XR32

XROUTER MULTI-PROTOCOL ROUTER

SYSOP'S MANUAL

by Paula Dowie G8PZT Revision 2.2 - 18th December 2002

XR32 Documentation Upgrade by Rod McCosker VK2DOT Revision 3.1 – 02 Dec 2012

Not Finished Yet!!!

PREFACE

This manual is far from complete, and the information is probably not in the most logical order. However, I thought you'd rather have some documentation than nothing at all! Getting the document perfect would have needlessly delayed the release of the software (and has in fact already done so), so I have chosen to refine the documentation after release. You may have to search for what you want, and in some cases you may not find it.

Writing documentation is very time consuming, and prevents me from programming, so I am trying to balance both activities.

If you don't find what you want, or you think something could be better explained or positioned, PLEASE let me know, don't leave it to others - they are leaving it to others too!

As a programmer, I am not the best person to write the documentation - I have tried to include what I think you need to know, but with intimate knowledge of the code I tend to make assumptions without realising it.

INDEX:

SECTION 1 - INTRODUCTION

XR32 is an upgrade of Xrouter 16bit [DOS Version], ie It is a 32 bit version. XR32 is a AX25 and IP packet router for the amateur packet radio network, using a standard PC running the Windows operating system. The original XRouter was 16 bit and nor intended for Windows or Linux machines. The original XRouter is referred from here as XR16.

The new XRouter 32 bit version was developed from XR16 to run on Windows machines on November 2011. Now referred as XR32.

XRouter is called a "router" rather than a "node" because its primary purpose is to route data packets. Node is a general purpose term meaning "junction", and can be an end user. This software simply routes packets. Whether those packets contain ax25, Netrom, IP, or any other protocol is irrelevant. It is not the job of a router to provide FTP, SMTP, NNTP, or BBS facilities, so they aren't provided.

Although some of the commands and configuration options may look familiar to NOS users, this is largely coincidental, and you should not expect NOS commands and options to work unreservedly. Some NOS commands were derived from UNIX, as were some of mine. Sometimes there is only one logical way to do things, and both NOS and Xrouter chose the same way. However, even if the commands look familiar, the syntax may not necessarily be identical, reflecting the fact that this system has different design criteria.

This program is not a variant of NOS and is not derived in any way from NOS. The latter has a reputation for being flaky, cumbersome and un-intuitive. This program is aimed at busy sysops who want to get the job done, not play around with software. It was designed from scratch (the console display was the first component), not "cut down" from someone else's software.

Whereas NOS is a complete TCP/IP networking system with rudimentary Netrom capability, Xrouter specialises in multi-protocol routing and user access. Although IP is one of the protocols, this is NOT a "TCP/IP program". It does not have, and never will have "hub" facilities - if you want a hub, use NOS.

FEATURES AND LIMITATIONS:

The following is a list of some of the features and limitations of Xrouter. It is by no means an exhaustive list, nor is it in any logical order. (* = advantages relative to BPQ)

XR32 Hardware / Software Compatibility

- Can run under Windows 2000, XP & 7, or Linux (using wine).
- Compatible with Netrom NODES recovery file for easy transition.
- Can use KISS and BPQKISS TNC's.
- KISS modes: normal, polled, checksum, and bpq slave.
* ASYNC interface supports any speed up to 115,200 bauds.
* ASYNC Protocols: KISS, "Netrom backend", SLIP, ASCII (TTY).
* Ethernet driver allows connection with Windows, Linux, NOS and BPQ
* Year 2000 compatible.

General Features

* Configurable software watchdog
* Hardware watchdog driver.
* External hardware control & monitoring.
* Budlist and Validcalls lists for each port.
- Digipeating on/off/UI-only from any port to any port.
* Digicasting
* Frame piping from port to port.
* Proxy connections.
* Cross-protocol bridging.
* Very comprehensive stats to aid network maintenance.
* MHeard list sizes can be customised for each channel.
* MHeard includes date, time, callsign, frame count and frame type.
* Integral multi-channel chat server available to all users.
* Configurable mtu.
- Can TX on a different port to RX.
* Callsign validation in nodes broadcasts.
- Ports may be "interlocked" so they don't transmit at the same time.
- Can support 32 bit Windows applications such as UI-View, Winpack...
* ID beacons may be configured independently for each port.
* Netrom echo and route record facilities.
* Modulo-128 capability, with frame resenqencing and selective reject.
* Message-Of-The-Day facility.
* Link parameters are self-adjusting.
* Integral Personal Message System (PMS).
* Can emulate a multi-port TNC2 for RS232 peripherals to use.

APRS Features

* MHeard can display APRS position, distance and heading.
* APRS generic digipeating (RELAY, WIDE, TRACE, TRACEn-N and WIDEn-N)
* Maintains APRS "Best DX" list.
* Decodes MIC-E packets.
* Integral APRS Packet <> Internet gateway.
* Integral APRS messaging shell.
* Integral APRS server.
* APRS "digipeating" via Netrom.
* Responds to APRS / UI-View queries.
* ID may be beaconned via digipeaters.
* Ports may be configured for "APRS-only" use.

TCP/IP

* Built in IP router, with full ARP & ICMP implementations.
* IP commands fully accessible on normal console, and to all users.
* Built in TCP, giving TELNET, ECHO, DISCARD, FINGER, CHAT, RLOGIN, FTP
* Domain name to ip address resolution.
* Inbuilt DNS server, with proxy capability.
- IP datagram, virtual circuit, netrom and encap modes.
* AXIP, AXUDP, IPIP and IPUDP protocols, allowing links via Internet.
* Additional IP addresses for each port.
* DHCP client
* Network and Port Address Translation
* Dial Up Networking
* RIP98 Routing Information Protocol
* Telnet -> Telnet / AX25 / NetRom proxy
* AX25 / NetRom -> TCP/IP / AX25 proxies.

Command Interface

- User commands: bye, connect, info, help, links, mheard; nodes, ports, routes, stats, users.
* Up to 128 command aliases can be defined
* Additional user commands: PING, ARP, TELNET, CHAT, J, ECHO, DX etc.
* Built in syntax help for most commands.
* Aliased commands can be "hidden".
* Graduated help system gives help in a progressive way.
* Extendible Info system for sysop to set up as s/he wishes.
* Command scheduling by time / day / date.

Console Interface

* Screen saver - automatic and manual.
* Editable command line.
* Fully configurable display colours.
* Up to 5 independent "Virtual" consoles, fully multitasking.
* Each virtual console has a configurable scrollback buffer
* Each virtual console can display ANSI colour.
* Each virtual console can run a separate session.
* Each console may have its own colour scheme and callsign.
* Configurable connect / disconnect and paging bells
* Per-console command history.

Local And Remote Sysop

- Most parameters configurable while on line.
* Tx for any port may be disabled/enabled by sysop command.
* Line editor allows any text file to be edited.
* IP routing adjustable without reboot.
* Transaction logging.
* Online sysop's manual.
* Each protocol layer independently traceable.
- Each port independently traceable.
* Trace o/p can be captured to file.

Additional Features For Remote System Maintenance

- Secure password system for remote sysops.
* Allows separate passwords for each sysop.
* Remote operation via RS232 wired links. (dialup coming soon).
* Secure FTP server, allowing remote upgrading
* Trace display can be viewed by remote sysop.

Limitations

- A Linux version is intended, but not yet implemented.
- No SMTP / POP3 / BBS - it's a router/Network Access Point not a server!

COPYING, USE AND DISTRIBUTION

You must read the following text carefully. It is not designed to take away your rights, but rather to protect mine. The software took a long time to write, at considerable cost to my health, and I don't see why others should make a profit from my hard work.

The terms "the code" and "the software" refer to Xrouter and all components thereof, and all documentation relating to it.

This software is "conscienceware". You may freely use and share it for any non-commercial purpose, and you are not compelled to pay a registration fee. You may however ease your conscience and aid the continuing development and support of the software by sending a small donation to the author if you wish (Cheques should be made payable to Ms P J Dowie at the address below).

You are not permitted to make any charge for, or gain any pecuniary advantage directly or indirectly from, the copying, distribution or use of this software, or any component thereof, or any driver or utility written for use with the software, unless by prior written agreement with the author. For example, you may not place the software on the Internet with the intention of attracting more visitors to a site.

You must not patch, decompile, disassemble, reverse engineer or in any way modify the software, or cause a modified version or part thereof, or any portion of the source code to be distributed.

Legal action will be taken against anyone who breaks the terms outlined above.

SOURCE CODE

After due consideration of the pros and cons of "open source" projects, I have decided that the source code will not be made available, and this is not negotiable. All protocols and API's developed by myself will however be published.

FEEDBACK

Because the terms and conditions outlined above prohibit patching, and the source is not available, the only way the bugs will be fixed and the software improved is for me to do it. I am therefore relying on you to report any bugs promptly, and to make known your ideas for future improvements. You are my beta testers. I promise to correct all reported bugs promptly where possible, and to carefully consider all suggestions.

Please also report any errors or omissions in the documentation. If you don't like the layout or feel something isn't properly explained, tell me. You can make a difference.

If you want to send Packet bulletins regarding this software I suggest the use of "XROUT" as the To field. I strongly suggest you join the Xrouter mailing list at YahooGroups.com - email xrouter@yahoogroups.com.

You can contact VK2DOT for document upgrade by the following means:

Packet: VK2DOT@VK2DOT.CC.NSW.AUS.OC

Email: vk2dot@tpg.com.au

CONVENTIONS USED IN THIS MANUAL

Certain conventions are used in this document to avoid unnecessary repetition. They are as follows:

|	**("or") Separates alternative options or arguments.**
<>	**Encloses a mandatory argument**
[]	**Encloses one or more optional arguments.**
callsign	**A valid amateur radio callsign, e.g. G8PZT**
dir	**Directory**
dirname	**Directory name**
file	
filename	**A unique file name without directory**
filespec	**Specifies one or more files, with or without directory**
he	**Represents both "he" and "she"**
path	**A fully qualified directory path**
pathname	**A full directory plus filename.**
string	**A contiguous sequence of non-whitespace characters.**
text	**One or more strings separated by whitespace.**
whitespace	**Spaces, tabs or newlines**

Examples: DIR [filespec] - DIR command has one optional argument, which may be a file or file group description, with or without directory.

In cases where a pathname is required, if only a filename is given, the program's working directory is assumed.

SECTION 2 - INSTALLATION & CONFIGURATION:

I won't bother providing detailed instructions, since every installation will be different and you won't read this anyway, unless you're feeling really bored ;-)

XR32 MACHINE REQUIREMENTS

There are no hard rules about this; it depends how many ports you want, how heavily the system will be loaded, what facilities you require etc. I have tried to make it as flexible as possible.

XR32 *will* run on memory stick.. A hard drive isn't essential

XR32 OPERATING SYSTEM

XR32 requires either Windows 2000, XP or 7 operating system. It will also run on Linux (using WINE).

DIRECTORY STRUCTURE

All files and directories required by the system are located within a single directory which can be located anywhere and have any name. For the purposes of this document I will assume it is named XROUTER.

The directory should contain at least XR32.EXE and XROUTER.CFG. All other files are optional and the system will run without them.

If you want to grant access to remote sysops you will need to add PASSWORD.SYS. If you want to store IP routing info, you will need IPROUTE.SYS. If you want domain lookup, add DOMAIN.SYS. If you want to control TCP/IP access, add ACCESS.SYS and USERPASS.SYS.

If you want to import existing nodes and routes information from a BPQ system, place a copy of the BPQNODES file (produced by BPQ's SAVENODES command) in the directory and rename it PZTNODES.

If you want the full help system, create a sub-directory called HELP and place all the .HLP files into it. Note that the help for the CHAT server goes in sub-directory HELP/CHAT, FTP server help goes in HELP/FTP and the APRS messaging help goes in HELP/AMSG.

If you want an INFO system, create the INFO directory. I have included some sample .INF files to give you some ideas.

If you want the online manual, create the MAN directory and put the *.MAN files into it.

SETTING UP XR32

The software will be supplied in ZIP format. Within the zip file, the system, help and manual files will be in separate zip files to make it easier for you to select the components you require. You may create the directory structure manually and unzip the components into it, or you may use PKUNZIP -d to create the sub-directories.

Once the required files are installed in the correct places, your next job is to edit the configuration file XROUTER.CFG, to suit your requirements.

If you have installed PASSWORD.SYS, IPROUTE.SYS or DOMAIN.SYS they will also need editing to suit your requirements. Again, they are all self-documenting, but are also detailed elsewhere in this document.

RUNNING THE XR32 PROGRAM

Execute the file XR32.EXE program. NOTE: the console size of the software is controlled by the line – ROWS=99; where the size can be from 25 to 200 lines. The larger the size, the easier to read BBS's etc.

STOPPING THE XR32 PROGRAM

Alt-X stops the program, saving the nodes and routes tables to file PZTNODES just before it exits back to DOS. It is protected by an irritating "are you sure" challenge.

XR32 Notes:

Original Revision 1.0, 9th August 2011 by G8PZT Paula
Modified and Formatted by Rod VK2DOT 13-Jun-2012.

STATUS:

This XR32 Notes: section of this document was initially released as a “not for publication”. It originally contained provisional information for use by XR32 alpha testers only. But now has been inserted into the XR16 / XR32 Sysop’s manual – Mainly for use by the XRouter 32 bit testers..

Paula G8PZT reserves the right to change the XR32 program and any specifications, instructions and supporting documentation without notice. For reasons of quality control and consistency, this document may change from time to time in regard to its contents.

This is not (yet) a sysop manual. It is a random collection of notes which may help you understand the XR32 software.

Introduction:

XR32 is a Windows 32 bit version of the DOS-based XRouter Amateur Radio packet networking software.

It is a "Console Application", intended as an interim version between DOS XRouter and the full GUI version (to be released later), and therefore to have the same look and feel as the DOS version. This should allow DOS XRouter users to upgrade to more modern operating systems and hardware with minimal effort.

Bearing the above in mind, XR32 is not GUI-aware, and probably never will be. It is the old familiar XRouter, just sitting on a new platform. This will be the optimum product for unattended systems, where a GUI is pointless.

For the purposes of this interim documentation, it is assumed that you are familiar with DOS XRouter. If you are new to XRouter, you may need to refer to other pages on XR16, in this XRouter sysop manual. An online (albeit rather out of date) version can be found at:

http://zl2tze.ath.cx/hamradio/offlinecopy/zl2arn.ath.cx/xrouter/docs1.htm

You are also recommended to read Rod's installation instructions at -

http://vk2dot.dyndns.org/XR32/XR32.htm

Supported Operating Systems:

XR32 has been tested on Windows 2000, XP and 7. It is not known at this stage whether it will work fully on earlier or later operating systems. It appears to load but hang on Windows 98. This will be investigated if there is any demand for Win98 suport.

The **NdisXpkt** driver, which allows XR32 to share the Ethernet NIC and have its own IP address, is only available for Windows 2000 and XP. However XR32 version 200 *should* work without it. The only difference being that for LAN operations, XR32 will share the Windows IP address.

Installation Folder

You can install and run XR32 anywhere you like, even on a USB memory stick.

For a number of reasons however, it is strongly recommended that you don't install XR32 under "Windows", "Documents and settings" or "Program Files". Firstly, if your motherboard dies, and/or you attempt to transfer the HD on another machine, the NTFS security will prevent you from accessing anything within those folders. Secondly, it is believed that on Vista/Windows7, programs are not allowed to write data within the "program files" tree. It is recommended to use **C:\XR32** directory.

To install XR32, create a folder in your chosen location and copy XR32.EXE and XROUTER.CFG into it. Edit XROUTER.CFG as required. You may need to install other files, e.g. IPROUTE.SYS, depending on your configuration. Please refer to the this XRouter sysop manual for more information.

XR32 will create any required subfolders when it runs. It does not write anything outside its own folder tree, and does not use the Windows registry. Therefore you may move the working folder around without problem. To remove XR32 from your system, simply delete the folder containing it.

Windows Console Settings:

At this stage, XR32 uses a Windows console 80 characters wide and variable height. [use command **ROWS=##**] in XROUTER.CFG File - to determine height of XR32 console. These dimensions are not yet dynamically adjustable, but this will be addressed in the near future.

The choice of font is left to the sysop, but the recommended font is 8x12 "raster fonts". This provides the best clarity and the most familiar aspect ratio, and allows XR32 to be run in "full screen" mode by hitting Alt-Return.

To set the font, right click on the console's title bar. Select "properties", then under "Fonts", select "size: 8x12" and "font: raster fonts". It should then say "selected font: Terminal". Click Apply.

Operation with any font other than the recommended one is not guaranteed.

Serial RS232 Ports:

XR32 can be used with any serial port which is recognised by Windows. This includes "real" ports, USB-to-serial adaptors, and "**virtual com ports**" provided by external programs.

If you are upgrading from DOS XRouter, your XROUTER.CFG file might need slight modification...

Serial ports only need "COM=1" etc. they don't use IOADDR and INTNUM now. For example:

```
; USB to Serial adaptor
INTERFACE=5
   ID=COM8 serial
   TYPE=ASYNC
   COM=8
   PROTOCOL=KISS
   KISSOPTIONS=POLLED
   SPEED=1200
   MTU=256
ENDINTERFACE
```

SCC Cards:

Multi-port TNC cards (DRSI, BAYCOM, RLC100, PC100, PA0HZP, PC120, ITACARD and ARKTILL) based on 8530 SCC chips are not yet supported in XR32. They will be supported in due course.

YAM Modem Support:

This *should* have survived the move from DOS to Windows, but is still untested at present. Reports would be appreciated.

Sound Cards:

These are not directly supported, but XR32 can use them via AGW Packet Engine. See "AGWPE Interface" below.

NDISXPKT Interface:

NdisXpkt is an optional kernal-mode driver which allows XR32 to share an Ethernet NIC with Windows, and to have its own completely independent IP address. Ie If your WinXP machine has an IP address of 192.168.1.3; Your XR32 may have an IP address of 192.168.1.6 – when using NdisXpkt.

Note: This driver can only be used with Windows 2000 or XP.

The NdisXpkt driver is only required if you wish to share an Ethernet NIC with Windows. XR32 will work without it, but you will not be able to trace any IP activity.

If you don't need any Ethernet / Internet networking (e.g. if you're only doing AX25/Netrom/IP over radio, or using KISS/SLIP links), you won't need NdisXpkt.

Before you can use NdisXpkt, it must be installed and started. Please follow the installation instructions in the install.htm document of the NdisXpkt package.

To start the driver, open a command window and type:

net start NdisXpkt

Note the driver name is case sensitive and must be typed EXACTLY as above. If you type it any other way, the driver will start but will not be available to XR32.

NdisXpkt only needs to be started once per session. You only need to start it again if you restart the operating system. It is hoped to automate the startup in future versions of XR32, but in the meantime it is recommends the use of "**startdelay**" to automate the startup of NdisXpkt and XR32. Startdelay is free software on the internet.

XR32 interfaces to NdisXpkt driver via an EXTERNAL interface, which is declared in XROUTER.CFG as follows:

```
INTERFACE=1
 TYPE=EXTERNAL    ; External device driver
 ID=Ethernet LAN
 PROTOCOL=ETHER
 MTU=1064
 ETHADDR=00:07:95:fa:d9:3b       ‘ acquired by using “ROUTE PRINT”
ENDINTERFACE
```

Note that the ETHADDR is mandatory and must match the MAC address used by the chosen NIC.

To find your NIC's MAC address, open a command window and type

IPCONFIG /ALL

then look for "Physical address", e.g.

```
Ethernet adapter Local Area Connection:
Physical Address. . . : 00-07-95-FA-D9-3B
```

Alternatively, type "**ROUTE PRINT**" and look for the name of the NIC in the list, e.g.

```
Interface List
0x1 ..... MS TCP Loopback interface
0x2 ...00 07 95 fa d9 3b .. SiS 900 PCI Fast Ethernet
```

As you can see, there are several different ways of depicting a MAC / physical / Ethernet address. XR32 requires the use of colons ':' between the 6 pairs of characters.

The Ethernet port is declared like this:

```
ID=Ethernet
INTERFACENUM=1
IPADDRESS=192.168.0.2
NETMASK=255.255.255.0
etc...
```

Make sure you choose a different IP address to the one Windows is using!

Note: The use of interface type EXTERNAL for NdisXpkt is a temporary measure, liable to change in fuure.

AGWPE Interface:

You can use AGW Packet Engine "underneath" XR32 to manage the hardware, and to provide access to hardware not directly supported by itself (e.g. soundcards). (Don't confuse this with XR32's AGW emulation feature, which is there to support apps which expect to see AGW)

Here's how to specify an AGWPE interface:

```
INTERFACE=1
   TYPE=AGW
   ; IP addr of AGWPE. Default is localhost
   IOADDR=192.168.0.76
   ; AGWPE port. Default is 8000
   INTNUM=8001
   MTU=256
   CONFIG=MyAgwPassword    ; Password for AGWPE
ENDINTERFACE
```

Note that IOADDR, INTNUM and CONFIG are all optional, and are only needed if you want to change the defaults.

If you don't specify IOADDR, it defaults to 127.0.0.1, which is the same computer as XR32 is on. If AGWPE is on a different computer to XR32, you need to enter its IP address here.

If you don't specify INTNUM, it defaults to 8000, which is the normal AGWPE port.

If you don't specify CONFIG, XR32 won't "authorise" with AGW. Authorisation is not usually needed if you're using XR32 and AGWPE on the same computer. You can adjust the requirement for authorisation in AGWPE. When CONFIG is used, the "username" sent to AGWPE is the NODECALL and the "password" is the string specified by CONFIG.

The AGWPE interface can support 16 PORTs, which are declared in the usual way, each PORT using a different CHANNEL on the INTERFACE - for example:

```
PORT=1
   ID=AGW Port 1
   INTERFACENUM=1
   CHANNEL=A
ENDPORT

PORT=2
   ID=Agw Port 2
   INTERFACENUM=1
   CHANNEL=B
ENDPORT
```

Starting XR32:

Firstly, if you are using the NdisXpkt interface, start NdisXpkt if not started already. Then double click the XR32 icon. The program should start.

If the window appears briefly, then disappears again, there is a configuration error. There may be some clues in LOG/BOOTLOG.TXT, but error logging there is still quite rudimentary. Your best option is to open a command window, navigate to the XR32 folder and type "xr32" to start the program. When it bombs out, it should print an error message to the window.

If you start more than one copy of XR32, make sure there is no competition for resources. e.g. if one copy uses COM1, no other copy may use COM1. Similarly, NdisXpkt cannot be shared (yet).

If you attempt to run 2 copies without one of them using NdisXpkt, the second copy will bomb out because it can't open the same TCP/IP server ports on Windows. The solution is simple: use the TCP service port override keywords (described below) to reassign or disable the ports.

Use Alt-X (preferred) or Ctrl-C (last resort) to exit the program.

AGW Application Support:

XR32 provides an AGW TCP host mode interface on TCP port 8000, allowing apps which would normally use AGWPE to use XR32 instead.

The AGW emulation port can be changed using the AGWPORT keyword in the GLOBAL section of XROUTER.CFG. You might need to do this if you are running AGW on the same machine. e.g.

AGWPORT=8001

Whether or not XR32 requires the applications to send a password is controlled by entries in ACCESS.SYS.

The following entry allows computers on the LAN to access XR32 without a password:

192.168.0.0/24 1

For further information on ACCESS.SYS see:

http://zl2tze.ath.cx/hamradio/offlinecopy/zl2arn.ath.cx/xrouter/doc95.htm

AGW host support has so far been tested with:

- UIView
- AGWTerminal

TCP/IP Stacks:

XR32 has its own TCP/IP "stack" that is completely independent of Windows. This stack is "multihomed", i.e. it can exist on several networks at the same time, with different IP addresses on each network.

If you use the NdisXpkt driver, XR32's TCP/IP stack can share the network card (NIC) used by Windows, whilst using its own IP address. Alternatively it could use any other NIC installed in the PC.

If you don't use the NdisXpkt driver (e.g. because you are using 64-bit Windows), XR32's TCP/IP stack is available only via the serial ports, using SLIP, PPP, dialup or KISS TNC's (IP over radio).

However, in addition to its own TCP/IP stack, XR32 can also use the one built into Windows. So even without NdisXpkt you can still use almost the full range of TCP/IP services, via Windows. The only difference is that traffic via XR32's stack is fully traceable, whereas traffic via Windows is not. And XR32 shares the Windows IP address for such operation.

The choice of how your TCP/IP packets are routed and which stack they use is entirely under your control (it could have been automated, but that would have deprived you of choice).

By default, all TCP/IP traffic is routed via XR32's own stack. This would be inappropriate if for example the only interface with the outside world was via TNC's and you wanted to route traffic other than amprnet (44.x.x.x). But perfectly valid if you were using NdisXpkt to access the Internet via Ethernet etc.

The way to force traffic via the Windows stack is by means of IP ROUTE entries with mode "w" instead of "d". Anything not caught by a "w" entry will be routed via XR32.

TCP Service Ports

TCP "service ports" (not to be confused with radio ports), are standard or "well known" numbers between 0 and 65535 which are used in combination with an IP address to access a particular process (usually a server) on a system.

Default service port numbers and configuration keywords used in XR32 are as follows:

ECHOPORT	**7**	**Echo server**
DISCARDPORT	**9**	**Discard server**
FTPPORT	**21**	**FTP server**
TELNETPORT	**23**	**Telnet server**
FINGERPORT	**79**	**Finger server**
HTTPPORT	**80**	**HTTP server**
TTYLINKPORT	**87**	**Raw TTY link**
RLOGINPORT	**513**	**Remote login**
SOCKSPORT	**1080**	**Socks proxy server**
APRSPORT	**1448**	**APRS server**
TELPROXYPORT	**2323**	**Telnet proxy server**
CHATPORT	**3600**	**Chat server**
AGWPORT	**8000**	**AGW TCP host API**
RHPPORT	**9000**	**XRouter host API**

Overriding Default TCP Service Ports:

By default, all services are enabled, using the above ports.

If you are using the NdisXpkt driver to share the Ethernet card, these services will use XR32's IP addresses, as specified in XROUTER.CFG.

If you are not using NdisXpkt, these services will use Windows' IP address. But they will also be available on XR32's IP address (if one is specified), allowing them to be used by TCP/IP over radio, TCP/IP over SLIP/PPP etc..

Use one or more of the above configuration keywords in the GLOBAL section of XROUTER.CFG to override the default configuration.

If you use the keyword without an argument, or with an argument of zero, that service is disabled.

If you use the keyword with a single argument, that value is used for both IP stacks.

If you supply TWO arguments, the first applies to XR32's stack and the second to the Windows stack. You may supply different numbers for each stack, or disable one and not the other. The numbers must be seperated by whitespace NOT commas.

Examples:

The following formats disable the Telnet server on both XR32 and Windows TCP/IP addresses:

TELNETPORT=
TELNETPORT=0

This enables the Telnet server, using port 88 on both TCP/IP stacks:

TELNETPORT=88

This specifies port 88 on XR32's TCP/IP stack, and port 89 on Windows:

TELNETPORT=88 89

This one disables the Telnet server on XR32's TCP/IP stack, while enabling it for port 88 on Windows:

TELNETPORT=0 88

Finally, this example disables the Telnet server on XR32's TCP/IP stack, whilst enabling it on port 88 of Windows:

TELNETPORT=88 0

MPORT Command:

There have been some changes since v187. Two new subcommands are ALL and NONE, which should be self explanatory. For example, "MPORT ALL" monitors all the ports at once, and "MPORT NONE" disables all tracing, a function which used to be accomplished using "MPORT 0".

Port zero is now a "pseudo-port" used to trace traffic coming and going via Windows' TCP/IP stack. So "**MPORT 0**" now traces port 0, instead of disabling tracing.

The binary MPORT flags have been moved up to accomodate port 0, so bit 0 now represents port 0 instead of port 1, and so on. So "MPORT #06" (binary 110) will trace ports 1 and 2, but not port 0.

APRS Generic Digipeating:

XR32 has inbuilt support for APRS (Automatic Packet Reporting System) generic digipeating for RELAY, WIDE, TRACE, TRACEn-N and WIDEn-N aliases.

Generic digipeating is configured on a port by port basis, using thw following flags in "DIGIFLAG":

4	Enable RELAY generic digipeating.
8	Enable TRACE generic digipeating.
16	Enable WIDE (Well sited stations only!)
256	Enable UITRACE digipeating
512	Enable UIFLOOD digipeating

Add the appropriate numbers together to enable your desired combination of services.

UITRACE and UIFLOOD are two special addresses that are suffixed with pseudo-SSID's, e.g. "TRACE4-4" and "WIDE2-2". These addresses can digipeat several times. The first digit specifies the maximum number of hops, and the second is the hop counter, which is decremented each time the frame is digipeated.

These two addresses behave slightly differently however. When a frame is digipeated on the address specified by UITRACE, each digipeater inserts its own callsign in the digipeater list and decrements the "SSID". Frames digipeated on the UIFLOOD address have their SSIDs decremented, but the digi doesn't insert its own callsign.

For the sake of consistency with UI-View, UITRACE defaults to "TRACE", giving TRACEn-n digipeating, and UIFLOOD defaults to WIDE, giving WIDEn-n digipeating.

However, according to the APRS "New Paradigm", RELAY, TRACE and WIDE are deprecated, UITRACE should be set to "WIDE", and UIFLOOD should be set to a "state" code (e.g. "GBR" for the UK).

One of the main reasons for the new paradigm was the fact that some of the older digipeaters would repeat the same packets over and over. This does not happen with XRouter however.

Not everyone agrees with the "New Paradigm, so the choice of which features to enable is left you you.

In quiet areas, you may wish to mix APRS and normal connected-mode operations on the same port, and that is the default if you enable any of the above flags.

AXIP and AXUDP:

By default, AXIP and AXUDP use XR32's TCP/IP stack, so your Internet router should be set up to "port forward" incoming UDP and/or IP protocol 93 to XR32's LAN IP address.

However, if your XR32 configuration doesn't include an Ethernet port, you must port forward to the Windows' IP address instead.

Even if you are using NdisXpkt, you can override XR32's IP stack and "force" XR32 to use Windows' IP address on a port by port basis, by setting a PORT's IPADDRESS to 127.0.0.1

DNS:

XR32 may still use an external domain server, just like its DOS predecessor. But you may prefer to use the Windows DNS instead. To do this, just remove (or comment out) any "DNS=x.x.x.x" statements in XROUTER.CFG.

Nodes Command:

The new "N C" command lists the nodes sorted in callsign alphabetical order, instead of sorted by alias. Don't forget that you can still have the nodes table sorted by callsign by default, by putting "**SORTBYCALL=1**" into XROUTER.CFG.

CONSOLE DISPLAY

The screen will display a single "virtual console". Each virtual console has the following format:

There is a single-line "status bar" at the top of the screen and a single-line "menu bar" at the bottom. Immediately above the menu bar is a single "command entry" line. The remaining lines central section is where monitored and received information is displayed, and this section is ANSI compatible.

At this stage, XR32 uses a Windows console 80 characters wide and variable height. [use command **ROWS=##**] in XROUTER.CFG File - to determine height of XR32 console. These dimensions are not yet dynamically adjustable, but this will be addressed in the near future.

The status bar shows the date and time, the virtual console number, the console callsign, the version number, the router's callsign and alias, and the amount of free memory. The menu bar shows the most important keys.

The text and background colours of the various sections are fully configurable. Default settings are built into the program, and can be overridden by entries in the global section of XROUTER.CFG. The defaults can be further modified for individual consoles by adding console definition blocks.

You may specify a different callsign for each console (used when making outgoing connections), or use the same callsign for each one.

CONSOLE KEYS

Here's a summary of the keys - they are fully explained later.

<F1>	**Display console help window.**
<F2>	**Toggle monitoring on/off.**
<F3>	**Select port(s) to monitor. (MPORT)**
<F4>	**Select monitored protocols. (MMASK)**
<F5>	**Toggle disk capture on/off.**
<ESC>	**Cancel operation / Disconnect.**
L/R arrow	**Select previous / next console.**
U/D arrow	**Scroll back/forward by one line.**
<PgUp>/<PgDn>	**Scroll back/forward by one page.**
<End>	**End scrollback.**
<Ctrl-LeftArrow>	**Previous command.**
<Ctrl-rightArrow>	**Next command.**
<Alt-C>	**Clear screen.**
<Alt-X>	**Exit program.**

<F1> - Help

Pressing <F1> pops up a window with the above information on it. The <ESC> key closes that window when you've finished with it. The contents of the window are context-sensitive, i.e. they will vary according to what you were doing when you press the <F1> key.

<F2> - Monitoring

The <F2> key provides a quick way to turn monitoring on and off, i.e. it has the same function as BPQ's <F1> key. Note that I didn't follow BPQ here because *most* software uses <F1> for help.

<F3> - Monitor port

The F3 key selects the ports to be monitored. At the moment only ports numbered 1 to 32 can be monitored. Upon pressing this key you will be invited to enter a number or string of numbers representing the combination of ports to monitor. If you press <F1> at this point a pop-up window will show you the following information:

To select port(s) to monitor, add together the following hex values:

Port	HEX	Port	HEX	Port	HEX	Port	HEX
1	1	9	100	17	10000	25	1000000
2	2	10	200	18	20000	26	2000000
3	4	11	400	19	40000	26	4000000
4	8	12	800	20	80000	28	8000000
5	10	13	1000	21	100000	29	10000000
6	20	14	2000	22	200000	30	20000000
7	40	15	4000	23	400000	31	40000000
8	80	16	8000	24	800000	32	80000000

e.g. #12 will monitor ports 5 and 2.

Alternatively you may enter a single decimal port number, or a combination of ports separated by "+" e.g. "1+5+23".

You may cancel the operation by pressing the <ESC> KEY.

<F4> - Monitor Mask

The <F4> key invites you to enter the monitor mask, i.e. the protocols you wish to monitor. Pressing <F1> at this point brings up the following information:

Add together the following HEX values:

0001 - Incoming frames
0002 - Outgoing frames
0004 - AX25 layer 2
0008 - AX25 info frames
0010 - AX25 layer 3
0020 - AX25 layer 4
0040 - IP frames
0080 - ARP frames
0100 - ICMP
0200 - TCP
0400 - KISS
0800 - SLIP
1000 - PASSALL
2000 - Hex Dump

E.g. 0201 will show incoming TCP frames only, while 0241 will show the underlying IP frames as well. The default setting is 03FF, which shows all incoming and outgoing traffic from AX25 layer 2 upwards.

<F5> - Disk capture

The <F5> key toggles disk capture on and off. Captured text goes to file CAPTURE.TXT. When capture is on, everything which appears on the central 22 line section of the console is sent to file. You can make it more specific using MMASK and MPORT. (Note this is a completely separate operation to the transaction logging enabled by setting LOG=1 in the configuration file.) Make sure your disk is fast enough and has enough space, otherwise Xrouter's performance will be impaired.

<ESC> - Escape

The <ESC> key "backtracks" if you have pressed <F1>, <F3> or <F4>, or disconnects a console session.

Switching consoles

The left and right "arrow" keys will switch between the virtual consoles if you have enabled more than one. The console number is shown on the top status bar. Note that when you reach the last console, you wrap back to the first and vice versa.

Review mode

The up and down "arrow" keys control scrollback, along with the <PgUp>, <PgDn> and <End> keys. You must enable the scrollback buffer for this to have any effect. Scrollback or "review" mode enables you to

look at something which went off the top of the screen. The console won't receive anything else until you exit review mode.

Command History

Each console remembers the last 5 commands used on that console. These can be selected using the <Ctrl-LeftArrow> and <Ctrl-RightArrow> keys, and actioned by hitting <RETURN>

SECTION 3 - MISCELLANEOUS TOPICS

These are just random notes which I haven't yet had time to organise properly.... One day I'll write a proper manual!!

Nodes/routes table saving

The Nodes & routes tables are saved to file PZTNODES 1 minute after starting the program, and at global NODESINTERVAL (usually 1 hour) intervals thereafter. The SAVENODES command dumps the node table at any time, and the tables are automatically saved when you exit the program using <ALT-X>. Thus if you exit and restart you will not lose the tables, only the active links.

You can specify an alternative filename when using the SAVENODES command. For example, you might want to make a periodic backup of the nodes table, from which things can be restored if the table gets empty. (a radio failure will cause nodes to expire from the table, and you may wish to get it restarted quickly after fixing the radio)

The LOADNODES command enables the table to be loaded from any file, at any time.

Remote Tracing

Remote sysops may invoke tracing using the MONITOR command, but this feature should be used with caution. Tracing generates a lot of data, and the link should be capable of handling it. It is mainly intended for use on TTY (wire) links, but can be used over any connection.

A remote sysop is prevented from tracing any activity on the port upon which he is uplinked.

Only one remote sysop may receive tracing at any one time.

ANSI

The console will respond to ANSI, so you can have full colour terminal sessions with ANSI-capable systems such as PZT BBS. ANSI sequences are not (yet) generated from the keyboard.

HELP SYSTEM

Basic syntax help for most commands is built into Xrouter, and is available using the query (?) command, e.g. "? N" will display the syntax of the NODES command.

More detailed help is implemented as separate files in the HELP directory, allowing you to customise it and add extra help topics as desired. Each topic occupies a separate file, with the filename the same as the topic name. The "H *" command displays a sorted directory of all the files ending in .HLP.

Within the .hlp files, lines beginning with a semicolon ';' are not displayed to users, so you can use them for comments, such as file modification details.

You may choose to omit some or all of the help files if you are running a floppy-only system.

In addition to the HELP system, sysops will find more information in the online sysop manual using the MAN command.

INFO SYSTEM

When the user issues the "I" command without any arguments, the text defined by INFOMSG is sent to him.

However, this command may also form the entry point for a much larger and more comprehensive information system which the sysop may implement how he wishes. For example you may wish to include details of the local packet network and clubs, and maybe a set of tutorials on general packet topics...

In order to do this, the INFO directory must exist and must contain the desired info in the form of text files with the extension .INF. The user will then, after using "I" by itself, be invited to enter "I *" to list the available info topics. If the user then enters that command, the filenames (minus the .INF) will be displayed in alphabetical order, and the user will be invited to read the files using the "I <topic>" form of the command.

The filename should be relevant to the contents, within the constraints of 8 DOS characters. You should try to keep each file concise in order to save air time, preferably breaking complex subjects into a series of small files with "see also" references to link them. It is very frustrating for users to begin reading a file only to discover it's not of interest, and then having to wait a long time for it to finish! You may use ANSI or HTML in the files, but not pure binary.

Within the .INF files, lines beginning with a semicolon ';' are not displayed to users, so you can use them for comments, such as file modification details.

IP ROUTING

IP routing is an integral part of Xrouter, and is fairly simple to set up. You may not see the point, but others will thank you for doing so!

In order to enable this function you must have an IP address. If you don't already have one, contact your local IP co-ordinator who should be only to pleased to assign you one. You should be able to find your local co-ordinator via the UKIP web site or a packet message.

Put your IP address in the IPADDRESS= line in XROUTER.CFG.

Xrouter normally uses a single IP address for all ports, but if you need different addresses on different ports you may specify an additional address for any port, by including an IPADDRESS= line in the port definition.

Example: IPADDRESS=44.131.91.245

The other thing you need to do is define the routing in IPROUTE.SYS - see the explanation in the section which relates to that file. If you don't know what routes to enter, your local co-ordinator should be able to help.

Having to enter a full IP address when using the PING and TELNET commands is tedious, so you should put some entries in DOMAIN.SYS to perform the translations between host names and IP addresses. For example, I have CNAME entries which translate the local node aliases into IP addresses. (see later section on DOMAIN.SYS)

If you have reasonably fast access (under 10 sec) to an external DNS (Domain Name Server), you may enable its use by specifying its IP address using the "DNS=" keyword in XROUTER.CFG.

Example: DNS=162.31.224.1

Finally, you should set Xrouter's host name for TCP operations, using the "HOSTNAME=" keyword in XROUTER.CFG. This is optional, and if you omit it, it will default to "NODEALIAS:NODECALL".

Example: HOSTNAME=g8pzt.ampr.org

The "Domain Suffix" is currently fixed at ".ampr.org". Commands such as "telnet g8pzt" will be automatically extended to "telnet gb7pzt.ampr.org", but "telnet fbb.org" will be left alone because the address is already complete.

AXIP TUNNELLING:

AXIP is AX25 "encapsulated" within (i.e. carried in the payload section of) IP datagrams. It **enables** AX25 systems to communicate with each other via any TCP/IP network such as the Internet. The AX25 links thus created can in turn support Netrom and amateur TCP/IP.

Assuming you have a prospective AXIP partner, you would set up an AXIP wormhole as follows:

1) Configure and test IP routing between you and your partner. If you don't have reliable IP routing there's no point in proceeding!

   If you are linking via the Internet, it makes sense to use the Internet IP addresses for this purpose, rather than the amateur ones, because the routing is more reliable and the throughput faster.

2) If you wish to use your partner's hostname (e.g. g8pzt.ath.cx) instead of the IP address (e.g. if the partner has a dynamic IP address), your system needs access to a Domain Name Server (DNS).

   If Xrouter is connected directly to the internet via a relatively dumb device such as a telephone or cable modem, it can obtain the address of a suitable DNS via PPP or DHCP negotiations.

   If it is connected indirectly, via a more sophisticated device such as a firewall, an Internet Connection Sharing system (Windows98 ICS) or a good quality xDSL modem, it is likely that such a device will have an inbuilt "DNS proxy server", which will answer DNS requests on behalf of the LAN. If the device is DHCP-capable, and you have configured Xrouter to use DHCP, Xrouter will automatically know the address of the DNS proxy.

   If you are not using DHCP or PPP you must include a valid "DNS=..." statement in XROUTER.CFG, e.g. "DNS=88.23.207.1". Note that you must supply the IP address of the DNS, not its hostname.

   At VK2DOT, we use DNS=192.168.1.1

   Verify that "PING {hostname}" resolves the address properly.

3) Add an AXIP interface to XROUTER.CFG as follows:

```
INTERFACE=13
    TYPE=AXIP
    MTU=256
ENDINTERFACE
```

(Choose the interface number to suit yourself).

This interface can support an unlimited number of AXIP ports. You may define more than one AXIP interface if your ports need differing MTU's.

Only TYPE=AXIP and MTU= are required, all other parameters will be ignored (at present).

4) For each AXIP partner, add an AXIP port similar to this:

```
PORT=95
   ID=AXIP link with VE3UIL
   INTERFACENUM=13
   IPADDRESS=44.136.16.6
   IPLINK=ve3uil.dyndns.org
ENDPORT
```

You must specify at least ID, INTERFACENUM, and IPLINK.

IPLINK is the remote host's IP address or hostname. It is more efficient to use IP addresses, if you are able to do so, because it removes the need to resolve the hostnames. The "protocol number" for AXIP is fixed at 93 (decimal).

IPADDRESS is only required if you wish to encapsulate IP traffic within the AXIP frames, and then only if it differs from Xrouter's main IP address.

MAC parameters such as TXDELAY, TXTAIL, SLOTTIME, PERSIST, FULLDUP, SOFTDCD etc. are meaningless for AXIP, but FRACK, RESPTIME, PACLEN, MAXFRAME, QUALITY etc. operate as normal.

On fast internet links you may wish to use a much lower FRACK, say 2000ms, than on radio. I would not recommend reducing it below 1000ms, as it needs to be *at least* twice the worst round-trip time plus the other end's RESPTIME.

RESPTIME is probably the one which will have most effect on the responsiveness of the system, because it controls the time delay between receiving a packet and sending an ACK. It should be just a little more than the time it takes to receive a maximum length packet. For example, at a data rate of 56Kbits/sec, a 256 byte packet lasts less than 50msec, so RESPTIME=50 would be adequate.

5) If Xrouter is indirectly connected to the internet via an intermediate router, that router will probably be using some form of NAT (Network Address Translation) to share one "public" IP address between several systems on your LAN. The "front end" router will probably route outgoing AXIP without problem, but it will not know to which machine to send incoming AXIP unless explicitly configured. Configuring such a router for AXIP usually involves specifying a protocol number (93), and the LAN IP address of a machine to which it should be routed, i.e. Xrouter's LAN IP address.

   You are advised that not all front end routers can be configured to route incoming AXIP as it is not a commercially recognised protocol. Some routers will only allow TCP and UDP port redirection, with no provision for any other protocol. Windows 98 ICS will not route incoming AXIP, but there are (at a price) non- Microsoft programs which will do so.

   If you are lumbered with such a router, you may need to consider AXUDP instead - see later.

AXUDP TUNNELLING

AXUDP is AX25 "encapsulated" within UDP datagrams. This enables AX25 systems to communicate with each other via TCP/IP networks such as the Internet. It is slightly less efficient than AXIP, because there is an extra 8 byte UDP header between the IP and AX25 headers in the frame, but the difference is not significant. The major advantage of AXUDP over AXIP is that it can usually be handled by front end routers such as Windows 98 ICS without much problem.

The procedure for setting up an AXUDP wormhole is very similar to that for setting up AXIP, and you should first read the AXIP section elsewhere in this manual.

The differences are as follows:

1: Instead of (or in addition to) the AXIP interface, add an AXUDP interface to XROUTER.CFG as follows:

INTERFACE=14
TYPE=AXUDP
MTU=256
ENDINTERFACE

As with AXIP, this interface can support an unlimited number of AXUDP ports. Multiple AXUDP interfaces are allowed.

2: For each AXUDP partner, add an AXUDP port similar to this:

PORT=39
ID=AXUDP link with N9PMO
INTERFACENUM=14
IPADDRESS=44.136.16.6
UDPLOCAL=10096
IPLINK=n9pmo.dyndns.org
UDPREMOTE=10096
ENDPORT

See the AXIP section for a discussion of the required parameters.

UDPLOCAL and UDPREMOTE are the UDP "service port" numbers for each end of the link, and if omitted they default to 93. Don't confuse these with *protocol number* 93, which is AXIP

3: If you are using a front end router, xDSL modem etc. you will need to "open" UDP port 93 (or your UDPLOCAL number if it differs from the default), to allow incoming UDP frames to be directed to Xrouter's IP address. For Windows 98, a very useful program called ICSCFG can be obtained from the internet for this purpose.

Some routers don't have the facility to open specific UDP ports, but at the very least should allow you to direct all UDP traffic to a specified IP address.

IPUDP TUNNELLING:

IPUDP is IP encapsulated in UDP/IP. It allows amateur IP to be transported across other TCP/IP networks such as the Internet, to form "Virtual Private Networks" (VPN's).

Whereas IPIP (commonly called Encap) transports the private (e.g. amateur) IP directly inside the payload portion of a public (e.g. Internet) IP datagram, IPUDP transports the private datagram in the payload portion of a UDP frame, which is itself transported as payload in a public IP datagram. This requires 8 bytes more overhead than IPIP, but it is far more flexible.

IPIP is a "portless" protocol, and it is therefore difficult (in come cases impossible) to get it to pass through some types of NAT / PAT router which rely on translating TCP and UDP service port numbers in order to share a public IP address among several LAN hosts.

IPUDP overcomes this limitation because it transports the data using a well known protocol (UDP) which NAT / PAT routers can understand, thus it can get through where IPIP cannot. For example, it is easy to configure Windows 98 Internet Connection Sharing (ICS) to route incoming IPUDP to a specific machine on the LAN, but it is not possible to do this with IPIP.

IPUDP currently uses UDP service port 94, simply because I found it easy to remember, and there seemed to be no other significant protocols using this number. As originator of this protocol, if difficulties are encountered with port 94, please tell me (G8PZT).

In order to establish an IPUDP link, you and your link partner must first set up and test IP routing between your public IP addresses. If you are using a "front end" router between Xrouter and the Internet, you must "open" UDP port 94 to direct incoming traffic to Xrouter. Finally you must add an extra IP route entry as follows:

IP ROUTE ADD 44.131.91.0/24 62.31.206.176 5 u

(Omit the "IP" if the entry is used in IPROUTE.SYS)

The first IP address is the amateur IP address, or range thereof, to be routed via this IPUDP link. If you don't fully understand this format, see the manual sections detailing IPROUTE.SYS and the IP ROUTE ADD command.

The second address is the IP address or hostname of the link partner to whom the first address(es) will be routed. It is more efficient to use an IP address if possible. Your system MUST be told how to reach that partner, so you will need an existing routing entry which applies to it.

The "5" is the number of the port on which the datagrams will be transmitted. This port obviously needs a public IP address, or one which will be translated to a public address by a front end router.

Mode "u" signifies IPUDP encapsulation.

TCP/IP SERVICES

ECHO Server

If a user TELNETs to TCP port 7, anything he sends will be echoed back to him. This is a useful tool for testing TCP/IP systems. It requires no setting up, other than the IP routing.

DISCARD Server

This uses TCP port 9, and merely dumps anything sent to it. It is another useful tool when developing and testing TCP/IP systems. It requires no setting up other than the IP routing.

FINGER Server

This allows users to find information about other users. It uses TCP port 79 and reads text files for local users from the FINGER sub-directory. It may also be accessed using the FINGER command at the main prompt. If the user isn't local, it attempts to establish communication with the Finger server on the specified host.

This server is only partly developed at present, and future versions will give more information. For the moment, if you wish to activate this feature, create a FINGER sub-directory within the XROUTER directory, then simply create a text file for each user, using the user's callsign or any other preferred "hostname" as the filename. The files can contain anything you like, typically the user's name, location, station details, QSL information etc.

For example the file finger/g8pzt might contain the text:

Name: Paula
Qth: Kidderminster, Worc's IO82VJ
Other: Sysop G8PZT:KIDDER router, Sysop GB7PZT BBS, Fourpak Secretary, Unemployed software author with special interest in comms software. Author of: G8PZT AX25/IP BBS, Xrouter, Uk White pages system, PEARL Off-line reader for Packet Radio...

RLOGIN

The secure password challenge/response mechanism, using the "@" command, is inconvenient for frequent use, and unnecessary in cases where the remote sysop can access the router via a "secure" link such as a wire link between two systems. Therefore a plain text login system is provided for these cases. This uses a TELNET connection to TCP port 513. The user is prompted to enter his callsign, and is then asked for his password. If the two match with an entry in PASSWORD.SYS, the user is granted access with sysop status. Needless to say, this facility should *NEVER* be used over a radio link, otherwise someone will see the password.

FTP server

An FTP server is included within the program, allowing remote sysops to upload, download, list, rename and delete files, create and remove directories etc., which is useful when the router is located somewhere inaccessible, such as a garage or remote hilltop. For example, new configuration and program files may be uploaded, and the system can then be restarted to perform a remote upgrade.

The FTP server is not available to non-sysops. It is protected by a password grid, and is only accessible to those who have a password registered in PASSWORD.SYS.

Access to all drives and directories is unrestricted, because the FTP server is intended only for system maintenance, not as a public file repository.

The server uses standard FTP commands, with the exception of the USER and PASS login sequence, which are necessarily tailored for use on a publicly visible channel. The password prompt presents the user with a matrix of 5 lines of 5 numbers, and the response may be either a string of characters, as with secure sysop login, or the full password in plain text. Needless to say you would only use the plain text password on a secure channel.

Although this is DOS program, the server format is "Windows_NT" because that is the format which gives the best results with the widest variety of FTP clients.

In order for the FTP server's HELP command to work, you must place the ftp help files in the HELP/FTP directory.

The FTP server commands are described in detail in the sysop command reference section.

Domain Name Server

A rudimentary DNS (Domain Name System) server is built into Xrouter, which will answer DNS queries from other systems via the standard UDP port 53.

The server only responds to requests directed at the *main* IP address, and will only answer one query per message. Only standard queries for type A are currently answered. Recursion is supported. Other query types will be supported in a later version.

The server will act as a DNS proxy, allowing Xrouter to function as an Internet Connection Sharing router for a network of machines. The machines on the local intranet send their DNS queries to Xrouter, which either resolves them using its own DNS lookup, querying an external DNS if necessary.

MODULO-128

Modulo-128 is an extension to AX25, giving sequence numbers from 0 to 127 instead of the usual 0 to 7. This allows a much bigger MAXFRAME (up to 63) to be used, and is primarily of use on good links.

On conventional (Modulo-8) ax25 links, much of the inter-node traffic consists of short frames containing level 4 control information. These frames are interspersed with data-bearing frames, which themselves may be only partly full. Thus a router may transmit 7 frames in one go, but the transmission may only be a few hundred bytes in total. This is inefficient, due to the delays associated with channel access, txdelay, txtail, resptime etc. On fast, error-free links the data arrives in short bursts, separated by long gaps of inactivity.

With Modulo-128, many more frames can be packed into a single transmission, so the channel overheads become less significant.

Because the largest allowed Maxframe value for Modulo-128 is 63, there can never be any sequence number ambiguity, and this allows frame "re-sequencing". With normal AX25, if the first frame of a multi-frame set is lost, the whole set of frames has to be re-transmitted. If Modulo-128 is used however, the "good" frames are stored by the recipient, and only the lost frame is re-sent. Together with the stored frames, this completes the set, and the whole set can be acked. This is a much more efficient strategy.

Xrouter is capable of Modulo-128, and frame resequencing. Modulo-128 frames are displayed on the trace screen as "EAX25" (Extended AX25) instead of "AX25", and are initiated by a new frame type "SABME" (Set Asynchronous Balanced Mode Extended). You will notice the sequence numbers being displayed as "<I R25 S32>"

The method of activation isn't fully decided as yet. If a caller requests Modulo-128 (by sending an SABME frame), Xrouter will respond with <UA> and go into Modulo-128 mode. The strategy for outgoing links is more complex. If the port maxframe is greater than 7, *ALL* level 2 downlinks on that port will be tried using Modulo-128 (i.e. Xrouter will send <SABME> instead of <SABM>. This is not recommended on user ports, since hardly any users are EAX25 compatible.

If the called system is Modulo-128 capable, the correct response to <SABME> is <UA>, otherwise the correct response according to the AX25 protocol is either <DM> or <FRMR>, which will cause Xrouter to fall back to normal Modulo-8 AX25. Most systems do answer correctly, but there may however be some systems which do not properly implement AX25, for example by silently discarding <SABME> frames, and it will not be possible to link to these systems if Maxframe is greater than 7. I hope to address this in a future version.

It is more likely that Modulo-128 would be used on inter-node links with other Xrouters, Linux and PE1CHL boxes, so to activate it, simply define a maxframe > 7 for those routes which require it. If the port is dedicated to one link, you can set the port maxframe > 7 instead.

MHEARD

The MHEARD facility lists recently heard stations, and may be enabled or disabled for any port. It records callsigns in order of reception, with the most recent at the top of the list, along with other useful information, such as the date/time and the number of frames heard.

This is useful for users to discover who else the router can hear, to aid the search for suitable digipeaters, and to diagnose problems.

Even on linking-only ports, where there is only usually one partner, it provides a useful indication when the frequency is being encroached, either by deliberate squatting, unauthorised attempts to link, or lift conditions. I recommend that you enable MHEARD on all ports.

If you have included an APRS-style position report in your ID beacon, Xrouter will know its own position and will display position, distance and bearing for any stations which broadcast APRS positions. This is a great aid to network mapping, and it would be nice if everyone were to make use of APRS.

If the station was heard via a digipeater, the digipeater call is also shown, and the letter "D" is shown in the "type" field. If the heard station is a node, the letter "N" is displayed, if it is sending IP traffic, "I" is shown, and if it is sending ARP traffic, "A" is shown.

Within XROUTER.CFG, the MHEARD= keyword is used in each port definition block to enable or disable the MHEARD facility and to set the size of the list. For example "MHEARD=50", enables it and sets the table size to record a maximum of 50 callsigns. If the keyword is omitted, the default is 15 callsigns.

You can control what sort of callsigns are recorded in the MH list using the MHFLAGS= keyword. The default is 255 (show everything), and the number is formed by adding the following values:

1 Show directly heard stations
2 Show directly heard digipeaters
4 Show digipeated stations

For example "MHFLAGS=3" would show directly heard stations and digipeaters, but not the stations heard via digipeaters.

The MHEARD list is available to sysops and users alike, using the MH command (see command reference section).

A port's mheard list may be cleared by using the MHCLEAR <port> command. This would typically be used if the port was changed to another frequency. You may clear all MH lists by specifying port 0.

WATCHDOGS

Xrouter has an inbuilt software watchdog.

A lock-up may be caused for example by faulty external driver or third-party server software, by certain hardware errors, or by "soft" errors due to RF break-through, supply spikes or ionising radiation.

Additionally, a DOS "shell" session of excessive duration is treated as a lock-up because the operator may have abandoned the console without terminating the shell, and the shell may be running a process which cannot be safely terminated. Therefore a complete reboot is the safest option.

The software watchdog is enabled by entering a non-zero setting in the "watchdog interval" parameter in INIT.SYS (see elsewhere). You should not use too low a setting, otherwise the system may reboot if an external process takes too long to run, or if you shell out to DOS using the F9 key. 120 sec (2 minutes) is a suggested figure.

The software watchdog cannot work miracles, because sometimes the crash is too severe, and sometimes the computer will not respond to a "warm" reboot. The use of an additional "hardware" watchdog is recommended.

Therefore, in addition to the software watchdog, the system will supply pulses via a spare parallel port, to drive an external hardware watchdog. This would for example consist of an opto-isolator driving a monostable circuit such that if the pulses ceased for more than 120 secs (must be longer than the boot-up time!) the mains supply would be interrupted for long enough to cause a "cold" reboot (say 5 secs).

The hardware watchdog pulses are enabled by specifying a suitable parallel port address for the WATCHADDR parameter. The D0 pin will be toggled at 18 Hz, D1 at 9Hz, D2 at 4.5 Hz etc. You can drive an optoisolator from one of these pins.

A hardware watchdog should detect the presence or absence of an AC signal from the parallel port. A sustained high or low value indicates a fatal lock-up. Your circuit should include a delay long enough to allow the system to reboot, including all programs (e.g. chkdsk) which are run at boot up. This is to prevent it from triggering again before the pulses are established, causing an endless reboot cycle!

Warning: If you run Xrouter in a Windows95 environment, the frequency of the hardware watchdog pulses will be greatly reduced (by a factor of 9) if the window is backgrounded. The software watchdog interval is similarly inflated, so a 2 minute timeout will become 18 minutes. I have yet to find a solution to this problem, which is due to Windows not properly emulating certain BIOS functions. However, I feel it is unlikely that anyone would want to run a remote system (and thus require watchdogs) on a Windows system, due to stability considerations!

The following section attempts to explain some of the figures produced by the "stats" command.

Entering S alone will produce a single page thus:

G8PZT:KIDDER} Stats:

```
Time active (mins):              4880 (3 days 9 hours 20 min 10 sec)
Warm restarts:                      0
Memory: Max/Cur/Min/Out        309832    278056    273720         0
Memory blocks: Min/Cur/Max        860       993      1099
Known nodes: Cur/Max              235       250
Total bytes sent/rcvd:       52840616  55556865
Ccts/MaxCcts L2/L3/L4/TCP:      16/21     19/34      3/14      6/13
IP Heard/Reasm/Rcvd/Routed:   3004712         0   2982347     21636
IP Bad Hdr/Chksum/Version:          0         0         0
IP Sent/Frag/Unsent/Total:    2998038       118       693   3019792

L4 Connects Tried/Made/Rcvd:     1967      1938       445
L4 total frames Sent/Rcvd:      50831     48688
L4 Sent/Rcvd/Resent/Reseq:      27514     26906       411         9
L4 Chokes Sent/Rcvd:                7        72
L4 Timeouts/Failures:             371        39
```

```
Use STATS * to display full stats
```

"Time active" is the elapsed time since Xrouter was last restarted.

"Warm restarts" shows how many times Xrouter was restarted using the RESTART command, or by automatic restart.

"Memory" displays Xrouter's RAM usage. "Max" is the maximum available free memory after the program is loaded, "Cur" is the current free memory, "Min" is the minimum that has been reached during operation, and "Out" is the number of times a memory request has failed.

"Memory blocks" shows the minimum, maximum and current number of allocated memory blocks.

"Known nodes" is the number of nodes heard about whose quality is greater than the minqual setting.

"Total bytes sent/rcvd" are the total bytes sent and received by all the ports. They include all ax25 overhead.

"Ccts/MaxCcts" shows the current and highest number of simultaneous circuits that have been active at any time. Separate figures are shown for Ax25 levels 2,3 and 4, and TCP/IP.

"IP Heard/Reasm/Rcvd/Routed" shows the total number of IP frames heard (i.e. addressed to us and to others), reassembled from fragments, received (i.e. addressed to us), and routed to others.

"IP Bad Hdr/Chksum/Version" shows the number of IP frames ignored due to corrupt IP headers (e.g. too short to be a legal IP frame), checksum mismatch, and incompatible IP version.

"IP Sent" is the number of IP datagrams originated by Xrouter, i.e. not routed from somewhere else. "Frag" is the number of datagrams which had to be fragmented to fit a link. "Unsent" is (if I remember correctly) the number of datagrams which couldn't be sent due to low memory or no route, and "Total" is the total number of datagrams or fragments thereof which were transmitted.

"L4 Connects Tried/Made/Rcvd" shows the total number of outgoing and incoming AX25 level 4 connections. "Tried" is the number of requests initiated, "Made" is the number which were successful, and "Rcvd" is the number of incoming connects.

"L4 total frames Sent/Rcvd" shows the total number of AX25 level 4 frames of all types sent and received by the router.

"L4 Sent/Rcvd/Resent/Reseq" shows the totals for AX25 level 4 information-bearing frames. "Sent" is the number of frames originated by us. "Rcvd" is the number addressed to us. "Resent" shows how many were re-transmitted because no ack was received. "Reseq" is the number of frames that re-sequenced, i.e. were received out of sequence and subsequently used when the missing frames arrived.

"L4 Chokes Sent/Rcvd" counts the number of ax25 level 4 choke frames sent and received by the router. A sent choke indicates that we are receiving L4 data faster than we can route it, and instructs the other end to back off for a while. A received choke indicates that we are sending data too fast for the other end to handle. Note that these figures do not necessarily indicate that there is something wrong with the router's links, as they apply to the "virtual circuit" from one application to another, which may span many intervening nodes.

"L4 Timeouts" shows the number of times the ax25 level 4 T1 timer timed out while waiting for an ack, causing re-transmission of a frame. "Failures" shows the number of level 4 circuits which were abandoned due to excessive retries.

Entering s * will produce the above followed by:

```
           Interface: 1      2      3
RX Overruns:          0      0      0
TX Underruns:         0      0      0
CRC/Framing Errors:   0      0      0
Break/Abort Errors:   0      0      0
Rx Overflow err:      0      0      0
Tx Overflow err:      0      0      0
Misc. errors:         0      0      0

L3 Frames Heard       0      0      0      0  12681      0  31364   6736
L3 Frames Rcvd        0      0      0      0   3904      0  23721   1698
L3 Frames Sent        0      0      0      0  13978      0  25438   7418
L3 Frames Relayed     0      0      0      0   8036      0   7643   5038
L2 Frames heard    4896      0     21   2171  25594    440  42977  28402
L2 Frames rcvd     2288      0      0    582  24816    378  42902  27976
L2 Resequenced        0      0      0      0      0      0      0      0
L2 Reassembled        0      0      0      0      0      0      0      0
L2 Info Received    593      0      0    104      0     63      0      0
L2 T1 Timeouts      134      0      0     18   1086     40   4453   3407
L2 Digipeated         0      0      0      0      0      0      0      0
L2 Info Sent       2301      0      0    755  15580    381  25436  12170
L2 Info re-sent     362      0      0     20   1507     38      4   3426
L2 Fragmented         0      0      0      0      0      0      0      0
L2 Frames Sent     4100   1991    263   3096  32154   2051  52573  32568
L2 REJ Received     177      0      0     10    732     27      0    691
L2 Rx out of seq     31      0      0      3    247     12      0   2764
L2 Frames Corrupt     2      0     56     13      0      2      0      0
L2 FRMRs Sent         0      0      0      0      0      0      0      0
L2 FRMRs Rcvd         0      0      0      0      0      0      0      3
L2 Bytes Rcvd     44107  31204   7209 302045  4253K  22043  1235K  7634K
L2 Bytes Sent    100133  10517   3112 317645  3105K  44132  1001K  6512K
Poll Timeouts         3 246969      6      7      9      5      0      7
```

The first section shows a set of counters for each interface.

"**RX Overruns**" counts the number of times a second byte was received before a first was read by the CPU. A high number indicates that the PC is too slow for the selected serial port or HDLC card data rate. A 16550-based serial card may help if the port doesn't already use one. Failing that, you will have to reduce the baud rate.

"**TX Underruns**" is used only by HDLC cards, and counts the number of times the TX went empty while waiting for another frame byte to be loaded. A significant figure indicates the computer is too slow for the baud rate. Each underrun causes an aborted frame.

"**CRC/Framing Errors**" counts either the number of bytes received without proper stop bits (ASYNC interfaces), or the number of received frames which corrupt (HDLC cards). For ASYNC interfaces, a significant count here can indicate a hardware problem, such as a faulty line driver, serious RS232 line noise or distortion, or significant baud rate mismatch. For HDLC cards it indicates a level 1 problem, such as distortion in the TX/RX RF or audio paths, or simply a lot of packet collisions.

"**Break/Abort Errors**" counts the number of times a line "space" condition longer than 1 word interval was received. For serial ports this can indicate a faulty line driver, a faulty diode matrix on a multikiss system, or even (as recently happened to me) a malfunctioning TNC eprom. On HDLC cards it can result from insufficient audio drive from the RX, a mismatched baud rate, squelch tails (I hope to prevent this in a later version), or genuine ABORT sequences transmitted by the other end of the link.

"**CRC Errors**" shows the number of frames abandoned due to CRC or checksum error. For KISS interfaces this is only maintained if the CHECKSUM kissoption is enabled.

"**Rx Overflow err**" shows the number of times a frame was abandoned because it was too large to fit into the receive buffer.

"**Tx Overflow err**" counts the number of times that the TX couldn't accept a character (serial devices) or a frame (block devices) because the TX buffer was full. If you persistently get a high value, it indicates that the device is too slow for the data throughput.

"**Misc. errors**" counts all sorts of miscellaneous errors and the meaning is different for each type of interface. For example, on KISS interfaces it counts KISS framing errors. It is mainly for my benefit.

Following the interface stats is the ax25 level 3 counters, one for each port. Note that on a system with more than 7 ports the display may wrap. I will be addressing this in a later version.

"**L3 Frames Heard**" is the total number of ax25 level 3 frames heard, no matter who they are addressed to.

"**L3 Frames Rcvd**" is the number of ax25 level 3 frames which were addressed to the router.

"L3 Frames Sent" is the number of ax25 level 3 frames which originated at the router.

"L3 Frames Relayed" is the number of ax25 level 3 frames which were routed through our system by other systems.

After the level 3 stats, there are the ax25 level 2 counters, one per port.

"L2 Frames heard" is the total number of ax25 level 2 frames heard, whether addressed to us or not.

"L2 Frames rcvd" is the number of ax25 level 2 frames received, which were addressed to the router.

"L2 Resequenced" is the number of ax25 level 2 frames received out of sequence and subsequently used when the missing frames arrived.

"L2 Reassembled" is the number of ax25 level 2 frames successfully reassembled from fragments.

"L2 Info Received" is the number of ax25 level 2 information-bearing frames received, i.e. addressed to the router.

"L2 T1 Timeouts" counts the number of times that the ax25 level 2 T1 (frack) timer timed out, causing transmission of a poll frame.

"L2 Digipeated" is the number of ax25 level 2 frames digipeated by the port. Note that if digiport isn't zero they may actually have been re-transmitted by another port, but are recorded on the "receiving" port only.

"L2 Info Sent" is the total number of ax25 level 2 information frames sent by the router.

Stats

"L2 Info re-sent" records how many ax25 level 2 information frames were re-sent due to frame loss. A high figure here, in proportion to the "info sent" figure, indicates a problem with the RF link, the L2 settings, or the other end's system (e.g. desense, or running out of buffers).

"L2 Fragmented" is the number of ax25 level 2 information frames which were fragmented to fit the outgoing link.

"L2 Frames Sent" is the total number of ax25 level 2 frames, of any type, sent by the router, i.e. it includes all info, supervisory, and digipeated frames.

"L2 REJ Received" is the number of ax25 level 2 "reject" frames received, which indicate that the other end of the link didn't receive some of our frames. There are many possible reasons for this, some of which are mentioned in the next paragraph.

"L2 Rx out of seq" shows how many ax25 level 2 frames were received out of sequence, and indicates that some incoming frames are getting lost or trashed. A few of the possible causes might be: signal too weak, fading, other signals on channel, natural or man made interference, desense or key clicks from adjacent transmitters, poor rx audio response, low received audio, over-deviation, RF frequency mismatch, badly aligned rx, TNC hardware problems, synthesised rig taking too long to stabilise on

RX after TX, other end's synthesised rig taking too long to stabilise on TX, hum, noise, distortion or disturbances on modulated audio... the list is endless.

"L2 Frames Corrupt" is the number of frames which were dumped because they were too short to be legal ax25 level 2 frames, or were in some way invalid. It is sometimes possible for a KISS TNC, especially if running "open squelch", to send garbage to the router, or the frame may be trashed by bit errors on the serial link between TNC and router, and in either case these frames are dumped if the error can be detected.

"L2 FRMRs Sent/rcvd" shows how many ax25 level 2 "Frame Reject" frames were sent by the router, or received by it, indicating serious protocol errors or deliberate interference.

"L2 Bytes Sent" and "L2 Bytes Rcvd" simply provide a port by port breakdown of the total bytes sent/rcvd figure.

"Poll Timeouts" counts the number of times a BPQKISS TNC was polled with no response being received from it. A large figure might indicate a crashed, disconnected or unpowered TNC, or data loss on the serial link.

FRAME PIPES

An optional facility allows frames received (and optionally sent) on one port to be copied to another port. A typical use would be to allow a PMS connected to one port to see the traffic on another port, e.g. UNPROTO headers from a local BBS. Note that this is *not* the same as digipeating.

The facility is enabled by including the PIPE keyword within a port definition, e.g. PIPE=4 would pipe frames to port 4. If PIPE=0, or the keyword is omitted altogether, no piping takes place.

Pipes can be made selective, by adding a comma-delimited callsign list,
e.g. "PIPE=4 GB7PZT,KDRBBS". This will reduce the loading on the destination port, by piping only the frames with the specified calls in the destination field.

A pipe normally only allows traffic in one direction - in order to have two way traffic you must either set up a reverse pipe on the partner port, for example:

(on port 3) PIPE=4 ; Pipe frames to port 4
(on port 4) PIPE=3 ; Pipe frames to port 3

Or you may configure the pipe to be bi-directional, by setting the appropriate value in PIPEFLAG (see below)

If a frame is piped on a bi-directional pipe, the source call is remembered so that responses will be piped back to the sender. This is useful in cases where a BBS has a front end router. Simply set up bi-directional selective pipes from each user port to the BBS port. The BBS will then allow direct connection and will respond to resync requests.

Bi-directional pipes are in reality only quasi-bi-directional, because they can only send *responses* in the reverse direction. In the above BBS example, outgoing connects cannot be initiated to callsigns which haven't already used the pipe. The only way to have truly unconditional two-way piping is to set up forward and reverse pipes as detailed above.

You may pipe several ports to a single destination port, but you can at present only have one *outgoing* pipe from any port.

Pipes are capable of generating an immense amount of traffic, so use them with care - your target port MUST be capable of dealing with the traffic load, and you should avoid setting up a network of pipes which could cause an endless loop!

The types of frame to be piped can be controlled using the optional PIPEFLAG keyword, which is ignored unless piping is active.

If PIPEFLAG is not specified, the default value is 3. The value is made up by adding together the following numbers:

```
1       - UI frames *not* addressed to nodecall/alias.
2       - Non-UI frames *not* addressed to nodecall/alias.
4       - UI frames addressed to nodecall/alias.
8       - Non-UI frames addressed to nodecall/alias.
16      - UI frames transmitted by the router.
32      - Non-UI frames transmitted by the router.
64      - Allow budlisted users to be piped.
128     - Netrom frames
256     - IP / ARP frames
512     - Bi-directional piping
```

e.g. PIPEFLAG=5 will pipe all received UI frames, but not those which are originated by the router itself.

You are *STRONGLY* recommended to use the default value unless you specifically need to see additional traffic.

BROADCASTING

Xrouter has the ability to "re-broadcast" a received frame on several ports at once. This can be thought of as a "one-to-many" pipe but there are subtle differences.

Whereas pipes conduct all frames of the selected type(s) virtually regardless of source and destination addresses (unless they are selective pipes), the broadcast function acts only upon UI frames, and only if the source and destination addresses match those specified by the sysop.

Another difference is that frames may be broadcast to different groups of ports depending on the destination address. For example, FBB unproto headers from a BBS may be rebroadcast on certain user-access ports only, while mail beacons may be distributed to a different, possibly overlapping, set of ports.

You may alternatively use this feature as a "filtered pipe" between two ports, allowing only UI frames with acceptable source and destination calls to pass through.

Note that this feature does not require a "connection" to the router, it is purely an unconnected mode. I devised it to allow BBS's to distribute mail beacons and unproto mail headers. You may use it (or not) how you wish.

You should be aware that broadcast traffic takes precedence over all other frames, so an excessively high level of broadcast activity may cause other outbound traffic on the destination port to be delayed. However, it is unlikely that anyone will notice this effect unless they are seriously overloading the channel.

Broadcasting is controlled by two keywords in each PORT section of the XROUTER.CFG file. The first keyword is BCAST, and is used to specify the destination addresses which will be broadcasted.

For example, BCAST=MAIL,FBB will re-broadcast received non-digipeater UI frames addressed either to FBB or MAIL. The frames addressed to FBB will be broadcast on all ports which have FBB in their BCAST list, and those addressed to MAIL will be broadcast on all ports which have MAIL in the BCAST list. If no matching ports are found, the frame is broadcast only on the port upon which it is received. If you don't need the broadcast function, simply omit (or comment out) the BCAST keyword.

The second keyword is BCFROM, which is used to specify a list of callsigns from whom frames will be accepted for broadcast. You may use this to restrict the broadcast facility to certain senders only.
e.g. BCFROM=GB7PZT,GB7MAX says that only frames from GB7MAX and GB7PZT will be accepted for broadcast. If the keyword is omitted, the broadcast facility is unrestricted. BCFROM applies only to frames *directly* received on the port for which it is specified.

For both keywords, the list of calls should be separated by commas, and should include no spaces.

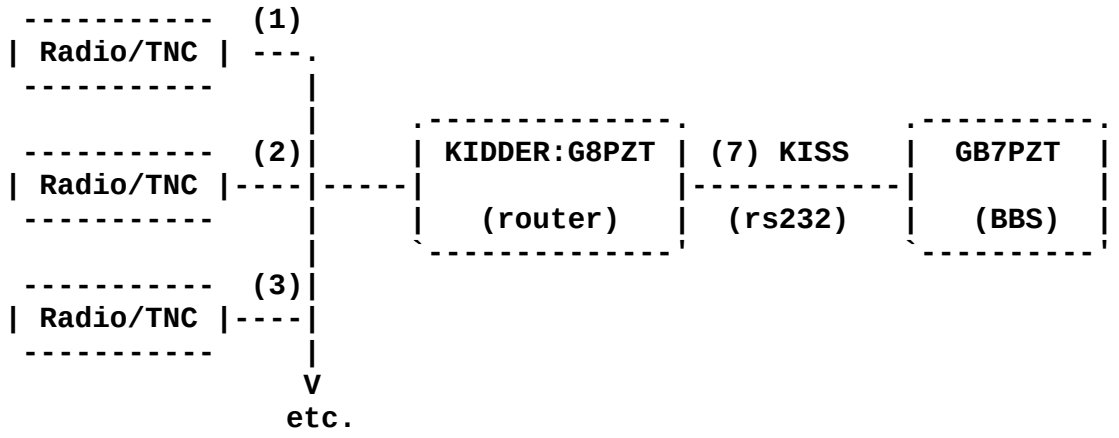
PROXY CONNECTIONS

The PROXY facility allows ax25 level 2 callers to connect "direct" to remote NET/ROM target callsigns, instead of having to connect first to the router then issue a downlink connect request.

I created this facility primarily for use in situations where a BBS or PMS is wire-linked to a "front-end" router, thus allowing the BBS callsign to be directly connectible on any port for which the proxy call is defined. You may find other uses....

A bi-directional selective frame pipe would provide a similar effect, but only for systems directly connected to the router, whereas PROXY allows the target system to be located several hops away. Pipes can handle UI frames, whereas PROXY is for connected-mode operations only.

Here's how it would be used on my set-up, which has two machines. Only the router is connected to the radios, the BBS being wire-linked to the router's port 7.



Without PROXY, level 2 users would have to connect to KIDDER:G8PZT, then issue the command "C GB7PZT", or "BBS" to connect to the BBS.

With the line "PROXY=GB7PZT" in port 1's definition (in XROUTER.CFG), users on port 1's frequency (144.850) simply connect to "GB7PZT" - the router sees the request and passes it via NET/ROM to GB7PZT, providing GB7PZT has a NET/ROM facility of course (in my case it does because GB7PZT runs on top of BPQ).

If the target system (GB7PZT in this case) is not reachable via Net/Rom level 4, the connect request is refused.

If you wish to use this facility, you must add a "PROXY=<call1>,<call2>,..." line to each user port concerned. Ports without the line will continue to function as normal. You obviously wouldn't enable it on your forwarding ports for example. You will notice that you can have several callsigns on the line, each separated by a comma.

Warning! DO NOT enable proxy on any frequency which is shared by the target system or you'll cause horrible problems (both the target and the proxy will respond to connects at the same time).

AX25 / NETROM -> TCP PROXY

This is an extension of my PROXY concept, allowing a remote TCP/IP-only system to have Netrom and AX25 connectivity.

In the previous example, GB7PZT BBS used G8BPQ node TSR underneath the BBS to provide Netrom connectivity across the KISS link. With the extended proxy, BPQ can be removed altogether and the BBS can be run in pure TCP/IP mode. This saves memory, whilst improving speed and reliability. The BBS no longer has to support multiple protocols and hardware types, that job being delegated to Xrouter.

Instead of having to connect first to KIDDER (Xrouter) then issue the awkward command "Telnet 44.131.91.2", users can simply connect directly" to the callsign GB7PZT on one of KIDDER's radio ports. Xrouter converts that connection into a TCP/IP connection with the BBS, and translates the data back and forth between AX25 and TCP/IP.

Furthermore, the BBS callsign "GB7PZT" can be connected to directly from Xrouter's command line, and also included in nodes broadcasts so it can be reached from a remote node.

This is achieved by putting an extended PROXY statement in the GLOBAL section of XROUTER.CFG, using the following format:

PROXY=<call> <alias> <qual> <ipaddr> <portnum> [passwrd]

For example:

PROXY=GB7PZT KDRBBS 150 192.168.0.4 8888 bloggs

<call> is the netrom and ax25 callsign for the proxied system.
<alias> is the netrom / ax25 alias for the proxied system.
<qual> is the netrom "quality" (0 - 255) controlling visibility.
<ipaddr> is the proxied system's IP address.
<portnum> is the TCP service port number of the proxied system.
<passwrd> is optional password sent to proxied system upon connect.

AX25 and Netrom are pure binary channels, whereas standard telnet is not. The proxied system must provide a pure binary service port in order to make full use of this facility for compressed forwarding etc. G8PZT's "XServ" BBS provides such a facility on port 8888.

If you are a software author interested in adapting your software to work with this proxy, the following information will be relevant:

Upon connection to the proxied system Xrouter sends one line of text ending in <CR><LF>, containing one or more space-delimited fields. The first field is the callsign of the connectee in the form "G8PZT-7" (ax25) or "G6YAK-2@GB7BM" (netrom). The second field is an agreed password which verifies the proxying system. Thereafter, the link switches to pure binary, and in accordance with AX25 practice both systems must send line ends as <CR> alone.

There is no limit to the number of proxies you may configure.

NETROM -> AX25 PROXY

This is similar to the NetRom -> TCP proxy described above, but is intended to allow an AX25-only system to have a NetRom presence.

This is enabled by putting an extended PROXY statement in the GLOBAL section of XROUTER.CFG, using the following format:

PROXY=<call> <alias> <qual> <ax_call> <portnum>

For example:

PROXY=MB7UYL UYLBBS 150 G6KDR-3 7

```
<call>    is the netrom and ax25 callsign for the proxied system.
<alias>   is the netrom / ax25 alias for the proxied system.
<qual>    is the netrom "quality" (0 - 255) controlling visibility.
<ax_call> is the proxied system's AX25 L2 callsign.
<portnum> is the radio port the proxied system is connected to.
```

The above example creates a Netrom node whose callsign is "MB7UYL", alias "UYLBBS", with quality 150. Anyone who makes a connection to either of these addresses will instead be connected to the AX25 system "G6KDR-3", attached to Xrouter's port 7.

Warning: Be *very* careful when mixing proxies and pipes, or you will end up generating lots of FRMR's, and possibly crashing the system. These are powerful tools and must be used carefully.

TELNET PROXY SERVER

The need may arise for a TCP/IP host on the LAN to connect first to Xrouter, before making an ongoing connection from Xrouter to a target host. Using Xrouter's normal telnet port 23 for this purpose is not always successful, especially if the link is required to carry binary data, due to the internal end-of-line translations and Telnet command sequences required by the Telnet protocol.

The Telnet proxy on TCP port 2323 provides a solution to this problem, allowing full binary connections not only to TCP/IP targets, but also to Netrom and AX25 targets.

Upon connection to port 2323, the caller may, or otherwise, be required to answer a security challenge, according to the security criteria specified in ACCESS.SYS. The options are: no challenge, callsign only, or callsign plus password, and all of these can be made dependent on the caller's IP source address.

After gaining entry, the user is presented with a short text explaining the syntax for the outgoing connection, followed by a short prompt ">". This text is read from file TELPROXY.MSG, so you may adjust it to your requirements. The user may now issue a telnet, netrom or ax25 downlink connect command. Xrouter will respond with "Trying ...".

If the downlink is successfully made, connection is reported, then the stream becomes pure binary for the duration of the session.

If the downlink fails to connect, an error message is sent before the uplink is terminated.

GB7PZT BBS has been successfully using this proxy for some time, to perform binary compressed forwarding with a variety of netrom, ax25 and TCP/IP targets.

CHAT SERVER

Xrouter has a built-in chat server, i.e. a device which allows groups of users to hold live multi-way conversations without the usual problems of having to manage several streams at once. It is available to all callers, and is accessed by using the CHAT command at the main prompt, or by connecting directly to its callsign or alias. TCP/IP users can additionally reach it by TELNETting to port 3600.

The Xrouter chat system has 32767 "channels", each of which can support an unlimited number of users, so it is possible for groups to have their own "private" channel, or to reserve certain channels for specific topics. Channel 1 (one) is the default at log-on and is used as a sort of "calling channel", from where users may QSY as required. Users may "join" as many channels as they wish, so they may take part in several separate conversations at once.

Channels 1 to 255 are "local" to each chat server, and the remaining channels 256-32767 are "global", i.e. they are linked with all other Xrouter chat servers, providing the sysop sets up at least one link to another server.

In addition to the "positive" channel numbers, there are another 32768 channels numbered 0 to -32767. These correspond to channels 0 to 32767 on the "Tampa Ping Pong" system. Xrouter allows only limited interconnection between the two systems, because the channel layouts and topologies are completely incompatible. Xrouter chat was specifically designed for use on an anarchic, slow radio network, whereas Ping-Pong requires a more planned network topology to avoid loops, and is largely carried on Internet links. The sheer volume of chat in the Ping-Pong system would overwhelm radio links.

Data received from Ping-pong is not propogated via the Xrouter chat interlinks and vice versa. In effect, Xrouter can be a stub Ping-Pong host, but not part of the Ping-Pong network.

Users may listen on any number of positive and negative channels simultaneously, but may only send on one channel at a time. Once logged onto a channel, anything sent by the user is copied to all other users of that channel, except lines beginning with a forward slash (/), which are interpreted as chat server commands. The copied text is preceded by the channel number and the sender's callsign and name, to allow the recipients to identify who sent it.

All chat server commands begin with a forward slash (/), and most may be abbreviated to the initial letter. The /? command will show the available commands and syntax, while /HELP will give more details.

The /NAME command is used to enter the user's first name, and serves as a "login". Users are not permitted to join any channels until they have supplied a name. TCP/IP users must additionally supply a callsign with the /USER command.

CHANNEL, /JOIN and /LEAVE are used to select the desired channel, WHO shows the active channels and who is using them, and /QUIT terminates the chat session. The full command set is shown in more detail in the command reference section.

Users who log on to more than one chat server at once are treated as separate entities, and must supply their name and callsign on each log-in.

If transaction logging has been enabled in XROUTER.CFG, chat server activity will be logged in file CHATSERV.LOG.

The chat server is fully automatic and requires minimal setting up. It has its own callsign and alias defined in XROUTER.CFG, plus a list of link partners.

The alias (6 chars max) should preferably end with "CHT" and begin with something geographically relevant, e.g. BHMCHT for Birmingham, LDSCHT for Leeds etc., so they can be identified easily in node tables. The only other setting up required is to install the chat help files in the directory help/chat.

At present, only Netrom links are allowed between Xrouter chat peers (so the peer must be in your node table), and only TCP/IP links are allowed to Tampa Ping-Pong peers. Peer links are specified using the

CHATLINKS keyword in XROUTER.CFG, or may be added and removed at any time using the /LInks command.

Links to Xrouter chat peers are specified thus:

CHATLINKS=<netrom_call>,<netrom_call>,...
e.g. CHATLINKS=GB7GH-7,GB7BM-8,N0LBA-8

or at the chat command line:

/LI ADD <netrom_call>
e.g. /LI ADD GB7BX-9

Unilateral linking is not allowed. You *must* co-ordinate it with your peers, such that you are in their CHATLINKS list and they are in yours.

Links to Tampa Ping-Pong converse servers are specified in Xrouter.cfg as follows:

CHATLINKS=<peername> <ip_address>:<tcp_port>
e.g. CHATLINKS=brmcht 80.195.22.67:3601

Alternatively, use the command "/LI ADD <peername> <ipaddr>:<port>",
e.g. "/LI ADD brmcht 80.195.22.67:3601"

You should avoid linking to peer servers via slow links. If the link isn't up to the job, frames will be dumped.

APRS

In addition to NetRom and IP routing, frame piping, broadcasting and normal digipeating, Xrouter supports APRS (Automatic Packet Reporting System) generic digipeating for RELAY, WIDE, TRACE, TRACEn-N and WIDEn-N aliases.

Generic digipeating is configured on a port by port basis, using flags in "digiflag" as follows:

```
4       Enable RELAY generic APRS digipeating.
8       Enable TRACE and TRACEn-N APRS digipeating.
16      Enable WIDE and WIDEn-N (Well sited stations only!)
```

In quiet areas, you may wish to mix APRS and normal connected-mode operations on the same port, and that is the default if you enable any of the above flags.

In most areas, APRS tends to be on a separate frequency reserved for "unconnected nets", and you may wish to prevent people from connecting to the router or downlinking from it on your APRS-only ports.

The CFLAGS keyword can be used in the PORT section of the CFG file to control uplinking and downlinking as follows:

0 Prevent uplinking and downlinking.
1 allow uplinking only.
2 allow downlinking only.
3 allow both up and downlinking.

You may choose to allow regular digipeating (i.e. using the router's normal callsign or alias) on APRS ports, allow regular UI-only digipeating, or disable regular digipeating altogether, by manipulating bits 0 and 1 of DIGIFLAG.

Whilst I urge all sysops to include an APRS position in their normal IDTEXT, a dedicated APRS port may require a more detailed and cryptic ID beacon, therefore you may define a different IDTEXT for each port if necessary. A "regular" port would include a position followed by some human-readable text, whereas the APRS-only ports would include additional data such as power / height / gain / direction, wind speed, bowel movements etc., in encoded format.

The IDTEXT may be sent via digipeaters by including the IDPATH keyword in the relevant port configuration section of XROUTER.CFG.

Xrouter checks for its own callsign or alias in previously-passed digipeaters to prevent digi looping. It will not digipeat frames it originated, and will not digipeat the same frame within 9 seconds.

The "DX [port]" command show the best received APRS DX. It only works if the router's APRS position has been defined by including it in the ID beacon.

The DXFLAGS keyword in the .CFG file controls whether or not the DX list will contain callsigns heard via digipeaters.

VK2DOT APRS SETUP:

The following setup is for the VK2DOT system using XR32. We do not use IGATE,

XROUTER.CFG

```
;--------------------------------------
; 145.175Mhz 1200bps APRS LAN
;--------------------------------------
;
INTERFACE=1
      TYPE=ASYNC
      COM=1
      SPEED=2400
      PROTOCOL=KISS
      FLOW=0
      MTU=230
ENDINTERFACE
;
; ====================================================================
;   Example APRS-only port.
;   Note the use of an alternate IDTEXT, and the use of CFLAGS to disable connected mode
;         operations, thus MAXFRAME,FRACK,PACLEN etc are not needed.
;
PORT=1
      ID=COM1: 145.175Mhz 1200bps APRS
      INTERFACENUM=1
      PORTCALL=VK2DOT-13
      RFBAUDS=1200
   CFLAGS=3
;   CFLAGS=0       ; Prevent up/downlinks on this port
   DIGIFLAG=5     ; Digi only UI frames addressed via RELAY
   IDPATH=APRS,SYS2-2
;
;   Default digipeater path for APRS frames originated by APRS messaging shell and Igate. If you
;         omit this, the frames will sent without any digipeaters. Messaging shell users may
;   override this path.
;
      APRSPATH=WIDE2-2
      MAXTT=0
      QUALITY=0
      NODESINTERVAL=0
      IDTEXT=!3323.21S/15121.42E#PHG5820 VK2DOT XRouter
      IDPATH=APXR87,WIDE2-2
      TXDELAY=350
      TXTAIL=50
      RETRIES=10
      MAXFRAME=1
      MHEARD=22      ; Nice big MH list
ENDPORT
;
;--------------------------------------
```

APRS / UI-View queries

Xrouter responds to the following general queries:

?APRS? All stations query.

~\xFD~ UI-View general query.

?IGATE? Igate query.

The response to the first two is the ID beacon for the port, which, if my recommendations were followed, should contain the APRS position and station type. The response to the ?IGATE? query shows the message and local user counts.

The following "directed" queries (directed at portcall) are supported:

?APRSD Directly heard stations list.

Responds with a list of stations heard directly on the receiving port (i.e. not via digipeaters or via 3rd party networks)

?APRSM Un-delivered messages query.

If there are any un-delivered or expired messages addressed to the sender of the query, they are re-activated and transmitted on the port which received the query.

?APRSP Station Position.

If the sysop has defined the router's position, it is sent in response to this query.

?APRSS Station status.

The response consists of the software type and version, plus a list of the enabled generic digipeater calls.

?PING?
?APRST Trace Route.

Both of these return the route by which the query was received.

~\xFE~n UI-View "ping".

The response to this query is a UI-View ack for the ping id.

~\FC~n UI-View "DX" query.

Responds with a UI-View ack, followed by details of the best DX heard directly by the router (digipeated packets are NOT dx as far as I'm concerned!)

APRS MESSAGING

Xrouter contains an APRS messaging shell, which allows NON-APRS users to exchange messages with APRS / UI-View users, and also to send and read bulletins and announcements.

Messaging mode is initiated by from the main command prompt by entering the command "AMSG <port>", where <port> is number of the radio port on which messages will be sent and received.

Within messaging mode, all commands begin with a forward slash (/), and anything else is treated as message text for transmission. The commands are as follows:

```
/A[nnouncements]    Show announcements
/B[ulletins]        Show bulletins
/C[ancel] [#]       List / cancel unacked message(s)
/D[irects]          Show directly heard stations
/H[elp] [cmd]       Display command help
/Monitor [on|off]   Query / set traffic Monitor mode
/Q[uit]             Quit (exit)
/T[arget] [call]    Query / set target for msg
/U[iview] [on|off]  Query / set UI-View mode
/V[ia] [digis]      Query / set digipeater path
/X                  Exit
```

Only the first letter of each command needs to be supplied. A few are worthy of further explanation....

The /D command shows a list of all the stations heard directly, i.e. not via digipeaters or 3rd party networks.

Before any type of message or query can be sent, the user must specify a "target" address, using "/T [call]". For messages, the target will be a callsign. For bulletins the target should be BLN#*, where "#" represents a single digit, and "*" represents the bulletin category of up to 5 characters. Announcements use the same format as bulletins, except that "#" represents a non-digit. Attempting to send a message without first defining a target will result in an error response. The target remains in force until a new target is specified. The current target can be displayed by entering "/T" alone, or cleared by entering an invalid target, e.g. "/T .".

Outgoing messages and bulletins are re-transmitted at intervals until either an acknowledgement is received, or too many retries have taken place. Bulletins are re-transmitted every 20 minutes for 4 hours, whilst announcements are re-transmitted every hour for 4 days. Messages are initially re-transmitted after 10 seconds, then the interval doubles with each re-send. When the interval exceeds approximately 1.5 hours, the message is expired and re-transmission ceases. The "cancel" command allows the re-transmission of outgoing messages and bulletins to be cancelled at any time before expiry.

The /M (Monitor) command allows other APRS and UI-View message traffic on the channel to be watched. The default is "off". Entering /M by itself shows the current state.

The /U (Ui-View mode) command sets the type of outgoing message to be used. The default is "off", which means that all outgoing messages will be in APRS format. If turned "on", outgoing messages will be in "UI-View" format. In either mode, both types of message can be received. UI-View messages will display with a tilde (~) between the message and its ID, whereas APRS-format messages will display with a curly opening bracket ({) if a message ID was supplied. In UI-View mode, "\<decimal>" will send a UIVIEW message whose text portion contains a single byte of value <decimal>, e.g. "\254" will send a PING request.

/Q (quit) and /X (exit) are identical in function, exiting message mode and returning the user to the router's main command prompt.

The /V (via) command sets the digipeater path for outgoing messages, or if used by itself displays the currently set path. The path defaults to the port APRSPATH specified in XROUTER.CFG. In APRS mode, the destination call is fixed at APZ###, where ### is the 3 digit Xrouter version number, whereas in UI-View mode the destination call is set by the /Target command.

The /H (help) command is used to display help for the messaging commands. If no argument is supplied, a very brief (low bandwidth) command resume is displayed. If the help files are installed, "/H *" will list the help available, and "/H <cmd>" can be used to obtain more detailed help for <cmd>, e.g. "/H /V". Note that the leading slash of the argument is ignored, so "/H V" is equally valid.

Messaging notes

If Xrouter receives an APRS message whose target address is a user currently logged into the APRS messaging shell, the message is delivered to the user and, if there was a message ID, an acknowledgement is sent. Each re-send of the message is acknowledged, because a re-send probably indicates that the sender didn't receive the previous ack.

If the same message is received twice within 30 seconds, the second copy is ignored. This helps to eliminate duplicates received via different digipeater routes.

Expired messages are retained for 1 day before being deleted. During this interval they will be reactivated if a "?APRSM" query is received from the target station. Outgoing bulletins and announcements are not retained after expiry. Incoming bulletins are retained for 4 hours after last received, and incoming announcements are retained for 4 days after last received.

The APRS spec limits the maximum message length to 67 characters. Because a message ID of up to 6 characters is appended to the message, Xrouter splits messages longer than 61 characters into separate messages no longer than 61 characters (excluding ID) each.

All APRS facilities are an ongoing experiment and may be liable to change as development continues. The so-called "APRS Protocol Reference" is rather fuzzy in places!

APRS Internet Gateway (IGATE)

Xrouter has an inbuilt APRS Igate, which allows received APRS packets to be gated to an Internet APRS server, and packets from the server to be gated to RF. You will need an APRS port and an internet connection to take advantage of this.

Igate operation is controlled mainly by configuration settings in file IGATE.CFG. If the file isn't present, the igate daemon can be started, but nothing will happen. The file includes keywords which specify the Internet APRS server addresses, various timers controlling connections, the filtering rules for gating packets, and the logging options. In addition to the settings in IGATE.CFG there are a few in XROUTER.CFG, which control whether or not the Igate daemon starts at boot-up, and which ports may send to and receive from the Internet.

Choosing and specifying Internet Servers

The choice of server, and the TCP port to use on that server, depends on where in the world you are located, and what sort of data you're interested in. There are several types of server, and the services they provide aren't always comparable.

Some servers provide a "raw" data feed containing APRS activity from all over the world, which can amount to a considerable volume. Some also provide "filtered" or "local" feeds, e.g. Ohio-only or messages-only. There's no point in connecting to a "raw" feed if there is a "local" feed available, as you will merely be wasting CPU time and internet bandwidth downloading data you are going to discard. APRS message-only feeds tend to be on port 1314, whereas "raw" feeds tend to be on port 10151, or 2023 if it's a Ahub server.

Using a browser to connect to http://server_address:14501 sometimes produces a status page containing useful server addresses. You could start your search at "first.aprs.net". There are various web sites which may assist your search, e.g. "www.dididahdahdidit.com/" (what a difficult and error-prone address to type!). I suggest you ask questions on the APRS newsgroups if you are unsure, but if you're reading this file you probably know more than me about APRS anyway :-)

In IGATE.CFG you may specify as many servers as you wish, each on a separate line with the following format:

SERVER <address>:<port>

e.g. SERVER 128.196.58.1:1314

The current version doesn't support name resolution, so you MUST supply the IP address not the domain name. The port number must be separated from the IP address by a colon so that the combined address:port forms a contiguous string of characters with no spaces.

The servers are tried in rotation, starting with the first on the list. If a connection attempt fails, or the link subsequently closes, the next serer is tried. When the end of the list is reached it starts all over again at the first one. This is probably a crummy algorithm, whose only merit is that it distributes the load across various servers, and I may have revised it by the time you read this! The behaviour is modified by various timer settings - see below.

You may find out which server is currently in use by using the "TCP STATUS" command.

Connection Timers

There are various keywords which may be used in IGATE.CFG to control the timings associated with connections to the Internet servers, and they are as follows:

WAIT <secs> -- Time to wait for connection establishment.

Specifies the number of seconds to wait for connection after sending a connect request. If not specified, the default is 60. If you have a permanent Internet connection, you may wish to set a lower value, but if you're using dialup you may wish to make it longer, e.g. 90 secs to allow time for dialling and modem negotiation. If no response is received from the server within the WAIT interval, the next server in the SERVER list will be tried.

PAUSE <secs> -- Interval between successive tries.

Specifies the number of seconds to wait between successive connection attempts to the same server. Default is 60 secs. If a server goes down or fails to respond, there's no point in aggressively trying to connect. For one thing, if you have connection logging enabled you will build a big logfile! This timer is mainly of use when you have only one server in you SERVERS list, or when several servers go down.

MAXTRIES <n> -- No. of failed connect attempts allowed.

If a server consistently fails to respond to connect attempts, there's a good chance it has gone temporarily or permanently off line. The MAXTRIES value specifies the maximum number of failed connect attempts allowed before a server is ignored, and the default is 10. If the limit is reached, that particular server will not be retried until the SKIP interval (see below) expires.

SKIP <secs> -- "Blacklist" interval.

Specifies the amount of time for which a server will not be tried after MAXTRIES failed connect attempts. This effectively removes a server from the list for the SKIP interval, after which it is placed back on the list. Default is 3600 secs (1 hour).

Packet Filtering

It is unlikely that you will find an internet server which provides a feed tailored exactly to your needs, so the Igate contains powerful packet filters, which can be applied to traffic gated in both directions. The filtering rules are specified in IGATE.CFG using IFILTER (internet to packet) and PFILTER (Packet to internet) statements, the general form of which are as follows:

xFILTER <from_call> <to_call> <text>

Each field may include the following wildcards:

```
*     Matches zero or more characters.
?     Matches any single character.
#     Matches a single digit.
@     Matches a single alphabetic character.
\     Escape - interpret next character literally.
```

The '*' character may only be used at the end of a field.

For example: "IFILTER G#@@@* * :*" will accept from the internet feed only those packets sent by valid G0 to G9 + 3 letter callsigns, addressed to anyone, which contain APRS messages.

The '\' (escape) character causes the next character to be interpreted literally, instead of as a wildcard, e.g. "*" will match '*' and "\\" will match '\'. A caret '^' at the start of any field will invert the sense of the whole test, causing matching packets to be REJECTED, e.g.

```
IFILTER ^M7ABC  *    *
IFILTER *       ^APZ244*   !*
```

The first statement will reject all packets from M7ABC, and the second will reject all static position reports sent to the destination APZ244 (e.g. if they can't be trusted). Rejection statements MUST be placed at the beginning of the filter rules, before any catch-alls.

The maximum length of each field (pattern) is currently 10 characters. If you feel this isn't enough, let me know. There is no limit to the number of xFILTER statements you may specify. If no rules are specified, nothing will be gated.

All valid APRS and UI-View packets (except 3rd party packets, and those with NOGATE or RFONLY somewhere in the digi path) received by the router are offered to the PFILTER, providing the appropriate DIGIFLAGS are set (see below). Mic-E packets are decoded to text before being offered to the Internet servers because they may contain characters which won't pass through the servers. This decoding takes place before filtering.

You should be VERY careful when designing your filter rules, as you could quite easily overload the RF channel by attempting to gate too much data to it. There is little point in gating non-local data.

To reduce unnecessary traffic, APRS "messages" are only gated to RF if the recipient has been seen locally on RF within the last hour.

Radius of Interest

In IGATE.CFG the statement "RADIUS <km>" sets a radius in Kilometres from Xrouter's position. Position reports from within this radius will be gated to RF, providing they pass the IFILTER rules. The default radius is 100Km. Setting "RADIUS 0" allows unlimited gating of positions.

Controlling gating direction / port(s)

Regardless of any other settings, an Xrouter port will not broadcast packets received from the Internet, or offer received packets to the internet, unless the appropriate bits are set in the port DIGIFLAG. You should add the following values:

```
 64   Enable gating from Packet to Internet.
128   Enable gating from Internet to Packet.
```

Packets accepted (i.e. passing IFILTER) from the Internet will be broadcast on ALL ports which have bit 7 of DIGIFLAG set, so be careful not to lazily set a value of 255!

Activity logging

Igate activity can be recorded in the file ./LOG/IGATE.LOG by including a LOG keyword with non zero argument in IGATE.CFG. The argument is a number made up as follows:

1 Record Igate daemon start and stop events.
2 Record Inet server connections / disconnections.
4 Record frames sent from Internet to Packet.
8 Record frames sent from Packet to Internet.

The "record frames" options can generate a lot of data, so they're intended mainly for testing purposes, e.g. making sure your filters are correctly set. If the logfile grows too big, DOS may take an appreciable time to perform the disk writes if you aren't running a write cache. This might affect router performance, so you are advised to periodically rename or otherwise remove the logfile, allowing a fresh one to start.

Starting and stopping the Igate daemon

The daemon may be started and stopped at any time with the commands "START IGATE" and "STOP IGATE". If the daemon is already running, attempting to restart it will simply produce an error message, as will trying to stop it if already stopped.

The IGATE.CFG file is re-read each time the daemon is started, so the configuration can be changed simply by editing the file using the PZTDOS line editor, then re-starting the daemon. Editing can take place while the daemon is running.

The daemon may be started automatically when Xrouter boots, by including IGATE=1 in the "global" section of XROUTER.CFG.

Getting an Internet Connection

None of the above is of any use if you haven't got some means of getting a connection to an Internet APRS server. You therefore need to set up dial-up-networking (see elsewhere) with a PSTN modem, or use a cable modem, or use an external router or Internet Connection Sharing (ICS) machine, such as one running Windows 98.

I chose to use Windows 98 with a dial-up connection, since the machine was already in use for my email and web activities. Your Windows machine and your Xrouter machine will need to be interconnected, either by SLIP or PPP null modem link, or by Ethernet. If Xrouter is located on the Windows machine, you will need two Ethernet cards in that machine, as it can't share a card - sorry :-(

You will also need to install (if not already installed) Windows "Internet Connection Sharing" (ICS). A wizard will guide you painlessly through the process of screwing up your machine :-) No, seriously I think my troubles arose because I already had several IP addresses in use on that machine and was unsure what I was doing. If you know anything about ICS, please share your wisdom to make this document more useful! (I know next to nothing about Windows)

Basically, ICS configures your windows machine's Ethernet port with the "private" IP address 192.168.0.1, and you simply configure other computers on the Ethernet LAN to use any address between 192.168.0.2 and 192.168.0.255. You should choose the option to use a static IP address not a "server assigned" address.

On the Xrouter machine you need to install a suitable Ethernet card driver plus ETHDRV.EXE, and set up an Ethernet port (see XROUTER.DOC for details). The port should be configured to use one of the "private" IP addresses other than 192.168.0.1 - you may do this by setting the "global" IP address and overriding it on the radio ports, or by setting only the Ethernet port to use the private address.

Finally, you need to set up IP routing to and from the Windows machine. If the wizard has done its job properly, Windows 98 will automatically route all "private" IP packets to the ethernet port, and all you have to do is tell Xrouter how to route stuff to Windows. I can't be precise on this, since your requirements will be different to mine. Basically you must arrange to route Internet-bound packets towards 192.168.0.1 on the Ethernet port. You can either do this using a ROUTE DEFAULT or using ROUTE ADD's for specific addresses.

In my case I have ROUTE ADD statements which route radio-reachable 44-land datagrams out on the appropriate radio port. I then have a "ROUTE ADD 192.168.0.0/24 * 14 d" which allows Xrouter to send datagrams directly to my other machines on port 14 (the ethernet LAN), a "ROUTE ADD 44.0.0.0/8 x x" which routes the rest of 44-land to my neighbour, and finally a "ROUTE DEFAULT 14 192.168.0.1 d" which routes everything else to the Windows machine for sending to the Internet.

Remember: The software is well proven. If it doesn't work, you haven't configured it correctly!

APRS SERVER

Xrouter includes a rudimentary APRS server, which enables suitable APRS clients, such as UI-View, to connect to it on your LAN and exchange APRS traffic. The number of simultaneous clients is not limited.

TCP Port Number

The server listens for incoming connections on TCP port 1448. There is a tradition of choosing port numbers which represent the frequencies used by APRS, so I chose that port number because many European countries use 144.800 MHz, hence it is easy for me to remember (sorry USA :-).

Overview Of Server

The following APRS packets are sent to clients:

- APRS traffic heard on any of Xrouter's radio ports.
- Traffic sent by other clients.
- Traffic sent by users of Xrouter's APRS messaging shell.
- Filtered traffic from Internet APRS servers (if Xrouter's IGATE is connected to an internet APRS server)

APRS packets from clients are distributed as follows:

- To other clients, excluding the sender.
- To Xrouter's APRS messaging shell.
- To radio ports (only if client is fully registered)
- To Internet APRS servers via IGATE (if IGATE is running).

APRS Registration and Login

Registration of clients is necessary to prevent unauthorised use of radio frequencies by unlicensed people.

This may seem overly restrictive if your system is only used on a private LAN, but if you are connected to the Internet, it is essential. For example, if an unlicensed user connects to your server via the Internet, he must be prevented from sending traffic to your local RF ports. He must also be prevented from sending traffic via your IGATE (if it is enabled) into the Internet system, and thence to other people's RF ports.

Therefore, clients are required to complete a log-in process before they are allowed to send any traffic. Log-in is not required for receive-only operations.

The server accepts two different types of login. When a user registers an APRS client program such as UI-View, he receives a "validation number" which Xrouter will use in combination with the callsign to verify the user. A verified user may send traffic to local RF ports, or if IGATE is active, via other IGATES.

If the user has not registered his copy, the default validation number of "-1" allows him to send traffic to other clients and to the Internet, but that traffic will not be gated locally to RF, and is marked in such a way that it will not be gated to RF by other IGATES. This allows unregistered clients to communicate with each other via the Internet, but not via RF. The client may only send APRS packets whose source callsign matches the login callsign.

The alternative login system allows clients to verify themselves by supplying their callsign and a password which has been agreed with the sysop. The password replaces the validation number in a login string.

The login string is the only "command" accepted by the server, and must take the form: "user <callsign> pass <password>", where <password> could be either a validation number or a text string, for example, "user g8pzt pass beanzmeanzheinz", or "user g7zzz pass 32751". Login is not acknowledged.

The Client Connection

There is no time-out on client connections, therefore there is no need for the client to send "keep-alive" signals.

If the client connection is too slow to cope with the incoming data rate, packets to the client may be discarded.

Local <> Internet Server Gating

If IGATE is not running, no packets will be gated to or from the Internet.

Packets received from the Internet are not gated to clients unless they satisfy the IFILTER filtering rules in IGATE.CFG. Likewise, packets received from clients are not gated to the internet unless they satisfy the PFILTER rules.

Packets in "third party" format, packets which do not include the network identifier "TCP/IP" in the digi path, and packets which include the dummy callsigns NOGATE or RFONLY in the digi path, are not gated to the Internet.

Using UI-View as a Client

Select Setup/APRS Server Setup.

In the box marked "Select A Server", enter the hostname and port number of Xrouter's APRS server, e.g. "myserver:1448" or "192.168.0.2:1448". (On my Windows98 system, neither form would work until I added a suitable entry into the HOSTS file in the WINDOWS directory.)

Check the boxes marked "Open the gateway" and "Gate local messages".

If you have a registered version of UI-View, check the box marked "APRServe log on required", and enter your validation number.

If your copy is unregistered, you will be able to log on with the default validation number of -1, but your packets will not be gated to RF.

To obtain full privileges using an unregistered copy, you must have a password, which must be registered with your callsign in Xrouter's USERPASS.SYS file. The callsign must not include the SSID, e.g. if UI-View's callsign is "G8PZT-11", the entry in USERPASS.SYS should simply be "G8PZT". Un-check the "APRServe log on required" box, and in the box marked "Text to send upon connection" enter UI-View's callsign (with SSID) and your password in the following form:

user g8pzt-11 pass virago

TNC2 EMULATOR

This is a feature which allows RS232 devices such as weather stations, dumb terminals, GPS and telemetry devices to send and receive packets as if connected to a real TNC2.

For example, imagine you have a weather station which is designed to be connected to a TNC via an RS232 cable. Now imagine that you already have an APRS port on your Xrouter. How would you get you weather station on air? You could use an additional TNC, radio and antenna, but that would be a pointless duplication of equipment. Far better to set up Xrouter to emulate a TNC so that it can interface directly to the weather station, allowing the weather station to send to, and receive from, the existing APRS port.

Or you may have a dumb terminal connected to Xrouter via an RS232 cable, and use it to monitor any port, make connections with other stations etc. This is completely independent of Xrouter's command interface, and does not reuire a session with the node.

Perhaps you wish to keep an eye on the channel with a data logging program, or send the channel activity to a serial printer? The possibilities are limited by your imagination.

The emulator is configured by defining an INTERFACE with TYPE=ASYNC and PROTOCOL=TNC2. Choose SPEED to suit the peripherals, and MTU=256. You don't attach a port to this interface.

Example:

```
INTERFACE=5
   TYPE=ASYNC
   COM=1
   SPEED=19200
   PROTOCOL=TNC2
   MTU=256
ENDINTERFACE
```

Standard TNC2 commands recognised are Ctrl-C, AUTOLF, CONNECT, CONVERSE, DISCONNECT, ECHO, FLOW, K, MCON, MONITOR, MYCALL, PORT, and UNPROTO. If you need any others I will be happy to add them.

You may set the TNC's callsign (using MYCALL) completely independently of Xrouter's callsign, and select any port with the PORT command.

You can have as many TNC emulators as you wish, providing you have an RS232 port for each one. You should preferably use a different MYCALL or SSID for each one if there is any chance of more than one TNC being used on the same radio port.

All settings are saved to the file TNCn.CFG where 'n' is the interface number. This file is created if it doesn't already exist, and is read at start up, so the TNC always returns to its previous configuration. The file contains binary data, so you must not attempt to edit it.

Note: The emulator does not accept incoming connections - I can't see the point, but if anyone really wants that facility I will add it.

AXIP TUNNELLING

AXIP is AX25 "encapsulated" within (i.e. carried in the payload section of) IP datagrams. It enables AX25 systems to communicate with each other via any TCP/IP network such as the Internet. The AX25 links thus created can in turn support Netrom and amateur TCP/IP.

Assuming you have a prospective AXIP partner, you would set up an AXIP wormhole as follows:

1) Configure and test IP routing between you and your partner.If you don't have reliable IP routing there's no point in proceeding!

   If you are linking via the Internet, it makes sense to use the Internet IP addresses for this purpose, rather than the amateur ones, because the routing is more reliable and the throughput faster.

2) If you wish to use your partner's hostname (e.g. g8pzt.ath.cx) instead of the IP address (e.g. if the partner has a dynamic IP address), your system needs access to a Domain Name Server (DNS).

   If Xrouter is connected directly to the internet via a relatively dumb device such as a telephone or cable modem, it can obtain the address of a suitable DNS via PPP or DHCP negotiations.

   If it is connected indirectly, via a more sophisticated device such as a firewall, an Internet Connection Sharing system (Windows98 ICS) or a good quality xDSL modem, it is likely that such a device will have an inbuilt "DNS proxy server", which will answer DNS requests on behalf of the LAN. If the device is DHCP-capable, and you have configured Xrouter to use DHCP, Xrouter will automatically know the address of the DNS proxy.

   If you are not using DHCP or PPP you must include a valid "DNS=..." statement in XROUTER.CFG, e.g. "DNS=88.23.207.1". Note that you must supply the IP address of the DNS, not its hostname.

   Verify that "PING <hostname>" resolves the address properly.

3) Add an AXIP interface to XROUTER.CFG as follows:

```
INTERFACE=7
   TYPE=AXIP
   MTU=256
ENDINTERFACE
```

   (Choose the interface number to suit yourself).

   This interface can support an unlimited number of AXIP ports. You may define more than one AXIP interface if your ports need differing MTU's.

   Only TYPE=AXIP and MTU= are required, all other parameters will be ignored (at present).

4) For each AXIP partner, add an AXIP port similar to this:

```
PORT=5
   ID=AXIP link with WA3DXX
   INTERFACENUM=7
   IPADDRESS=44.131.91.245
   IPLINK=55.73.88.69
ENDPORT
```

   You must specify at least ID, INTERFACENUM, and IPLINK.

IPLINK is the remote host's IP address or hostname. It is more efficient to use IP addresses, if you are able to do so, because it removes the need to resolve the hostnames. The "protocol number" for AXIP is fixed at 93 (decimal).

IPADDRESS is only required if you wish to encapsulate IP traffic within the AXIP frames, and then only if it differs from Xrouter's main IP address.

MAC parameters such as TXDELAY, TXTAIL, SLOTTIME, PERSIST, FULLDUP, SOFTDCD etc. are meaningless for AXIP, but FRACK, RESPTIME, PACLEN, MAXFRAME, QUALITY etc. operate as normal.

On fast internet links you may wish to use a much lower FRACK, say 2000ms, than on radio. I would not recommend reducing it below 1000ms, as it needs to be *at least* twice the worst round-trip time plus the other end's RESPTIME.

RESPTIME is probably the one which will have most effect on the responsiveness of the system, because it controls the time delay between receiving a packet and sending an ACK. It should be just a little more than the time it takes to receive a maximum length packet. For example, at a data rate of 56Kbits/sec, a 256 byte packet lasts less than 50msec, so RESPTIME=50 would be adequate.

5) If Xrouter is indirectly connected to the internet via an intermediate router, that router will probably be using some form of NAT (Network Address Translation) to share one "public" IP address between several systems on your LAN. The "front end" router will probably route outgoing AXIP without problem, but it will not know to which machine to send incoming AXIP unless explicitly configured.

Configuring such a router for AXIP usually involves specifying a protocol number (93), and the LAN IP address of a machine to which it should be routed, i.e. Xrouter's LAN IP address.

You are advised that not all front end routers can be configured to route incoming AXIP as it is not a commercially recognised protocol. Some routers will only allow TCP and UDP port redirection, with no provision for any other protocol. Windows 98 ICS will not route incoming AXIP, but there are (at a price) non-Microsoft programs which will do so.

If you are lumbered with such a router, you may need to consider AXUDP instead - see later.

AXUDP TUNNELLING

AXUDP is AX25 "encapsulated" within UDP datagrams. This enables AX25 systems to communicate with each other via TCP/IP networks such as the Internet. It is slightly less efficient than AXIP, because there is an extra 8 byte UDP header between the IP and AX25 headers in the frame, but the difference is not significant. The major advantage of AXUDP over AXIP is that it can usually be handled by front end routers such as Windows 98 ICS without much problem.

The procedure for setting up an AXUDP wormhole is very similar to that for setting up AXIP, and you should first read the AXIP section elsewhere in this manual.

The differences are as follows:

1) Instead of (or in addition to) the AXIP interface, add an AXUDP interface to XROUTER.CFG as follows:

```
INTERFACE=9
   TYPE=AXUDP
   MTU=256
ENDINTERFACE
```

As with AXIP, this interface can support an unlimited number of AXUDP ports. Multiple AXUDP interfaces are allowed.

2) For each AXUDP partner, add an AXUDP port similar to this:

```
PORT=8
   ID=AXUDP link with VK1UDP
   INTERFACENUM=9
   IPADDRESS=44.131.91.245
   UDPLOCAL=93
   IPLINK=27.69.88.73
   UDPREMOTE=93
ENDPORT
```

See the AXIP section for a discussion of the required parameters.

UDPLOCAL and UDPREMOTE are the UDP "service port" numbers for each end of the link, and if omitted they default to 93. Don't confuse these with *protocol number* 93, which is AXIP.

3) If you are using a front end router, xDSL modem etc. you will need to "open" UDP port 93 (or your UDPLOCAL number if it differs from the default), to allow incoming UDP frames to be directed to Xrouter's IP address. For Windows 98, a very useful program called ICSCFG can be obtained from the internet for this purpose.

Some routers don't have the facility to open specific UDP ports, but at the very least should allow you to direct all UDP traffic to a specified IP address.

IPUDP TUNELLING

IPUDP is IP encapsulated in UDP/IP. It allows amateur IP to be transported across other TCP/IP networks such as the Internet, to form "Virtual Private Networks" (VPN's).

Whereas IPIP (commonly called Encap) transports the private (e.g. amateur) IP directly inside the payload portion of a public (e.g. Internet) IP datagram, IPUDP transports the private datagram in the payload portion of a UDP frame, which is itself transported as payload in a public IP datagram. This equires 8 bytes more overhead than IPIP, but it is far more flexible.

IPIP is a "portless" protocol, and it is therefore difficult (in come cases impossible) to get it to pass through some types of NAT / PAT router which rely on translating TCP and UDP service port numbers in order to share a public IP address among several LAN hosts.

IPUDP overcomes this limitation because it transports the data using a well known protocol (UDP) which NAT / PAT routers can understand, thus it can get through where IPIP cannot. For example, it is easy to configure Windows 98 Internet Connection Sharing (ICS) to route incoming IPUDP to a specific machine on the LAN, but it is not possible to do this with IPIP.

IPUDP currently uses UDP service port 94, simply because I found it easy to remember, and there seemed to be no other significant protocols using this number. As originator of this protocol, if difficulties are encountered with port 94, please tell me (G8PZT).

In order to establish an IPUDP link, you and your link partner must first set up and test IP routing between your public IP addresses. If you are using a "front end" router between Xrouter and the Internet, you must "open" UDP port 94 to direct incoming traffic to Xrouter. Finally you must add an extra IP route entry as follows:

IP ROUTE ADD 44.131.91.0/24 62.31.206.176 5 u

(Omit the "IP" if the entry is used in IPROUTE.SYS)

The first IP address is the amateur IP address, or range thereof, to be routed via this IPUDP link. If you don't fully understand this format, see the manual sections detailing IPROUTE.SYS and the IP ROUTE ADD command.

The second address is the IP address or hostname of the link partner to whom the first address(es) will be routed. It is more efficient to use an IP address if possible. Your system MUST be told how to reach that partner, so you will need an existing routing entry which applies to it.

The "5" is the number of the port on which the datagrams will be transmitted. This port obviously needs a public IP address, or one which will be translated to a public address by a front end router.

Mode "u" signifies IPUDP encapsulation.

POINT TO POINT PROTOCOL

Point to Point Protocol (PPP) is a protocol suite intended for use over simple links which transport packets between two peers, such as an RS232 link.

Unlike SLIP, it includes facilities for link control (e.g. management of a dial up link), user authentication, negotiation of IP addresses, and the multiplexing of several protocols across a common link.

It is designed to be self configuring. Most of the options have standard default values, and peers negotiate anything which differs from the default.

Xrouter currently implements the most common PPP sub-protocols:

- Link Control Protocol (LCP)
- Password Authentication Protocol (PAP)
- IP Control Protocol (IPCP)

Most sysops will have no need to remember these protocol names and acronyms. However, for those who wish to experiment, each protocol has its own set of commands.

PPP has largely replaced SLIP for dial-up Internet access, and the interconnection of peers via RS232.

If you want to know more about the internal workings of PPP, it is specified in RFC1661.

Configuring Xrouter for PPP

There are three ways in which PPP may be used:

- Wired links.
- Dial-in.
- Dial-out.

The first case requires an interface to be configured to use PPP, but the other two cases temporarily establish a PPP link on a MODEM interface. Each case is described in more detail below:

Using PPP on Wire Links

In XROUTER.CFG, define an interface with TYPE=ASYNC and PROTOCOL=PPP. For example:

```
INTERFACE=1
        TYPE=ASYNC
        COM=1                   ; 1-4 or specify INTNUM/IOADDR
        SPEED=19200             ; Adjust as necessary
        PROTOCOL=PPP
        MTU=1600
ENDINTERFACE
```

Now "bind" a port to that interface thus:

```
PORT=1
    ID=PPP link to Win98    ; Text to identify port
    INTERFACENUM=1          ; Bind to PPP interface
    IPADDRESS=192.168.0.4   ; Optional (see below)
ENDPORT
```

The use of IPADDRESS is optional. If the keyword is omitted, Xrouter's "global" IP address will be used on that port. If the address is specified as 0.0.0.0, it signifies that a "dynamic" IP address should be obtained from the peer, otherwise the specified address will be used.

The port is now ready for PPP using the standard PPP defaults, but if it requires any further configuration you may include the relevant PPP configuration commands in the file BOOTCMDS.SYS, which is read by Xrouter after it has finished executing XROUTER.CFG.

PPP Configuration Commands

All PPP configuration commands start with "PPP", and are described in the sysop command reference section. When used from the command line, or with a BOOTCMDS.SYS file, the first argument must be a port number. However, PPP commands used within PPPHOST and dialler scripts *do not* include a port number, because Xrouter knows which port is executing the script.

e.g. at the command line: PPP 3 IDLE 300 in a dialler script: PPP IDLE 300

IP Routing with PPP

When a PPP link opens, a route to the peer is automatically added to Xrouter's IP routing table. If the peer indicates that it knows the addresses of a primary and secondary DNS, those are added to Xrouter's DNS list.

PPP Logging

PPP system activity can be recorded in file PPPLOG.TXT for diagnostic purposes. The amount of detail is controlled by number specified in the "PPP <port> LOG n" command as follows:

0 No logging
1 PPP start / stop / timeout events
2 As 1, plus layer up / down events
3 As 2, plus layer start / stop events
4 as 3, plus option accept / reject events
5 as 4, plus hexdump of configuration packets

You are advised against using the higher levels other than on a temporary basis because they can create very large logfiles. On a slow machine, the disk writes might take so long that configuration timeouts could occur, with consequent retries and failure.

If logging is in use, you should regularly remove the logfiles to prevent them growing too large.

DHCP

DHCP stands for Dynamic Host Configuration Protocol. This is a client-server based protocol which allows clients on a TCP/IP network to obtain their configuration parameters from a server. The protocol supports the transfer of a wide range of configuration parameters, such as the client's IP address, netmask, DNS and gateway addresses, plus TCP/IP parameters such as MSS, but is most commonly used to allocate dynamic IP addresses to clients.

IP addresses are "leased" to clients for a period of time, after which the client must renew the lease. Servers generally attempt to re-assign the same IP address to a given client.

DHCP in Xrouter

Xrouter includes a DHCP client, and a DHCP server will be included in future. The full range of configuration options is not supported, since in most Xrouter application scenarios they are not required. The options currently supported are client's IP address and lease time, DNS and gateway IP addresses.

The DHCP client is used only on Ethernet interfaces. Lease negotiation and renewal are completely automatic, and the sysop need not be concerned with the process.

Do you need DHCP?

If you wish to connect Xrouter to an ISP via a cable modem e.g. to use it as an Internet Connection Sharing router, you will probably need DHCP if your ISP uses dynamic IP addressing. If your ISP assigns you a static IP address you won't need DHCP.

You will not need DHCP if your connection to the ISP is via dial-up PPP, because dynamic IP addresses are assigned as part of the PPP log-in process. You will not need DHCP for normal Ethernet or amateur radio operations.

Enabling DHCP

In XROUTER.CFG, put "DHCP=1" in the appropriate port definition block. There is no need to specify a port IPADDRESS because one will be assigned by the DHCP server. If however, a port IPADDRESS is specified (or it is not specified but a global IP address is specified), that address will be used for non-DHCP traffic until DHCP succeeds in leasing a (possibly different) address. If the global IPADDRESS is 0.0.0.0 or not specified, it will be assigned by the first client which obtains a lease.

To disable DHCP, put "DHCP=0" in the PORT definition block, or simply omit the keyword altogether.

The DHCP command displays DHCP status information.

TCP/IP Access Control

This section is only of concern if your system is directly connected to the Internet.

Some of the users who access your system via the internet may be genuine Radio Amateurs, who may legitimately downlink on radio frequencies, while other users may not. And different countries may have different rules governing the interconnection of radio and non-radio networks.

Therefore some form of configurable access control is required, and this is provided by the entries in ACCESS.SYS, which specifies the login requirements appropriate to the caller's IP address.

If ACCESS.SYS is not present, the default action is for logins to require a valid callsign only.

The entries in ACCESS.SYS allow various levels of access control, e.g. username only, valid amateur radio callsign only, username plus password, and valid callsign plus password.

If an entry in ACCESS.SYS specifies that a login password is required, it should be located in file USERPASS.SYS.

Failure to meet the access requirements results in immediate disconnection of the caller.

Sysop access using the "@" command, RLOGIN (tcp port 513) and FTP are all controlled by entries in PASSWORD.SYS. The "@" command, which is normally performed on publicly-visible radio links, uses the password to send a grid of numbers, from which the caller must select one line and send the matching characters from the password. RLOGIN must only be used on secure wired networks because the caller must send the password itself. FTP uses the password grid method, but can be configured to use use plain password (secure wired links only) if SYSOP=1 has been included in the appropriate PORT configuration.

Access to the APRS server is normally controlled by the "Validation number" which the user obtains from the author of his APRS client program upon registration. However, if the user has not registered his client, he may be granted access to the server by including his callsign and a password in USERPASS.SYS.

For further information you are referred to the sections detailing the ACCESS.SYS, PASSWORD.SYS and USERPASS.SYS files.

ROUTING INFORMATION PROTOCOL

Routing Information Protocol (RIP) allows routers to learn about each other's routing, similar to the way Netrom exchanges nodes broadcasts.

There are various versions of RIP, and Xrouter implements only RIP98, which was developed by G8BPQ specifically for radio-based routers. If there is a need for older versions of RIP, I may include them at a later date.

RIP98 works by sending its routing table to nominated peers at regular intervals, and accepting routing information from peers. The routes learned by this means are added as temporary entries to the IP routing table. These entries have a finite lifetime, and if not updated for a while they will drop out of the routing table.

Configuring Xrouter to use RIP98

The following configuration commands are available:

```
RIP ACCEPT      Remove a peer from the refuse list.
RIP ADD         Add a peer to the broadcast list. (*)
RIP DROP        Remove a peer from the broadcast list.
RIP LEARN       Allow / disallow route learning. (*)
RIP REFUSE      Ignore broadcasts from a peer. (*)
RIP STATUS      Show status of RIP.
RIP TIMEOUT     Specify lifetime of learned routes. (*)
```

Commands marked (*) may be used in BOOTCMDS.SYS or IPROUTE.SYS to configure the system automatically.

By default, RIP98 route learning is OFF, so you need to include at least a "RIP LEARN ON" command, to enable your system to learn routes from others. Your neighbours will also need to add your IP address to their RIP broadcast lists.

If you wish to inform your neighbours of your existence and your routing capabilities, you need some RIP ADD commands, one for each neighbour to who you wish to send RIP updates.

If you have LEARN enabled, and there is a neighbour who is sending unwanted RIP updates to you, you may ignore them using RIP REFUSE.

The RIP commands are explained in more detail in the sysop command reference section later in this manual.

NETWORK ADDRESS TRANSLATION

Section 1 - About Network Address Translation

In order for a computer (sometimes called a "host") to communicate with other computers using TCP/IP it must have an IP address. This is a 32 bit number unique to that machine, and it is composed of four 8 bit fields which hosts use to make routing decisions.

Theoretically, there could be 2**32 unique addresses (just over 4000 million), but in practice the figure is a lot lower because some of the addresses are not used. This is due to the way addresses are separated into classes and assigned to networks.

For example, the "class A" address series 44.x.x.x is assigned to the amateur radio network and contains 2**24 (16.7 million) addresses. Within this series, the series 44.131.x.x is assigned to the UK, and contains 2**16 (65536) addresses.

Taking this further, the series 44.131.91.x is assigned to north Worcestershire and contains 2*8 (256) addresses, yet there are only a couple of IP stations in north Worcestershire. So the remaining IP addresses are "wasted". A similar wastage occurs in every county in the UK and every country in the world.

Although there are only a couple of registered IP stations in the 44.131.91.x series, each station may be running more than one computer. For instance, at any one time, my station may be running between 2 and 6 machines, linked via Ethernet.

Some of these machines are nothing to do with amateur radio and therefore are not entitled to a 44.x.x.x series IP address. In addition, registering IP addresses is a lengthy procedure which is impractical for dealing with a home network in which computers are rearranged frequently.

In order to cater for these private and experimental networks, a number of address ranges were set aside for "unregistered" use. One such series is 192.168.0.x which can cater for up to 256 hosts. Any one of these addresses may be used in millions of private networks at once, thus alleviating the shortage of IP addresses.

A problem arises if an "unregistered" host on a private network needs to communicate with a host on the "registered" network. Packets from the unregistered (local) host can be routed normally to the registered (global) host, but since the local host is unregistered, and the same address is used many places at once, the network has no way of knowing how to route the replies.

This is where Network Address Translation (NAT) comes to the rescue. NAT operates on each datagram, substituting one IP address with another.

For instance a local address such as 192.168.0.2 can be translated to a globally recognised one, such as 44.131.91.2, allowing a host on a LAN to access, and be visible to, the global network.

Consider this example:

Registered IP	Unregistered IP
44.131.91.2	192.168.0.2
44.131.91.3	192.168.0.16

Packets arriving from the global network addressed to 44.131.91.2 are re-addressed to 192.168.0.2 and routed to the appropriate machine on the local network, while those addressed to 44.131.91.3 are re-addressed to 192.168.0.16 and routed to that machine.

In the reverse direction, packets originating from host 192.168.0.2 on the LAN, destined for the global network, have their source address changed to the globally recognised 44.131.91.2, and 192.168.0.16 is translated to 44.131.91.3.

This one to one mapping of one address to another is called STATIC NAT, (also known as RFC1631 NAT) and is implemented in Xrouter.

Port Address Translation

With Static NAT, if you have more than one local host which need to be visible on the global network, you must own more than one registered IP address. A refinement of NAT, called Port Address Translation (PAT) allows one registered IP address to be shared by many unregistered hosts, by manipulating the "service port" numbers in TCP and UDP packet headers.

For example, consider the following mappings:

Global address	Port	Local address	Port
44.131.91.2	777	192.168.0.1	1024
44.131.91.2	778	192.168.0.2	1024
44.131.91.2	779	192.168.0.5	1631

TCP segments originating from port 1024 on local host 192.168.0.1, and bound for the global network, are re-addressed to look like they originated from port 777 on globally-recognised host 44.131.91.3. Likewise, segments from port 1631 on local host 192.168.0.5 are made to look like they originated from port 779 on global host 44.131.91.2. The translations are reversed for incoming segments.

PAT can be static or dynamic. With static PAT, the sysop must set up the translation table as in the example above. Dynamic PAT builds the table automatically, and the entries are removed when they're finished with. This makes the two systems behave very differently, as discussed below.

Static PAT

This form of PAT consistently translates a global address/port pair to an equivalent local address/port pair and vice versa. Since the mappings are permanent and predictable, the designated ports on the local hosts are visible to the global network. This is ideal for making local servers (e.g. SMTP, HTTP, POP3) globally visible.

Static PAT can be used to multiplex several hosts onto one IP address, or it can simply be used to manipulate port numbers, for example to create "virtual hosts" as shown below:

Global address	Port	Local Address	Port
44.131.91.2	80	192.168.0.1	8000
44.131.91.3	80	192.168.0.1	8001
44.131.91.4	80	192.168.0.1	8002

The outside world sees 3 web servers (port 80 is the HTTP port), with the IP addresses 44.131.91.2, 91.3 and 91.4, yet in reality there are 3 separate web server processes (ports 8001, 8002, 8003) running on one host.

There is a problem with static PAT however. TCP/IP server processes use predictable port numbers. For instance, HTTP servers usually use port 80 (although they can often be re-configured for a different port), which means that incoming TCP segments addressed to port 80 will always go to the web server, and the server will reply using the same port as the source. TCP/IP *clients* on the other hand tend to use unpredictable port numbers. For example, the first Telnet client session on a freshly booted Windows98 system will use port 1024, the next session will use port 1025 and so on. With static PAT, set up to translate port 1024 to an outside world address/port pair, the first Telnet session will succeed, but the second and subsequent ones will fail.

Therefore as a general rule, static PAT is useful for making local SERVERS globally visible, but not for accessing the global network using local CLIENTS. It's a one way street. This effect can be exploited to prevent LAN users from accessing the global network.

Dynamic PAT

Dynamic PAT is commonly used to multiplex several hosts onto one IP address a process called "Overloading", and it tends to act as a one way street in the opposite direction to static PAT.

Translation entries are created when a local host originates a connection to the global network, and are removed when that connection is closed. Thus dynamic PAT cannot generally be used to make local servers globally visible, but outgoing connections can be made without hindrance.

This creates a simple "firewall", preventing your local hosts from attacks from the global network.

Both static and overloaded PAT are implemented in Xrouter.

Limitations of NAT / PAT

Unfortunately, NAT and PAT are not completely transparent to the user, and there are certain situations which they cannot handle.

IP, ICMP, TCP and UDP packet headers each contain a "checksum", and the IP addresses and service port numbers are included in the calculation. This means that any change to the address or port numbers requires all the checksums to be recalculated and re-inserted.

In most cases it is sufficient to manipulate the packet headers alone, but some protocols convey IP address and TCP port numbers in the data portion of the packet, and these present more of a problem.

ICMP error reports return part of the faulty datagram, and that part must be re-translated and the checksums recalculated if the process is to remain completely invisible to the user.

Certain FTP transactions convey an IP address and port number, expressed in ASCII, in the data. NAT must look for these and change them. Besides being a non-trivial operation, the problem with this is that the translated addresses may occupy a different amount of space when expressed in ASCII, so NAT must build a new packet, and must adjust the TCP sequence numbers on every subsequent packet.

There are other applications which similarly embed addressing information in the data portion of the packet, and strictly speaking, the TCP/IP layers must remain unaware of this information as it is of a higher layer. In this respect NAT breaks normal layering rules.

PAT achieves multiplexing by translating service port numbers, but some protocols do not use service port numbers at all, so these can not pass through PAT. For example, ICMP, IPIP and AXIP can only pass through static NAT, whereas AXUDP and IPUDP can pass through PAT.

The following protocols and traffic will pass through NAT:

```
Protocol  Supported Traffic / Applications
-----------------------------------------------
RIP ?
ICMP       Ping, Traceroute etc.
AXIP       Ax25 tunnelling
IPIP       IP tunnelling.
UDP        AXUDP, IPUDP, DNS, TFTP
TCP        Telnet, HTTP, SMTP, NNTP, POP, Finger, Rlogin
           Plus any other traffic which does not use
           TCP/IP addresses in the application data.
```

Only the following protocols and traffic will pass through PAT:

```
Protocol  Supported Traffic / Applications
-----------------------------------------------
UDP       AXUDP, IPUDP, DNS, TFTP
TCP       Telnet, HTTP, SMTP, NNTP, POP, Finger, Rlogin
          Plus any other traffic which does not use
          TCP/IP addresses in the application data.
```

Section 2 - Configuring NAT / PAT on Xrouter

The "NAT" commands are used for configuring both NAT and PAT on the fly. They can also be embedded in the IPROUTE.SYS or BOOTCMDS.SYS files to configure the system at startup.

The following commands are available:

NAT ADD Adds entries to the translation table.
NAT DROP Deletes entries from the translation table.
NAT LIST Displays the translation table.

Syntax is as follows:

NAT ADD STATIC <localaddr>[:port] <globaladdr>[:port] [tcp | udp]
NAT ADD OVERLOAD <localaddr> <globaladdr> <subnet_mask>

The first form adds static NAT and PAT entries, and the second form is used only for adding overloaded dynamic PAT entries.

In each case <localaddr> represents the "private" or "unregistered" IP address of a host on the stub network, and <globaladdr> represents a globally recognised or "registered" address.

In the STATIC case, if port numbers are specified, TCP and UDP traffic matching the specified IP addresses will be translated ONLY if it also matches the specified ports. If ports are not specified, all traffic is translated. If the optional protocol is specified, only traffic of that protocol will be translated by that entry.

The OVERLOAD case does not accept port numbers, and it requires a subnet mask to be specified. The mask is used in combination with the local address to form a range of addresses which will be accepted for

translation. For example, if the local address is 192.168.0.0 and the netmask is 255.255.255.240, addresses 192.168.0.0 to 192.168.0.15 inclusive will be translated.

NAT DROP <local>[:port] [tcp | udp]

Simple entries, i.e. those in which the protocol shows "ALL" and the port numbers are zero, can be removed by the form:"NAT DROP <localaddr>" e.g. "NAT DROP 192.168.0.2".

If the translation table entry includes port numbers, the form:

"NAT DROP <local_address>[:port]" is required, e.g.
"NAT DROP 192.168.0.2:23".

If the translation entry is protocol-specific, the protocol must be specified when removing the entry, e.g.:

"NAT DROP 192.168.0.2 TCP".

NAT LIST

This command currently requires no arguments.

Order of Entries in NAT table

The order of entries within the translation table is important, because Xrouter will translate upon the first matching entry. This gives you maximum flexibility to cater for your particular needs. As a general rule, port-specific and protocol-specific entries should be declared before non-specific "catch-all" entries. Overload entries can be declared anywhere, providing they don't inadvertently "hide" a static translation for the same address.

Consider this example:

```
NAT ADD STATIC    192.168.0.2:87  44.131.91.2:23  TCP
NAT ADD STATIC    192.168.0.2     44.131.91.2
NAT ADD OVERLOAD  192.168.0.0     44.131.91.3     255.255.255.240
```

In the above example, TCP frames originating from port 87 on local host 192.168.0.2 will be translated by the first entry, to look like they originated from port 23 on global host 44.131.91.2.

UDP and all other TCP frames from that host will be translated by the second entry, which leaves the port numbers alone. This entry also translates "portless" protocols such as AXIP, ICMP, IPIP etc.

The third entry translates any TCP or UDP frame originating from local hosts 192.168.0.0 to 192.168.0.15, excluding 192.168.0.2, to look as if it originated at 44.131.91.3.

If the second entry had been placed before the first, it would never have been executed, because the non-specific static entry would have intercepted *every" frame from 192.168.0.2. If the overload entry had configured.

Network Address Translation is applied *before* the packet is routed. This means that for "outbound" packets, i.e. those originating on the "private" subnet, routing to the "public" net should be defined. For "inbound" packets, i.e. those originating on the public net, addressed to one of the global NAT addresses, the should be a routing entry which will route the translated address to the *private* subnet. This is best illustrated with an example:

```
In IPROUTE.SYS....

ROUTE ADD 44.0.0.0/8          44.131.90.6     11      d
ROUTE ADD 192.168.0.1/32      *               14      d

NAT ADD STATIC 192.168.0.1    44.131.91.3
```

The first entry routes all outbound 44-net packets to peer 44.131.90.6 on port 11.

The second entry routes packets addressed to 192.168.0.1 onto port 14 which is the Ethernet LAN.

The third entry translates destination address 44.131.91.3 on incoming packets into 192.168.0.1 before sending the packet on the LAN as dictated by the second entry.

Internet Connection Sharing

If one of Xrouter's ports is connected to the Internet (e.g. by dialup or cable modem), you may use dynamic PAT to allow other computers on a LAN to share the connection. Assuming the computers on your LAN use addresses in the range 192.168.0.1 to 192.168.0.255 (this is the range normally used by Windows), the NAT entry should look like this:

NAT ADD OVERLOAD 192.168.0.0 0.0.0.0 255.255.255.0

Note the special 0.0.0.0 entry, which tells Xrouter to translate the source address on outgoing frames to the address of the port on which they are sent. This is required if your internet connection uses a dynamic IP address, but if your address is static you may insert it in place of the 0.0.0.0.

If you don't have a DNS server on your LAN, you will need to set up the LAN computers to use Xrouter as their DNS. You will also need to tell Xrouter the address of at least one external DNS, either by including a "DNS=<ipaddr>" statement in xrouter.cfg, or by using a "DNS ADD <ipaddr>" command. Xrouter will then act as proxy DNS for the computers on the LAN.

Summary

a) Use NAT ADD... in IPROUTE.SYS to define translations.
b) Add IP routing for the *global* NAT addresses.
c) Use NAT... commands to display / adjust NAT on the fly.
d) Report problems to me.

References:

RFC791 - Internet Protocol
RFC792 - Internet Control Message protocol
RFC793 - Transmission Control Protocol
RFC1631 - The IP Network Address Translator
RFC1700 - Assigned Numbers
RFC1918 - Address Allocation for Private Intranets

Section 3 – Manual NAT:

NAT(1) XROUTER REFERENCE MANUAL 23/7/02

COMMAND

NAT -- Network Address Translation commands.

SYNOPSIS

NAT ADD STATIC <local>[:port] <global>[:port] [tcp | udp]
NAT ADD OVERLOAD <local> <global> <subnet_mask>
NAT DROP <local>[:port] [tcp | udp]
NAT LIST

DESCRIPTION

The NAT commands controls Network Address Translation, i.e. the process whereby the IP addresses contained in datagrams are manipulated to allow hosts on one network to communicate with hosts on a different network.

For example, hosts on a private intranet using unregistered 192.168.0.x addresses cannot communicate with hosts on the wider Internet because no-one would know where to route the return datagrams. NAT basically translates the unregistered addresses into registered ones and vice versa.

PAT (Port Address Translation) manipulates TCP and UDP service port numbers, for example to allow several hosts to share one IP address. The NAT commands are also used to configure PAT.

The arguments for NAT commands are as follows:

<local>	**Local private (unregistered) IP address.**
<global>	**Globally recognised IP address.**
<port>	**TCP or UDP service port number.**
<subnet_mask>	**Bit pattern used for matching addresses. e.g. 255.255.255.0**

NAT ADD adds an entry to the NAT table. There are two forms: STATIC and OVERLOAD. STATIC is used to add static NAT and PAT entries, i.e. those where there is a one-to-one mapping between private and public IP addresses. OVERLOAD is used only for dynamic PAT, where several hosts share one public IP address.

NAT DROP removes an entry from the NAT table.

NAT LIST lists the NAT table entries.

EXAMPLES

NAT ADD STATIC 192.168.0.2:87 44.131.91.2:23 tcp
NAT ADD OVERLOAD 192.168.0.0 44.131.91.3 255.255.255.240
NAT DROP 192.168.0.5:23 tcp

AVAILABILITY

The NAT commands are only available to sysops.

NOTE

The NAT ADD command can also be used in IPROUTE.SYS or BOOTCMDS.SYS.

TCP/IP STEALTH MODE

The experimental command IP QUIET [n] controls whether or not ICMP error messages are generated. The command may be used at the command prompt, in BOOTCMDS.SYS, or (without the "IP") in IROUTE.SYS. I am not convinced it is a good idea, and I may remove the command or modify its action as experience is gained.

Hackers use automated software to "probe" the network, looking for unprotected TCP or UDP services and "back doors" such as NetBios. If found, they will usually exploit known bugs, such as buffer overflow problems, to crash the machine or gain access to sensitive areas.

Xrouter has only a handful of standard TCP / UDP services, none of which pose much of a security risk if configured correctly, so the probes are more of a bandwidth-wasting nuisance than a real threat.

Issuing the command "IP QUIET n", where n is a number between 1 and 255 puts Xrouter into "stealth mode", the level of which depends on the number n, which is the sum of the following values:

1 Suppress ICMP echo replies.
2 Suppress ICMP "Unknown Protocol" messages
4 Suppress TCP resets
8 Suppress all other ICMP error messages.

A value of 0 disables stealth mode and lets TCP/IP operate normally.

Suppressing ICMP messages, may reduce the bandwidth wasted and slow up the rate of probing. But it won't confer any extra security, and will certainly have a detrimental effect on normal TCP/IP operations.

ICMP error messages are an integral part of TCP/IP, and are used to inform a sender of network problems such as un-routable frames, unsupported protocols, processing errors etc. They are also used for diagnostic purposes, by applications such as "Ping" and "TraceRoute".

Using stealth mode therefore prevents the use of diagnostics and the detection of network problems, and may under some conditions make everything run more slowly, or fail completely.

If you suppress ICMP echo replies, Xrouter will not respond to "pings". This may be temporarily useful if you are being attacked with echo requests, but is anti-social because you would also be denying others the use of a valuable network diagnostic tool. It will not *prevent* a pingstorm attack, but it will halve the traffic by suppressing the replies.

If you suppress ICMP "unknown protocol" messages, it will reduce the bandwidth wasted by protocol scans, i.e. those in which the protocol number is incremented with each probe.

"Suppressing TCP resets" causes the TCP layer to ignore connect
requests aimed at non-existent TCP services, instead of refusing them.This may be useful in slowing up the action of "port scanners".

I do not recommend the "Suppress all ICMP error messages" option.It is provided for experimentation only.

NETROM INTERLINKS

Unlike BPQ, which closes AX25 level 2 links with neighbouring nodes after a period of inactivity, Xrouter attempts to maintain a permanently open link, and you may find this disconcerting.

There are a number of reasons for this strategy. Most importantly, it is able to detect a common and disruptive problem, namely the "one way link". These occur when a transmitter, receiver, antenna, or TNC malfunctions, leaving one peer hearing the other but not vice versa.

When a one-way link occurs, one peer can hear the other's nodes broadcasts, but no connected-mode traffic can be transferred. Thus one peer thinks it has a route via its peer, but that route is unusable, and all nodes advertised by the peer are unreachable.

Xrouter solves this problem by detecting when the link cannot accept traffic, and marking all nodes advertised by the peer as unusable, so that alternative routes are used.

Another good reason for keeping links open is to enable continuous measurement of route quality and round trip time, used in making routing decisions.

A further reason is to remove the L2 link setup time when L3 frames arrive infrequently.

The "R" commands will show a ">" in the left-most column if the interlink is fully open, a "~" if it is opening or closing, or a blank if the link can't be opened. This provides a handy way of detecting link problems which might otherwise go un-noticed.

If port quality and nodesinterval are both non-zero, as soon as Xrouter hears a nodes broadcast from a previously unknown neighbour, it will attempt to connect to it. If the path is marginal, the attempts may fail, and you can prevent it from re-trying by locking in the neighbour with a zero quality.

TIME DOMAIN ROUTING

Conventional NetRom makes routing decisions based on a fairly arbitrary metric, i.e. the "route quality", which is assigned by sysops, and disseminated in nodes broadcasts. There is no standard for assigning quality, and not only will each sysop have a different notion of the quality of his links relative to others, but he will probably wrongly assign the qualities of those links relative to each other. This leads to inconsistency and distorted routing.

Xrouter includes two systems which attempt to alleviate this problem, namely automatic Route Quality Measurement" (Autoqual) and "Time Domain Routing" (TDR). Both systems rely on a slightly different understanding of the "goodness" of a link.

In the better-managed parts of the NetRom network, route qualities tend to be assigned according to the baud rate of the link, with adjustments for retry rates, duplex / simplex and shared channels (in the worst-managed parts, route qualities are simply assigned to 192 regardless of how good or bad the link is). The quality is fixed at a compromise value.

But the actual "goodness" of a link may continually change with atmospheric conditions, data throughput, other channel activity, QRM etc. At certain times of day for example, it may be better to use an alternative link.

A more accurate notion of "goodness" is simply the "Round Trip Time" (RTT) for the link, i.e. the time taken to send a packet and get a reply. After all, this is what *really* matters to users. A link which responds quickly (i.e. with a low RTT) is perceived by users to be better than a link which responds slowly. The RTT will track changes in retry rate, channel loading etc.

The RTT can be easily and consistently measured by software on a continuous basis, thus the "quality" of the link is accurately known at all times, and all routers of the same type will give comparable values independently of the sysop's notions of quality.

Xrouter continually measures RTT and uses it to calculate route quality. This quality is displayed by the "R Q" command, and can either be used as a guide to allow the sysop to fix the RQ at a sensible value, or Xrouter can use it dynamically, by setting the route to use Autoqual. (Autoqual is engaged by setting an RQ between 256 and 511). The RTT to quality conversion is tailored to the British notion of quality, which gives somewhat lower but more meaningful qualities.

Autoqual is merely a simple tool to aid sysops, and the route qualities are still under sysop control, and thus open to distortion. However, by simply broadcasting RTT values instead of qualities, the influence of the sysop is removed, and a network based on indisputable *times* rather than arbitrary *quality* can be created. This is a network which has its routing metric in the time domain, hence the

name "Time Domain Routing". It may to some extent overlap the quality-domain network, but the boundaries may be different, and the two schemes are not compatible.

Xrouter implements both time-domain and quality-domain routing schemes, and will consider information from both domains when making routing decisions. The same node table is used for both schemes, since only the metric is different. In some cases a node may have both quality and time metrics.

As sysop, you have several tools at your disposal for determining the size and balance of the two domains. For the quality domain you have, QUALITY which defines the "goodness" of the links to your neighbours and the "de-rating" of the qualities which they send to you, MINQUAL, which determines which nodes get into the table and which are excluded, MINTXQUAL, which determines how much information you send to your neighbour nodes, and MAXNODES which determines the maximum number of nodes visible to you.

For the time domain, you have MAXTT, which defines the furthest node in "Trip Time" (i.e. RTT/2) terms, MAXHOPS which defines the furthest node as a function of the number of intervening nodes, and MAXNODES as above.

You may adjust these parameters to favour one domain over the other, to exclude one domain altogether, or to strike a balance between the number of nodes which exist solely in one domain or the other. For example, setting MAXTT or MAXHOPS to 0 will exclude all time-domain information, and Xrouter will behave as a pure NetRom router. Or you may set MINQUAL to 255, excluding all quality-domain information (e.g. if there is a nearby node distorting the netrom qualities), providing of course you have neighbours which are cabable of time-domain routing. Or you may wish to limit the visibility of a subnet from one port (e.g. to a foreign network) by setting a low MAXHOPS or MAXTT on that port only. This gives you control which was not possible by quality alone.

Xrouter currently (but see compatibility issues) uses the INP3 protocol to disseminate time-domain routing information. Unlike NetRom, which uses unconnected-mode "broadcasts" to all neighbours simultaneously, INP3 sends data to each neighbour individually, using connected-mode. Whilst it is usual to make NetRom nodes broadcasts at 1 hour intervals, INP3 updates are sent every 10 minutes, with additional updates whenever changes occur.

The time-domain network thus responds much more rapidly to changes than NetRom, but if the network is unreliable (i.e. frequent outages and variable RTT's), a lot of updates are generated. Although INP3 updates are more compact than NetRom nodes broadcasts, some sysops may feel that the amount of routing information is too much for a poor quality RF link. If so, you may disable INP3 completely by setting the route MAXTT to 0, or you can agree a low MAXTT with your neighbour node, which will reduce the volume of data.

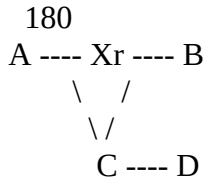
You may notice that nodes which are learned solely via INP3 are all stored with a Netrom quality of 1, which is probably below MINQUAL. This is deliberate, because these nodes have no presence in the quality domain. They should have a quality of 0, and I will probably correct that in a later version.

Compatibility issues:

Although this is a router manual, not a treatise on advanced networking, I must stress that time-domain and quality-domain routing information are incompatible, and information from one domain must *never* be "translated" to the other. Xrouter measures, uses and disseminates both time and quality, but always keeps them separate.

Unfortunately, *some* software (which I shall call "Brand-X") does translate information between the domains, and you should be aware that this may cause you problems if they are within the horizon of either domain.

For example, in the diagram below, imagine that Xrouter (Xr) measures the trip time (TT) to one of its neighbour nodes (A) as 1.5 seconds, and the Netrom quality of that route to 180.



Xrouter then broadcast both pieces of information to neighbour node (B) which is using "Brand-X" software. That software will ignore the 180 and will instead manufacture a quality of 253 from the trip time. By itself this isn't a problem, until the "Brand-X" software broadcasts it to node (C) which is Netrom-only. Now the Netrom node thinks it has a higher quality to (A) via (B) than via (Xr). What's worse, it will tell (Xr) that it knows a better route to (A), and the network will decend into chaos.

A fully interconnected network is very robust, yet the "Brand-X" sysops seem remarkably reluctant to implement links which result in network "triangles". I wonder why? :-)

Another problem occurs when the "Brand-X" software translates a Netrom quality, which is a potentially unreliable piece of information, into a trip time and hop count. A node which exists only in the untrustworthy quality domain, and may have been beyond our horizon in that domain, now exists in the more trusted time domain where it can bypass the Netrom de-rating process. The information could subsequently pass back into the Netrom network with a higher "quality" than it would have when received via a more direct pure netrom route.

Early versions of Xrouter used a proprietary protocol to exchange RTT, hops, IP routing, position and other information between themselves. The INP3 protocol used by other node software was remarkably similar, so I later decided to adapt my code for INP3 compatibility, believing that it would be a good idea to interconnect my time domain scheme with theirs. I now realise this was a big mistake, and I firmly believe that, unless the "Brand-X" software authors correct the errors, Xrouter and the Netrom network should not be connected to any other network which includes "Brand-X" nodes.

NETROM QUALITY MANIPULATION

This section describes a feature which allows you to reduce the quality of "foreign" nodes on your system, or to exclude certain nodes or groups of nodes altogether. I do not necessarily condone its use.

With ever-increasing connectivity via the Internet, the NetRom network is crossing traditional boundaries, and this is causing problems. Because Internet qualities are much higher than radio qualities, there are far more nodes in the tables.

There is a limit to the size of tables, and more importantly there is a limit to the number of nodes that can be broadcast on a low bandwidth RF channel. In addition, with too many nodes in the table, the "N" commands become useless because the response becomes too large for the user to download on a limited bandwidth channel. Even if there is sufficient bandwidth, the response may occupy several pages and the user may not be able to review anything which has scrolled off the screen. Experience has shown that 150 nodes is roughly the maximum comfortable table size.

But how do you decide *which* 150 nodes to include? How do you achieve the balance between slow, unreliable, radio-routed nodes and fast, reliable, Internet-routed ones? Some people advocate setting low route qualities for the Internet links, but unless everyone agrees on the qualities, this can lead to traffic being routed via slow, unreliable links when faster and more direct routes exist! And do you want your table full of high quality foreign nodes, to the exclusion of your local nodes? Can your users find the nodes they want?

Quality de-rating by callsign can help with this management issue. It llows you to de-rate the Netrom quality of a node or group of nodes based on the netrom callsign, instead of the route on which they were received. Thus, no matter what relative route qualities you use, you can change the relative qualities to favour your local nodes, or those which share the same language. These are more relevant.

This feature uses the global QUALADJUST keyword as follows:

```
QUALADJUST <call | "default"> <0-255>

e.g.    QUALADJUST default 120
        QUALADJUST G* 255
        QUALADJUST ZL* 200
```

The "default" argument sets the default value which will be used to de-rate all nodes not matched by any other qualadjust statement. The normal netrom de-rate algorithm is used, so 255 gives no de-rate and 0 gives full de-rate (i.e. lock out a call or group of calls). If there are no qualadjust statements the default is 255.

Qualadjust is applied to neighbour nodes and all nodes learned from netrom broadcasts. It is not applied to nodes learned by INP3 because they have no quality to de-rate, and is not applied to nodes entered by a NODE ADD command or an entry in the PZTNODES file.

SECTION 4 - FILE FORMATS

All configuration files are commented to aid setting up, but I will document them in more detail here too, just in case you damage a file or wish to study a paper copy in the garden! They are presented in alphabetical order....

ACCESS.SYS

This optional file specifies TCP/IP access requirements according to the caller's IP address. If not present, the default action is for logins to require a valid callsign only.

The entries in ACCESS.SYS are of the form:

<subnet>[/bits] <access_flags>

e.g. "44.0.0.0/8 1"

The <subnet> and [bits] parameters defines a range of IP addresses from whom Telnet connects will be accepted, and <access_flags> defines the login requirements for that subnet.

The [bits] parameter specifies how many bits, from left to right, of the source address should be matched against the corresponding <subnet> address. For example, 44.131.0.0/16 will test the incoming IP address against the left-most 16 bits of 44.131.0.0, i.e. it will match any source address beginning with 44.131. And 0.0.0.0/0 will match any IP address, which is useful for specifying the default. The chosen match will be the one with the highest [bits] value. If [bits] is not specified, it defaults to 32, i.e. an exact match is required.

The <access_flags> parameter is the sum of these flag values.

1 Valid callsigns only
2 Password required

A value of 0 means any "callsign" longer than 1 character will be accepted, and no password is required. In this context, "callsign" could be a user name. This is a zero security option, for use only for the sysop's convenience on physically secure subnets.

A value of 1 requires the user to enter a valid amateur radio callsign, i.e. a string containing alphanumeric characters in the correct format, but no password is required. This is a low security configuration with minimal inconvenience, and is suitable for use within amateur radio subnets which are not connected to the Internet. This configuration is recommended for callers who have 44.x.x.x source address, as they must have entered the network via radio, or via a password-protected gateway.

A value of 2 will cause Xrouter to accept any "username" longer than one character, providing a valid password is given. This is a medium security configuration, suitable for use on private wire subnets where amateur radio callsigns are not used.

A value of 3 requires both a valid amateur radio callsign and a
matching password to be supplied. This configuration is recommended
for use at the Internet-to-Amprnet interface, i.e. for all source IP
addresses other than 44.x.x.x

If passwords are required for user access, they should be located in
file USERPASS.SYS (see below). Do not confuse this with PASSWORD.SYS,
which is used for AX25 logins, Rlogin and FTP. USERPASS.SYS is used
for "normal" telnet (port 23) logins.

Each entry in ACCESS.SYS must begin on a new line.

Note: This is a prototype access control method, and may be subject
to change if the need arises. I will probably add other flags
to control access to other services such as the APRS server.

Example ACCESS.SYS file:
~~~~~~~~~~~~~~~~~~~~~~~~

```
# File:         ACCESS.SYS
# Purpose:      Xrouter access control for incoming Telnet port 23
#               connections.
#               Defines access control flags for given source IP address.
#
# Fields: <subnet>[/len] <flags>
#
# Flags - add together (default=1):
#
#       1       Valid callsign required.
#       2       Password required.
#
#
# Subnet/bits           Flags
# ============================
  0.0.0.0/0             3
  44.0.0.0/8            1
  192.168.0.0/16        0
#
```

BOOTCMDS.SYS

This optional file is read by Xrouter at boot time, after XROUTER.CFG and IPROUTE.SYS. It may contain certain configuration commands which would otherwise have to be typed in at the command line.
~~~~~~~~~~~~~~~~~~~~~~~~

For example, if a PPP port required any special configuration to override the defaults, the appropriate PPP commands could be included.

Commands currently accepted in BOOTCMDS.SYS are:

```
ARP         <add | publish>
BELL        <times>
DUN         <add | log>
IP          <quiet | route | ttl>
NAT         <add>
NETROM      <node add>
PPP         <port> <idle | ipcp | lcp | log | pap>
RIP         <add | learn |refuse | timeout>
```

It ignores all other commands, and those which normally produce a display, such as ARP LIST.

Note that IP, ARP, RIP, and DUN commands can also be used in IPROUTE.SYS.

CRONTAB.SYS

This optional file allows commands to be executed at certain times of the day, week, month or year.

A few of the possible uses would be:

- Additional AX25 beacons.
- Beaconning APRS objects, status, bulletins and announcememnts.
- Adjusting Netrom / IP routing to account for part-time neighbours.
- Enabling / disabling transmitters.
- Controlling peripherals via the CTRL port.
- Adjusting parameters to cope with diurnal propogation changes.
- Enabling / disabling IGATE at certain times of day / week.
- Automatic reboots / restarts / exits for batch file processing etc.

In the current version, all responses from the command processor are discarded.

An example of the file layout follows:

```
# CRONTAB.SYS - Xrouter v179 event control file.
#
# <command> will be executed if <min> <hour> and <month> match current time,
# and either the date (day of month) or day (of week) match.
#
# Fields must be separated by one or more spaces or tabs.
#
# Dates / times may be specified as single numbers, multiple numbers,
# ranges, or a mixture thereof, e.g. "0-5,7,9,15-22".  Note there must
# be no spaces within the string of characters.
#
# Use '*' in any of the first 5 fields to signify "all".
#
# <min> <hour> <date> <month> <day> <command> [args]
# 0-59   0-23   1-31   1-12    0-6
#
# Trivial examples:
#
# Every 20 mins, advertise the BBS using an APRS symbol on port 13
# 0,20,40  *  *  *  *  send 13 APZ179 v WIDE !5224.00N/00215.00WB GB7PZT BBS
#
# Every Thursday at 03:05 save nodes table to an archive
# 5 3 * * 4 SAVENODES PZTNODES.ACV
#
```

DOMAIN.SYS

This optional file is used by Xrouter to resolve host names such as "g8jvm.ampr.org" and aliases such as "lgsbbs" into their corresponding IP addresses. If not present, and an external DNS is not specified, you will have to enter the full IP address of any target host when using the PING, TELNET, TTY, and FINGER commands. It is read every time an address needs resolving, so any changes you make will take effect immediately.

Although DOMAIN.SYS was originally designed to be simpler than the more standard DOMAIN.TXT format, Xrouter is capable of understanding both formats, so you may use an existing DOMAIN.TXT simply by renaming it to DOMAIN.SYS.

Name resolution using an external DNS (Domain Name Server), if enabled, takes a finite time so you should add entries to this file for frequently contacted hosts whose addresses are stable. Externally resolved names are added automatically to DOMAIN.SYS at regular intervals, and expired entries are purged.

Each host is listed on a separate line. Each field should be separated by one or more spaces or tabs. Comments, spaces, tabs and blank lines are permissible to aid clarity. The records are case- insensitive, but most people use lower case for hostnames. The format of each record is as follows:

```
<hostname> [ttl] IN A <ip-address>

<alias>    [ttl] IN CNAME <hostname>

<hostname> [ttl] IN MX [pref] <hostname>
```

The first form maps a hostname to an IP address, and the second and third forms map an alternative hostname to a host already defined.

For example, the IP address for the GB7PZT mailbox is 44.131.91.2 so it would have an Internet Address (IN A) record like so:

gb7pzt IN A 44.131.91.2

But gb7pzt is also known locally as "pztbbs", and "kdrbbs". While there is nothing to stop you adding further "IN A" records for gb7pzt, one for each alias, but you could instead use the second form shown above, the CNAME or "Canonical Name" record like so:

pztbbs. IN CNAME gb7pzt
kdrbbs. IN CNAME gb7pzt

Thus if the user types "TEL pztbbs" or "TEL kdrbbs", the gb7pzt record is used. This removes the need to keep repeating the IP address in multiple "A" records, and makes it easier if the IP address is changed.

The MX or "Mail Exchange" records are usually used for defining alternate names for mail servers, but as Xrouter is not concerned with mail you can use them in the same way as CNAME entries, although there would be no point in doing so. The format of the additional records would be:

pztbbs. IN MX gb7pzt
kdrbbs. IN MX gb7pzt

The optional "preference" field of MX records is ignored. The DNS server doesn't currently respond to MX queries, but will eventually do so.

The optional [ttl] field in all types of entry is the "time to live" of the entry in seconds, used to expire records whose addresses are liable to change. If omitted or set to zero, the record has an unlimited lifetime.

In order to simplify the file, the ".ampr.org" is usually omitted from the records, and appended automatically when the file is read. However, hostnames which contain or end with a dot will not be extended in this manner. Thus "gb7pzt" will be extended to "gb7pzt.ampr.org", whereas "lgs." and "ns.cyberphile.co.uk" will be left alone.

ENCAP.TXT

This is *not* an Xrouter configuration file, but I was asked to make Xrouter capable of reading it.

Encap.txt is a closely guarded secret, available only to a selected few overlords, but is simply a text file which contains a huge list of amateur IP routes and the encap (IPIP) gateways which handle them. If this file is present when Xrouter boots up, it will read the routes into its routing table.

If you are not routing IP, or have set up your own encap routes in IPROUTE.SYS you do not need this file.

If you're interested, the format of routing entries is as follows:

route addprivate 44.131.91.0/24 encap 62.31.206.176

The "addprivate" signifies that the route should not be visible to users, because these people are paranoid about revealing their non-amateur IP addresses and routing secrets.

The "encap" simply tells the router to use IPIP encapsulation, i.e. it is the same as Xrouter routing mode "e".

ENCAP.TXT contains over 500 entries, and will use over 50Kb of memory if you load it. It will also make your IP routing slower, because all those routes must be searched recursively for every single datagram routed by your system. With a fast computer or low data rate you probably wouldn't notice the difference in speed, but at least I've warned you.

You are also warned that ENCAP.TXT is subject to frequent modification so you will need to obtain updated copies from time to time, and restart Xrouter to load them.

IGATE.CFG

This file is required only by the IGATE daemon, therefore you do not need it if you are not using IGATE. It is a plain text file read each time the daemon is started. The following keywords are accepted:

```
IFILTER   <from> <to> <text>  Filter rules for Internet->Packet
LOG       <0-255>             Activity logging level
MAXTRIES  <n>                 Max. server connect attempts.
PAUSE     <seconds>           Min. interval to try same server.
PFILTER   <from> <to> <text>  Filter rules for Packet->Internet
SERVER    <ipaddr:port>       Specifies Internet server to use
SKIP      <seconds>           Blacklist interval after failure
WAIT      <seconds>           Time to wait for connection
```

These are fully documented in the section relating to IGATE.

Abbreviated example file:

```
; APRS IGATE Configuration file for Xrouter version 177
;
; You can list as many servers as you like. They are tried in rotation.
SERVER 213.180.75.122:2023
;
; Wait up to 60 secs for connection before trying next server
WAIT 60
;
; Wait 60 secs between successive attempts to same server
PAUSE 60
;
; Max connect attempts before blacklisting the server
MAXTRIES 10
;
; If blacklisted, don't try the server for 1 hour
SKIP 3600
;
; IFILTER controls gating from internet to packet:
;
; IFILTER      From    To      Text
; -----------------------------------
  IFILTER      G#*     *       *
;
; PFILTER statements control gating from packet to internet:
;
; PFILTER      From    To      Text
; -----------------------------------
  PFILTER      G*      AP*     *
  PFILTER      MB7*    AP*     *
;
; Radius of Interest
;
RADIUS 150
;
; Activity logging
;
LOG     255
```

HOST

COMMAND

HOST -- Display information about a TCP/IP host

SYNOPSIS

HO[st] <hostname> | <ip address>

DESCRIPTION

The HOST command displays all known information about the specified TCP/IP host.

It would typically be used to find the IP address, given the hostname, or vice versa.

The information is limited in scope at present.

EXAMPLE

ho kidder

G8PZT:KIDDER} Host name information for kidder:
Hostname: kidder.ampr.org.
Address: 44.131.91.245

AVAILABILITY

The HOST command is available to all users.

NOTE

The information is gained from DOMAIN.SYS and DNS lookups.

IPROUTE.SYS

This optional file defines how IP datagrams should be routed, and is not required if you don't intend to route IP traffic. It is read only at boot-up, or by an "IP ROUTE LOAD" command so if you make changes to it, they won't have any effect until you reboot or use that command.

If iproute.sys is present, it saves you having to enter the IP routing manually. To reduce the number of files, this file also contains the permanent ARP (Address Resolution Protocol) entries.

If you are familiar with IP you can skip the next 4 paragraphs....

All IP addresses consist of a 32 bit binary number, which is composed of four 8 bit binary numbers. For clarity they are usually expressed as four decimal numbers separated by dots, the so called "dotted quad" form, for example 44.131.91.2 is the address for gb7pzt.

Each of the numbers which make up the quad can range from 0 to 255,
i.e. 256 numbers in total. The numbers 0, 128 and 255 are usually reserved for special purposes.

The most significant number identifies the "network" within the whole Internet. 44 is allocated to Amateur Packet Radio, or "ampr.org". Within ampr, the second number identifies the country, and in ur example 131 is the code for UK. The third number identifies the "region", and in our example region 91 is North Worcestershire. The last number identifies up to 256 separate users within the region. The addresses within a region are sometimes allocated on a first come first served" basis, or sometimes in groups to allow further subdivision of a region.

Unlike NetRom, IP routing has to be explicitly defined by the sysop, in our case using entries in IPROUTE.SYS.

Each entry should be on a separate line, and there should be one or more spaces or tabs between each field. Entries are not case sensitive. Comments are allowed in the file, providing they are on a line beginning with a semicolon.

All IP routing entries begin with the word "route", and there are two types. A "Route default" entry defines how datagrams should be routed if no other route can be found. "Route add" entries add routes to the table (what a surprise! ;-)

The general form of a route default entry is:

ROUTE DEFAULT <portnum> [<gateway> [d|v|n|e|u|r|s]]

<**portnum**> is the port number on which to route the datagram.

<**gateway**> is the IP address or hostname of the neighbouring router to whom datagrams should be routed if there is no other applicable route. If this address is not specified, the destination will be tried directly on the default port.

It is more efficient to use an IP address here, but you might need to use a hostname if the gateway has a dynamic IP address for example. In order to use hostnames, your system must have access to a DNS, and must be configured to use it.

<**d|v|n|e|u|r|s**> is the mode: (d)atagram, (v)irtual circuit, (n)etrom, (e)ncap, ip(u)dp, (r)eject or (s)ilent discard.

If omitted, the default is "Datagram", which transmits datagrams "raw" inside SLIP, PPP or AX25 UI frames, according to protocol used on the the destination port.

There is no error correction at the link layer, so datagram mode should only be used on wire links, or RF links with low loss rates.

Better performance can be obtained on lossy RF links by using Virtual Circuit mode, which transports the IP datagrams inside AX25 <I> frames, detecting and correcting errors at the link layer.

Netrom mode is less efficient, but can "tunnel" datagrams across non-ip parts of the network by wrapping them in Netrom layer 3 frames.

Encap mode is used for IP/IP encapsulation, i.e. sending 44-net datagrams across the internet by "wrapping" them in datagrams with internet addresses.

IPUDP mode is similar to Encap, except that the datagrams are first wrapped in UDP before being transported in IP.

The advantage is that UDP is usually able to pass through routers which don't support IPIP, and can be selectively routed according to the UDP service port numbers.

Datagrams matching a "reject" entry are rejected by returning an ICMP "destination unreachable" report to the sender. The "gateway" ip address should be 0.0.0.0 andd the port number is ignored. This type of entry is used to reject traffic destined for systems which don't exist, or are not reachable via any port. If simply routed on the default port, such datagrams would waste resources and would probably end up looping back to us.

Datagrams matching a "silent discard" entry are simply dropped without sending an ICMP error report. This is more suitable for suppressing malicious network probes.

At least a port number is required. A typical entry would look like:

route default 11 44.131.90.6 d

which means "route all unknown destinations via neighbouring router 44.131.90.6, on port 11, using datagram mode".

The general form of the route add entry is:

ROUTE ADD <host/bits> <gateway> <portnum> <d|v|n|e|u|r|s>

<**host**> is the target host IP address in dotted quad form.

<**bits**> is the number of bits to be matched (subnet mask)

If the <**gateway**> address is 0.0.0.0 or just "*", the destination will be tried direct on the specified port.

When making routing decisions, only the specified number of bits (0-32) of the address are compared, starting at the most significant (leftmost) bit, the remainder being ignored. This allows a group of IP addresses to be matched by a single entry.

For example, a setting of 24 would cause only the most significant 24 bits of the address to be compared, allowing a whole region (256 addresses) to be routed to a single gateway. Important - You must not omit the subnet mask!

A typical routing entry would look like this:

route add 44.131.93.0/24 44.131.93.240 5 d

which would route all region 93 traffic (44.131.93.0 - 44.131.93.255)
to the gateway 44.131.93.240 on port 5 using datagram mode.

You may specify more than one route to the same group of addresses using different bit masks. For each address within the group, Xrouter will choose the route with the highest number of matching bits.

ARP Entries

ARP entries are responsible for mapping gateway IP addresses to hardware" (i.e. AX25 or Ethernet) addresses.

In order to send an IP datagram over an AX25 or Ethernet network, it must be "wrapped" in an AX25, Ethernet, or Netrom packet, and that packet will need a destination address appropriate to that network.
For example, if I want to route a datagram to 44.131.91.2 it must be wrapped in an AX25 packet addressed to GB7PZT-5.

The system *will* sometimes work without any ARP entries, due to the process of "ARP resolution", whereby a router can make a broadcast asking adjacent systems if they know the hardware address for a given IP address, but this process takes time and the adjacent routers may not know the answer. Thus it is advisable at least to have one ARP entry for each of your direct RF neighbours. The general form of an ARP entry is:

ARP ADD <host> <ax25 | ether | netrom> <callsign | ether_address>

or

ARP PUBLISH <host> <ax25 | ether | netrom> <callsign | ethaddr>

Example ARP entries:

This one causes datagrams bound for 44.131.90.6 to be wrapped in AX25 packets addressed to GB7IPT-9:

arp add 44.131.90.6 ax25 GB7IPT-9

Whereas the following will send datagrams bound for 44.131.95.7 to the G7GHP-5 system via the GB7DIG digipeater. Up to 8 digipeaters may be used in a single comma-delimited string:

arp add 44.131.95.7 ax25 G7GHP-5,GB7DIG

And this one will wrap datagrams destined for 44.131.24.1 in Netrom packets addressed to a distant system GB7CX:

arp add 44.131.24.1 netrom GB7CX

The following will wrap the datagrams in ethernet packets.arp add 44.131.91.9 ether 00:00:1B:2C:04:81

ARP PUBLISH is used in cases where one system is "hidden" behind another, and allows other systems to discover the correct hardware address to use.

For example, my 44.131.91.127 system is only reachable via my main router 44.131.91.245. Unless all the local systems were specifically configured to route to 91.127 via 91.245, they wouldn't know how to do it.

Including the entry: "arp publish 44.131.91.127 ax25 g8pzt" on the 91.245 (g8pzt) router causes it to respond to anyone who asks for the hardware address for 91.127, giving its own ax25 address.

Manual IPROUTES:

COMMAND

IPROUTE -- Display IP routing table.

SYNOPSIS

IPR[OUTE]

DESCRIPTION

The IPROUTE command, which may be abbreviated to IPR, displays the contents of the table responsible for routing of IP datagrams.

For each route it displays the IP address, the subnet mask, the gateway address, the port and the mode (Datagram, VC or Netrom).

The command "IP ROUTES" produces an identical result.

AVAILABILITY

The IPROUTE command is available to all users.

NOTE

The IP routing table is initialised from file IPROUTE.SYS when the router is started, and may contain other entries "learned" by the system, or entered by the sysop. It is not required in any way for normal AX25 and NETROM activities.

Manual IP:

COMMAND

IP -- Display / Change IP parameters and routing.

SYNOPSIS

IP QUIET [0-255]
IP ROUTES
IP ROUTE ADD <host>[/len] <gateway> <port> [d|v|n|e|u|r|s]
IP ROUTE DROP <host> <len>
IP ROUTE DEFAULT <port> [gateway [d|v|n|e|u|r|s]]
IP ROUTE LOAD
IP ROUTE LOOKUP <host>
IP TTL [0-255]

DESCRIPTION

The IP commands are used to display and alter IP parameters and the contents of the tables responsible for routing of IP datagrams.

The IP routing table contains a series of entries which are checked whenever a datagram is routed. The "best" match is chosen, i.e. the one which matches the address and mask with the highest <bits>.

The IP ROUTE command arguments are as follows:

```
<host>           Target host IP address in dotted quad form.
<len>            No. of bits to be matched (from left) 0-32.
<gateway>        Destination gateway IP address or hostname.
<port>           Port number on which to route the datagram.
<d|v|n|e|u|r|s>  Mode: (d)atagram, (v)irtual circuit, (n)etrom,
                       (e)ncap, ip(u)dp, (r)eject, (s)ilent.
```

The IP ROUTE DEFAULT command sets up a default route which is used when no other route is found. If no gateway is specified, the target will be assumed to be a direct neighbour. If not specified, the mode defaults to datagram.

The IP ROUTE ADD command adds an entry to the routing table.

The first argument is the target host IP address, with optional mask. e.g. 44.131.90.1/32 means "match all 32 bits", whereas 44.131.90.0/24 means "match the most significant 24 bits", and would route all 256 addresses from 44.131.90.0 to 44.131.90.255. If not specified, the mask defaults to 32.

The second argument is the "gateway" address, i.e. the IP address or hostname of the system which can handle the datagram (hostname requires access to a Domain Name Server and may be slower). You may use "*" to signify a direct route (no gateway).

The third argument is the port to route the datagram on, and the last argument is the mode:

Datagram mode should be used on links with low loss rates.Virtual Circuit mode gives better performance on lossy links.- Netrom mode "tunnels" datagrams through the NetRom network.- Encap mode tunnels datagrams through non-ham IP networks.- Ipudp mode tunnels datagrams within commercial UDP/IP.- Reject mode rejects unroutable datagrams.- Silent mode silently discards unroutable datagrams.

The IP ROUTE DROP command removes an entry from the routing table. Both the target host and the mask must match.

The IP ROUTE LOAD command deletes all routes and re-loads the information from the IPROUTE.SYS file.

The IP ROUTE LOOKUP command displays the route which Xrouter will use to reach to a specified destination host.

The IP TTL command displays or changes the IP "Time To Live" on datagrams originating from Xrouter.

The IP QUIET command is used to display or set the "stealth" level. The default value of 0 causes TCP/IP to operate normally. A non-zero value disables various ICMP and TCP error responses, and should be used with caution. Please refer to the manual entry STEALTH for more information.

EXAMPLES

IP ROUTE DEFAULT 3 44.131.90.6 v
IP ROUTE ADD 44.131.95.0/24 44.131.95.240 9 d
IP ROUTE DROP 44.131.97.1 32
IP ROUTE LOOKUP 62.31.87.22
IP TTL 127
IP QUIET 2

AVAILABILITY

The only command available to non-sysops is "IP ROUTES".

NOTE

The IP routing table is necessary only for IP, and does not take any part in normal ax25 and Netrom activities. See the full manual for details on how to set up the IP system.

Manual Routes:

COMMAND

ROUTES -- Display / Edit **NetRom** routes.

SYNOPSIS

R[outes] [Q | X | Y | Z | *] [port]
R[outes] ADD <call> <port> <qual> [!] [V digis] [opts]
R[outes] DROP <call> <port>

DESCRIPTION

The ROUTES command, which may be abbreviated to "R", lists the immediately adjacent NetRom nodes, i.e. those who can be heard directly, providing those nodes are making NetRom nodes broadcasts.

For each neighbour node the display shows the port number, the neighbour's callsign, the route quality, and the number of nodes accessible through that neighbour.

A chevron (>) in the left-most column indicates a route which is in use, and an exclamation mark (!) in the right-most column indicates that the data has been "locked in" by the sysop. See example...

If any argument is supplied, it will give additional information mainly of interest to sysops. The additional fields are the current MAXFRAME, FRACK and PACLEN settings, the number of information frames sent, the number of information frames re-sent, the retry rate, which is the ratio of the two preceding figures (or * if both are zero), and the time a nodes broadcast was last heard from the neighbour.

The "R X" form shows the long term average retry rate (Rty%), plus a "running average" (Now%). It also records the peak value of the running average (Max%) and the date/time when it occurred.

The "R Y" form shows the "Smoothed Trip Time" (STT) in seconds, the number of time-domain routes (TDR) learned via that neighbour, some flags, plus the MAXTT (milliseconds) and MAXHOPS figures for each route. An stt of 0.00 indicates that the link isn't open or the value hasn't yet been measured.

The flags field is mainly for my use and is subject to change. The current flag values are:

1 Locked route
2 Neighbour is INP capable
4 Neighbour is L3RTT capable
8 Neighbour is Xrouter
16 Dynamic route quality enabled.

The "R Z" form is experimental and may be subject to change. It shows the perentage of time the neighbour has been connected (Con%), the data throughputs in bytes/sec and the date / time of last L3 activity.

The "R Q" form shows the calculated Netrom quality for each route, plus the minimum, maximum and mean deviation of the value. You may use this as a guide when deciding what quality to assign to a route. A low mean deviation indicates a quality which doesn't vary much.

If the first argument is a port number, or it is one of the above options and the second argument is a port number, e.g."R 2" or "R R 2", it will display only the routes which use that port.

ROUTE ADD adds a new route or modifies an existing one, and
ROUTE DROP removes a route.

<call> **is the callsign of the neighbour node.**

<**port**> is the radio port via which the neighbour is reached.

<qual> is the netrom "quality" to use for that route. A quality between 256 and 511 will instruct Xrouter to use "automatic" quality, with a starting point of (qual-256).

[**!**] locks the entry to prevent it being overridden by learned information.

[V digis] specifies a digipeated route, where "digis" is a string of digipeater calls seperated by commas, i.e. in the form "DIGI,DIGI,DIGI".

[opts] are optional maxframe, frack, paclen, maxtt and maxhops values to override the port defaults. The format is:[maxframe [frack [paclen [maxtt [maxhops]]]]], i.e. in order to specify maxtt you must also specify maxframe, frack and paclen Use zero in any field for which you don't wish to overwrite the port default.

EXAMPLES

If no arguments are supplied, the output looks like this:

```
G8PZT:KIDDER} Routes:
Port Callsign  Qty Nod
>  5 G4FPV     150  70!
>  7 GB7PZT    250   1!
>  8 GB7WV-12  100  32!
>  9 GB7GH     150 104!
  10 GB7CL     150   1!
> 11 GB7IPT-7  150   3!
  12 G1LOA-10  150   2!
```

The R R form produces an output similar to this:

G8PZT:KIDDER} Routes:

```
Port Callsign Qty Nod Max Frack Pac  Sent Resent Rty%  Last Heard
 >  5 G4FPV    150  70! 5   7000 160 15550   1507   9% 09/06 13:46
 >  7 GB7PZT   250   1! 7   5000 240 25387      4   0% 09/06 13:47
 >  8 GB7WV-12 100  32! 1   4000 120 12170   3426  28% 09/06 13:35
 >  9 GB7GH    150 104! 4   7000 120  7442   1335  17% 09/06 13:47
   10 GB7CL    150   1! 3   7000 240   330      1   0% 09/06 13:23
 > 11 GB7IPT-7 150   3! 2   7000 120 19401   2928  15% 09/06 13:47
   12 G1LOA-10 150   2! 4   7000 120 10745   3379  31% 09/06 13:39
```

The R X form produces an output like this:

G8PZT:KIDDER} Routes:

```
Port Callsign  Qty Nod    Sent  Resent  Rty%  Now%   Max% @dd/mm hh:mm
>  7 GB7PZT    250   1!  10067       0    0%   0.0    0.0
>  9 GB7GH     150  93!    333      17    5%   5.3    5.6  26/08 20:26
> 12 G1WXA-1   150   2    1231      83    6%   1.6   24.4  26/08 12:37
(End of list)
```

route add g8pzt 5 100

route add g6yak 2 100 ! V G8EPR,G8NTU 5 7000
route add g8klm 3 150 ! 0 0 245 2000 3
route drop mb7uyl 14

AVAILABILITY

The ROUTES command is available to all users.
The ADD and DROP subcommand is only available to sysops.

Manual ARP:

COMMAND

ARP -- Display / Edit the ARP table.

SYNOPSIS

```
ARP [ADD <host> <hwtype> <addr>]
     [DROP <host> <hwtype>]
     [FLUSH]
     [LIST]
     [PUBLISH <host> <hwtype> <addr>]
```

DESCRIPTION

The ARP command is used to display and edit the Address Resolution Table, responsible for mapping IP datagrams to callsigns.

If no arguments are supplied, or the argument is LIST, the table is listed.

The forms ARP ADD and ARP PUBLISH are used to add an entry to the table. PUBLISH is used when a host is "hidden" on a network which is only accessible via Xrouter, and allows Xrouter to respond to arp requests for the hidden system, by returning its own hardware address.

ARP DROP is used to delete an entry, and ARP FLUSH removes temporary entries, i.e. those pending resolution.

<host> is an IP address in dotted quad form.

<hwtype> is the hardware type, i.e. "ax25" or "ether".

<addr> is the hardware address, i.e. callsign or ethernet.

EXAMPLES

```
ARP ADD 44.131.91.2 ax25 gb7pzt-5    Add ax25 entry
 ARP PUB 44.131.91.127 ax25 g8pzt     Publish 91.127
 ARP DROP 44.131.91.7 ax25            Delete ax25 entry
 ARP LIST                             List the table
 ARP                                  List the table
```

AVAILABILITY

The LIST subcommand is available to everyone, but ADD, DROP, PUBLISH and FLUSH are sysop-only.

NOTES

In order for this command to have any meaning, the router must have an IP address and be connected to an IP-capable network.

You cannot add hardware address types for which there is no compatible interface, e.g. an attempt to add an Ethernet address on a system with no Ethernet interfaces will fail.

PASSWORD.SYS

This optional file contains the passwords for AX25 remote sysop access, RLOGIN and FTP. If the file is not present, these forms of access will not be possible.

There should be one entry for each remote sysop. Each entry should be on a separate line and consists of two fields separated by one or more spaces or tabs. The first field is the remote sysop's callsign (without an SSID) and the second field is the corresponding password string. For example:

```
G8PZT   thisispaulaspassword
G1LOA   thisisrobertspassword
```

The fields are not case sensitive, and the passwords can be anything you like, with a couple of limitations, namely PASSWORDS MUST NOT CONTAIN SPACES, and the TOTAL LINE LENGTH MUST NOT EXCEED 80 CHARACTERS. You should try to choose a password that you can easily remember, but that others would find hard to guess. It should preferably be longer than 5 characters, to reduce the chances of someone guessing it. The longer you make it, the more secure it will be. A random string is the most secure, because even if someone works out part of it, they won't be able to guess the remainder.

You may have a different password for each sysop, or you might choose to give everyone the same password. Obviously the former is the most secure, but it's completely up to you.

PPPHOST.n

This optional file is required only if a PSTN modem is connected to Xrouter and auto-answer is enabled.

It is read only when a modem caller issues the command "XLINK PPP", and may contain PPP commands to override the default PPP configuration for the duration of the call. The default configuration is probably OK for most purposes, but you may for example wish to change the inactivity timeout or the IP address.

There may be several of these files, one for each modem port. The "n" represents the port number on which the caller has accessed Xrouter.

PZTNODES

This file is identical in format and function to the BPQNODES file used by BPQ for node & route table saving and recovery.

The nodes and routes tables are saved to this file every NODESINTERVAL, at close-down, and whenever the SAVENODES command is used (unless a different filename is specified).

The ROUTE entries are first, and have the following format:

ROUTE ADD <call> <port> <quality> [!] [VIA call call ...] [options]

Where <call> is the callsign of the neighbour node, <port> is the port upon which it is reached, and <quality> is a figure between 0 and 255 indicating how "good" the route is. "!" indicates a locked route (i.e. one which will not expire). If the neighbour is reached via digipeaters, the next field should start with "VIA" and each digi call should be separated by exactly ONE space, with the end marked by exactly TWO spaces.

The [options] fields are "non-standard" parameters which can override the port defaults for that route as follows:

[maxframe [frack [paclen [maxtt [maxhops]]]]]

For example:

```
ROUTE ADD W7XCV 1 100
ROUTE ADD G8UYL 2 240 ! 5 7000 120
ROUTE ADD G7DIG 5 ! VIA M7FRT M3RED  2
```

The first line shows unlocked neighbour W7XCV on port 1 with quality of 100. The second line shows neighbour G8UYL locked in on port 2 with a quality of 240, maxframe of 3, frack of 7000, and paclen 120. The third line shows neighbour G7DIG locked in using a digipeated path via M7FRT and M3RED, with maxframe of 2.

Following the ROUTE entries, the remainder of the file consists of NODE entries, one per line. The format is as follows:

NODE ADD <alias:call> <route1> <port1> <qual1> [!] [<route2> ...]

Where <alias:call> is the alias and callsign of the distant node, <route1> is the callsign of the primary route to that node, for which there must be a route entry, <port1> is the port used to reach that neighbour and <qual1> is the quality of that route. "!" indicates a locked route. There can be up to 3 different routes listed for each node.

For example:

```
NODE ADD #TLFRD:GB7IPT-7 G8PZT 1 142 ! G8UYL 2 139
NODE ADD BRUM:GB7BM G8PZT 1 94 G8UYL 2 92
NODE ADD BUXTON:GB7DAD-8 G8PZT 1 22 G8UYL 2 21
```

TELPROXY.MSG

This optional file is only required if you are using the Telnet Proxy feature. It contains text which is sent to the user upon uplink connection to the proxy.

If the file is not present, the following default text is displayed:

```
Use: Tel[net] <ipaddr> [port]
Ready
>
```

---------- example follows ----------

To connect to another system, enter "TEL[net] <ipaddr> [port]"

e.g. Tel 44.131.91.2 7777

Type HELP for any other help

---------- example ends ----------

USERPASS.SYS

The entries in this optional file are used, if dictated by entries in ACCESS.SYS, to control normal Telnet (port 23) logins, and APRS server (port 1448) logins They are not used for sysop logins, Rlogin or FTP.

Each line consists of two fields, i.e. the same format as PASSWORD.SYS. The first field is a user callsign, without SSID, and the second is the user's password. For extra security, it is advisable for sysops to use different passwords for telnet and rlogin.

Example USERPASS.SYS file:

```
; USERPASS.SYS for Xrouter
;
; This file contains passwords for Telnet and Modem logins.
;
; Fields are: <callsign> <password>
; Callsigns should not include SSID
;
G8PZT amazon
G7CXZ drvxcdfre
;
```

XROUTER.CFG

This file is mandatory, and contains most of the configuration info
for Xrouter. It is read only when the program starts.

It consists of entries of the general form <keyword>=<value>, each on
a separate line. Keywords are not case sensitive, and in general they
may be in any order, but interfaces MUST be defined before ports.
Blank lines are allowed, and comment lines must begin with a semicolon
(;) in the leftmost column. Lines must not exceed 255 characters.

Note that some timings are in millisecs, some are secs.

Global Keywords
~~~~~~~~~~~~~~~~
The following keywords are accepted in the "global" section of
XROUTER.CFG, i.e. outside CONSOLE, INTERFACE, PORT, APPL and ROUTES
definition blocks. (* = mandatory)

APPL

Marks the start of an APPL definition block. The application
number must be between 1 and 8, and corresponds to the
application's position on a BPQ "APPLICATIONS=" line.

Some applications have fixed application numbers, e.g. BBS's
and PMS's are usually hardwired as the first application,
while Host programs such as PAC4 are usually the second.

Example: APPL=3

APPLQUAL

Default NetRom quality to broadcast for BPQHOST and socket
applications. Default = 150. See APPL section.

APRSCALL

Callsign used by the Igate to identify itself in beacons and
third party messages. If omitted, it defaults to Nodecall.

Example: APRSCALL=MB7UXX

BOTWINBGCOLOR

See CONSOLE definition section.

BOTWINTXCOLOR
~~~~~~~~~~~~~~~~

See CONSOLE definition section.

CHATALIAS *

Alias for chat server (6 chars max).

I suggest this should end with "CHT" and begin with something geographically relevant, e.g. BHMCHT for Birmingham, LDSCHT for Leeds etc., so it can be easily identified in node tables.

If you make CHATALIAS the same as NODEALIAS, the chat server will not be directly connectible.

Example: CHATALIAS=KDRCHT

CHATCALL *

Chat server callsign. This may be the same as the NODECALL, but must use a different SSID, otherwise the chat server will not be directly connectible.

Note - the chat server is an integral part of the system and cannot be disabled. You may prevent it from being directly connectible by setting CHATCALL and CHATALIAS the same as NODECALL and NODEALIAS, but it will still be available to all users via the "chat" command.

Example: CHATCALL=G8PZT-8

CHATLINKS

List of chat servers to which this server will link. For netrom links you must supply the *callsigns* not the aliases. You may use multiple CHATLINKS lines if required.

Unilateral links are not allowed - each partner in this list must place your CHATCALL in their CHATLINKS list. You may only use partners who are in your node tables.

Xrouter allows only limited interconnection with "Ping-Pong Converse" servers, because the two systems are completely different (Ping Pong doesn't have local channels for instance).

At present, links to Xrouter peers must use Nterom, and links to Ping-Pong peers must use TCP/IP.

Don't link with distant servers - if the links are too slow your users will get poor service. You may disable all outgoing and incoming linking by omitting this keyword.

Examples: CHATLINKS=G9XOT-8,G7DTY-8

CHATLINKS=brmcht 80.185.22.37:3601

CHATQUAL

Chatcall quality to broadcast (0-255, default=255).

This can be used to limit the visibility of your server to a reasonable geographical area, and discourage chat server "dxing".

Example: CHATQUAL=150

CMDWINBGCOLOR

See CONSOLE definition section.

CMDWINTXCOLOR

See CONSOLE definition section.

COMMAND

Creates a custom command.

Up to 16 single-word commands can be created, allowing the sysop to automate frequently used command strings. For example you might wish to set up BBS, CONV and DXCLUSTER to point to local systems.

Each command is defined using a separate "COMMAND=" string, and the arguments are <cmd_name> <real_cmd>.
e.g. "COMMAND=BBS C 7 GB7PZT" means "create a new command called BBS, which translates to the sequence C 7 GB7PZT"

CONSOLE

Begins a CONSOLE definition block. The argument can be between 1 and NUMCONSOLES.

A console definition block is used to override the default console settings for one console. Only the values which differ from the globals need be specified. Alternatively you may omit the global console settings altogether and fully define each console.

Example: CONSOLE=3

CONSOLECALL

See CONSOLE definition section.

CTEXT

Connect text. Specifies one or more lines of text which is sent to an incoming caller upon connection. CTFLAGS controls which callers receive the text.

There is no limit on the number of lines of text you can specify, but no line must exceed 255 characters. The end of text is marked by *** on a line by itself.

Example:
CTEXT
Kidderminster spur AX25/IP Router.
Type ? for list of commands.
***

CTFLAGS

Connect Text Flags. These flags control which incoming connects will receive CTEXT. The value is the sum of the required flags from the following list:

1 Send ctext if connect is to Node/port alias
2 Send ctext if call is to Node/port call
4 Send ctext on L4 connects.
8 Send ctext to TCP (TELNET) callers.

The default is 9 (Alias and TCP only).

Example: CTFLAGS=1

CTRLADDR

This is the address (in hexadecimal) of the parallel port (if any) used for controlling / monitoring external hardware.

If this is the same as WATCHADDR, only the most significant 7 bits will be available for control if the hardware watchdog is enabled.

Example: CTRLADDR=378

DESQVIEW

Specifies whether or not Xrouter is running within a Desqview environment.

If Desqview is used, this must be set to 1, otherwise Desqview may not function properly. If Desqview is not in use set the argument to 0 (default) or omit the keyword altogether.

Example: DESQVIEW=1

DNS

Specifies the IP address of a Domain Name Server, which will be consulted when trying to resolve hostnames which aren't in DOMAIN.SYS.

You may use this keyword more than once to specify several DNS servers if necessary, which will be consulted in the order they are specified. If no DNS servers are specified, resolution will use DOMAIN.SYS only.

Example: DNS=62.31.176.115

DXFLAGS

Optional flags to control the DX heard display (default = 0).
Add together required options:

0 Show directly heard stations
1 Show digipeated stations

Example: DXFLAGS=0

ECHOCOLOR

See CONSOLE definition section.

ENABLE_LINKED

Specifies who, if anyone, may use the "*** LINKED AS" command.
The default is "N" and the possible values are:

Y Available to everyone.
A Usable by applications only.
N Disabled entirely.

Example: ENABLE_LINKED=A

HIDENODES

If this is set to 1, nodes whose alias begins with "#" will not be displayed.

Example: HIDENODES=1

HOSTINTERRUPT

Interrupt number to use for the BPQ-compatible host interface

The host interface may only be used in a Windows 95/98 or Desqview environment. To use it you must load PZTHOST.EXE before Windows then run XROUTER.EXE and each application within separate dos windows.

If you are not using host interface, comment this parameter out, or set it to 0.

Example: HOSTINTERRUPT=127

HOSTNAME

Host name for TCP (optional). If you omit this, it will default to "NODEALIAS:NODECALL".

Example: HOSTNAME=g8pzt.ampr.org

IDINTERVAL

The interval in minutes between broadcasts of IDTEXT. Set to 0 to disable ID broadcasts. Default is 15 minutes.

Example: IDINTERVAL=29

IDLETIME

Idle link shutdown time in seconds (default=900). A Netrom link will be closed after this period of inactivity.

Example: IDLETIME=300

IDTEXT

Specifies a one-line ID message to be sent every IDINTERVAL.

If your APRS-format static position code is included, starting within the first 40 characters, you will be visible on APRS maps and the MHeard function will record distances to heard stations.

The format is "!ddmm.mmN/dddmm.mmE#" where dd represents

degrees of latitude/longitude and mm.mm represents minutes to two decimal places. "N" and "E" may be replaced by "S" and "W" as appropriate. The end of text is marked by *** on a line by itself.

Example:
IDTEXT
!5224.00N/00215.00W# Kidderminster Router (KDRMIN)
***

IGATE

Controls whether or not the APRS Packet<>internet gateway is started at boot-up.

A zero value (default) doesn't start the IGATE (but it can be started anytime using "start igate"), and a non-zero value starts it immediately. Leave this commented out, or set to zero if you aren't running a gateway.

Example: IGATE=1

INFOTEXT

Specifies one or more lines of text to be sent in response to the plain 'I' command. The end of text is marked by *** on a line by itself.

Note that you may create info "topics" which the user can access using the "I <topic>" command. Each topic should occupy a separate, appropriately named file in a directory named "INFO", and the filenames must have the extension ".INF"

Example:
INFOTEXT
You are connected to part of the Kidderminster Packet Radio Network, run on behalf of Fourpak by Paula G8PZT.

For information on Fourpak, contact G4FPV @ GB7GLO, or see the website at www.g8pzt.pwp.blueyonder.co.uk/fourpak/index.htm
***

INTERFACE

Begins an INTERFACE definition block. The argument is a number between 1 and 255 which identifies the interface in subsequent PORT blocks.

Example: INTERFACE=2

IPADDRESS

Specifies the router's IP address. If you aren't routing IP, comment this out or set it to 0.0.0.0

The router normally uses a single IP address for all ports, but you may specify additional addresses for each port.

Example: IPADDRESS=44.131.91.4

IPTTL

IP "Time To Live" for datagrams originated by Xrouter.

IPTTL overrides the default "Time To Live" (TTL) of 255. This is the maximum number of hops an IP datagram can make before it is killed.

A low value ensures that datagrams stuck in routing loops will die quickly, but be aware that internet-routed packets may easily make 20 or 30 hops, so don't set it too low. Ignore this if you haven't enabled IP routing.

Example: IPTTL=100

L3TTL

Layer 3 "Time To Live" for NetRom packets originated by Xrouter.

This is the maximum number of NetRom systems a packet will traverse before it is killed, and prevents packets from being stuck in routing loops for ever. Default is 25.

Example: L3TTL=30

L4DELAY

NetRom Layer 4 delay. This is the number of seconds to delay a L4 ack in case further frames arrive (default = 3).

10 secs is probably OK on normal AX25 networks, but is excessive on wire links. However, the system will attempt to adjust this this value to cope with prevailing conditions.

Example: L4DELAY=10

L4RETRIES

Netrom layer 4 retries (default = 3).

This is the number of consecutive L4 connect/disconnect or re-transmission attempts which are allowed before the link is abandoned.

Example: L4RETRIES=3

L4TIMEOUT

Netrom layer 4 timeout. This is the interval in seconds between L4 retries and L4 connect/disconnect attempts.

Default is 120, and you are advised against setting this much lower.

Example: L4TIMEOUT=90

L4WINDOW

NetRom layer 4 window. This is the number of unacknowledged L4 frames allowed before we must stop to await an ack. The default is 10.

Example: L4WINDOW=4

LOG

Controls activity logging. Setting LOG=1 will log all connects, disconnects, user-entered commands and chat server activity. LOG=0 (default) disables logging.

Make sure you have enough disk space. Not recommended for long term use on floppy-based machines. Can be overridden by LOG command at the command line.

Example: LOG=0

LOWMEM

Minimum free memory below which an automatic RESTART will take place. Default is 5000 bytes. Use LOWMEM=0 to disable the automatic restart.

MAXHOPS

Specifies the maximum acceptable hop count for nodes, as another means of limiting the Time Domain network horizon (see MAXTT) It can be overridden for individual ports by using

MAXHOPS in the PORT definition block, or for individual routes by using ROUTE ADD. Default is 30 hops.

MAXLINKS

Maximum simultaneous AX25 level 2 links (default=30). The higher the value you specify, the less free memory will be available.

MAXNODES

Maximum no. of nodes allowed in Netrom nodes table. Default is 200. An excessive number of nodes has many disadvantages, so you should be very careful if you choose to exceed this.

MAXTT

Specifies the maximum accepted trip time for nodes, to define the Time Domain network horizon, like MINQUAL does for Quality Domain. It can be overridden for individual ports by using MAXTT in the PORT definition, or for individual routes using ROUTE ADD. Default is 5000 (50 seconds).

MIDWINBGCOLOR

See CONSOLE definition section.

MIDWINTXCOLOR

See CONSOLE definition section.

MINQUAL

Minimum node quality to add to node table (default = 10).

This is the global value which will apply to all ports unless overridden by a port minqual.

Example: MINQUAL=10

NODEALIAS *

Node alias (6 characters max.).

Aliases beginning with "#" are not displayed in node lists, and are typically used for "linking only" nodes.

You should preferably choose an alias which is geographically relevant beyond your own local area, for example BRSTOL, LEEDS, or BRUM are good, because users can recognise them in node tables, whereas GAB1 and WBA are bad - where on earth are they?

Example: NODEALIAS=KDRMIN

NODECALL *

Node callsign. Up to 6 chars plus optional SSID between 1 and 15.

This is the callsign used for all L3/4 operations, and the default for L2 operations on each port.

Example: NODECALL=G8PZT-6

NODESINTERVAL

Interval, in minutes, between NetRom nodes broadcasts. Default is 60 mins.

Example: NODESINTERVAL=30

NUMCONSOLES

Number of consoles (default = 3).

You may have up to 5 "virtual" consoles, upon which the sysop may conduct independent sessions.

Consoles are selected by using Alt-1 through alt-5, or the left and right arrow keys. Setting this to 0 disables all console activity, saving memory and preventing the direct screen writes which upset Desqview.

Example: NUMCONSOLES=4

OBSINIT

NetRom obsolescence counter initial value (default = 5).

The obsolescence counter is maintained for each neighbour in the routes table, and is reset to the OBSINIT value whenever a nodes broadcast is heard from the neighbour.

Example: OBSINIT=5

OBSMIN

NetRom obsolescence count minimum to broadcast (default = 3).

The obsolescence count (see above) for each NetRom neighbour is decremented every NODESINTERVAL, and if it drops below OBSMIN, the route is considered obsolete, and nodes learned via that neighbour are no longer included in nodes broadcasts.

Example: OBSMIN=3

PACLEN

Sets global default Paclen, i.e. the maximum length of the information-bearing portion of an AX25 packet.

The maximum is 256, and default is 120. This may be overridden on a port by port basis by PORT paclens.

The choice of paclen depends on many factors such as the link data rate, transmission and addressing overheads, link contention and error rate etc.

Small packets are less likely to be corrupted by errors or QRM, but are inefficient when addressing or transmission overheads (e.g. txdelay) are large. Big packets are more efficient, but are more likely to be lost if the link has errors or QRM, and the re-transmissions will take longer.

Example: PACLEN=180

PMSALIAS

Ax25/Netrom alias for integral PMS (Personal Message System).

This is required if the PMS is to be visible via NetRom. It must not be the same as any other alias or callsign used on the system.

Example: PMSALIAS=PZTPMS

PMSCALL

AX25 / NetRom callsign for integral PMS.

This is required if the PMS is to be connectible via AX25 or NetRom. It must not be the same as any other callsign or alias used on the system.

Example: PMSCALL=G8PZT-2

PMSQUAL

PMS quality to include in Netrom nodes broadcast.

Set this to 0 (default) to suppress L4 visibility. You may only use a non-zero value, if both PMSCALL and PMSALIAS are defined.

Example: PMSQUAL=50

PORT

Begins a PORT definition block. The argument is the port number, which must be between 1 and 32767 and not used by any other port.

Example: PORT=3

QTH

Station location. 32 characters maximum.

Example: QTH=Kidderminster, Worcs.

QUALADJUST

Adjusts the quality of node entries according to callsign.

Syntax: QUALADJUST <call | "default> <0-255>

Examples: QUALADJUST default 120
QUALADJUST G* 255

See section entitled "Netrom Quality Manipulation".

ROUTES

Specifies one or more NetRom routes to lock in - Syntax is as follows:

<callsign> <port> <quality> [! [maxframe [frack [paclen]]]]

You must specify at least Callsign, port and quality. If you include the lock flag (!) the route will be locked in, and will only be changed by a replacement entry with the lock flag set.

If you don't include the lock flag, the route will eventually expire if not confirmed by the reception of nodes broadcasts.

In either case, if the file PZTNODES is present, its contents will override the entries here, subject to the locking rules above, and the sysop may also edit routes while the router is running.

Maxframe, frack and paclen are optional. If specified they override port defaults for that route. Note FRACK is expressed in millisecs, e.g. 7000 = 7 secs. Maxframe > 7 will cause Modulo-128 to be attempted on that route. The section ends with *** on a blank line.

Example: Lock in a link to g6yak on port 1 with quality 100 and maxframe 32 :-
ROUTES
g6yak 1 100 ! 32
***

ROWS

XR32 only - Rows allows a variable number of screen rows. You need to add ROWS=50 (or whatever value you want) into XRouter.CFG, preferably as the first config command. The default is ROWS=25

Example: ROWS=60

RXCOLOR

See CONSOLE definition section.

SAVER

Screen saver interval in seconds. (0 = default = disable screen saver)

Example: SAVER=300

SESSLIMIT

Overall limit on no. of concurrent sessions per user, across all ports. You might like to restrict "troublesome" users this way! Max setting = default = 255

Example: SESSLIMIT=10

T3

Ax25 level 2 T3 timer (link check interval) in seconds. (default = 180). This is the period of inactivity after which a poll frame is sent to test if the link is still open.

Example: T3=150

TOPWINBGCOLOR

See CONSOLE definition section.

TOPWINTXCOLOR

See CONSOLE definition section.

TXCOLOR

See CONSOLE definition section.

WATCHADDR

Address (hex) of parallel port used to drive hardware watchdog.

If you set a valid parallel port address here, the least significant data bit of the port will be toggled several times per second if the router is working correctly.

You can use external circuitry to detect if the output stops toggling for more than a few seconds, and use it to reboot the computer. Note that your circuit must look for an AC signal, because the output could lock up high or low if the computer crashes.

Example: WATCHADDR=378

WATCHDOG

Software Watchdog timeout in seconds.

If you set a non-zero value here, the router will attempt to reboot if the code goes into an endless loop.

Note: the watchdog keeps running if you shell to DOS using Alt-F9, so keep shell sessions short or the router will think you've forgotten to return and will reboot!

Example: WATCHDOG=120

Console Definition Keywords
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
The following keywords can be used within a CONSOLE definition block (* = mandatory):
~~~~~~~~~~~~~~~~~~~~~~~~~~~~

BOTWINBGCOLOR

Background colour for the bottom window, i.e. the menu bar. Default = CYAN.

Example: BOTWINBGCOLOR=GREEN

BOTWINTXCOLOR

Text colour for the bottom window, i.e. the menu bar. Default is BLACK.

CMDWINBGCOLOR

Background colour for the command line, i.e. where the console commands are entered. Default is BLUE.

CMDWINTXCOLOR

Text colour for the command line. Default is YELLOW.

CONSOLECALL

Callsign for console operations. You can set this independently of NODECALL or you may set them the same. You may at any time override this callsign using the "linked as" command.

Example: CONSOLECALL=G8PZT

ECHOCOLOR

Colour used for echoing sysop's commands to main window. Default is YELLOW.

ENDCONSOLE *

Ends a console definition block. Mandatory.

MIDWINBGCOLOR

Background colour for the main window. Default is BLACK.

MIDWINTXCOLOR

Text colour for the main window. Default is WHITE.

MPORTS

Default ports to monitor on this console. See MPORT command in sysop command section for details of the argument format.

MMASK

Default monitor (trace) mask for this port. See sysop command section for more details.

RXCOLOR

Text colour for displaying received data. Default is GREEN.

TOPWINBGCOLOR

Background colour for the top window, i.e. the status bar. Default is CYAN.

TOPWINTXCOLOR

Text colour for the top window, i.e. the status bar. Default is BLACK.

TXCOLOR

Text colour for displaying transmitted data. Default is RED.

Console Text / Background Colours
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Permissible TEXT Colours are:

BLACK, NAVY, GREEN, CYAN, RED, MAGENTA, BROWN, SILVER, GREY, BLUE, LIME, TURQUOISE, PINK, CERISE, YELLOW, WHITE

Permissible BACKGROUND colours are:

BLACK, NAVY, GREEN, CYAN, RED, MAGENTA, BROWN, SILVER

Interface Definition Keywords
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The following keywords may be used within INTERFACE definition blocks (* = mandatory):

CHANNEL

Used for SCC cards only. This is the physical channel or port on the card (A, B, C or D). You may use CHANNEL=A etc. or COM=1 etc. but not both.

COM

Com number (1 - 16), used by ASYNC, YAM and SCC types only.

COM is mandatory for ASYNC interfaces. Mandatory for YAM interfaces only if IOADDR and INTNUM are not specified. For SCC cards, you may use COM=1 etc. instead of CHANNEL=A etc.

ENDINTERFACE *

Marks the end of the INTERFACE definition block. Subsequent keywords will be treated as GLOBAL.

ETHADDR

Ethernet address in form xx:xx:xx:xx:xx:xx, used only for Ethernet interfaces. You shouldn't need this because Xrouter will read the address from the card.

FLOW

Flow control options (ASYNC interfaces only):

0 = No flow control
1 = Hardware (RTS/CTS) flow control
2 = Software (XON/XOFF) flow control (TTY link only)
3 = Hardware AND software flow control

If not specified, flow control defaults to NONE.
Don't use Xon/Xoff with KISS.

INTNUM

Hardware or software interrupt number in decimal.
For ASYNC interfaces, intnum is optional for com 1 - 4.
For SCC cards, use same INTNUM for all channels on card.

IOADDR

Hardware device I/O address in hex, e.g. UART base register address.

For ASYNC interfaces, IOADDR is optional for com 1 - 4.
For SCC cards, use same IOADDR for all channels on card.

KISSOPTIONS

Options for KISS interfaces only:

NONE - Plain KISS (most TNC's use this) (default)
POLLED - For TNCs which send only when polled.
CHECKSUM - Packets protected by checksum. You can only use this option if your TNC supports it.
ACKMODE - For TNC's which inform the router when a frame has been transmitted on air.
SLAVE - Router will act like a polled KISS TNC, sending only when commanded to do so.

Polled and slave are mutually exclusive.
BPQKISS eproms require POLLED and CHECKSUM, and their use of ackmode is optional.

MTU *

Maximum Transmission Unit. This specifies the maximum size of the data portion of an IP packet.

Setting MTU over 256 will crash most BPQ nodes, or at the very least cause the packet to be lost. Xrouter can handle frames up to 1500 bytes, but BPQKISS TNC's are limited by 340 byte buffers. Received IP datagrams larger than MTU are fragmented to suit the outgoing link MTU.

PROTOCOL

Protocol to use on the interface:

NONE - Use this with type=loopback
ASCII - Remote consoles (TTY) via ASYNC ports
SLIP - For TCP/IP over RS232
PPP - For TCP/IP over RS232
KISS - For driving KISS TNCs or wired links.
MODEM - Hayes compatible PSTN modem.
NETROM - Netrom backend serial link.
ETHER - Ethernet (requires ETHDRV.EXE)
TNC2 - TNC2 emulation.
HDLC - For use with SCC cards, YAM modem, INTERNAL interfaces, and some EXTERNAL drivers.

SPEED

The serial port baud rate for ASYNC interfaces, or the radio baud rate for HDLC cards. 50-115200 bauds (Decimal). Don't include a comma. If set to zero (HDLC cards only), the modem must supply a x1 clock (TXC on RTXC pin, RXC on TRXC pin).

TYPE *

Interface type as follows:

ASYNC	Serial port
LOOPBACK	For test purposes
EXTERNAL	User supplied driver (as BPQ)
BAYCOM	Baycom USCC card.
RLC100	Thorcom RLC100 scc card.
DRSI	DRSI scc card
PC100	Pac-comm PC100/120 scc card
PC120	4 port PC100 scc card.
PA0HZP	PA0HZP opto-scc card
AXIP	For AX25 over IP wormhole
AXUDP	For AX25 over UDP wormhole
INTERNAL	For HDLC applications.
ITACARD	Yet another SCC card.
YAM	YAM 1200/2400/9600 modem

Port Definition Keywords

The following keywords are used within PORT definition blocks (* = mandatory):

APPLMASK

The APPLMASK parameter is used only if you are running applications via the host interface. It specifies which applications will be directly connectible on this port.

The default is 255, which allows all applications. The value is made up by adding together the following numbers:

1 - Enable Application 1
2 - Enable Application 2
4 - Enable Application 3
8 - Enable Application 4
16 - Enable Application 5
32 - Enable Application 6
64 - Enable Application 7
128 - Enable Application 8

If you want an application to be directly connectible on a port, it must have a callsign, an alias or both, and the corresponding bit in that port's applmask must be set.

Example: APPLMASK=5 (enable applications 1 and 3)

APRSPATH

APRSPATH specifies the default digipeater path for APRS frames originated by APRS messaging shell and Igate. If you omit this, the frames will be sent without any digipeaters. Messaging shell users may override this path.

Example: APRSPATH=RELAY

BCAST

BCAST specifies a list of destinations for "broadcasting". Received non-digipeater UI frames, addressed to one of these destinations, will be re-broadcasted on all ports which have a matching address in their BCAST list, e.g. to broadcast mail beacons from a BBS on one port onto several other ports.

The callsigns must be separated by commas and there should be no spaces in the list.

Example: BCAST=MAIL,ALL

BCFROM

BCFROM is used to specify a list of callsigns who may use the broadcast facility (see BCAST).

If you wish to restrict the broadcast facility to certain senders only, list the callsigns here. If no calls are specified, the facility is unrestricted. Separate the calls by commas, and don't include any spaces in the list.

Example: BCFROM=GB7PZT,GB7MAX

CFLAGS

CFLAGS allows level 2 uplinking and/or downlinking to be prevented, e.g. on APRS-only ports. Use VERY carefully!

Default is 3. Sysop can always downlink. Add together the desired options from this list:

1 Allow incoming connections (uplinks)
2 Allow outgoing connections (downlinks)

Example: CFLAGS=2 ; (allow downlinks only)

CHANNEL

Specifies which channel (A to P) on the interface will be used by this port. The default is "A".

Example: CHANNEL=C

CWID

CWID is used only by SCC ports. It specifies a callsign (8 characters max.) which will be sent every 30 minutes using Morse code. If omitted, no cwid is sent.

Example: CWID=G8PZT

DHCP

The DHCP keyword specifies whether or not the port IP address will be obtained dynamically using DHCP (DHCP=1) or specified statically (DHCP=0). Default is 0.

DIGIFLAG

Digipeater control flag. 0=no digipeat. Default=7
Add together required options from this list:

```
1    Digipeat UI frames
2    Digipeat non-UI frames
4    Enable RELAY generic APRS digipeating
8    Enable TRACE and TRACEn-N APRS digipeating.
16   Enable WIDE and WIDEn-N (Well sited stations only!)
32   Enable L4 APRS tunnelling
64   Enable frames from RF to Igate
128  Enable frames from Igate to RF
```

Example: DIGIFLAG=5 ; all UI + RELAY generic.

DIGIPORT

Specifies the port on which digipeated frames will be transmitted. Default is 0, i.e. this port.

Example: DIGIPORT=3 ; Digipeat onto port 3

ENDPORT *

Marks the end of the PORT definition block. Subsequent keywords will be interpreted as GLOBAL.

EXCLUDE

EXCLUDE specifies a list of one or more callsigns from whom

AX25 L2 frames will be ignored.

It would typically be used on a user-access port to prevent connections from trouble-makers and pirates. Separate callsigns with commas, and don't include any spaces.

Example: EXCLUDE=NOCALL,P1RAT ; Ignore these users

FRACK

AX25 "T1" timer, i.e. frame acknowledgement time, in milliseconds. Default is 7000.

After sending an AX25 packet, this is the amount of time Xrouter will wait for an "ack" before considering the packet to be lost.

The T1 timer starts when the link layer dispatches the packet to the MAC (Media Access Control) layer, so you must allow at least enough time for (MAXFRAME * PACLEN) packets to be sent, plus an ack packet to be received, plus the other end's RESPTIME, plus both end's TXDELAYs, plus a generous allowance for channel congestion.

I strongly recommend a frack no less than 7000 millisecs for average 1200 baud links.

Example: FRACK=7000 ; 7 seconds

FULLDUP

If you set FULLDUP=1, Xrouter will transmit whenever it needs to, without waiting for the other end to stop.

Used only by hardware which is capable of simultaneous transmission and reception, such as full duplex radio or wire links. Default is 0 (simplex / half duplex).

Example: FULLDUP=1 ; Full duplex

ID *

Mandatory text string to identify port on "PORTS" display. May contain spaces. Please make it informative.

Example: ID=144.825 MHz 9k6 TCP/IP users

IDPATH

Specifies the destination and digipeater path for ID beacons.

The default AX25 destination and path is "ID" with no digipeaters. You may wish to modify this, for example on APRS

ports, to digipeat your beacon.

Example: IDPATH=APRS,RELAY,WIDE

IDTEXT

Optional ID text to use, instead of global IDTEXT, on this port only (e.g. for APRS ports).

Only one line of text (248 characters max.) may be specified.

Example: IDTEXT=!5224.00N/00215.00W# (Kidder)

INITSTR

Modem initialisation string (MODEM ports only). This is a
string of characters which will be sent to the modem when Xrouter is started.

Example: INITSTR=ATM0

INTERFACENUM *

Number of the interface this port is attached to.

INTERLOCK

Interlock is only used by SCC cards, because KISS TNC's make
their own decisions about when to transmit, and Xrouter has no control over that process.

If a non-zero INTERLOCK value is specified, no two ports with
the same value will transmit at the same time. Default=0.

Example: INTERLOCK=4

IPADDRESS

IP address for use on this port. If this is specified, it
will be used instead of the global IP address for all TCP/IP operations on this port.

Example: IPADDRESS=44.131.91.5

IPLINK

IP address of AXIP or AXUDP link partner (AXIP and AXUDP ports only).

Example: IPLINK=44.33.22.11

L3ONLY

L3ONLY=1 prevents users from making AX25 level 2 downlinks on the port, whilst allowing L3/4 (Netrom) operation.

The default is 0 (L2 and L3/4 allowed).

MAXFRAME

Specifies the maximum number of frames Xrouter will send to an AX25 partner before it must wait for an ack.

Normal AX25 allows up to maxframe=7, but if you set a value between 8 and 63 Xrouter will attempt to use Modulo-128 (Extended AX25) on outgoing links.

If the port PACLEN is set to 0, Xrouter will dynamically adapt MAXFRAME (and PACLEN) to the link conditions, to maximise throughput.

The default value of MAXFRAME is 3.

MAXHOPS

Specifies the default maximum hops for all neighbour routes learned on this port. It can be overridden for individual routes using by "locking-in" the route at the end of XROUTER.CFG, or by a ROUTE ADD command in the PZTNODES file or at the command prompt. Defaults to global MAXHOPS.

MAXTT

Specifies the default maximum trip time for all neighbour routes learned on this port. It can be overridden for individual routes using by "locking-in" the route at the end of XROUTER.CFG, or by a ROUTE ADD command in the PZTNODES file or at the command prompt. Defaults to global MAXTT.

MHEARD

Enable/disable the MHEARD function on this port.

The number specifies how many callsigns to maintain in the list. Set to 0 to disable MHEARD. Default is 15 calls.

Example: MHEARD=10 ; Mheard enabled, 10 calls

MHFLAGS

MHFLAGS controls which callsigns are recorded in the MH list, and defaults to 255 (show everything).

The argument is the sum of the required options from this list:

1 Show directly heard stations
2 Show directly heard digipeaters
4 Show digipeated stations

Example: MHFLAGS=1 ; show directly heard stations only

MINQUAL

Minimum quality to add to node table for nodes received via this port. Defaults to global MINQUAL.

If specified, this overrides the global minqual, and can be used to exclude unreachable and marginal nodes.

Example: MINQUAL=10

MINTXQUAL

Minimum node quality to include in broadcasts on this port. Range 0 to 255, Default is 0.
This is used to limit the size of nodes broadcasts on ports which are low bandwidth, low quality, or where the neighbours have limited nodes table capacity.

NODESINTERVAL

Interval, in minutes, between NetRom nodes broadcasts on this port only. Defaults to global NODESINTERVAL.

Whilst the Netrom network usually works on a 60 minute broadcast cycle, some types of software insist on a much smaller broadcast interval.

It would be harmful to the established network if sysops tried to accommodate these neighbours by setting the global NODESINTERVAL to a smaller value, but using this keyword on a per-port basis you can keep these neighbours happy without disrupting the rest of the Netrom network.

If you set NODESINTERVAL=0, Xrouter will ignore received nodes broadcasts on this port, but will allow L3/L4 activity if QUALITY is non-zero.

Example: NODESINTERVAL=10

PACLEN

Specifies the maximum length of the information-bearing portion of ax25 packets transmitted on this port only.

If not specified, it defaults to the global PACLEN. See global paclen for more details.

If the port PACLEN is set to 0, Xrouter will dynamically adapt PACLEN (and MAXFRAME) to the link conditions, to maximise throughput.

Example: PACLEN=160 ; Allow 160 byte packets on this port

PERSIST

PERSIST is the AX25 "Probability to transmit" in a given time slot (see SLOTTIME), used to minimise media contention.

Persist should be set to (255 / (no. of users on channel)), e.g. for a channel with an average of 10 users on at any one time you would set Persist to 25. Default is 64.

Example: PERSIST=85 ; Average 3 users

PIPE

PIPE allows frames received (and optionally sent) on this port to be copied to another port, e.g. to allow a PMS on one port to see the traffic on another port.

Unless the "bi-directional" option (see below) is specified, pipes are one way. In order to have two way traffic using a uni-directional pipe, you must set up a reverse pipe on the opposite port.

You may pipe several ports to a single destination port, but you can only have one *outgoing* pipe from any port.

Pipes are capable of generating an immense amount of traffic, so use them with care - your target port MUST be capable of handling the traffic load.

Pipes can be made "selective", by adding a comma-delimited callsign list, e.g. "PIPE=4 GB7PZT,KDRBBS". This will reduce the loading on the target port, by piping only the frames with the specified calls in the destination field.

Pipes can be made "bi-directional" by adding 512 to the PIPEFLAG value (see below: suggested value = 515). If a frame

is piped on a bi-directional pipe, the source call is remembered so that responses will be piped back to the sender. Thus a reverse pipe is not needed.

Bi-directional pipes are useful where a BBS has a front end router - simply set up bi-directional selective pipes from each user port to the BBS port, and set up the BCAST option so that the UI mail headers are broadcast on each user port. The BBS will then allow direct connect and will respond to resync requests.

To disable piping, set PIPE=0 or just omit the command.

Example: PIPE=2 ; Pipe frames from this port to port 2

PIPEFLAG

PIPEFLAG is only used when piping is active, and controls which frames are piped. The default is 3.

The argument is the sum of the required options as follows:

```
1    - UI frames *not* addressed to nodecall/alias.
2    - Non-UI frames *not* addressed to nodecall/alias.
4    - UI frames addressed to nodecall/alias.
8    - Non-UI frames addressed to nodecall/alias.
16   - UI frames transmitted by us.
32   - Non-UI frames transmitted by us.
64   - Allow budlisted users to be piped.
128  - Netrom frames
256  - IP / ARP frames
512  - Bi-directional piping
```

Example: PIPEFLAG=5 ; Pipe received UI frames only

PORTALIAS

Specifies an AX25 alias for this port, to be used in addition to the NODEALIAS.

Example: PORTALIAS=KDRMIN

PORTALIAS2

Yet another PORTALIAS (see above).

PORTCALL

Specifies an AX25 callsign to use on this port, in addition to the NODECALL.

Example: PORTCALL=G8PZT-1

PROXY

Remote NET/ROM systems to whom we will tunnel L2 frames.
(See PROXY section of manual for full explanation)

Example: PROXY=GB7PZT,GB7BBS

QUALADJUST

Qualadjust is accepted, but doesn't do anything. I may remove it at some time in future.

QUALITY

Default quality for nodes whose broadcasts are received on
this port (default=10). Setting this to 0 disables all L3/4 activity on this port.

Example: QUALITY=30

RESPTIME

AX25 "T2" (delayed ack) timer in milliseconds, i.e. the time
to wait after receiving a frame before sending an ack.

This allows multi-frame transmissions to be acknowledged with a single ack, increasing link efficiency.

Resptime should be long enough for a link partner to transmit
a full-sized information frame, i.e. approximately
((their_paclen * 10000) / RFbauds) millisecs, otherwise you will send unnecessary ack frames.

1500 (default) is OK for 1200 bauds with paclen=120, but
2200 would be more appropriate if their paclen was 256.

Example: RESPTIME=2000 ; 2 seconds

RETRIES

AX25 maximum consecutive connect / disconnect / resend
attempts before giving up (default = 10).

I can't see any point in setting retries more than 10, other
than for test purposes. If you need so many retries it's a
useless link and you're just wasting everyone else's airtime.
The higher you set this value, the longer users will have to
wait to get a "failure with" for a non-contactable destination.

Example: RETRIES=5

RFBAUDS

RF baud rate (default = 1200).

This parameter is used with "real" tnc's and YAM modems attached via RS232, because the RF baud rate is usually different to the RS232 baud rate. It simply helps Xrouter make timing decisions (such as nodes broadcast inter-packet timing) and report certain stats correctly.

Example: RFBAUDS=2400

SESSLIMIT

Sesslimit specifies the maximum simultaneous connects each user is allowed on this port. Default is 255.

Example: SESSLIMIT=5

SLOTTIME

CSMA interval timer (millisecs), used with PERSIST to make channel access decisions (default=100).

If the channel is clear, the TNC (or SCC/YAM card) will generate a random number which is then compared with the PERSIST value. If it is less than PERSIST, the TNC will wait for the SLOTTIME interval, then repeat the action, otherwise it will transmit immediately.

This improves throughput by ensuring that everyone doesn't transmit as soon as the channel is clear, which would cause collisions and retries.

Example: SLOTTIME=100

SOFTDCD

Softdcd is used only by SCC cards and defaults to 0.

If SOFTDCD=1 the real dcd will be ignored, and the SCC driver will use the presence of HDLC data as a DCD indication.

Using SOFTDCD=1 with an open squelch generates a *huge* interrupt loading, which may cause degradation of performance, depending on the PC type, so I don't recommend it.

Example: SOFTDCD=0

SYSOP

If you set SYSOP=1, all users who connect on this port will get full sysop status without answering a password challenge.

This is intended ONLY FOR USE ON SECURE LINKS, such as RS232 or Ethernet, and the default is zero.

TXDELAY

Transmit keyup delay in milliseconds (default=300).

After keying the transmitter, the TNC (or SCC / YAM card) will wait for this interval before sending HDLC data. This allows the TX to stabilise and the partner's squelch to open.

Example: TXDELAY=500 ; Synthesiser is slow to lock

TXPORT

Port to transmit on (default = 0 = this port).

You would typically use this when several ports share a single transmitter, with separate receivers.

Example: TXPORT=5 ; Transmit on port 5

TXTAIL

TX keydown delay in milliseconds (default = 30).

This is the interval, after sending a frame, for which the transmitter remains active. It is intended to allow CRC and closing flags to be sent.

Due to PC timing inaccuracies, you should set this no lower than 100 for SCC cards!

Example: TXTAIL=100

UDPLOCAL

UDPLOCAL is the UDP port number for the local end of an AXUDP link. Used only by AXUDP ports. Default is 93

Example: UDPLOCAL=7388

UDPREMOTE

UDPREMOTE is the UDP port number for the remote end of an AXUDP link. Used only by AXUDP ports. Default is 93.

UNPROTO

Destination callsign and optional digipeater string used for unproto broadcasts from applications on this port. Separate the callsigns with commas.

Example: UNPROTO=CQ,G6YAK

USERS

Maximum no. of simultaneous users on this port. Default is 255 which means "no limit".

Example: USERS=9

VALIDCALLS

Validcalls allows AX25 frames only from the specified callsigns (opposite of BUDLIST), and is typically used to prevent users from connecting to link-only ports.

Example: VALIDCALLS=G6YAK,G6AMU ; Accept only these users

Application Configuration Keywords

The following keywords can be used within APPL definition blocks (* = mandatory):

APPLALIAS

Specifies an ax25 and NetRom alias for the application. It is required if the application is to be visible to Netrom. AX25 connects are allowed subject to the port APPLMASK parameter (see PORT keyword section).

Example: APPLALIAS=PZTBBS

APPLCALL

Specifies callsign for the application, allowing ax25/NetRom callers to connect directly to it.

AX25 connects are allowed subject to the port APPLMASK parameter (see PORT keyword section). APPLCALL is required if the application is to be NetRom visible.

Example: APPLCALL=GB7PZT

APPLNAME

The name by which the application is accessed from the

router's command line. e.g. "BBS". If a user types this name, they will be connected to the application.

Example: APPLNAME=BBS

APPLQUAL

Netrom quality to broadcast (0-255) for this application. Default is global APPLQUAL.

If a non-zero value is specified, and *both* APPLCALL and APPLALIAS are defined, they will be included in nodes broadcasts and the application will be connectible at level 4.

Example: APPLQUAL=150

ENDAPPL *

Marks the end of the APPL definition block. Subsequent keywords will be treated as GLOBAL.

Example XROUTER.CFG file

The following example is for illustration purposes only, and I have "borrowed" a few callsigns for use in the EXCLUDE, ROUTES and VALIDCALLS fields. This does not represent my setup.

```
; XROUTER.CFG  Configuration file for Xrouter version 1.76
; ======================================================
;
; The XROUTER.EXE program reads this file only when it starts.
;
; Keywords can be in almost any order, but interfaces MUST be defined
; before ports.
;
; Note some timings are in millisecs, some are secs.
;
; Blank lines are allowed.  Comment lines must begin with semicolon in
; the leftmost column.  Lines must not exceed 255 characters in length.
;
; This is a sample, to illustrate all the options and typical settings.
; You *must* edit it to your requirements!  I suggest you de-activate
; the bits you don't need by putting semicolons at the beginning of
; the lines.
;-----------------------------------------------------------------
;
;       Interrupt number to use for the BPQ-compatible host interface.
;       The host interface may only be used in a Windows 95 or Desqview
;       environment. To use it you must load PZTHOST.EXE before Windows
;       then run XROUTER.EXE and each application within separate dos
;       windows.
;       If you are not using host interface, comment this parameter out,
;       or set it to 0.
;
ROWS=50                 ; For XR32 Only. Default is ROWS=25.  Number of screen ROWS.
;
;HOSTINTERRUPT=127
;
;       DESQVIEW specifies whether or not Xrouter is running in a
;       Desqview environment, and must be set to 1 if Desqview is used.
;       The default is 0.
;
; DESQVIEW=1
;
;       Station Identification:
;       ~~~~~~~~~~~~~~~~~~~~~~~
;
;       Node callsign: Up to 6 chars plus optional SSID between 1 and 15
;       This is the callsign used for all L3/4 operations, and the default
;       for L2 operations on each port.
;
NODECALL=G8PZT-6
;
;       Node alias: Up to 6 characters.
;       Aliases beginning with "#" are not displayed in node lists, and
```

```
;       are typically used for "linking only" nodes.
;       You should preferably choose an alias which is geographically
;       relevant beyond your own local area, for example BRSTOL, LEEDS,
;       or BRUM are good, because users can recognise them in node tables,
;       whereas GAB1 and WBA are bad - where on earth are they?
;
NODEALIAS=KDRMIN
;
;       Callsign for APRS IGATE (optional).
;       This callsign is used by the Igate to identify itself in beacons
;       and third party messages.  If omitted, it defaults to Nodecall.
;
; APRSCALL=MB7Uxx
;
;       Callsign for console operations.  You can set this independently
;       of NODECALL or you may set them the same.  You may at any time
;       override this callsign using the "linked as" command.
;
CONSOLECALL=G8PZT
;
;       IP address for IP routing.  Your local IP co-ordinator should
;       be able to assign you one.  If you aren't routing IP (shame on
;       you!) comment this out or set it to 0.0.0.0
;       The router normally uses a single IP address for all ports, but
;       you may specify additional addresses for each port.
;       (If you are routing, remember to define routes and hostnames in
;       IPROUTE.SYS and DOMAIN.SYS respectively)
;
IPADDRESS=44.131.91.4
;
;       IP address of primary Domain Name Server, which will be consulted
;       when trying to resolve hostnames which aren't in DOMAIN.SYS
;       If not specified, resolution will use DOMAIN.SYS only.
;
; DNS=62.31.176.115
;
;       Host name for TCP (optional).  If you omit this, it will default
;       to "NODEALIAS:NODECALL".
;
HOSTNAME=g8pzt.ampr.org
;
;       Chat Server parameters:
;       ~~~~~~~~~~~~~~~~~~~~~~~
;       Note - the chat server is an integral part of the system and
;       cannot be disabled.  You may prevent it from being directly
;       connectible (if you wish to deprive users of facilities) by
;       setting CHATCALL and CHATALIAS the same as NODECALL and
;       NODEALIAS, but it will still be available to all users via
;       the "chat" command.
;
;       Chat server callsign. This may be the same as the nodecall, but
;       must use a different SSID, otherwise the chat server will not
;       be directly connectible.
;
```

```
CHATCALL=G8PZT-8
;
;       Alias for chat server (6 chars max).  I suggest this should end
;       with "CHT" and begin with something geographically relevant,
;       e.g. BHMCHT for Birmingham, LDSCHT for Leeds etc., so it can be
;       easily identified in node tables. If you specify the NODEALIAS
;       here, the chat server will not be directly connectible.
;
CHATALIAS=KDRCHT
;
;       List of chat servers to which this server will link.  You must
;       supply the *callsigns* not the aliases.  Unilateral links are not
;       allowed - each partner in this list must place your CHATCALL in
;       their CHATLINKS list.  You may only use partners who are in your
;       node tables.  PZT chat servers may be interconnected with each
;       other, but *NOT* with "Ping-Pong Converse" servers.  Don't link
;       with distant servers - if the links are too slow your users will
;       get poor service. You may disable all outgoing and incoming
;       linking by omitting this line (or commenting it out).
;
CHATLINKS=G9XOT-8,G7DTY-8
;
;       Chatcall quality to broadcast.  This can be used to limit the
;       visibility of your server to a reasonable geographical area,
;       and discourage chat server "dxing". Default is 255, i.e. chat
;       call is visible over same distance as nodecall.
;
CHATQUAL=150
;
;       Callsign and alias of integral PMS.  If you define neither, the
;       PMS will only be accessible using the "PMS" command.
;
PMSCALL=G8PZT-2
;
PMSALIAS=PZTPMS
;
;       PMS quality to include in Netrom nodes broadcast.
;       Set this to 0 to suppress L4 visibility.
;       You may only use a non-zero value, if both PMSCALL and PMSALIAS
;       are defined.
;
PMSQUAL=50
;
;       Station location (32 chars max).
;
QTH=Kidderminster, Worc's
;
;       IGATE controls whether or not the APRS Packet<>internet gateway
;       is started at boot-up. A zero value (default) doesn't start the
;       igate (but it can be started anytime using "start igate"), and
;       a non-zero value starts it immediately.
;       Leave this commented out, or set to zero if you aren't running
;       a gateway.
;
```

```
; IGATE=1
;
;       Optional Software Watchdog.  If you set a non-zero value here,
;       the router will attempt to reboot if the code goes into an
;       endless loop.  The value is in seconds.  Note: the watchdog
;       keeps running if you shell to DOS using Alt-F9, so keep shell
;       sessions short or the router will think you've forgotten to
;       return and will reboot!
;
WATCHDOG=120
;
;       Address (hex) of parallel port used to drive hardware watchdog.
;       If you set a valid address here, the least significant data bit
;       of the port will be toggled several times per second if the router
;       is working correctly.  You can use external circuitry to detect
;       if the output stops toggling for more than a few seconds, and use
;       it to reboot the computer.  Note that your circuit must look for
;       an AC signal, because the output could lock up high or low if
;       the computer crashes.
;
; WATCHADDR=378
;
;       Address (hex) of parallel port used for controlling / monitoring
;       external hardware.  If this is the same as WATCHADDR, only the
;       most significant 7 bits will be available for control if the
;       hardware watchdog is enabled.
;
; CTRLADDR=378
;
;       IPTTL overrides the default "Time To Live" (TTL) of 255.  This
;       is the maximum number of hops an IP datagram can make before it
;       is killed.  A low value ensures that datagrams stuck in routing
;       loops will die quickly, but be aware that internet-routed packets
;       may easily make 20 or 30 hops, so don't set it too low.  Ignore
;       this if you haven't enabled IP routing.
;
; IPTTL=100
;
;       Colour Settings:
;
;       Note the defaults have been chosen for their relative luminances
;       and contrast, and you may find certain combinations may not be
;       visible.
;
;       Permissible TEXT Colours are:
;
;       BLACK, NAVY, GREEN, CYAN,     RED, MAGENTA, BROWN, SILVER,
;       GREY,  BLUE, LIME, TURQUOISE, PINK, CERISE, YELLOW, WHITE
;
;       Permissible BACKGROUND colours are:
;
;       BLACK, NAVY, GREEN, CYAN, RED, MAGENTA, BROWN, SILVER
;
;
```

```
;       Top status bar background colour
;
TopWinBgColor=CYAN
;
;       Top status bar text colour
;
TopWinTxtColor=BLACK
;
;       Main window background colour
;
MidWinBgColor=BLACK
;
;       Main window text colour
;
MidWinTxtColor=WHITE
;
;       Command line background colour
;
CmdWinBgColor=NAVY
;
;       Command line text colour
;
CmdWinTxtColor=YELLOW
;
;       Bottom menu bar background colour
;
BotWinBgColor=CYAN
;
;       Bottom menu bar text colour
;
BotWinTxtColor=BLACK
;
;       Colour for displaying outgoing (transmitted) data
;
TxColor=RED
;
;       Colour for displaying incoming (received) data
;
RxColor=GREEN
;
;       Colour used for echoing Sysop's commands to main window.
;
EchoColor=YELLOW
;
;
;       The default console settings may be overridden on a per-console
;       basis by using an optional console definition block as shown
;       in the example below.  Only the values which differ from the
;       globals defined above need be specified. (You may note that I
;       have omitted some values from the gaudy example below).
;       Alternatively you may omit the globals and fully specify each
;       console.
;
CONSOLE=3
```

```
        TOPWINBGCOLOR=SILVER
        MIDWINBGCOLOR=NAVY
        MIDWINTXTCOLOR=WHITE
        CMDWINBGCOLOR=GREEN
        BOTWINBGCOLOR=SILVER
        CONSOLECALL=G8PZT-4
        TXCOLOR=PINK
        RXCOLOR=LIME
ENDCONSOLE
;
;       Screen saver interval in seconds. (0 = disable screen saver)
;
SAVER=300
;
;       In the following section there is no limit on the number of
;       lines of text you can specify, but no line must exceed 255
;       characters.  The end of text is marked by *** on a line by itself.
;
;       This text is sent to an incoming caller.  CTFLAGS controls which
;       callers receive the text.
;
CTEXT
Kidderminster spur AX25/IP Router.
Type ? for list of commands.
***
;
;       This text is the response to the plain 'I' command.
;       Note that you may create info "topics" which the user can access
;       using the "I <topic>" command.  Each topic should occupy a
;       separate, appropriately named file in a directory named "INFO",
;       and the filenames must have the extension ".INF"
;
INFOTEXT

You are connected to part of the Kidderminster Packet Radio Network,
run on behalf of Fourpak by Paula G8PZT.

For information on Fourpak, contact G4FPV @ GB7GLO, or see the website at
www.g8pzt.pwp.blueyonder.co.uk/fourpak/index.htm

IP = 44.131.91.4 CHAT = G8PZT-8:KDRCHT or telnet port 3600

This system runs G8PZT's Xrouter software - Type ? for list of commands.

***
;
;       This ID message is sent every IDINTERVAL.
;       Only the first line is sent.
;       If your APRS-format static position code is included, starting
;       within the first 40 characters, you will be visible on APRS maps
;       and the MHeard function will record distances to heard stations.
;       The format is "!ddmm.mmN/dddmm.mmE#" where dd represents degrees
;       of latitude/longitude and mm.mm represents minutes to two decimal
;       places. "N" and "E" may be replaced by "S" and "W" as appropriate.
```

```
;
IDTEXT
!5224.00N/00215.00W# Kidderminster Router (KDRMIN) 44.131.91.4, Chat=KDRCHT:G8PZT-8
***
;
;       CTFLAGS controls which connects receive CTEXT.
;       Add together the following numbers:
;
;               1       Send ctext if connect is to Node/port alias
;               2       Send ctext if call is to Node/port call
;               4       Send ctext on L4 connects.
;               8       Send ctext to TCP (TELNET) callers.
;
;       Default is 9 (Alias and TCP only).
;
; CTFLAGS=1
;
;       You may have up to 5 "virtual" consoles, upon which the sysop
;       may conduct independent sessions.  Consoles are selected by
;       using Alt-1 through alt-5, or the left and right arrow keys.
;       Setting this to 0 disables all console activity, saving memory
;       and preventing the direct screen writes which upset Desqview.
;
NUMCONSOLES=3           ; No. of virtual consoles (max=5)
;
;       Usage Log:
;       Setting LOG=1 will log all connects, disconnects, user-entered
;       commands and chat server activity. LOG=0 disables this function.
;       Make sure you have enough disk space.  Not recommended for long
;       term use on floppy-based machines.  Can be overridden by LOG
;       command at the command line.
;
LOG=0
;
;       Overall limit on no. of concurrent sessions per user, across
;       all ports.  You might like to restrict "troublesome" users
;       this way! Max setting = default = 255
;
SESSLIMIT=255
;
;       Optional flags to control the DX heard display (default=0)
;       Add together:
;
;               0       Show directly heard stations
;               1       Show digipeated stations
;
DXFLAGS=0
;
;       Enable_linked controls who, if anyone, may use the "*** LINKED AS"
;       command.  The default is "N", and the possible values are:
;
;               Y       Command is unrestricted.
;               A       Only applications may use the command.
;               N       No-one may use the command.
```

```
;
; ENABLE_LINKED=A
;
;       L4 PARAMETERS
;       =============
;
;       (Don't adjust these unless you *really* understand all the
;       implications.  I've set them the same as BPQ)
;
;       No. of seconds between L4 retries and L4 connect/disconnect
;       attempts.
;
L4TIMEOUT=90
;
;       No. of seconds to delay a L4 ack in case further frames arrive.
;       10 secs is probably OK on normal AX25 links, but is excessive
;       on wire links.  However, the system will attempt to adjust this
;       this value to cope with prevailing conditions.
;
L4DELAY=10
;
;       No. of unacked L4 frames allowed before we stop to await an
;       ack.
;
L4WINDOW=4
;
;       No. of L4 connect/disconnect or retransmission attempts before
;       link is abandoned.
;
L4RETRIES=3
;
;       L3 PARAMETERS
;       =============
;
;       Obsolescence counter initial value
;
OBSINIT=5
;
;       Obsolescence counter minimum to broadcast
;
OBSMIN=3
;
NODESINTERVAL=60        ; Mins between nodes b/casts. (0 = disable)
L3TTL=25                ; Max L3 hops
;
;       If this is set to 1, nodes whose alias begins with "#" will
;       not be displayed.
;
HIDENODES=1
;
;       Minimum quality to add to node table.  This is the global value
;       which will apply to all ports unless overridden by a port minqual.
;       If not specified, the default is 10.
;
```

```
MINQUAL=10
;
; MAXNODES=150     ; Maximum nodes to include in table (default=200)
;
; ======================================================================
;      Ax25 Level 2 Global Parameters
; ======================================================================
;
T3=180              ; Link check interval in secs (180).
IDLETIME=900        ; Idle link shutdown timer in secs (900)
IDINTERVAL=15       ; Minutes between ID broadcasts (0=disable)
PACLEN=120          ; Global paclen (default=120)
; MAXLINKS=20       ; Max. simltaneous L2 links (default=30)
;
; ======================================================================
; Interface definitions - These MUST come before any port definitions
; ======================================================================
;
;      Unlike BPQ, you first define the interfaces with the outside world,
;      then you define the ports that use those interfaces.  This is
;      because some interfaces (e.g. ASYNC) can support several channels
;      by use of the appropriate protocol (e.g. KISS).
;      Each interface has a number by which the ports identify it.  With
;      this method you may easily switch a whole group of ports onto a
;      different com port if you get a hardware breakdown for example.
;
;      TYPE:        ASYNC        ; Serial port
;                   LOOPBACK     ; For test purposes
;                   EXTERNAL     ; User supplied driver (as BPQ)
;                   BAYCOM       ; Baycom USCC card.
;                   RLC100       ; Thorcom RLC100 scc card.
;                   DRSI         ; DRSI scc card
;                   PC100        ; Pac-comm PC100/120 scc card
;                   PC120        ; 4 port PC100 scc card.
;                   PA0HZP       ; PA0HZP opto-scc card
;                   AXIP         ; For AX25 over IP wormhole
;                   AXUDP        ; For AX25 over UDP wormhole
;                   INTERNAL     ; For HDLC applications.
;                   ITACARD      ; Yet another SCC card.
;                   YAM          ; YAM 1200/2400/9600 modem
;
;      COM:         For ASYNC interfaces, this is the Com number (1 - 16)
;                   For SCC cards, you may use COM=1 etc. instead of
;                   CHANNEL=A etc.
;
;      CHANNEL      For SCC cards only!  This is the physical channel
;                   or port on the card (A, B, C or D).  You may use
;                   CHANNEL=A etc. or COM=1 etc. but not both.
;
;      (For ASYNC interfaces, ioaddr and intnum are optional for com 1 - 4)
;      (For SCC cards, use same IOADDR and INTNUM for all channels on card)
;      IOADDR       IO device address (hex)
;      INTNUM       Interrupt number (decimal)
;
```

```
;       SPEED         The serial port baud rate for ASYNC interfaces, or
;                     the radio baud rate for HDLC cards.
;                     50-115200 bauds (Decimal). Don't include a comma.
;                     If set to zero (HDLC cards only), the modem must
;                     supply a x1 clock (TXC on RTXC pin, RXC on TRXC pin).
;
;
;       PROTOCOL:     Protocol to use on the interface:
;                     NONE    - Use this with type=loopback
;                     ASCII   - Remote consoles (TTY) via ASYNC ports
;                     SLIP    - For TCP/IP over RS232
;                     PPP     - For TCP/IP over RS232
;                     KISS    - For driving KISS TNCs or wired links.
;                     MODEM   - Hayes compatible PSTN modem.
;                     NETROM  - Netrom backend serial link.
;                     ETHER   - Ethernet (requires ETHDRV.EXE)
;                     TNC2    - TNC2 emulation.
;                     HDLC    - For use with SCC cards, YAM modem,
;                               INTERNAL interfaces, and some EXTERNAL
;                               drivers.
;
;       KISSOPTIONS:  Options for KISS interfaces only...
;                     NONE    - Plain KISS (most TNC's use this) (default)
;                     POLLED  - For TNCs which send only when polled.
;                     CHECKSUM - Packets protected by checksum.  You can
;                                only use this option if your TNC supports it.
;                     ACKMODE - For TNC's which inform the router when a
;                               frame has been transmitted on air.
;                     SLAVE   - Router will act like a polled KISS TNC,
;                               sending only when commanded to do so.
;                               (Polled and slave are mutually exclusive)
;                     (BPQKISS eproms require POLLED and CHECKSUM, and
;                     their use of ackmode is optional)
;
;       FLOW:         Flow control options (ASYNC interfaces only):
;                     0 = No flow control
;                     1 = Hardware (RTS/CTS) flow control
;                     2 = Software (XON/XOFF) flow control (TTY link only)
;                     3 = Hardware AND software flow control
;                     If not specified, flow control defaults to NONE.
;                     Don't use Xon/Xoff with KISS.
;
;       MTU:          Specifies largest IP packet.  Setting this over 256
;                     will crash most BPQ nodes, or at the very least cause
;                     the packet to be lost.
;                     The router can handle single frames up to 1500 bytes,
;                     but BPQKISS TNC's are limited by 340 byte buffers.
;                     Received IP datagrams larger than MTU are
;                     fragmented to suit the outgoing link MTU.
;
;       Each interface definition starts with INTERFACE= and ends with
;       ENDINTERFACE.  The number following INTERFACE= is merely a label
;       allowing ports to refer to the interface.  It must be specified,
;       but the numbers themselves, and the order in which they're defined
```

```
;       are not important.
;
INTERFACE=1
        TYPE=ASYNC
        COM=1
;       IOADDR=3F8
;       INTNUM=4
        SPEED=9600
        PROTOCOL=KISS
        KISSOPTIONS=CHECKSUM
        MTU=256
ENDINTERFACE
;
;
INTERFACE=2
        TYPE=ASYNC
        COM=2
        SPEED=4800
        PROTOCOL=KISS
        KISSOPTIONS=CHECKSUM
        MTU=256
ENDINTERFACE
;
;
;       Example "loopback" interface, allowing self-connects without going
;       via an external system
;
INTERFACE=3
        TYPE=LOOPBACK
        PROTOCOL=KISS
        MTU=576
ENDINTERFACE
;
;       Example interface for drivers conforming to BPQ's "external"
;       spec such as baycom modem driver or ODIDRV ethernet driver.
;
;INTERFACE=4
;       TYPE=EXTERNAL           ; BPQ-compatible "External" driver
;       PROTOCOL=HDLC
;       INTNUM=96
;ENDINTERFACE
;
;       Example of an interface for TTY (remote console).  You would
;       connect the com port via a null modem to a dumb terminal or
;       computer running a terminal emulator program, such as TELIX.
;
;INTERFACE=5
;       TYPE=ASYNC
;       COM=1
;       SPEED=19200
;       MTU=256
;       PROTOCOL=ASCII
;       Flow control is optional. Be VERY careful if you are monitoring
;       over a flow controlled link, because if you pause the display for
```

```
;       a long time, the data will back-up in the router until it becomes
;       strangled.
;       FLOW=2          ; Xon/xoff flow control
;ENDINTERFACE
;
;       Example of an interface for SCC card - in this case BAYCOM USCC.
;       You require one interface per channel on the card.
;       Use the same base address for all channels.
;
;INTERFACE=6
;       TYPE=BAYCOM
;       COM=2           ; Channel B
;       IOADDR=300      ; Card base address=300H
;       INTNUM=5        ; Card IRQ=5
;       SPEED=0         ; DF9IC 1200/9600 NRZ modem
;       PROTOCOL=HDLC   ; Only HDLC supported at present
;       MTU=256
;ENDINTERFACE
;
;       Example AXIP pseudo-interface. Only TYPE=AXIP and MTU required,
;       all other parameters will be ignored (at present).
;       At least one AXIP interface is needed if you intend to do AX over IP.
;       You can attach an unlimited number of ports to one AXIP interface,
;       or you can use separate interfaces if you need different MTU's.
;
;INTERFACE=7
;       TYPE=AXIP
;       MTU=256
;ENDINTERFACE
;
;       Example multi-protocol Ethernet interface using PZT's ETHDRV.EXE
;       driver. You must first load the appropriate driver for your
;       ethernet card (such as NE2000.COM), then ETHDRV.EXE, before
;       starting XROUTER.  See documentation for further details.
;       This interface will directly support IP, ARP and AX25.
;
;INTERFACE=8
;       TYPE=EXTERNAL
;       PROTOCOL=ETHER
;       MTU=1600
;       INTNUM=125
;ENDINTERFACE
;
;       Example AXUDP pseudo-interface. Apart from "TYPE=AXUDP" see the
;       comments relating to AXIP interface.
;
;INTERFACE=9
;       TYPE=AXUDP
;       MTU=256
;ENDINTERFACE
;
;       Example YAM interface. One of these required for each YAM port.
;       Modem must be initialised with YAMINIT.EXE before starting Xrouter.
;
```

```
;INTERFACE=10
;       TYPE=YAM
;       COM=1           ; 1-4 or specify IOADDR/INTNUM instead
;       MTU=256
;       SPEED=1200      ; Radio speed
;       PROTOCOL=HDLC   ; Only HDLC supported at present
;ENDINTERFACE
;
; ================================================================
; Port definitions. Each one begins with PORT=n and ends with ENDPORT
; ================================================================
;
;       The number following PORT= is the port number as displayed by
;       the P[orts] command. They do not need to be sequential, you
;       may use any number, but you must define them in the order in
;       which they are to appear in the list.
;
PORT=1
;
;       Text string to identify port on "PORTS" display
;
        ID=Link to KIDDER
;
;       Interfacenum specifies which interface this port should use.
;
        INTERFACENUM=1
;
;       The rest of these are optional, since there are defaults
;       built into the program.
;
;       PORTCALL=G8PZT-1        ; Additional L2 callsign for port
;       PORTALIAS=PZT1          ; Additional L2 alias for this port
;       PORTALIAS2=RELAY        ; Yet another alias, for APRS
        CHANNEL=A               ; Channel to use on interface (A - P)
;       PACLEN=160              ; Overrides global paclen for this port
;                       ; If set to 0, paclen will be adaptive.
        MAXFRAME=2              ; Only 1-7 at present.
;       TXDELAY=300             ; Tx keyup delay (millisecs)
;       TXTAIL=100              ; TX keydown delay (millisecs)
;                       ; Don't go less than 100 for SCC cards!
;       SLOTTIME=100            ; (millisecs)
;
;       I strongly recommend frack=7000 for 1200 bauds. (see manual)
;
        FRACK=7000              ; L2 T1 time (millisecs)
;
;       Resptime should be *at least* ((paclen * 10000) / RFbauds) millisecs,
;       where "paclen" is the other end's paclen, otherwise you will send
;       unnecessary poll frames. 1500 is OK for 1200 bauds with paclen=120
;
        RESPTIME=1500           ; L2 delayed ack T2 (millisecs)
;
;       Persist should be set to (255 / (no. of users on frequency)).
;       e.g. for a frequency with an average of 10 users on at any one
```

```
;       time you'd set it to 25. Default is 64.
;
        PERSIST=64              ; Probability to transmit (0-255)
;
;       I can't see any point in setting retries more than 10, other than
;       for test purposes.  If you need so many retries it's a useless
;       link and you're just wasting everyone else's airtime.  The higher
;       you set this value, the longer users will have to wait to get
;       a "failure with" for a non-contactable destination.
;
        RETRIES=10
;
;       If you set fulldup=1, the router will transmit whenever it needs
;       to, without waiting for the other end to stop.  Used only for
;       hardware which is capable of simultaneous transmission and
;       reception, such as full duplex radio or wire links.
;
        FULLDUP=0               ; 1 = Full duplex, 0 = simplex/half duplex
;
;       Softdcd is used only by SCC cards.  If set to non-zero, the real
;       dcd will be ignored, and the driver will use the presence of
;       hdlc data as a DCD indication.  Using SOFTDCD with an open squelch
;       generates a *huge* interrupt loading on the PC, which may cause
;       degradation of performance, depending on the PC type, so I don't
;       recommend it.
;
;       SOFTDCD=0
;
;       Rfbauds defaults to 1200 if not specified.  It is intended for use
;       with "real" tnc's attached via RS232, because the RF baud rate is
;       usually different to the serial baud rate.  It simply helps the
;       router make timing decisions.
;
;       RFBAUDS=2400
;
;       L3ONLY=0                ; 1 = Ban L2 downlinks on this port.
;
;       Enable/disable the MHEARD function on this port. The number
;       specifies how many callsigns to maintain in the list. Set to 0
;       to disable MHEARD.
;
        MHEARD=10               ; Mheard enabled, 10 calls
;
;       MHFLAGS controls which callsigns are recorded in the MH list,
;       and defaults to 255 (show everything).
;       The number is formed by adding the following values:
;
;           1       Show directly heard stations
;           2       Show directly heard digipeaters
;           4       Show digipeated stations
;
;       MHFLAGS=1       ; show directly heard stations only
;
;       Digipeat flag for this port.  0 = no digipeat.  Default=7
```

```
;       Add together:
;
;               1       Digipeat UI frames
;               2       Digipeat non-UI frames
;               4       Enable RELAY generic APRS digipeating
;               8       Enable TRACE and TRACEn-N APRS digipeating.
;               16      Enable WIDE and WIDEn-N (Well sited stations only!)
;               32      Enable L4 APRS tunnelling
;               64      Enable frames from RF to Igate
;               128     Enable frames from Igate to RF
;
        DIGIFLAG=7              ; Normal digi + RELAY.
;
;       DIGIPORT=0              ; Port to relay digipeated frames on
;                               ; (0=this port)
;
;       List of destinations for "broadcasting".
;       Received non-digipeater UI frames, addressed to one of these
;       destinations, will be re-broadcasted on all ports which have a
;       matching address in their BCAST list.  This would for example be
;       used to broadcast mail beacons from a BBS onto several frequencies.
;
;       BCAST=MAIL,ALL
;
;       List of approved broadcasters.
;       If you wish to restrict the broadcast facility to certain senders
;       only, list the callsigns here.  If no calls are specified, the
;       facility is unrestricted.  Separate the calls by commas, and don't
;       include any spaces in the list.
;
;       BCFROM=GB7PZT,GB7MAX
;
;       Default quality for nodes whose broadcasts are received on this
;       port. Set to 0 to disable all L3/4 activity on this port.
;
        QUALITY=10
;
;       Minimum quality to add to node table for nodes received via this
;       port.  If specified, this overrides the global minqual, and can
;       be used to exclude unreachable and marginal nodes.
;
;       MINQUAL=10
;
;       Port to transmit on. 0=this port.  You would typically use this
;       where a single transmitter is used in conjunction with several
;       receivers
;
;       TXPORT=0
;
;       Interlock is only used by SCC cards - KISS TNC's make their own
;       decisions when to transmit, and the router has no control over
;       that process.  If a non-zero value is specified, no two ports
;       with the same value will transmit at the same time.
;
```

```
;       INTERLOCK=4
;
;       Maximum no. of simultaneous users on this port. Default is 255
;       which means "no limit".
;
        USERS=255
;
;       Sesslimit specifies the maximum simultaneous connects each user
;       is allowed.  You may wish to set a low value if you are a
;       control freak.
;
;       SESSLIMIT=5
;
;       The following two commands are mutually exclusive... Use one or
;       the other, but not both!
;       Callsigns should be separated by commas or spaces, and there is no
;       limit to the number of calls. You can have multiple validcalls= or
;       exclude= lines, but you should be aware that, depending on your
;       hardware, there may be a performance penalty if you use long lists.
;
;       Validcalls allows L2 frames only from the specified users, and is
;       typically used to keep users from connecting to link-only ports.
;
;       VALIDCALLS=G6YAK,G6AMU  ; Accept only these users
;
;       Exclude allows L2 frames from everyone EXCEPT specified users,
;       and would typically be used on a user-access port to prevent
;       connections from trouble-makers and pirates.  Just because I
;       provide this command it doesn't mean I condone its use.
;
;       EXCLUDE=NOCALL,P1RAT     ; Ignore these users
;
        CWID=G8PZT              ; Used only by SCC cards.
                                ; Callsign is sent every 30 mins.
;
;       PIPE allows frames received (and optionally sent) on this port to
;       be copied to another port, e.g. to allow a PMS on one port to see
;       the traffic on another port.  Unless the "bi-directional" option
;       is specified, pipes are one way.  In order to have two way traffic
;       using a uni-directional pipe, you must set up a reverse pipe on the
;       opposite port.  You may pipe several ports to a single destination
;       port, but you can only have one *outgoing* pipe from any port.
;       Pipes are capable of generating an immense amount of traffic, so
;       use them with care - your target port MUST be capable of handling
;       the traffic load.
;       Pipes can be made "selective", by adding a comma-delimited callsign
;       list, e.g. "PIPE=4 GB7PZT,KDRBBS".  This will reduce the loading on
;       the target port, by piping only the frames with the specified calls
;       in the destination field.
;       Pipes can be made "bi-directional" by adding 512 to the PIPEFLAG
;       value (see below: suggested value = 515).  If a frame is piped on
;       a bi-directional pipe, the source call is remembered so that responses
;       will be piped back to the sender.  Thus a reverse pipe is not needed.
;       This is useful in cases where a BBS has a front end router - simply
```

```
;       set up bi-directional selective pipes from each user port to the BBS
;       port, and set up the BCAST option so that the UI mail headers are
;       broadcast on each user port.  The BBS will then allow direct
;       connect and will respond to resync requests.
;       To disable piping, set PIPE=0 or just omit the command.
;
;       PIPE=2              ; Pipe frames from this port to port 2
;
;       PIPEFLAG is only used when piping is active, and controls which
;       frames are piped.  The default if not specified is 3.  The value
;       is made up by adding together the following numbers:
;
;            1      - UI frames *not* addressed to nodecall/alias.
;            2      - Non-UI frames *not* addressed to nodecall/alias.
;            4      - UI frames addressed to nodecall/alias.
;            8      - Non-UI frames addressed to nodecall/alias.
;            16     - UI frames transmitted by us.
;            32     - Non-UI frames transmitted by us.
;            64     - Allow budlisted users to be piped.
;            128    - Netrom frames
;            256    - IP / ARP frames
;            512    - Bi-directional piping
;
;       PIPEFLAG=5          ; Pipe all rcvd UI frames only
;
;       UNPROTO=CQ,G6YAK
;
;       Optional alternative IP address for use on this port.  If this
;       is specified, it will be used instead of the global IP address
;       for this port only.
;
;       IPADDRESS=44.131.91.5
;
;       The DHCP keyword specifies whether or not the port IP address
;       will be obtained dynamically using DHCP (DHCP=1) or specified
;       statically (DHCP=0).  Default is 0.
;
;       DHCP=0
;
;       If you set SYSOP=1, all users who connect on this port will get
;       full sysop status without answering a password challenge.  This
;       is intended ONLY FOR USE ON SECURE LINKS, such as RS232 or
;       Ethernet.  Be aware that, if the remote system is capable of
;       gatewaying or digipeating, users could downlink via the remote
;       system back into this port, thus gaining sysop status.  The default
;       for this parameter is of course zero!!
;
;       SYSOP=0
;
;       The APPLMASK parameter is used only if you are running applications
;       via the host interface.  It specifies which applications will be
;       directly connectible on this port.  Default is 255, which allows
;       all applications.  The value is made up by adding together the
;       following numbers:
```

```
;
;           1      - Enable Application 1
;           2      - Enable Application 2
;           4      - Enable Application 3
;           8      - Enable Application 4
;           16     - Enable Application 5
;           32     - Enable Application 6
;           64     - Enable Application 7
;           128    - Enable Application 8
;
;       If you want an application to be directly connectible on a port,
;       it must have a callsign, an alias or both, and the corresponding
;       bit in that port's applmask must be set.
;
;       APPLMASK=3
;
;       Optional port-specific ID text, sent every IDINTERVAL in place of
;       the global IDTEXT (e.g. for APRS-only ports).  Only one line may
;       be sent.
;
;       IDTEXT=!5224.00N/00215.00W# (Kidder)
;
;       CFLAGS allows level 2 uplinking and/or downlinking to be prevented,
;       e.g. on digipeat-only ports. Use VERY carefully! Default is 3.
;       Sysop can always downlink.  Add together:
;
;           1      Allow incoming connections (uplinks)
;           2      Allow outgoing connections (downlinks)
;
;       CFLAGS=3
;
;       Remote NET/ROM systems to whom we will tunnel L2 frames.
;       (See manual or PROXY.TXT for full explanation)
;
;       PROXY=GB7PZT,GB7BBS
;
;       Modem Initialisation string (Modem interfaces only)
;
;       INITSTR=ATM0
;
ENDPORT
;
; ------------------------------------------------
PORT=2
        ID=Link to BRUM
        INTERFACENUM=2
        CHANNEL=M
        FRACK=7000
        RESPTIME=1500
        MHEARD=10
        QUALITY=100
        TXDELAY=500     ; Synthesiser is slow to lock!
ENDPORT
;
```

```
; ------------------------------------------------
PORT=3
      ID=Internal Loopback
      INTERFACENUM=3
ENDPORT
;
; ------------------------------------------------
;PORT=4
;      ID=External loopback
;      INTERFACENUM=4
;ENDPORT
;
; ------------------------------------------------
;      Example AXIP (AX25 over IP wormhole) port.
;      You need one of these for each axip link
;      At least ID, INTERFACENUM, and IPLINK must be specified.
;      The IPLINK address is the remote host's IP address or hostname.
;      Parameters such as TXDELAY, TXTAIL, SLOTTIME, PERSIST, FULLDUP,
;      SOFTDCD etc. are meaningless for AXIP, but FRACK, RESPTIME, PACLEN,
;      MAXFRAME, QUALITY etc. operate as normal.
;
;
;PORT=5
;      ID=AXIP link with WA3DXX
;      INTERFACENUM=7
;      IPLINK=44.73.88.69
;ENDPORT
;
; ----------------------------------------
;      Example Ethernet port, supporting IP, ARP and AX25.
;      Note the reduced timings (only used for ax25 connections), as
;      the defaults are a bit too relaxed for ethernet.
;
;PORT=6
;      ID=Ethernet LAN
;      INTERFACENUM=8
;      CHANNEL=A             ; Ignored
;      FRACK=1000
;      RESPTIME=200
;      MAXFRAME=7
;      PACLEN=240
;ENDPORT
;
; -------------------------------------
;      Example APRS-only port.
;      Note the use of an alternate IDTEXT, and the use of CFLAGS to
;      disable connected mode operations, thus MAXFRAME,FRACK,PACLEN
;      etc. are not needed.
;
;PORT=7
;      ID=144.800 MHz 1200 baud APRS
;      INTERFACENUM=2
;      CHANNEL=B
;      CFLAGS=0              ; Prevent up/downlinks on this port
```

```
;       DIGIFLAG=5              ; Digi only UI frames addressed via RELAY
;       MHEARD=22               ; Nice big MH list
;       IDTEXT=!5224.00N/00215.00W#PHG3021 Kidderminster APRS digi
;
;       Override the default destination & path for ID beacons
;
;       IDPATH=APRS,RELAY,WIDE
;
;       Default digipeater path for APRS frames originated by APRS
;       messaging shell and Igate. If you omit this, the frames will
;       be sent without any digipeaters. Messaging shell users may
;       override this path.
;
;       APRSPATH=RELAY
;
;ENDPORT
;
; ------------------------------------------------
;       Example AXUDP (AX25 over UDP wormhole) port.
;       You need one of these for each axudp link
;       At least ID, INTERFACENUM, and IPLINK must be specified.
;       The IPLINK address is the remote host's IP address or hostname.
;       UDPLOCAL and UDPREMOTE are the UDP port numbers for each end of
;       the link, and if omitted they default to 93
;       Parameters such as TXDELAY, TXTAIL, SLOTTIME, PERSIST, FULLDUP,
;       SOFTDCD etc. are meaningless for AXUDP, but FRACK, RESPTIME, PACLEN,
;       MAXFRAME, QUALITY etc. operate as normal.
;
;PORT=8
;       ID=AXUDP link with VK1UDP
;       INTERFACENUM=9
;       IPADDRESS=44.131.91.77
;       UDPLOCAL=93
;       IPLINK=44.69.88.73
;       UDPREMOTE=93
;ENDPORT
;
; ================================================================
;       Applications. Each must begin with APPL= and end with ENDAPPL
;       Application number must be between 1 and 8.
;       Most BBS/PMS applications have to use appl=1, and HOST (e.g. PAC4,
;       TERM4) must use APPL=2
;       Each field is optional - you may have an applname without a call
APPL=1
        APPLNAME=PBBS
        APPLCALL=G8PZT-3
        APPLALIAS=PZTBBS
        APPLQUAL=50             ; Netrom quality to broadcast
ENDAPPL
;
;       In this example, the application has no callsigns or quality, so
;       it can only be reached by issuing the command "HOST" during a
;       command session.
APPL=2
```

```
      APPLNAME=HOST
ENDAPPL
;
; ======================================================================
ROUTES
;
;       Routes to lock in - Syntax is as follows:
;
;       <callsign> <port> <quality> [! [maxframe [frack [paclen]]]]
;
;       You must specify at least Callsign, port and quality.  If you
;       include the lock flag ( ! ) the route will be locked in, and
;       will only be changed by a replacement entry with the lock flag
;       set. If you don't include the lock flag, the route will
;       eventually expire if not confirmed by the reception of nodes
;       broadcasts.  In either case, if the file PZTNODES is present,
;       its contents will override the entries here, subject to the
;       locking rules above, and the sysop may also edit routes while
;       the router is running.
;       Maxframe, frack and paclen are optional.  If specified they
;       override port defaults for that route.
;       Note FRACK is expressed in millisecs, e.g. 7000 = 7 secs.
;       Maxframe > 7 will cause Modulo-128 to be attempted on that route.
;
;       End the section with *** as usual.
;
; Lock in a link to g6yak on port 1 with quality 100 and maxframe 32 :-
;
g6yak 1 100 ! 32
***
;
;       Sysop-defined commands:
;
;       Up to 8 commands can be defined, allowing the sysop to set up
;       single-word aliases for frequently used command strings.  For
;       example you might wish to set up BBS, CONV and DXCLUSTER to
;       point to local systems.
;
;       Each command is defined using a separate "COMMAND=" string,
;       and the arguments are <alias> <real_cmd>.
;       e.g. "COMMAND=BBS C 7 GB7PZT" means "create a new command
;       called BBS, which translates to the sequence C 7 GB7PZT"
;
COMMAND=BBS C 1 GB7PZT
COMMAND=CONV TELNET 44.131.90.1 3600
COMMAND=DXCLUSTER C GB7DXC
;
```