dhsrv
plain bootp requests received: 0
plain bootp replys sent: 0
discover packets received: 1
offer packets sent: 1
dhcp request packets received: 5
declines received: 0
releases received: 0
acks sent: 5
naks sent: 0
requests for other servers: 0
protocol errors; all types: 0
INET>

**Usage**

All of the packet types in the example are described in RFC2131. BOOTP packets are kept in separate categories.
4.1.3  dhlist

The dhlist command displays the list of IP addresses assigned by the DHCP Server.

Syntax

dhlist

Example

INET> dhlist
no DHCP/BOOTP entries in database
INET> dhlist
<table>
<thead>
<tr>
<th>IP addr</th>
<th>client ID</th>
<th>type</th>
<th>status</th>
<th>lease</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.1.95</td>
<td>00:20:AF:CA:0A:AE</td>
<td>dbase</td>
<td>Unassigned</td>
<td>N/A</td>
</tr>
<tr>
<td>10.0.1.98</td>
<td>00:60:08:4D:C3:AA</td>
<td>dbase</td>
<td>Assigned via DHCP</td>
<td>60</td>
</tr>
</tbody>
</table>

2 Entries
INET>

Usage

The sample list may include clients that are defined in the database files, but that have not yet been assigned using DHCP.

The Unassigned status may indicate that:

- a client MAC address has been mistyped in the database file
- a client lease has expired
- a client machine has been switched off.
4.1.4 dhentry

The dhentry command shows the details about a particular entry in the DHCP database. For each IP address assigned by the DHCP Server, an entry is made in the database.

Syntax

dhentry index_number

where:

index_number Is the number that identifies the IP address in the database.

Example

INET> dhlist
  IP addr   client ID   type   status   lease
  1 10.0.1.95 00:20:AF:CA:0A:AE dbase Unassigned N/A
  2 10.0.1.98 00:60:08:4D:C3:AA dbase Assigned via DHCP 60

2 Entries
INET> dhentry 1
IP:10.0.1.95 - client ID:00:20:AF:CA:0A:AE - status:Unassigned
subnet:255.0.0.0 gateway:10.0.0.1 DNS:10.0.0.1
lease 367, type: dbase , name: bpc18
INET>
4.1.5  dhdelete

The dhdelete command deletes an entry from the DHCP database.

**Syntax**

dhdelete index_number

where:

index_number  Is the number that identifies the IP address in the database.

**Example**

INET> dhlist
IP addr  client ID        type      status    lease
1 10.0.1.95 00:20:AF:CA:0A:AE dbase    Unassigned N/A
2 10.0.1.98 00:60:08:4D:C3:AA dbase    Assigned via DHCP 60
2 Entries
INET> dhdelete 1
deleted
INET> dhlist
IP addr  client ID        type      status    lease
1 10.0.1.98 00:60:08:4D:C3:AA dbase    Assigned via DHCP 60
1 Entries
INET>

**Usage**

Use dhlist to get information about a DHCP entry and then use dhdelete to delete that entry from the DHCP database.

--- Note ---

This command only affects the DHCP database. The DHCP Server cannot stop the remote device/computer using the IP address that would be freed by this call.

---
4.1.6 dhpools

The dhpools command displays a list of pools of free IP addresses. The DHCP server can dynamically assign an address taken from these pools to a client.

**Syntax**

dhpools

**Example**

INET> dhpools
Free DHCP address pools:
  range: 10.0.3.1 - 10.0.3.99, iface: 0
  range: 10.0.3.101 - 10.0.3.199, iface: 0

INET>
4.2 Email Alerter commands

The commands in this section are:

- `help smtp` on page 4-8
- `mdel` on page 4-9
- `mport` on page 4-10
- `mrcpt` on page 4-11
- `msserver` on page 4-12
- `mtest` on page 4-12
- `mfile` on page 4-13
- `mstat` on page 4-13
- `mverbose` on page 4-14.

4.2.1 help smtp

The `help smtp` command displays the list of SMTP commands for the Email Alerter.

**Syntax**

`help smtp`

**Example**

INET> `help smtp`
`mdel` delete an SMTP alert recipient
`mport` TCP port for SMTP alerts
`mrcpt` add/view SMTP alert recipients
`msserver` SMTP server IP address
`mtest` Send a test SMTP alert
`mfile` Email a disk file
`mstat` dump SMTP info
`mverbose` toggle SMTP verbose mode

INET> _
4.2.2 mdel

The mdel command deletes the email address of a selected recipient.

Syntax

mdel email_address

where:

email_address Is the email address to be deleted. If you enter mdel without this parameter, it lists the current email recipients.

Example

INET> mdel
Specify recipient to delete.
SMTP Alerts Enabled

SMTP Server IP address 207.156.252.7:25, currently disconnected, state:0.
Alert Recipients:
   jbrown@company.com
   lsmit@company.com
Alert Msps in Queue: 0
Message fates; OK: 0, bad code: 0, so_error: 0
INET> mdel lsmit@company.com
recipient deleted
INET> _

Usage

Use the mrct command add email addresses to the recipient list.
4.2.3 \texttt{mport}

The \texttt{mport} command shows the port number used for setting up a TCP connection to send the emails. This command can also be used to change the port number.

\textbf{Syntax}

\texttt{mport new\_port}

where:

\textit{new\_port} is the port number on the remote server machine where the attempted connection are made. If you enter \texttt{mport} without this parameter, it displays the current server port.

\textbf{Example}

\begin{verbatim}
INET> \texttt{mport}
smtp server port is currently 25
to change, enter new port number after this command
INET> \texttt{mport 27}
INET> \texttt{mport}
smtp server port is currently 27
to change, enter new port number after this command
INET> \\
\end{verbatim}
4.2.4 mrcpt

The mrcpt command displays the list of email addresses in the recipient list, and can also adds a new recipient to the list.

Syntax

mrcpt new_address

where:

new_address Is the new email address to be added to the recipient list. If you enter mrcpt without this parameter, it displays the current recipient list.

Example

INET> mrcpt
SMTP Alerts Enabled
SMTP Server IP address 207.155.248.7:25, currently disconnected, state:0.
Alert Recipients:
jbrown@company.com
Alert Msgs in Queue: 0
Message fates; OK: 0, bad code: 0, so_error: 0
To add recipient, type email address after command
INET> mrcpt lsmith@company.com
INET> mrcpt
SMTP Alerts Enabled
SMTP Server IP address 207.155.248.7:25, currently disconnected, state:0.
Alert Recipients:
jbrown@company.com
 lsmith@company.com
Alert Msgs in Queue: 0
Message fates; OK: 0, bad code: 0, so_error: 1
To add recipient, type email address after command
INET> _
4.2.5 mserver

The mserver command specifies the IP address of the SMTP Server.

Syntax

mserver new_address

where:

new_address Is the new address for the server.

Example

INET> mserver
smtp server is currently 207.155.248.7
to change, enter new IP address after this command
INET> mserver 207.156.252.7
INET> mserver
smtp server is currently 207.156.252.7
to change, enter new IP address after this command
INET> 

Usage

When an email is to be sent, an SMTP connection is made to this server. Without an SMTP server, no emails can be sent.

4.2.6 mtest

The mtest command sends a test email to everybody in the recipient list.

Syntax

mtest

Usage

Before using this command, you need to set up the recipient list and the IP address of the SMTP Server.
4.2.7   mfile

The **mfile** command sends the specified file as an email attachment to everybody in the recipient list.

**Syntax**

```
mfile file_name
```

where:

`file_name`  Is the name of the file to be sent.

**Usage**

Before using this command, you need to set up the recipient list and the IP address of the SMTP Server.

4.2.8   mstat

The **mstat** command shows statistics about the Email Alerter module.

**Syntax**

```
mstat
```

**Example**

```
INET> mstat
SMTP Alerts Enabled
SMTP Server IP address 207.155.248.7:25, currently disconnected, state:1.
Alert Recipients:
  jbrown@company.com
  lsmith@company.com
Alert Msgs in Queue: 2
  msg at 65E8:F930, state: 1
  msg at 65E8:FAEC, state: 1
last message reply 0 polls (5267 seconds) ago
Message fates; OK: 0, bad code: 0, so_error: 0
INET> _
```
4.2.9  mverbose

The mverbose command turns SMTP verbose mode on or off. When verbose mode is on, detailed information is displayed for all emails sent.

Syntax

mverbose

Example

INET> mverbose
verbose mode ON
INET> mverbose
verbose mode OFF
INET> _
4.3 FTP client commands

The commands in this section are:

- `help ftpc` on page 4-16
- `ascii` on page 4-16
- `binary` on page 4-16
- `cd` on page 4-17
- `fclose` on page 4-17
- `fverb` on page 4-17
- `fpasv` on page 4-17
- `ftp` on page 4-18
- `hash` on page 4-18
- `get` on page 4-19
- `put` on page 4-19
- `pwd` on page 4-19
- `ls` on page 4-20
- `fstate` on page 4-20.
4.3.1 help ftpc

The `help ftpc` command displays the list of FTP client commands.

**Syntax**

```
help ftpc
```

**Example**

```
INET> help ftp
ascii    use ASCII transfer mode
binary   use Binary transfer mode
cd       change server's directory
fclose   close FTP command connection
fverb    toggle verbose mode
fpasv    set server to passive mode
ftp       open an FTP connection
hash     toggle hash mark printing
get      GET a file
put      PUT a file
pwd      print working directory
ls       list files in server directory
fstate   display FTP client state
INET> _
```

4.3.2 ascii

The `ascii` command specifies that files are to be transferred in ASCII form, as opposed to binary form. This is the same as in a standard FTP client program.

**Syntax**

```
ascii
```

4.3.3 binary

The `binary` command specifies that files are to be transferred in binary form, as opposed to ASCII form. This is the same as in a standard FTP client program.

**Syntax**

```
binary
```
4.3.4 cd

The cd command changes the current directory on the server, in the same way as the standard FTP cd command.

Syntax

cd

4.3.5 fclose

The fclose command closes the FTP command connection.

Syntax

close

4.3.6 fverb

The fverb command toggles verbose mode on and off.

Syntax

fverb

Example

INET> fverb
ftp verbose mode off
INET> fverb
ftp verbose mode on
INET> _

4.3.7 fpasv

The fpasv command sets the FTP server to passive mode.

Syntax

fpasv
4.3.8 ftp

The ftp command tries to create an FTP command connection to the specified server.

**Syntax**

ftp IP_address [user_name] [password]

where:

*IP_address*  
Is the IP address of the target FTP server.

*user_name*  
(optional) Is the login name on the selected server.

*password*  
(optional) Is the password for the login name.

**Example**

INET> ftp 10.0.0.22 guest sesame
ftp> _

**Usage**

See `fclose` on page 4-17 for information on how to close this command connection.

4.3.9 hash

The hash command turns hash mark printing on or off.

**Syntax**

hash

**Usage**

Most FTP clients support this option so that you can monitor the progress of an FTP data transfer. When hash printing is enabled, the client displays hash (#) marks at intervals to show that the data transfer is progressing.
4.3.10 get

The get command retrieves a file from a remote FTP server.

**Syntax**

```
get remote_file [local_file]
```

where:

- `remote_file` is the name of the file on the remote FTP server that is to be retrieved to the local client machine.
- `local_file` (optional) specifies the name for the file that is created on the local machine. If `local_file` is not specified, the remote filename is used for the local file.

4.3.11 put

The put command puts a file onto a remote FTP server.

**Syntax**

```
put local_file [remote_file]
```

where:

- `local_file` specifies the file on the local machine.
- `remote_file` (optional) is the name for the file that is created on the remote FTP server. If `remote_file` is not specified, the local filename is used for the remote file.

4.3.12 pwd

The pwd command prints the name of the current working directory on the remote FTP server file system.

**Syntax**

```
pwd
```
4.3.13  ls

The `ls` command lists the contents of the current working directory on the remote FTP server.

**Syntax**

`ls`

4.3.14  fstate

The `fstate` command displays information about FTP clients.

This command is included in the menu list if NET_STATS was defined in `ipport.h.h`.

**Syntax**

`fstate`

**Example**

```
INET> fstate
state: command in progress, mode:ascii
server: 10.0.0.70, data port:20
Hashing: OFF, passive: off
INET> _
```
4.4  Ping commands

The commands in this section are:

- help ping on page 4-21
- ping on page 4-22
- delay on page 4-23
- host on page 4-24
- length on page 4-25
- endping on page 4-26
- pstats on page 4-27.

4.4.1  help ping

The help ping command shows the command set for ping.

Syntax

help ping

Example

INET> help ping
ping commands:
  ping    - Ping [host] [#times]
  delay   - set milliseconds to wait between pings
  host    - set default active IP host
  length  - set default ping packet length
  endping - terminate the current ping session
  pstats  - display statistics about ping
INET>
4.4.2 ping

The ping command is an ordinary IP Ping utility. It can be sent to any host at any time.

Syntax

ping [host] [repeat]

where:

host (optional) Is an IP host address. If no IP address is specified, ping uses the default host.

repeat (optional) Is the number of times to ping.

Example

Assuming no host IP address or interface IP address has been specified:

INET> ping 10.0.2.1
ping 0 to 10.0.0.1: Arping for host...
INET> setip 10.0.2.3
WARNING: 'setip' will kill all current net connections!!!!
replacing net[0] IP address 192.168.5.34 with 10.0.2.3
INET> ping 10.0.2.1
ping 0 to 10.0.2.1: Arping for host...
got ping reply; len :62 seq 0 from 10.0.2.1
INET> ping 10.0.2.1 3
...use endping command to stop pinging...
ping 0 to 10.0.2.1: ping 0 sent...
got ping reply; len :62 seq 0 from 10.0.2.1
INET> ping 1 10.0.2.1: ping 1 sent...
got ping reply; len :62 seq 1 from 10.0.2.1
INET> ping 2 10.0.2.1: ping 2 sent...
got ping reply; len :62 seq 2 from 10.0.2.1
INET> ping complete; sent 3, received 3
INET> ping
specify valid IP host, or use default active host
INET> host 10.0.0.1
INET> ping
ping 0 to 10.0.0.1: Arping for host...
got ping reply; len :62 seq 0 from 10.0.0.1
INET> ping 3
...use endping command to stop pinging...
ping 0 to 10.0.0.1: ping 0 sent...
got ping reply; len :62 seq 0 from 10.0.0.1
INET> ping 1 10.0.0.1: ping 1 sent...
got ping reply; len :62 seq 1 from 10.0.0.1
INET> ping 2 10.0.0.1: ping 2 sent...
got ping reply; len :62 seq 2 from 10.0.0.1
INET> ping complete; sent 3, received 3
INET>

Usage

See length on page 4-25 to set the length of the ping packets, and delay on page 4-23 to set the inter-packet delay of the pings.

4.4.3 delay

The delay command sets the time to wait between pings when the multi-packet ping option is used.

Syntax

delay milliseconds

where:

milliseconds Is the number of milliseconds to wait between pings. The value is rounded off to the nearest clock tick (1/18th of a second, which is approximately 56 milliseconds). The default is 1000 milliseconds. Setting the time to less than one clock tick sends the pings with no inter-frame delay.

If you do not enter this parameter, the delay command displays the current delay value.

Example

INET> delay
current ping delay is 1008
to set, enter number of milliseconds on command line.
INET> delay 100
set inter-ping delay to (approx) 100 ms
INET> delay
current ping delay is 56
to set, enter number of milliseconds on command line.
INET> _
### 4.4.4 host

The `host` command sets the default host for subsequent commands.

#### Syntax

```
host IP_host
```

where:

- `IP_host` Is the dot notation IP address of the new host.

#### Usage

The default ping command (with no `host` parameter) uses the host that was specified set with a `host` command. If you change the host while a session is open, the session is closed and a new session is opened with the new host.

For more information on ping settings, see `ping` on page 4-22.
4.4.5 length

The length command sets the length of ping packets.

Syntax

length packet

where:

packet Is a number, usually in the range 60 - 1500, and represents the length of the ICMP data sent in the ping.

If you do not enter this parameter, the length command displays the current ping packet length.

Example

INET> length
default ping length is 64

INET> ping
ping 0 to 10.0.0.1: Arping for host...
got ping reply; len :62 seq 0 from 10.0.0.1
INET> length 59
CAUTION: 59 is unusual length
INET> length 60
INET> length
default ping length is 60
To change it, put new number on command line
INET> ping
ping 0 to 10.0.0.1: ping 0 sent...
got ping reply; len :58 seq 0 from 10.0.0.1
INET> length 1501
CAUTION: 1501 is unusual length
INET> length 1500
INET> ping
ping 0 to 10.0.0.1: ping 0 sent...
got ping reply; len :1498 seq 0 from 10.0.0.1
INET> _
4.4.6 endping

The endping command terminates the current ping session.

Syntax

endping

Example

INET> ping 10.0.0.20 15
...use endping command to stop pinging...
ping 0 to 10.0.0.20: Arping for host...
got ping reply; len :1022 seq 0 from 10.0.0.20
INET> ping 1 to 10.0.0.20: ping 2 sent...
got ping reply; len :1022 seq 1 from 10.0.0.20
INET> ping 2 to 10.0.0.20: ping 3 sent...
got ping reply; len :1022 seq 2 from 10.0.0.20
INET> ping 3 to 10.0.0.20: ping 4 sent...
got ping reply; len :1022 seq 3 from 10.0.0.20
INET> endping
ping complete; sent 4, received 4
INET> _

Usage

The characters of the typed endping command might be separated because of incoming ping replies. Even if this is the case, the endping command still works.
4.4.7 pstats

The pstats command displays statistics about the ping settings.

Syntax

pstats

Example

INET> ping
ping 0 to 10.0.0.1: ping 0 sent...
got ping reply; len :1498 seq 0 from 10.0.0.1
INET> pstats
Default ping delay time: 1000 ms.
Default ping host: 10.0.0.1
Default ping pkt length: 1500 bytes
There are 0 ongoing ping sessions.
INET> delay 5000
set inter-ping delay to (approx) 5000 ms.
INET> ping
ping 0 to 10.0.0.1: ping 0 sent...
got ping reply; len :1498 seq 0 from 10.0.0.1
INET> pstats
Default ping delay time: 5000 ms.
Default ping host: 10.0.0.1
Default ping pkt length: 1500 bytes
Statistics about pinging 10.0.0.1
Times=0, Length=1500, Delay=5000
Packets : sent=1, received=1

There are 1 ongoing ping sessions.
INET>
4.5 NAT Router commands

The NAT menu routines are used in the menuing system in ..\misc\lib\menu*.s. These routines should be portable to systems using the menus, but they are not required for basic NAT functionality.

The NAT commands in this section are:

- `help nat` on page 4-28
- `natstats` on page 4-29
- `natconns` on page 4-30
- `natentry` on page 4-31
- `naliases` on page 4-32
- `nproxies` on page 4-32
- `nxip` on page 4-33.

4.5.1 help nat

The `help nat` command displays the command set for the NAT Router.

**Syntax**

`help nat`

**Example**

```
INET> ? nat
natstats   display general NAT statistics
natconns   display NAT connection table
natentry   NAT connection detail
naliases   show alias list
nproxies   show proxy list
nxip       expunge IP address from NAT tables
INET> _
```
4.5.2 natstats

The natstats command displays the general statistics for the NAT Router.

Syntax

natstats

Example

INET> natstats
local IP: 10.0.0.1 local mask: 255.0.0.0
Internet IP: 209.220.44.220 local mask: 255.255.255.0
timeouts: TCP: 500, UDP: 60
local to inet: pkts:1804, bytes:103876
inet to local: pkts:509, bytes:225773
maxmss: 0, max TCP window: 0
Connections: TCP:4, UDP:0, ICMP:1, created: 496, deleted: 491
Errors: cksum: 0, retries: 478, bad packets: 0
Total IP pkts: 2160, Reserved addresses: 363
ENCAP: rx: 00000000, encap: 00000000, decap: 00000000
mkfrag: 00000000, rxfrag: 00000000
INET> _
4.5.3 natconns

The natconns command lists statistics for all open NAT connections.

Syntax

natconns

Example

INET> natconns
No open Connections
INET> natconns
TCP: 149.1.1.31:17027 <-> 10.0.0.4:1104
   Out_port: 1584, pkts: out 27, in 26, state: 4 encap: 0
TCP: 149.1.1.31:17027 <-> 10.0.0.4:1103
   Out_port: 1583, pkts: out 1, in 1, state: 4 encap: 0
TCP: 207.82.70.13:443 <-> 10.0.0.5:3370
   Out_port: 1582, pkts: out 28, in 21, state: 4 encap: 0
TCP: 207.82.70.13:443 <-> 10.0.0.5:3368
   Out_port: 1580, pkts: out 25, in 20, state: 4 encap: 0
TCP: 207.82.70.13:443 <-> 10.0.0.5:3367
   Out_port: 1579, pkts: out 22, in 22, state: 4 encap: 0
TCP: 207.82.70.13:443 <-> 10.0.0.5:3366
   Out_port: 1578, pkts: out 26, in 22, state: 4 encap: 0
TCP: 207.82.70.13:443 <-> 10.0.0.5:3365
   Out_port: 1577, pkts: out 26, in 24, state: 4 encap: 0
TCP: 207.82.70.13:443 <-> 10.0.0.5:3363
   Out_port: 1575, pkts: out 36, in 31, state: 4 encap: 0
TCP: 207.155.252.4:80 <-> 10.0.0.5:3321
   Out_port: 1490, pkts: out 7, in 1, state: 4 encap: 0
INET> _
4.5.4 natentry

The natentry command shows information about a specified NAT connection.

**Syntax**

```
natentry port
```

where:

- **port** is an outside port number.

**Example**

```
INET> natentry
enter outside port number of connection on command line
use "natconns" command to get port list
INET> natentry 1525
Foreign IP: 205.188.247.0, Local IP: 10.0.0.5
Ports: outside: 1525, inside: 3336, foreign: 80
outgoing: pkts: 8, bytes: 938
incoming: pkts: 6, bytes: 3689
Type TCP, seconds since use 114
TCP Seq: 381308120, Ack: 2083411835, state: 5
Retrys: Local: 0, Foreign: 0
Bad checksum: Local: 0, Foreign: 0
INET> _
```
4.5.5 naliases

The *naliases* command shows the list of NAT aliases.

This command is included in the menu list if NAT_ALIASLIST was defined in *ipport.h*.

**Syntax**

```
naliases
```

**Example**

```
INET> naliases
205.206.207.3 aliased to 10.0.0.52
205.206.207.2 aliased to 214.69.218.2
INET> _
```

**Usage**

You set up aliases in the *natdb.nv* file.

4.5.6 nproxies

The *nproxies* command shows the list of NAT proxies.

This command is included in the menu list if NAT_PROXYLIST was defined in *ipport.h*.

**Syntax**

```
nproxies
```

**Example**

```
INET> nproxies
TCP port 19 mapped to 10.0.0.52:19
TCP port 21 mapped to 10.0.0.52:21
TCP port 80 mapped to 10.0.0.52:80
TCP port 61 mapped to 10.0.0.52:61
INET> _
```

**Usage**

You set up proxies in the *natdb.nv* file.
4.5.7 nxip

The nxip command removes an IP address from NAT tables. This is useful if an IP address on the local network has been changed, and needs to be removed.

**Syntax**

nxip *IP_address*

where:

*IP_address* Is the address to remove.

**Example**

INET> nxip enter IP address to expunge on command line
INET> nxip 10.0.0.75
INET> _
4.6 Routing Information Protocol (RIP) commands

The ripmenu.c file contains code to support RIP commands from the main menu.

The RIP commands in this section are:
- help rip on page 4-34
- ripstatistics on page 4-35
- riproute on page 4-36
- ripauth on page 4-36
- riprefuse on page 4-37
- ripglobals on page 4-37
- ripaddroute on page 4-38.

4.6.1 help rip

The help rip command shows the command set for RIP.

Syntax

help rip

Example

INET> help rip
SNMP Station: Rip Commands:
  ripstatistics - display rip statistics
  riproute     - display rip route table
  ripauth      - display rip authentication table
  riprefuse    - display rip refuse list
  ripglobals   - display rip global list
  ripaddroute  - add a route to route table
INET> _

Usage

You can add new routes using the route or ripaddroute command. If the new route is for RIP, it is recommended that you use ripaddroute, because RIP entries keep more details (for example, metrics) about a route.
4.6.2 ripstatistics

The ripstatistics command displays RIP statistics.

Syntax

ripstatistics

Example

INET> ripstatistics
Showing statistics gathered for RIP protocol.
Number of version errors= 0
Number of address family errors= 0
Number of packets dropped from a host on the refuse list= 0
Refused due to wrong domain for interface= 0
Authentication failures= 0
Unknown authentication type= 0

Now some statistics about each of RIP versions.
Version Number=0
Packets sent; request 0, response 0, reply 0
Packets received; total 0, request 0, response 0
Number of unknown command pkts received = 0

Version Number=1
Packets sent; request 0, response 0, reply 0
Packets received; total 0, request 0, response 0
Number of unknown command pkts received = 0

Version Number=2
Packets sent; request 0, response 0, reply 0
Packets received; total 0, request 0, response 0
Number of unknown command pkts received = 0
INET> _
4.6.3 riproute

The riproute command displays the RIP route table.

**Syntax**

riproute

**Example**

INET> riproute
Destination..Gateway......Metric..Mask.......MainTimer..SecTimer..Iface  
10.0.0.0 10.0.0.22 1 255.0.0.0 1 0 1  
Number of routes = 1
INET> ripaddroute 192.9.200.54,255.0.0.0,10.0.0.1,0,1,180,0,0.0.0.0  
INET> riproute  
Destination..Gateway......Metric..Mask.......MainTimer..SecTimer..Iface  
10.0.0.0 10.0.0.22 1 255.0.0.0 1 0 1  
192.9.200.54 10.0.0.1 1 255.0.0.0 29627 0 0  
Number of routes = 2
INET> _

4.6.4 ripauth

The ripauth command displays the RIP authentication table.

**Syntax**

ripath

**Example**

INET> ripauth  
Number of entries = 0
INET> _
4.6.5 riprefuse

The riprefuse command displays the IP addresses in list of refused RIP requests.

Syntax

riprefuse

Example

INET> riprefuse
Number of entries = 0
INET> _

4.6.6 ripglobals

The ripglobals command displays the values of global variables used by RIP.

Syntax

ripglobals

Example

INET> ripglobals
Values of global variables of RIP
    rip_default_flags       =1
    rip_default_ttl         =180
    rip_def_bcast_interval  =30
    rip_def_deletion_interval =120
    rip_def_trigger_interval =5
    rip_num_of_ifaces       =1
    rip_bcast_timer         =24796
    rip_trigger_timer       =1
    rip_trigger_timer_interval =2
    rip_allow_default_gateways =0
Interface..RIP Version Flag(Receive,Send)
    1 (3,3)
INET> _
4.6.7 ripaddroute

The ripaddroute command adds a route to the RIP table.

Syntax

```
ripaddroute dest subnetmask gw iface metric ttl flags proxy
```

where:

- `dest` Is the destination IP address.
- `subnetmask` Is the subnet mask.
- `gw` Specifies the gateway.
- `iface` Specifies the interface number.
- `metric` Enters the number of hops.
- `ttl` Gives the time to live for this entry.
- `flags` Specifies the flags for this entry, where:
  - 0 is the default.
  - 1 is (RIP_PRIVATE), to be used for permanent entries.
- `proxy` Is the proxy IP address for RIP-2.

Example

```
INET> ripaddroute
usage: ripaddroute dest,subnetmask,gw,iface,metric,ttl,flags,proxy
Eg: ripaddroute 192.9.200.54,255.255.255.0,10.0.0.1,1,1,180,0,0,0,0
INET> ripaddroute 192.9.200.54,255.0.0.0,10.0.0.1,0,1,180,0,0,0,0
INET> riproute
Destination..Gateway......Metric..Mask.......MainTimer..SecTimer..Iface
10.0.0.0 10.0.0.22 1 255.0.0.0 1 0 1
192.9.200.54 10.0.0.1 1 255.0.0.0 29627 0 0
Number of routes = 2
INET> _
```

Usage

It is helpful to type ripaddroute without any arguments first to get information about the arguments, before you edit any values.
4.7 TELNET commands

The commands in this section are:

- help telnet on page 4-39
- tshow on page 4-40
- tstats on page 4-41
- logout on page 4-41
- exit on page 4-41.

4.7.1 help telnet

The help telnet command shows the commands for the TELNET Server.

**Syntax**

help telnet

**Example**

INET> help telnet
telnet commands:
tshow - show the options values for all sessions
tstats - show the statistics of all TELNET sessions
logout - logout of the TELNET session
exit - logout of the TELNET session
INET>
4.7.2 tshow

The tshow command shows the values for options used by each TELNET connection.

**Syntax**

```
tshow
```

**Example**

```
INET> tshow
Showing OPTION values for each telnet session....

Session 1 : Socket is 436104......
[0]Binary: configurable=1
For Local Session:value=0, req sent=0, nego=0
For Remote Session:value=0, req sent=0, nego=0
[1]Echo: configurable=1
For Local Session:value=1, req sent=0, nego=1
For Remote Session:value=1, req sent=0, nego=1
[3]Supress Go Ahead: configurable=1
For Local Session:value=1, req sent=0, nego=1
For Remote Session:value=1, req sent=0, nego=1
[5]Status: configurable=1
For Local Session:value=0, req sent=0, nego=1
For Remote Session:value=0, req sent=0, nego=1
INET>
```
4.7.3 tstats

The tstats command shows the statistics for all TELNET sessions.

Syntax

tstats

Example

INET> tstats
Total connections opened = 2
Total connections closed = 0

Telnet Session 1: Showing statistics for socket 1870265322.
Bytes rcvd=103, Cmds rcvd = 12

Telnet Session 2: Showing statistics for socket 1870263406.
Bytes rcvd=47, Cmds rcvd = 3
Number of ongoing telnet sessions = 2.
INET> _

4.7.4 logout

The logout command logs you out of the TELNET session.

Syntax

logout

4.7.5 exit

The exit command logs you out of the TELNET session.

Syntax

exit
Protocol-specific Commands
Glossary

**ADS**  
ARM Developer Suite.

**ARP**  
Address Resolution Protocol.

**BSD**  
Berkeley System Distribution.

**CLI**  
Command Line Interface.

**DHCP**  
Dynamic Host Configuration Protocol.

**DNS**  
Domain Name System.

**FTP**  
File Transfer Protocol.

**HTTP**  
Hypertext Transfer Protocol.

**ICMP**  
Internet Control Message Protocol.

**IP**  
Internet Protocol.

**MAC**  
Media Access Control.

**MIB**  
Management Information Base.

**NAT**  
Network Address Translation.

**NV**  
Nonvolatile.

**PPP**  
Point-to-Point Protocol.
### Glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RIP</strong></td>
<td>Routing Information Protocol.</td>
</tr>
<tr>
<td><strong>SNMP</strong></td>
<td>Simple Network Management Protocol</td>
</tr>
<tr>
<td><strong>SMPT</strong></td>
<td>Simple Mail Transfer Protocol.</td>
</tr>
<tr>
<td><strong>TCP</strong></td>
<td>Transmission Control Protocol.</td>
</tr>
<tr>
<td><strong>TFTP</strong></td>
<td>Trivial File Transfer Protocol.</td>
</tr>
<tr>
<td><strong>UDP</strong></td>
<td>User Datagram Protocol.</td>
</tr>
<tr>
<td><strong>VFS</strong></td>
<td>Virtual File System.</td>
</tr>
</tbody>
</table>
The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.

A

aliases, NAT Router 4--32
allocsize command 3--11
ARP table 3--14
arps command 3--14
ascii command 4--16
ASCII files 4--16

B

binary command 4--16
binary files 4--16
bits, UPTRACE 3--12
BOOTP packets 4--3
breakpoints, setting 3--11
buffers command 3--4

C

cd command 4--17
changing directories 4--17
diagnostic

dchap command 3--32
CHAP statistics 3--32
CHAP_SUPPORT variable 3--32
CLI
console 1--3
menu 3--2
output 2--9
overview 1--2
quitting from 2--4
client sockets 2--13
closing
all protocols 3--26
FTP connections 4--17
ping session 4--26
PPP link 2--22
TCP echo clients 2--19
TCP echo server 2--18
Telnet session 4--41
UDP echo client 2--13
UDP echo server 2--14
Command-line Interface. See CLI 1--2
commands
Index

modem 3--27
nslookup 3--19
pcons 3--31
pfile 3--32
queues 3--5
routes 3--35
rtadd 3--36
rtdel 3--37
snmpinfo 3--39
snmpstat 3--38
sockets 3--22
statistics 3--14
swirl 3--13
tbconn 3--23
thrcv 3--25
tcp 3--22
udp 3--16
upcall 3--12
entering 1--4
general 2--2
delays between 2--8
help 2--2
history 2--5
host 2--12
list of used 2--5
logfile 2--7
number displayed 2--5
NV parameters 2--24
nvset 2--24
obey 2--6
operating system 2--9
PPP 2--21
pppdown 2--22
ppopup 2--21
quit 2--4
setup 2--8
sleep 2--8
SNMP 2--23
state 2--4
TCP echo 2--16
techalt 2--19
tesend 2--17
teshalt 2--18
tesinit 2--18
testats 2--20
trap 2--23
UDP echo 2--11
uechalt 2--13
uesend 2--12
ueshalt 2--14
uesinit 2--13
uestats 2--15
version 2--9
protocol_specific
ascii 4--16
binary 4--16
cd 4--17
delay 4--23
DHCP 4--2
dhdelete 4--6
dhentry 4--5
dhlist 4--4
dhpools 4--7
dhserv 4--3
Email Alerter 4--8
ending 4--26
exit 4--41
close 4--17
fpasv 4--17
fstate 4--20
ftp 4--18
fverb 4--17
get 4--19
hash 4--18
help dhcp 4--2
help ftp 4--16
help nat 4--28
help ping 4--21
help rip 4--34
help smtp 4--8
help telnet 4--39
host 4--24
length 4--25
logout 4--41
ls 4--20
mdel 4--9
mfile 4--13
mimport 4--10
mrpt 4--11
mserver 4--12
mstat 4--13
mttest 4--12
mverbos 4--14
naliases 4--32
natconns 4--30
natentry 4--31
natstats 4--29
nproxies 4--32
nxip 4--33
ping 4--22
pstats 4--27
put 4--19
pwd 4--19
ripaddroute 4--38
ripauth 4--36
rippools 4--37
riprefuse 4--37
riproute 4--36
ripstatistics 4--35
tshow 4--40
tstats 4--41
runtime 1--4
console 1--3, 3--31
output to file 2--7
test patterns 3--13
current state 2--4

D
database entries, DHCP 4--5, 4--6
dbytes command 3--6
dcstats command 3--17
ddebug command 3--7
debugging
hooking the debugger 3--9
IP stack 3--7
memory blocks 3--6
trace types 3--7
delay command 4--23
delays between commands 2--8
deleting, email addresses 4--9
DHCP
assigning addresses entries 4--7
commands 4--2
database entries 4--5, 4--6
IP addresses 4--4
statistics 4--3
DHCP_CLIENT variable 3--17
dhdelete command 4--6
dhentry command 4--5
dhlist command 4--4
dhpools command 4--7
dherv command 4--3
dir command 3--29
directories
changing between 4--17
current FTP 4--19, 4--20
displaying
  FTP transfer progress 4--18
socket list 3--22
DNS commands 3--18
dnsstats command 3--18
DNS_CLIENT variable 3--18, 3--19
dtrap command 3--9
dump command 3--9
dumping
  bytes 3--9
  memory blocks 3--34
  packet buffer queues 3--5
dynamic memory 3--20

E
email addresses
  adding 4--11
  listing 4--11
Email Alerter
  adding email addresses 4--11
  commands 4--8
  deleting email addresses 4--9
  detailed messages 4--14
  email attachments 4--13
IP address 4--12
  listing email addresses 4--11
  statistics 4--13
TCP connection 4--10
  test emails 4--12
endping command 4--26
entering 1--4
  commands 1--4
  parameters 1--4
exit command 4--41

F
fclose command 4--17
FD_SETSIZE variable 2--17
files
  ASCII format 4--16
  binary format 4--16
  email attachments 4--13
  getting via FTP 4--19
  log 2--7
  obey 2--6
  ppp.log 3--32
  putting via FTP 4--19
  ripmenu.c 4--34
  webport.nv 2--24
  fpasv command 4--17
  fstate command 4--20
FTP
  client statistics 4--20
  clients 4--20
  closing a connection 4--17
  commands 4--15
  displaying transfer progress 4--18
  getting a file 4--19
  opening a connection 4--18
  passive server 4--17
  putting a file 4--19
  working directory 4--19, 4--20
ftp command 4--18
  functions, npalloc 3--11
  fverb command 4--17

G
general commands 2--2
general diagnostic commands 3--2
get command 4--19

H
hangup command 3--26
hardware statistics 3--10
hash command 4--18
hash mark printing 4--18
help command 2--2
help dhcp command 4--2
  help diagnostic command 3--3
help fverb command 4--16
help nat command 4--28
help ping command 4--21
help rip command 4--34
help smtp command 4--8
help telnet command 4--39
history command 2--5
host command 2--12, 4--24
host IP address 4--24
host names 3--19
hstat command 3--28
HTTP
  commands 3--28
  statistics 3--28

I
icmpstat command 3--16
iface command 3--33
INCLUDE_NVPARAMS variable
  2--24
INCLUDE_SNMP variable 2--23,
  3--38, 3--39
INCLUDE_TCP variable 3--20
interface statistics 3--33
IN_MENUS variable 3--2
IP
  4--37
    adding a route 3--36
    addresses from DHCP server 4--4
    commands 3--35
    deleting a route 3--37
    free addresses 4--7
    host address 4--24
    NAT Router addresses 4--33
    ping utility 4--22
    route tables 3--35
    setting addresses 2--8
    SMTP server address 4--12
    stack debugging 3--7
    statistics 3--15
    TCP echo server address 2--17
ipstat command 3--15
IP_ROUTING variable 3--35, 3--36,
  3--37

L
length command 4--25
linkstats command 3--10
listing used commands 2--5
logfile command 2--7
logout command 4--41
is command 4--20
Index

M
MANUAL_PPP variable 2--21, 2--22
mbuf command 3--20
mbufs 3--20, 3--21
mdel command 4--9
memory
allocation 3--11
blocks 3--6
diagnostic command for 3--34
dynamic 3--20
leaks 3--34
statistics 3--34
memory command 3--34
MEM_BLOCKS variable 3--34
MENU_HISTORY variable 2--5
messages
Email Alerter 4--14
status 3--7
mfile command 4--13
mlist command 3--21
modem
diagnostic commands for 3--26
hangup/reset 3--26
statistics 3--27
modem command 3--27
mport command 4--10
mrctl command 4--11
mserver command 4--12
mstat command 4--13
mtest command 4--12
mverbose command 4--14

N
naccons command 4--30
nalias command 4--32
NAT Router 4--28
aliases 4--32
commands 4--28
connections 4--30, 4--31
proxies 4--32
removing IP addresses 4--33
statistics 4--29, 4--30, 4--31
NAT tables 4--33
natentry command 4--31
natstats command 4--29
NAT_ALIASLIST variable 4--32
NAT_PROXYLIST variable 4--32
network interfaces, setting IP addresses 2--8
NET_STATS variable 3--2, 3--14, 3--27, 3--32, 3--33, 3--34
npalloc function 3--11
nproxies command 4--32
nslookup command 3--19
NV parameters command 2--24
nvset command 2--24
nxip command 4--33

O
obey command 2--6
obey file 2--6
OBEY prompt 2--6
opening 2--12
FTP connections 4--18
PPP link 2--21
RIP routes 4--38
TCP echo client 2--17
UDP echo client 2--12
operating system commands 2--9

P
packet buffers
queues 3--5
statistics 3--4
packets
BOOTP 4--3
ping delay 4--23
setting length of 4--25
tracing 3--12
parameters 1--4
entering 1--4
nonvolatile 2--24
passive FTP server 4--17
pcoms command 3--31
pfie command 3--32
ping
closing the session 4--26
length of packets 4--25
multi-packet delay 4--23
statistics 4--27
ping command 4--22

Q
queues command 3--5
quit command 2--4

R
refused addresses 4--37
resetting a modem 3--26
RIP
adding a route 4--38
authentication table 4--36
commands 4--34
global variables 4--37
IP addresses refused 4--37
route table 4--36
statistics 4--35
ripaddroute command 4--38
ripassault command 4--36
ripglobals command 4--37
ripmenu.c file 4--34
riprefuse command 4--37
riproute command 4--36
ripstatistics command 4--35
routes
adding to IP table 3--36
deleting from IP table 3--37
routes command 3--35
rtadd command 3--36
rtdel command 3--37
Index

running scripts 2--8
runtime commands 1--4

S
setip command 2--8
setting
breakpoints 3--11
IP addresses 2--8
nonvolatile parameters 2--24
sleep command 2--8
SNMP
agent 3--39
commands 2--23, 3--38
counters 3--38
sending a trap 2--23
snmpinfo command 3--39
snmpstat command 3--38
sockets
client 2--13
displaying list of 3--22
sockets command 3--22
starting
TCP echo server 2--18
UDP echo server 2--13
state
current 2--4
protocol stack 3--7
state command 2--4
statistics 4--20
ARP 3--14
BSD 3--23
CHAP 3--32
DHCP 3--17, 4--3
diagnostic commands for 3--14
DNS 3--18
Email Alerter 4--13
hardware 3--10
HTTP 3--28
interface 3--33
IP 3--15, 3--16
IP route table 3--35
link layer 3--10
memory 3--34
modem 3--27
NAT Router 4--29, 4--30, 4--31
packet buffers 3--4
ping 4--27
RIP 4--35
SNMP 3--38
TCP 3--22, 3--23, 3--25
TCP echo server 2--20
Telnet 4--41
UART 3--27
UDP 3--16
UDP echoes 2--15
status messages 3--7
swirl command 3--13
scp command 2--15

T
	bconn command 3--23
	bibrcv command 3--25
TCP
commands 3--20
e-mail connections 4--10
maximum number of connections 2--17
tcp command 3--22
TCP echo clients 2--17, 2--19
closing 2--19
opening 2--17
TCP echo commands 2--16
TCP echo server 2--19
closing 2--18
IP address 2--17
starting 2--18
statistics 2--20
TCP/IP stack version 2--9
TCP_ECHOTEST variable 2--16
TCP_IDLE_TIMEOUT variable 2--17
technalt command 2--19
telnet
commands 4--39
connections 4--40
logging out 4--41
quitting from 4--41
statistics 4--41
tesend command 2--17
teshalt command 2--18
testinit command 2--18
teststats command 2--20
testing
e-mail addresses 4--12
serial drivers 3--13
telnet server code 3--13
UDP protocol 2--13
timeout 2--17
tracing
debug types 3--7
PPP information 3--31, 3--32
received packets 3--12
trap command (SNMP) 2--23
tshow command 4--40
tstats command 4--41

U
UART statistics 3--27
UDP client socket 2--12
udp command 3--16
UDP echo client
closing 2--13
opening 2--12
UDP echo commands 2--11
UDP echo packet 2--12
UDP echo server
closing 2--14
starting 2--13
statistics 2--15
UDP protocol testing 2--13
UDPSTEST variable 2--11
UDP_IDLE_TIMEOUT variable 2--12
uechalt command 2--13
uesend command 2--12
ueshalt command 2--14
uestinit command 2--13
uestats command 2--15
upcall command 3--12
UPTRACE bit 3--12
USE_MODEM variable 3--26, 3--27
USE_PPP variable 3--31, 3--32

V
variables
CHAP_SUPPORT 3--32
DHCP_CLIENT 3--17
DNS_CLIENT 3--18, 3--19
FD_SETSIZE 2--17
global(RIP) 4--37
INCLUDE_NVPARAMS 2--24
Index

INCLUDE_SNMP  2--23, 3--38, 3--39
INCLUDE_TCP  3--20
IN_MENUS  3--2
IP_ROUTING  3--35, 3--36, 3--37
MANUAL_PPP  2--21, 2--22
MEM_BLOCKS  3--34
MENU_HISTORY  2--5
NAT_ALIASLIST  4--32
NAT_PROXYLIST  4--32
NET_STATS  3--2, 3--14, 3--27, 3--32, 3--33, 3--34
TCP_ECHOTEST  2--16
TCP_IDLE_TIMEOUT  2--17
UDPSTEST  2--11
UDPTEST  2--11
UDP_IDLE_TIMEOUT  2--12
USE_MODEM  3--26, 3--27
USE_PPP  3--31, 3--32
WEBPORT  3--28
verbose mode  4--17
version command (TCP/IP stack)  2--9

W

WEBPORT variable  3--28
webport.nv file  2--24

Symbols

! command  2--9