Datalight ROM-DOS®

Single User Network Add-On

Printed: August 2003

Single User Network Add-On

Part Number: 3010-0200-0531

Contents

Chapter 1, SOCKETS Introduction

About SOCKETS

System Requirements

SOCKETS requires an IBM compatible 186 or higher system with a minimum:

- 512KB of RAM
- ROM-DOS 6.22 or compatible

SOCKETS network operation requires one or more of the following, depending on the network configuration:

- Any network interface supporting the packet driver specification class 1, 3, 6 or 11.
- PC compatible asyncronous serial ports (COM ports).
- Any network supporting the ODI specification

Datalight SOCKETS is an Internet protocol software extension to ROM-DOS that provides a powerful data communication facility whereby embedded systems and users of embedded systems can communicate with other computers (including PCs and mainframes) and their printers. Datalight SOCKETS also provides the facilities to run custom-written applications which allows you to:

- Run applications on a TCP/IP host system from a remote embedded system.
- Transfer data between an embedded system and TCP/IP hosts.
- Run network aware applications on an embedded system.
- Print to an embedded system from TCP/IP hosts and vice versa.

Datalight SOCKETS consists of :

- A TSR kernel:

  Connecting to a physical Ethernet or Token Ring network using a network interface with associated Packet Driver and/or to a point-to-point serial network using standard serial communication ports with or without modem dial in/out.

  Implementing standard Internet protocols ARP, PPP, LCP, IPCP, PAP, CHAP MD5, IP, ICMP, IGMP, RIP, UDP, TCP, BOOTP, DHCP and DNS.

  Providing IP routing support.

  Providing two Application Programming Interfaces (APIs)

  Providing a Socket Print client

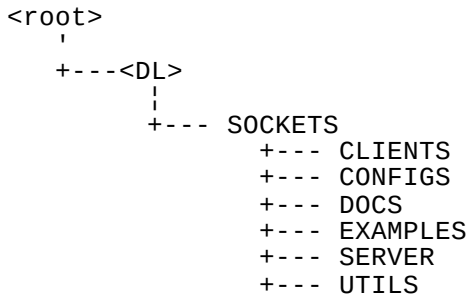
  Providing a Socket Print Server and LPD Server
- A SOCKETS configuration program.
- Utility programs to test the network and display the status of the kernel.
- Mail programs in source and binary format.

- Resident servers for FTP, HTTP and Remote Console including a CGI API for serving dynamic web-pages and a Remote Console Java applet to emulate a DOS console of the embedded system on a Java capable browser.
- Support for telnet clients, including an ANSI/VT emulator.
- An FTP client and a simple HTTP file GET utility.
- Print clients for SOCKETS printing and LPD printing (LPR).
- A resident RFC compliant NETBIOS API.

Chapter 2, SOCKETS Installation

Installing and Running SOCKETS

Upon receiving SOCKETS, electronic or CD, SDK or demo, the first step involved is installing to a Windows 9X or WIN NT based development system. The install process will create, by default, a directory of C:\DL\SOCKETS.



Default Directory Structure

The files installed cover:

- Kernel applications (the core of TCP/IP communication)
- Configuration examples
- Utilities
- Optional applications

The SDK does not include drivers for Network Interface controllers; hereafter referred to as a NIC. Please refer to Chapter 3 for information on obtaining and configuring network drivers as well as serial configuration details.

The remainder of this chapter describes how to proceed after SOCKETS has been installed to the development system.

Development System Procedure

Insert the ROM-DOS / SOCKETS SDK into your CD-ROM and run "D:\SETUP" where "d" is the appropriate drive letter for your CD. The install process will create a DL\SOCKETS directory. Within the directory DL\SOCKETS will exist the kernel applications SOCKETM.EXE and SOCKETP.EXE. Beyond that, there are subdirectories for:

- UTILS - contains tools for troubleshooting and configuring SOCKETS.
- CONFIGS - contains example configuration files for SOCKETS connections.
- EXAMPLES - contains programming examples.
- CLIENTS - contains the client applications.

- SERVER - contains the SOCKETS server providing HTTP and FTP services.

Environment Variables

SOCKETS uses environment variables to determine the location of necessary configuration files. They can be set, using DOS *SET* command, within the autoexec.bat file at startup or within a batch file prior to SOCKETS being loaded. They are:

- SOCKETS
- HTTP_DIR
- HOSTNAME
- FTPDIR

Examples

Set SOCKETS=C:\NETWORK

Set HTTP_DIR=C:\HTML

Set HOSTNAME=FTPDEMO

Set FTPDIR=C:\FTP

Remarks

The environment variable *SOCKETS* is used to indicate to the SOCKETS kernel where configuration files, license file for the demo version, and access rights files are located. For example "Set SOCKETS=C:\NETWORK" would cause SOCKETS to look for configuration files within the directory of C:\NETWORK.

The environment variable *HTTP_DIR* is used by the SOCKETS server and indicates the location of html files if the files are stored in a separate directory from the server executable. To continue with the previous example if the server were in the directory of C:\NETWORK and the html files were in C:\HTML then the environment variable would read "SET HTTP_DIR=C:\HTML".

The environment variable *HOSTNAME* is used by SOCKETS to indicate the banner name displayed during an FTP session.

The environment variable *FTPDIR* is used by SOCKETS to indicate the location of the temporary file created by the SOCKETS server during an FTP session. Please refer to Chapter 7 for more details.

File Selection

What you wish to do with the SOCKETS TCP stack will dictate which files are to be transferred to the target hardware. At a minimum your target hardware will require the TCP stack executable:

- SOCKETM.EXE for serial / ppp connection
- SOCKETP.EXE for an Ethernet connection
- SOCKET.CFG for both serial and ethernet use
- MODEM.MCF for a serial / ppp connection

If you wish to use the Datalight Web Server with SOCKETS you will need the HTTPD.exe file located within DL\SOCKETS\ \SERVER. You will also need an index.htm file that dictates what information the web server displays. One has been provided within the same directory.

Full explanations of the files are available in"** Chapter 7, SOCKETS Server **"on page 81.

Configuration

The configuration of SOCKETS is determined by what you would like to do with the TCP/IP stack. Some examples are provided within "**Chapter 5, SOCKETS Configuration Examples**" of the types of configurations easily accomplished with SOCKETS. For quick testing of SOCKETS please reference the sample configuration files supplied within the **\CONFIGS** directory. Their explanations are found in "**Chapter 5, SOCKETS Configuration Examples**".

To make your own configuration, run the supplied file sconfig.exe to configure SOCKETS for your target hardware. SCONFIG.EXE is a text-based tool that will prompt for various settings necessary to customize SOCKETS. At completion sconfig.exe creates the file SOCKET.CFG that must accompany SOCKETS to the target hardware. Running SCONFIG may take place on either your target hardware or your development system. If you wish to configure the stack after transferring files to your target hardware, please include the file SCONFIG.EXE in your list of files to be transferred to your target hardware.

Transfer

Transfer the selected files onto the target system into a \DL\SOCKETS\ directory. As Datalight ROM-DOS has various methods of serial port transfers available, please refer to the ROM-DOS manual for specific details.

Testing

If you are using an Ethernet TCP stack you must first load a packet driver specific to your NIC. The packet driver provides a software interrupt to allow communication between the NIC and SOCKETS (default is 0x60). If you do not have a packet driver or are unsure of its use please contact Datalight Technical Support. Although Datalight does not manufacture packet drivers our support team is knowledgeable about packet driver implementation.

Type SOCKETM.EXE or SOCKETP.EXE to launch SOCKETS. The various options for launching SOCKETS are:

/n=normal_sockets

/i=capi_interrupt

/d=dos_compatible_sockets

/m=memory_size

/p=print_delay

/s=stack_size

/v=interrupt_vector

/q

/0

/u

Once SOCKETS has been loaded correctly, the most basic test to ensure that everything is working correctly is to “XPING” a known server or gateway that the SOCKETS machine is connected to. If the ping is returned everything is working, if the ping fails please refer to the troubleshooting section of the SOCKETS manual.

Chapter 3, SOCKETS Configuration

Configuring SOCKETS consists of setting up environment variables, loading required drivers, creating configuration files and running the appropriate SOCKETS kernel with the correct command line parameters.

The SOCKETS environment variables are discussed in **Chapter 2, SOCKETS Installation**.

Packet Driver

When SOCKETS is required to interface to a Local Area Network or any special network, one or more Packet Drivers must first be loaded. A Packet Driver presents a standard interface to underlying network hardware and is normally supplied by the vendor of the NIC used to connect to the physical network.

The following packet-driver classes are supported:

- Class 1 DIX Ethernet
- Class 3 Token Ring
- Class 6 SLIP
- Class 11 IEEE Ethernet

Most Ethernet packet drivers support both classes 1 and 11. As the default, Class 1 should normally be used.

Each Packet Driver requires a software interrupt vector through which it is accessed. For a single Packet Driver, Interrupt 0x60 or 0x69 is normally used. Avoid using Interrupt 0x61 and 0x7f as those are the defaults used by the SOCKETS APIs. Interupt 0x62 is the default used by the SOCKETS FTP API and 0x63 that of the SOCKETS HTTP Extension CGI API.

For specific information on loading packet drivers, refer to the documentation of the specific driver you are using.

Example:

```
rtspkt 0x60
```

When only one Packet Driver is used, the SOCKETS kernel will search for the interrupt vector when the interface command in the configuration file specifies that a Packet Driver must be used. The interrupt can also be explicitly specified and must be so specified when more than one Packet Driver are used.

Serial Operation

Hardware considerations

For asyncronous serial operation SOCKETS supports the standard PC COM ports without an external driver. Up to six COM ports with or without interrupt sharing may be used. SOCKETS

checks for and uses FIFO buffered 16550 UARTs for faster throughput. It is strongly recommended that buffered UARTs be used for high-speed applications.

The interface command in the configuration file specifies the I/O address, hardware interrupt and speed to be used and parameter commands can be used to specify character data format and flow control. Flow control may be performed by means of modem control signals (hardware flow control) or XON/XOFF control characters (software flow control). Software flow control is only possible when PPP is used. It will not function for SLIP or CSLIP.

PPP Funtionality

PPP can be set up for "client" and/or "server" operation. "Client" operation normally refers to a dial-out operation and "server" to a dial-in operation.

For "server" operation, log-in is controlled by a list of Username/Password pairs, each coupled to a remote IP address, which is assigned to the peer if requested to do so during the IPCP negotiation. As an option, a Username/Password pair can be flagged to provide Callback Verification (CBV). In this case, the call is terminated as soon as the PPP negotiations have been successfully completed. Then a reverse call is made by the server; using the number in the modem command file if a modem is involved. During the first subsequent set of PPP negotiations, the CBV flag is ignored to prevent another callback. When the PPP session terminates, the CBV flag is again enabled.

Another PPP feature is the ability to specify that a PPP session should start immediately when SOCKETM is loaded, or to delay that until traffic is generated. It is also possible to specify two sets of PPP parameters per interface. One set of parameters is for outgoing connections; the other set is for incoming connections.

Modem Operation

When using serial operations, modem dial-in and dial-out, may optionally be used. Simple modem scripting commands contained in modem configuration files may be used to establish modem connections. Commands in the modem file can specify that a dial-up connection will always be dialed whenever DCD is not detected, or specify that dial on demand is disabled.

In order to work properly, the DCD modem signal must be supported by the hardware and the modem must drop a connection when the DTR modem signal is lowered. The modem must also support a command set like the AT command set. If the hardware does not comply with these requirements, a modem may still be used. In this case the interface will be specified as not having a modem and a user program must perform the modem connect/disconnect functions.

SOCKETS Configuration Files: SOCKET.CFG, HOSTS

SOCKETS uses two files in the \DL\SOCKETS directory (default), or any other directory specified by the SOCKETS environment variable. These files are SOCKET.CFG, the default start-up file, and HOSTS, the host names file. If not found, SOCKETS uses the default SOCKET.CFG in the \DL\SOCKETS directory.

SOCKET.CFG is a text file containing configuration commands. Empty lines and lines starting with # are ignored. Commands are used to specify protocol parameters like the IP address of the

stack, interface parameters like Packet Driver or Asyncronous Serial lines, routes and various other parameters. Here is a simple example:

```
ip address demo
   Set the IP address of this host to 192.6.1.1.
interface pdr if0 dix 1500 5
   Use Packet Driver, naming the interface 'if0', MTU=1500, Receive
   buffers = 5
route add default if0 router
   Route all traffic to unkown destinations via 'if0' using 'router'
   as a gateway
tcp mss 1460
   TCP Maximum Segment Size = 1460.
tcp window 2920
   TCP Maximum window = 2920.
start prntserv
   Start printer server on PRN using default port of 10.
```

HOSTS is an optional file containing mappings of IP addresses in dotted decimal notation to names.

Sample HOSTS file:

```
192.6.1.1     demo
192.6.1.2     router
192.6.1.3     server
```

SCONFIG Overview

SCONFIG is a SOCKETS configuration utility. It supports only the barest configurations: Ethernet over a packet driver and PPP over a serial modem. Just answer the multiple choice questions, and then fill in a few data fields and **SCONFIG** writes simple *SOCKET.CFG* and *MODEM.MCF* files for you.

SOCKET.CFG Samples

The following configuration file contains the minimum possible commands for a valid configuration file: just one. This is to specify that the interface should use a Packet Driver, the interrupt vector which must be searched for. It should use DIX encapsulation, have an MTU of 1500 and have a maximum of 5 receive buffers. Since no IP address is specified, BOOTP will be used and the required operating parameters will be retrieved from a BOOTP server, which must be available on the network.

SOCKET.CFG:

```
interface pdr if0 dix 1500 5
```

The following is a more typical example specifying a static IP address, a Packet Driver interface, a default route, the TCP MSS and WINDOW, and the SOCKET PRINT server to be started.

SOCKET.CFG:

```
# Sample configuration file
ip address 192.6.1.1
interface pdr if0 dix 1500 5
route add default if0 192.6.1.2
tcp mss 1460
tcp window 2920
start prntserv
```

The next example is a configuration file for running PPP over a direct serial link (no modem). It uses COM1 at the default I/O address of 0x3f8 using interrupt level 4 at 9600 baud with no flow control. The local IP address in this case will be obtained from the PPP peer. No compression of PPP headers and no TCP/IP header compression will be negotiated. No authentication will be done.

SOCKET.CFG:

```
iface asy p0 ppp 1500 30 0x3f8 4 9600
route add default p0
par p0 ipcp local address 0.0.0.0
par p0 lcp start
par p0 ipcp open
```

The following example is a a more typical configuration file for running PPP over a direct serial link with a modem. It uses COM1 at the default I/O address of 0x3f8 using interrupt level 4 at 28800 baud with Xon/Xoff flow control. The local IP address in this case will be obtained from the PPP peer. Compression of PPP headers and TCP/IP header compression will be negotiated. PAP authentication will be negotiated and “PppUser” as a username and “PppPasword” as a password will be used for authentication. Modem commands will be retrieved from the MODEM.MCF file.

SOCKET.CFG:

```
iface asy p0 ppp 1500 30 0x3f8 4 28800 modem.mcf
param p0 288000 x x
route add default p0
par p0 lcp local accm 0x0a00000
par p0 lcp local acfc on
par p0 lcp local pfc on
par p0 lcp local magic on
par p0 pap user PppUser PppPassword
par p0 ipcp local compress tcp 16 1
par p0 ipcp local address 0.0.0.0
par p0 lcp start
par p0 ipcp open
```

SOCKETP, SOCKETM Overview

Two versions of the SOCKETS kernel are available: SOCKETM.EXE provides support for serial connections as well as Packet Driver connections. SOCKETP.EXE only provides support for Packet Drivers and has a smaller memory footprint than SOCKETM.EXE. SOCKETM or SOCKETP runs as a Terminate and Stay Resident (TSR) program.

Running SOCKETP or SOCKETM without any parameters will be sufficient in simple configurations and will use the default %SOCKETS%\SOCKET.CFG and %SOCKETS%\HOSTS configuration files. Parameters are used to specify another configuration file and to tailor the facilities like memory and stack usage, number of simultaneous connections and the API interrupt vectors to use. The /U parameter may be used to unload an already resident SOCKETS kernel.

XPING

The **XPING.EXE** (external ping) program gives a quick method to test your SOCKETS installation. The source code is also supplied as an example. XPING starts a continuous string of pings until stopped by a keystroke.

Syntax

XPING IP_address [interval]

Where the *IP_address* may be a numeric or symbolic address and the *interval* is the time to wait between pings in timer clock ticks. The default is 10 ticks.

SOCKETS COMMAND SUMMARY

Command	Description	Location
ARPSTAT	Displays information about the current ARP table and allows entries to be removed.	Page 63
DHCPSTAT	Displays status about the DHCP settings and allows a DHCP lease to be renewed.	Page 63
FTP	File Transfer Protocol (FTP) client application.	Page 64
FTPD	A server providing FTP services.	Page 87
GETMAIL	A POP3 mail client.	Page 67
HTTPD	A server providing HTTP services.	Page 81
HTTPFTPD	A server providing both HTTP and FTP services.	Page 89
HTTPGET	An application that can retrieve files from an HTTP server.	Page 67
IFSTAT	Displays information about configured SOCKETS interfaces.	Page 68
IOCTL	A diagnostic utility for testing the SOCKETS TCP/IP Basic API IOCTL functions.	Page 68
IOCTLH	A diagnostic utility for testing the SOCKETS TCP/IP Basic API IOCTL Hotswap functions for wireless or removable Ethernet cards.	Page 69
IPSTAT	Displays information about SOCKETS memory usage and IP interface.	Page 69
LPR	A printer client for UNIX style print servers.	Page 70
MAKEMAIL	Prepares text and data for sending through **SENDMAIL**.	Page 71

Command	Description	Location
NETBIOS	A TSR that provides a NETBIOS compatible network interface.	Page 72
PDTEST	A diagnostic program that tests loaded packet drivers.	Page 73
RC.JAR	A Java remote console applet for connecting to hosts running **HTTPD.**	Page 73
RC_DK.JAR	A version of RC.JAR with Danish keyboard support.	Page 74
RCCLI	A DOS remote console client for connecting to hosts running **HTTPD.**	Page 74
SCONFIG	A SOCKETS configuration utility.	Page 9
SENDMAIL	delivers e-mail messages packaged by **MAKEMAIL** to an Internet mail server.	Page 75
SETHOST	Sets an environment variable to the name of the SOCKETS client based upon the client's MAC address.	Page 75
SNTPCLI	Used for getting the system network time information.	Page 75
SPRINT	A simple printer client that prints through a SOCKETS host configured as a print server.	Page 78
TCP	a utility used to examine and change the TCP parameters.	Page 79
XPING	A utility to send ICMP messages continuously to a specified host.	Page 79

Chapter 4, SOCKETS Configuration Commands

Commands

Overview

This section describes the SOCKETS configuration commands available when starting and running SOCKETS. To execute any of the commands described in the following sections, insert the appropriate entries in the SOCKET.CFG or another configuration file.

Notations and Conventions

In the command summary, use is made of the *hostid* notation, which denotes a host, router, gateway, or network. *hostid* may be specified either by a symbolic name listed in the HOSTS file, or a numeric IP address with decimal notation; for example, 192.10.240.1.

The following conventions apply to command syntax.

- *italics* indicate that the term is a parameter to be specified by the user.
- [ ] Square brackets indicate that the enclosed item is optional.
- / A forward slash is used as a leading character for an optional switch in some commands. Both the forward slash and the switch that follows it form part of the command syntax. The equal sign before extra parameters is optional in most cases.
- | The vertical bar indicates that there is a choice between two or more selections, but that only one of the options indicated may be specified.
- **bold** type indicates a reserved key word as part of the command syntax and is to be typed exactly as indicated.
- # Commands proceeded by a hash sign (#) are ignored. They are used for comments in configuration files.

Command Reference

---

arp

arp adds a new entry to the Address Resolution Cache. **arp** can also specify the type of ARP used to determine IP address conflicts.

Syntax

arp add hostid ether | ieee hw_addr
arp init {DHCP | GRAT | BOTH}

Remarks

'arp add' includes a new entry in the Address Resolution Cache. Do not add entries with duplicate IP or hardware addresses as this will cause malfunctioning of the network.

'arp init' specifies the type of ARP used to determine IP address conflicts. By default both DHCP and Gratuitous ARPs are sent to determine if the SOCKETS IP address conflicts with any other IP address.

The DHCP and Gratuitous ARP types are slightly different methods of determining IP address conflicts, but each causes a different set of events to occur. In both cases, if a machine is currently using the IP address specified in the ARP packet, that machine will respond to the ARP request which sigals an IP address conflict.

DHCP ARP was designed to be used in conjunction with an IP address received from a DHCP server. A machine could lose it's lease an IP address, but continue to use that IP address. As a result, the DHCP ARP is used to make sure that another machine is not urrently using the specified IP address. Fortunately, the DHCP ARP can also be used with static IP addresses. A difficulty arises when a Proxy Server receives a DHCP ARP packet, the server responds on behalf of the IP address.

A Proxy Server responds to ARP requests for a set of machines that may not always be available for network traffic. If a machine using that IP address is being served by a Proxy Server, even if the DHCP ARP came from a machine being served by a Proxy Server, it responds to the DHCP ARP packet which signals an IP address conflict.

The Gratuitious ARP can also be used to determine IP address conflicts, but it has one major side effect. When a machine receives a Gratuitous ARP packet, and has the requested IP address in it's ARP table, that machine will update it's ARP table to match the hardware address contained in the Gratuitous ARP packet. If another machine is currently using the requested IP address, then that machine can no longer communicate with the network until it sends out another Gratuitous ARP.

By specifying BOTH, a DHCP ARP is broadcast to the network, and if no other machine responds, then a Gratuitous ARP is broadcase to the network. The followup Gratuitous ARP causes APR tables with the requested IP address to be updated with the MAC address with the Gratuitous ARP.

Options

hostid

The IP address of a remote host that is to be added to the ARP cache. This value may be a symbolic name from the HOSTS file or a decimal (dotted) address.

hw_addr

Used with *add* it denotes the hardware (node) address of the remote host whose IP address is given in *hostid*. This must be a six-digit hexadecimal address separated by colons for Ethernet.

DHCP

Used to do only DHCP ARP to determine IP address conflicts.

GRAT

Used to do only Gratuitous ARP to determine IP address conflicts.

BOTH

Used to do DHCP and Gratuitous ARP to determine IP address conflicts. This is the default action.

Example Commands

```
arp add  unix_host  ether  00:00:65:0D:E6:04
arp add  127.0.1.3  ether  00:00:65:0D:E6:04
arp init grat
```

domain

If a host name is not a decimal (dotted) address and it is not found in the HOSTS file and at least one Domain Name Server has been defined, an attempt is made to obtain the address from the defined DNS server(s). The number of times any server is polled (retries), in addition to the time to wait for a response, can also be specified. A suffix may be specified and is attached to all names not containing any dots.

All of the following sub-commands can be issued without the optional parameters to obtain information on the current status.

Syntax

domain server [host_name]

domain retry [retry_count]

domain time [wait_time]

domain suffix [domain]

Remarks

domain server adds a DNS address or lists the current servers if host_name not specified.

domain retry specifies the retry count for polling each server. domain retry lists the retry count if retry_count not specified.

domain time specifies the time (milliseconds) to wait for a response before attempting retry. domain time lists the time (milliseconds) to wait if wait_time not specified.

domain suffix specifies the domain suffix to add to all simple names; names that contains no dots. domain suffix lists the domain suffix if domain is not specified.

Example Commands

```
domain retry 3
domain server 196.2.1.1
domain suffix myorg.co.za
domain time 2000
```

iface

iface is a synonym for the **interface command**.

interface

interface informs SOCKETS of the hardware or software communications interface(s) to be used at the network interface level. At least one network interface is required, and two or more are used in gateway (router) applications.

The class, or mode, of each interface defines the encapsulation used for packaging the data frame into the transport frame. Some types of interface support only one class.

When SOCKETS is defined with multiple interfaces, you first declare an IP address which is associated with the immediately following interface. The defined net mask is then used to add a route through the interface to the connected network. Using the same IP address would result in multiple routes to the same network. The default route is set on the first interface with an IP address with a zero net mask (for example, IP address 19.63.10.11/0).

Each interface command uses the IP address from the last supplied IP address command.

Syntax (general)

interface type name class other parameters

Syntax (specific)

interface pdr name dix mtu numbuf [intvec [irq]]

interface asy name [slip | cslip | ppp] mtu buflim ioaddr iovec speed [modemfile]

interface mdd name [slip | cslip] mtu buflim ioaddr iovec speed [modemfile]

interface aslink name [slip | cslip] mtu buflim ioaddr iovec speed [modemfile]

Options

type

type defines the type of hardware or software interface.

interface supports the following software interfaces.

Interface	Description
asy	Standard PC asynchronous interface (RS232 port)
pdr	packet driver interface

name

name defines the name by which the interface is known on the local host. *name* is a symbolic name known only to the local host on which it is used.

name may be arbitrarily assigned. Each interface command on the same host must have a unique name assigned. This name is used by other commands referring to this interface, e.g. the **param** command.

class

class specifies how IP datagrams are to be encapsulated in the link level protocol of the interface. Some interfaces offer a choice between classes while others use a fixed class. The following classes are available and are listed with their associated types.

Type	**Class** (defined in the following list)
pdr	dix, ieee, token, driver, slip

Type	**Class** (defined in the following list)
asy	slip, cslip, ppp
mdd	slip, cslip
aslink	slip, cslip

Class	**Description**
dix	The DEC/Intel/Xerox Ethernet interface also known as Blue Book Ethernet or Ethernet II.
token	IBM Token Ring. Source routing is supported for multiple rings.
ieee	IEEE: 802.3 Ethernet with SNAP headers.
driver	Use the default class for the packet driver.
slip	Serial Link Internet Protocol (SLIP) for point-to-point asynchronous links. This mode is compatible with UNIX SLIP.
cslip	Compressed Serial Link Internet Protocol (SLIP) for faster reaction over point-to-point synchronous links.
ppp	Point-to-point protocol over asynchronous links.

mtu

mtu specifies the Maximum Transmission Unit size, in bytes. Datagrams larger than this limit are fragmented into smaller pieces at the IP layer. The maximum value of *mtu* for the various interfaces is:

Ethernet - 1500

For serial links a standard value for *mtu* is 576. (576 is the maximum according to specifications, but may be increased on reliable connections as long as both sides use the same value.)

numbuf

numbuf specifies how many incoming datagrams may be queued on the receive queue at one time. If this limit is exceeded, further received datagrams are discarded. This mechanism is used to prevent fast interfaces from filling up memory when data cannot be handled fast enough.

buflim

buflim specifies the maximum number of outgoing datagrams or packets to queue before starting to discard datagrams. This mechanism is used to prevent the memory from filling up when a serial link goes down.

bufsize

bufsize specifies the size of the ring buffer in bytes to be allocated to the receiver in raw mode

intvec

intvec specifies the software interrupt number (vector) in hexadecimal to use for resident packet drivers.

ioaddr

ioaddr is the I/O base address in hexadecimal of a serial port or the hardware controller and must correspond with the jumper or switch settings used during the setup of the controller board. The standard values for serial ports are:

COM1 03F8h

COM2	02F8h
COM3	03E8h
COM4	02E8h

iovec

iovec is the hardware interrupt vector used by the serial port or controller and must correspond with the jumper or switch settings used during setup of the controller. The standard values for serial ports are:

COM1	4
COM2	3
COM3	4
COM4	3

irq

irq is the hardware interrupt vector used by the network interface controller. This is only used for faster response in SOCKETS.

modemfile

A file containing the modem commands and scripts.

speed

speed specifies the transmission speed for serial interface devices (baud rate). Before using a serial connection you have to set flow control with the **par** command.

Examples

```
interface pdr if0 dix 1500 5 0x60
interface asy ser0 cslip 576 15 0x3f8 4 9600
interface asy p0 ppp 576 5 0x3f8 4 9600 pppmod.mcf
```

IP

IP displays or sets the values of the options selected when defining the IP (Internet Protocol) host address of the next interface to be defined.

Syntax

IP address [hostid [/net_bits]]

IP status

IP ttl [number]

Remarks

IP address sets the IP host address of the next interface to be defined. A route is automatically added to each interface for the default or specified net mask for its address. To make an automatic route the default, specify the net bits as zero. When specified without the optional parameters, IP address displays the current value(s) of the local host IP address(es). To assign different IP addresses to different interfaces on the same host, an IP address statement must precede each interface definition. The last IP address given is used in case of missing IP address statements.

IP status displays Internet Protocol (IP) statistics, such as total packet counts and error counters of various types. It also displays statistics on the Internet Control Message

Protocol (ICMP). This includes the number of ICMP messages of each type sent or received.

IP ttl sets the default time-to-live value which is placed in each outgoing IP datagram. The ttl value limits the number of gateway hops the datagram is allowed to take in order to kill datagrams that got stuck in loops.

Options

hostid

hostid specifies the IP host address to assign to the next interface to be defined. This may be a symbolic name from the HOSTS file, or a dotted decimal address.

/net_bits

A net mask can be specified for the host. In the **IP address** command an optional */net_bits* can be used to indicate the number of bits in the network ID. The net mask is used to determine whether an incoming datagram is a broadcast and also for sending UDP broadcasts.

Net masks are more easily represented in binary or hexadecimal format. For example, the IP address 128.1.1.5/24 corresponds to a net mask of 255.255.255.0 (FFFFFF00h), 25 bits to 255.255.255.128 (FFFFFF80h) and 26 bits to 255.255.255.192 (FFF FFC0h).

The default net mask used corresponds to the class of address used if not explicitly specified.

Net Bits	Net Mask	Class	IP address range
8	255.0.0.0	A	0.x.x.x to 127.x.x.x
16	255.255.0.0	B	128.x.x.x to 191.x.x.x
24	255.255.255.0	C and higher	192.x.x.x or higher

If you want to subdivide your network, you can divide it by two for every net bit added. The following table provides information on converting between net bits and net mask. The number of net bits to add when changing a 0 in the net mask to:

Net Bits	Net Mask	Net Bits	Net Mask
1	128	5	248
2	192	6	252
3	224	7	254
4	240	8	255

number

When *number* is omitted, **IP ttl** displays the current value of the time to live parameter.

par

par invokes a device-specific control routine. **par** operates differently for each interface type and even interface mode.

Syntax

par ifname [arg1...argn]

Options

ifname

ifname defines the name used in the interface command for the device to be controlled.

arg1...argn

These parameters depend on the type of interface in use.

Example

To change the baud rate of a serial interface and specify software flow control, use:

```
par sl0 19200 x x
```

par, Configuring the PPP Interface

The PPP interface is optionally configured by using the **par** command. Various parameters at the LCP, IPCP and PAP levels can be specified.

Setting PPP Options, Local and Remote LCP/IPCP

When a local option is specified, the value of the option is used in the initial Configuration Request to the peer. Options not specified, are not be requested. For each option, the 'allow off' parameter disallows the peer to include that option in its response. By default, all options are allowed in the response, even if the option is not included in the request.

When a remote option is specified, the value of the option is used in the initial response to the configuration request from the peer. If an option is disallowed, it does not allow the remote to specify that option in its request. By default all options are allowed.

Local and remote options are specified by:

par *iface* lcp|ipcp local|remote *option* [*parameters* ...] [allow [on|off]]

The **par** command options are as follows:

Syntax	Description
par *iface* lcp|lcpin local|remote accm *bitmap*	Set the Asynch Control Character Map. The default is 0xffffffff.
par *iface* lcplcpin local|remote authent [pap|chap|none|allow [on|off]]	Set the Authentication protocol. The default is none.
par *iface* lcp|lcpin local|remote acfc [on|off|allow [on|off]]	Set Address and Control Field Compression. The default is off.
par *iface* lcp|lcpin local|remote pfc [on|off|allow [on|off]]	Set Protocol Field Compression. The default is off.
par *iface* lcp|lcpin local|remote magic [on|off|*value*|allow [on|off]]	Set the Magic number option (detects looped back circuits)

Syntax	Description
par *iface* lcp|lcpin local|remote mru [*size*|allow [on|off]]	Set the Maximum Receive Unit. The default is 1500.
par *iface* lcp|lcpin [open|start|listen]	Set the LCP connection mode. *start* is used to initiate a connection as soon as the stack has been loaded. If a modem is used, this will also initiate an immediate dial-out. *open* is used to delay a connection until data needs to be sent. If a modem is used, this provides a dial-on-demand facility. *listen* is used when the peer must initiate the connection. Note that two commands can be given to specify both *open* and *listen* in which case a connection can be initiated from either side.
par *iface* ipcp|ipcpin local|remote adress [*ip_address*>|allow [on|off]]	Set the IP address option. If set to 0.0.0.0, the peer must supply it.
par *iface* ipcp|ipcpin local|remote compress [[tcp *slots* [*flag*]] | [allow [on|off]]]	Set the TCP/IP header compression. *slots* should be equal to the maximum number of concurrent TCP connections. Low values preserve memory. 4 - 16 are good values. *flag* = 0 to not compress the slot number; and =1 to compress. The default is 1.

Setting PPP Options, Retry Counters

The following retry counters can be set:

par <iface> lcp|lcpin retry <configure>|<failure>|<terminate> <count>

par <iface> ipcp|ipcpin retry <configure>|<failure>|<terminate> <count>

iface is the name assigned to the interface.

configure is the number of configuration requests (default 20).

failure is the number of bad configuration requests allowed from peer (10).

terminate is the number of termination requests before shutdown (2).

count is the number of retries.

Setting PPP Options, Timeout Values - in milliseconds

par *iface* lcp|lcpin|ipcp|ipcpin|ap|apin timeout *milliseconds*

Setting PPP Options, Authentication - username/password

For PAP or CHAP authentication on PPP connections:

par iface ap user username password

Setting PPP Options, Open - a specified layer

par *iface* lcp|lcpin|ipcp|ipcpin open

Start – an lcp connection

param iface lcp|lcpin start

When **lcp** is specified, a PPP connection is immediately initiated.

Listen – for an lcp connection

param iface lcp|lcpin listen

When **lcpin** is specified, a PPP connection is delayed until an incoming call is detected.

par, Alternative Routing Control Sub-commands

The Alternative Routing Control Sub-commands set up and check the SOCKETS alternative route mechanism. More than one route can be specified to a target host or network. The first route that has an associated interface in the up state is used.

An interface is in the up state when it is defined by the interface command. It enters the query state when it does not receive valid input within a specified up-time period after data expecting a response is sent.. At this stage three (catering for links with a high data loss) ICMP echo requests (ping) are sent to a query IP address. It enters the down state by a SOCKETS command or when it does not receive valid input within the specified up-time period after entering the query state. If an up time has never been specified or a value of 0 is specified, the interface will stay in the up state whether valid input is received or not.

An interface enters the up state by a SOCKETS command or when valid input is received on that interface while it is in the down or query states. An ICMP echo request is sent on an interface in the down state every downtime period. If a downtime has never been specified or a value of 0 is specified, the ICMP echo request is not sent. Up time and downtime is specified in seconds.

Syntax

par ifname [uptime | downtime] time

par ifname query hostname

Options

Ifname

Ifname is the interface name of an asy interface.

Uptime

Uptime is the time to allow for no response on a defined connection.

Downtime

Downtime is the period of time to retry a defined connection.

Example Alternative Routing

Two SLIP interfaces are used to get to the target network 192.6.1.0. The first interface, named if0 should preferably be used, but if it stops receiving for a period of 20 seconds,

it should try to ping 192.6.1.2 and if no response is received within another 20 seconds, if1 should take over, but if0 should be tried every five seconds. Interface if1 should disconnect after 80 seconds of no traffic.

The SOCKET.CFG file should contain the following:

```
interface asy if0 slip ...  ...  ...
par if0 uptime 20
par if0 downtime 5
par if0 query 192.6.1.2
interface asy if1 slip ...  ...  ...
par if1 uptime 80
par if1 downtime 5
par if1 query 192.6.1.2
route add 192.6.1.0 if0
route add 192.6.1.0 if1
```

In the case of both if0 and if1 failing, both will retry every five seconds until one comes up. The return paths should also be maintained in a similar way with SOCKETS or by using RIP.

par, COM Port Speed and Flow Control Sub-command

This par sub-command allows the baud rate, flow control and data parameters to be set and is only to be used on asy type interfaces.

Syntax

par ifname speed outflow inflow bits

Options

ifname

ifname is the interface name of an asy interface.

speed

speed sets the baud rate for a serial link. The standard speeds are: 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 and 57600.

outflow

outflow sets the output flow control for a serial link. When not supplied it defaults to none.

inflow

inflow sets the input flow control for a serial link. When not supplied it defaults to none. The following list shows *outflow* and *inflow* selections.

none no flow control

Xon	Xmit on/Xmit off (Ctrl-Q, Ctrl-S)
DCD	Data Carrier Detect modem signal
CTS	Clear To Send modem signal
DSR	Data Set Ready modem signal
DTR	Data Terminal Ready modem signal
RTS	Request To Send modem signal
ixxx	inverse of modem signal xxx

Invalid selections are ignored or replaced by more logical selections. The usage of Xon/Xoff is not recommended for most applications. The ixxx support is for non-standard equipment that uses a reversed signal on the required pin.

bits

bits sets the number of data bits per character, number of stop bits, and parity for a serial link. It is a three-character string consisting of dsp from the following table. When not supplied, *bits* defaults to value of 81n.

Values for d	5, 6, 7, or 8 data bits
Values for s	1 or 2 stop bits
Values for p	o = odd parity e = even parity n = no parity m = mark s = space

Example

```
par  asy1  57600 cts rts 81n
par  slp0  4800 xon xon 72e
```

par, RIP Advertising Sub-command for Interfaces

When the RIP advertise command has been used, this par sub-command makes allowance to disable and re-enable RIP advertising on a specific interface.

Syntax

par ifname [ripadv | noripadv]

Options

Ifname

ifname is the interface name of an asy interface.

Ripadv

Ripadv indicates to use route advertising on the defined interface.

Noripadv

Noripadv indicates to not use route advertising on the defined interface.

Examples

```
par if0 noripadv
par if1 ripadv
```

printer

The **printer** redirector operates by intercepting Interrupt 17 and passing printer output to a local or remote print server. A print session is started when output is first sent to a specific port, and is stopped after a user-specified period of no output. When a printer port is redirected, the IP address

and TCP port number of the destination print server, in addition to the timeout period, is specified. A printer port can both be used as a print server port and as a redirected port.

Syntax

printer printer_port timeout IP_address [TCP_port]

Options

printer_port

Printer port 0, 1, 2 or 3 to redirect. Port 0 corresponds with PRN, 1 with LPT2 and so on. Serial printers can be defined with the SRPRINT.EXE TSR that forms part of the SOCKETS package.

timeout

Timeout period in seconds for closing the TCP connection.

IP_address

IP address of print server host (could be the local host).

TCP_port

TCP port of print server (Default is 10).

Comments

To ensure proper queuing of remote and local print sessions to a local printer, the printer port must be redirected to the local IP address and TCP port of the print server using that port.

Examples

The IP address of print_host in the following example is defined in the HOSTS file.

```
printer 0 15 print_host
```

The following example uses printer port 1 (LPT2) for both local and remote printing. (If the printer command is placed before start prntserv, SOCKETS will not work correctly.)

```
ip address this_host
 .
 .
start prntserv 1010 1
printer 1 15 this_host 1010
```

RIP

The Routing Information Protocol (RIP) allows a SOCKETS gateway to advertise routes and allows SOCKETS to recognize advertised routes from SOCKETS and other gateways. These two facilities can be individually selected.

Syntax

RIP advertise [time [1|2] [self]]

RIP use [time]

Options

time

time is the number of elapsed seconds before the advertisement is repeated when used in the advertise sub-command.

time is the period during which routes are added or amended as a result of RIP, and are valid for *time* seconds. Such added or amended routes are dropped if another RIP advertisement is not received within *time.*

self

self advertises SOCKETS connection to the network.

RIP Advertise Sub-command

RIP advertise causes an immediate advertisement of all relevant routes and repeats it every time seconds. The default value for time is 30 seconds. RIP version 2 advertisements are sent by default. To send only version 1 advertisements, add a 1 on the command line.

When RIP advertise is selected, all interfaces advertise all routes except those routes making use of that specific interface (split horizon) and routes marked Private. (To prevent certain interfaces from using RIP, see the parameter command.) A route which is dropped as a result of a RIP update or which becomes unavailable as a result of its associated interface going into the down state, is immediately advertised as being infinite (metric = 16) and is not advertised until it becomes available again. The advertisement is sent as a sub-net broadcast using the net mask and IP address of the interface.

Example

```
RIP advertise 30 self
```

RIP use, Update RIP Sub-command

RIP use causes a RIP request for routes and a continuous update of routes according to RIP advertisements. Routes added or amended as a result of RIP are valid for time seconds and are dropped if another RIP advertisement is not received within that time. The default value of time is 240 seconds.

When RIP use is selected, routes are updated according to received RIP advertisements. Routes added or amended as a result of RIP, have a timeout associated with them. If another RIP advertisement is not received during that time, the route is dropped. A route is also dropped if an advertisement of infinity (metric = 16) is received. To prevent dropping a route, it must be marked as Static.

On RIP and static routes: — The metric of a route marked static is never updated by a RIP advertisement. Instead a duplicate route is added before the static route. If the duplicate route is dropped as a result of a timeout or RIP, the static route is used again.

On RIP and mdd or x25 interfaces: — On mdd or x25 interfaces, a RIP route is only sent when the last datagram was not a RIP route. This is done to allow an interface to time out and be disconnected when not used.

Also, the route on the receiving end still times-out and becomes unavailable. This means that inactivity timers should be shorter than route timers; as specified in RIP use timer.

When an aslink to mdd association is broken, all non-static routes using the specific mdd are marked as having a hop-count of infinity; the metrics are set to 16.

route

route creates an entry in the IP routing table for SOCKETS to determine where to send data. The Alternative Routing mechanism allows more than one route to be specified to a particular host or network. Failure of one route causes an automatic switch to the next route.

Refer also to the **ip** address command for specifying the net mask, because a route is automatically added to each interface for the default or specified net mask for that address. When multiple routes are defined to the same address, SOCKETS uses the route with the network size (largest number of bits in the net mask).

Syntax (general)

route [add | drop destination ifname [gateid |none [metric [proxy] [private] [static]]]]

Syntax (specific)

route

route add [hostid | netid] ifname [gateid]

route add [hostid | netid[/mask]] ifname [gateid]

route add default ifname

route drop [hostid | netid]

route drop [hostid | netid[/mask]]

route drop default

Options

add or drop

Sub-command to add or drop (remove) a route from the routing table.

default

All transmissions to IP addresses not otherwise defined in routing commands are sent via the network interface specified by *ifname.*

hostid

hostid is the IP address of a destination remote host to which data must be sent, or a remote host that must be removed from the routing table (dropped).

netid

netid is the IP address of a destination network to which data must be sent. Any host with this IP network address is able to receive the data. Whether a particular host will use the data depends on the host portion of the specific IP address in the IP header.

mask

mask specifies the number of bits in the network portion of the address if sub-netting is used. If not used, the network portion of the address is determined according to the class (A, B or C) of the address.

ifname

ifname defines the name used in the interface command for the immediate network on which the data for the designated host must be sent. This is the network level interface to be used by the local host to reach the remote host.

gateid

gateid parameter specifies the IP address of a host, on the same physical network as the local host, which is used as a gateway or router to a different network. The gateway or router host specified in *gateid* must be directly reachable on the same physical network as the local host defining this gateway. In other words, this must be the nearest gateway to this local host.

metric

When using RIP or Proxy ARP a value from 0 to 16 for *metric* must be specified indicating the distance or cost of that route. A *metric* of 16 indicates that the route is down.

proxy, private and static

To support the Routing Information Protocol (RIP) the route command utilizes the proxy, private and static key words. These words can be used in any order following *metric*.

Proxy ARP should be used with care and not in conjunction with RIP. When more than one host responds to an ARP request, it can cause confusion and even lead to system crashes. This is possible in situations where more than one gateway implements Proxy ARP to a common destination. See also ROM-DOS Developers' Guide, "About TCP/IP" for more information.

When "RIP advertising" is selected, all interfaces advertise all routes except those routes making use of that specific interface (split horizon) and routes marked private. A route which is dropped as a result of a RIP update or which becomes unavailable as a result if its associated interface going into the down state, is immediately advertised as being infinite (metric = 16) and is not advertised until it becomes available again. In order for an interface to be used for advertising, a route without a gateway using that interface must be available. The advertisement is sent as a sub-net broadcast using the net mask of the host and the IP address of the interface.

When "RIP using" is selected, routes are updated according to received RIP advertisements. Routes added or amended as a result of RIP, have a timeout associated with them. If another RIP advertisement is not received during that time, the route is dropped. A route is also dropped if an advertisement of infinity (metric = 16) is received. To prevent dropping a route, it must be marked as static. The metric of a route marked static is never updated by a RIP advertisement. Instead a duplicate route is added before the static route. If the duplicate route is dropped as a result of a timeout or RIP, the static route is used again.

Examples

```
route add default ipx0
route add unix_net eth0
route add unix_host ipx1 unx_gate
route add unix_net2 eth0 /eth 1
route add unix_ net ipx0 unx_ gate
route add subnet/26 eth0 sub_gw
route drop unix_net
```

route can specify a Proxy ARP on a route, as follows:

route add net interface gateway metric [proxy]

When using Proxy ARP, gateway and metric must be specified. If no gateway is used, none can be specified. For example:

```
route add 192.6.1.0 ifx25 none 5 proxy
```

start

start starts a SOCKETS Print server (prntserv) or LPD and makes it available for access by remote hosts .

Syntax

start prntserv [port [printer_number]] [/m=mask] [/s=status] [/r=report] [/d=print_delay]

start lpd [printer_number] /n=printer_name [/m=mask] [/s=status] [/r=report] [/d=print_delay] [/f]

Options

port

port is the TCP port to use (the default is 10)

printer_number

printer_number is a value of 0, 1, or 2 to correspond with LPT1(PRN), LPT2 and LPT3. Any number up to 3 may be defined by the serial printer TSR (SRPRINT) as a printer on a COM port. The default printer number is 0.

Note: Both **LPD** and **prntserv** may use the same printer.

printer_name

The name of the printer queue.

mask

Used to activate status reporting and cope with non-standard printer status reporting.

status

Used to activate status reporting and cope with non-standard printer status reporting.

report

Used to activate status reporting and cope with non-standard printer status reporting.

print_delay

A decimal number specifying the number of times a printer will be tested for busy status before the busy status is accepted. This solves the problem that a single character is sent to the printer every timer tick (55 milliseconds), resulting in extremely slow printing by the print server. *print_delay* should be the lowest value still permitting fast printing. The default value is 40 which works fine on a 33 Mhz 386 CPU. On a slower CPU the value can be less and on a faster CPU it may need to be more.

/f

Send a form-feed (new page) to the printer at the end of the print job.

Printer Status Reporting

Optional status reporting has been implemented to give print client implementations more control over print jobs. The following print server enhancements have been enabled:

1. Optional status reporting to Print Client. The status reporting includes messages like:
   - Timeout
   - Last character has been sent to printer
   - I/O Error
   - Printer selected
   - Out of paper
   - Acknowledged
   - Printer Ready
2. Customization of printer status checking.
3. Print queue flushing.

Raw printer status bytes as defined for BIOS INT 17 services can be sent from the server to the client on the established connection when:

1. The first data is being printed.
2. The status changes while attempting to print.
3. The last character in the stream (the character before the FIN) has been sent to the printer. The unused bit Bit1 in the status is used to indicate this event.

The status byte is defined as follows:

Bit7	80h	Printer ready
Bit6	40h	Acknowledged
Bit5	20h	Out of paper
Bit4	10h	Printer selected
Bit3	08h	I/O Error
Bit2	04h	Unused
Bit1	02h	Last character has been sent to printer
Bit0	01h	Timeout

The status bits are returned by the Int 17 BIOS call (with the exception of Bit1) and are not always consistent, but depend on the BIOS of the particular server.

The utility PRNTEST.EXE in the UTILS subdirectory checks the response from the selected parallel port where LPT1 = 0; LPT2 = 1 and LPT3 = 2. Start PRNTEST.EXE 0 to test LPT1 for example. PRNTEST shows the returned BIOS code from the printer port.

To print successfully, three conditions must be met: 1) the printer must be selected, 2) be ready, and 3) contain paper.

Sum the hexadecimal values for these conditions and use that as the /s and /m parameters for the start prntserv commands. The default values used are /mA8 for the mask and /s80 for the status which implies that data is sent to the printer when it is not busy and out -of-paper, or I/O Error status is not set. Printer selected is ignored as this status bit is often not reported correctly. To include the printer selected bit, specify /mB8 and /s90.

Example

```
start prntserv 10 0
```

Notes on SOCKETS printing

The socket print server accepts TCP connections and routes output to one of four system printers on a parallel port or serial port. Up to four print servers can be started on different TCP ports, connecting to different printers. The server accepts multiple calls, but prints strictly in order of received calls. This means that a connection must be closed to allow another client to print, thereby providing print queuing. There is no indication of queuing status, so a non-ready printer will not hang SOCKETS. Other operations are not affected.

The print server is instructed to send status bytes in the start prntserv command by specifying an optional argument /r=report where report is a hexadecimal value specifying the status bits which are checked for changes to initiate a report. (A report is always sent at the beginning of a job.) The default value is 0 that means that the print server will not report any status changes. A logical value to use is /r=3F which causes any changes, except Printer ready and Acknowledged, to be reported. (The Acknowledged bit is not useful in this application because it toggles at each character; too fast for most networks to carry all the generated traffic.) Note that the full printer status byte is transmitted without filtering any bits.

Notes on printing

Provision is made for non-standard status responses from printers. The *mask* specifies a hexadecimal value to perform a logical AND operation with the status indication from the printer and *status* is the hexadecimal value of the result which causes the printer to print. These options are only used locally at the server and the result is not passed on to the client. Change these options only when error conditions (for example, printer switched off or printer off-line) causes output to be lost. The default values used are 0xA8 for the mask and 0x80 for the ready status which implies that data is sent to the printer when it is not busy and Out of paper or I/O Error status is not set. Note that Printer selected is ignored as this status bit is often not reported correctly. To include the Printer selected bit, specify /m=B8 and /s=90.

The SOCKETS utility PRNTEST.EXE can be used to test the status returned by a specific printer.

tcp

tcp commands display or set various TCP operating parameters.

Syntax

tcp ackonpush 1

tcp irtt [time]

tcp lport [port_number]

tcp mss [size]

tcp retry [number]

tcp rtt [time]

tcp smss [size]

tcp timemax [time]

tcp window [size]

Options

time

time is the new time value in seconds, or milliseconds if "ms" is appended to the number, as in 2000ms.

port_number

port_number is the local port starting number.

size

For **tcp mss**, *size* is the maximum segment size in bytes sent on all outgoing TCP connect requests (SYN segments). *size* tells the remote host the size of the largest segment that may be received by this host. When changing the MSS value, any existing connections remain unchanged.

For **tcp smss**, *size* is the send maximum segment size in bytes sent on all outgoing TCP connect requests. This limits the size of the largest segment that may be sent by this host. When changing the SMSS value, any existing connections remain unchanged.

For **tcp window**, *size* is the size of the receive window in bytes for any new TCP connections. Existing connections are unaffected.

number

number is the number of retries attempted without receiving an acknowledgement from the remote host before the connection is broken. If the value exceeds 255, it implies an infinite number of retries; such a connection does not time-out. The default value for *number* is 6.

tcp ackonpush Sub-command

tcp ackonpush disabled "Delayed ACK". Normally TCP acknowledment should occur as RFC1122 states:

"4.2.3.2 When to Send an ACK Segment

A host that is receiving a stream of TCP data segments can increase efficiency in both the Internet and the hosts by sending fiew than one ACK (acknowledgment) segment per data segment received; this is known as "delayed ACK" [TCP:5].

A TCP SHOULD implement a delayed ACK, but an ACK should not be excessively delayed; in particular, the delay MUST be less than 0.5 seconds, and in a stream of full-sized segments there SHOULD be an ACK for at least every second segment."

The purpose of using "delayed ACK" is to decrease the amount of traffic generated on a TCP connection. In certain applications a delayed ACK can lead to a slower response. The user can now disable the "delayed ACK" functionality by using this command in the SOCKETS configuration file.

ACKONPUSH can be abbreviated to ACK.

Example

```
tcp ackonpush 1
tcp ack 1
```

tcp irtt Sub-command

tcp irtt displays or sets the initial round-trip-time estimate. When specified without an argument, the command displays the current values of TCP parameters including the initial round-trip-time in milliseconds.

time is the initial round-trip-time (IRTT) estimate and is used for new TCP connections until the actual value can be measured and adapted to. By increasing this value when operating over slow communication links, unnecessary retransmissions that otherwise occur before the smoothed estimate value approaches the correct value are minimized. The system default is 5000 milliseconds.

To affect incoming connections, tcp irtt should be executed before the servers are started.

Example

```
tcp irtt 120
```

Sample Output

```
TCP: IRTT 5 ms Retry 6 MSS 1460 SMSS 1460 Window 2920
```

tcp lport Sub-command

tcp lport specifies the local port starting number. When specified without a number the current value of the next free local port number is displayed.

Example

```
tcp  lport  2004
```

Sample output

```
lport = 2004
```

tcp mss Sub-command

tcp mss displays or sets the TCP maximum segment size in bytes. When size is not specified, the current values of the TCP parameters, including the maximum segment size, are displayed. It is recommended to reduce the MSS and SMSS on bad network connections. The SOCKETS queuing overhead is 12 bytes per WriteSocket request i.e. you will need 52 bytes for each request. It will queue 2*MSS bytes for you if it can't send right away because of window constraints by the peer or the Nagle heuristic. That means that if your MSS is set to the default of 1460, 2920 bytes will be queued, consuming 3796 bytes in your case. If you have 5 connections, you need 18980 bytes just to buffer your outgoing data. Use IPSTAT to determine how much memory you have available for everything. (You need memory for each connection, incoming data and all other TCP functions).

Example

```
tcp mss 1460
```

tcp retry Sub-command

tcp retry displays or sets the retry count before a connection is broken. When specified without the number parameter, tcp retry displays the current values of TCP parameters, including the retry count. Refer also to "**TCP Retry Strategy**" on page 101.

tcp rtt Sub-command

tcp rtt replaces the automatically computed round-trip time (RTT) for the specified connection with the time in milliseconds. SOCKETS calculates the RTT as a smooth average of past measured RTTs, starting with the IRTT on a new connection. To get the current RTT in use for a connection n, use the tcp status n command that gives the smoothed average RTT indicated by SRTT. Because tcp rtt provides a manual override of the normal back-off retransmission timing mechanisms, it may be used to speed up recovery from a series of lost packets.

Example

```
tcp rtt 4 100
```

tcp smss Sub-command

tcp mss displays or sets the TCP send maximum segment size in bytes. When size is not specified, the current values of the TCP parameters, including the SMSS, are displayed. A small SMSS causes the remote to reduce its segment size. tcp mss can reduce the MSS and SMSS on bad network connections with high loss rates or where large packets get lost.

Example

```
tcp smss 512
```

tcp window sub-command

tcp window displays or sets the default and maximum receive window size. When specified without the size parameter the current TCP parameters, including the current window size, are displayed.

Example

```
tcp window 2920
```

tcp timemax Sub-command

tcp timemax sets the maximum duration of a tcp retry. If a value greater than 255 seconds is specified, connections never timeout. This is very useful in wireless applications where nodes roam in and out of service.

Example

```
tcp timemax 2000ms
```

SOCKETS command line options

The command line options for SOCKETM and SOCKETP are identical.

Syntax

Socketm [/options…] [*config_file* [*arguments_for_ config_file…*]]

Options

/?

/0

/a

/n=*normal_sockets*

/i=*capi_interrupt*

/d=*dos_compatible_sockets*

/h

/m=*memory_size*

/p

/s=*stack_size*

/v=*api_interrupt*

/q

/u

config_file

The configuration file to use.

If *config_file* is omitted, SOCKETS searches the following paths:
%SOCKETS%\DOS\SOCKET.CFG
%SOCKETS%\SOCKET.CFG
SOCKET.CFG

If the SOCKETS environment variable is not set, %SOCKETS% above defaults to \DL\SOCKETS

arguments_for_ config_file

Can be used to replace %1 to %9 in the configuration file. It is often used to set the IP address or other variable parameters in the configuration file. It can also be used to simplify many functions.

/?

/h

Displays help about command line options.

/0

Instructs SOCKETS to not attempt to save and restore the extended register set of an 80386 processor. This is the only 386-specific code in SOCKETS/DOS, and disabling it makes SOCKETS/DOS fully 80186 compatible.

/a

Instructs SOCKETS to enable the reception of Multicast packets by the Packet Driver. Use this when using multicast functions.

/n=normal_sockets

The number of normal sockets to reserve for use by the Compatibility API. If that API is not needed or no normal socket function is used, set this to 0. This value defaults to 8 i.e. /n=8.

/d=dos_compatible_sockets

The number of DOS compatible sockets to reserve for use by the Compatibility API. If that API is not needed or no DOS compatible function is used, set this to 0. Note that the TCP/IP SOCKETS API (BSD) does not use any DOS compatible sockets. This value defaults to 12 i.e. /n=12.

/i=capi_interrupt

The compatibility API uses interrupt 61hex i.e. /i=61 To change it to a different value, specify the value in hexadecimal eg. /i=62.

/v=api_interrupt

The SOCKETS API normally uses interrupt 7Fh i.e. /v=7fTo change it to a different value, specify the value in hexadecimal eg. /v=7e.

/m=memory_size

Specifies the working memory or heap size for SOCKETS and has a default value of 16384 (16KB) for SOCKETP and 22528 (22KB) for SOCKETM. Increase this value if IPSTAT shows Memory allocation failures during operation.

/p

Sets the minimum size for allocation requests from packet drivers to 64 bytes. Some packet drivers request a specific amount of memory, but proceed to write a minimum of 64 bytes to the allocated memory space.

/s=stack_size

Specifies the stack size used by SOCKETS and has a default value of 2048 (2k). Increase this value if problems are experienced when using nested Async Notification handlers.

/q

Load SOCKETS without displaying any startup messages or diagnostics.

/u

Run SOCKETS with the /u switch to unload it from memory. If another program has taken over the interrupts SOCKETS has hooked, SOCKETS just disables itself when the /u switch is used.

Modem Configuration File Syntax

A modem configuration file is a text file containing lines defining modem “scripts” defining the operation of a modem to make and receive calls. A modem configuration file can be given any name which is referenced in the **interface** command.

A modem configuration file consists of separate commands lines, each starting with one of the following characters in the first column:

i	***initialisation string/script*** (for the modem)
m *time*	**modem pacing -** send a script character to modem every ***time*** milliseconds. Default is 55 ms. Use m 0 for no pacing.
n	***telephone number*** to dial (one number per line)

r	***retry_count*** (when a connection or script failed)
x	***exchange_id*** (XID for user identification)
d	***dial command/script*** (talk to modem till connected)
a	***answer prompt script*** (to remote for him to login)
c	***connect string/script*** (to remote for me to login)
p	***parameters*** (for debugging/watching and dial control)
t *time*	**timeout value -** Lower DTR for ***time*** milliseconds to disconnect modem. Default is 10,000 milliseconds.
b	***command_character*** (default is @ in this syntax)
#	***comments*** (what all good programmers do)

The *initialisation* ***strings***, ***answer prompt***, ***connect string*** and ***dial commands*** consist of modem and login commands or prompts and special functions to cause delays and wait for DCD or strings. The simple scripts are one line strings where commands start with the ***command_character*** (default is @) followed by a script command, followed by a time in milliseconds (and a receive search string in the case of **@r**). The following script commands can be used:

@t *time*	
@w ***time***	wait for the full ***time*** specified
@d ***time***	wait for DCD (modem to get connected)
@f ***time***	wait for ^F XID ^M and find **mdd** with the matching XID
@a ***time***	wait for IP Address (anywhere in data stream) and use it as your own for this interface
@r ***time string***@	wait for ***time*** to receive ***string***
time@.***string***@	terminate ***time*** if followed by a string starting with digits
string	send ***string*** to modem/remote (default in all scripts)
@n	insert telephone number (used in dial string)
@x	insert XID (used in connect string)
@@	send ***command_character*** (@) to modem/remote

Note:

- The ***time*** is given in milliseconds and is terminated by the first character that is not a decimal digit. Maximum ***time*** is one hour (3 600 000ms).
- The script is aborted when a conditional wait times out, without making the connection.
- Receive strings (**@r**) are terminated with a ***command_character*** (@).
- Send strings are terminated with the next command or the end of a line. A carriage return character (CR) is not automatically added.
- To put control characters in the string use ^***n*** where ***n*** is A for 0x01, B for 0x02 and so on. A CR is ^M and a LF is ^J. Use ^***SPACE*** to send the ^ character to the modem.
- Do NOT include spaces as separation characters since all characters are interpreted.
- The @'s are strictly interpreted from left to right - do not confuse them with terminating and initiating @'s. After **@r,** a terminating @ must follow. An initiating @ for the next command MUST be given - it is not automatically assumed since a send string is implied by default.

Example

`@w1000`	wait for 1 second
`@d30000`	wait for DCD, retry dial after 30 seconds.
`@@`	send @ to modem/remote
atdt`@n^M^J`	dial the next number in the list
`@r2000login@`	wait for the string "login"

parameters comprise an optional string specifying whether dial communications must be displayed and how it must be controlled. Use **x** to display transmit data and **r** for received data. The display is only active while DCD is low. Use **d** to always dial when DCD is or becomes low. Use **n** for no dial on demand. Note that **d** and **n** are mutually exclusive.

An example or two is the fastest way to understand the explanations above. For a simple connection where no passwords or identity checks are needed, try the following:

```
# Modem definitions for Zoltrix Hayes compatible modem
# This example is for 'asy' interfaces making outgoing
# connections with no logon sequence.
#
# initialisation string
# (Warning: consult the manual for your own modem)
# send "a", wait half a second, send "a", wait half a second,
# send "ats0=0" - do not use auto answer
# send "&c1" - use state of carrier for DCD
# send "&d3" - disconnect when DTR low
# send "&k3" - enable RTS/CTS flow control
# send "&q5" - select error correction
# send "<CR><LF>", wait 100 milliseconds
i a@w500a@w500ats0=0&c1&d3&k3&q5^M^J@w100
#
# dial command - send "<CR><LF>", wait 2 seconds
# send "atdt<number><CR><LF>"
# wait 30 seconds for DCD
d ^M^J@w2000atdt@n^M^J@d30000
#
# two numbers to dial in rotation
n 790-1234
n 790-1235
#
# parameters: r=show modem receive, x=show modem xmit
p r
#
# number of retries
r 5
```

A more typical login sequence (where a user ID and password is required) will use the connect script as in this example:

```
# modem definitions for US Robotics Hayes compatible modem
# send "a", wait a tenth of a second
# send another "a", wait a tenth of a second,
# send "at&c1" - use state of carrier for DCD
# send "&d3" - disconnect when DTR low
# send "&i0&r2&h1" - enable RTS/CTS flow control
# send "&m4" - select error correction
# send "s0=0" - do not use auto answer
# send "<CR><LF>", wait 100 milliseconds
i a@w100a@w100at&c1&d3&i0&r2&h1&m4s0=0^M^J@w100
# number to dial
n 03456789
# number of retries
r 4
#
# dial command - send "<CR><LF>", wait 1 second
# send "atdt<number><CR><LF>"
# wait 40 seconds for DCD
d ^M^J@w1000atdt@n^M^J@d40000
#
# connect script - wait 7 seconds for login prompt
# send user ID, wait 2 seconds for password prompt
# send password, wait 2 second for acknowledgement
c @r7000ogin@myuserid^M@r2000sword@mypassw^M @r2000Welcome@
# parameters: r=show modem receive, x=show modem xmit
p rx
```

Note that the '@m's in the above connect string are not commands since the @ is interpreted as the end of a **@r** 'wait for receive' string. Sometimes you would have two @'s like this example: Should you want to put a small delay before replying to a prompt it would contain @@ as follows:

```
c @r7000ogin@@w100myuserid^M
```

For receiving incoming calls, the answer command/script is used. You can use it to do a single user ID/password controlled login, just a user ID login, or unconditional acceptance. The more general case is to use **mdd** interfaces with an XID (exchange ID) to validate multiple users. Examples for using Multi Destination Drivers (MDD) are given below. If your SOCKETS workstation accepts only incoming calls, you do not need to enter dial commands.

Retry Strategy on Time-out

For outgoing calls, when a wait for DCD (@d) times out without receiving DCD, SOCKETS will drop the call (by lowering DTR). If the number of retries has not been exhausted, SOCKETS will retry the dial command with the next phone number rotating through the list of numbers. If a connection with the remote modem is successful (received a DCD) and there is a timeout on the wait for ***string*** command in the connect script, the connect script is re-started for the number of retries specified. If the number of retries has been exhausted, SOCKETS will retry the dial and

connect commands only after being prompted again by receiving traffic for this destination host or net.

For incoming calls, when there is a time out on the wait for XID or ***string*** commands, the answer-prompt script is re-started for the number of retries specified. If the number of retries has been exhausted, SOCKETS will drop the call (by lowering DTR).

Multi Destination Drivers

When you have more than one destination to or from dial-up links using SLIP, and are using more than one modem, you need a mechanism to link the logical interface (with IP address and routing info) to the physical interface (COM port with modem definitions). Using an **asy** type interface permanently links the logical and physical interface with one IP address. This works OK for dialling out to multiple destinations only where you are the end user and nobody routes through you.

Two interface types - **mdd** and **aslink -** have been defined to enable dial-up operation to other networks using SLIP.

- **mdd** defines a logical interface with associated IP address, routes, MTU, buffer limit and a modem configuration file but without a physical interface. For each dial-up destination a **mdd** interface is created, and a route to each destination specified.

- **aslink** interface defines an uncommitted asynchronous interface with an associated COM port as specified by the I/O address, and interrupt vector.

When initial traffic is to be sent to a destination through a **mdd** interface, SOCKETS scans all **aslink** interfaces for the first available one and assigns it to the **mdd** interface. This association is broken when the dial connection is broken or when the dial attempt fails. A dial attempt fails when the retry count specified in the modem file expires.

Incoming calls can be received on any **aslink** interface. This interface must then be associated with the correct **mdd** interface (and IP address with route) for the calling host. An exchange identifier (XID) defined by both the calling host and the called host achieve this.

The protocol for exchanging the XID is implemented partly by SOCKETS and partly by the user defined 'connect' and 'answer' scripts in the modem configuration files. When an incoming call is received by an **aslink** interface, as seen when DCD is raised, the 'answer' script is executed. This script must contain a command to wait for the XID and match it to a local **mdd** containing the same XID and an acknowledgement to the sender that the exchange has been successful. The XID *must* be preceded by an ACK (^F) and followed by a CR (^M). The 'connect' script of the calling host is used for this purpose. The following ‘answer’ and 'connect' scripts may be used:

```
a @w100^M^JYour ID?@f1000ccc
```

This means: Wait for 100 milliseconds (after receiving the DCD) send a CR/LF to write on a new line and prompt the user with 'Your ID?’ Wait 1 second to receive a valid XID matching with an **mdd** interface and then send a few 'c's to acknowledge 'connected' for the caller.

```
c @r500ID?@^F@x^M@r500c@
```

This means: Wait half a second to receive the 'ID?' prompt, send the required ACK (^F), XID (@x) and CR (^M). Wait another half second for acknowledgement from the remote with a 'c'.

It can occur that a request for making an outward connection and an attempt by an incoming connection clash at the same **mdd** interface. One example is when a connection is broken and both sides redial each other to restore the connection. A mechanism must be used to allow only one of the attempts to be successful. Therefore a **mdd** interface will drop an **aslink** trying to make an outgoing connection as soon as it senses another **aslink** trying to connect to it with an incoming connection.

In summary the modem commands used by the **mdd** and the **aslink** in their respective modem files for the following functions are:

- For initiating the modem: Always use the '**i**-' command in the **aslink** modem file.
- For making a call: Use the telephone numbers from the **n**-command in the **mdd** modem file, select any available **aslink**, and use its **d**-command. When the modems are connected, use the **c**-command in the **mdd** modem file to logon with its XID (**x**-command).
- For answering a call: Use the **a**-command in the **aslink** modem file to query the user for his XID (logon sequence), find the **mdd** modem file with the matching XID (**x**-command).

More examples are available in the HAYES.MOD file. See the command descriptions in the **Modem Configuration File Syntax** section above.

Example

An **mdd** modem configuration file for both incoming and outgoing connections:

```
n 790-1234
n 790-1235
x id-abc
c @r20000ID?@^F@x^M@r500c@
r 5
```

An **aslink** modem configuration file for both incoming and outgoing connections:

```
i a@w500a@w500ats0=1&c1&d3&k3&q5^M^J@w100
a ^M^JYour ID?@f5000@w200ccc
d ^M^J@w2000atdt@n^M^J@d30000
r 5
```

Example

An **aslink** modem configuration file for incoming connections:

```
i a@w500a@w500ats0=1&c1&d3&k3&q5^M^J@w100
```

```
a ^M^JYour ID?@f5000@w200ccc
r 5
```

An **mdd** modem configuration file for incoming connections:

```
x id-abc
```

Example

An **mdd** modem configuration file for outgoing connections:

```
n 790-1234
n 790-1235
x id-abc
c @r20000ID?@^F@x^M@r500c@
r 5
```

An **aslink** modem configuration file for outgoing connections:

```
i a@w500a@w500ats0=0&c1&d3&k3&q5^M^J@w100
d ^M^J@w2000atdt@n^M^J@d30000
r 5
```

Configuration Considerations

MTU (Maximum Transmission Unit)

The MTU is the maximum size in bytes of a data packet (including all IP and TCP header information.) Information is sent across the Internet in packets, which are reassembled into a whole when they reach their destination. The size of these packets is dependent on the MTU of the machines along the route the packets travel.

The MTU can be specified by each individual computer. When you send out a request for information, the computer you are are requesting information from will read your request and hopefully send out packets of the size that you request. If the destination machine has a smaller MTU, the packets will be broken into pieces, or fragmented.

MSS (Maximum Segment Size)

MSS is the maximum size in bytes of the data portion of a TCP/IP packet This value is normally the MTU setting minus 40. MSS is used by SOCKETS to determine the WriteSocket queue size. The SOCKETS queuing overhead is 12 bytes per WriteSocket request so you will need 52 bytes for each request. SOCKETS will queue 2*MSS bytes if it can't send right away because of window constraints by the peer or the Nagle Algorithm. This means that if your MSS is set to the ethernet default of 1460, 2920 bytes will be queued, consuming 3796 bytes with overhead per connection. If you have five connections, you need 18980 bytes just to buffer your outgoing data. Use IPSTAT to determine how much memory you have available. (You need memory for each connection, incoming data, and all other features of the TCP/IP stack).

Buffers

Buffers are used to store data packets that are sent and received. The size of each buffer is determined by the MTU setting. Additionally, SOCKETS adds an additional 12 bytes for each WriteSocket request. The maximum amount of memory required for buffers is determined by the following formula

Maximum Buffer Memory Use = MTU * Buffers

A buffer limit of 30 is excessive if you have 1500 byte buffers (1500 * 30 = 45000). By default, SOCKETS allocates 20000 bytes, which is used for a lot of other processes as well, so you are bound to encounter out-of-memory situations. If you set MTU to 1500, then 5 would be a recommended setting for the number of buffers.

Chapter 5, SOCKETS Configuration Examples

The following examples within this chapter refer to specific files, such as "EXAMPLE1.CFG" and "MODEM.MC2". These example files can be found in the \SOCKETS\CONFIGS directory created during the install process.

- Ethernet connection with SOCKETS acting as a server to a Win9X or NT system. The DL web server can be launched and a standard desktop browser will "surf" to the SOCKETS system.
- Dial-Up serial connection to an ISP.
- Single Dial-In Connection.
- Single Dial-In connection with ASY interface.
- Direct serial connection with SOCKETS as a server.
- Direct serial connection with SOCKETS as a client.
- Dial-up slip connection with SOCKETS as an IP router.
- Multiple dial–in connections.

Example 1: Ethernet Connection – SOCKETS Serving a Web Page

SOCKETP is loaded from the command line as follows:

SOCKETP.EXE EXAMPLE1.CFG

To demonstrate the web interface, next launch HTTPD.EXE. If the index file is not within the current directory please remember to set the environment variable HTTP_DIR to the location of the index files or nothing will be displayed to the desktop browser.

SET HTTP_DIR=C:\DL\SOCKETS\SERVER

HTTPD.EXE /R

The configuration file EXAMPLE1.CFG contains:

Ip address 196.6.1.111/24	Sets an IP address of 196.6.1.111 with a 24-bit subnet mask to the following interface.
Iface pdr if0 dix 1500 5 0x60	Creates a standard Ethernet connection, type pdr (packet driver), interface name if0, type dix, MTU 1500, Buffer Limit 5, Interrupt vector of the packet driver at 0x60.
route add default if0 196.0.0.1	Set the default route for packets travelling to and from if0 to 196.0.0.1
domain server 196.0.0.1	Set the domain name server to 196.0.0.1 This server is used to convert names to IP addresses.
ip address	Cause SOCKETS to display IP status

# set TCP parameters	
tcp window 2920	Set the windows size to 2920 bytes
tcp retry 6	Set the retry count to 6
tcp irtt 500ms	Set the Initial Round Trip Time to 500ms
tcp mss 1460	Set the Maximum Segment Size to 1460 bytes

Example 2: Serial Connection – SOCKETS Dial-up to ISP

SOCKETM is loaded from the command line as follows:

SOCKETM.EXE EXAMPLE2.CFG

The configuration file EXAMPLE2.CFG contains:

interface asy p0 ppp 576 10 0x3f8 4 19200 modem.mc2	Creates an asynchronous ppp connection on COM1 IRQ4 named p0 with an MTU of 576 and buffer limit of 10. Uses the file modem.mc2 for modem configuration information.
route add default p0	Route all packets through interface p0
ip ttl 64	Set time to live for packets at 64 “hops”
tcp mss 1460	Set Maximum Segment Size to 1460 bytes
tcp window 2920	Set Windows Size to 2920 bytes
par p0 ipcp local compress tcp 16 1	Enables header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par p0 ipcp local address 0.0.0.0	Server assigned IP address
par p0 lcp local accm 0	Asynch control character map set all bits to zero.
par p0 lcp local acfc on	Address control field compression on or off.
par p0 lcp local pfc on	Protocol field compression on or off.
par p0 lcp local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a configure request, then comparing the number received. If it is the same then there is a possible loop back, repeat test.
par p0 pap user test example	Set the user name to “test” and password to “example”
par p0 ipcp open par p0 lcp open	Open specified layer. ipcp = ip control protocol. Lcp = link control protocol. When an immediate dial operation is required the "par if0 lcp start" command should be used. When dial-on-demand is desired the “par if0 lcp open" command should be used and when connect-on-

	received-call is desired, the "par if0 lcp listen" command should be used. The start command implies the open command and the open and listen commands may both be used. Dial-on-demand can also be disabled by the "p n" modem configuration file parameter and by an API call. Dial-on-demand can be enabled by an API call.

MODEM.MC2

i A@w100atdt^M^J@w100	Initialise modem
d ^M^J@w2000ATDT@n^M^J@d40000	Dialling string
n 123-4567	Number to dial
r 3	Number of retries
p rx	Enable debugging output to screen

Example 3: Single Dial-in Connection

Allow only single users to dial in and issue each with an IP address on the fly. Setup **asy** interfaces, each with its own IP address, COM port, modem and telephone number. In the answer script, a password may be asked (no need for XID: does it in the script), and on success an IP address is sent in dotted decimal form. (Should you want to use different passwords, an **mdd/aslink** setup as described in Example 2 may be used, but then every user will have his own IP address fixed to the XID.)

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP parameters is to allow the SOCKETS implementation to act as both a “PPP server” and a “PPP client” without having to re-configure it.

The configuration file EXAMPLE3.CFG contains interface configurations for all modem ports as well as other connections (like Ethernet) that might be used. An **asy** interface could be:

IPAddress=192.6.3.1/30	Set the IP Address of the following interface to 192.6.3.1 with a subnet mask set to 30 bits.
iface asy sl0 cslip 576 10 0x3F8 4 19200 modem.mc3	An asy modem connection on COM1 connecting to the smallest subnet allowing 2 hosts.

The modem definition file is MODEM.MC3:

i a@w100atz^M^J@w500ats0=1^M^J@ w100	Initialise modem (consult the manual for your own modem initialisation string)
x 192.6.3.2	Specify IP address (in the XID variable to make the answer scripts more uniform).
a @w1000^M^JPassword? @r9000Sockets@@w200^M^JYour address: @x ^M^J@r1000GOTIP@	Answer script to prompt remote for logon: Wait 1 second, send "<CR><LF>" to get cursor on a new line, send "Password? " and wait 9 seconds to receive "Sockets" somewhere in the data stream # (the connection will break if failed), wait 200 milliseconds and send the IP address with @x on a new line. Wait 9 seconds to receive "GOTIP" as confirmation.
p rx	Parameters for debugging (show modem data): r=receive and/or x=xmit

Example 4: Single Dial-in Connection with ASY Interface

A user who dials into the system in Example 3 which supplies the IP address for the user, has to use an **asy** interface to his modem. The user must specify the **@a** address command in the connect script in the modem file. The ***@a time*** command waits for ***time*** milliseconds to see the dotted decimal form of an IP address. It assigns this address to the **asy** interface. The user should have a default route to this **asy** interface to be able to see the whole world through it, or just a specific route for what he wants to see. The configuration file SOCKET4.CFG should contain at least the **asy** interface configuration similar to the previous examples.

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP parameters is to allow the SOCKETS implementation to act as both a “PPP server” and a “PPP client” without having to re-configure it.

The modem definition file is MODEM.MC4:

i a@w100atz^M^J@w500ats0=0^M^J@ w100	Initialise the modem (consult the manual for your own modem initialisation string).

d ^M^J@w2000atdt@n^M^J@d40000	Dial command
n 0800123456	Number to dial
r 5	# Number of retries
c @r7000sword@Sockets^M^J@a1000GOTIP	Connect script to login at the remote: wait 7 seconds after DCD to receive password prompt, send "Sockets" as password, wait 1 second for IP address, send "GOTIP" as confirmation.
p rx	Parameters for debugging (show modem data): R=receive and/or x=xmit

Example 5: Direct Serial Connection with SOCKETS as a Server

This section explains how to configure SOCKETS as a server to listen on the designated serial port and wait for a valid connection and use standard dial up networking to connect.

The Client Connection

There are two methods of connection available - direct serial connection or dial in on a standard phone line. The Client here can be any Win95, Win98, or WinNT system.

For a direct serial connection a modem driver must be installed. Such a driver does not ship standard with SOCKETS but can be obtained as freeware at **http://www.aeriden.com**. The client must then create a dial up networking connection with the phone number of CLIEN1, no username or password. Please read the manual distributed with the aeriden driver to ensure correct setup.

Configuring SOCKETS

Configuration of the SOCKETS software is handled within two files, namely EXAMPLE5.CFG and MODEM.MC5. SOCKETS can be loaded by means of a batch file that contains the following commands:

SOCKETM EXAMPLE5.CFG

These commands allow SOCKETS to allocate more memory, than allowed by the default values, when initiating a TCP/IP connection over a serial link. SOCKETS processes the EXAMPLE5.CFG file followed by the MODEM.MC5 file.

SOCKETS Configuration File Details

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP

parameters is to allow the SOCKETS implementation to act as both a “PPP server” and a “PPP client” without having to re-configure it.

EXAMPLE5.CFG contains:

iface asy if0 ppp 576 5 0x2f8 3 19200 modem.mc5	Creates an asynchronous connection on com 2, IRQ 3 with a Baud rate of 19200, MTU of 576, and a Buffer Limit of 5. Access the modem.mc5 file for specific initialization strings.
User SocketsUser SocketsPassword 196.10.229.18	For user connecting with name “SocketsUser” and Password “SocketsPassword” assign the ip address of 196.10.229.18
par if0 ipcpin local compress tcp 16 1	Enables header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par if0 ipcpin local address 196.10.229.2	
par if0 ipcp local compress tcp 16 1	Enables header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par if0 ipcp local address 196.10.229.4	
par if0 lcpin local accm 0	Asynch control character map, set all bits to zero.
par if0 lcpin local acfc on	Address control field compression on or off.
par if0 lcpin local pfc on	Protocol field compression on or off.
par if0 lcpin local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a configure request, then comparing the number received. If it is the same, and then there is a possible loop back, repeat test.
par if0 lcpin local authen pap	Set local authentication to pap
par if0 lcp local accm 0	Asynch control character map set all bits to zero.
par if0 lcp local acfc on	Address control field compression on or off.
par if0 lcp local pfc on	Protocol field compression on or off.
par if0 lcp local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a configure request, then comparing the

	number received. If it is the same, then there is a possible loop back, repeat test.
Par if0 ipcp open Par if0 ipcpin open par if0 lcp listen par if0 lcpin listen	Open specified layer. ipcp = ip control protocol. Lcp = link control protocol. When an immediate dial operation is required the "par if0 lcp start" command should be used. When dial-on-demand is desired the “par if0 lcp open" command should be used and when connect-on-received-call is desired, the "par if0 lcp listen" command should be used. The start command implies the open command and the open and listen commands may both be used. Dial-on-demand can also be disabled by the "p n" modem configuration file parameter and by an API call. Dial-on-demand can be enabled by an API call.
route add default if0	Route all packets through interface if0
ip address route	Prints out the systems ip address and route. Useful for debugging only.

MODEM.MC5

	Initialize Modem. For a direct serial connection no initialization string is necessary.
a @r20000CLIEN1@CLIENTSERVER	When Sockets receives the string “Clien1” send the response “ClientServer”
r 3	Number of Retries
p rx	Debugging information parameters: r=show modem receive, x=show modem xmit d=always dial, n=no dial-on-demand

Example 6: Direct Serial Connection with SOCKETS as a Client

Setting the Server

The NT server must have Remote Access Services enabled, commonly referred to as RAS, as well as the "Direct Serial cable between two PCs " modem driver installed on the proper COM port. The following example also connects using DHCP. If this feature is to be used then DHCP services must also be setup on the NT server. Other options would include the use of static IP addressing or BOOTP. Please review the cabling requirements from the WinNT documentation or

help files. Standard cables often will not work with direct connections due to the cabling demands of WinNT.

Configuring the Software

Configuration of the SOCKETS software is handled within two files, namely EXAMPLE6.CFG and MODEM.MC4. SOCKETS can be loaded by means of a batch file that contains the following commands:

SOCKETM EXAMPLE6.CFG

These commands allow SOCKETS to allocate more memory than allowed by the default values when initiating a TCP/IP connection over a serial link. SOCKETS processes the ECAMPLE6.CFG file followed by the MODEM.MC6 file.

Configuration File Details

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP parameters is to allow the SOCKETS implementation to act as both a "PPP server" and a "PPP client" without having to re-configure it.

`EXAMPLE6.CFG:`

Iface asy if0 ppp 1500 30 0x2f8 3 19200 modem.mc6	Creates an asynchronous connection on com 2, IRQ 3 with a baud rate of 19200 and access the modem.cfg file for specific initialization strings.
par if0 ipcp local compress tcp 16 1	Enables header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par if0 ipcp local address 0.0.0.0	The ip address 0.0.0.0 indicates that the server must assign the client a valid ip address
par if0 lcp local accm 0	Asynch control character map, set all bits to zero.
par if0 lcp local acfc on	Address control field compression on or off.
par if0 lcp local pfc on	Protocol field compression on or off.
par if0 lcp local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a

	configure request, then comparing the number received. If it is the same, then there is a possible loop back, repeat test
par if0 pap user id password	Replace "id" and "password" with a valid username and password on the NT server that you are connecting to.
par if0 ipcp open par if0 lcp open	Open specified layer. ipcp = ip control protocol. lcp = link control protocol.
route add default if0	Route all packets to interface if0
ip address route	Prints to the screen the systems ip address and route, useful for debugging only.

MODEM.MC6

i CLIENT@r5000CLIENTSERVER@ @a2000	Initialize Modem and connect as client
r 3 p rx	Number of Retries Debugging parameters r display what we receive x display what we transmit

Example 7: SOCKETS Machine Using Call Back Verification.

Scenario:

If the NT machine logs into the SOCKETS machine with SocketsUser1 / SocketsPassword1, the NT machine is assigned IP address 196.10.229.18 and the SOCKETS machine is 196.10.229.2.

If the NT machine logs into the SOCKETS machine with "SocketsUser / SocketsPassword," the SOCKETS machine will break the connection and dial back to 08036501 logging in as "NtUser" with password "NtPassword." The NT machine assigns the IP addresses of both machines.

The SOCKETS machine never initiates a modem call unless it has been called first. Once SOCKETM is running, the IOCTL program can be used to initiate a dial operation or to enable dial on demand.

SOCKETS Configuration File Details

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP parameters is to allow the SOCKETS implementation to act as both a "PPP server" and a "PPP client" without having to re-configure it.

EXAMPLE7.CFG

iface asy p ppp 1500 30 0x3f8 4 9600 modem.mc7	Creates an asynchronous connection designated interface “p” type ppp on COM1 IRQ4 with an MTU of 1500, Buffer limit of 30, baud rate 9600 and modem information contained within the modem.mc7 file
route add default p	Routes all packets through interface “p”
user SocketsUser SocketsPassword 196.10.229.18 cbv	If user is SocketsUser with password SocketsPassword assign IP address of 196.10.229.18 then disconnect session and call back.
user SocketsUser1 SocketsPassword1 196.10.229.18	If user is SocketsUser1 with password SocketsPassword1 assign IP address of 196.10.229.18 and continue session.
par p ipcpin local compress tcp 16 1	Enables header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par p ipcpin local address 196.10.229.2	Assign local IP address 196.10.229.2
par p ipcp local compress tcp 16 1	Enables header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par p ipcp local address 0.0.0.0	The ip address 0.0.0.0 indicates that the server must assign the client a valid ip address.
par p lcpin local accm 0	Sets the Asynch Control Character Map to 0.
par p lcpin local acfc on	Sets local Address and Control Field Compression on.
par p lcpin local pfc on	Sets local Protocol Field Compression on.
par p lcpin local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a configure request, then comparing the number received. If it is the same, then there is a possible loop back: repeat test
par p lcpin local authen pap	Sets the local Authentication protocol to pap.
par p lcp local accm 0	Sets the local Asynch Control Character Map to 0.

par p lcp local acfc on	Sets the local Address and Control Field Compression to on.
par p lcp local pfc on	Sets the local Protocol Field Compression to on.
par p lcp local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a configure request, then comparing the number received. If it is the same, then there is a possible loop back, repeat test. .
par p pap user NtUser NtPassword	Allow login for user NtUser with password NtPassword. PAP corresponds with the previous set authentication protocol.
par p ipcp open	Set incoming dial-in method to open.
par p ipcpin open	Set incoming dial-in method to open.
par p lcp listen	Set dial-out method to listen.
par p lcpin listen	Set dial-out method to listen.

MODEM.MC7

i a@w500a@w500atl1w1%e2&q5&c1&d2s11=70&k3s30=18s0=0^M^J@w100	Modem Initialization string. Please refer to the modem manufacturer manual for the specific initialization string to match your modem.
a	
n 08036501	Number to dial
d ^M^J@w1000atd@w110@n@w110^M@d40000	Dial command
c	
r 4	Number of retries to dial the connection.
p n	Debugging parameters: r=show modem receive, x=show modem xmit d=always dial, n=no dial-on-demand

Example 8: SOCKETS Machine with CBV and Logging-in.

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP parameters is to allow the SOCKETS implementation to act as both a “PPP server” and a “PPP client” without having to re-configure it.

EXAMPLE8.CFG

iface asy p ppp 1500 30 0x3f8 4 9600 modem.mc8	Creates an asynchronous connection designated interface “p” type ppp on COM1 IRQ4 with an MTU of 1500, Buffer limit of 30, baud rate 9600 and modem information contained within the modem.mc8 file
route add default p	Routes all packets through interface “p”
user NtUser NtPassword 196.10.229.208	If user is NtUser with password NtPassword assign IP address of 196.10.229.208 then disconnect session
par p ipcp local compress tcp 16 1	Enables header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par p ipcp local address 0.0.0.0	The ip address 0.0.0.0 indicates that the server must assign the client a valid ip address.
par p lcp local accm 0	Sets the local Asynch Control Character Map to 0.
par p lcp local acfc on	Sets the local Address and Control Field Compression to on.
par p lcp local pfc on	Sets the local Protocol Field Compression to on.
par p lcp local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a configure request, then comparing the number received. If it is the same, then there is a possible loop back, repeat test.
par p pap user SockectsUser SocketsPassword	Allow login for user SocketsUser with password SocketsPassword. PAP corresponds with the previous set authentication protocol.
par p ipcp open	Set incoming dial-in method to open.
par p lcp open	Set incoming dial-in method to open.
par p ipcpin local compress tcp 16 1	Enables incoming header compression, where 16 are the maximum number of concurrent TCP/IP connections. 1 turns on compression, 0 turns off compression.
par p ipcpin local address 196.10.229.202	Set the local IP address on an incoming transmission to 196.10.229.202
par p lcpin local accm 0	Sets the local Asynch Control Character Map to 0.
par p lcpin local acfc on	Sets the local Address and Control Field

	Compression to on.
par p lcpin local pfc on	Sets the local Protocol Field Compression to on.
par p lcpin local magic on	Magic number option on or off. The magic number is used to detect loop back links by creating a number, sending a configure request, then comparing the number received. If it is the same, then there is a possible loop back: repeat test
par p lcpin local authen pap	Set the authentication protocol for incoming lcpin to pap.
par p ipcpin open	Set dial-in method for ipcpin to open.
par p lcpin open	Set dial-in method for lcpin to open

MODEM.MC8:

I a@w500a@w500atl1w1%e2&q5 &c1&d2s11=70&k3s30=18s0=0^ M^J@w100	Initialize the modem; please refer to the manufacturer documentation for the specific modem initialization string.
a	
n 08034131	Number to dial
d ^M^J@w1000atd@w110@n@w1 10^M@d40000	Dial string.
c	
r 4	Number of retries.
p r	Debugging information. Parameters: r=show modem receive, x=show modem xmit d=always dial, n=no dial-on-demand

When the “NT look-alike” side starts up, it dials 08034131. The “SOCKETS” side answers and after the successful login, drops the connection and dials 08036501. During PPP negotiation, the “NT look-alike” side is assigned IP address 196.10.229.202 and the “SOCKETS” side, IP address 196.10.229.108.

Example 9: Dial-up SLIP Connection with SOCKETS as an IP Router

You want to connect your LAN to another network via a modem using a dial-up SLIP link. The SOCKETS PC will act as an IP router (or gateway) to the other network. This example is symmetrical: You can use the same setup on both sides to enable any side to initiate the call. (The IP addresses will have to be unique.)

Use an **asy** interface with CSLIP to get better throughput. Your network is 192.6.1.0 and your address 192.6.1.111. The **asy** interface will also be 192.6.1.111 linking the 192.6.2.0 network (or the rest of the world) on the other side. The modems establish a connection with no logon required.

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP parameters is to allow the SOCKETS implementation to act as both a "PPP server" and a "PPP client" without having to re-configure it.

The configuration file EXAMPLE9.CFG contains:

Ip address 196.6.1.111/24	Sets an IP address of 196.6.1.111 with a 24-bit subnet mask to the following interface.
Iface pdr if0 dix 1500 10 0x60	Creates a standard Ethernet connection, type pdr (packet driver), interface name if0, type dix, MTU 1500, Buffer Limit 10, Interrupt vector of the packet driver at 0x60.
IPAddress=192.6.1.111/0	Sets an IP address of 192.6.1.111 with a 0-bit subnet mask to the following interface.
Iface asy sl0 cslip 576 10 3f8 4 19200 asy-io.mod	Creates a standard modem connection on COM1. The connection is asynchronous, interface name sl0, MTU 576, Buffer Limit 4, baud rate 19200, and reference the file asy-io.mod for specific modem information.
; Make this the default route:	
par sl0 19200 CTS RTS	For interface sl0 set the serial baud rate to 19200, output flow to CTS, input flow to RTS. CTS = Clear to Send modem signal. RTS = Request To Send modem signal. *Use only for asy type interfaces.
par sl0 ripadv	Enable Routing Information Protocol on interface sl0.
rip advertise 30	Set RIP advertise time to 30. Time is the number of elapsed seconds before the advertisement is repeated.
rip use 200	Set RIP use time to 200. Time is the period during which routes are added or amended as a result of RIP, and are valid for time seconds.

The start up file SOCKETS.STU will display the current status with:

ip address	Display the IP address
route	Display the routes
ifstat	Display the interfaces

The modem definition file is ASY-IO.MOD:

I a@w100atz^M^J@w500ats0=1^M ^J@w100	Initialise modem (Caution: consult the manual for your own modem)
d ^M^J@w2000atdt@n^M^J@d400 00	Dial command: send "<CR><LF>", wait 2 seconds, send "atdt<number><CR><LF>", wait 40 seconds for DCD
n 0,011-790-1234	The number to dial
r 5	Number of retries
p r	Parameters for debugging (show modem data): r=receive and/or x=xmit
	Password protected logon facilities can be added with answer and connect scripts. (See next examples or ASY-IOL.MOD on disk)

Example 10: Multiple Dial-in Connections

You have a number of modems for dial-in of various users or other networks each with a fixed IP address. When somebody dials into your system, SOCKETS will need to know his IP address to be able to set up a return route to him. The network size for each remote host must also be known. (You have to understand sub-networks for this example.)

The way to do this is to set up a **mdd** interface for each IP address (using an IP address on that network). Each **mdd** interface has a modem definition file containing a unique exchange identifier (XID). Each modem has an **aslink** interface with a modem file containing an answer script. When the remote dials in, he has to specify his XID in his connect script (like a login). The **aslink** then links to the **mdd** with the same XID and your remote user is connected using the routes set for his **mdd** interface.

PPP Parameters:

Ipcp = ip control protocol.
Lcp = link control protocol.

The commands **lcpin, papin, apin** and **ipcpin** can be specified for incoming parameters. These correspond with the **lcp, pap, ap** and **ipcp** commands. The reason for having a different set of PPP parameters is to allow the SOCKETS implementation to act as both a "PPP server" and a "PPP client" without having to re-configure it.

The configuration file EXAMPL10.CFG should contain:

IPAddress=197.55.2.9	Set the IP address of the following interface to 197.55.2.9
iface pdr if0 dix 1500 10 0x60	Creates a standard Ethernet connection, type pdr (packet driver), interface name if0, type dix, MTU 1500, Buffer Limit 10, Interrupt vector of the packet driver at 0x60.
; A Multi Destination Driver	
; connecting to the smallest subnet allowing 2 hosts	
IPAddress=193.101.51.1/30	Set the IP address of the following interface to 193.101.51.1 with a 30-bit subnet mask.
iface mdd mdd0 slip 576 10 mdd10.mod	Create a multiple destination interface, type mdd, interface name mdd0, class slip, MTU 576, Buffer Limit 10, access the file mdd10.mod for specific interface commands.
iface asy as10 cslip 576 10 0x3F8 4 19200 aslink.mod	Create an aslink connection on COM1.

All the modems (**aslink** interfaces) may use the ASLINK.MOD file:

i a@w100atz^M^J@w500ats0=1^ M^J@w100	Initialise modem (Warning: consult the manual for your specific modem initialisation string)
a @w500^M^JXID?@f5000@w20 0ccc	Answer script to prompt for XID
p r	Parameters for debugging (show modem data): r=receive and/or x=xmit

All the **mdd** interfaces will use a specific MDD*n*.MOD file with its own unique XID. The XID links to the IP address in the configuration of the **mdd** interface. MDD0.MOD will be:

x id0	Set the XID variable
p r	Parameters for debugging (show modem data): r=receive and/or x=xmit

The users dialling into the system above may use **asy** interfaces with a modem file containing the following commands (see file ASY-OX.MOD):

i a@w100atz^M^J@w500ats0=1^M ^J@w100	Initialise modem (Warning: consult the manual for your own modem)
d ^M^J@w2000atdt@n^M^J@d400 00	Dial command: send "<CR><LF>" wait 2 seconds, send "atdt<number><CR><LF>" wait 40 seconds for DCD.
n 790-1234	The number to dial
r 5	Number of retries
x id0	Set the XID variable
c @r1000XID@^F@x^M@r100c	Connect script to login at the remote: wait 1 second after DCD to receive XID prompt, send the ^F XID ^M sequence, receive a "c" as confirmation.
p rx	Parameters for debugging (show modem data): r=receive and/or x=xmit

Chapter 6, SOCKETS Utility Descriptions

SOCKETS Utilities

SOCKETS utilities make use of command-line parameters and/or configuration files. Please be careful to note the name and location of the configuration file used by the application you are working with. All SOCKETS applications require that the kernel be loaded before the application is run in order to function properly.

ARPSTAT

ARPSTAT is a demonstration program to illustrate the use of the ArpApi function. It is hard coded to only remove Ethernet entries,

Syntax

ARPSTAT *ip address*

Remarks

ip address

The IP address of the ARP entry to be removed. If this is 0.0.0.0 then all entries will be removed.

Example

```
ARPSTAT 198.162.1.1
```

Removes the ARP entry associated with the IP address of 198.162.1.1

DHCPSTAT

DHCPSTAT displays the DHCP information for the machine.

Syntax

DHCPSTAT [r | v]

Options

The r option is for forcing a renewal of the DHCP lease.

The v option displays the SOCKETS version information.

Example

```
DHCPSTAT
```

This will display all the DHCP information, such as IP address and lease time.

FTP

FTP is a file transmitting and retrieving client that runs in interactive or batch mode.

Syntax

FTP server [options]

Options

/n

/v

/p=Port

/f=ScriptFile [ScriptParameters]

Remarks

Server

The name or ip address of a server to connect to.

/n

Suppress progress indicator.

/v

Verbose output for troubleshooting.

/p=Port

Connect to a server port other than the standard FTP port number of 21.

/f=ScriptFile

A file containing commands for the client to send to the server upon connection. Simple parameter substitution is performed, with the first element of *ScriptParameters* accessible as "%1," etc.

ScriptParameters

Parameters to pass into the *ScriptFile*.

Return Codes

0 Success

1 Parameter error

2 SOCKETS not loaded

3 User aborted

4 Transfer aborted

5 Error writing local file

6 Error reading local file

Other Server returned error response code; to find that error code, add 390 to the response code returned by FTP. The result will always be greater than or equal to 400 in this case.

Example

```
FTP /n FTP.cdrom.com /f=getfile.scr /.2/simtelnet/msdos DIRS.TXT
```

```
(The file GETFILE.SCR):
user anonymous
pass root@
cd %1
binary
get %2
quit
```

FTP Commands

The commands entered at the FTP client can be interpreted and translated to standard FTP commands to be sent to the server. The FTP server might recognise more, or less, commands than the standard list of commands as specified in RFC 959. The **site** command is always server dependent. Some of the standard commands are implemented differently in various servers.

Useful things to note are:

1. The **put** and **get** commands allow multiple file transfers by usage of wild card characters. When **get**ting files with paths or long names, no translation of foreign file names are done. Specify a valid DOS ***local_file*** name.
2. A short directory list (NLST) is obtained by **ls** and the long list with **dir**.
3. Some of the commands can be abbreviated.
4. Some commands are aliases added for user comfort like **bye**, **exit** and **quit**; **get** and **mget**; and **put** and **mput**.
5. The optional **[*local_file*]** parameter will, when specified, cause the output of that command to be logged to a file. By specifying the file as PRN you can get immediate printouts.
6. On some servers you might specify the optional **[*remote_file*]** parameter as PRN or the printer output device to do remote printing. (See also the **site nopath** command for the SOCKETS FTP server.)
7. The **F3** key and **spacebar** can be used to recall the last command word by word.

Below is a list of commands recognised by the SOCKETS FTP client (some FTP servers might not offer all the facilities):

Command	Description
abort	Cancel an incomplete transfer
append	"Put" a file at the server but append it if the file exists
ascii	Synonym for type a
binary	Synonym for type i
bye	Synonym for quit
cd directory	Synonym for cwd
cwd directory	Change server directory

dele file	Delete a server file
dir [file l directory [local_file]]	Synonym for list
exit	Synonym for quit
get remote_file(s) [local_file]	Transfer a file from the server in the current mode (type)
image	Synonym for type i
ls [file l directory [local_file]]	Synonym for nlst
lcd directory	Perform a local change directory
ldir [file l directory]	Give a local directory listing
list [file l directory [local_file]]	Give a long directory listing
mget remote_file(s) [local_file]	Synonym for get
mkdir remote_directory	Create a server directory
mput local_file(s) [remote_file]	Synonym for put
nlst [file l directory [local_file]]	Give a short names-only directory listing
pass [password]	Password for username
pasv [on | off]	Report or change the status of the passive transfer mode to enable firewall friendly file transfers. (The SOCKETS FTP client always tries to switch passive mode on at the start of a session.)
put local_file(s) [remote_file]	Transfer a file to the server in the current mode (type)
Pwd	Print working directory at server
quit	Terminate FTP session
quote remote_command [args ...]	Send a command to the server without any interpretation
rmdir remote_directory	Remove (delete) a server directory
rnfr existing_filename	Rename a file, command 1 of 2
rnto new_filename	Rename a file, command 2 of 2
site sub-command	Send server specific commands
size file	Report the file size in bytes as a 213 message
shell	Shell to DOS for IFTP.EXE
stat	Report the status of a transfer or active connections
System	Return operating system information from the server
type [i I a]	Report or select the file transfer mode: image (binary) or ASCII
user [username]	Username to logon
verbose [on | off]	Verbose mode reports more of the FTP negotiations

GETMAIL

GETMAIL retrieves all of the messages from a POP3 (Post Office Protocol version 3) Internet mail server. Each message is stored as an individual file on the local machine. *GETMAIL* also creates a log file to indicate successful downloads or errors.

Syntax

GETMAIL server user password

Options

Server

The IP address or DNS name of the Internet mail server from which to download messages. The messages that are downloaded are named by sequential file number and are placed in the current working directory.

User

The username for server identification purposes.

Password

The secret for account authentication on the server.

Logging Format

Timestamp, Code String

Timestamp

Weekday Month Day Time Year

Code

Three digit integer. 000 means perfect success, 100-199 mean usage error and 200-299 means TCP/IP error from server.

String

Human–readable explanation of the error code.

Example

```
GETMAIL 10.0.0.1
GETMAIL 10.0.0.1 guest secret
```

HTTPGET

HTTPGET is a simple web client that can retrieve the contents of a URL to a local file.

Syntax

HTTPGET [-p] [–s] [–v]URL

Options

-p=Port

-s=Server

-v

localfile

Remarks

Port

Use to specify a remote port other than 80 to connect to.

Server

Use to specify a server name if the URL doesn't contain one.

-v

Display extra output for troubleshooting.

localfile

Rather than keeping the filename from the URL, the contents may be saved to a named file.

Example

```
HTTPGET http://www.datalight.com/images/logohead.gif
HTTPGET –v http://www.datalight.com/images/logohead.gif logo.gif
```

IFSTAT

IFSTAT displays the status of the Interface, Modem, Serial, PPP, and it displays the version information for SOCKETS.

Syntax

IFSTAT [i] [m] [p] [s] [v]

Options

The i option shows the Interface status.

The m option shows the Modem status.

The p option shows the PPP status.

The s option shows the Serial status.

The v option shows the version information.

Example

```
IFSTAT v
```

This will display the SOCKETS version information

IOCTL

IOCTL is a diagnostic utility for the SOCKETS IOCTL functions using the TCP/IP Basic API

Syntax

IOCTL *interface name*

Remarks

interface name The name of the interface to be tested.

Example

```
IOCTL if0
```

Tests the IOCTL functions through the interface “if0” based on the commands given during the diagnostics.

IOCTLH

IOCTLH is a diagnostic utility for the SOCKETS IOCTL functions using the TCP/IP Basic API. This version supports the hot-swappable functionality.

Syntax

IOCTLH *interface name*

Remarks

interface name The name of the interface to be tested.

Example

```
IOCTLH if0
```

Tests the IOCTL functions through the interface “if0” based on the commands given during the diagnostics.

IPSTAT

The IPSTAT utility returns statistics on IP and memory. Use IPSTAT to check for error conditions and memory problems.

Syntax

IPSTAT

Example

```
IPSTAT
```

The follwing will be displayed (The values may differ):

IP stats at 160F:04C8:

Total Packets	2671
Smaller than minimum size	0
IP header length too small	0
Wrong IP version	0
Unsupported protocol	0
Memory available	9016
Memory allocation failures	0
Memory free errors	0

Minimum stack observed 886

LPR

LPR is a printer client for UNIX-style printer servers. There is no matching **LPD** server for SOCKETS.

Syntax

LPR /s=Server /p=Printer /u=Agent Filename

Options

/r

/q

/l=Port

/h=LocalHostName

/c=JobClass

/j=JobName

/n

/t

Remarks

/q

Query Mode. Can be followed by agent names or job numbers to filter output.

/r

Remove Mode. May be followed by job numbers to specify jobs to remove.

/s=Server

Hostname of the print server.

/p=Printer

Name of the printer device on the server to be used for output.

/u=Agent

User name on the server. Used for identification.

Filename

Local file name to spool to the server.

/l=Port

Connect to the specified port on the server rather than the standard port number of 515.

/h=LocalHostName

Name of the local host for job identification purposes.

/c=JobClass

Name of the job class for job identification and scheduling purposes.

/j=JobName

Name of the job, for identification purposes; defaults to the local file name if not specified.

/n

Run without user interaction.

/t

Text filter. Strips all unprintable characters before printing.

Example

```
LPR /n /s=10.0.0.1 /p=prn0 /u=Tester output.dat
LPR /n /s=10.0.0.1 /p=prn0 /u=Tester /q
LPR /n /s=10.0.0.1 /p=prn0 /u=Tester /r
```

MAKEMAIL

MAKEMAIL packages the body text and any attachments for delivery using the **SENDMAIL** application.

Syntax

MAKEMAIL –tToAddress –fFromAddress –sSubject –bBodyTextFile -oOutputFile-aAttachment

Options

ToAddress

The e-mail address of the recipient(s) of this mail. Additional recipients are specified by repeated use of the **–t** parameter. If the *ToAddress* is a name that can be resolved by either the DNS server or host file then the *@servername* is not necessary.

FromAddress

Used to identify the sender of the message.

Subject

The subject line of the e-mail message.

BodyTextFile

The local file containing the body text of the e-mail message to deliver.

OutputFile

The local file name in which to store the prepared file for delivery by **SENDMAIL**. This file is overwritten if it already exists!

Attachment

The name of a local file to be binary attached to this e-mail message. Multiple attachments are created by repeated use of the **–a** parameter. Files are attached as MIME parts, encoded with the application/x-uuencode content type.

Example

```
MAKEMAIL -tfred@yahoo.com -fmary@yahoo.com –sStatus –bmessage.txt –omail.dat
MAKEMAIL –tfred –tbarney –fwilma –sDinner –bmenu.txt –omail.dat
```

```
MAKEMAIL –tfred –fwilma –sBowling –bbody.txt –aStone.jpg –aRock.jpg –
omail.dat
```

NETBIOS

NETBIOS is a TSR that provides a NETBIOS-compatible network interface.

Syntax

NETBIOS [options]

Options

/a=NameCount

/b=BroadcastFile

/c=NCBs

/l=LanAdapterNumber

/n=HostsFile

/s=SessionCount

/u

Remarks

/a=NameCount

Maximum number of names to cache.

/b=BroadcastFile

Local file containing IP addresses not on the local network segment that should be considered part of the “broadcast” group.

/c=NCBs

Number of NCBs to allocate.

/l=LanAdapterNumber

For the NETBIOS API, the adapter number of this interface.

/n=HostsFile

Local file name containing name-to-IP mappings.

/s=SessionCount

Number of simultaneous sessions to allow.

/u

Running **NETBIOS** again with this option unloads the TSR.

Configuration File

NETBIOS uses a file to statically map names to IP addresses. This is the *HostsFile*. The file contains space-separated parameters in the format:

name IP

NETBIOS uses a second file to list machines that should be added to the local "broadcast" group. This is the *BroadcastFile*. The file contains one parameter per line in the format:

IP

Configuration File Parameters

name

The local alias for this entry.

IP

The remote IP address of this entry.

Example

```
NETBIOS /n=lmhosts
```

PDTEST

The PDTEST utility is a diagnostic program that tests loaded packet drivers. PDTEST does not perform a full test since it does not transmit traffic through the drivers, but just checks their status. It reports such things as the interrupt vector of the packet driver, its class, the MAC address, and information on the status of the driver. PDTEST is useful for checking that the packet driver was actually loaded which interrupt was used, and the class of packet driver. Classes supported by SOCKETS are marked with an asterisk (*) in the following list of recognized classes:

Class 1*	DIX Ethernet_II
Class 3*	802.5 Token Ring
Class 5	Appletalk
Class 6*	SLIP
Class 9	AX.25 Amateur Radio
Class 11*	802.3 with 802.2 headers IEEE
Class 12	FDDI with 802.2 headers
Class 13	Internet X.25
Class 14	Northern Telecom LANSTAR encapsulating DIX
Class 16	Point to Point Protocol for serial lines
Class 17	802.5 Token Ring w/expanded RIFs

Most Ethernet packet drivers support both classes 1 and 11. Class 1 is the default class and should normally be used.

Syntax

pdtest

Example (output)

Packet driver found at 0x60

Version 9, class 1, type 57, number 0, functionality 6

Name: MAC/DIS converter

High performance driver

Rev 1.09 par_len 14 add_len 6 mtu 1514 multicast_buf 0

Rcv_bufs 0 xmt_bufs 0 int_num 0x0

Address: 00:00:c0:08:d7:15

Extended driver

Packets:in 511 out 595 bytes:in 112664 out 74476

Errors:in 0 out 0 packets lost 0

RC.JAR

RC.JAR is a Java remote console applet for connecting to hosts running **HTTPD**. It shows the text-mode contents of the **HTTPD** machine's display memory, and allows fully interactive keyboard input. This is an alternative to the **RCCLI** applet.

Syntax

The link to RC.JAR should be embedded in an HTML page served by the HTTPD server. An example is available in the CGI/ subdirectory of the HTTP applications.

Remarks

RC.JAR has been compiled for use with the Java Development Kit version 1.3.1. The browsers Mozilla 1.x and newer, as well as Microsoft's Internet Explorer 4.0 and newer are supported. In order to use with a Netscape Browser a security certificate must be compiled into RC.JAR.

RC_DK.JAR

RC_DK.JAR is a Java remote console applet specifically for Danish keyboards. Please see **RC.JAR** for more information.

RCCLI

RCCLI is a DOS remote console client for connecting to hosts running **HTTPD**, and is an alternative to the **RC.JAR** applet.. It shows the text-mode contents of the **HTTPD** machine's display memory, and allows fully interactive keyboard input. To exit the remote console client, press Ctrl-Alt-X.

Syntax

RCCLI server_address <tcp_port>

Options

server_address

Specify the IP address or DNS name of the machine running **remcon**.

tcp_port

tcp_port defaults to 81, but must be changed if a nonstandard remote console port is chosen for the remote console session during server configuration.

Example

```
RCCLI 10.0.0.1
```

SENDMAIL

SENDMAIL delivers e-mail messages packaged by the **MAKEMAIL** application to an Internet mail server. **SENDMAIL** also creates a local log file to indicate successful send or failures.

Syntax

SENDMAIL server file

Options

Server

The IP address or DNS name of the Internet mail server to receive the message.

File

The file, created by the **MAKEMAIL** utility, to deliver.

Logging Format

Timestamp, Code String

Timestamp

Weekday Month Day Time Year

Code

Three digit integer. 000 means perfect success, 100-199 mean usage error and 200-299 means TCP/IP error from server.

String

Human–readable explanation of the error code.

Example

```
SENDMAIL mail.datalight.com mail.dat
```

SETHOST

The SETHOST utility sets an environment variable (default HOSTNAME) to the name contained in a map file (default MACHOST.MF) according to the hardware address (MAC or Ethernet) found by searching for a packet driver tsr. Network management is simplified when using %HOSTNAME% in the SOCKET.CFG files to set IP addresses.

SETHOST can be used in either of two modes:

- To update the MACHOST.MF file or
- To set the HOSTNAME environment variable.

Syntax

sethost [/f=*n* | /n=*hostid*] [/c=] [/m= *map_file*] [/v=*variable*]

Options

without a *hostid* set the variable

/f=*n*

Use *n*th environment block (required for some systems - try /n=1 first.)

/n=*hostid*

This is the IP address of the local PC in symbolic form as in HOSTS or in decimal form to add to the mapfile

/c=

Preserve the case of the environment variable

/m=*map_file*

Use map_file instead of default MACHOST.MF

/v=*variable*

Use variable instead of the default name: SETHOST

Note: The equal signs are required in this case since SETHOST supports applications where the n= is optional. A slash without an equal sign indicates the setting of a host id.

Example

To modify or add an entry in MACHOST.MF:

```
sethost /n=ws_name
```

To set the variable HOSTNAME:

```
sethost
```

Installing SETHOST

Installing SETHOST requires a working installation of SOCKETS. Also, all workstations run SOCKETS from a server disk.

Edit the HOSTS file to add all the workstations' names and IP addresses. We recommend that all the workstation IP addresses be preceded with an asterisk to make them hidden to other users looking into the list of hosts. For example, *198.147.35.120 admin03

Copy the SETHOST.EXE program and an empty file MACHOST.MF to your server, in a directory such as X:\SOCKETS, for example.

At each workstation, log in as supervisor (or have write access to X:\SOCKETS) and execute the DOS commands:

```
x:
CD \SOCKETS\UTILS
SETHOST /N=WS_NAME
SETHOST
```

```
SET
```

The SETHOST /N=WS_NAME command creates an entry in the MACHOST.MF file with the name of the workstation and its MAC (Ethernet card) address. This is the important "once-only" command. The next two commands are to verify that the ws_name is stored in the environment variable HOSTNAME.

In the AUTOEXEC.BAT or any batch file executed after login and before running SOCKETS, put the following commands:

```
x:
CD \SOCKETS\UTILS
SETHOST
```

The IP addresses are now linked to the MAC addresses of the network cards. If a network adapter or host system is changed, update the MACHOST.MF file with

```
sethost /ws_name
```

The MACHOST.MF file keeps the mapping from the MAC addresses to the workstation names and the HOSTS file maps the workstation name to its IP address.

The variable name HOSTNAME and filename MACHOST.MF are the defaults for SETHOST.EXE but can be user-specified. Run SETHOST /? to display the available options.

SNTPCLI

SNTPCLI is used for getting system network time information from a system network time protocol (SNTP) server.

Syntax

SNTPCLI *server* [minutes]

Remarks

server The server providing the time service.

Options

minutes An optional parameter which provides an adjustment to the server time in minutes.

Example

```
SNTPCLI 182.109.2.80
```

Retrieves the time information from the server specified.

Time Zones

An additional environment variable for time zones is also supported. The TZ variable allows SNTPCLI to compensate for time zone differences on the SNTP server when setting the local system time By default, if no time zone is set, SNTPCLI assume Greenwich Mean Time (GMT).

Setting the TZ variable is not necessary when using all programs. The syntax for the TZ environment variable is:

SET TZ= <*abbreviation*> +|- *value*

Abbreviation represents any three-letter abbreviation for the chosen time zone. The variable serves as a reminder to the user. For example, if setting the time zone for Pacific Standard Time, the variable could be set as PST and for Eastern Standard Time as EST. The abbreviation is only a placeholder in the syntax for the TZ variable. There are no incorrect abbreviation choices as long as only three letters are used.

Value represents the number of hours this time zone varies from GMT. For example, the west coast of the United States and Canada is –8 hours relative to GMT. This value may have to be adjusted to compensate for daylight savings time. There should be no spaces between the abbreviation, plus or minus sign, and the value. Some examples are:

```
SET TZ=PST -8
SET TZ=CMT -3
SET TZ=GMT+2
```

If an incorrect format for the time zone is entered, the default of GMT is used.

SPRINT

SPRINT is a simple printer client that prints through a SOCKETS host configured as a print server. To configure a SOCKETS host as a print server, see the **start prntsrv** command in Chapter 4, SOCKETS Configuration .

Syntax

SPRINT Server Filename [options]

Options

Port

Mask

Remarks

Server

Hostname of the print server.

Filename

Local file name to spool to the server.

Port

Connect to the specified port on the server rather than the standard port number of 10.

Mask

Used for status reporting, this parameter only works if the server was configured with a mask.

Example

```
SPRINT 10.0.0.1 output.dat
```

TCP

TCP.EXE is a utility used to examine and change the TCP parameters.

Syntax

TCP *close* n

TCP *irtt* [n]

TCP *mss* [n]

TCP *reset* n

TCP *retry* [n]

TCP *status* [/columns [refresh time]]

TCP *status* n

TCP *window* [n]

Remarks

close	Close connection specified by ‘n’.
irtt	Set, using ‘n’, or display the Initial Round Trip Time.
mss	Set, using ‘n’, or display the Maximum Segment Size.
reset	Reset connection ‘n’.
retry	Set, using ‘n’, or display the retry count.
status	Display summary status or all connections.
status n	Display detailed status for connection ‘n’.
window	Set, using ‘n’, or display window size.

Options

/columns	The number of columns to display status with manual or automatic refresh.
refresh time	The time in seconds after which an automatic refresh occurs.

Example

```
TCP irtt
```

Displays the current irtt value.

XPING

XPING starts a continuous string of pings until stopped by a keystroke.

Syntax

XPING *ip address* [interval]

Remarks

ip address	This may be a numeric address or a name address.

Options

interval The time to wait between pings in clock ticks.

Example

```
XPING 10.0.0.1 20
```

This will ping the address of 10.0.0.1 every 20 clock ticks.

Chapter 7, SOCKETS Server and Client Applications

HTTP Server

Overview

The SOCKETS HTTP server, HTTPD.EXE, is a small, fast, reliable and extendable web server that can run as either an application or TSR. Apart from the minimum required file download capability, the following additional capabilities are provided:

1. Remote Console Server- ability to gain terminal-type access to the server system, using a standard browser, without the need to install any software on the browser computer
2. Authentication – Both system wide and directory wise
3. CGI Extendibility – The ability to extend the server to create dynamic web pages, perform specialized tasks, etc.
4. A Server Side Includes (SSI) interface is provided using the CGI interface, enabling a user to create web pages using HTML templates with variable names, which is substituted in-time with specific values
5. Ability to run as a background process
6. Flexibility to control physical parameters such as memory usage and number of connections

Server

The HTTP server is used to send static web pages existing as files on the server or dynamically generated web pages to a remote client (browser). Dynamic pages can be generated in two ways:

1. Extension CGI. By calling an external CGI handler, the server provides an API to external handlers. A Server Side Includes (SSI) interface is provided as well, which makes it very easy to create powerful interactive web pages.
2. Spawning CGI. By spawning programs with a relatively short execution time to generate the pages through a mechanism similar to CGI, the basic mechanism used by CGI is that arbitrary programs can be spawned from the web server with input as received from the remote browser and output that can be sent to the browser.

The Remote Console Server accepts input from a remote client that is fed to the keyboard buffer for use by an arbitrary program using it. It also monitors the screen display buffer area and sends screen information to the remote client.

The SOCKETS password file controls authentication. Authentication is user specific and may also differ from directory to directory. It may also be put off for either some or all users. See the section on authentication.

The HTTP server can support multiple simultaneous sessions. The GET and POST request methods are implemented as well as the following MIME types:

text/html, text/plain, image/gif, image/jpeg, image/jpeg and application/octet-stream.

The MIME type is determined by the file extension.

Remote Console Server

Initialization

The client (browser) will initialize a remote session. An HTTP connection will be made to the HTTP server. The downloaded page will contain the applet that will automatically connect to the RCS on TCP port 81. An example download page is supplied as REMCON.HTM.

Almost any application e.g. a text editor can be run on the server. The remote keyboard and display control the application as if they were locally attached.

On the remote side, the Java Applet acts as a simple terminal emulator that displays what it receives from the server and sends what is entered from the keyboard to the server.

It is not required to have a real display adapter on the embedded system server, only to have display buffer memory.

When a new connection is made, all the screen data, as well as the cursor position, is sent to the client. Subsequently the RCS keeps a watch on the video memory and cursor position and whenever a change is detected, the RCS sends the changed data to the Java applet.

Keyboard data received from the client is passed to the keyboard buffer making it available as keyboard input for use by any application executing on the server.

Remote Console Client

The remote console client exists as a Java 1.3.1 applet, supplied as RC.JAR, and will function on any Java 1.3.1 compliant browser. Please note that a security certificate has not been compiled into RC.JAR so it is not compliant with versions of the Netscape browser that require a security certificate to run Java applets. A DOS based client using SOCKETS is also supplied as RCCLI.EXE. For additional information about RC.JAR or RCCLI.EXE, please see the Utility Description Chapter.

Extension CGI

The SOCKETS HTTP servers (HTTPD/HTTPFTPD) provide a facility to call functions in other modules which may be TSR or transient programs. These functions are referred to as “HTTPD extensions.” For more information please see the “ROM-DOS Developer’s Guide” section “CGI Application API.

Extension CGI Examples

Five very simple examples are included to demonstrate the implementation of CGI. Source code is included.

Put all *.htm* and *.exe* files in the %HTTP_DIR% directory and start *HTTPD*. Load all the cgi programs (you may use *cgi.bat*). All is in place now and the examples may be accessed through *index.htm.*

The first four examples may operate in one of two modes:

As a TSR (resident) program: this is the default behavior. At this stage unloading of the TSR is not supported. De-registration is possible by loading the program again. This routine may be repeated.

As a transient program: use '/t' command line switch to activate. This option will immediately spawn 'command.com'. From this prompt other cgi programs may be loaded. The program exits when 'command.com' is exited by typing 'exit' at the prompt.

These programs are:

1. ***cgiecho*** A very simple program that accepts data from a user and echoes it back nicely formatted. Get *echoform.htm* from the browser.

2. ***cgicount*** A page visit counter. Only updates between sessions if transient (cgicount /t) Get *num.htm* from the browser.

3. ***cgiform*** Does the same as the old 'fill out the form and submit' utility. Get *caform.htm* from the browser.

4. ***SSI*** A very simple SSI implementation that demonstrates the SSI interfaces. *Template.htm* is filled by some variables. Get *ssi.htm* from the browser.

The fifth example, ***FFUR***, (Form-base File Upload Receiver) is only a transient program, but can easily be adapted to be similar to the rest. It handles the upload of a file as a POST command by filling out *ffur.htm.*

Passive Mode

The server may be run in passive mode by specifying a '/p' command line switch. When passive, the server will record network events but only handle them once it is triggered by a CGI user.

Server Memory

The server's memory usage may be controlled in two ways:

1. By specifying the amount of memory when going TSR.

2. By specifying the maximum number of connections the server will allow.

Option 1 is the recommended option. Use Option 2 if you have 'heavy' web pages – usually the type where pages consist of frames and many images, etc. Connections are generally reset when

more connections are attempted than the defined maximum. The client then must retry to establish the lost connections, leading to a more distributed load on the server.

Spawning CGI

An external program, indicated by the requested URL, is spawned. All relevant information is passed as environment variables. The CGI program gets all input (e.g. posted data) from *standard in* and sends all response through *standard out*. Spawning CGI is discouraged in favor of Extension *CGI*. For more information please see the "ROM-DOS Developer's Guide" section "CGI Application API.

Authentication

Default authentication matches the capabilities of the FTP server as documented in the section "FTP Server" on page 87. A file called "SOCKET.UPW" should exist in the SOCKETS (environment variable) directory.

The default permission file controls remote console access. Each listed user has a single-letter privilege code set if he has privilege to use the Remote Console. The code should be missing if that user does not have Remote Console privilege.

An additional authentication feature is implemented - **htaccess**. This feature provides a per-directory permission override mechanism. It is enabled using '/t' as command line switch. If htaccess is enabled, the default mechanism may be skipped (but no default users or remote console access will be available).

A file called HTACCESS (typically hidden) contains authentication overrides to enable partial anonymous access or additional password security to subdirectories, etc. If this feature is activated, the server code will look for HTACCESS files in each directory starting from the requested path and continuing upward in the directory structure (assuming the root directory to be at the top) until an HTACCESS file is found. If no file is found, then the default settings are used. An anonymous access entry is available for the developer to specify that some subdirectory is authorized for any user, although its parent directory is password-protected. CGI scripts can also be controlled via the HTACCESS mechanism.

HTTPD Program

The syntax for HTTPD is:

HTTPD [options] [<http_port>] [<rc_port>]

Any combination of these switches may be used. They should be separated by at least one space.

Option	Description
/? /h	display help screen
/r	run server in TSR mode
/s	display server status
/t	enable htaccess directory level authentication
/u	unload if resident
/c	close listen

/d	do not start remote console
/g	allow old type (spawning) CGI
/p	Passive mode
/i=<InterruptNumber>	Interrupt number for cgi API
/m=<MemorySize>	set memory size
/n=<MaximumConnections>	number of simultaneous connections>
/a=<ScreenX>, <ScreenY>	set screen aspect
/v=<ScreenBufferSegment>[:<ScreenBufferOffset>]	set video buffer address (hex)
/k	Unload and abort all active connections

Remarks

ScreenX, ScreenY

The width and height of the screen area to serve for the remote console session. These values default to 80 and 25, respectively.

ScreenBufferSegment, ScreenBufferOffset

Together, a pointer to the top-left corner of the display memory to serve for the remote console session. These values default to B000 and 0000 respectively, for monochrome display adapters and to B800 and 0000 respectively, for color display adapters.

MemorySize

The maximum amount of memory available to the server. The default value is 32K. The value of m can range from 8192 to 63472.

MaximumConnections

The maximum number of simultaneous connections allowed by the server.

InterruptNumber

The interrupt number to access the CGI API.

http_port

HTTP port to listen on. This parameter defaults to the standard HTTP port number of 80.

rc_port

Remote Console port to listen on. This parameter defaults to 81.

The “root” directory for web content is the current directory when HTTPD is started. This can be changed by setting an environment variable HTTP_DIR e.g.

SET HTTP_DIR=D:\SERVER\WEB

Format of "SOCKET.UPW"

This is the same file used for the FTP server’s (*FTPD.EXE*) permissions. This file consists of lines where each line contains a user's information. A line starting with a # is considered a comment and is ignored. Each line consists of four fields:

<Username> <Password> <Working Directory> <Permissions> [# comment]

Username: The name of this user. If it is *, it will be used when the client does not specify a username.

Password: This user's password. If it is *, no password is required

Working Directory: The user will only have access to this directory and its subdirectories. If it is '/', this user has access to the whole system. HTTP_DIR can be referred to as '\'. If a relative path is specified, it is appended to HTTP_DIR.

Permissions: IMPORTANT when a user is granted both FTP and HTTP permissions, the FTP permissions must appear **first**, otherwise they will be ignored. Operations allowed. May contain any combination of the following tokens:

e - User may 'g**e**t' files
p - User may '**p**ost' files
g - User may use c**g**i
m - User may use Re**m**ote Console

Fields should be separated by single spaces. If any field is missing the entry is ignored. A comment may follow the last field (permissions) of the line.

Note: If a default user is supplied, it should always appear first in the list of users. Only users below the default user will be considered.

Format of "htaccess"

Any directory may contain this file, and serve as overrides to the general permissions for the containing directory and all its subs until another htaccess is found. This file consists of lines where each line contains a user's information. A line starting with a # is considered a comment and is ignored. Each line consists of three fields:

<Username> <Password> <Permissions> [# comment]

username: The name of this user. If it is *, it will be used when the client didn't specify a username.

Password This user's password. If it is *, no password is required.

Permissions Operations allowed. may contain any combination of following tokens:
e - User may 'g**e**t' files
p - User may '**p**ost' files
g - User may use c**g**i

Fields should be separated by single spaces.If any field is missing the entry is ignored. A comment may follow the last field (permissions) of the line.

Note: If a default user is supplied, it should always appear first in the list of users. Only users below the default user will be considered.

FTP Server

FTPD is a file server that can run either as an application or as a TSR. The name of the server as displayed in the banner is determined by the HOSTNAME environment variable. If the environment variable is not set, the name "Socket" is used. The user password file, SOCKET.UPW, in the SOCKETS directory (indicated by the SOCKETS environment variable) controls access.

A temporary file is created when a directory listing is requested. This file is created in the current directory, but can be created in any directory as specified in the FTPDIR environment variable.

FTPD Program

The syntax for FTPD is:

FTPD [options] [<ftp_port>]

Option	Description
/? /h	display help screen
/r	run server in TSR mode
/s	display server status
/u	unload if resident
/c	close listen
/m=<MemorySize>	set memory size
/n=<MaximumConnections>	number of simultaneous connections
/k	Abort all active connections and unload

Remarks

MemorySize

The number of bytes of memory available to the server. This value defaults to 32768.

MaximumConnections

The maximum number of simultaneous connections allowed by the server.

ftp_port

FTPD will listen on the listed port. This parameter defaults to the standard FTP port number of 21.

Configuration File

FTPD uses the standard SOCKET.UPW file for validating logins. The file is composed of text lines, each representing a login name, password, and the configuration to use for a session opened with those credentials. Space characters separate the parameters in the file, which are in the following format:

name password directory rights

The location of the username/password file to be used by the server is specified by the environment variable SOCKETS as follows:

%SOCKETS%\SOCKET.UPW

If the variable SOCKETS is not specified, the following file is used:

\DL\SOCKETS\SOCKET.UPW

Configuration File Parameters

name

The login name of this record.

password

The password to authenticate a user trying to login as this name.

directory

The starting directory for this user.

rights

Up t o four characters specifying which permissions this user is granted:

r means that this user has read access.

w means that this user has write access.

c means that this user has permission to make new directories.

d means that this user has permission to change to a directory other than his starting location and subdirectories from the starting location.

Example Socket.upw

```
Admin admin c:\ drwc
Guest * c:\guest dr
```

Example Command Line

```
FTPD /m=40000 /r
```

FTP Server Commands

The following commands are recognised by the SOCKETS FTP server:

Command	Description
abort	cancel an incomplete transfer
append	"put" a file at the server but append it if the file exists
cwd *directory*	change server directory
dele *file*	delete a server file
list [*file* l *directory*]	give a long directory listing
mkd *remote_directory*	create a server directory
nlst [*file* l *directory*]	gives a short names-only directory listing
pass [*password*]	password for username
pasv [on | off]	report or change the status of the passive transfer mode to enable firewall friendly file transfers. (The SOCKETS FTP client always tries to switch passive mode on at the start of a session.)
retr *remote_file*	transfer a file from the server in the current mode
stor *local_file*	transfer a file to the server in the current mode
pwd	print working directory
quit	terminate FTP session
rmd *remote_directory*	remove (delete) directory
rnfr *existing_filename*	rename a file, command 1 of 2

rnto *new_filename*	rename a file, command 2 of 2
site [path I nopath]	use full path description (see
site raw [*interface*]	open a session to a raw host using one of the raw lines (interfaces) specified
site *sub-command*	command to be passed on to raw host
size *file*	report the file size in bytes as a message prefixed with 213
stat	report the status of a transfer or active connections
system	return operating system information from the server
type [i I a]	report or select the file transfer mode: image (binary) or ASCII
user [*username*]	username to logon

Combined HTTP and FTP Server

HTTPFTPD is a combined HTTP and FTP server that can run either as an application or as a TSR. By default, it processes normal HTTP requests on port 80 and normal FTP requests on port 21. It also serves a proprietary session displaying the contents of text-mode display memory to the **RC.JAR** and **RCCLI** client applications. This feature is commonly called the "remote console."

If the **HTTPFTPD** server is loaded as a DOS TSR program, set the environment variable, HTTP_DIR, to the location of the INDEX.HTML file; for example, SET HTTP_DIR=C:\DL\SOCKETS\SERVER

HTTPFTPD Program

The syntax for FTTPD is:

HTTPFTPD [options] [<http_port> [<ftp_port> [<rc_port>]]]

Any combination of these switches may be used. They should be separated by at least one space.

Option	Description
/? /h	display help screen
/r	run server in TSR mode
/s	display server status
/t	enable htaccess directory level authentication
/u	unload if resident
/c	close listen
/d	do not start remote console
/g	allow old type (spawning) CGI
/p	Passive mode
/i=<InterruptNumber>	Interrupt number for cgi API
/m=<MemorySize>	set memory size
/n=<MaximumConnections>	number of simultaneous connections
/a=<ScreenX>, <ScreenY>	set screen aspect
/v=<ScreenBufferSegment>[:<ScreenBufferOffset>]	set video buffer address (hex)
/k	Abort all active connections and unload

Remarks

ScreenX, ScreenY

The width and height of the screen area to serve for the remote console session. These values default to 80 and 25, respectively.

ScreenBufferSegment, ScreenBufferOffset

Together, a pointer to the top -left corner of the display memory to serve for the remote console session. These values default to B000 and 0000 respectively, for monochrome display adapters and to B800 and 0000 respectively, for color display adapters.

MemorySize

The maximum amount of memory available to the server. The default value is 32K. The value of m can range from 8192 to 63472.

MaximumConnections

The maximum number of simultaneous connections allowed by the server.

InterruptNumber

The interrupt number to access the CGI API.

http_port

HTTP port to listen on. This parameter defaults to the standard HTTP port number of 80.

ftp_port

FTP port to listen on. This parameter defaults to the standard FTP port number of 21

rc_port

Remote Console port to listen on. This parameter defaults to 81.

Configuration File

HTTPFTPD uses the standard SOCKET.UPW file for validating logins. The file is composed of text lines, each representing a login name, password, and the configuration to use for a session opened with those credentials. Space characters separate the parameters in the file, which are in the following format:

name password directory rights

The location of the username/password file to be used by the server is specified by the environment variable SOCKETS as follows:

%SOCKETS%\SOCKET.UPW

If the variable SOCKETS is not specified, the following file is used:

\DL\SOCKETS\SOCKET.UPW

Configuration File Parameters

name

The login name of this record.

password

The password to authenticate a user trying to login as this name.

directory

The starting directory for this user.

rights

May contain any combination of the following characters specifying which permissions this user is granted (FTP rights must be specified first.):

r means that this user has read access.

w means that this user has write access.

c means that this user has permission to make new directories.

d means that this user has permission to change to a directory other than his starting location and subdirectories from the starting location.

e means that this user may 'g**e**t' files

p means that this user may '**p**ost' files

g means that this user may use c**g**i

m means that this user may use Re**m**ote Console

Example Command Lines

```
HTTPFTPD /m=40000 /r
HTTPFTPD /a=80,25 /v=a000:0000 /r
```

Appendix A, Packet Drivers

Overview

SOCKETS provides support for a wide range of LAN adapters supporting the Packet driver specifications.

The packet driver specification is adhered to by many vendors of LAN adapters and is also freely available as public domain software from the Internet e.g. from www.crynwr.com. SOCKETS only supports the packet driver specification, but public domain converters or "shims" are available to convert from NDIS or ODI to packet driver. Some "shims" may be obtained from Datalight Inc. but are not distributed with SOCKETS. SOCKETS may use up to eight packet drivers.

Using the same network board, SOCKETS can operate simultaneously with other network operating systems such as Novell, Microsoft, Banyan and others.

Note: Software configuration of network hardware, such as Plug and Play settings or packet drivers, must be completed before SOCKETS is loaded. Failure to properly load and configure the appropriate software results in error messages from SOCKETS. To help you isolate which device is failing; the **interface** command is referenced in those error messages.

For development using Microsoft Windows, it is recommended to purchase an NDIS3 Virtual Packet Driver from www.danlan.com . Versions are available for both Windows 9X and Windows NT/2000. Use a different IP address for SOCKETS than that used by Windows.

Note: SOCKETS can be run in a Console Window (DOS box) using Windows NT/2000, but problems may be experienced scheduling it unless constantly calling an API function. XPING can be used for this purpose to run TSR servers.

The supplied utility PDTEST.EXE can be used to test or diagnose your packet driver before attempting to start SOCKETS. For further information on PDTEST, refer to Appendix ?? "Managing the Network and Troubleshooting."

Packet Driver Installation

After you have installed SOCKETS, copy your NIC (Network Interface Controller) packet driver, from the NIC Driver disk. The various manufacturers supply their own packet drivers that may differ from what is documented here. Always consult the software and documentation supplied with your network controller first.

Note: Many PCI packet drivers require no command line parameters to load and, by default, set up an interrupt vector of 0x60.

Using a Memory Manager with a Packet Driver

If a board using mapped memory (for example, SMC) is used with an upper memory block manager (EMM386), the shared memory must be excluded:

```
DEVICE=C:\DOS\EMM386.EXE X=D000-D3FF
```

Packet Driver over ODI Driver Installation

If you already have Novell using ODI installed, just modify AUTOEXEC.BAT and the existing NET.CFG, otherwise make a \ODI directory on your hard drive and copy the following files to it:

Filename	Location
LSL.COM	Novell WSGEN floppy disk
NETX.EXE or VLM.EXE	Novell WSGEN floppy disk
IPXODI.COM	Novell WSGEN floppy disk
ODIPKT.COM	NIC Driver disk *
NIC ODI Driver	NIC Driver disk

*** Note:** ODIPKT.COM may not be provided by your vendor. In that case, there are many solutions available on the Web – just do a search for "odipkt.com".

Examples of ODI Drivers:

SMC8000.COM, SMCPLUS.COM, NE2000.COM, 3C5X9.COM or 3C503.COM

If you do not have a NET.CFG file, create an ASCII text file in the \ODI directory called NET.CFG that should look similar to the following:

```
Link Support
   buffers 8 1600
Protocol IPX
   Bind ODI_driver
Link Driver ODI_driver
   int irq
   port io port
   frame ETHERNET_802.3
   frame ETHERNET_II
```

The important parts to check are the buffers in the Link Support section and the order of the Ethernet Frame (also called Envelope Type) lines. Each Frame line specifies a logical board,

starting from number 0. ODIPKT should link to the Ethernet_II board, which, in this example, would be board number 1. The file mu st contain at least the following:

```
Link Support
   buffers 8 1600
Link Driver ODI_driver
   frame ETHERNET_II
```

Example for SMC:

```
Link Support
   buffers 8 1600
Link Driver SMCPLUS
   Port #1 280
   mem #1 000D0000 2000/10
   Int #1 3
   frame ETHERNET_802.3
   frame ETHERNET_II
```

Add the following lines to your AUTOEXEC.BAT file:

```
CD\ODI
LSL
rem Your NIC ODI driver
ODIPKT 1 96
IPXODI
NETX
F:
```

The syntax for ODIPKT.COM is:

ODIPKT logical_board interrupt_vector

The interrupt vector must be specified in decimal; for example 0x60 = 96.

logical_board is the index of the Frame type entry starting at 0. The normal frame type to use with SOCKETS on Ethernet is ETHERNET_II which is the second entry (or logical board 1) in the preceding example.

Example

```
CD\ODI
LSL
SMCPLUS
ODIPKT 1 96
```

Using a Memory Manager with an ODI Driver

If a board using mapped memory (for example, SMC) is used with an upper memory block manager (for example, EMM386), the shared memory must be excluded as follows:

```
DEVICE = C:\DOS\EMM386.EXE X=D000-D3FF
```

Packet Driver over NDIS2 Driver Installation

After you have installed SOCKETS, make a \LANMAN directory on your hard disk and copy the following files to it:

Filename

PRO.MSG
PROH.MSG
PROTMAN.DOS
DIS_PKT.DOS
NETBIND.COM
NDIS 2 Driver for NIC

Add the following lines to your CONFIG.SYS file in the following order and not separated by any other command:

```
DEVICE=C:\LANMAN\PROTMAN.DOS
DEVICE=C:\LANMAN\<Your NIC's NDIS driver>
DEVICE=C:\LANMAN\DIS_PKT.SYS
```

Example

```
DEVICE=C:\LANMAN\PROTMAN.DOS
DEVICE=C:\LANMAN\SMC8000.DOS
DEVICE=C:\LANMAN\DIS_PKT.DOS
```

Add the following line to your AUTOEXEC.BAT file

```
C:\LANMAN\NETBIND
```

Create an ASCII text file called PROTOCOL.INI in the \LANMAN directory to pass parameters to the various drivers:

```
[PROTOCOL MANAGER]
   DRIVERNAME=PROTMAN$
[PKTDRV]
   DRIVERNAME=PKTDRV$
   BINDINGS=<label_name>
   INTVEC=0x60
[label_name]
   DRIVERNAME= Your NIC's NDIS driver name
   IRQ=irq
   RAMADDRESS= ram_base_addr
   IOBASE= io_base_addr
```

Example

```
[PROTOCOL MANAGER]
   DRIVERNAME=PROTMAN$
[PKTDRV]
   DRIVERNAME=PKTDRV$
   BINDINGS=WDMAC
   INTVEC=0x60
[WDMAC]
   DRIVERNAME=WDMAC$
   IRQ=3
   RAMADDRESS=0xD000
   IOBASE=0x280
```

Using a Memory Manager with an NDIS Driver

If a card using mapped memory (e.g. SMC) is used with an upper memory block manager (e.g. EMM386), the shared memory must be excluded as follows:

```
DEVICE = C:\DOS\EMM386.EXE X=D000-D3FF
```

For more information on the NDIS drivers consult your network operating system documentation.

Appendix B, Network Management and Troubleshooting

This chapter describes solutions to common LAN problems, both configuration and performance.

Network Management

The Network Manager is expected to:

- Setup each SOCKETS application according to user requirements.
- Monitor the status of various connections and trace traffic as it traverses the network, in order to resolve problems.
- Modify the software configuration to align it with changes in the physical environment on which it runs.

Configuration Case Studies

Managing Host Names on a File Server-Based LAN

The SETHOST.EXE program manages the host names on a file server based LAN. The purpose of running SETHOST.EXE is to keep all the IP addresses in a single file on the server and allow all workstations to run the same software setup and yet maintain a unique IP address at each workstation. An alternative for non-file server-based networks is to use BOOTP or DHCP where a suitable server is available.

Installing SETHOST

Edit the HOSTS file to add all the workstation names and IP addresses. We recommend that all the workstation IP addresses be preceded with an asterisk to make them hidden to other users looking into the list of hosts. For example,

```
*198.147.35.120    admin03
```

Copy the SETHOST.EXE program and an empty file MACHOST.MF to your server in a directory that is named, for this example, X:\SOCKETS.

At each workstation, log in as supervisor (or have write access to X:\SOCKETS) and execute the DOS commands:

```
x:
cd \SOCKETS\DOS
SETHOST /N=WS_NAME
SETHOST
SET
```

The SETHOST /N=WS_NAME command creates an entry in the MACHOST.MF file with the name of the workstation and its MAC (Ethernet board) address. This is the important “once-only”

command. The next two commands are to verify that the *ws_name* is stored in the environment variable HOSTNAME.

In the AUTOEXEC.BAT, or any batch file executed after login and before running SOCKETS, put the following commands:

```
x:
CD \SOCKETS\DOS
SETHOST
```

In SOCKETS, set the IP address in SOCKET.CFG as follows:

```
ip address \HOSTNAME\
```

The IP addresses are now linked to the MAC addresses of the network cards. If you change a network board or swap a PC, must update the MACHOST.MF file with

```
sethost /ws_name
```

The MACHOST.MF file keeps the mapping from the MAC addresses to the workstation names and the HOSTS is used to map the workstation name to its IP address.

The variable name HOSTNAME and filename MACHOST.MF are the defaults for SETHOST.EXE but they can be user specified. Execute sethost /? to see the available options. See also ".

System Timer Interrupt Use

The timer interrupt (INT 1CH) and the hardware interrupt vectors specified in the ***interface*** command(s) or found automatically in Packet Drivers by SOCKETS, are hooked. Whenever such an interrupt occurs or when the API is called, the SOCKETS *scheduler* is called. The *scheduler* looks for queued incoming packets (queued at interrupt time by the Packet Driver or serial port driver) as well as the timer queue for timeouts, such as not receiving a TCP acknowledgment in time.

Packets are generally sent when a packet is received or the API is invoked, unless an ARP resolution is in progress, the TCP connection is not yet established, the Nagle heuristic is in operation or the offered window does not allow it. UDP packets are generally sent immediately when received by the API unless an ARP resolution is in progress.

Whenever SOCKETS is executing, it sets a BUSY flag and when the API is called during this time, which is only possible if it is called from an interrupt service routine, the API call fails with ERR_RE_ENTRY. Testing the BUSY flag before executing an API call can circumvent this error. The address of the BUSY flag can be obtained by the GET_BUSY_FLAG low-level API call.

The timer interrupt is assumed to be the standard PC timer interrupt with a period of approximately 55 ms. The exact period of the timer interrupt at INT 1CH is not important, but modifying the period will affect the scheduling of the kernel.. Too long a period may cause erratic behaviour and too short a period may cause excessive overhead. The important value for timing purposes is the BIOS timer variable at absolute location 046CH. The DWORD at 046CH MUST increment at a rate of 65536 increments per hour. (About 18 per second).

Advanced Network Configuration

SOCKETS offers additional networking capabilities for large networks. Using the Routing Information Protocol (RIP), SOCKETS can be configured to be aware of multiple IP routers/gateways.

BOOTP servers are detected when SOCKETS is started without specifying an IP address or an address of 0.0.0.0. To trace the BOOTP negotiations, use a trace all **iodt** command in your .CFG file before defining any interfaces.

DHCP servers are detected and used when SOCKETS is started with an IP address of 0.0.0.1.

For file server linked networks the SOCKETS utility SETHOST can be used to centralize management of IP addresses. SETHOST maintains a file mapping of the Ethernet (MAC) address of each machine to an IP address.

SOCKETS' Alternative Routing feature allows more than one route to be specified to a particular host or network. Failure of one route causes an automatic switch to the next route. The failed route is tested periodically and used again when it becomes available.

Tuning TCP/IP

Tuning a computer is a trade-off between the speed of operation and the amount of memory it uses. Performance depends on the number of TCP connections for the computer; more sessions require mo re memory.

TCP Retry Strategy

If there is a delay in your network connections, SOCKETS employs an intelligent retry strategy. A retry is attempted after a retry interval that gets longer as a function of the number of the retry. The first retry is attempted after the current RTT (Round Trip Time) plus one PC clock tick (18 milliseconds) has elapsed without a response. SOCKETS calculates the RTT as a smooth average of past measured RTTs, starting with the IRTT on a new connection.

For the first five intervals, the time doubles, giving interval lengths of 2, 4, 8, 16 and 25 times the RTT. Then, the square of the interval number is used starting with 5-squared, giving 25 times RTT. Consequently, increasing the number of retries can cause the total elapsed time to become quite long. More than 255 retries results in an infinite number of retries, causing connections to never time out.

When a SOCKETS station starts up, it sends a broadcast ARP request with its IP address to check for duplicate IP addresses. Any SOCKETS client in a retry mode picks up this ARP. Then it retries immediately without waiting for the next scheduled retry time.

Number of retries	1	2	3	4	5	6	7	8	9	10	11	12
Interval length (in RTT)	1	2	4	8	16	25	36	49	64	81	100	121
Total time (in RTT)	1	3	7	15	31	56	92	141	205	286	386	507

To get the current RTT in use for a connection n, use the **tcp status n** command that gives the smoothed average RTT indicated by SRTT.

Keep-alive

All SOCKETS servers test established connections after a minute of no-traffic by sending an empty packet with a decreased sequence number and waiting for the acknowledgement. If the server does not receive an acknowledgement it retries using the retry strategy described above with the smoothed Round Trip Time (RTT). When all the retries have failed, the connection has timed out and the server resets that connection. This prevents servers from hanging in a connected state when the remote client has stopped responding.

Troubleshooting

Problems with LICENSE.DAT File

For SOCKETS demo only (socketmd.exe and socketpd.exe).

The SOCKET environment variable is required in the demonstration version to indicate the directory with the LICENSE.DAT file, which contains the license information.

The LICENSE.DAT file contains your license information in demonstration versions. When this error message appears, the most frequent problem is that the file is in the wrong directory. SOCKETS looks in the SOCKETS directory (as given in the SOCKET environment variable) for the LICENSE.DAT file and then (to accommodate simple new installations) in the \SOCKETS directory of the current drive. If the file contents have been damaged, it will not run. The information in the LICENSE.DAT file is displayed when you start. The number next to SL indicates the number of concurrent users allowed.

XPING

The XPING utility is the most basic test to see if connections are working. Always first try to ping a host that does not seem to respond. Failure to get a response to a ping can usually be traced to one of the following reasons:

- Network cable not plugged in.
- Incomplete bind of the drivers on the local machine. (Carefully check all of the diagnostic messages while booting).
- Inadequate routing information. It may also mean an invalid return route somewhere.
- Lost packages along the route. Sending many ping requests and get only some back; could relate to network hardware problems.
- Remote host is not responding (not switched on, software not loaded, not connected, and so on).

A ping response displays the time taken to get a response. Note that the timing depends on the clock ticks. In an 80x86 PC these ticks are approximately 55ms, so limit the accuracy of the reported times to 55ms.

Utility Programs

The utilities described in the following sections will help you to configure and test your SOCKETS installation.

PDTEST, Packet Driver Test Utility

The PDTEST utility is a diagnostic program that tests loaded packet drivers. See PDTEST in "**Chapter 6, SOCKETS Utility Descriptions**" for more information.

SETHOST, IP Address Maintenance Utility

The SETHOST utility sets an environment variable (default HOSTNAME) to the name contained in a map file (default MACHOST.MF) according to the hardware address (MAC or Ethernet) found by searching for a packet driver tsr. See SETHOST in "**Chapter 6, SOCKETS Utility Descriptions**" for more information.

IPSTAT, IP and Memory Statistics Utility

The IPSTAT utility returns statistics on IP and memory. See IPSTAT in "**Chapter 6, SOCKETS Utility Descriptions**" for more information.

SOCKETS Glossary

Address Mask (also referred to as NetMask)

A bit mask used to select bits from an IP address for subnet addressing. The mask is 32 bits long, and selects the network portion of the IP address and one or more bits of the local portion.

ANSI (American National Standards Institute)

A group that defines U.S. standards for the information processing industry. ANSI participates in defining network protocol standards.

API (Application Program Interface)

An API is a specification of the methods an application programmer can use to access services provided by a software module. In the case of a network, the API specifies the interface to the network software. In TCP/IP, the idea of a "Socket" as the endpoint of a connection is used. A "socket" then refers to an abstraction to define the endpoint of a connection as far as the API is concerned. A socket can be created, opened, read, written, closed, and deleted in much the same way a file is handled in DOS. The difference is that two sockets must exist, normally on two hosts, before a connection can be made. A read operation on one side must always have a matching write operation on the other side.
A common way of interfacing a terminal emulator to networking software in a PC is to use Interrupt 14h. This is the PC BIOS entry point for serial port support, but when used for networking purposes, the original entry point is reused to provide a similar, but much expanded function. In addition to the native character at a time transfer, block transfers are also offered to increase throughput.

ARP (Address Resolution Protocol)

The TCP/IP protocol used to dynamically bind a high-level IP Address to a low-level physical hardware address. ARP is used across a single physical network and is limited to networks that support hardware broadcast.

Baud

Literally, the number of times per second the signal can change on a transmission line. Commonly, the transmission line uses only two signal states making the baud rate equal to the number of bits per second that can be transferred. The underlying transmission technique may use some of the bandwidth, so it may not be the case that users experience data transfers at the line's specified bit rate.

BIOS

Basic Input Output System – software that interfaces directly with the hardware.

BIOS extension

A short program that the BIOS recognizes and executes as the BIOS initializes the system.

Boot

Booting is restarting and reloading DOS. A PC can be booted by turning it off and then turning it on or by pressing the Ctrl, Alt, and Del keys simultaneously.

Bootable disk

A system disk that contains the files necessary to start and run the computer.

BOOTP (Bootstrap Protocol)

A protocol a host uses to obtain startup information, including its IP address, from a server.

Broadcast

A packet delivery system that delivers a copy of a given packet to all hosts that attach to it is said to broadcast the packet. Broadcast may be implemented with hardware or software.

Built-in device

Built-in device is an input/output device which is part of the DOS kernel.

CSLIP (Compressed Serial Line Internet Protocol)

CSLIP is an enhancement of SLIP by implementing Van Jacobson header compression. CSLIP uses more memory than SLIP but provides better throughput and faster response times, especially on small packets.

Datagram

The basic unit of information passed across a TCP/IP connection. An IP datagram is to an Internet as a hardware packet is to a physical network. It contains a source and destination address along with data.

DHCP (Dynamic Host Configuration Protocol)

A protocol that a host uses to obtain all necessary configuration information including IP address.

DNS (Domain Name Server)

The on-line distributed database system used to map human-readable machine names into IP addresses. DNS servers throughout the connected Internet implement a hierarchical namespace that allows sites freedom in assigning machine names and addresses. DNS also supports separate mappings between main destinations and IP addresses.

Domain

A part of the DNS naming hierarchy. Syntactically, a domain name consists of a sequence of names separated by periods.

DOS

Disk Operating System – an operating system that relies on disks for file storage.

DOS kernel

The DOS kernel is the part of DOS that handles a standard DOS call (Int 21h). It handles opening, reading/writing of files, loads programs, and manages memory.

FAT

File Allocation Table – a data table which allows DOS to keep track of file location on the disk so that they can be accessed by programs running on DOS.

Flow control

Control of the rate at which hosts or routers inject packets into a network or Internet, usually to avoid congestion.

FTP (File Transfer Protocol)

The TCP/IP standard, high-level protocol for transferring files from one machine to another. FTP uses TCP.

Gateway

Originally, researchers used the term IP gateway for dedicated computers that route packets; vendors have adopted the term IP router. Gateway now refers to an application program that interconnects two services.

ICMP (Internet Control Message Protocol)

An integral part of the Internet Protocol that handles error and control messages. Specifically, router and hosts use ICMP to send reports of problems about datagrams back to the original source that sent the datagram. ICMP also includes an echo request/reply used to test whether a destination is reachable and responding.

IPCP (IP Control Protocol)

A PPP protocol responsible for configuring the IP protocol parameters on both ends of the point-to-point link.

IPCPIN

A PPP protocol monitoring incoming requests responsible for configuring the IP protocol parameters on both ends of the point-to-point link. This was implemented to allow one instance of SOCKETS to act as both a client and server.

IP (Internet Protocol)

The TCP/IP standard protocol that defines the IP datagram as the unit of information passed across an Internet and provides the basis for connectionless, best-effort packet delivery service. IP includes the ICMP control and error message protocol as an integral part. The entire protocol suite is often referred to as TCP/IP because TCP and IP are the two fundamental protocols.

LAN (Local Area Network)

Any physical network technology designed to span short distances (up to a few thousand meters). Usually, LANs operate at tens of megabits per second through several gigabits per second.

LCP (Link Control Protocol)

A PPP protocol responsible for establishing, configuring, and testing the data link connection.

LCPIN

A PPP protocol monitoring incoming requests which is responsible for establishing, configuring, and testing the data link connection.

Memory disk

A disk that uses either ROM or RAM for the disk media. The memory disk has a FAT, directories, and file data.

MIME (Multipurpose Internet Mail Extensions)

A standard used to encode data such as images as printable ASCII text for transmission through e-mail.

Modem

A modem (modulator/demodulator) converts digital computer signals into analog signals as used in telephone equipment. The data is sent across the telephone lines and converted back to digital signals by another modem at the destination node.
Using dial-up modems, a remote client can gain access to a network through the telephone line. Remote client users can gain access to the network resources just as if they were physically connected to the LAN. SOCKETS supports the PPP, SLIP and CSLIP protocols.

MSS (maximum segment size)

The largest segment allowed for communicating across a TCP/IP connection.

MTU (maximum transmission unit)

The largest amount of data that can be transferred across a given physical network.

Multicast

A technique that allows copies of a single packet to be passes to a selected subset of all possible destinations.

Nagle Algorithm

This algorithm states that under some circumstances, there will be a waiting period of 200 ms before data is sent over a connection.. The following are the specific rules used by the Nagle Algorithm in deciding when to send data:

- If a packet is equal or larger than the segment size (or MTU), and the TCP window is not full, send an MTU size buffer immediately
- If the interface is idle, or the TCP_NODELAY flag is set, and the TCP window is not full, send the buffer immediately.
- If there is less than ½ of the TCP window in outstanding data, send the buffer immediately.
- If sending less than a segment size buffer, and if more than ½ the TCP window is outstanding, and TCP_NODELAY is not set, wait up to 200 msec for more data before sending the buffer.

For more information please see RFC-896, "Congestion Control in IP/TCP"

Nagle Heuristic
See Nagle Algorithm.

Packet
Used loosely to refer to any small block of data sent across a packet switching network.

Packet Driver
Local area network software that divides data into packets for sending on the network, and reassembles the data into its original form when it arrives at its destination.

PCMCIA

Personal Computer Memory Card Interface Association. PCMCIA is a group that defined the standard for a credit-card size card that may act as a memory RAMDISK, ROMDISK, or FLASHDISK. These cards are commonly referred to as PC cards.

POST

Power On Self Test – a test performed by the BIOS that checks the computer hardware for problems before fully initializing the computer.

PPP (Point-to-Point Protocol)
A protocol for framing IP when sending across a serial line.

RAM disk

A disk drive that uses RAM for the media in place of the usual rotating disk drive.

RFC (Request for Comment)
The name of a series of notes that contain surveys, measurements, ideas, techniques, and observations, as well as proposed and accepted TCP/IP protocol standards.

RIP (Routing Information Protocol)
A protocol used to propagate routing information inside an autonomous system.

ROM-DOS

Datalight operating system that can be placed in and execute from within a ROM.

ROM disk

A disk drive that uses ROM for the media in place of the usual rotating disk drive.

ROM scan

The scanning of the ROM area for BIOS extensions performed by the BIOS at initialization time.

ROM

Read Only Memory. This is memory that is not changeable once placed in a computer.

Router

A special purpose, dedicated computer that attaches to two or more networks and forwards packets from one to the other.

RTT (round-trip time)

A measure of delay between two hosts.

Shell

The Shell is the command interpreter, usually COMMAND.COM. The shell takes text commands and calls the DOS kernel to implement them.

SLIP (Serial Line Internet Protocol)

A framing protocol used to send IP across a serial line. SLIP is popular when sending IP over dialup phone lines.

SMTP (Simple Mail Transfer Protocol)

The TCP/IP standard protocol for transferring electronic mail messages from one machine to another.

System disk

See Bootable disk.

TCP (Transmission Control Protocol)

The TCP/IP standard transport level protocol that provides the reliable, full duplex, stream service on which many application protocols depend.

TTL (time-to-live)

A technique used in best-effort delivery systems to avoid endlessly looping packets.

UDP (User Datagram Protocol)

The TCP/IP standard protocol that allows an application program on one machine to send a datagram to an application program on another.

WWW (World Wide Web)

The large-scale information service that allows a user to browse information. WWW offers a hypermedia system that can store information as text, graphics, audio, etc.

Index