JNOS40

CONFIGURATION MANUAL

(NOS for the Kantronics Data Engine(tm))

(Document ID: 40_CFG02)

JNOS40 Program by

Johan K. Reinalda, WG7J

Release 1.00
February 28, 1994

(based in part on the NOS Reference Manual,
by Phil Karn, KA9Q
and
Gerard van der Grinten, PA0GRI)

DISCLAIMER

The authors make no guarantees, explicit or implied, about the functionality or any other aspect of the programs described herein.

This manual is applicable to JNOS40 (JNOS for the Data Engine) Version 1.00

JNOS40 is an adaptation of Phil Karn's, KA9Q, NOS.EXE program. It runs standalone on a Kantronics Data Engine and is intended for use as a stand-alone, hilltop packet switch. JNOS40 provides TCP/IP, NET/ROM and AX.25 switch and routing functions as well as Converse and Domain Name Servers. JNOS40 can act simultaneously as a server and a packet switch for all three sets of protocols because it has an internal multitasking operating system. That is, while a remote user accesses the switch (i.e. 'the node'), the system can also switch IP, NET/ROM and AX.25 packets and frames between other users and remote systems.

The user interface for network operation of JNOS40 is very similar to the well-known G8BPQ, TheNet and NetRom-based nodes. The user interface in the JNOS.EXE program is commonly called the 'mailbox'. Since JNOS40 doesn't provide any mail services, the terms 'node shell' or simply 'node' seem more appropriate. All three terms are used interchangeably throughout this document.

This document is not a beginner's guide to tcp/ip. Experience with PC, Atari, Macintosh or Unix-based versions of the KA9Q NOS program, or with other tcp/ip systems will help much in getting things setup and configured correctly. We attempted to make this manual easy to follow by taking things in small steps and providing lots of examples.

See the ADDENDUM file for the 'Intronos' document written by John Ackerman, AG9V, which offers a good tutorial on NOS and tcp/ip. It is included with the author's permission. All credit goes to John for his work! Also included is the Frequently Asked Questions list that the tcp-development group maintains. You can contact the group on Internet via the tcp-group@ucsd.edu mail-list. These documents have a lot of useful information about Amateur Radio tcp/ip.

These are some of the abbreviations and terms used throughout this manual.

HOSTNAME is the tcp/ip name of a computer or packet systems

INTERNET is a world wide high speed computer network. It has thousand of computers at schools, companies and amateur packet radio systems connected to it.

MTU, or Maximum Transmission Unit, is the maximum data size in one packet. Most often the data referred to with the mtu is the transported data, i.e. data in a network connection. With tcp/ip, the size of the tcp/ip frame inside the ax.25 packet is the mtu; with net/rom, the size of the data inside the netrom packet is the mtu.

NRS, or Net/Rom Serial protocol, is what TNCs with Net/Rom or TheNet eproms talk on the serial port.

NODE, NODESHELL, MAILBOX are terms used interchangeably for the user interface when connected to the node.

PACLEN, or packet length, is most often used to refer to data size in a link packet. The data in an ax.25 packet can be up to paclen bytes.

PORT or INTERFACE means the physical connection to a radio or other system (i.e. radio port or serial interface). The two terms are used interchangeably

RFCs, or Requests For Comment, are standard papers used by the Internet Engineering Task Force (IETF) to discuss and propose new networking protocols and other related topics.

RSPF, or Radio Shortest Path First, is a tcp/ip routing protocol especially targetted at radio environments.

RTT, or Round Trip Time, indicates the time needed for data to be sent and acknowledged.

SLIP, or Serial Line IP, is a way to send IP frames over a serial port without using ax.25 or ethernet to carry the data. You can use SLIP to connect to PC's or Unix systems also running SLIP and exchange tcp/ip data.

```
*****************************************************************
*
* If you use JNOS40, I would appreciate if you drop me a note with
*
* your thoughts and suggestions.  You may use Internet, packet, or
*
* postal service.    Thanks in advance,   Johan
*
*****************************************************************
*
```

Questions, remarks and suggestions about JNOS and JNOS40 are welcome and should be sent to:

Johan Reinalda, WG7J/PA3DIS
420 NW 9th
Corvallis, OR 97330
U.S.A.

email: johan@ece.orst.edu
(or the sloooower WG7J@WG7J.OR.USA.NA via packet)

or for the documentation only:

Doug Thompson, WG0B
PO Box 21108
Wichita, KS 67208-7108

email: wg0b@delphi.com
or packet wg0b@k0hyd.#scks.ks.usa.na

Corrections (and comments) to the documentation must include the following information:

1) Document ID (See the Title Page)

2) Page Number

3) Text as it exists

This does not have to be the complete text. But it must be enough to ensure unambiguous identification of the area under discussion.

4) Text as it is proposed to be or an explanation of the problem which I will convert into appropriate text.

DO NOT send a copy of the whole document with revisions scattered throughout. I have neither the time nor the inclination to wade through that much text. - wg0b

Send the corrections to WG0B at one of the addresses on the preceding page. If it comes to the PO Box, please send it on floppy disk, IBM format, 1.44 MB or less.
The documents have been prepared using Microsoft Word Version 5.0. Submittals using MS Word 4.0 or 5.0 format, plain ASCII text, or Rich Text Format (RTF) (supported by WordPerfect) are all easily handled

JNOS40 is distributed in a zipped archive, containing:

```
     nos40lo.bin    - low eprom image
     nos40hi.bin    - high eprom image
     40_cfg02.txt   - this document in plain ASCII text format
     jn_cmd02.txt   - the JNOS/JNOS40 Commands Manual in ASCII
                      text
     addendum.txt   - FAQ, "IntroNOS", and other useful info
     cfg.exe        - the eprom configuration program
     check.exe      - the autoexec.nos checking program
     autoexec.nos   - a sample configuration file
     domain.txt     - a sample domain configuration file
     history        - revision and history file
kiss.zip       - various kiss eproms for tnc2 and compatibles
```

Additionally, the document files are separately available in a zipped archive containing:

```
     40_cfg02.doc   - this document in MS-Word 5.0 format
     jn_cmd02.doc   - JNOS/JNOS40 Commands Manual in MS Word
                      format
```

Hardware Requirements

To run JNOS40 you will need the following:

A Kantronics Data Engine (This is a surpise?)
  Optional - One or two internal modems for the Data Engine
  (You could run all comm through TNCs attached to the serial
   port, but that's sure wasteful.)

2 128kx8 (1 mbit) EPROMs (27c010, 27c1001, 27c101...)
     150 nsec speed or faster is recommended.

(Optional) 1 or 2 128Kx8 Static RAM ICs (See text below)

You will also require access to the following equipment to prepare the eproms for the data engine:

Eprom programmer
PC-compatible computer to run the CFG.EXE and CHECK.EXE programs.

(See the file 'HISTORY' for a complete account of the evolution of different versions.)

JNOS40 Version 1.00

NETROM Node with "Stay Here" capability

IP Router

IP Domain Name Server with domain.txt file in ROM
Additional entries cached in RAM

Accepts multiple TELNET, AX.25, and NETROM connections in any combination

Converse (Conference) Server
- Supports linked conferences

IP Finger server

RIP (Routing Internet Protocol) and RSPF (Radio Shortest Path First) path determination modes are supported.

ARP (Address Resolution Protocol) mapping supported

Serial Port supports the following protocols:
RS-232 (console mode)
KISS
G8BPQ-style Polled KISS
WG7J Bus Contention KISS
NETROM Serial (NRS) protocol
SLIP

Four different menu response modes
- No prompt or Netrom ID
- No Menu (Netrom node ID only (NRID))
- Single Line Menu (NRID plus first letter of available commands)
- Full Menu (NRID plus full word list of available commands.)

Comprehensive on-line help

Automatic user-to-sysop TTYLINK using Sysop command when enabled

ALL configuration commands are available to remote sysop when properly authenticated via any connection type (tcp/ip, netrom, or ax.25)

Hidden 'memory' and 'links' commands allow checking these

performance parameters from the nodeshell instead of requiring sysop mode to be entered.

Capability to ADD configuration commands which will be stored in bbram and executed in the event of power loss or sysop commanded warm restart. These commands may either supplement or modify commands contained in eprom.

All configuration parameters may be set from autoexec.nos instead of having to use the "advanced setup" function in CFG.EXE. This makes setup for JNOS40 functionally the same as for JNOS.

CHECK.EXE allows the autoexec.nos file to be checked and 'what-if' games to be played before burning eproms.

CFG.EXE automatically calls CHECK.EXE for syntax and parameter checking while preparing data for the eproms. Both of these programs run on PC compatible (x86) computers only.

NOTE: Many operators have reported problems getting the Data Engine to start up when new eproms are installed. This problem can be corrected by removing battery jumper J5 while initially booting with the new eproms. Re-install the jumper once the Data Engine is operating.

Even though the Data Engine may start up with new eproms without having removed J5, data saved in battery-backed ram (bbram) may produce surprise(!) configurations. Removing jumper J5 allows the saved data to be dumped so that the new start will use only the configuration data in eprom.

NOTE: Both CHECK.EXE and CFG.EXE are version specific. You must use the copies of these programs which are distributed with a particular version of JNOS40. If you attempt to use incorrect versions, an error message will be displayed and program execution will stop.

a) AUX switch setting for running the program

To run the JNOS40 node program the AUX switch on the front panel of the Data Engine needs to be OUT (unlike the commonly used G8BPQ software where the AUX switch needs to be IN.) Depress the AUX switch to use the serial port for the console.

b) Data Engine Hardware Ports and NOS servers

This release supports the two internal radio ports, the serial port, and most of the servers available in the NOS.EXE program.

RADIO PORT A has been tested with a type A modem (e.g. DE1200), with both simplex and duplex. A type B modem (e.g. DE9k6/19k2) has been tested with full duplex audio-loop back, as well as simplex with a pair of D4-10s. A simple Type D loop back modem has also been tested, up to 57600Bd. Port A is always DMA driven for optimal performance. Code for type C modems is present BUT HAS NOT BEEN TESTED due to the lack of such hardware.

RADIO PORT B has been tested with a type A and B modems in half-duplex only. Type C and D drivers are present, but again NOT TESTED. Port B currently only supports simplex.

The SERIAL PORT can operate either as a console port or a serial network interface. The serial port data rate should be limited to 19200 baud or less. A problem of interrupt latency is being worked on in order to provide higher serial port data rates.

The network interface mode currently supports **SLIP**, **NRS**, and **KISS**

(AX25) modes. Multi-port KISS is supported, i.e., one serial line to a TNC with 2 radio ports. Polled KISS support (as in G8BPQ's Multidrop KISS) is now available. 'Bus contention kiss', a new scheme to attach more than one tnc in kiss mode to the serial port is also supported. TNC2 compatible eprom images and other info are in the file **kiss.zip**. The serial port can use hardware handshaking via the CTS and DTR lines for any connection scheme. See the section "Hardware Handshaking on the Serial Port" for more.

Servers supported are:

```
telnet  (started automatically)
netrom    "    "
ax25 "    "
finger    "    "
remote    "    "
```

Other services available are:

```
convers   Start from autoexec.nos, console, or remote
rip       Start from autoexec.nos, console, or remote
rspf2.0   (untested!!)
trace     (only when in console mode)
```

c) Data Engine Memory

A standard Data Engine contains two 32K RAM chips. This minimum configuration will work and results in around 25k RAM available. However, it is STRONGLY advised you upgrade to more memory by replacing one or both of the RAM chips with 128kb Static RAM chips (120ns or faster). You may install any combination of 32K and 128K Static RAM chips in the Data Engine for use with JNOS40. There are no restrictions on what size IC is installed in which socket, BUT BOTH SOCKETS need to have an IC installed.
JNOS40 will automatically sense which memory configuration is present at startup and does not require any entries in **autoexec.nos**.

d) Memory addresses:

Because all allocated memory in NOS has an offset of 8 (i.e. it shows like 23ef0008), all displays have been modified to simply show the segment of the memory location, i.e., the above location will show as 23ef. Kicking and resetting also only need the segment descriptor, e.g. 'tcp reset 23ef' .

e) Battery Backup.

Many configuration parameters are kept in battery-backed ram (BBRAM) and will be maintained across power outages or warm restarts UNLESS

they are re-set in the eprom startup configuration. These parameters as well as configuration data entered with the **'add'** command are stored in BBRAM and are protected against corruption using a 16-bit CRC (variables and config-data have separate CRCs...) If the data becomes corrupt, the next restart will load original values from ROM or blank out the added config-data area, or both. Data can be changed using console mode or as remote sysop, and will be maintained across power outages as long as the battery is okay and no corruption occurs. The bbram variables are marked with (B) in the JNOS/JNOS40 Commands Manual.

f) Passwords:

Both the node sysop password and remote server password default to '0123456789' unless changed in the configuration file. Password length is limited to 30 characters.

g) Console Mode Connects

JNOS40 does not support connections initiated from the console while in Console Mode.

The following steps are the absolute minimum to prepare a Data Engine for use with JNOS40:

1) Edit the autoexec.nos file
2) Run CFG.EXE to prepare the eprom images
3) Burn the eprom images into two 1-mbit eproms
4) Install the eproms in the data engine.
5) (Optional) Install one or two 128kx8 bit static RAM chips
6) (Optional) Install the internal modems
7) Apply power

Other steps which may be included plus detailed instruction for each step are in the sections which follow.

The following sections will guide you through the steps needed to produce a minimal configuration that will allow you to get going. Then you can start experimenting with some of the more 'exotic' things related to NOS and tcp/ip.
All commands shown in the following sections, and all other commands available, are described in the "JNOS and JNOS40 COMMANDS Manual". Please refer to that document for detail about the commands used.
All examples shown are just that: examples. They are by no means the only way to set the parameters. They merely serve as a place to get you started. Settings will often depend on the systems around you, and on the parameters preferred by the packet network operators in your area. Please contact your local packet organization for the guidelines and parameter settings preferred in your area packet network !

Interface Buffers

There are two different types of buffers associated with attaching interfaces.
The first type is the ring buffer or fifo (first in, first out) that is used when attaching the serial port. It is used for receiving only. This buffer is allocated just once, and is used throughout the life of the interface. The asynchronous (or serial port) receiver interrupt code puts characters in this buffer in a circular fashion. When the end of the buffer is reached, the next character is stored at the beginning and continues through the buffer again.
The receiver process for the serial interface reads the characters from this fifo-buffer into memory buffers, or mbufs, that are used internally to handle and pass around data. Setting the size of the fifo buffer is an empirical process. A good place to start is to set it to twice the size of the packet length used over the serial port. Once you have the interface running, you can monitor the usage of the fifo buffer with the 'asy' command. This will show you

the 'buf hi' value which is the same as 'sw hi' for PC based NOS. 'buf hi' is the highest value of the number of characters that were waiting in the fifo buffer to be read by the receiver process. If 'buf hi' is close to the fifo buffer size, or if the 'asy' command shows buffer overflows ('buf over' for JNOS40 code), you should increase the buffer size. If however the number is significantly smaller, you could decrease the buffer size.
The second type of interface buffer is also a receiver buffer. However, this type of buffer is allocated, then released as needed. This buffer almost always is allocated during interrupt service routines, i.e., when the interrupts are off! In order to keep the service routine short, the buffer is allocated from a special 'interrupt buffer queue' because a regular memory allocation would take far too long.
The two internal modem drivers use this second type of buffer. Since one interrupt buffer pool services all the drivers that need buffers during interrupt, the size and number of these buffers are quite critical to a system's 'well-being'. A buffer acquired from the interrupt buffer pool needs to be large enough to handle the largest packet received from any of the internal modem interfaces. A good rule to estimate the **size** of the interrupt buffers needed is as follows:
Set '**memory ibufsize**' to the (largest + 'extra') of:

1 - the **largest ax.25 paclen** parameter of the internal modem interfaces, OR

2 - the **largest ip mtu parameter** of the internal modem interfaces.

PLUS 'extra' which accounts for the longest possible ax.25 header (source, destination, control and digipeaters). Use 80 bytes for' extra' as a general rule.
The MTU is important because on ax.25 interfaces where the MTU is larger than the paclen, there is a possibility of receiving larger ip frames when IP traffic is carried in Datagram mode. This possibility does not exist if IP traffic is carried in Virtual Connect (VC) mode, since the data will be split in several paclen-sized ax.25 packets. (This is AX.25 V2.1 fragmentation at work for you!) When using VC mode you can ignore the MTU values when finding the ibuf size.
The above discussion assumes that paclen and mtu values are coordinated between all users in your area packet network. If this is not the case, someone else might send packets larger then what your system can handle. Such packets will cause receive buffer overflow and will be dumped!
If you have an ax.25 interface with paclen of 256, mtu of 256, and another with paclen of 256 and mtu of 512, you should set the 'memory ibufsize' to at least 512 + 80 !
JNOS40 ibufs default to 600 bytes, and should suffice for paclen's

or mtu's up to 512. NOS.EXE has a default ibufsize of 2k (ie 2048 bytes), wich suffices for the standard Ethernet MTU of 1500 bytes. Determining the **number** of buffers needed is another empirical process. Start with, say, ten buffers (ie 'memory nibufs 10'). If you get a lot of memory ibuffails in the 'mem stat' display, you should increase the number of buffers.
NOTE: if you are not using one or more of the internal modems or any drivers that require interrupt buffers, there is no need to keep them around. In that case, simply set 'mem nibuf 0'.

There are two types of attach commands. The first type attaches physical interfaces or ports. These commands identify the serial port and the two internal radio ports to the system. The JNOS40 hardware attach commands are 'attach 1', 'attach 2', and 'attach 3'. The second type attaches 'pseudo' interfaces, that is, interfaces which do not directly relate to physical hardware ports. These commands are 'attach axip', 'attach kiss', and 'attach netrom'

Data Engine Ports

In the JNOS40 attach command, the ports in the Data Engine are named (numbered) 1, 2, and 3.
1 is the serial port,
2 is internal port A, and
3 is internal port B.

You may still give a mnemonic name to each port; the numbers are equivalent to the port addresses in the JNOS attach commands so that the program knows which port you want to use.

To attach the serial port, the syntax is:

'attach 1 <mode> <name> <buffer> <mtu> <speed> [c]'

where:

mode - is one of 'ax25','slip' or 'nrs', or 'pkiss'
name - is the interface name, e.g., 'port1'or '2m'
buffer - is the receive buffer size
(see INTERFACE BUFFERS for more)
mtu - is the maximum transmission unit (which for an AX.25 V2 interface should be 256.)
speed - any common speed from 300 - 9600 Bd.
(19200 and higher should not be used. See text)
[c] - enables RTS/CTS handshaking except in g8bpq-polled-kiss mode (see HARDWARE HANDSHAKING...)

There are four different **modes** currently supported and some of the modes support multiple configurations. As a result, you have several options when using the serial port. You can configure JNOS40 to attach one of the following:

- a single-port tnc in kiss mode

- a multi-port tnc in kiss mode

- multiple single-port TNCs in wg7j-kiss mode

- multiple single-port TNCs in g8bpq-polled-kiss mode

- one or more TNCs with Net/Rom or TheNet software using NRS protocol

- a slip connection to a tcp/ip host

All configurations for the serial port are listed below.

.

To attach a TNC to the serial port use the 'ax25' mode. This is also called KISS and means you have a TNC in kiss mode or with a kiss eprom in it connected to the serial port. To attach a tnc to the serial port in KISS mode use:

attach 1 <mode> <name> <buffer> <mtu> <speed> <c>

Example:
attach 1 ax25 port1 512 256 9600

See the section on '**Physical Connections / Single and Multiport TNCs**' for information on how to connect the tnc to the Data Engine.

.

Attaching the second port of a single tnc with multiple ports, i.e., a multi-port KISS tnc, like a KPC-4, is done with the command:

attach kiss <asy_iface_label> <port#> <label> [mtu]

You first need to have attached the Data Engine serial port in ax25 mode. The attached serial port will always default to the first

port (tnc port 0) in the multiport tnc. Next you can attach the second port (tnc port 1)in the dual port tnc with the 'attach kiss' command.

To attach both ports of a dualport tnc in kiss mode at 9600 baud:

attach 1 ax25 port1 512 256 9600
attach kiss port1 1 kiss2

In this example, the first port is addressed with the name 'port1', and the second port with the name 'kiss2'

WG7J's Kiss eproms allow multiple separate TNCs to be connected on one serial port. These TNCs signal each other when they need to send a frame to the host (the Data Engine) by raising the CTS line active. If the DTR line is active when a serial transmission needs to occur, the TNC will wait until the DTR line becomes inactive. This approach can be more efficient than polled TNCs, where time is spent polling TNCs that might not have any data.

To use the WG7JKISS eprom to have multiple TNCs in kiss mode on the serial port, configure the address for each eprom according to information in the KISSROMS.TXT file. Also, connect the TNCs together as described in the '**Physical Connections / Multiple TNCs with WG7JKISS roms**' section of this document. Then 'attach' the first TNC on the serial port as an ax25 type, and for each additional TNC attach a kiss interface.

EXAMPLE: If you have 3 TNCs, with addresses 0 (mandatory !), 10 , and 14, running at 9600 bd, with interface names 'port0', 'lan10',and ' users14', the attach statements would be:

attach 1 ax25 port0 512 256 9600
attach kiss port0 10 lan10
attach kiss port0 14 users14

This method allows multiple TNCs to be hooked up to the serial port. Each tnc will be polled for data at a regular interval. The 'c' option for handshaking is ignored in this mode.

To attach multiple TNCs on the serial port using the G8BPQ polled kiss scheme, configure each tnc according to the G8BPQ

documentation. To connect the TNCs to the Data Engine, refer to G8BPQ's KISSROMS.DOC documentation, or to the '**Physical Connections / Attaching TNCs with G8BPQ Polled Kiss roms**' section in this document. Then attach the first tnc on the serial port as a PKISS type, and for each additional tnc attach a kiss interface.

EXAMPLE: If you have 3 TNCs, with addresses 0 (mandatory!), 10 , and 14, running at 9600 bd, with interface names port0, lan10, users14, the attach statements would be

```
attach 1 pkiss port0 512 256 9600
attach kiss port0 10 lan10
attach kiss port0 14 users14
```

Note that only the first interface attached (which is the one that actually configures the serial port!) needs to be specified as a 'polled kiss' interface. The additional interfaces will be automatically attached in polled kiss mode.

.

slip - is serial line ip. It means you want to connect another Data Engine, or a PC or Unix(tm) system also running slip to the serial port.

```
attach 1 slip sl0 1025 768 9600
```

Consult the manual of your host PC or Unix(tm) system for proper physical connection information.

nrs - is Net/Rom Serial protocol. TNCs with Net/Rom or Thenet eproms use this protocol for serial port communication.

EXAMPLE: Attach the Data Engine serial port at 2400 baud as follows:

```
attach 1 nrs nrs1 512 236 2400
```

If you want to connect only one TNC with Net/Rom or TheNet software to the DE, see the section on '**Connecting Single and Multiport TNCs**'

If you want to connect the DE to two or more TNCs with Net/Rom

or TheNet code, see the section on **'Connecting to Multiple TNCs in NRS mode'**

.

NOTE: port B is simplex only !

The modem type installed in each internal port is automatically sensed when the port is 'attached'.

Syntax is:

'attach 2|3 name mtu speed [f][n]'

where:
name - is the interface name
mtu - is the maximum transmission unit.
MTU is tested against interrupt buffer size when the ports are attached. If MTU is too large, an error message will result and the 'attach' will be aborted. See below.
speed - is the radio speed.
[f][n] - are optional parameters. f indicates full duplex and n is the value written to the mode AUX pins. If both are used, f should lead n !(This is used in the Kantronics 1200Bd modem to choose the type of CD circuitry to be used.
n = 0 (DEFAULT) is sine wave detection as DCD
n = 1 is the signal from the 3105 modem chip as DCD
n = 2 is the external cd signal as DCD)

Examples:
attach 2 port2 256 1200 2
attach 3 port2 768 2400 f1
attach 2 port2 1024 9600 f

Modem types A,B and D have been tested by WG7J. Type D (the simple loop back for testing) is set to full duplex always, and can NOT (yet) be used in port B. Type C modems (TAPR K9NG, etc.) have not been tested, but might work.
Note: with the type B modem, the speed parameter is a "don't care" value since both rx and tx are externally clocked. The value for speed will show up in the 'ports' display though.

HINT: If you have a DE1200 modem that you might want to replace with a DE9600 modem in the future, and don't want to have to reburn eproms, or if you want to change the modems to opposite slots or whatever, consider the following:

With the type B modem (i.e. DE9600) the speed parameter is ignored. You may enter a value of 1200 which will not hurt anything while a Type B modem is used. Later you may plug in a DE1200, modem, type A will be sensed, and the speed correctly set to 1200Bd! The only problem with this approach is that optimal MTU for higher speeds is usually larger than values useable for 1200 Bd.

The TXTAIL is automatically set to 4 characters at the radio port speed which at 1200 Bd amounts to 26ms. If you attach the ports with the 1200bd set as shown above, the 'param <iface>' command will always show txtail to be 26ms, since this is calculated from the port speed given. However, if you plug in a 9600/19200 baud modem, this calculation will be in error. The actual txtail then is about 3ms or 1.5 ms, respectively.

If you always will be using DE1200 and DE9600 modems, you should attach them as:

```
attach 2 port2 256 1200
```

The MTU size for the two internal radio ports is checked against the interrupt buffer size when the ports are attached. If the MTU is too large, the 'attach' will be aborted and an error message will be displayed. CHECK.EXE can be used to test for acceptable values before attempting to compile the eprom images.
IBUFSIZE must be set to the appropriate value BEFORE the ports are attached in AUTOEXEC.NOS. The default value is ibufsize=600 which allows for an MTU of 512. See the appendices for additional information about the relationship of MTU, PACLEN, and interrupt buffer size.

Port Descriptions

You can give each port a short description that will be displayed when users type the **'P'** command at the node.

```
ifconfig port1 de "144.92 MHz local lan port"
ifconfig port2 de "223.42 MHz cluster port"
ifconfig port3 de "430MHz 19200Bd link to Eugene"
```

The second type of attach commands is for attaching servers rather than hardware ports. These commands have the same format as in the JNOS program. Throughout the rest of these sections, we will use

the names port1, port2 and port3 to indicate the serial port and the two internal ports, respectively.

The netrom interface is attached with the **'attach netrom'** command. This command should not be needed in autoexec.nos as it is normally executed by default when the netrom server is started.

To attach an AXIP tunnel, the format is

attach axip <name> <mtu> <ipaddress> <call>

If we have two systems capable of running axip,
'jnos1', call wg7j-1, ip 44.26.0.162, and the other
'jnos2', call k7uyx-1, ip 44.26.0.98, then:

On one end of the tunnel, jnos1 has the following:

attach axip tunnel1 256 44.26.0.98 wg7j-10

On the other end of the tunnel, jnos2 has the line:

attach axip tunnel2 256 44.26.0.162 k7yux-10

A user connected to jnos1 will now see a new port called 'tunnel1' in the 'P' command. Users connected to jnos2 see a new port called 'tunnel2'. You can set descriptions with the 'ifconfig <iface> description' command.

If ax.25 station ka7ehk wants to connect cross band to jnos1 via jnos2, ka7ehk needs to know the callsign of the tunnel interface to use. Here, this call is k7uyx-10. Thus the following should be sent:

connect wg7j-10 via k7uyx-10.

This is what goes around:

ka7ehk sends the connect attempt.

ka7ehk -> wg7j-10 v k7uyx-10

jnos2 receives this and decides the digi call is for the tunnel

interface.

jnos2 swaps calls to keep track of the return path, and digis to port 'tunnel2'

ka7ehk -> wg7j-10 v k7uyx-1*

jnos1 receives this, and replies with the connect acknowledge (via tunnel1)

wg7j-10 -> ka7ehk v k7uyx-1

jnos2 receives this, examines call, swaps and digis to the 'real' radio port

wg7j-10 -> ka7ehk v k7uyx-10*

ka7ehk receives this, and the connection is established. All further data exchange will follow the same route !

The autoexec.nos file is used to supply all of the "cold boot" information to the node. It is the contents of this file plus the optional 'domain.txt' that are burned into eprom to set the operational parameters of the node. In beta versions of JNOS40, many parameters could only be set using "Advanced Setup" in CFG.EXE. Now, all setup is normally done from autoexec.nos.
The "JNOS/JNOS40 Commands Manual" contains complete information about the commands used here.
There are some parameters that need to be configured before you attach interfaces to the system because these parameters are used when attaching interfaces.
First, you should set the tcp/ip **host name** of your system.

hostname switch.wg7j.ampr.org.

Next, set the system's **IP address**. If you have NOT had an IP address assigned, contact your area ip-address coordinator to get an assignment. The default "experimental" address should never be used for a network node address.

ip address 44.26.1.19

Set the system's **AX.25 callsign**,

ax25 mycall wg7j-1

If you want the system to be known by an alias, you can set it. The ax25 alias command is synonymous with the 'netrom alias' command, but does NOT activate netrom. (See the Commands Manual.)

ax25 alias jnos40

If you are using packets or MTUs larger than 512 byte paclen, you must change the interrupt buffer size. If you only use smaller packet sizes, you could decrease the buffer size to save memory. CHECK.EXE will produce an error message if the interrupt buffers are too small. See also the section on INTERFACE BUFFERS. The default values, which support 512 byte MTUs:

mem nibufs 10
mem ibufsize 600

You can change the default ax.25 paclen that will be used when attaching interfaces. You can also change the paclen for each interface afterward. To change the **default paclen** from 256 to another value, set

ax25 paclen <nnn>

In order to define routes, assign descriptions and set other functions the interfaces must be attached first. A detailed description of how to attach interfaces in JNOS40 is presented earlier.

attach 1 ax25 port1 512 1200 c
attach 2 port2 512 9600 f1
attach 3 port3 1024 9600 f

Setting up everything for AX.25 use can be fairly simple. You have already set the system's ax.25 callsign and alias.

Connections will be cut-off after a certain time of inactivity. Cutoff is controlled by the T4 timer. T4 default is 900 seconds.

To change time-out to 10 minutes (600 seconds)
ax25 t4 600

Digipeating is controlled per interface and defaults to OFF.

If you want to allow digipeating via certain interfaces, set:

ax25 digi port1 on|off
ax25 digi port2 on|off
ax25 digi port3 on|off

You might want to set some digipeater routes which is done with the 'ax25 routes' command. However, since you do this on a per interface basis, you need to have attached the interface first !

Example: IPNODE via k7uyx-2 on port 2

ax25 route add IPNODE port2 k7uyx-2

If you want to allow other than 10 retries, set:

ax25 retries n ; where n = 1 to 10 (5 is a good value)

You may want to set AX.25 id broadcasting. Read the FCC rules and do as you think is correct...

You need to set the broadcast interval, the broadcast text (if any), and you need to activate each interface you want to beacon on.

ax25 bcinterval 600
ax25 bctext "NOS for the Data Engine by Johan, WG7J"
ax25 bcport port1 on
ax25 bcport port2 on
ax25 bcport port3 on

In certain cases you might want different beacon text on each interface. This can be accomplished with the 'ifconfig <iface> bctext' command.

ifconfig port1 bctext "This is the bctext for port1"

In certain cases, you might want different ax.25 paclen for different interfaces. Paclen inititally defaults to the **'ax25 paclen'** value (which in turn defaults to 256), but can be changed with:

ifconfig port1 paclen 384
ifconfig port2 paclen 64
ifconfig port3 paclen 192

The default settings provided in JNOS40 should meet most requirements for initial startup; however your particular network might require other values or some experimentation may be in order to find the best set of parameters. Please contact your local packet organization for the guidelines and parameter settings preferred in your area packet network !

First, you have to make netrom available to the system. If you use the default eprom setting that starts the netrom server and attaches the netrom interface automatically,

attach netrom < - not needed normally

You also need to set an alias

netrom alias jnos40

Only if you want a netrom call different from your ax.25 call, use:

netrom mycall wg7j-5

You then need to state which ports you want to activate for netrom and give each port an appropriate quality

To disable verbose route broadcasts, add an optional n.

Activating the interface will automatically poll for routes on that interface.

netrom interface port1 244
netrom interface port2 192 n
netrom interface port3 191

It is a good idea to tell the other nodes that we just came up.

netrom bcnodes port1
netrom bcnodes port2
netrom bcnodes port3

You might want to adjust the minimum acceptable route quality a little higher

netrom minquality 150

Other default parameters will suffice for most installations. Please contact your local packet organization for the guidelines and parameter settings preferred in your area packet network!

Setting up tcp/ip can be as 'simple' as setting up a few routes. You might also want to adjust the MSS and other settings. See the section ON MSS, ... and the 'intronos' document (in APPENDICES) for more.
You will probably want to setup some permanent routes to the local ip subnet, and possibly other subnets. We have a local subnet in which all address start with 44.26.1.x, i.e. the subnet mask is 24 bits. It is located on port1, thus

route add 44.26.1/24 port1

We use local subnets of 8 bits size (i.e. 256 systems), where each gateway (or node) is the x.x.x.0 ip address. There are a few neighbor nodes running Data Engines, that have local subnets as well. (These could, of course, also be Thenet X1-J nodes.) Those nodes have routes for 44.26.2.x, 44.26.3.x, and 44.26.4.x and are all on the 220 cluster frequency (port 2)

route add 44.26.2/24 port2 44.26.2.0
route add 44.26.3/24 port2 44.26.3.0
route add 44.26.4/24 port2 44.26.4.0

The rest of the state goes south via the high speed link (port3)

route add 44.26/16 port3 44.26.5.0

North is the 44.116/16 subnet, which we have to reach via netrom. The NetRom to IP gateway in that area is W0RLI-3, with ip address 44.116.0.70. Thus we need to add an ip route, as well as an arp statement to tell the system what the netrom call is that goes with the gateway ip address.

route add 44.116/16 netrom 44.116.0.70
arp add 44.116.0.70 netrom W0RLI-3

Everything else goes to the WG7J Internet gateway (this is the default route):

route add default port1 44.26.1.16

The system can log and show IP activity. The node command 'IHeard' will list recently heard tcp/ip systems. The size of the list defaults to 8, but can be changed.

You need to enable the ip-heard facility for each port :

ip hsize 8
ip hport port1 on
ip hport port2 on
ip hport port3 on

Most other parameters should have reasonable default settings; however your particular network might require other values and/or some experimentation. Please contact your local packet organization for the guidelines and parameter settings preferred in your area packet network !

The conference facilities in JNOS40 can be accessed in three different ways described below. Each way can be turned on or off independently. Before you enable any of these methods, you should set the **convers host name** to an appropriate name with the 'convers host' command. If not, it will default to the string set with 'hostname'.
convers host Corvallis

CONFERENCE CALL ACCESS

In the first method, a user can do an ax25 connect to the conference call. This gives direct conference bridge access on the ports you have enabled. You can set this call with:

convers mycall qso

You can set a separate inactivity time-out for these ax.25 conference connections with the **'convers t4'** command.

convers t4 3600 (default is 7200, or 2 hours)

Next, you need to enable the interfaces you want the conference call to be active on. You might not want conference call connections on backbones, or wish to avoid confusion with other systems using the same alias for the conference call, etc.

Note: you can only enable the interfaces AFTER you have attached them!

convers interface port1 on
convers interface port2 on
convers interface port3 on

NODE ACCESS

In the second method, a user can connect to the regular node by connecting to the netrom alias (if used), the interface call, or the netrom call (if used). Then the user can give the 'C' command to join the conference bridge. The 'C' command defaults to ON, but can be turned off with **'mbox convers off'**.

CONVERS SERVER ACCESS

Third, there is the network convers server. This server listens to telnet port 3600 and allows both users and remote conference network servers to link to you. It defaults to ON, but can be stopped with the **'stop convers'** command.

LINKING TO REMOTE SERVERS

If you want to link to other conference servers, configure as many as you need as shown below. You can add new links at any time.

convers link 44.26.2.0
convers link 44.26.3.0

NOTE: it is very important to avoid link loops. They cause messages to fly around in circles, thus overloading the network. E.g. if you link to w0xyz, who in turn links to w0abc, there is no need for

either one of those to link back to you ! To minimize the chances of loops, there is loop detection code built in to the conference server. This will cause links creating loops to be closed as soon as the loop is detected. A 'loop detected' message will be sent to the host creating the loop, and this will keep that host from trying to reestablish the link.

If for any reason you want to **refuse** links from other hosts (or users for that matter), use the '**convers refuse <address>**' command to setup addresses to refuse. (Note that this only works for convers server access, not for regular users accessing the conference bridge via the conference call or the node 'C' command)

convers refuse 44.26.0.19

You can set an upper limit to the time the system will wait to reestablish (ie. re-link) a convers link it originated, after it has been lost.

To wait 10 minutes:

convers maxwait 600

A little on CONFERENCE INTERNALS.

The conference server in JNOS40 is modified from the convers code in NOS.EXE, but is identical to the stuff in the JNOS releases. Messages sent by a user get sent to all users on the local system as well as all users on remote systems. All local users get their own copy of a message. For users at remote systems, only one copy of the message is send across all the remote links available. Say there are 3 local users, and 2 remote links with 5 and 4 users respectively. If a local user sends a message, there will be 4 copies sent: 2 to the 2 remaining local users, and 1 message each across the 2 links. The message sent across the links will then be distributed to the users at each of the linked servers.

Sometimes a user connection or a remote links gets backlogged with data to be sent. This can happen if the connection goes bad and no more data makes it through. When this condition appears, no more data will be sent across the connection. This will remain so until this backlog condition clears or the connection is closed (due to retry time-out or whatever). Not sending new data on backlogged connections avoids data from piling up on the connection, thus keeping memory resources busy and slowly grinding the system to a

halt. If the condition clears, any new data to be sent will be sent again. All data that was being handled by the convers system while the connection was backlogged is lost.

The domain name system in JNOS40 consists of two separate parts. There is a Domain Client, and a Domain Name Server, or DNS. The Domain Client is the part that figures out what the IP-address is that goes with a certain hostname (among many possible other things). The DNS is the network server that provides answers to queries from other systems about hostname-ip mappings.

Both the Domain Client and the DNS have a minimum configuration to work properly. To better understand the workings of the Domain Name System, let's overview the process of resolving a hostname to ip-address translation in a little more detail.

When domain names need to be translated to ip-addresses, both the Domain Client and the system's DNS will follow three steps:

First, the Domain Cache is searched for the needed information. The Domain Cache is a small database in the RAM of the system which holds recently accessed Domain Records. At startup, the Domain Cache will be empty. Each time a query is resolved, the answer is given to the requester, and it is also stored in the cache. When the cache gets full, the oldest record will be overwritten by the new information.

Second, if the answer is not found in the Domain Cache, the system will try to get it from the internal 'domain.txt' file. This is a file similar to the domain.txt for the PC versions of NOS. It can contain lots of information on name-to-ip mappings and more. See below for more information on how to create a valid domain.txt file. See the section 'Configuring eproms' on how to configure the eprom images with the domain.txt file.

Third, if the information is not found in either the cache or the internal domain.txt 'file', and f a remote DNS has been configured, the system will try to contact the remote DNS next to resolve the query over the network.

In order for steps two and three to work properly, some configuration has to be done. Let's first do the easy parts.

First, activate the domain name server:

domain dns on

To add a remote DNS for the system to use,

domain addserver 44.26.1.16

If you add more than one remote DNS, each will be queried in a sequential fashion. Each query will first be sent to the most recently configured DNS (i.e., the last line in the autoexec.nos file or the last entered from the console). If this DNS times out or does not know the answer, the next most recently configured will be tried, and the process repeated until either an ip-to-hostname mapping is retrieved or all remote DNS have been tried and failed.

The DOMAIN.TXT file

The domain.txt file will be read by the configuration program CFG.EXE and placed in ROM at the appropriate place. The domain.txt file has the same format as in other disk-based NOS systems.

The three most important formats of lines in the 'domain.txt' file are described below.

This first format is the most common, and could be called 'address' records :

<domain.name> IN A <dotted-decimal-ip-address>

Example:

wg7j.ampr.org. IN A 44.26.1.20

These records allow the domain name system to handle queries that attempt to resolve a hostname into an ip address. Note the mandatory ending period AFTER the full name!

The second format is the 'canonical name' record:

<alias.name> IN CNAME <domain.name>

Example:

bbs.wg7j.ampr.org. IN CNAME wg7j.ampr.org.

The canonical name record allows a system to be known by both its primary domain name as well as one or more alias names. NOTE the

period after both the alias.name and the domain.name. You can have as many CNAME records as you wish, but more than one or two is probably not very practical.

The third format is a so-called pointer record:

<reverse-dotted-decimal-ip-address>.IN-ADDR.ARPA. IN PTR <domain.name>

Example:

20.1.2.6.44.IN-ADDR.ARPA. IN PTR wg7j.ampr.org.

This pointer record allows the domain name system to handle queries that attempt to translate an ip address into a hostname. NOTE the REVERSE order of the ip address at the start of the line, with the string '.IN-ADDR.ARPA' attached. Also note the periods AFTER the 'ARPA' part AND after the full domain name.

There are additional valid formats, but they are beyond the scope of this document. Consult your local TCP/IP guru or the appropriate RFC's on the Domain Name System for more.

NOTE: Domain.txt is loaded in ROM and CANNOT be changed. Even when linked to other domain servers via the 'domain addserver' command, updated records are stored only in domain cache so that even though they could override records in domain.txt, they will NOT be saved in the event of power failure or a commanded re-boot. They will be stored in the Domain Cache, and available there for later use as long as the node remains powered up.

For additional valid formats, see the sample 'domain.txt' file distributed with this package as well as the Request For Comments documents from various Internet sites and books on tcp/ip, or contact your local tcp/ip guru. See also the section on Configuring JNOS40 for more...

RSPF is a dynamic path determination scheme. It is a method to allow the network to periodically review resources and evaluate whether paths in use are optimal. To activate RSPF, first set the broadcast address for the destination interface.

example port1:

ifconfig port1 broadcast 44.255.255.255 (DO NOT USE THIS ADDRESS!)

This sample automatically would create a routing entry for 44.255.255.255 in the routing table. This address is the one that you are broadcasting to make the network aware of your existence, the same as a netrom node broadcasts information about its existence. If you intend to use RSPF on more than one interface, each interface must have its own address. Otherwise, the routing entries will be overwritten by the next definition entered in the table.

Next, configure port1 as an RSPF interface with horizon 32 and a quality of 1 (hops). These are typical values for an end node. Replace the 1 with an 8 for immediate nodes.

rspf interface port1 1 32

Set the interval between RRH messages.

rspf rrhtimer 900

Define how long it takes until an idle link is suspected to be bad.

rspf suspecttimer 2000

Set the interval between routing updates.

rspf timer 900

There is not much else to do except monitor performance and adjust the parameters as you see deviations from your goal.

The RSPF specification is available from most of the sources listed near the front of this document. As of this writing, the most recent version is RSPF20.ZIP.

The JNOS40 code resides in two 1-Mbit (128kx8) eproms. These eproms replace the Kantronics firmware rom that is in socket U7 on the Data Engine board. Please consult the manuals provided with your Data Engine for instructions on opening up the unit.
JNOS40 is distributed in two code images. They are both binary eprom images, each a 128Kb large. The CFG.EXE program will modify both distributed images as described below. The resulting images will be "burned" into eproms which will be placed into the sockets marked U7 and U8.

NOTE: Both CHECK.EXE and CFG.EXE are version specific. You must use the copies of these programs which are distributed with a particular version of JNOS40. If you attempt to use incorrect versions, an error message will be displayed and program execution will stop.

The CHECK.EXE program checks the autoexec.nos file for obvious problems. It will report errors, and allows you to play 'what if' games from a command prompt. It also allows you to step through your autoexec.nos, simplifying debugging.
Error messages are the same as given from the console or remote sysop connection of a running JNOS40 system. Using CHECK.EXE, error messages also indicate the line number where the error occured.
CHECK.EXE has several command line options that can be listed by running it with the '-?' help option. Options can be in any order and are separated by spaces or tabs.

The command line options for CHECK.EXE:

```
<filename> alternate 'autoexec.nos' config file.
-i         interactive after loading of config file.
-q         quiet mode (no output).
-t         trace mode (stop after each line parsed).
-v         verbose mode (show lines being parsed, stop after
errors).
-?         produce this help.
```

If <filename> is not given, 'autoexec.nos' in the current directory is assumed. If the configuration file cannot be found an error message is printed.

Command line options explanations:

-i places CHECK.EXE in interactive mode, AFTER the configuration file (autoexec.nos or another file indicated with the <filename> option) is parsed. Interactive mode causes CHECK.EXE to emulate a JNOS40 system. You will get a command prompt similar to JNOS40 or JNOS systems. All commands are interpreted, but not all will be effective because CHECK.EXE is NOT a full working JNOS, it only emulates one. Therefore, some command results will be different from a running JNOS40 system. For example, the 'ps' or 'socket' output will differ, because many processes and sockets are not started or used. You can exit the interactive mode by typing 'exit' at the prompt.

-q suppresses all output. CHECK.EXE will end with a return code indicating the number of errors that occured in the configuration file. CFG.EXE uses this mode to test the configuration file before configuring the emprom images. The -q option overrides the -i, -t and -v options.

-t and -v options are similar. They both print the line being parsed before it is parsed. If errors occur, messages will be displayed. After parsing each line, the -t option will always stop and ask you to hit <enter> (or <cr>) to continue. The -v option will stop only if the line produces an error. In both cases, when you are asked for keyboard input, typing 'Q<enter>' will stop the parsing of the configuration file. This is handy when used with the -i option, because it allows partial parsing of the configuration file to see what the results were to that point.

The CFG.EXE program provided with this distribution will configure new eprom image files from autoexec.nos (or whatever other name you give the configuration file). CFG.EXE will tell you how many bytes of space the eproms have remaining for autoexec.nos and domain.txt. Next, you will be requested to give a (maximum) seven (7) character name for the configured binary eprom images which are to be produced. The program will append the numbers 7 or 8 to the filename to indicate installation in socket U7 or U8. If the name you give is an existing and valid eprom image pair, those images will be used.
For example, if you give the name 'CONFIG' then CFG.EXE will check for existence of the files CONFIG7.BIN and CONFIG8.BIN. If the name is not an existing or valid pair of images, CFG.EXE will create a pair by copying NOS40LO.BIN and NOS40HI.BIN to filenames CONFIG7.BIN and CONFIG8.BIN.
CFG.EXE will then read the configuration files. It will ask you for both autoexec.nos and domain.txt file names. These files can have

any name as long they are valid filenames. The CFG.EXE program will call CHECK.EXE to do validity checking of the autoexec.nos and domain.txt files.
Next CFG.EXE will read the configuration files from disk and eliminate all comments, unneeded spaces, and tabs; and copy the remaining lines into the eprom images. The resulting binary images should be burned into two 128kx8 eproms and be put in sockets U7 and U8. In the example below, you should use the file SAMPLE7.BIN for U7 and SAMPLE8.BIN for U8.

The following is a screen dump of a configuration run. User entries and program output are not differentiated. You must type in the full names of the autoexec.nos and domain.txt configuration files if they are different from the defaults. This listing was generated using the autoexec.nos and domain.txt sample files which are part of the distribution. The files were renamed to show how file names different from the defaults are entered.

```
JNOS for the Data Engine(tm)
(c) 1994 Johan. K. Reinalda, WG7J.
v1.00 Feb 20 1994
Eprom Configuration Utility.

16432 bytes space for autoexec.nos and domain.txt .

Enter ROM file name (7 chars max): SAMPLE
Creating new copies of the eprom images...

Enter configuration file name
(<cr>=autoexec.nos): SAMPLE.NOS
Reading SAMPLE.NOS...
Adding 1979 chars in 65 lines (457 comment lines deleted)
to AUTOEXEC.NOS area...

14453 bytes space left for domain.txt .
Enter domain file name
(<cr>=domain.txt or 'none' to skip): SMPLDOMN.TXT
Reading SMPLDOMN.TXT...
Adding 1760 chars in 42 lines (23 comment lines deleted)
to DOMAIN.TXT area...


Outputfile 'SAMPLE7.BIN' created for DE socket marked U7.
Outputfile 'SAMPLE8.BIN' created for DE socket marked U8.
```

The following is taken from John Wiseman's (G8BPQ) excellent BPQ node package. All credit goes to him for the work he continues to put into that project. After the excerpt from G8BPQ, there is a discussion of an alternate approach - WG7JKISS or "bus-contention KISS."

KISS Proms for use with TheNode and JNOS/JNOS40

"Several PROM images are supplied for for use with TNC2 (or clones), and one for use with the TNC220.

KISS is as released with the TCPIP package. I have used this code, and it seems to work, but it does have loopholes in its buffer management. As I have experienced buffering problems with other KISS mode TNCs with TheNode, I've done a version which will reset if it runs out. This is a bit drastic, but should keep the system going. (Higher level software will retry the discarded messages). If it improves things, I'll refine it to discard the oldest. The new eprom image is in the file JKISS.

220KISS is a version of JKISS, modified to run on the TNC220. This version only supports the VHF port (port 2) at 1200 baud,and the aync link to the PC is fixed at 2400. The DCD led is driven by software, but is controlled by the DCD signal from the modem (ie DCD processing is the same as with the TNC2 - the SOFTDCD mode of the 220 is not implemented). Other versions are possible if there is sufficient interest - the main problem is that the KISS command set would have to be extended to include commands for port and speed switching.

Note the software is now set up to run with the clock speed jumper in the 'Low' speed position - several people have had problems running in the 'Fast' mode.

BPQKISS - A Multidropped KISS system. (TNC2 and clones only)

I have implemented a system to allow more than one KISS-like TNC to connectto a single Async port. This is primarily for those running machines with little expansion capability, but can also enable the TNCs and transceivers to be located remotely from the PC with a simple 3 wire link. This could be useful on the lower frequencies, where QRM from the PC blocks the receiver (I have real problems running a 50meg RX near the PC). A simple checksum is also added to

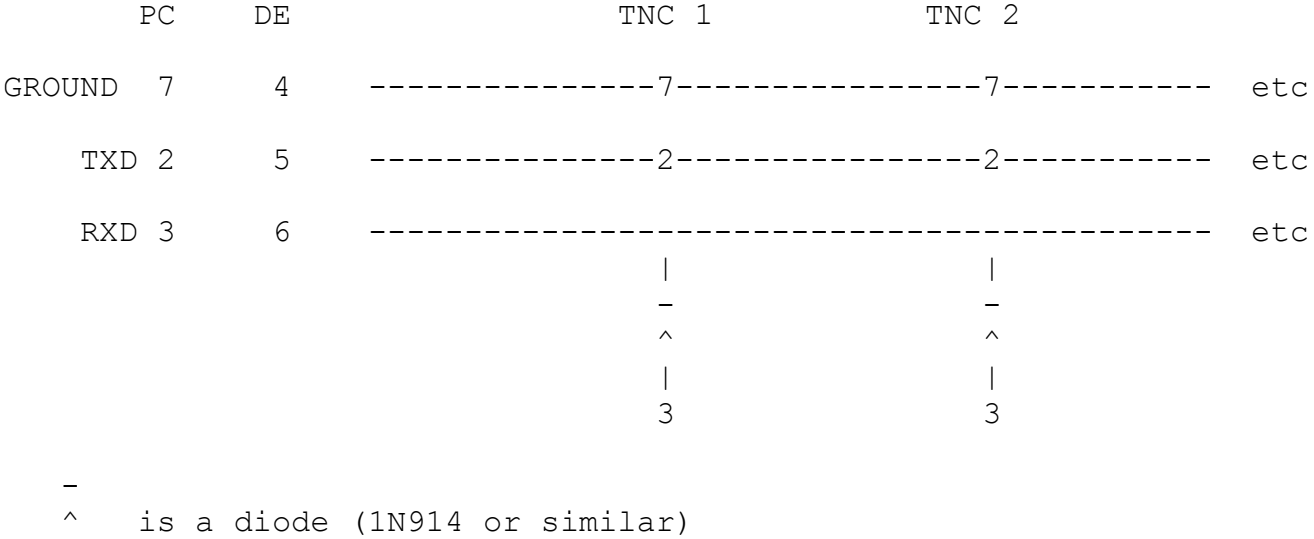
each packet, to reduce the risk of corruption if running on long leads (or even over a modem link).

The system uses polling to prevent contention on the link. Each TNC must run the BPQKISS program, and each must have a different 'address' byte patched in at location 20hex. In the PORTS section of BPQCFG.TXT, PROTOCOL must be set to KISS, KISSOPTIONS to POLLED,CHECKSUM,ACKMODE and CHANNEL set to correspond to the address in the PROM -

CHANNEL	Address (in byte 20h of PROM)
A	00h
B	10h
C	20h
D	30h etc

In theory you can have up to 16, but in practice the maximum will depend on the power of the PC and the speed of the radio ports.

Wiring. (PC indicates IBM PC compatibles with DB25 serial port. DE indicates a Kantronics Data Engine (tm).)



\-
^ is a diode (1N914 or similar)

With some TNCs and serial cards, a pulldown resistor may be required from pin 3 on the PC (10k to -12v is suggested). Thanks to G3ZFJ for this information.

The protocol used for this multidropped option was changed from

version 3.59a onwards to be compatible with similar software produced by KANTRONICS for their range of TNCs. The new version is called BPQKISS, and replaces the old JKISSP.

For those of you unfamiliar with KISS TNCs, the STA led indicates frames being received from the PC and the CON led frames being sent to the PC. On powerup, some LEDS should flash about 3 times - which ones depends on the version and the RAM size in the TNC.

CWID

I have added CWID to my JKISS and BPQKISS EPROMs. The Callsign to be sent is patched into the EPROM image with the program PATCHID, which takes two parameters, the required callsign, and the file to patch (either JKISS or BPQKISS). Note that the specified file is overwritten, so I suggest you make a copy of the original first.

```
     PATCHID G8BPQ JKISS    - Note call must be in upper case
```

The CW patttern which will be sent is displayed on the screen - please check it, just in case my translate table is wrong!

The ID is sent after one minute, then at 29 minute intervals, with a dot length of 60ms. If my calculations are correct, this equates to 20 WPM. The ID is send in AFSK (assuming a normal AFSK modem), but because of the hardware design it is not possible to control which tone corresponds to mark or space - it depends on what was sent just before the ID starts.
If you are using a modem with a scrambler (eg G3RUH), then the system wont work - the two tones will sound the same. I'd like to know if a CWID facility for RUH modems would be useful - if there is a significant demand I'll see if I can find a solution. A simple on-off keying may be possible, but would depend on the PTT characteristics of the TX. If you have a better idea, please let me know!

The normal SLOTTIME/PERSISTENCE code is used to minimise collisions with other stations (hence the interval of 29 mins, allowing a bit of time for congestion, without going over the statutory 30 mins interval)."

John Wiseman, G8BPQ

7 March 1991

/
**/

WG7J's bus-contention KISS is a modification of the original KISS program. It also solves the buffer problem by resetting the TNC which is not very graceful, but it works. In addition, it contains serial port contention code which will allow multiple TNCs to be connected to a serial 'bus' between the TNCs and the host (either Personal Computer (PC) or Data Engine (DE))

'Bus' contention is done with a similar scheme used in the popular Net/Rom and TheNet diode matrix approach.

When a TNC is about to transmit a frame on the serial port, it checks the status of the DTR line. If the line is active, the TNC will wait untill it becomes inactive. If the DTR line is inactive, the TNC will wait a random interval, 0, 10, 20 or 30 milliseconds. If after this time the DTR is still inactive, the TNC will activate the CTS line and start transmitting the frame. However, if the DTR line is found active again (meaning another TNC started transmitting), the TNC will wait until the DTR line becomes inactive again. By default each TNC will wait no longer than 60 seconds before transmitting a frame. After this time, it is assumed there is a problem and the outstanding frame is 'jammed' out the serial port.

Each TNC must run the WG7JKISS program and each must have a different 'address' byte patched in at location 20hex. Addresses should be as described above for the BPQKISS rom:

CHANNEL		Address (in byte 20h of PROM)
(BPQ)	JNOS	
A	0	00h
B	1	10h
C	2	20h
D	3	30h etc

NOTE:
When using the JNOS and JNOS40 tcp/ip programs, the channels above are numbers. BPQ channel A is 0, B is 1, C is 2, D is 3, etc.

The WG7JKISS eproms have not been tested with G8BPQ code in a PC or Data Engine. However, it should be possible to use them by configuring multiple kiss ports. The WG7JKISS rom currently does not support POLLED, CHECKSUM or ACKMODE operation.

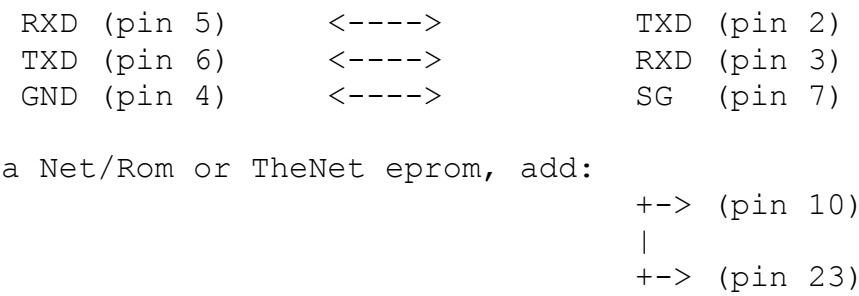
When the serial port is used as a network interface, hardware handshaking is supported via the CTS and DTR lines - pins 7 (CTS) and 3 (DTR) on the serial cable provided with the Data Engine. Note that handshaking is independent the mode of the serial port. It works with SLIP, KISS, and NRS !
If you don't want to use hardware handshaking, you may either leave DTR and CTS unconnected or you can issue the attach command without the [c] parameter which disables hardware handshaking. Hardware handshaking is not appropriate and is disabled automatically for G8BPQ polled kiss (or 'pkiss') type serial port connections. When using WG7JKISS eproms serial port hardware handshaking is not needed and the [c] parameter is a "don't care" as long as the the CTS and RTS lines are not connected to the serial port.
The following describes how hardware handshaking is implemented in the Data Engine when:
the [c] parameter is activated in the 'attach' command and

the handshaking signal lines are connected.

When the Data Engine serial port driver is ready to begin transmission of a frame, it senses the DTR line. If DTR is logical

'0' (a positive voltage on the RS-232 line), it will start transmitting the frame. Logical '0' is the default state when the DTR line is not connected, allowing frames to be transmitted when DTR is not connected. If the DTR line is a logical '1' (a negative voltage on the RS-232 line), transmission of the frame will be delayed until the DTR line clears to a logical '0'. During the delay period, the DTR state is checked every 50 ms by the Data Engine.
While transmitting a frame on the serial port, the Data Engine will set the DTR line from a logical 0 to a logical 1 which is a negative voltage on the RS-232 line. Handshaking is on a per-frame basis. Once transmission of a frame has started, raising the DTR line will NOT result in that packet being interrupted. Any following frame cannot be sent until the DTR line is lowered again.

Hooking up a single TNC is easy. You don't need the hardware handshaking. Simply connect RXD,TXD and GROUND between the Data Engine and the TNC. Make sure the Data Engine is configured correctly - serial speed, serial mode (slip, ax25 or netrom), etc. See the appendix ATTACHING INTERFACES for more details.
If you hook up a TNC with a Netrom or TheNet prom installed, don't forget to jumper pin 10 to pin 23 on the TNC. This connection puts the TNC into network mode, as opposed to hostmode. See the next section for additional information about using Netrom and NRS protocol. Of course, you could just burn a kiss eprom to use the full services of the DE running JNOS40 and eliminate such concerns.



The hardware handshaking described earlier allows the commonly used 'diode matrix' scheme to work with the Data Engine. If you use this approach with NET/ROM-configured TNCs, you should also refer to the manual that came with your favorite flavor of the NET/ROM compatible code, and to the manual of your TNC.
The TheNet V2.10 manual states the following about hooking up TNC2 clones to the diode matrix:

"On EACH DB-25, pins 10 and 23 are jumpered together as are pins 4 and 20. Comment: The original TAPR TNC-2 circuit board layout mistakenly had CTS connected to pin 20, instead of pin 4. It took awhile before manufacturers licensed by TAPR caught and corrected this error. Meanwhile it is good policy to simply jumper pins 4 and 20 in case of a mix of older and newer TNCs being used in the stack."

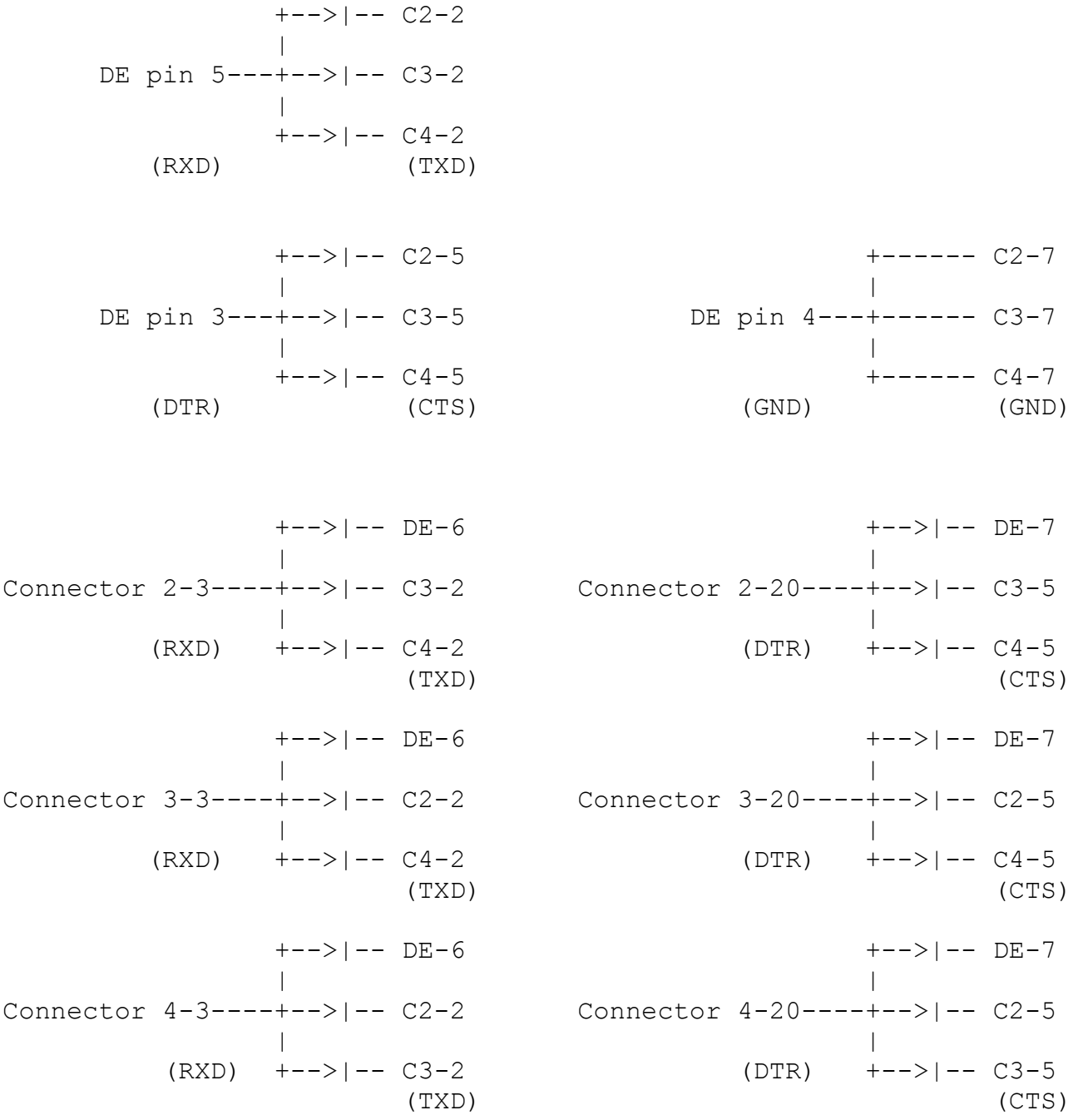
The following example shows how to hook up 3 TNCs to the Data Engine via a diode matrix. It can easily be seen how to expand this to more. The diagram is basically the same as that in the TheNet 2.10 manual (tnet210_20.doc) In that diagram, connector 1 would be the Data Engine serial port, and Connectors 2,3 and 4 are TNCs
Diode matrices can be either homebrewed or purchased. For information on a commercially available 6-port diode matrix board, send a SASE to: NorthEast Digital Assn, PO Box 563, Manchester, NH 03105-0563.
If you buy or make, or already have, a diode matrix board with DB25 connectors, you simply have to connect a DB25 connector to the RJ45 serial cable connecter supplied with the Data Engine. In order to have this work with the diode matrix board, this should be hooked up as follows:

Function	**DE**		**DB25**
(DTR)	pin 3	<-->	pin 20
(GND)	pin 4	<-->	pin 7
(RXD)	pin 5	<-->	pin 3
(TXD)	pin 6	<-->	pin 2
(CTS)	pin 7	<-->	pin 5

Connection Diagrams for the NRS Diode Matrix

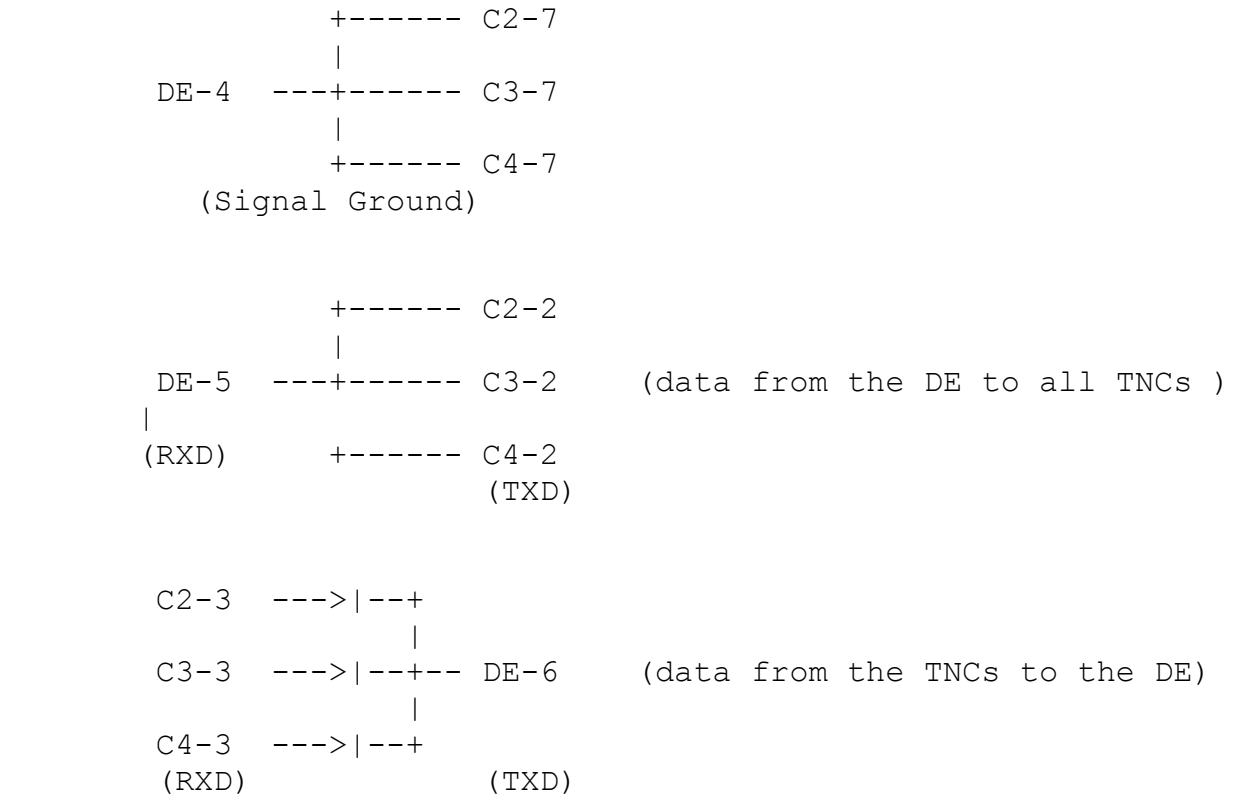
Data Engine pins are marked as DE-x, the others are the TNC connecters. TNC pin numbers are for DB25 connectors.

-->|-- is a 1N914 diode or equiv.



This connection is simpler than the diode matrix scheme described for NRS connections. TNCs send data (not to be confused with handshaking) to only the DE and not to other TNCs resulting in fewer connections.

Data Engine pins are marked as DE-x, the TNC connectors are marked C.

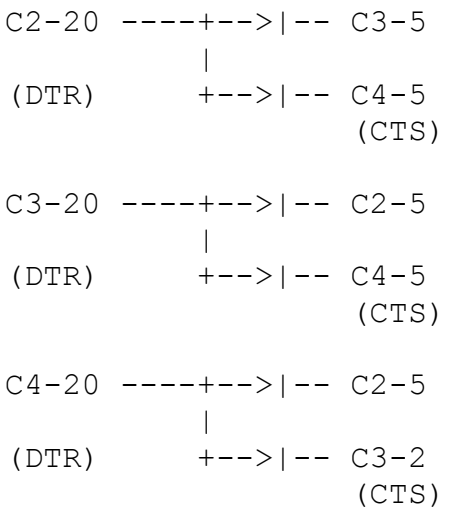


There are no handshaking signals used with G8BPQ polled-kiss.

The simplicity of connecting the Data Engine to multiple TNCs with G8BPQ is limited to the polled kiss environment. WG7J KISS allows the use of TNCs in KISS mode without the overhead caused by polling. The data connections are identical to G8BPQ polled-kiss. WG7J-KISS adds handshaking similar to the NRS scheme.

Signal Data and Ground connections are as described above for G8BPQ.

The handshaking signals are between the TNCs only.



The Data Engine serial port can be used to connect a console instead of a network interface. When connected in console mode, the serial port is unavailable as a network interface meaning you cannot **attach** it. The console can be used to control node functions, perform debugging, and trace activity on the internal radio ports. You cannot originate connections to other stations from the console as with Net/Rom and TheNet.
You only need a simple 3-wire connection to connect the receive data, transmit data and common lines to connect a terminal or computer to the serial port .
The console is configured with the CFG.EXE program the same as any other interface. The baud rate can be any valid baud rate between 1200 and 9600 Bd. Data format is 8 bits, no parity, 1 stop bit (8n1). The console 'verbose' command in autoexec.nos sets how much information is displayed on the console during startup.
Battery-backed configuration changes can be made from the console using the **'add'** command. Configuration data entered in this manner will be retained across power cycles and mode switches including

commanded state changes. For example, if you start a server in console mode, when you restart the Data Engine with the serial port in network mode, that server will be started again. If you do not want the server in this example to be restarted, then you must delete the 'add' line while in console mode.
Use the 'trace' command in order to perform protocol tracing on the two internal radio interfaces. Trace works the same as in JNOS. (See the SYSTEM COMMANDS section and the Commands Manual for more detail.)

To use the console:

1) Turn off power to the Data Engine.
2) Press IN the AUX switch (at the left of the power switch).
3) Turn on power to the Data Engine.

You will get a prompt similar to the JNOS.EXE program, and probably some error messages. These error messages result from the serial port not being available as a network interface, and are of no concern. From this point, you have the same interface as in SYSOP mode.

This section is written to provide insight into normal operation of the JNOS40 node. It is intended to help a sysop or a user who is new to JNOS40 understand basic operating characteristics.
The behavior of the node differs depending on the way a connection is established. The node can be in 'verbose' or 'non-verbose' mode after you've connected.
When connected to the **node alias** (with any ssid), or via a **telnet session**, the node's response is '**verbose**'; i.e. the node gives more feedback about what is going on than regular Net/Rom nodes.

When connected to any of the **callsigns or** via **netrom**, the node response is '**non verbose**', like a regular net/rom system, i.e. with minimal feedback. These differences affect the login procedure, and the node 'B' and 'C' commands.

In the following examples, lines marked 'U)' come from the user or his TNC, whereas lines marked 'N)' come from the node. '<-' and [...] indicate some comments. Assume the user call is K7UYX, the NODE callsign is WG7J-1, and the Node alias is JNOS40.

Here is a typical exchange when connecting to the alias ('verbose' mode). In telnet connects, there is a login procedure before the message of the day which is not shown here.

```
U) cmd: c jnos40
U) *** connected to JNOS40

N) This is the message of the day! <- this is set with 'motd' cmd.
N)
N) Type ? for help.                <- this always shows
N)
U) C NOSBBS
N) JNOS40:WG7J-1} Trying... the escape character is: CTRL-T

     [after some time, during which you can hit the escape char
     to quit the connection.....]

N) JNOS40:WG7J-1} Connected to NOSBBS (escape enabled)
.
.
.        [stuff with the bbs,
.              until you disconnect.....]
.
.
N) Reconnected to JNOS40:WG7J-1
```

```
U) B
N) JNOS40:WG7J-1} thank you k7uyx,
N) for using jnos.wg7j.
U) *** disconnected from JNOS40
```

This is the exchange when connected to any call or via netrom ('non-verbose' mode):

```
U) cmd: c wg7j-1
U) *** connected to WG7J-1
N)                              <- the node says nothing more!
U) C NOSBBS
.
     .....[after some time, during which you can send the
     escape character to quit the connection]
.
N) JNOS40:WG7J-1} Connected to NOSBBS
.
.
.       [stuff with the bbs,
.               until you disconnect from the BBS]
.
.
.
N) Reconnected to JNOS40:WG7J-1
U) B
U) *** disconnected from WG7J-1
```

Making NET/ROM (node) connections with JNOS40 is the same as with Net/rom, G8BPQ and TheNet systems. Simply type 'C NODENAME'. Use the 'N' command to get the Nodes destination table list.
AX.25 downlink connections are identical to the G8BPQ format. You specify the radio port you want to use to make the connection. Use the P command to get a list of ports with a description of each port. To initiate the connection, type 'C PORT CALL'.
TELNET connections are simple; you only have to know the ip-address of the system to connect to. If your JNOS40 node has been configured to use a name-server, you can simply enter the destination system's hostname (often the callsign).
T 44.26.0.224 or T home.wg7j

TTYLINK connections are the same as telnet connections, only they go to the keyboard-to-keyboard server of the tcp/ip system.
TT 44.26.0.225 or TT home.wg7j

JNOS40 has a stay here feature which allows the circuit to remain complete to the last node specified in the 'connect' command. 'Stay here' can be disabled by added 'd' to the end of the connect command line. It can be used with or without the 'escape' character described in the next section.
The syntax of the connect command is:
C <port> <destination> ['e']['d'] where:

[e] controls escape checking and
[d] disables 'stay here' if present in the command string.
See the JNOS and JONS40 COMMANDS MANUAL for additional explanation of this command.

The escape character can be set to any desired character with the 'E <c>' command, where <c> is the character you want to use. You type an 'E', then a space, and then the ONE character you want to use as the escape character, followed by a return or enter.
Example: 'E \<enter>'
The default escape character is CTRL-T, the combination of the control and 'T' keys on a pc-style keyboard.

The escape command serves two functions in JNOS40.
1) It serves as a 'connect attempt abort' command, and,

2) if enabled, as a 'remote connection abort' command.

The **'connect attempt abort'** function is ALWAYS active, even when you aren't notified by the node; that is, when you connect over netrom or to the node callsign (instead of the node's alias.)
The **'remote connection abort'** function defaults to ON when connected to the ax.25 alias or via telnet, that is, when the node is in 'verbose' mode. If connected via net/rom or to the callsign (i.e. 'non-verbose' mode), 'remote connection abort' defaults to OFF! After you've connected, you may toggle the 'remote connection abort' function of the escape character with the 'E on' or 'E off' commands

'CONNECTION ATTEMPT ABORT'

Unlike Net/Rom and Thenet, JNOS40 nodes do NOT accept any commands while trying to establish a connection for you. With Net/Rom or Thenet, if you type 'R' while waiting for your connection to be established, you get a routes list from the node. This cannot be done with JNOS40. You can, however, abort the connection attempt in progress by typing the 'escape' character. The node will then return you to the normal command mode.

For example:

```
N) JNOS40:WG7J-1}
U) C NOSBBS
N) JNOS40:WG7J-1} Trying... the escape character is: CTRL-T
.....[you get impatient :-)]
U) <ctrl-t>
N) JNOS40:WG7J-1}
```

'REMOTE CONNECTION CLOSE'

If Escape checking is enabled, once connected you can disconnect from a remote connection by simply typing the escape character. You will be returned to the command session of the node to which you are uplinked.

Example:

```
N) JNOS40:WG7J-1}
U) C NOSBBS
N) JNOS40:WG7J-1} Trying... the escape character is: CTRL-T
N) JNOS40:WG7J-1} Connected to NOSBBS (escape enabled)
.
...... [blah blah with bbs]
.        [you're done, or the link is slow or whatever]
.
U) <ctrl-t>
N) Reconnected to JNOS40:WG7J-1
```

Finally, if you connect to the 'none-verbose' mode (netrom or callsign), which means the escape character checking defaults to off, there is another way of turning it on:
You can add an 'E' or 'e' as the LAST character of the command line requesting the next outgoing connection. You may only do this for the connection being added.

Example: You connect to netrom node JNOS40. Escape character checking is off by default. You want to use escape character checking while connecting to some other station.

```
U) C JNOS40
N) NODE:W7ABC} Connected to JNOS40
U) C NOSBBS E                          <- Request escape checking
.....[after some time].....
N) JNOS40:WG7J-1} Connected to NOSBBS (escape enabled)  <- escape
                                                          char is on!
.        [blah blah with bbs]
.        [you're done, or the link is slow or whatever]
.
U) <ctrl-t>                        <- The Escape character
N) Reconnected to JNOS40:WG7J-1
```

The conference server is a round-table chat system that allows multiple users to talk to one-another. Through linking the conference server has the capability to span whole states, or countries and parts of the world as is currently done.
The following is a list of conference server commands. All commands start with the '/' character.

Anything NOT starting with the '/' will be taken as text, and sent to ALL users on the channel. Text will show as:
'<user name>: text you typed '

Commands may be abbreviated to their first letter and are not case-sensitive.

/? or **/Help**	Print help
/Bye	End the convers session
/Channel n	Switch to channel n
/Exit	Same as /B
/Invite user	Invite user to join your channel
/Links [Long]	List all connections to other hosts
/Msg <user text>	Send a private message to user
/Personal	Set personal data
/Quit	Same as /B
/Sounds y|n	Turn on bell
/Users	List all users
/Who	List all users
/Write <user text>	Same as /M

Once logged into the server, you may use /B, /E or /Q to logoff.

To see the users and some info on where they are, type /U or /W This shows the following type of display:

User	Host	Via	Ch.	How long	Personal data
wg7j	Crv-Or		0	2m:34s	Johan in Corvallis, OR
wb5bbw	SC-Texas	Ottawa 10		1d: 2h	
kd5iz	Alamo	Ottawa 10		4h:58s	Jack
k7uyx	NODE				

This display shows there are currently 3 users in the conference system.

'Host' shows the names of the conference system each user is logged on to.

'Via' shows how your conference system found out about that user. In other words, the Ottawa conference server told us (Crv-Or) that on the server SC-Texas there is a user named wb5bbw

'Ch.' shows the channel each user is currently on. There can be up to 65000 different channels, so you can have conversations in little groups without bothering others.

'How long' shows how long each user has been logged in.
d = days, h = hours, m = minutes, s = seconds

'Personal data' shows some info each user has set about him/herself with the /p command. Personal data is carried across links only if the remote servers support it. Any system running JNOS40, or JNOS1.04 (or later) for the PC will show personal data throughout the conference system.

The last line in the user display indicates that the user K7UYX is currently connected to the NODE associated with this conference server.

OTHER CONFERENCE COMMAND EXAMPLES:

/L Links are connections to other conference servers that carry the information about users, as well as the data sent across to the other system. The links in a system can be shown with the /L command. '/L L' will show a somewhat more elaborate display...

/P Personal Data can be set with the /P command. It will show in the user display. To show it, type /P. To set or change, simply type '/P what ever you want' This data will be updated across all links available.

/C Changing channels is easy, simply type '/C number'. A plain '/C' will show your current channel.

/I If you want to invite a conference user or NODE user to join you on your current channel, simply type '/I name'. An invitation message will be sent to the user indicating you would like them to join you in a conversation.

/M or **/W** You can send a private message to a user. A private message is only seen by the user addressed, no matter how many others there are on his channel. This works even if you are not on the same channel.

Example: To send Jack a private message you would type

'/M kd5iz Hi Jack, how are you ?'

Jack would see this as

<*wg7j*>: Hi Jack, how are you ?

Note the '*'s which indicate the message was sent as private.

From left to right, the front panel LED's indicate the following:

1 - Push-to-talk port A (Transmitting data when led is on)
2 - Data Carrier Detect port A (Receiving data when led is on)

3 - Push-to-talk port B
4 - Data Carrier Detect port B

5 - Serial port transmit packet
6 - Serial port receive packet

7 - At least one user connected to the node shell
8 - 'Health' indicator (Gets toggled on/off every second);

To reduce power consumption, the LEDs can be toggled on/off with the 'led on|off' command. Add 'led off' to your configuration if you don't need the LEDs to indicate the status of the ports.

The hardware watchdog in the Data Engine is triggered each time a task switch is attempted. If a process hangs in a loop (i.e. the system crashes), a hard reset will occur automatically.

ARRL Computer Networking Conference Proceedings
Available from ARRL HQ, Newington CT.
Send mail to info@arrl.org for an automatic response pointing at more information about the ARRL.
Some of these papers are available online in the directory ucsd.edu:/hamradio/packet/tcpip/docs.

This list is not exhaustive; there are many other interesting articles, but these are the ones most relevant to NOS and TCP/IP.

NOS Overviews and Documentation

NOS Command Set Reference
Ian Wade G3NRW 10th (1991)

NOSVIEW: The On-Line Documentation Package for NOS
Ian Wade G3NRW 11th (1992)

The KA9Q Internet (TCP/IP) Package: A Progress Report
Phil Karn KA9Q 6th (1987)

Amateur TCP/IP: An Update
Phil Karn KA9Q 7th (1988)

Amateur TCP/IP in 1989
Phil Karn KA9Q 8th (1989)

Services and Protocols

The Design of a Mail System for the KA9Q Internet protocol
Bdale Garbee, N3EUA 6th (1987)
Gerard van der Grinten, PA0GRI

Finger - A User Information Lookup Service
Michael T. Horne, KA7AXD 7th (1988)

Callsign Server for the KA9Q Internet Protocol Package
Doug Thom, N6OYU 8th (1989)
Dewayne Hendricks, WA8DZP

The Network News Transfer Protocol and its Use in Packet Radio
Anders Klemets, SM0RGV 9th (1990)

A Routing Agent for TCP/IP: RFC 1058 Implemented for the KA9Q Internet Protocol Package 7th (1988)
Albert G. Broscius, N3FCT

Thoughts on the Issues of Address Resolution and Routing in Amateur Packet Radio TCP/IP Networks
Bdale Garbee, N3EUA 6th (1987)

Another Look at Authentication
Phil Karn KA9Q 6th (1987)

LZW Compression of Interactive Network Traffic
Anders Klemets, SM0RGV 10th (1991)

PACSAT Protocol Suite -- An Overview
Harold Price, NK6K 9th (1990)
Jeff Ward, G0/K8KA

BULLPRO -- A Simple Bulletin Distribution Protocol
Tom Clark, W3IWI 9th (1990)

Macintosh

KA9Q Internet Protocol Package on the Apple Macintosh
Dewayne Hendricks, WA8DZP 8th (1989)
Doug Thom, N6OYU

Status Report on the KA9Q Internet Protocol Package for the Apple Macintosh
Dewayne Hendricks, WA8DZP 9th (1990)
Doug Thom, N6OYU

Higher Speed Amateur Packet Radio using the Apple Macintosh Computer
Doug Yuill, VE3OCU 10th (1991)

Network design

The Implications of High-Speed RF Networking
Mike Chepponis, K3MC 8th (1989)
Glenn Elmore, N6GN
Bdale Garbee, N3EUA
Phil Karn, KA9Q
Kevin Rowett, N6RCE

Design of a Next-Generation Packet Network
Bdale Garbee, N3EUA 8th (1989)

More and Faster Bits: A Look at Packet Radio's Future
Bdale Garbee, N3EUA 7th (1988)

Physical Layer Considerations in Building a High Speed Amateur

Radio Network
Glenn Elmore, N6GN 9th (1990)

Spectral Efficiency Considerations for Packet Radio
Phil Karn, KA9Q 10th (1991)

This should be considered to be required reading.

MACA - A New Channel Acess Method for Packet Radio
Phil Karn, KA9Q 9th (1990)

A Duplex Packet Radio Repeater Approach to Layer One Efficiency
Robert Finch, N6CXB 6th (1987)
Scott Avent, N6BGW

A Duplex Packet Radio Repeater Approach to Layer One Efficiency, Part Two
Scott Avent, N6BGW 7th (1988)
Robert Finch, N6CXB

Network Implementation

Packet Radio at 19.2 kB -- A Progress Report
John Ackermann, AG9V 11th (1992)

Implementation of a 1Mbps Packet Data Link
Glenn Elmore, N6GN 8th (1989)
Kevin Rowett, N6RCE

Hubmaster: Cluster-Based Access to High-Speed Netowrks
Glenn Elmore, N6GN 9th (1990)
Kevin Rowett, N6RCE
Ed Satterthwaite, N6PLO

Recent Hubmaster Networking Progress in Northern California
Glenn Elmore, N6GN 9th (1990)
Kevin Rowett, N6RCE

The 56 kb/s Modem as a Network Building Block: Some Design Considerations
Barry McLarnon, VE3JF 10th (1991)

Digital Networking with the WA4DSY Modem - Adjacent Channel and Co-Channel Frequency Reuse Considerations
Ian McEachern, VE3PFH 10th (1991)

A Full-Duplex 56kb/s CSMA/CD Packet Radio Repeater System
Mike Chepponis, K3MC 10th (1991)
Lars Karlsson, AA6IW

A High Performance, Collision-Free Packet Radio Network
Phil Karn KA9Q 6th (1987)

Adaptation of the KA9Q TCP/IP Package for Standalone Packet Switch Operation
Bdale Garbee, N3EUA 9th (1990)
Don Lemley, N4PCR
Milt Heath

Hardware

The KISS TNC: A Simple Host-to-TNC Communications Protocol
Mike Chepponis, K3MC 6th (1987)
Phil Karn, KA9Q

The Ottawa Packet Interface (PI) A Syncrhonous Serial PC Interface for Medium Speed Packet Radio
Dave Perry, VE3IFB 10th (1991)

HAPN-2: A Digital Multi-Mode Controller fo the IBM PC
John Vanden Berg, VE3DVV 11th (1992)

The PackeTen system - The Next Generation Packet Switch
Don Lemley, N4PCR 9th (1990)
Milt Heath

```
##This is a SAMPLE ONLY ! Please modify per your own needs.
##This configuration has nothing fancy, just the basics to get you
going...
##MY COMMENTS start with ##, other comment lines with a single #
##are possible commands that might or might not be useful...
##
##Comments and blank lines, and extra (unneeded) tabs and spaces
##are automatically removed by CFG.EXE!

##set the console baudrate to 4800 bd (the default is 9600)
#cbaud 4800

##when in console mode, show autoexec.nos lines as they are executed
#verbose on

##---------------------
##SYSTEM VARIABLES

##minimum memory allocation size
memory minalloc 64

##the tcp/ip hostname
hostname jnos.wg7j

##the ip address. You can set per-interface ip addresses with
##the 'ifconfig' command. See further down.
ip address 44.26.1.17

##the ax.25 callsign. By defaut, this call is also used for our
## netrom call (if configured). This can be changed, see further
down
ax25 mycall wg7j-11

##ax.25 and net/rom alias. You can set a different netrom alias if
needed
##see further down
ax25 alias jnos40


##---------------------------------------
## AX.25 SYSTEM DEFAULTS
## If you want to change ax.25 parameters on a system wide basis,
## you need to do so BEFORE you attach any interface. After
attaching,
## you should use the 'ifconfig <iface> ax25' command to change
them.
## Values shown below (other then bctext) are the system defaults
```

```
##ax25 id broadcasting. Read the Fcc rules and do as you think is
correct :-)
##Every 10 minutes (default).
#ax25 bcinterval 600
##You can turn this off with
#ax25 bcinterval 0
##set the default broadcast text. Can be changed per interface.
ax25 bctext "JNOS for the Data Engine by Johan, WG7J"

##the backoff limit (number of times the retry timer is backed off
maximally)
#ax25 blimit 31

##initial round trip time estimate
#ax25 irtt 5000

##number of unacked outstanding frames
#ax25 maxframe 1

##wait a maximum of 30 secs for a retry timeout. (default)
#ax25 maxwait 30000

##the packet length
#ax25 paclen 256

##the poll threshhold
#ax25 pthresh 128

##the number of retries
#ax25 retries 5

##use linear backoff scheme (default)
#ax25 timer linear

##the t3 (keep alive) timer
#ax25 t3 0

##the connection redundancy timer (15 minutes)
#ax25 t4 900

##the protocol version
#ax25 version 2

##the window size
#ax25 window 2048

##---------------------------------------
## TCP SYSTEM DEFAULTS
```

```
## If you want to change tcp parameters on a system wide basis,
## you need to do so BEFORE you attach any interface. After
attaching,
## you should use the 'ifconfig <iface> tcp' command to change them.
## Values shown below are the system defaults

##maximum segment size; this is automatically adjusted to iface-mtu
- 40,
##or the mss value, wich ever is smaller...
#tcp mss 432

##initial round trip time estimate
#tcp irtt 5000

##the backoff limit (number of times the retry timer is backed off
maximally)
#tcp blimit 31

##the window size
#tcp window 864

##number of retries before failing...
#tcp retries 5

#tracing of tcp state changes only works in console-mode !
#tcp trace 0

##send a sync and possible data in one frame
#tcp syndata 1

##default to linear backoff, a bit more aggressive then exponential
#tcp timer linear

##set a maximum to the wait-for-reply time. Default is off, no limit
#tcp maxwait 0




##--------------------------
##ATTACHING INTERFACES

##serial port. A single TNC with a kiss eprom
##name = 'home', buffer = 1k, mtu 512, speed 9600Bd
attach 1 ax25 home 1024 512 9600
ifconfig home descr "KISS connection to development PC."


##if you want cts handshaking, see the docs,
```

```
##and add a 'c' as the last parameter.
#attach 1 ax25 home 1024 512 9600 c

##you can also attach it as  serial ip (SLIP)
#attach 1 slip home 1024 512 9600
#ifconfig home descr "SLIP connection to development PC."

##or as a NRS (Netrom Serial) to a NetRom Tnc
#attach 1 nrs home 1024 512 9600
##if more then one (ie. a diode matrix) add cts handshaking
#attach 1 nrs home 1024 512 9600 c
#ifconfig home descr "NRS to Net/Rom cluster"

##the interrupt buffer pool is used during receive of the radio
ports
##default is 10 buffer of 600 bytes each. See the docs for more
info.
memory nibufs 10
memory ibufsize 600

##port A (ie attach 2)
##using a speed of 1200 bd will allow both 9600 and 1200 bd modems
##to be interchanged WITHOUT reburning eproms! (see the docs as
well)
##The modem type is sensed automatically
##name is "430", mtu 256, speed 1200
attach 2 430 256 1200
ifconfig 430 descr "19k2 test on 430.55 MHz"

##if you want a different ip address for this port use
#ifconfig 430 ipaddress 1.2.3.4

##port B (ie attach 3)
##name is "2m", mtu 256, speed 1200.
attach 3 2m 256 1200
ifconfig 2m descr "1200Bd 144.92 MHz Corvallis LAN port."


##-------------------------------------------------
## NETROM System variables
## These are the defaults

##set the 'choke' state timeout
#netrom choketime 180000

##derate routes if connections fail
#netrom derate on

##don't emulate g8bpq nodes
```

```
#netrom g8bpq off

##don't show 'hidden' nodes in 'N' listing
#netrom hidden off

##initial connection round trip time
#netrom irtt 45000

##the window of outstanding data##the minimum quality routes to
accepted from broadcasts
#netrom minquality 10

##broadcast our routes every 30 minutes
#netrom nodetimer 1800

##keep a maximum of 3 different routes to a node
#netrom numroutes 3

##decrement the obsolecsence count every 30 minutes
#netrom obsotimer 1800

##the initial 'obsolescence' count (I didn't invent that :-) )
#netrom obsoinit 6

##the minimum count needed to broadcast a route
#netrom obsominbc 4

##don't accept 'weird' nodes in broadcasts...
#netrom promiscuous off

##set the transmit queue limit
#netrom qlimit 512

##set the maximum number of retries before failing
#netrom retries 3

##set the inactivity timeout, 15 minutes
#netrom tdisc 900

##use linear timers
#netrom timertype linear

##if you want to reach nodes many hops away, you might adjust the
ttl
#netrom ttl 10

## the sliding window size...
#netrom window 2
```

```
##------------------------------
## ACTIVATE NETROM INTERFACES


##the pseudo netrom interface. attached automatically !
#attach netrom

##if 'netrom mycall' isn't set, the ax25 call (ie wg7j-1) is used
#netrom mycall wg7j-1

##set the alias ! Only needed if 'ax25 alias' is not set...
#netrom alias jnos40

##'netrom interface <iface> <qual> <minbcqual>'
##activate the interfaces, set their qualities, and their minimum
##broadcast qualities. NOTE: this has changed from versions < 1.00 !
##if no minbcqual is given, the 'autofloor' value is used.
##if minqual < qual, in effect all routes are broadcast!
##if minqual = 0, NO routes are broadcast (but still our own id!)
##no verbose broadcasting on the local lan frequency (interface 2m)

netrom interface home 255 200
netrom interface 2m 191 0
netrom interface 430 224 175

##announce we're there
netrom bcnodes home
netrom bcnodes 2m
netrom bcnodes 430

##poll others for routes, this is automatically done when an
interface
##is activated !
#netrom bcpoll home
#netrom bcpoll 2m
#netrom bcpoll 430


##------------------------------
##CONVERS stuff

##the convers system name
convers hostname Corvallis

##message of the day
convers motd "This is a Data Engine running JNOS40,\nlocated in
Corvallis, OR, USA"
```

```
##an easy call for the local users to remember
convers mycall chat

##set an inactivity timeout for users. default is 2 hours
#convers t4 7200

##I link to a server in Ottawa, Canada at hs.ve3jf.ampr.org.
#make sure you have a route configured in the routing section!
#convers link 44.135.96.7

##if the link times out, retry wait a max of 10 minutes (600 secs)
#convers maxwait 600

##activate all interfaces for ax.25 convers call access
convers interface home on
convers interface 2m on
convers interface 430 on

##give mailbox users access to convers via the 'CONV' command
mbox convers on

##set maximum queue sizes for users and hosts. If the outstanding
data
##reaches this limit, the link is terminated. This keeps the system
##from wedging out...
##these are the defaults

##users
#convers umaxq 512

##host links
##note that if you have many links, and little ram (ie only 64k)
##this limit is on the high side; 1024 or less would be better!
#convers hmaxq 2048

##if you want filter out certain host links
#convers filter mode refuse
#convers filter 1.2.3.4
#convers filter 1.2.3.5

##ior, if you only want to accept certain host links
#convers filter mode accept
#convers filter 1.2.3.4
#convers filter 1.2.3.5

##if you want to refuse anything (host link, and tcp users)
##from certain ip addresses
#convers refuse 1.2.3.6
```

```
##start the tcp/ip convers server. NOT started automatically!
##you need to this for tcp users to access convers, and for others
to
##be able to link to you !
start convers



##-----------------------
## STARTING SERVERS

##start the servers. These are started by default,
##unless you changed it with CFG.EXE
#start ax25
#start netrom
#start telnet
#start finger
#start remote

##you can offcourse stop any of the above
#stop ax25
#stop netrom
#stop telnet
#stop finger
#stop remote

##NOTE: The following are NOT started automatically !
#start rip


##-----------------------------
## ROUTING
##now the hard part, the ip routes
##
##Note that i don't use rip or rspf, because we have a very
##limited number of tcp nodes and few users...
##thus, there are no examples either on how to use those...

##my 2nd Data Engine is on port 430
route add 44.26.1.18 430

##the local lan is on port 2m
route add 44.26.1/24 2m

##my 'home.wg7j' development system is on the serial port
route add 44.26.1.19 home

##add some netrom routes. First add an arp for their netrom callsign
```

```
arp add 44.116.0.48 netrom wa7gfe-3 netrom
##now add the actual route
route add 44.116.0.48 netrom 44.116.0.48

arp add 44.116.0.139 netrom ac7n-3 netrom
route add 44.116.0.139 netrom 44.116.0.139

arp add 44.24.0.8 netrom wb7dch-3 netrom
route add 44.24.0.8 netrom 44.24.0.8

##w0rli is the gateway for the other pdx area stuff
arp add 44.116.0.70 netrom w0rli-3 netrom
route add 44.116/16 netrom 44.116.0.70

##other traffic goes to the gateway, also on port 2m
route add default 2m 44.26.1.16

##here is a little trick. I want 'home.wg7j' to appear to be
directly
##on the local LAN ! We set up a proxy arp for it on the 2m lan
freq.
##such that others think our ax.25 call on the lan port goes with
the ip
##address for 'home.wg7j' . The above routes take care of the rest !
arp publish 44.26.1.19 ax25 wg7j-11 2m


##------------------------
## AX.25 CONFIGURATION

##regular digipeating can be enabled (default is off!)
ax25 digipeat home on
ax25 digipeat 2m on
ax25 digipeat 430 on

##broadcasting is ON by default, but can be turned off.
#ax25 bcport home off
#ax25 bcport 2m off
#ax25 bcport 430 off

##setup crossband digipeating. I use the interface name is the
##crossband digipeat callsign...
ifconfig home ax25 cdigi home
ifconfig 2m ax25 cdigi 2m
ifconfig 430 ax25 cdigi 430


##We can set AX.25 parameters per interface, if different from
defaults
```

```
##On the high speed port, set a different broadcast message.
ifconfig 430 ax25 bctext "19k2 Bd High Speed Port !"

##i want a shorter AX.25 Irtt
ifconfig 430 ax25 irtt 2000

##also a longer paclen
ifconfig 430 ax25 paclen 512

##and a > 1 maxframe
ifconfig 430 ax25 maxframe 4

#and a short maxwait before retry times out (10 secs)
ifconfig 430 ax25 maxwait 10000



##----------------------
## NODE CONFIGURATION & MESSAGES

##nodeshell sign-on message
motd "JNOS for the Data Engine."

##set some emergency information
ax25 ecall pwrdwn
etext "System is running on backup power. Please contact sysop!"

##the node shell 'I' command shows this
info "Data Engine Switch in Corvallis, OR. Run by Johan, WG7J"

##redirect sysop chat attempts to my home pc's tty-link listener
##default port is 87, the ttylink port
sysop 44.26.1.19 87

##use the local callsign server running Buckbook CD-Rom
##(with the default tcp port 1235)
callserver 44.26.1.3 1235

##set an alias to make local bbs access on port 2m easier !
mbox alias bbs "c 2m crvbbs"
##access a weather info server...
#mbox alias wx "f wx@1.2.3.4"

##set passwords
password 0123456789
remote password 0123456789
```

```
##-----------------------------
## DOMAIN SYSTEM SETUP

##setup a domain name server, if any. The last argument is the
timeout
##to use for the query (in seconds)
domain addserver 44.26.1.16 30

##if you have loaded a good size domain.txt,
##and want other to use your system as a DNS
#domain dns on

##also if you have loaded a domain.txt that should resolve
##most or all of the local ip->name translations, you might do:
#domain translate on


##---------------------------------------
## ICMP setup
##

##respond to echo requests (ping's)
#icmp echo on

##don't send a source quench when memory is low
#icmp quench off

##allow 'traceroutes' to show us. ie send a 'ttl exceeded' message
#icmp time on

#show icmp status tracing; only works on console mode
#icmp trace off



##------------------------
## HEARD LOGGING

##heard logging on all interfaces
##this is the default when attaching ax.25 interfaces
##you can turn it off
#ax25 hport home off
#ax25 hport 2m off
#ax25 hpport 430 off

##IP heard logging on all interfaces
##this is the default when attaching interfaces
#ip hport netrom on
#ip hport home on
```

```
#ip hport 2m on
#ip hport 430 on

##if you want to listen to arp's on the network
##turn on arp eaves dropping
arp eaves home on
arp eaves 2m on
arp eaves 430 on
```

Setting Bufsize, Paclen, Maxframe, MTU, MSS and Window

Many Nos users are confused by these parameters and do not know how to set them properly. This chapter will first review these parameters and then discuss how to choose values for them. Special emphasis is given to avoiding interoperability problems that may appear when communicating with non-Nos implementations of AX.25.

1. AX25 Parameters

1.1. Paclen

Paclen limits the size of the data field in an AX.25 I-frame. This value does not include the AX.25 protocol header (source, destination, digipeater addresses, control and pid bytes). The AX.25 V2 protocol specifies a maximum of 256 bytes for the paclen. Be aware that some AX.25 implementations can not handle paclen greater then this, however NOS derived systems can handle this situation. This may cause interoperability problems. Even Nos may have trouble with certain KISS TNCs because of fixed-size buffers. The original KISS TNC code for the TNC-2 by K3MC can handle frames limited in size only by the RAM in the TNC, but some other KISS TNCs cannot.
Since unconnected-mode (datagram) AX.25 uses UI frames, the paclen value has no effect in unconnected mode. IE. if you run IP in Datagram mode, you can still have an MTU > Paclen ! (As long as your receive buffers can handle the mtu size; see INTERFACE BUFFERS...)
In JNOS40, and JNOS v1.05 (and greater), paclen can be set on a per interface basis with the 'ifconfig <iface> paclen ###' command. Thus you can have a paclen of 256 on one interface and a smaller or larger on other interfaces. When you first attach an interface, the paclen defaults to the value of the 'ax25 paclen' setting (this defaults to 256.)
If you want to carry netrom data over ax.25 links, the system needs to make sure the paclen is large enough to handle the netrom MTU sized data. Net/rom mtu is a maximum of 236; with a 20 byte header for the routing, this gives a data size of 256 to be send with ax.25 packets.
In most versions of NOS.EXE, you can get problems when you use netrom and set the paclen < 256. When the ax.25 layer gets to send netrom frames, it cannot handle the whole data in one packet. It will then split it up in several smaller frames, using the AX.25 Version 2.1 fragmentation scheme. However, TheNet, Net/Rom, G8BPQ, MSYS and other nodes CAN NOT handle this type of fragmentation, resulting in impossible connections !

However, if you are running JNOS40, or JNOS v1.05 (or greater) for the PC, you need not worry about this. These 2 version of NOS have been modified to never provide more then paclen sized netrom data to the ax.25 layer ! Thus the above problem will never happen...

1.2. Maxframe

This parameter controls the number of I-frames that may be send on an AX.25 connection before it must stop and wait for an acknowledgement. Since the AX.25/LAPB sequence number field is 3 bits wide, this number cannot be larger than 7. It can be shown that the optimal value is 1.

Since unconnected-mode (datagram) AX.25 uses UI frames that do not have sequence numbers, this parameter does not apply to unconnected mode.

2. IP and TCP Parameters

2.1. MTU

The MTU (Maximum Transmission Unit) is an interface parameter that limits the size of the largest IP datagram that it may handle. The MTU is the sum of the size of the data inside the IP header (TCP of UDP most likely), and the size of the IP header itself. IP datagrams routed to an interface that are larger than its MTU are each split into two or more IP fragments. Each fragment has its own IP header and is handled by the network as if it were a distinct IP datagram, but when it arrives at the destination it is held by the IP layer until all of the other fragments belonging to the original datagram have arrived. Then they are reassembled back into the complete, original IP datagram. The minimum acceptable interface MTU is 28 bytes: 20 bytes for the IP (fragment) header, plus 8 bytes of data.

There is no default MTU in Nos; it must be explicitly specified for each interface as part of the 'attach' command. This is not the case for the netrom interface. Since ip data is carried inside a 'regular' netrom frame, the ip data should never be larger then the maximum data size the netrom protocol can handle. The default mtu here is 236, wich is the protocol maximum.

If you plan on running IP in Datagram mode (the default), you can have an MTU that is larger then the paclen for that interface. However, you should still be aware of the constraints of the receiver buffer size! (See INTERFACE BUFFERS)

If you plan on using IP in Virtual Connect, or VC, mode, you should be aware of the following. If you set the IP MTU larger then paclen for the interface, you will hand a packet to the ax.25 layer that is larger then what it can send in one packet. Thus you will get AX.25 V2.1 fragmentation. If you are exchanging ip data with NOS based stations only, you have no problems (other then the fragmentation). If you are interfacing with stations other then NOS bases systems, you again will have troubles, like with netrom. They will most

likely not be able to handle the packets correctly. Thus be aware that in this case you should set the mtu to equal the paclen, to avoid these problems.

2.2. MSS

MSS (Maximum Segment Size) is a TCP-level parameter that limits the amount of data that the remote TCP will send in a single TCP packet. MSS values are exchanged in the SYN (connection request) packets that open a TCP connection. In the Nos implementation of TCP, the MSS actually used by TCP is further reduced in order to avoid fragmentation at the local IP interface. That is, the local TCP asks IP for the MTU of the interface that will be used to reach the destination. It then subtracts 40 from the MTU value to allow for the overhead of the TCP (20 bytes) and IP (20 bytes) headers. If the result is less than the MSS received from the remote TCP, it is used instead.

2.3. Window

This is a TCP-level parameter that controls how much data the local TCP will allow the remote TCP to send before it must stop and wait for an acknowledgment. The actual window value used by TCP when deciding how much more data to send is referred to as the effective window. This is the smaller of two values: the window advertised by the remote TCP minus the unacknowledged data in flight, and the congestion window, an automatically computed time-varying estimate of how much data the network can handle.

2.4. Discussion

2.4.1. IP Fragmentation vs. AX.25 Segmentation

IP-level fragmentation often makes it possible to interconnect two dissimilar networks, but it is best avoided whenever possible. One reason is that when a single IP fragment is lost, all other fragments belonging to the same datagram are effectively also lost and the entire datagram must be retransmitted by the source. Even without loss, fragments require the allocation of temporary buffer memory at the destination, and it is never easy to decide how long to wait for missing fragments before giving up and discarding those that have already arrived. A reassembly timer controls this process. In Nos it is (re)initialized with the 'ip rtimer' parameter (default 30 seconds) whenever progress is made in reassembling a datagram (i.e., a new fragment is received). It is not necessary that all of the fragments belonging to a datagram arrive within a single time-out interval, only that the interval between fragments be less than the time-out.

Most commercial sub networks that carry IP have MTUs of 576 or more, so interconnecting them with sub networks having smaller values can result in considerable fragmentation. For this reason, IP implementors working with links or subnets having unusually small packet size limits are encouraged to use transparent fragmentation, that is, to devise schemes to break up large IP datagrams into a sequence of link or subnet frames that are immediately reassembled on the other end of the link or subnet into the original, whole IP datagram without the use of IP-level fragmentation.

Such a scheme is provided in AX.25 Version 2.1. It can break a large IP or NET/ROM datagram into a series of paclen-sized AX.25 segments (not to be confused with TCP segments), one per AX.25 I-frame, for transmission. Subsequently, it can reassemble them into a single datagram at the other end of the link before handing it up to the IP or NET/ROM module.

Unfortunately, the segmentation procedure is a new feature in AX.25 and is not yet widely implemented; in fact, Nos and derived software, is so far the only known implementation. For regular NOS systems this can create some interoperability problems between Nos

and non-Nos nodes. However, JNOS40 and JNOS 1.05 (or later) have provisions built in to avoid these problems. This problem is discussed further in the section on setting the MTU.

2.4.2. Setting MTU

TCP/IP header overhead considerations similar to those of the AX.25 layer when setting paclen apply when choosing an MTU. However, certain sub network types supported by Nos have well-established MTUs, and these should always be used unless you know what you're doing: 1500 bytes for Ethernet, and 508 bytes for ARCNET. The MTU for PPP is automatically negotiated, and defaults to 1500. Other subnet types, including SLIP and AX.25, are not as well standardized.

SLIP has no official MTU, but the most common implementation (for BSD UNIX) uses an MTU of 1006 bytes. Although Nos has no hard wired limit on the size of a received SLIP frame, this is not true for all other systems. Interoperability problems may therefore result if larger MTUs are used in Nos.

Choosing an MTU for an AX.25 interface is more complex. When the interface operates in datagram (UI-frame) mode, the paclen parameter does not apply. The MTU effectively becomes the paclen of the link.

However, when using AX.25 CONNECTIONS to send IP packets, data will be automatically segmented into I-frames no larger than paclen bytes. Unfortunately, as also mentioned earlier, Nos is so far the only known implementation of the new AX.25 segmentation procedure. Thus, if you are exchanging IP over connections (i.e. VC mode) with none-NOS based systems, it might be needed to avoid AX.25 segmentation. Thus you should make sure that packets larger than paclen are never handed to AX.25. In order to assure this, you should not set the MTU greater then the paclen.

On the other hand, if you do not send IP over connections, but in datagram mode, you can use a larger MTU. Here, you are limited by the receive buffer size (and the tolerance of other network users for your large packets and proportionally greater bandwidth share !).

For connections carrying IP between NOS systems (i.e. NOS-NOS VC mode), you can let AX.25 segmentation do it's thing. If you choose an MTU on the order of 1000-1500 bytes, you can largely avoid IP-level fragmentation and reduce TCP/IP-level header overhead on file transfers to a very low level. And you are still free to pick whatever paclen value is appropriate for the link.

2.4.5. Setting MSS

The setting of this TCP-level parameter is somewhat less critical than the IP and AX.25 level parameters already discussed, mainly because it is automatically lowered according to the MTU of the local interface when a connection is created. Although this is, strictly speaking, a protocol layering violation (TCP is not

supposed to have any knowledge of the workings of lower layers) this technique does work well in practice. However, it can be fooled; for example, if a routing change occurs after the connection has been opened and the new local interface has a smaller MTU than the previous one, IP fragmentation may occur in the local system.
The only drawback to setting a large MSS is that it might cause avoidable fragmentation at some other point within the network path if it includes a "bottleneck" subnet with an MTU smaller than that of the local interface. (Unfortunately, there is presently no way to know when this is the case.
There is ongoing work within the Internet Engineering Task Force on a "MTU Discovery" procedure to determine the largest datagram that may be sent over a given path without fragmentation, but it is not yet complete.) Also, since the MSS you specify is sent to the remote system, and not all other TCPs do the MSS-lowering procedure yet, this might cause the remote system to generate IP fragments unnecessarily.
On the other hand, a too-small MSS can result in a considerable performance loss, especially when operating over fast LANs and networks that can handle larger packets. So the best value for MSS is probably 40 less than the largest MTU on your system, with the 40-byte margin allowing for the TCP and IP headers. For example, if you have a SLIP interface with a 1006 byte MTU and an Ethernet interface with a 1500 byte MTU, set MSS to 1460 bytes. This allows you to receive maximum-sized Ethernet packets, assuming the path to your system does not have any bottleneck subnets with smaller MTUs.

2.4.6. Setting Window

A sliding window protocol like TCP cannot transfer more than one window's worth of data per round trip time interval. So this TCP-level parameter controls the ability of the remote TCP to keep a long "pipe" full. That is, when operating over a path with many hops, offering a large TCP window will help keep all those hops busy when you're receiving data. On the other hand, offering too large a window can congest the network if it cannot buffer all that data. Fortunately, new algorithms for dynamic controlling the effective TCP flow control window have been developed over the past few years and are now widely deployed. Nos includes them, and you can watch them in action with the 'tcp status <tcb>' or 'socket <#>' commands. Look at the cwind (congestion window) value. In most cases it is safe to set the TCP window to a small integer multiple of the MSS, (e.g. 4times), or larger if necessary to fully utilize a high bandwidth*delay product path. One thing to keep in mind, however, is that advertising a certain TCP window value declares that the system has that much buffer space available for incoming data. Nos does not actually pre-allocate this space; it keeps it in a common pool and may well overbook" it, exploiting the fact that many TCP connections are idle for long periods and

gambling that most applications will read incoming data from an active connection as soon as it arrives, thereby quickly freeing the buffer memory. However, it is possible to run Nos out of memory if excessive TCP window sizes are advertised and either the applications go to sleep indefinitely (e.g. suspended Telnet sessions) or a lot of out-of-sequence data arrives. It is wise to keep an eye on the amount of available memory and to decrease the TCP window size (or limit the number of simultaneous connections) if it gets too low.

Depending on the channel access method and link level protocol, the use of a window setting that exceeds the MSS may cause an increase in channel collisions. In particular, collisions between data packets and returning acknowledgments during a bulk file transfer may become common. Although this is, strictly peaking, not TCP's fault, it is possible to work around the problem at the TCP level by decreasing the window so that the protocol operates in stop-and-wait mode. This is done by making the window value equal to the MSS.

2.5. Summary

In most cases, the default values provided by JNOS40 for each of these parameters will work correctly and give reasonable performance. Only in special circumstances such as operation over a very poor link or experimentation with high speed modems should it be necessary to change them.

NOS supports a number of versions of the attach command to deal with different hardware. We'll discuss three of them here: asy, used for serial port connections; pi, used to connect to the Ottawa PI card; and packet, used to interface to hardware supporting the FTP, Inc., packet driver protocol. As usual, this discussion covers the basics; see the JNOS/JNOS40 Commands Manual or the NOS reference manual for more information on all the many options.

Hosts normally have a separate IP address for each interface. If you are running more than one interface, you can include that interface's IP address (in [xx.xx.xx.xx] form) at the end of the attach command.

The asy version provides an interface to a standard PC serial port. The syntax is:

attach asy <ioaddr> <vector> <mode> <if> <bufsize> <mtu> <speed>

In English, these parameters are:

ioaddr -- the address of the COM port being used. COM1 is usually 0x3f8 and COM2 is usually 0x2f8. COM3 and COM4 aren't standardized; using them will require looking at the documentation for your serial card, and probably some experimentation.

vector -- the IRQ used by the hardware. COM1 is usually 4, and COM2 is usually 3. Again, COM3 and COM4 vary.

mode -- this specifies the nature of the interface. ax25 is for a connection to a KISS TNC, slip for a hardwired connection to another host, ppp for a dial-up connection, and nrs is for attaching a NOS station to a NetRom node.

if -- the interface name. The convention is to use ax0, ax1, etc., for KISS interfaces.

bufsize -- the buffer for incoming data, in bytes. Usually a value of 1024 is more than sufficient for a 1200 baud channel.

mtu -- the maximum transmission unit size, in bytes. See the discussion in the main text on this subject.

speed -- the speed of the serial (not radio) link, in baud. The best setting for this will depend on the speed of your computer, but generally two to four times the radio speed is adequate.

Some sample attach asy commands are:

```
# COM1, KISS TNC as ax0, MTU 256, 4800 BAUD
attach asy 0x3f8 4 ax25 ax0 1024 256 4800

# COM2, KISS TNC as ax1, MTU 256, 2400 BAUD
attach asy 0x2f8 3 ax25 ax1 1024 256 2400

# SLIP link, COM1 as sl0, MTU 256, 9600 BAUD
attach asy 0x3f8 4 slip sl0 1024 256 9600
```

The Ottawa PI card is a plug-in board for PCs designed for high-speed performance. It has two ports, one DMA driven for high speed and the other interrupt driven. The attach syntax is:

attach pi <ioaddr> <vector> <DMA chn> <mode> <name> <bufsize> <mtu> <speed a> <speed b>

A sample attach command (using the PI's default jumper settings) is:

attach pi 380 7 1 ax25 pi0 1750 1024 0 1200

In this example, the interface name for the DMA port is "pi0a" and the second port is "pi0b". Because the port a speed is 0, the PI card expects the modem to provide its own clocking. The PI attach syntax is explained in the manual provided with the card.

Finally, the packet interface is used to connect to ethernet cards and other hardware that supports the FTP, Inc. "packet driver" standard. There's a packet driver for the PI card. The syntax is:

attach packet <ioaddr> <vector> <if> <bufsize> <mtu>

In this case, ioaddr and vector need to match those used for the packet TSR that supports the hardware. bufsize is the number of packets (not bytes) that may be outstanding. For ethernet, the standard mtu is 1500.

Once NOS is installed and your configuration files set, you need to do one more thing: get your TNC talking to your computer in KISS (Keep It Simple, Stupid) mode. KISS is a special protocol that lets your computer do the work of processing packets; the TNC does only the very low-level packet assembly and disassembly functions. Nearly all TNCs support KISS in one way or another.

Typically, you'll need to issue commands to the TNC to set the serial line baud rate to the same speed as you've specified in the attach command, to 8 bit data, and to no parity. Then, issue the KISS command (on a TNC2, kiss on), and the TNCs software reset command. After that, you won't be able to talk to your TNC via the terminal program, but NOS will be able to. (And don't worry, you can easily return the TNC to normal mode if you want to.) Once you've done this, you're set to run NOS.

The 'comm' command will send a string of text to the named interface allowing you to send ascii commands to your TNC. For example, to force a Kantronics DataEngine or KAM into KISS mode every time you start NOS, include the following commands in AUTOEXEC.NOS (after you've defined the interface with the attach command):

```
comm ax0 "interface kiss"
comm ax0 "reset"
```

Note that surrounding the text with quote characters will preserve spaces in the command.

APPENDIX E BBS Sites and Internet Conferences

All of the files in the distribution package are available on several anonymous FTP servers on the Internet as well as many land-line BBSes. Places where you may expect to always find current releases are:

Internet:

ftp.ece.orst.edu	directory	pub/ham/wg7j/jnos40
ucsd.edu	directory	hamradio/packet/tcpip/wg7j

Landline:

CyberGate	(316) 688-0915
N8EMR's Ham BBS	(614) 895-2553
ChowdaNet	(401) 331-0334

There is one Internet conference which focuses on development and use of the JNOS variants of Karn's NOS.EXE.

nos-bbs@hydra.carleton.ca

You may subscribe to this conference by sending an email to

nos-bbs-request@hydra.carleton.ca

In the body of the message put one line:

"subscribe nos-bbs(-digest) [address]"

If you add the word "-digest", you will receive the daily summary of messages to the conference instead of copies of all messages. If you want the conference sent to an email address DIFFERENT from the one where you originate the subscription request, specify that different address as shown.

There is also the the advanced networking topics group

tcp-group@ucsd.edu.

You can subscribe to this group by sending an email to:

listserv@ucsd.edu

In the body of the message, enter on one line:

subscribe tcp-group(-digest) [address]

This works as described above.

To UNSUBSCRIBE, use the same format described above with the one word change.