The IP Network Device Driver

The IP Network Device Driver reinstates the QL Network, allowing you to network together multiple PC's running QL emulators on a Local Area Network, or multiple QL emulators running on one PC.

The IP Network Device Driver is the IPNet, and IPLocalNet, device drivers combined into one package. When you start the driver, you choose whether you what it to be an external (IPNet) driver, where you can network multiple PC's on a Local Area Network, or an internal (IPLocalNet) driver, where you can network multiple QL emulators on one PC.

If you want to communicate between an internal and an external Network, you will require the IP Network Driver Router program.

This driver based on the TCP/IP device drivers by Jonathan Hudson & Richard Zidlicky.

The driver supports all the usual QL network commands, including the file server, FSERVE, and the NFS_USE command.

With a few differences -

There is no broadcast station (0).

Stations numbers are no longer limited to 64, you can have up to 254 stations on the network (1 to 254).

You cannot send/receive, to/from your own station number.

The driver is based on the Network Device driver from the SMSQ/E source code (version 3.16) by Tony Tebby (see licence notice at the end of this document).

The IP Network device driver was developed for use with QPC2. However it should operate on other emulators which support the TCP/IP device driver. See the compatibility section below.

Installing the Driver

Two versions of the driver are supplied. A RAM based one, and a 16K byte ROM image.

To load the RAM based version of the driver, load the driver into memory and call it.

example: i. **LRESPR flp1_NetDriver_cde**
ii. **x=RESPR(7482)** {if you don't have Toolkit 2, or equivalent}
LBYTES flp1_NetDriver_cde,x
CALL x

An installation message, and a version number will be displayed in #0

To load the ROM based version, see the user instructions of your emulator. For example in QPC2 use the command:

EPROM_LOAD flp1_NetDriver_rom

Note: You may need to disable your firewall, or give your QL emulators permissions for using your network.

Compatibility

SMSQemulator Compatible. Requires version 2.21 or above.

Qemulator Mostly compatible. There is a problem with the **FSERVE** & **NASERVE** commands that causes SuperBASIC to appear to stop working, and Qemulator appears to crash with 'Not Responding' when you try to click on Qemulator's menu bar.

Qemulator and SuperBASIC are still working, and if you try to make a remote access from another computer to the one running **FSERVE** or **NASERVE**, the remote access will work OK.

This is due to one of the IP device driver commands (IP_ACCEPT) working differently between QPC2 and Qemulator. In Qemulator it stops the emulated QL until a network connection is made.

This can also cause a problem when using the **NETI_** device. If you use **NETI_** and no other computer uses a corresponding **NETO_** You will not be able to use **BREAK** to abort the **NETI_** command, and it will also not timeout.

QPC2 Compatible. If you use the **NETI_** device, **BREAK** may not work. QPC2 does not scan the keyboard during a channel open. The keyboard scan can be reinstated in version 4 of QPC2, in the Configuration screen, under the Keyboard setting, when changed to SMSQ/E.
However the timeout will still work.

UQLX Untested, But should be compatible.

Using the Driver

Before you can use the IP Network device driver, you need to tell it the kind of network driver you want with the **NET_START** command.

The type of parameter you supply to the **NET_START** command determines how the driver will start. If you supply a number it will start as an internal Network driver. If you supply the IP address of the computer as a string, it will start as an external Network driver.

NET_START network station number , or "ipaddress"
{where ipaddress, is the IP address of the computer}

example: **NET_START "192.168.0.5"**

In this example, The driver will set itself up for networking over multiple PC's. The device driver will use "192.168.0" as the network and the QL network station number will be "5"

So if you have a LAN, with two PC's running QL emulators. One with an IP address of 192.168.0.5 and one with an IP address of 192.168.0.8

You can use commands like **SAVE NETO_8**, and **LOAD NETI_5** to copy the basic program in the computer with IP address 192.168.0.5 to the computer with IP address 192.168.0.8

example: **NET_START 3**

In this example, The driver will set itself up for networking multiple QL emulators on one PC. And this will be QL network station number 3

So if you have two QL emulators running. One started with **NET_START 3**, and one started with **NET_START 8**.

You can use commands like **SAVE NETO_8**, and **LOAD NETI_3** to copy the basic program in the emulator with network station number 5 to the emulator with network station number 8.

The following is an edited and updated extract from the Toolkit 2 manual (c) Tony Tebby

The IP Network Driver

Network Improvements

Each QL emulator connected to a network should have a unique 'station number' in the range 1 to 254. This is set using the **NET_START** command.

NET_START "IPaddress" or **NET_START** station number

Where "IPaddress" is the IP address of the network adapter to use
e.g. **NET_START "192.168.0.5"** In this case the 'station number' will be 5

The device names for the network are:

NETO_ primary station number**_**secondary station number

The primary station number, is the network station number you wish to send to.

The secondary station number is for use when accessing the IP Network Driver Router. If you are not using the IP Network Driver Router, then ignore this parameter, it will default to 0.

For example

i	**SAVE neto_6**	Save the program in memory to network station 6
ii	**OPEN#4,neto_6**	Open a channel to network station 6
iii	**COPY flp1_fred TO neto_2**	Copy a file on flp1_ to network station 2

When a network output channel is closed, then (as with the QL network driver) the network driver will keep trying to send the last buffer for approximately 20 seconds in case the receiving station is busy.

The IP Network driver will, after about 5 seconds, start checking for a **BREAK**.

NETI_primary station number**_**secondary station number**_**timeout

The primary station number, is the network station number you wish to receive from.

The secondary station number is for use when accessing the IP Network Driver Router. If you are not using the IP Network Router, then it will default to 0, unless you are going to set a station timeout. In which case it must be set to 0.

The timeout is the number of seconds (default 2 minutes) that NETI will wait for a connection request before returning with a ‘Not Complete’ error.

The maximum timeout is 32767 seconds (about 9 hours). And a timeout of 0 is treated as wait forever.

For example

i	**LOAD neti_4**	Wait for up to 2 minutes to load a program from network station 4
ii	**LOAD neti_4_0_30**	Wait for up to 30 seconds to load a program from network station 4
iii	**COPY neti_1 TO ram1_test**	Copy a file from network station 1 to ram1_

NET_ADD$ is a function that will return the IP Address entered in the **NET_START** command for an external network, or the string, '127.0.0.1' for an internal network.

PRINT NET_ADD$	display the IP Address

NET_NUM is a function that will return the network station number.

PRINT NET_NUM	display the network station number

File Servers

There are now two file servers available. The original **FSERVE** and the new **NASERVE**.

FSERVE allows free remote access to all of the servers devices and files, whereas **NASERVE** provides access controls to the sharing of it's devices and files.

Usage of the two file servers is similar. The use of **FSERVE** will be described first, then the added features of **NASERVE** will be described.

FSERVE

The file server provided in the IP Network device driver, is a program which allows I/O resources attached to one QL to be accessed from another QL. This means that, for example, disk drives attached to just one QL can be accessed from several different QL's. The file server only needs to be running on the QL with the shared I/O resource. The I/O resources may be pure serial devices (such as modems or printers) or windows on the QL display as well as file system devices (such as disk drives).

FSERVE	invokes the 'file server'

A client QL may only access up to 8 network file servers at any one time.

It is possible that files opened across the network may be left open. This can occur if a remote QL is removed from the network, is turned off or is reset. To correct this condition, wait until all other remote QLs have finished their operations on this QL, then remove the file server and restart with the commands

RJOB SERVER
FSERVE

Alternatively abandoned Server child jobs (ServCH) can be removed with **RJOB**

Accessing the File Server

The network file servers are accessed from remote QL's using a compound device name:

Nstation number_IO device	the name of a remote I/O device (e.g. **N2_FLP1_** is floppy 1 on network station 2)

For example

LOAD n2_flp1_fred	loads file 'fred' from floppy 1 on network station 2
OPEN_IN #3,n1_flp2_myfile	opens 'myfile' on floppy 2 on network station 1
OPEN #3,n1_con_120x20a0x0	opens a 20 column 2 row window on net station 2

The use of directory default names makes this rather simpler. For example

PROG_USE n1_win1_progs	by default all programs will be loaded from directory 'progs' on Winchester disk 1 on network station 1
SPL_USE n1_ser	set the default spooler destination to SER1 on network station 1

Note: There is a problem in Toolkit 2 and SMSQ/E up to version 3.32, with the **N8_** device. If the Server is Network station 8, and on a remote station you try to do a directory of a device on the server. e.g. **DIR n8_flp1_**

You will not receive a list of the files on flp1_, just the medium name and the sector counts. The problem also affects all the wildcard commands like **WCOPY** and **WSTAT**.

The problem does not effect normal loading and saving, So **LOAD n8_flp1_program** will operate correctly.

You can get around the problem with the **NFS_USE** command. for example **NFS_USE mdv,n8_flp1_**

DIR mdv1_ will now give the directory of n8_flp1_

The station numbers for remote access devices are limited to network station numbers 1 to 8. If the network station number of the file server is greater than 8. Use the **MAP_N** command to re-map a **N** station number, to the required network station number.

For example

If the computer running **FSERVE** has an IP address of 192.168.0.150 so it's network station number is 150.

A remote station cannot access a network station with n150_

Using **MAP_N 1,150** will re-map the n1_ device to now access network station 150

There is a function **MAP_LIST** that allows you to interrogate the assignments made with the **MAP_N** command. Using the above example.

PRINT MAP_LIST(1)	returns 150

It is possible to hide the network from applications by setting a special name for a network file server.

NFS_USE name, network names	sets the network file server name

The 'network names' should be complete directory names, and up to eight network names may be given in the command. Each one of these network names is associated with one of the eight possible directory devices ('name'1 to 'name'8).

For example

NFS_USE mdv,n2_flp1_,n2_flp2_	sets the network file server name so that any reference to 'mdv1' on this remote QL, will be taken to be a reference flp1 on net station 2, likewise 'mdv2' will be taken to be flp2 on net station 2
OPEN_NEW #3,mdv2_fred	now this will open file 'fred' on floppy 2 on network station 2

The network names will normally just be a network number followed by a device name as above and will end with an underscore to indicate that the name is a directory. Indeed if the network file server name is to be used with the wild card file maintenance commands, this is the only acceptable form. QUILL, however, tends to open a file with the name DEF_TMP on mdv2_. Clearly, there will be problems if more than one copy of QUILL is run across the network at any one time. This can be avoided if the network name for mdv2_ is set to be a directory:

NFS_USE mdv,n1_flp1_,n1_flp2_fred_	DEF_TMP opened on mdv2_ will now appear in directory 'fred' on flp2_ on network station 1

FLP_USE FLP is invoked after reset so if FLP is to be used as the device name in the **NFS_USE** command remember to include **FLP_USE XXX**.
This will stop the emulator from trying to access its own disk port instead of the network.

FLP_USE xyz	set device name for floppies to xyz
NFS_USE flp,n1_flp1_,n1_flp2_	any reference to 'flp1' on this QL will access flp1 on net station 1, etc.

NFS_USE$ is a function that will return the network file server name, and the network names set using the **NFS_USE** command.

PRINT NFS_USE$(0)	returns the network file server name
PRINT NFS_USE$(2)	returns the second network name

NFS_SET is a command that allows you individually set the network names for **NFS_USE** without having to reset all of them as with the **NFS_USE** command.

NFS_SET 2,n1_win1_	just sets the second network name without Affecting the others

NFS_TYPE is a function that will return the type of file server that is running. If no file server is running, it will return zero. For **FSERVE** it will return 1. And for **NASERVE** it will return minus 1

PRINT NFS_TYPE	return the file server type

Messaging

The IP Network device driver network facilities may also be used for messaging. A window may be opened, a message sent, and a reply read using a simple SuperBASIC program. If particularly pretty messages are required, then the graphics facilities of SuperBASIC may also be used. The only standard I/O facilities not available across the network are SD.EXTOP (extended operations) and SD.FOUNT (setting the founts).

For example

```
ch = FOPEN (n2_con_150x10a0x0): CLS #ch
INPUT #ch,'Do you want coffee? ';rep$
IF 'y' INSTR rep$ = 1 : PRINT 'Fred wants coffee'
CLS #ch: CLOSE #ch
```

NASERVE

While **FSERVE** allows all network stations full access to the devices and files on the file server. **NASERVE** (Network Access SERVEr) will by default, block all other network stations access to it's devices and files.

Devices and files on the Network Access Server must be specifically shared, either globally across the network, or on an individual station by station basis.

Devices and files on the Network Access Server are shared using the **NAS_SHARE** command, and unshared by the **NAS_UNSHARE** command. Additionally a list of the currently set shared devices and files can be displayed with the **NAS_SHARED** command.

The **NAS_SHARE** command can have an additional parameter added to set a directory device, a file, or a sub-directory as read only to the remote network stations.

While it is possible to flag non directory devices as read only, it does not always make sense and should be avoided.

Care should be taken when entering a list of network shares, as **NASERVE** checks network share requests in the order in which the **NAS_SHARE** commands were entered.

For example:
Suppose you want to share the file servers **win1_shared_** directory with everyone, except for network station 4, which will have read only access.
If you enter

```
NAS_SHARE 0,win1_shared_
NAS_SHARE 4,win1_shared_,1
```

This would allow network station 4 to write to **win1_shared_** Because when **NASERVE** checks the **OPEN** command from network station 4, it will first see the share **0,win1_shared_** . Network station 4 comes under everyone, so it will allow network station 4 to write to **win1_shared_**

However if you enter:

```
NAS_SHARE 4,win1_shared_,1
NAS_SHARE 0,win1_shared_
```

Then **NASERVE** will first check the share **4,win1_shared_** which will match, and it will give network station 4 the read only access required.

Note also that the **DEV** device could also cause possible problems.
If you enter

```
DEV_USE 6,ram1_
NAS_SHARE 2,ram1_,1
NAS_SHARE 0,ram1_
NAS_SHARE 0,dev6_
```

So that **ram1_** is shared with full access to all stations, except station 2, which is read only. However, station 2 will have full access to **ram1_** via **dev6_**

You would need to add an extra **NAS_SHARE** to stop station 2 writing to **dev6_**

```
DEV_USE 6,ram1_
NAS_SHARE 2,ram1_,1
NAS_SHARE 0,ram1_
NAS_SHARE 2,dev6_,1
NAS_SHARE 0,dev6_
```

When using the Network Access Sever and the IP Network Driver Router, There is a compound form of the **NAS_SHARE** command used. Where the share station number is 265 times the Router station number, plus the station number on the other network that you wish to share with.

NAS_SHARE 256*8+2,win1_

Share my **win1_** with network station number 2, on the other network, which is accessed via the Router station number 8.

The same applies to the **NAS_UNSHARE** command, and the following will remove the above sharing.

NAS_UNSHARE 256*8+2,win1_

Note: IP Network Driver Router version 1.02 or above will be required for **NAS_SHARE** to operate correctly.

When a remote network station tries to open a device or file. The order that **NASERVE** checks the requested **OPEN** command against each of the share entries, is as follows:

1. The requesters station number, unless the share entries station number is 0
2. Compares the names
3. Checks the read only status

The only **OPEN** commands that **NASERVE** will allow through for a read only access device or file, is **OPEN_IN** and **OPEN_DIR**

Operation of the **FSERVE** fileserver is unaffected, however **FSERVE** and **NASERVE** may not be used at the same time. If you attempt to start one server type, while the other type is already running, you will receive an ‘is in use’ error.

NET_START
FNET_START

The **NET_START** command is used to determine the type of network driver to use, and then starts the driver. It set the IP address to be used and/or the network station number used by this QL.

FNET_START is a function version of **NET_START** that returns zero, or an error code, without stopping the running program.

NET_START should only be used once. Just after installing the driver. If you try to use the command again, you will receive an 'already exists' error.

If an IP Address string is supplied, The driver will set itself up as an external Network to use the Local Area Network of the computer. If a network station number is supplied, The driver will set itself up as an internal Network, to network between multiple QL emulators on the same computer.

syntax: *ip_address* := *string_expression*
station_number : = *numeric_expression* {1 to 254}

NET_START *ip_address* | *station_number*
FNET_START (*ip_address* | *station_number* **)**

example: i. **NET_START "192.168.0.5"**
ii. **NET_START 4**
iii. **result=FNET_START(2)**

comment: All QL emulators on a Local Area Network must have the first three octets the same. The network station number used, will be the last octet. In the first example, the network station number of this QL will be will be 5

NET_RESTART
FNET_RESTART

The **NET_RESTART** command is used to re-assign the network station number/IP Address of the driver after the **NET_START** command has been used. You can restart the driver as either an internal, or external network. Using **NET_RESTART** will not effect any **NFS_USE** or **MAP_N** settings which have previously been set up.

FNET_RESTART is a function version of **NET_RESTART** that returns zero, or an error code, without stopping the running program.

syntax: *ip_address* := *string_expression*
station_number : = *numeric_expression* {1 to 254}

NET_RESTART (*ip_address* | *station_number* **)**
FNET_RESTART (*ip_address* | *station_number* **)**

example: i. **NET_RESTART "192.168.0.5"**
ii. **NET_RESTART 4**
iii. **result=FNET_RESTART(2)**

NET_ADD$

NET_ADD$ is a function which returns the IP Address used by the IP Network device driver as a string.

If the driver is operating in internal mode it will return the string '**127.0.0.1'**. If it is operating in external mode, it will return the IP Address string entered in the **NET_START** command.

syntax: **NET_ADD$**

example: **PRINT NET_ADD$**

NET_NUM

NET_NUM is a function which returns the network station number as an integer.

If the driver is operating in internal mode it will return the station number entered in the **NET_START** command. If it is operating in external mode, it will return the last octet of the IP Address string entered in the **NET_START** command.

syntax: **NET_NUM**

example: **myStationNumber = NET_NUM**

FSERVE

The **FSERVE** command is used to start the file server.

syntax: **FSERVE**

example: **FSERVE**

NASERVE

The **NASERVE** command is used to start the share file server.

If you attempt to start **NASERVE** while **FSERVE** is already running, you will receive an 'is in use' error.

If **FSERVE** is running it may be stopped with a **RJOB SERVER** command

syntax: **NASERVE**

example: **NASERVE**

NAS_SHARE
FNAS_SHARE

The **NAS_SHARE** command is used to add a device or file to the list of shared resources.

A network station number defines the station that can use this device or file, Specifying a station number of 0, shares the resource for all users.

An optional access type defines whether the device or file has read/write or read only access. 0 (the default) defines that the device or file has read/write access, and a positive number defines the device or file has read only access.

Note that a read only protected device or file can only be accessed by an **OPEN_IN** or an **OPEN_DIR**

If you just want to change a shared device or file access type, it is not necessary to unshare, and then re-share the device or file again. Just share it again with the same network station number and device/file name.

When using the Network Access Sever and the IP Network Driver Router, There is a compound form of the **NAS_SHARE** command used. Where the share station number is 265 times the Router station number, plus the station number on the other network that you wish to share with.

FNAS_SHARE is a function version of **NAS_SHARE** that returns zero, or an error code, without stopping the running program.

syntax: *station_number* := *numeric_expression* {0 to 254}
resource := *device/file name*
access_type := *numeric_expression*

NAS_SHARE *station_number*,*resource*,[*access_type*]
FNAS_SHARE (*station_number*,*resource*,[*access_type*] **)**

example:
i. **NAS_SHARE 0,ram1_** {share ram1_ with everybody}
ii. **NAS_SHARE 1,win1_shared_** {share win1_shared_ with net 1}
iii. **NAS_SHARE 1,win1_shared_,1** {make win1_shared_ read only to net 1}
iv. **NAS_SHARE 3,con_** {allow net 3 to open consoles}
v. **NAS_SHARE 2,ser1** {allow net 2 use my ser1 port}
vi. **result=FNAS_SHARE (2,flp1_)** {allow net 2 to use my flp1_}
vii. **NAS_SHARE 256*8+2,win1_** {share win1_ with net 2 via the Router 8}

Note: When **NAS_SHARE** is used. Entries are added in the order that they are supplied. Unless you are changing the access type of an existing entry.

NAS_UNSHARE
FNAS_UNSHARE

The **NAS_UNSHARE** command will remove a shared device or file from the share list.

NAS_UNSHARE without parameters will remove all the shared permissions.

FNAS_UNSHARE is a function version of **NAS_UNSHARE** that returns zero, or an error code, without stopping the running program.

syntax: *station_number* := *numeric_expression* {0 to 254}
resource := *device/file name*

NAS_UNSHARE *station_number*,*resource*
FNAS_UNSHARE (*station_number*,*resource* **)**

example:
i. **NAS_UNSHARE 1,win1_shared_** {stop sharing win1_shared_ with net 1}
ii. **NAS_UNSHARE 0,ram1_** {stop sharing ram1_ with everybody}
iii. **NAS_UNSHARE** {remove all shares}
iv. **result=FNAS_UNSHARE(2,flp1_)** {stop sharing flp1_ with net 2}
v. **NAS_UNSHARE 256*8+2,win1_** {stop sharing win1_ with net 2 via the Router 8}

Comment: Un-sharing a device or file which already has an open channel, will have no effect on that connected channel as sharing rights are only tested during **OPEN**s.

NAS_SHARED

The **NAS_SHARED** command is used to display a list of the currently shared resources.

It will display a list in the format of Router station number, Network station number, Resource name, Access type. Where the access type is 0 for read/write access, and 1 for read only access.

An optional Basic channel number may be specified. With channel #1 being the default.

syntax: *channel_number* := *numeric_expression*

NAS_SHARED [#*channel_number*]

example:
i. **NAS_SHARED** {send shared list to Basic channel 1}
ii. **NAS_SHARED#2** {send shared list to Basic channel 2}

MAP_N

The **MAP_N** command is used to re-assign the network station number used by one of the 8 **N** devices.

Using **MAP_N** without parameters will reset the assignments back to the default settings of the **N** devices n1_ to use network station 1, n2_to use network station 2 etc.

syntax: *Ndevice_number* := *numeric_expression* {1 to 8}
station_number := *numeric_expression* {1 to 254}

MAP_N [*Ndevice_number*, *station_number*]

example:
i. **MAP_N 1,150** {n1_ will access network station 150}
ii. **MAP_N 3,50**
DIR n3_flp1_ {do a directory of flp1_ on network station 50}
iii. **MAP_N** {mappings reset to defaults}

MAP_LIST

MAP_LIST is a function that returns the station number associated to the re-assignments made with the **MAP_N** command.

syntax: *Ndevice_number* := *numeric_expression* {1 to 8}

MAP_LIST (*Ndevice_number* **)**

example: i. **PRINT MAP_LIST(1)** {get station number associated with n1_ }
ii. **MAP_N 3,50**
PRINT MAP_LIST(3) {prints 50}

NFS_USE

The **NFS_USE** command sets the network file servers name, so that any reference to the supplied 3 letter device name on this remote QL, will be taken to be a reference to one of the supplied server devices.

The network names should be complete directory names, and up to eight network names may be given in the command. Each one of these network names is associated with one of the eight possible directory devices (name1_ to name8_).

syntax: *name* := *3_letter_device _name*
network_name := *server_device*

NFS_USE *name*, *network_names* {up to 8 network names}

example: i. **NFS_USE mdv,n2_flp1_,n2_flp2_** {mdv1_ will reference n2_flp1_ and mdv2_ will reference n2_flp2_}
ii. **NFS_USE mdv,n1_win1_,n2_flp1_,n1_ram1_**
COPY mdv2_fred TO mdv3_fred {copy the file fred from flp1_ of server 2, to ram1_ of server 1}

comment: The network names will normally just be a network number followed by a device name, and will end with an underscore to indicate that the name is a directory. Indeed if the network file server name is to be used with the wild card file maintenance commands, this is the only acceptable form. QUILL, however, tends to open a file with the name DEF_TMP on mdv2_. Clearly, there will be problems if more than one copy of QUILL is run across the network at any one time. This can be avoided if the network name for mdv2_ is set to be a directory:

NFS_USE mdv,n1_flp1_,n1_flp2_fred_

DEF_TMP opened on mdv2_ will now appear in directory 'fred' on flp2_ on network station 1

NFS_USE$

NFS_USE$ is a function which returns the name, or the network names entered in a **NFS_USE** command.

The entry number parameter determines which network name is returned, with entry number 0 being the **NFS_USE** name.

If the name, or the required network name is missing. **NFS_USE$** will return an empty string.

syntax: *entry_no* := *numeric_expression*

NFS_USE$(*entry_no***)** {0 - 8}

example: i. **PRINT NFS_USE$(0)** {the **NFS_USE** name will be displayed}

ii. **a$=NFS_USE(2)** {a$ is set to the second network name}

comment: If the command **NFS_USE mdv,n1_win1_,n2_flp1_,n1_ram1_** has been used, then example (i) above will return '**mdv**'. And example (ii) will return '**N2_flp1_**'.

NFS_SET

The **NFS_SET** command allows any of the eight network names set by the **NFS_USE** command to be individually changed without having to reset all of the network names at the same time with as with **NFS_USE** command.

syntax: *entry_no* := *numeric_expression* {1 – 8}
network_name := *server_device*

NFS_SET *entry_no*, *network_name*

example: **NFS_SET 2,n2_flp2_** {change the second entry that was set with **NFS_USE**}

comment: If you wish to change the device name used by **NFS_USE**, just use the **NFS_USE** command with one parameter e.g. **NFS_USE nfa**.

NFS_TYPE

NFS_TYPE is a function that will return the type of file server that is running.

If no file server is running, it will return zero. For **FSERVE** it will return 1. And for **NASERVE** it will return minus 1

syntax: **NFS_TYPE**

example: **PRINT NFS_TYPE**

IP Network Protocols

The device driver uses the TCP protocol

The external device driver use two port numbers 7001 & 7002. While the internal driver uses ports 53248 + station number & 53504 + station number.

Port 7001 & 53248 (plus station number) is used by the NETI_ / NETO_ device.
Port 7002 & 53504 (plus station number) is used by the remote file server, the Nx device.

Each data packet sent across the network is comprised of a header and a data block section.

The header is ten bytes long in the following format:

00	Packet length (2 bytes)
02	Destination station number
03	Sending station number
04	Block number (high byte)
05	Block number (low byte)
06	Block type (0 normal, 1=last block of file) NETI/O (D0 for a Trap call, 255=block request, 254=supplied block) Nx
07	Number of bytes in block (0 to 255) NETI/O ((0 to 255)*4) Nx
08	Final destination station number (NETO device Router) Originator station number (Nx device with Router)
09	Originator station number (NETI device with Router)

If the number of bytes in a block is 0, 256 data bytes are actually sent.

IP Network Server Protocol

The IP Network server protocol is physically the same as the Standard QL protocol, but the header has been slightly changed allow blocks of up to 1000 bytes to be sent. A server packet can not be confused with a standard packet, as a different port number is used.

Acknowledgements

I would like to thank Urs Koenig, Wolfgang Lenerz and Giorgio Garabello for their suggestions, advice, and help in the development and testing of IPNet, IPLocalNet and IPNet Router.

Copyright and Disclaimer

This driver should not cause any problems, damage, or loss of data. However by using this device driver, you do so at your own risk, and I do not accept responsibility for any damage, or loss of data.