TheNetNode

V1.79mh05

Made by

Nord><Link

Ausgabe: 18. Oktober. 2006

Benutzerhandbuch zu der Packet-Radio Knotensoftware

TheNetNode (GO32)/(Linux)/(TNC3) V 1.79mh05

Änderungen gegenüber der Beschreibung von der Version TheNetNode 1.78

(Ausgabe Oktober 1999)

1.) Neuer Sysop-Befehl: ALIAS. Damit kann man Aliaskommandos erstellen.

2.) Beim Portbefehl ist die Option MAXCON hinzugekommen.

3.) EAX25-Modus möglich.

4.) Bei der Linuxversion ist das Kernelinterface neu. Befehl ist KERN.

5.) Auto-IPR wurde erweitert.

6.) Neues L1-Interface: 6PACK (vorerst nur unter Linux).

7.) Connect im FlexNet-Stil (c -3 etc.) möglich.

8.) Interaktive Shell unter Linux.

9.) TNN läuft jetzt auch auf MIPS-Systemen wie dem MeshCube oder dem WRT54G(S).

Benutzerhandbuch zu der Packet-Radio Knotensoftware

TheNetNode V 1.79mh05

Nachdruck mit Quellenangabe für nicht kommerzielle Zwecke erlaubt.

Inhaltsverzeichnis

Änderungen gegenüber der Beschreibung von der Version TheNetNode 1.78 2

Vorwort: 7

Das Programmpaket TheNetNode: 7

BAYCOM SCC-KARTEN: 7
DER TOKENRING 7
DER KISSLINK 8
DIE VANESSAKARTE 8
6PACK 9
EXTERNE TREIBER 9
ETHERNET-SCHNITTSTELLE: 10
DPMI ODER NUTZUNG DES MEIST BRACHLIEGENDEN SPEICHERS ÜBER 1MB: 11

TheNetNode-Programm: 11

DAMA ... VERKEHRSREGELUNG DURCH DEN KNOTEN: 11
DAMA-Ablaufsteuerung: 11
SYSOP-ARBEITEN AM KNOTEN: 12
USER-PASSWORT-FUNKTION 12
RECHNERKONFIGURATION: 13
AUTOEXEC.BAT 13
START.BAT 13
TNN179.PAS 14
TNN179.TNB 16

SYSOP-Befehle: 19

(**ALIAS**) 19
(**AX**)IPR 19
(**BE**)ACON 19
(**CL**)EAR 21
(**CONV**)ERS 21
(**DCD**) 22
(**DOS**) <KOMMANDO> 22
(**DX**)CLUSTER 22
(**E**)DIT 22
(**K**)ILL 23
(**L3)**MHEARD 23
(**L**)INKS 24
(**LOA**)D <FILENAME.EXT> 24
(**MH**)EARD 24
(**P**)ARMS 25
(**PAC**)SAT 25
(**PR**)OMPT 29
(**RE**)AD 29
(**RES**)ET SYSTEM 30
(**RES**)ET <PORT> 30
(**RU**)BATCH 30
(**SH**)ELL <KOMMANDO> 30
(**S**)TAT 30

(**SP**)ARM ODER BESSER **S**AVE **P**ARAMETER 30
(**STAR**)T <PROGRAMM> 30
(**SUS**)PEND 30
(**SY**)SOP 31
(**TE**)ST <PORT> 31
(**TI**)ME 31
(**TR**)ACE 31
TCPIP KOMMANDOS 33
(**ARP**) 33
(**IPR**)OUTE 34
ESC/ALT KOMMANDOS 35
DIE BESONDERE <ALT> X KOMBINATION 35
DIE VERSCHIEDENEN <ESC> TASTEN 35
(HOST)MODE-BEFEHLE 36

(########.TNB) FILES 38

(RU)NBATCH <FILENAME.TNB> 38
(**O**)UTPUT 40
(**CWER**) 40
(**CPERS**) 40
(**SETCALL**) 41
(**SH**)OWSYS 41
(**MSY**) 41
(**SYSH**)ELP 42
(**TOP**) 42
(**STARTCNT**) 42

HILFSPROGRAMME für den SYSOP zu Hause: 44

(MAKEDAT.EXE) 44
(TNNSET.EXE) 44

Befehle für alle User 45

(**6**)PACK 45
(**AX**)IPR 45
(**BE**)ACON 45
(**C**)ONNECT 45
(**CONV**)ERS UND SEINE BEFEHLE 46
(**CQ**) 50
(**D**)EST 51
(**DX**)CLUSTER 52
(**G**)RAPH 52
(**H**)ELP 55
(**L3)**MHEARD 55
(**L**)INKS 56
(**M**)AILBOX 56
(**MH**)EARD 56
(**N**)ODES 57
(**P**)ARAMETER 60
(**PAC**)SAT 61
(**PI**)NG 61
(**PO**)RT * oder (**PO**)RT + ... 62
(**Q**)UIT 63
(**R**)OUTES 64
(**R ***) oder (**R +) =** Routes Version 65
(S)TAT 65
(**TA**)LK 69
(**TI**)ME 69
(**U**)SER 69
(**V**)ERSION 73
(**V**)ERSION * (**V**)ERSION + ... 73

EXTERNE Programme für alle User 75

DIGIMAIL: 75
(**MSG**) 75
LOCATORBERECHNUNG: 75
(**QTH**) 75
SATELLITEN-STANDORT-BERECHNUNG: 76
(**SAT**) 76
(**TOP**) 76

VISITENKARTE: 77
(SAV)ECALL 77
(SH)OWCALL 77
ONLINEHILFE: 78
(H)ELP 78

ANHANG A: 79

DIE VERSCHIEDENEN TEXTDATEIEN UND IHRE BEDEUTUNG 79

ANHANG B: 80

TNC2C UND THENETNODE 80
KISS-SOFTWARE FÜR DEN TNC IM TOKENRING UND WATCHDOG: 81

ANHANG C: 83

VANESSA 83

ANHANG D: 89

(ETHERNET) ...WAS MUSS ICH TUN ? 89

ANHANG E: 92

DER PACSAT-BROADCAST-SERVER VON TNN 92

ANHANG F: 94

AUFBAU DER LAYER 3 / 4 FRAMES 94

ANHANG G: 98

DIE GO32-VERSION VON TNN 98

ANHANG H: 101

D A M A - EIN NEUES VERFAHREN FÜR PACKET-RADIO ! 101

ANHANG I: 105

TheNetNode 1.79 unter Linux 105

WAS IST NEU SEIT VERSION 1.78 ? 105
WELCHE MÖGLICHKEITEN BIETET DIE SOFTWARE 105
WAS GEHT NICHT? 106
ANFORDERUNGEN AN DIE HARDWARE 106
INSTALLATION VON LINUX 106
SERIELLE SCHNITTSTELLEN UNTER LINUX 106
ZUGRIFFSRECHTE 107
VERZEICHNISSE UND DATEIEN FÜR TNN 107
INSTALLATION VON TNN 107
DIE DATEI TNN.INI 108
DIE DATEI TNN179.PAS 109
DIE DATEI TNN179.TNB 109
BEFEHLE DER LINUX-TNN 109
AUTOMATISCHER BETRIEB VON TNN 110
EXTERNE PROGRAMME 111
Verbindung zum Hostmodus 111
Verbindung über Pseudo-Terminals 112
TNC-EPROMS 113
KRITIK, PROBLEME, FEHLERMELDUNGEN 113
LITERATUR 113

ANHANG J: 117

TCP/IP UND TNN 117
Hier noch ein paar Begriffserklärungen verfasst von DL8XAS: 120

ANHANG K: 124

AX.25 LINK-LAYER PROTOKOLL SPEZIFIKATION 124
Vorwort 124
1. Bedingungen für die Verbindung der Ebene 1 124
1.1 Frequenzen oberhalb von 144 MHz 124
1.2 Frequenzen unterhalb von 30 MHz 124
2. AX.25 Link-Layer Protokoll Spezifikation 124
2.1 Rahmen und Anwendungbereich 124

2.2 Aufbau des Datenübertragungsblocks ... 125
2.2.1 Blockbegrenzung (Flag) ... 125
2.2.2 Adress-Feld ... 125
2.2.3 Kontrollfeld ... 125
2.2.4 PID-Feld ... 125
2.2.5 Informations-Feld ... 126
2.2.6 Bit-Stuffing ... 126
2.2.7 Blockprüfzeichenfolge ... 126
2.2.8 Übertragunsreihenfolge der Bits ... 126
2.2.9 Ungültige Blocks ... 126
2.2.10 Block-Abbruch ... 126
2.2.11 Füllzeichen zwischen Blocks ... 126
2.2.12 Zustände am Übermittlungsabschnitt ... 126
2.2.13 Kodierung des Adressfeldes ... 127
2.3 Kenngrössen des Steuerungsverfahrens ... 128
2.3.1 Elemente des Verfahrens ... 128
2.3.2 Formate des Kontrollfeldes und der Folgezähler ... 129
2.3.3 Aufgabe des Poll/Final-Bits ... 130
2.3.4 Kommandos und Meldungen ... 130
2.3.5 Meldung von Fehlern und deren Berichtigung ... 132
2.4 Beschreibung der Übermittlungsvorschrift ... 133
2.4.1 Vorschriften für die Adressierung ... 133
2.4.2 Vorschriften für das P/F-Bit ... 134
2.4.3 Vorschriften für den Auf- und Abbau einer Verbindung ... 134
2.4.4 Vorschriften für die Datenübermittlung ... 135
2.4.5 Zustände nach Rückweisung eines Blocks ... 137
2.4.6 Rücksetzen der Verbindung ... 137
2.4.7 Liste der Systemparameter ... 137

Versionshistory: ... 142

1.78NK01 ... 142
1.78NK02 ... 142
1.78MH01 ... 142
1.78MH03 ... 143
1.78MH04 ... 143
1.78MH05 ... 144
1.78MH06 ... 144
1.78MH07 ... 145
1.78MH08 ... 145
1.78MH09 ... 146
1.78MH10 ... 146
1.79PRE1 ... 147
1.78OR01 ... 147
1.78OR02 ... 147
1.78OR03 ... 147
1.79PRE2 ... 148
1.79PRE3 ... 148
1.79PRE4 ... 148
1.79PRE5 ... 148
1.79PRE6 ... 149
1.79PRE7 ... 149
1.79PRE8 ... 149
1.79 ... 149
1.79MH01 ... 149
1.79MH02 ... 150
1.79MH03 ... 151
1.79MH04 ... 152

LIZENZ: ... 155

ALLGEMEINE LIZENZ FÜR AMATEURFUNK SOFTWARE (ALAS) ... 155

ABSCHLUSS: ... 157

Vorwort:

TheNetNode wird von **Funkamateuren** für **Funkamateure** geschrieben, die sich bereit erklären, die nicht geringe finanzielle und zeitraubende Belastung auf sich zu nehmen, einen Packet-Knoten aufzubauen, zu betreiben und zu verlinken. TheNetNode erhebt keinen Anspruch darauf, FEHLERFREI zu sein. Aber die Softwareschreiber sind bemüht, die ihnen bekannten Fehler, die auch nachvollziehbar sein müssen, zu beseitigen. Deshalb gibt es über das Jahr verteilt immer wieder kleinere Updates. Weiteres bitte aus der ALAS am Schluss dieses Dokumentes entnehmen.

Die Programmversion für den Atari (ST) wird nun von Odo, DL1XAO, weitergeführt. Leider aber lassen sich die zukünftigen Erweiterungen von TNN dort nicht mehr alle realisieren. (z.B.: Einsatz von Vanessakarten o.ä.)

Die TheNetNode Version, für den TNC3 (von DK9SJ entwickelt) mit der 68302 CPU, wird ebenfalls von Odo DL1XAO gepflegt und beim Erscheinen einer neuen PC-Version kurzfristig zur Verfügung gestellt. Die Linux-Version von TheNetNode wird von Marc-André, DG9OBU, gepflegt, um die DOS/GO32-Version kümmert sich Bernd, DG8BR.

TheNetNode ist somit auf den verschiedensten Hardwareplattformen zu Hause. Die damit verbundene Flexibilität dürfte jedem Betreiber eines Knotens entgegenkommen.

Dokumentation: Sie enthält in der Masse deutsche, eingedeutschte- und auch englische Begriffe. Dies soll an dem Wort Paßwort (alte deutsche Schreibweise oder Passwort (neue deutsche Schreibweise) was sich im englischen als Password schreibt erklärt werden. In der TNN179.PAS steht natürlich allgemein gehalten der englische Begriff. Der Rest der Dokumentation ist natürlich in Deutsch, was natürlich nur ein Passwort erlaubt. Der Begriff in GROSSCHRIFT lässt nur ein PASSWORT zu. Dies ist hoffentlich Verständlich.

Das Programmpaket TheNetNode:

TheNetNode ist ein Softwarepaket, in dem das gesamte Knotenprotokoll abgearbeitet wird, z.B.:

- Up - und Down-Link zu Usern mit oder ohne DAMA-Protokoll
- Interlink-Protokoll mit Routing (NET/ROM, INP, FlexNet)
- Benutzeroberfläche (Hilfen und Informationen)
- Einbinden verschiedenster Schnittstellen
- Zugriff auf externe Programme wie QTH- und SAT-Berechnungen, Callbook

um nur einiges zu nennen.

Dem Programm stehen für die Kommunikation mit der Außenwelt verschiedene Schnittstellen zur Verfügung wie z.B.:

- Serielle I/O mit Tokenring Protokoll (9k6 bis 115kBit, bei Linux bis 230kBit)
- Maximal 4 Serielle I/O (bei Linux maximal 16) mit Kisslink-Protokoll (ähnlich Tokenring)
- Parallele Schnittstelle zum Frontendrechner Vanessa (bis 140 Kbit/s je Interlink)
- Parallele Schnittstelle zum Ethernet (10 MBit/s) mit externen Treibern möglich
- BayCom USCC mit internem Treiber
- unter Linux Verwendung des Kernel-AX.25 Interfaces in der normalen Version und in der Variante von DG1KJD

BAYCOM SCC-Karten:

TheNetNode unterstützt über einen integrierter SCC-Treiber die Karten USCC, OptoSCC und DSCC (Digi-SCC BayCom). Es ist nur möglich, **EINE** Karte mit maximal 8 Ports anzusteuern.

Hierzu muss im Enviroment eingetragen werden:

SET USCC=base_hex,irq_dez für USCC(BayCom) oder
SET DSCC=base_hex,irq_dez für DSCC(BayCom) oder
SET OSCC=base_hex,irq_dez für OptoSCC

In TheNetNode kann dann die jeweilige Hardware (USCC0...x, OSCC0...x, DSCC0.xx) an einen beliebigen Port gebunden werden:
PORT 4 USCC2

Die Konfiguration erfolgt über den Mode-Parameter.
DF9IC -> MODE=9600rtz
AFSK -> MODE=1200c

Der SCC-Treiber ist kein neues Feature, läuft aber endlich vernünftig.

Die Baycom PCISCC4-Karte von Jens, DG1KJD, wird unter Linux über sein Kernel-AX.25 unterstützt.

Der Tokenring

Um über eine Rechnerschnittstelle mehrere HF-Ports bedienen zu können, wurde der Tokenring eingeführt, der auch über etliche Jahre eine preiswerte und ausreichende Schnittstelle zur Verfügung stellte.

Die TNC bedienen mit einer speziellen KISS-Soft die Funkseite, als da sind Sendertastung, Daten senden, Daten empfangen, zum Rechner weiterleiten und von ihm empfangen.

Für die Weiterleitung wird ein Tokenring eingesetzt. Der Rechner sendet ein sogenanntes Token (bestimmte Bytefolge) auf den Ring. Da alle TNC in einer Reihenschaltung hintereinander angeordnet sind, kommt es bei dem ersten TNC im Ring an. Hat der nun auf der Funkseite etwas empfangen, so hält er das Token zurück, sendet stattdessen die soeben empfangenen Daten an seinen Ringnachbarn und hängt an diese Daten das Token wieder an. Da jeder TNC immer auf das Token wartet, bevor er Daten auf den Ring legt, lässt jeder Nachbar-TNC die Daten vom vorhergehenden TNC unverändert passieren. Sowie er aber das Token erkennt, hält auch er es zurück und legt seine eigenen Daten auf den Ring und dann erst wieder das Token. Irgendwann, je nach Anzahl an TNC, kommen die Daten nun beim Rechner an.

Hier wird nun das TheNet-Protokoll abgearbeitet und die Daten werden, mit einer Port-Adresse versehen, wieder an ein Token angehängt und vom Rechner auf den Ring gesendet. Die TNC prüfen aber auch, ob bei den Daten, die sie empfangen, welche dabei sind, die an sie bestimmt sind (Portnummer). Diese werden von dem Token abgehängt und auf der Funkseite mit den entsprechenden Parametern ausgesendet. Das ganze Verfahren scheint auf den ersten Blick einiges langsamer zu sein, oder auch nicht schneller als das Vernetzen der TNC untereinander mit Diodenmatrix. Dem ist aber nicht so! Man kann eine deutlich spürbare Geschwindigkeitssteigerung feststellen! Woran liegt das? Einmal daran, dass der Tokenring immer zu annähernd 100% ausgelastet ist, Kollisionen zwischen den TNC nicht möglich sind und keiner der TNC Warteprozeduren ausführen muss, um Kollisionen zu vermeiden. Aber der Hauptgrund der Geschwindigkeitssteigerung liegt darin, dass der Z80 im TNC mit Hilfe des KISS-Programmes bei weitem weniger zu tun hat als mit dem kompletten TheNet. Dieses Programm besteht fast nur aus Interruptroutinen (z.B. Umsetzung Synchron- auf Asynchron-Übertragung), welche die SIO bedienen und die Pakete zwischen V24 und Funkschnittstelle verteilen. Es wird keine unnötige Rechenzeit damit vergeudet, sich um den Inhalt der Pakete zu kümmern. Dazu ist ja der Atari-ST bzw. der PC da, der das bei weitem schneller kann. Die Aufarbeitung der Pakete und die Behandlung der einzelnen AX.25-Protokollebenen läuft auch deshalb weit schneller, weil sie trotz mehrerer angeschlossener TNC nur einmal erfolgt und nicht, wie vom TNC-TheNet gewohnt, in jedem TNC separat für sich.

Die TNC sollten mit 9,8304 MHz Quarzen und 10 MHz SIO + CPU ausgestattet sein.

Im Übrigen wird die KISS-Software (allerdings ohne die Token-Ring-Erweiterung) schon seit langem bei den WAMPES-Knotenrechnern, dem TCP/IP Packet von KA9Q und auch dem Packet-Terminalprogramm SUPERKISS mit Erfolg verwendet.

Auf der Schnittstelle, beim PC muss sie nun mit SET TOKENCOM=<com Nr.> definiert werden, sind Übertragungsraten von 9.600, 19.200, 38.400 Bit/Sec möglich. Mit dem FIFO-Schnittstellenbaustein 16550 AFN von National als RS232 auch darüber hinaus noch 57.600 und 115.200 Bit/Sec. Das Programm erkennt diesen Baustein selbständig (Anzeige beim Hochfahren des Rechners und auch bei Abrufen der Statistik ersichtlich) und wendet diesen Mode an. Vereinfachte Arbeitsweise: Es werden in dem 16550 AFN bei TNN bis zu 8 Byte zwischengespeichert und erst dann gibt es für den Rechner einen Interrupt. Intern im Chip ist natürlich ein Timer, der den Datenfluss überwacht und auch ggf. mal z.B. 3 Byte weiterreicht.

Vorteil des Ringes ist, jeder TNC sendet nur an EINEN anderen Empfänger oder andersherum betrachtet, jeder TNC empfängt nur von EINEM anderen Sender Daten. Das bedeutet, dass Pegelprobleme, wie sie beim Parallelbetrieb von mehreren TNC auf einer gemeinsamen Diodenmatrix auftreten, hier nicht vorkommen.

Doch Vorteile haben meist auch Nachteile. Fällt in einem Ring auch nur ein TNC aus, so fällt, wenn man keine Vorsorge trifft, der Ring und somit der gesamte Knoten aus. Ein weiterer Nachteil ist, dass sich die Baudrate des Tokenring auf die einzelnen Funkstrecken aufgeteilt wird.

Bei Verwendung der GO32 Programme sind FIFO-Bausteine in den COM-Schnittstellen PFLICHT!

Informationen zur TNC-Software und zu Watchdog siehe Anhang B.

Der Kisslink

Da immer wieder Drahtanbindungen zu anderen Systemen wie Mailboxen, Cluster oder anderen Knoten geschaffen werden müssen, verfügt TheNetNode auch über eine KISS-Schnittstelle. Dieses KISS-Protokoll darf aber nicht mit dem anfangs beschriebenen speziellen TOKENRING-KISS verwechselt werden. Die KISS-Schnittstelle kann mit SET KISS1=1 bis SET KISS4=4 einer COM-Schnittstelle zugewiesen werden. Siehe AUTOEXEC.BAT und (PO)rt-Befehl. Als KISS-Treiber für andere Systeme sind z.B. TFKISS, TFPCR und TFPCX zu nennen. Auch der Anschluss eines TNC2c mit einem KISS-/SMACK-EPROM ist möglich. Über die (**MO**)de Funktion beim (**PO**)rt-Befehl lassen sich unterschiedliche CRC-Verfahren einstellen. Aber auch hier nochmals der Hinweis auf den Schnittstellenbaustein 16550 AFN.

Bei Verwendung der GO32 Programme sind FIFO-Bausteine in den COM-Schnittstellen PFLICHT!

Die Vanessakarte

Bei der Vanessakarte handelt es sich um Slot-Karten, die mit einer schnellen CPU und SCC ausgestattet sind. Als Schnittstelle zum Rechner benötigt sie einen ISA-Slot und zur Funkseite können z.B. RS 422 Schnittstellentreiber eingesetzt werden (Stromschnittstelle), die den abgesetzten Betrieb eines Modems über mehrere 100 Meter ermöglichen.

Durch die parallele Datenverarbeitung auf den Bus des Rechners hat es einen mächtigen Ruck im Datendurchsatz gegeben. Die Verarbeitungsgeschwindigkeit des Knotens wird nun durch seine Links (Baudrate und das Übertragungsverfahren (Semi - oder Vollduplex)) bestimmt. Die Laufzeiten innerhalb des Knotens sind durch die parallele Verarbeitung der Daten so gering geworden, dass sie auch bei einem 64 Kbit/s Vollduplexlink nicht ins Gewicht fallen. Bei externem Takt (also synchroner Datenverbindung) kann die Vanessakarte bis zu 140 Kbit/s je Port.

Auf einer Vanessakarte befinden sich 2 Ports.

Ausführliches zur Vanessakarte siehe **Anhang C**.

6Pack

6PACK ist ein zum KISS-Tokenring ähnliches, jedoch von Kanalzugriff weiter ausgereiftes Protokoll. Die gesamte Steuerung des TNC erfolgt durch die Software, der TNC ist nur ein dummer Befehlsempfänger. Genau wie beim Tokenring können mehrere TNC in Reihe geschaltet werden. Im TNC ist ein spezielles EPROM mit einer 6PACK-Firmware notwendig!

Achtung, dieser L1-Treiber befindet sich noch in der Erprobung, er sollte nur nach ausgiebigen Tests produktiv eingesetzt werden! Bitte etwaige Fehler melden an DG9OBU! Außerdem sind das Copyright und die Bestimmungen der aus dem FlexNet-Paket stammenden 6PACK-Firmware für TNC2 zu beachten, insbesondere was den Einsatz im CB-Funk betrifft!

6PACK funktioniert (derzeit) nur unter Linux und auch nur auf einer Schnittstelle mit insgesamt maximal 8 TNCs!

Externe Treiber

In der GO32-Version sind die Schnittstellentreiber für VANESSA, KISSLINK, BayCom USCC und TOKENRING in der Software integriert. Anders sieht es bei Ethernet aus. Für diese Anwendungen wurde die TNL1.EXE geschrieben das den Kontakt zwischen TNN und den EXTERNEN-Treibern herstellt. Als Treiber können nun die Kartentreiber die auch PC-Flexnet verwendet zum Einsatz kommen.

Seit der Version V1.73 ist es möglich, externe Treiber vor TNN zu laden. Diese werden automatisch erkannt und eingesetzt.

Zur Nutzung der externen Treibern in TheNetNode muss zuerst ein Programm geladen werden, welches die Kommunikation zwischen TNN und Treiber ermöglicht. Dieses Programm (TNL1.EXE) muss immer VOR den Treibern geladen werden. Ausnahme: Treiber für z.B. Netzwerkkarten müssen vorher geladen werden. Diese setzen auch nicht auf den TNL1 auf und sind meist herstellerspezifisch.

TNL1 enthält ein 16Bit- und ein 32Bit-Interface, d.h. man kann mit TNL1 auch Treiber laden, die von der GO32-TNN angesprochen werden können.

TNL1.EXE kann für mehr Informationen mit -v aufgerufen werden und ein paar Parameter angeben werden.

```
TheNetNode Layer1 Interface V5.2 (c) 1996 by NORD><LINK (DB7KG, DG1KWA)
Resident portion loaded, handler installed to IRQ 0x60
32-Bit Protected Mode handler installed to IRQ 0x61
Manifest is at 222A:0000
511312  bytes free memory
2432    bytes CODE
2672    bytes static DATA
528     bytes STACK

Usage: tnl1 [[-?] [Options]]

-h -?   diese Hilfe anzeigen / give this help
-32     32bit Interface abschalten / disable 32bit interface
-u      TNL1 und Treiber entladen / unload tnl1
-v      ausführliche Meldungen / verbose
```

Der Ablauf für das Starten eine TNN mit externen Treibern könnte z.B. so aussehen:

```
; hier einen evtl. Packet Driver für die Ethernet-Karte laden
;
NE2000 0x65 10 0x300
;TNL1 laden
TNL1.EXE
; AXIP-Treiber laden
; ein NE2000-Treiber muss geladen sein !
IPPD -i:0x65 -m:44.130.88.1 -p:44.130.88.2
; TheNetNode
TNN179.EXE
; alles entladen
TNL1.EXE -u
```

Die Treiber für VANESSA, KISSLINK (bis zu 4 Stück) und TOKENRING sind weiterhin fest in TNN eingebunden und sollten den externen Treibern (nur VANESSA und KISSLINK) vorgezogen werden, da die internen Treiber wesentlich schneller sind und einige Verbesserungen bieten (z.B. KISS-SMACK oder externer Clock bei der VANESSA). Nach dem Start erkennt TheNetNode automatisch die Treiber, sie müssen dann nur noch auf die entspr. Baudrate und Mode konfiguriert werden.

Port-Nummerierung

Da die Treiber von Port 0 an aufsteigend angesiedelt werden, kann es Probleme mit anderer Hardware z.B. VANESSA geben. Deshalb kann als Platzhalter ein Dummy-Treiber geladen werden. Mit diesem ist es möglich, eine anzugebende Anzahl von Ports freizuhalten.

Syntax : DUMMY <Port-Anzahl>

Diese zwei Anweisungen
DUMMY 6
IPPD -i:0x65 -m:44.130.88.1 -p:44.130.88.2
bedeuten, dass 6 Ports freigehalten werden und dann der AXIP-Treiber geladen wird.

Welche Treiber werden verwendet

Zurzeit werden die Treiber des PC/FlexNet verwendet (aktuelle Version). ACHTUNG: Die Treiber der PC/FlexNet 3.3c gehen nicht mit TNN!

GO32-TNN

Die externen Treiber funktionieren zwar gut mit der GO32-Version von TheNetNode, allerdings gibt es hier eine Sache zu beachten. Geräte die viele IRQ erzeugen, sind für eine GO32-Version Gift, da sie die CPU zwingen, zwischen Real-Mode und Protected-Mode laufend hin und her zuschalten. Dieses Umschalten kostet aber viel an Rechnerpower. Treiber wie SER12 oder PAR96 laufen sicher nicht ohne Probleme. Bei den Netzwerktreibern sieht die Sache etwas besser aus, diese sollten ohne Probleme laufen. In einem Test lief sogar die GO32-Version schneller (!) als eine DOS-Version!

Folgende Treiber gibt es zurzeit, die Liste erhebt aber keinen Anspruch auf Vollständigkeit!

IPPD	AXIP-Treiber; damit kann man seinen LINUX-Rechner (WAMPES oder DP-BOX) an TNN anbinden. Läuft mit GO32- und DOS-TNN
IPXPD	Netzwerktreiber mit IPX-Protokoll, setzt auf einen geladenen PacketDriver auf. GETESTET ....geht UFB!
IPXN	IPX in einem Novell-Netzwerk NICHT GETESTET!
KISS	KISS-Treiber, der interne TNN-Treiber sollte aber vorgezogen werden Läuft mit DOS und GO32
VANESSA	Vanessatreiber; der interne TNN-Treiber sollte aber vorgezogen werden, da er wesentlich schneller ist! Läuft mit DOS und GO32

Für alle Treiber gilt: vor Einsatz die mitgelieferte DOKUMENTATION lesen!!! Wenn jemand TheNetNode mit einen der Treiber verwendet, würden wir uns über etwas Feedback sehr freuen!

Ethernet-Schnittstelle:

TheNetNode(GO32) bietet die Möglichkeit, mehrere TNN oder Mailboxen per Ethernet miteinander zu koppeln. Erforderlich sind lediglich preiswerte Ethernet-Karten und ein passender Treiber nach der TCP/IP PKTDRV-Spezifikation.

Nun braucht man nur ein dünnes RG58 Kabel zu verlegen und 2 Stück 50 Ohm Abschlusswiderstände oder ein Twisted-Pair und kann dort beliebig viele Rechner anschließen, z.B. eine Mailbox oder einen DX-Cluster usw.

Für Mailbox oder Useranbindung:

Das Programm „TFX_NET.COM" verhält sich wie ein TFKISS-Treiber, was die Schnittstelle zum Terminalprogramm angeht. TFX_NET wird einfach als residentes Programm installiert, nach dem der PKTDRV-Treiber geladen wurde. Somit kann man z.B. Terminalprogramme wie TOP, GP, THP und sogar WinGT betreiben und den Knotenrechner über das Ethernet-Netzwerk connecten! So etwas ist z.B. prima für abgesetzte Sysop-Terminals im QRL. Natürlich geht dies genauso auch mit der DIEBOX-Mailbox und anderen Programmen, welche die TFPCR-Softwareschnittstelle unterstützen. Da die AX.25 Daten einfach in Ethernet-Frames „verpackt" und im Kabel „gebroadcastet" werden, können sich alle Teilnehmer auch untereinander connecten.

Die Vorteile liegen auf der Hand:

- Ethernet-Karten (meist NE2000 kompatibel) sind sehr preiswert und kosten nur einen Bruchteil vom Preis einfacher TNCs.
- Doppel-TNCs oder VANESSA/TNC-Kombinationen können entfallen, dadurch spart man einiges an Finanzen und Aufwand ein.
- Der KISSLINK mit all den Problemen der PC-RS232-Schnittstellen kann entfallen.
- es reicht ein „dünnes" RG58 oder Twisted-Pair quer durch alle Räumlichkeiten, wo gerade Platz ist.
- galvanische Entkopplung.

Mehr dazu im ANHANG D.

DPMI oder Nutzung des meist brachliegenden Speichers über 1MB:

Unter MS-DOS hatten wir bisher immer damit zu kämpfen, dass der verfügbare Speicher auf 1 MB (+ HMA) beschränkt war. Diesen Speicher mussten sich alle Programme teilen. Damit waren Programme wie TNN in einen Speicherbereich eingezwängt. Mit dem Dos-Protected-Mode-Interface stehen bis zu 16 MB bzw. 4 GB theoretischen Speicher zur Verfügung. Aber warum noch MS-DOS unterstützen, wo es so gute Betriebssystem wie OS/2, LINUX oder WINDOWS gibt? Alle diese Betriebssysteme haben einen recht hohen Speicherbedarf, außerdem schleppt man bei WINDOWS eine für unsere Zwecke unnötige Oberfläche mit. Vielleicht wird sich nicht sofort jeder Sysop mit LINUX anfreunden können - aber es lohnt sich trotzdem, sich damit zu befassen.

Deshalb die weitere Unterstützung von MS-DOS.

Da die GO32-Version eine Erweiterung der „normalen" DOS-Version ist, wurden die Erklärungen hierzu in den **ANHANG G** verlagert.

TheNetNode-Programm:

Damit sind wir eigentlich beim Programm angekommen. Zu erwähnen sei noch, dass auch der Rechner von einem Watchdog auf der Schnittstelle überwacht und notfalls neu gestartet wird, denn auch das beste Programm kann sich mal irgendwohin verirren....

Das Programmpaket enthält das Programm TNN179.EXE. Passwort, IDENT (Rufzeichen), ALIAS usw. sind im File TNN179.PAS abgelegt, das nach Möglichkeit nicht mehr verändert werden soll. Die mit (S)tat abrufbaren statistischen Werte werden je nach Einstellung des Parameters SaveConfig in X mal 10 Minutenabständen, also bei der Einstellung „6" jede Stunde einmal, auf der Festplatte in dem File TNN179.STA abgespeichert. Somit kann maximal noch eine Stunde in der Statistik je Neustart fehlen. Wie lange das Programm ohne Neustart in Betrieb ist, kann aus der ersten Zeile der Statistik ersehen werden. Die Parameter werden nun über die TNN179.TNB eingestellt. In ihr sind alle benötigten voreingestellten Parameter enthalten. Beim Programmstart sucht TNN.EXE nach den Files, TNN179.PAS, TNN179.STA und TNN179.TNB und lädt daraus seine Betriebsparameter in den Rechner. Sind die Files nicht vorhanden oder defekt, so wird beim Start am Monitor angezeigt:

WARNING: Fehler in TNN179.PAS ***
*** TNN179.STA - Open-Error ***
WARNING: Fehler in TNN179.STA ***

ACHTUNG: Packet-Treiber (PKTDRV) fehlt!!

Login: TNN179.TNB not found!

Das Programm benutzt dann die Default-Parameter und legt die Files TNN179.PAS und TNN179.STA neu an. Sind die Files in Ordnung, erfolgt kein besonderer Hinweis.

ACHTUNG: Die Meldung „Packet-Treiber fehlt" ist NUR der Hinweis, dass eine Ethernet-Schnittstelle NICHT angesprochen werden kann. Sie hat aber für den Betrieb OHNE Ethernet keine weiteren Nachteile!

Nach dem Ändern von Parametern wird TNN179.STA jeweils nach Verlassen des Sysopmodus durch Quit, Disconnect oder Connect jeweils neu auf die Festplatte geschrieben. Damit ist sichergestellt, dass bei einem Programmneustart die Parameter wieder richtig gesetzt werden.

Die Files TNN179.PAS und TNN179.STA haben die Versionsnummer im Filenamen. Damit ist es dann auch möglich, unterschiedliche Programmversionen im Knoten abzulegen.

TheNetNode ist im Moment auf 400 Level 2 und 200 Level 4 Verbindungen eingestellt. Dieses sind aber keine Endwerte. Die Verwaltung von einer größeren Menge an Verbindungen ist nur noch eine Frage des Speichervolumens des Knotenrechners und seiner Taktfrequenz.

DAMA ... Verkehrsregelung durch den Knoten:

DAMA ist heute auf den Userzugängen eine weitverbreitete Betriebsart. Es ist nun nur noch der entsprechende Port <nr> und DAMA <nr> (<nr> = 1..16 DAMA-Master) zu setzen. Parametereinstellungen sind keine mehr nötig. Siehe auch Porteinstellungen.

Im Port-TNC, auf dem DAMA laufen soll, muss das DAMA-Bit im EPROM gesetzt sein. Bei gesetzten DAMA-Bit sendet der TNC an den Knotenrechner die Information zurück, wenn ein Paket an einen User abgesendet wurde. Diese Information wird für die Steuerung der DAMA-Funktion benötigt. **(Bei der Vanessakarte nicht erforderlich.)**

DAMA-Ablaufsteuerung:

Stationen, die keine DAMA-fähige Usersoftware benutzen und OHNE Aufforderung durch den DAMA-Master senden, werden IM QSO verwarnt. Sind mehr Verwarnungen nötig, als unter DAMA-MaxPol angegeben, so wird der User disconnected.

Für den TNC2 gibt es ein DAMA-fähiges EPROM, es steht in der Mailbox unter der Rubrik 'TNC2'. Dort wird auch meist ein kostenloser Eprom-Service angeboten. Wer die Vorteile von DAMA nutzen möchte, sollte unbedingt **The Firmware 2.7b** nachrüsten! Seit der **The Firmware 2.7b** kommt es auch bei Multiconnect nicht mehr zu Meckermeldungen. Der Softwarefehler, dass der TNC nach Anpollen einer DAMA-Verbindung die anstehenden Daten aller bestehenden Verbindungen gesendet hat, ist beseitigt.

Alle anderen Benutzer sollten unbedingt FRACK auf den Maximalwert einstellen.

***** Wichtig *****

Auf dem DAMA-Einstieg sind direkte QSO unter Umgehung des Knotens zu unterlassen, da sie, wie jede nicht vom Master angeforderte Aussendung, zur Störung des gesamten Betriebes führen!!! Dies gilt auch für Betrieb über eine andere Station als „Digipeater", um den Knoten auf der gleichen Frequenz zu erreichen.

L2 Digipeating auf einem DAMA - Port führt nun zu einem sauberen Eintragen der Linkverbindung und damit zu einem reibungslosen DAMA - Verkehr.

USER mit Multiconnect kommen gegenüber USERN mit nur einer Verbindung zum Knoten nicht öfters an die Reihe. Aber „Leerpolls" werden nun auch entsprechend beachtet.

SYSOP-Arbeiten am Knoten:

Bedienen kann man das Programm einmal von der Konsole (Tastatur) aus oder auch über Funk. Für beide Fälle ist eine Identifikation als SYSOP notwendig. Von der Konsole aus wird der Benutzer aufgefordert das „Passwort" einzugeben. Als Default-Passwort ist „Geheim" im Programm festgelegt. Ist es korrekt eingegeben (Gross- und Kleinschreibung beachten), wird man mit

Welcome to TheNetNode (GO32), Version 1.79 (Datum)

begrüßt. Mit <ESC> c <RETURN> kann man sich nun als HOST mit dem Knoten verbinden. Als HOST ist man auch automatisch SYSOP und kann alle Änderungen an den Parametern vornehmen.

Nach Ende der Arbeiten kann man ihn wieder mit <ESC> logout <RETURN> verschließen. Ein automatisches Ausloggen findet vor jedem Stundenübergang statt.

Von der Funkseite ist der Knoten erst zu connecten und dann der Sysop-String mit mindestens „sy" abzufragen. Daraufhin bekommt man folgenden Text zugesandt: „KS:DB0EAM> 62 36 65 13 34". Diese Ziffern geben jeweils die Position eines Zeichens im Passwort an. Mit den zu den Ziffern gehörenden Zeichen muss nun geantwortet werden. In dem Passwort dürfen KEINE Leerzeichen vorkommen und die angeforderten Zeichen müssen direkt nacheinander folgen. Vor und nach den angeforderten Zeichen dürfen auch andere Zeichen stehen, so dass nicht mehr ohne weiteres ersichtlich ist, welche Zeichen die Antwort darstellen. Eine Quittung, ob die zurückgesendete Zeichenkette richtig oder fehlerhaft war, kommt nicht. Diesen Vorgang kann (und sollte) man mehrmals hintereinander durchführen, wobei er nur einmal richtig sein muss. Ob dann der 1. 2. oder 3. Versuch das richtige Passwort enthielt, spielt keine Rolle. Die mehrfache Identifikation dient nur der Verwirrung der „Geister", die allzu gern auch das Passwort hätten, um dem Sysop „hilfreich" unter die Arme zu greifen. Ein wahrer Sport soll sich da mancherorts schon entwickelt haben, hihi. Die meisten Terminalprogramme für Packet Radio unterstützen die automatische Privilegierung als Sysop.

Um den hilfreichen „Geistern" auch den letzten Zahn zu ziehen, so steht z.B. wenn man das Programm zum Knoten als Binärfile fernläd, stets das Passwort auf den Default-Parametern „12345678901234 usw". Damit ist nun auch das Mitschreiben des Binärfiles sinnlos. Einerseits kann das Programm jeder bekommen, andererseits kommt man auch auf diesem Weg nicht an das Passwort heran. Das aktuelle Passwort steht ja bereits in TNN179.PAS.

In der Datei SYSOP.PRO wird zusätzlich noch ein Logbuch geführt, das jede Eingabe des SYSOP-Befehls registriert. Es werden Rufzeichen, Datum und Uhrzeit gespeichert. Diese Datei kann vom SYSOP ausgelesen und bei Bedarf auch wieder gelöscht werden, wenn sie zu lang wird.

Nun sollte zumindest dem Sysop die „gewaltige" Herrschaft über den Knoten offen stehen.

User-Passwort-Funktion

Die neue User-Passwort-Funktion ermöglicht dem Sysop des Knotens den Zugang auf einzelnen Ports auf User zu beschränken, die zuvor vom Sysop ein Passwort erhalten haben.

Um die User-Passwort-Funktion einzuschalten, wird der entsprechende Port zunächst auf SYSOP gesetzt. Weiterhin wird eine Datei „PERMS.TNN" im TNN-Verzeichnis benötigt, in der jeweils in einer Zeile das Call eines Users und durch Leerzeichen abgetrennt der Name einer Passwort-Datei (wenn gewünscht auch incl. Pfad) aufgeführt sind. In der Passwort-Datei befindet sich nur das 80 Zeichen lange Passwort des Users. In der Datei „PERMS.TNN" dürfen auch Kommentarzeilen eingefügt werden, diese müssen mit „#" beginnen.

Beispiel für PERMS.TNN (GO32-Version):

```
DH2LAB C:\TNN\PWD\DH2LAB.PWD
DF5LMD C:\TNN\PWD\DF5LMD.PWD
```

Beispiel für perms.tnn (Linux-Version):

```
DH2LAB /usr/local/tnn/pwd/dh2lab.tnn
DF5LMD /usr/local/tnn/pwd/df5lmd.tnn
```

Beispiel für eine Passwort-Datei:

```
ABCDEFGHI  usw 80 Zeichen  TUVWXYZ
```

Wird beim SYSOP-Befehl eine entsprechende Passwort-Datei gefunden, so wird daraus das Passwort gelesen. Weiterhin werden dem User anschließend alle normalen TNN-Befehle freigegeben, allerdings hat der User damit keinen Sysop-Status. Wer auch den Sysop-Status erhalten soll, darf keine Passwort-Datei erhalten - dann gilt das normale Knoten-Passwort.

Nach dem Connecten des Knotens ist auf dem entsprechenden Port nur der SYSOP-Befehl zulässig. Die Funktion des SYSOP-Befehls ist unverändert, so dass auf die 5 Passwort-Zahlen auch mit einer langen Zeichenfolge geantwortet werden kann, in der die gefragten Zeichen versteckt sind. Selbstverständlich kann der SYSOP-Befehl auch mehrfach eingegeben werden.

Im Quelltext ist die User-Passwort-Funktion mit einem Compiler-Switch abschaltbar.

Noch ein sehr wichtiger Punkt zu TheNetNode :

Da von TNN viele Daten gespeichert werden, ist die Verwendung einer (kleinen) Festplatte sehr zu empfehlen.

Rechnerkonfiguration: (Hier als Beispiel für einen 386 oder 486 mit HD)

```
CONFIG.SYS
**********
DEVICE=c:\HIMEM.SYS
DOS=HIGH
COUNTRY=049,850,C:\DOS\COUNTRY.SYS
FILES=30
BUFFERS=40
STACKS=9,256
```

AUTOEXEC.BAT

Übersichtshalber wurde nun die AUTOEXEC.BAT aufgeteilt und für die TNN-Spezifischen Einstellungen eine START.BAT angelegt. Die AUTOEXEC.BAT enthält die rechnerspezifischen Einstellungen, die START.BAT die Einstellungen für TheNetNode.

```
@ECHO OFF
prompt=$p$g
KEYB GR,,a:\KEYBOARD.SYS
path c:\dos; c:\tnn
START.BAT
```

START.BAT

```
set tnncfg=1009,32
set tnn32buffers=5000
set conversd=P:0
set TZ=UTC0
set msgpath=C:\TNN\MSG\
set QTH=<koordinaten> oder <locator>
set CONSOLE=3
set TOKENCOM=2
set KISS1=1
set copycmd=/y
set TOGGLEPORT=x
b-log.exe
cd TNN
TNN179.exe
```

Erklärungen zur START.BAT:

SET TNNCFG=<1.Anzahl>,<2.Anzahl>

Die 1.Anzahl gibt die Grösse der Nodes / Destinations Liste an. Default Einstellung: 1009.
Die 2. Anzahl ist die Länge der Linkliste. Default: 32.

Die Verwaltung der Routing-Tabellen von TheNetNode erfolgt mit einem so genannten Hash-Algorithmus. Dieser arbeitet sehr effizient, solange die Nodesliste zu weniger als 50% gefüllt ist. Ist der Füllungsgrad höher, erhöht sich die Zugriffszeit auf die Tabelle überproportional hoch. Auf Systemen mit wenig Speicher sollte die Nodes-Tabelle auf mindestens 30% mehr als die zu erwartende Nodes-Anzahl eingestellt werden. Auf Systemen mit viel Speicher kann die Liste ruhig um Faktor 2-3 überdimensioniert werden.

SET TNN32BUFFERS=

Über die Umgebungsvariable TNN32BUFFERS kann TNN mitgeteilt werden, wie viele Buffer benutzt werden sollen. Sie wird in der START.BAT mit z.B. „SET TNN32BUFFERS=2000" gesetzt. Die Auswirkungen nach der Änderung dieser Einstellung auf die Buffer-Verwaltung lassen sich nach dem Starten von TNN (im connecteten Zustand), mit dem Befehl DOS MEM /c ablesen.

SET CONVERSD=P:0

P bestimmt, ob das Protokoll auf Kanal 32767 angezeigt wird. Das Protokoll zeigt eingehende und ausgehende Hostbefehle an und kann zu Analysezwecken eingesetzt werden.

P:0 (default): Protokoll aus
P:1 Protokoll an

SET TZ:

Die Variable TZ wird auch für den Conversmode und die SAT-Berechnung benötigt. Damit die Zeiten stimmen, ist sie, je nach eingestellter Rechnerzeit, auf UTC0, MEZ-1, EST-2 oder CET-1 zu stellen.

SET MSGPATH:

Pfad für MSG-Files

SET QTH=<koordinaten> oder <locator>:

Mit der Variablen QTH werden für das Programm QTH.EXE und SAT.EXE im Verzeichnis USEREXE die Koordinaten oder Locator des Knotens eingegeben. Im Nahbereich sind die Koordinaten natürlich weitaus besser geeignet. Siehe: externe Programme für alle User.

SET CONSOLE=<com Nr.>,<Adresse>,<irq>

Die Schnittstelle für die CONSOLE kann auf COM 1 oder auf COM 2 gelegt werden. Die Angaben <Adresse> und <irq> sind nur für exotische Adressen und Interrupts notwendig. Wird die Console durch HOSTMODE belegt, so steht die Tastatur jedoch nicht mehr zur Verfügung. Mit <ESC> I <call> muss dann das Call des angeschlossenen Dienstes eingetragen werden. Das Call kann dann auch mit M <call> oder DX <call> verbunden werden damit es in den Nodes - / Destinations – Listen eingetragen wird. Das Host - Interface wird über Interrupts gesteuert. Der Speed ist derzeit auf 19200 Bit/s festgelegt.

SET TOKENCOM=<com Nr.>,<Adresse>,<irq>

Die Schnittstelle für der TOKENRING kann auf COM 1 oder auf COM 2 gelegt werden. Die Angaben <Adresse> und <irq> sind nur für exotische Adressen und Interrupts notwendig.

SET KISS1=<com Nr.>,<Adresse>,<irq>

Es sind nun 4 KISS-Schnittstellen mit KISS1..4=1..4 aktivierbar. Wie auch bei der TOKENCOM-Schnittstelle sind auch hier die Einstellungen der <Adresse> und des <irq> nur bei Abweichungen vom Standard notwendig. Wird die KISS-Schnittstelle nicht benötigt, kann sie mit SET KISS1=0 deaktiviert werden.

SET TOGGLEPORT=<lpt Nr.>

An einem LPT Port des TNN-Rechners kann ein zusätzlicher Watchdog angeschlossen werden. Es wird jede Minute einmal die Resetleitung (Pin 16) der LPT getoggelt. Es darf auch eine LPT auf einer Videokarte sein.

B-LOG.EXE:

Auf der Diskette ist des Weiteren ein Programm B-LOG.EXE. Es legt eine Datei LOG.DAT an. Wird B-LOG.EXE von der AUTOEXEC.BAT vor dem Programmstart TNN.EXE aufgerufen, so bekommt man eine lückenlose Übersicht über die Bootvorgänge des Knotenrechners.

TNN179.PAS

Durch die externen Kommandos wurden zusätzliche Einträge in der PAS-Datei erforderlich. Um die Lesbarkeit der Datei zu verbessern, werden von TNN automatisch Kommentare hineingeschrieben.

Die Datei hat jetzt folgenden Aufbau und MUSS von Sysop entsprechend editiert werden:

Beim Node-Ident ist zu beachten: Werden Sonderzeichen verwendet, so kann der Knoten **N I C H T** mit dem Node Ident oder ALIAS von dem / den Userzugängen connected werden. Also „c Lippe" oder „c Lippe-1 ... -15" werden verhindert.

```
; TheNetNode Konfiguration File
;
; DO NOT CHANGE THE ORDER OF THE KONFIGURATION LINES !
; DO NOT CLEAR ANY LINES !
;
; NET/ROM-Sysop-Password, 80 Characters (01234567890123...)
12345678901234567890123456734567890 usw. bis 80 Zeichen...
;
; Console Password (Geheim)
Geheim
;
; Node Ident (Test)
Test
;
; Node MyCall (XX0XX)
XX0XX
;
; Workpath, Path to the Help-Files (\TNN\)
; TNN should be started from this path.
C:\TNN\
;
; Path to the executable Text-Files (\TNN\TEXTCMD\)
C:\TNN\TEXTCMD\
;
; Path to the extern Programs for User (\TNN\USEREXE\)
C:\TNN\USEREXE\
;
; Path to the extern Programs only for Sysop (\TNN\SYSEXE\)
C:\TNN\SYSEXE\
;
; Path to the PACSAT-Files (\TNN\PACSAT\)
C:\TNN\PACSAT\
```

Verzeichnisstruktur:

STARTPATH

Hier stehen das ausführbare TNNxxx.EXE und die Konfigurationsdateien (TNNxxx.PAS TNNxxx.STA MHEARD.TAB TNN179.TNB). Dieser Pfad ist nicht einstellbar. Er richtet sich immer nach dem Ort, an dem TNNxxx.EXE steht. Wird TNN mit „ A:\TNNxxx.EXE „ gestartet, so werden im gleichen Verzeichnis auch die Konfigurationsdateien gesucht und geschrieben.

WORKPATH

enthält alle von TNN verwendeten Dateien (.TNB, .XHF), .TXT, sofern es sich um Systemmeldungen handelt (HELP.TXT, SUSPEND.TXT, QUIT.TXT, CTEXT.*).

TNN wechselt nach dem Start automatisch auf das unter WORKPATH eingetragene Verzeichnis.

TEXTCMD

Hier gehören die knotenspezifischen Texte hin, die unter HELP.TXT angegeben sind. (MAP.TXT, INFO.TXT, AKTUELL.TXT usw.)

USEREXE

Hier stehen die externen Kommandos, die von allen Usern ausgeführt werden dürfen.

SYSEXE

Hier stehen alle externen Kommandos, die nur mit SYSOP-Privileg ausgeführt werden dürfen.

PACSAT

Unterverzeichnis für BROADCAST-Betrieb. Siehe hierzu **ANHANG E**:

Folgende Verzeichnisstruktur arbeitet problemlos für Systeme mit Harddisk und mit RAMDISK:

C:\TNN\
C:\TNN\MSG\
C:\TNN\TEXTCMD\
C:\TNN\USEREXE\
C:\TNN\SYSEXE\
C:\TNN\PACSAT\

Die Bearbeitungsreihenfolge der Kommandos ist: INTERN - TEXTCMD - USEREXE - SYSEXE.

TNN179.TNB

Mit der TNN179.TNB wird die Grundeinstellung des Knotens vorgenommen.

```
;
outp 0 1    ; Grundeinstellung der parallelen Schnittstelle LPT 1
....
outp 7 1
;
; Call der Console muss VOR dem conn. gesetzt werden mit "ESC I CALL"
;
; HOSTMODE-Call muss hier oder im conn. Zustand gesetzt werden.
; dabei wird ein fester Linkeintrag auf Port 16 gesetzt.
; esc i DG9FU-13 #Host
;
; Host-IF Y=0 nicht zu connecten...oder Anzahl der feigegebenen Kanaele 1..30
esc y 0
;
; Tokenring Geschwindigkeit
esc t 19200
;
; diverse L2 und L3-L4 Parameter
;
par   1    20  ; NoAck Buffer
par   2     0  ; L3-MaxTime
par   3     3  ; Config speichern in (10min)
par   4  4800  ; DAMA-Speedfaktor
par   5    10  ; DAMA-MaxPri cnt (Maximaler Zählerstand)
par   6     5  ; Maximale Verstösse gegen DAMA Protokoll dann Disconnect.
par   7    50  ; DAMA Tout Wartezeit in ms wenn ein User nicht antwortet.
par   8     0  ; command.log 0=kein 1=Sysop.. 2=Alle Befehle festhalten
par   9     1  ; sysop.log 0=kein 1=wird geführt
par  10    15  ; TestSSID verwendete SSID für L+ Links
par  11     1  ; ConvSSID Verwendete SSID für Convers
par  12     0  ; AutoIPR  Automatische Ip-Eintragung
;
;-------------------------------------------------------------------------
;
;  Mögliche Linkeinträge
;  LINK +/- <Typ> <Port nr> <Alias> <Call> [<Digi> <Digi>]
;                                          [Wahlweise 1-2 Digi´s]
;                                    Link Call
;                             Alias 6 Zeichen
;             L  = Local keine Messung
;             L+ = Localer Link wird gemessen
;             F  = FlexNet
;             N+ = NetRom Protokoll mit ON5ZS Änderungen
;             N  = NetRom
;             N- = Altes TheNet Protokoll
;             I  = INP-Protokoll
;
li + L+ 5  KSBOX  DB0EAM-3
li + L  3  LOH    DB0VFK
li + I  6  GOE    DB0GOE
li + F  9  BIDFLX DB0BID
li + N  9  BIDTNN DB0BID-7
li + N  9  HR     DB0KH DB0BID
li + N  11 HR     DB0KH
;
;   Locale Mailbox kann nun unabhängig von einem "eigenen" Port gesetzt
;   werden. m call setzt die Mailbox M - löscht den EintragI
m DB0EAM-3
;Genauso verhält es sich mit einem DX-Cluster. dx call setzt und dxc – löscht den Eintrag
dx db0eam-7
;
;-------------------------------------------------------------------------
```

```
;
;  PORT <NR> [<CMD>[=<PAR>]] [<CMD>[=<PAR>]] [...]
;
;
; CMD:
;  Off                    Port abschalten
;  ON                     Port einschalten (nur Linux),
;  SCC                    Port auf USCC-Karte einstellen (DOS/GO32) bzw.
;                         keine TNC-Parameter setzen (nur Linux)
;  TOkenring              Port auf Tokenring schalten
;  Vanessa                Port auf Vanessa schalten
;  EXTdrv                 Port auf den extern geladenen Treiber schalten
;  KISS1..4               Port auf Kiss ohne CRC
;  RKiss1..4              Port auf Kiss mit RMNC-CRC
;  SMACK1..4              Port auf Kiss mit SMACK-CRC
;  MODEM1..3              Port auf Modem (TNC3)
;  HSBUS                  Port auf HSBUS (TNC3)
;  LOOP                   Port auf Loopback
;  Name                   Port-Namen vergeben, z.B. Name=User1k2
;  MODE                   Port-Mode setzen, Format: 9600d
;                         Baudrate: 300...4915200
;            Flags:
;              d :        Duplex
;              c :        DCD bei 1k2-Modem
;              r :        ext. Takt (rx)
;              t :        ext. Takt (tx)
;              e :        ext. Takt beide (Vanessa)
;              m :        Multibaud-Kopplung (Vanessa, SCC)
;              z :        NRZ statt NRZI
;  MAXFRAME  <wert>(a)    Maxframe setzen. 1-7. a sagt Maxframeautomatik ist an.
;  TXDelay   <wert>       setzen
;  DAMA   0/1..16         Port auf DAMA Master 1-16 setzen; 0=aus
;  CTEXT  on/off          .. Ctext senden
;  SYSOP  on/off          .. nach Connect nur SYS möglich, gilt nur
;                            für Verbindungen zum CCP, NETROM und
;                            TCPIP gehen weiterhin
;  MH     on/off          .. MH-Liste führen
;  EAXMAXF 7 – 32         Maxframe für Extented AX25 setzen
;  EAXMODE 1 – 2          Mode für Extented AX25 setzen

;---------------------------------Port 0 setzen:--------------------------
port 0 NAME=70cm_1200;
port 0 VANESSA;
port 0 Mode=1200m;
port 0 Maxframe=2a;
port 0 TXdelay=20;
port 0 CTEXT=on;
port 0 DAMA=1;
port 0 SYSOP=off;
port 0 MH=on;
port 0 EAXMAXF=16;
port 0 EAXMODE=1;
be 0 10 0 ID1200;
be 0 = 1200 Bit/s User-QRG TheNetNode Digi DB0EAM : Rx =438,400 MHz -7,6 MHz
;---------------------------------Port 1 setzen:--------------------------
;---------------------------------Port 2 setzen:--------------------------
usw. die anderen Ports ....
;-------------------------------------------------------------------------
;
; MH-Liste                        ; Grösse der MH-Liste (Max 5000)
MH = 500
; L3MH-Liste
L3MH = 500
;
; Convers-Einstellungen
conv c DB0GOE;                      ; Goettingen
conv c DB0II 9 DB0BID;              ; Mönchengladbach L2
;
```

```
; Prompt-Einstellungen
;                 %a = Alias des Knoten           %c = Ident User
;                 %d = Ident des Knotens          %C = Ident User mit SSID
;                 %D = Ident des Knotens mit SSDI %r = ein Carriage Return
;                 %t = Uhrzeit des Knotens HH:MM  %0 = Keine Aussendung
;
pr =%C de %d (%t) >%r
;  ^^^ wichtig: kein Leerzeichen hinter dem "="
;
; sonstige Funktionen
; SUS + 0   DB5xxx     ; Sperrungen Port 0
; SUS + 254 DB5xxx     ; Sperrungen Level 2
; SUS + 255 DB5xxx 2   ; Beschränkung auf 2 ports
; SUS + 0   DB5xxx 0   ; Sperrungen von Usern
;
sus + 254 DG7xx
sus + 255 DG7xx 0
;
pac c XX0XXX               ; PacSatcall setzen
;
pac p 01 250               ; PacSatTimer
pac p 02 20                ; PacSatFrms
pac p 03 1500              ; PacSatMail
;
par r                      ; Filesystem neu laden
;
; pac + 2                  ; Port für PacSat einschalten
;
; Commando-Aliasses
alias funkruf c db0whv-12  ; Der FunkrufMaster in Wilhelmshaven
;
;IP-Config
IPA 44.130.3.105/32
IPR 44.130.3.0/24 +  70cm
```

SYSOP-Befehle:

Die folgenden Befehle sollten nun den Sysop in die Lage versetzen, den Knoten für die örtlichen Belange einzurichten.

(ALIAS)
Gibt die derzeitigen Synonyme aus

(**ALIAS**) <Synonym> <zu ersetzender Befehl>
Setzt Synonyme für andere Befehle (z.B. „ALIAS funkruf c db0xyz-12“ erzeugt den neuen Befehl „funkruf“). Auf diese Weise neu erzeugte Befehle stehen auch den Benutzern zur Verfügung. Die ALIAS-Befehle müssen komplett ausgeschrieben werden.

(ALIAS) <Synonym>
löscht einen Eintrag

(AX)IPR
Neuer Kommandoparser und Syntax für AXIPR, der Befehl orientiert sich nun grob am Linux "route"-Kommando. Gibt den derzeitigen Status der AXIPR-Einträge aus.

ACHTUNG, es wird immer entweder NUR die neue ODER die alte Syntax verstanden!!! => tnb's in der alten Syntax versteht der neue Parser nicht und die AXIPRs fehlen dann!

Soll die alte Syntax (= der alte Parser) wieder verwendet werden, dann #define AXIPROLDSYNTAX in all.h setzen! Bei "SP" wird ebenfalls in der neuen Syntax geschrieben falls nicht auf alte Syntax zurückdefine'd wurde. Nur Einträge in der ax25ip.cfg werden von beiden Versionen verstanden!

(AX)IPR {add, +} {call, "default"} <IP / Hostname> [<UDP> [<UDP-Port>]]
Fügt eine neue Route hinzu.
Beispiele:

IP-Route hinzufügen:	axipr add dg9obu-1 1.2.3.4
UDP-Route hinzufügen:	axipr add dg9obu-1 1.2.3.4 udp 12345
Defaultroute (IP):	axipr + default 1.2.3.4

(AX)IPR {delete, del, -} {call, "default"}
Löscht eine Route wieder.
Beispiele:

Route löschen:	axipr - dg9obu-1
Defaultroute löschen:	axipr del default

(AX)IPR myudp [UDP-Port]
Ändert den eigenen UDP-Port.
Beispiel:

UDP-Port ändern:	axipr myudp 12345

(AX)IPR axipr {loglevel, log} [Loglevel]
Setzt den Loglevel.
Beispiel:

Loglevel ändern:	axipr loglevel 3

loglevel 0	no output
loglevel 1	config info only
loglevel 2	major events and errors
loglevel 3	major events, errors, and AX25 frame trace
loglevel 4	all events

(AX)IPR axipr timeout [seconds]
Setzt das Timeout für dynamisch gelernte Routen.
Beispiel:

Timeout ändern:	axipr timeout 7200

Bei erfolgreicher Abarbeitung erfolgt bei den Kommandos keine erneute Ausgabe der Liste und man erhält gleich wieder das Prompt. Nur im Fehlerfall erfolgen Fehlerhinweise. Bei Verwendung der neuen Syntax kann (eigentlich) auf die ax25ip.cfg verzichtet werden, alle Einstellungen können nun dynamisch verändert werden und sind über die tnn179.tnb einlesbar.

(BE)ACON <port> = <text>
Setzt bzw. ändert die Bakentexte für die einzelnen Ports.

(**BE**)ACON <port> =
Entfernt den Bakentext.

(**BE**)ACON <port> <mins> <metric> <call oder alias via call>
Schaltet die Bake ein.

Port 0..16	Portnummer.	
mins 0, n	0 =	Bake aus.
	n =	Intervall in Minuten. Default ist n = 10 Minuten.
metric 0, 1, 2, 3	0 =	Keine Metric.
	1 =	Metric.
	2 =	Zusätzlich STAT.
	3 =	Datenbank gerechte Ausgabe zum direkten Einlesen.
call oder alias	QST =	UI-Frame von Knoten an Call 'QST'.

Es gibt Terminalprogramme, welche die METRIC- und/oder STAT- Bake direkt auswerten und anzeigen können. Besonders sind an dieser Stelle zu erwähnen: THP (Turbo Host Packet) und TOP (The Other Packet).

Die Baken haben dann den folgenden Inhalt und Aussehen:

Am Knotenterminal kann diese Bake auch mitgeschrieben werden. Eingabe: <ESC> MU + METRIC STAT oder <ESC> MCU + METRIC STAT. Beim Neustart wird der Monitor ausgeschaltet.

Die Bake mit metric=1 hat diese Informationen:

```
050208 210306 Up= 101148 Mem=1773568 Buf=5956 Rps=  524
Lnk=154 Cir= 45 Sum=2602324027 Thr= 51464
050208 210406 Up= 101159 Mem=1773568 Buf=6645 Rps=  522
Lnk=154 Cir= 43 Sum=2602669277 Thr= 48512
```

`050208 210306`	Aktuelles Datum und Uhrzeit. 08.02.2005 21:03:06.
`Up=101158`	Betriebszeit in Tagen, Stunden und Minuten. 10 Tage, 11 Stunden und 58 Minuten.
`Mem=1773568`	Freier DOS RAM (Coreleft, Bytes)
`Buf=5956`	Freier TNN-Buffer (Blöcke).
`Rps=524`	Hauptschleifendurchläufe/Sekunde.
`R=0`	Anzahl der Token-Recoveries.
`Lnk=154`	Anzahl der aktiven L2-Links.
`Cir=45`	Anzahl der aktiven Circuits (L4-Links).
`Sum=229961`	Zählerstand des Statistik-Gesamtzählers.
`Thr=51464`	Derzeitiger Datendurchsatz in Baud.

Bei metric=2 wird zusätzlich bei jedem empfangenen L3-RTT Messframe ein UI-Paket des Knotens an METRIC abgestrahlt.

```
10.12.97 16:26:52 DB0LBG-7(09)
L3RTT=15490ms L3SRTT=7350ms (7350ms/6760ms) L2SRTT=820ms SUM=4302022
10.12.97 16:26:54 DB0NHM(10)
L3RTT=840ms L3SRTT=790ms (1090ms/500ms) L2SRTT=640ms SUM=1750605
10.12.97 16:27:02 DB0BID(09)
L3RTT=2210ms L3SRTT=1340ms (1780ms/900ms) L2SRTT=680ms SUM=1510236
```

	Aktuelles Datum und Uhrzeit
`DB0LBG-7`	Rufzeichen des Nachbarknotens.
`(09)`	Empfangsport des Messframe.
`L3rtt=000185ms`	Soeben gemessener Level-3 RTT.
`L3SRTT=1.Zeit(2.Zeit/3.Zeit).`	
`1. Zeit`	Ermittelter SRTT aus der Messung. Bei FlexNet mittel aus 2. Zeit und 3. Zeit.
`2. Zeit`	Von diesem Knoten ermittelter SRTT in ms.
`3. Zeit`	ohne Erklärungen ausgestrahlt.

```
Datum    Uhrzeit  Betriebs Freies Freie   RPS LNK  CIR Summe       Durch-
                  Zeit (s) Ram    Blöcke               Byte        satz
-------------------------------------------------------------------------
13.03.98 09:12:06   41159  1642496 7122   696 128  32  114642456   30488
13.03.98 09:13:06   41219  1642496 7201   689 129  33  114924309   34568
13.03.98 09:14:06   41279  1642496 7409   695 126  32  115162851   32848
13.03.98 09:15:06   41339  1642496 7276   674 130  35  115401514   32264
13.03.98 09:16:06   41399  1642496 7296   707 121  33  115613708   30120
```

```
Call    Port  Datum   Uhrzeit  L3RTT L3SRTT L3SRTT L3SRTT L2SRTT Übertragene
                                     mittel MyQual HisQual       Byte
-------------------------------------------------------------------------
DB0BID     9 10.12.97 16:02:44   330   1190   1590    800    590 1488645
HB9AK      9 10.12.97 16:03:11 14350   6320   6320   8220    350 2026217
DB0GOE     6 10.12.97 16:03:16   480    620    620    700    420 29427844
DB0BRO    10 10.12.97 16:03:26  3350   2620   2620   2310    860 4626794
DB0LIP     8 10.12.97 16:03:45   810   1230   1230    610    900 13049378
DB0KH     11 10.12.97 16:04:30   490   1210   1210   1130    420 24740302
```

(**BE**)ACON <port> 0 0 ID
Schaltet die Bake wieder aus.

(**CL**)EAR
Löscht alle Statistikeinträge.

(**CONV**)ers (**C**)stat
...oder im Conversmode mit: /Link
Zeigt die Liste und den Status der Nachbar-Convers-Host an sowie die Laufzeiten zu allen Convers-Host und deren Versionsnummer. **NUR** im Sysopmodus wird zusätzlich der eingetragene Weg (ob l4 oder l2 mit Port und Via-Einträgen) angezeigt.

(**CONV**)ers (**C**)stat <call>
...oder im Conversmode mit: /Link <call>
Nimmt einen Nachbarknoten für Convers-Betrieb auf. Unter <call> ist das Rufzeichen des Nachbarknoten einzusetzen. Der Nachbarknoten muss dann in der Nodesliste stehen.

(**CONV**)ers (**C**)stat <call> <port> <via-call>
...oder im Conversmode mit: /Link <call> <port> <via-call>
Nimmt einen Nachbarknoten für Convers-Betrieb über eine Level2 Verbindung auf. Unter <call> ist das Rufzeichen des Nachbarknotens einzusetzen. Unter <port> ist die Portnummer anzugeben und bei Bedarf ein oder mehrere via-call.

(**CONV**)ers (**C**)stat <call> 254
...oder im Conversmode mit: /Link <call> 254
Gibt einen Nachbarknoten für Convers-Betrieb frei. Unter <call> ist das Rufzeichen des Nachbarknotens einzusetzen. Es wird zu dem <call> KEIN Link aufgebaut sondern TNN erwartet, dass dieses <call> den Link aufbaut. Dieses ist z.B. für Wampe Rechner die zwar selbst einen Link aufbauen können aber nicht mit einem ankommenden Connect zurecht kommen. Der Eintrag wird gekennzeichnet mit: (trusted host).

ES MÜSSEN UNBEDINGT LOOPS VERHINDERT WERDEN. DESHALB: BITTE ABSPRECHEN !

(**CONV**)ers (**C**)stat - <call>
...oder im Conversmode mit: /Link - <call>
Löscht den Eintrag.

(**CONV**)ers - intern -

PingPong Convers aus Sysopsicht:

Links:

Links werden mit dem „/links" Befehl eingetragen, der auch außerhalb des Convers mittels „conv c" verfügbar ist. So eingetragene Links sind permanent und werden in bestimmten Intervallen (9s, 150s, 300s, 600s,..., 3h) connected. Es gibt auch nichtpermanente Links, diese Option ist aber durch Konfiguration einstellbar. Diese Links werden bei einem Linkabriss automatisch ausgetragen, der Knoten connected den neuen Partnern nicht hinterher. (Das Eintragen geschieht durch Eingeben des „/..host" Befehls.)

Syntax ist: /l [-][call [port [via]]]
bzw. ausserhalb von Convers: conv c [-][call [port [via]]]

-	Dient zum Löschen eines Eintrags (<call> reicht aus).
<call>	Ist das Rufzeichen des Nachbarknotens.
<port>	Downlinkport (wird nur bei L2 Verbindungen gebraucht).
<via>	Via-call (optional für L2 Verbindungen).

Für L4-Verbindungen genügt die Angabe des Calls des gewünschten Knotens, es wird NICHT versucht, eine L2-Verbindung bei fehlendem Nodeeintrag aufzubauen. Dem Sysop wird bei diesem Befehl unter jedem Linkeintrag der Connectweg in Klammern angezeigt, z.B.:

im Falle L4	(DB0AGM),
im Falle L2	(DB0XYZ on port 2 via DB0ZXY).

Hilfe:

Ist in einer Datei namens **CONVERSD.XHF**. Diese enthält alle Texte, getrennt durch Markierungen. Es ist erlaubt, die Texte zu ändern, Reihenfolge spielt bis auf den obersten Abschnitt keine Rolle. Die Markierungen sind durch Convers vorgegeben, Einfügen von neuen Markierungen ist daher sinnlos. Die Datei enthält Umlaute nach ISO, bitte beim Überarbeiten daran denken. Wird die Hilfe angefordert, konvertiert der Convers die Umlaute selber in die vom User gewünschte Form.

Sonstige Dateien:

CONVERS.PRS enthält die persönlichen Daten aller, die diese am eigenen Knoten eingegeben haben. Diese Datei ist ebenfalls im Textverzeichnis, wird aber beim Beenden von TNN im Configverzeichnis gesichert bzw. beim periodischen Sichern der Konfiguration. Darunter fallen der Personaltext, die Zeichensatzwahl und die Zeilenbreite.

Loop detected:

Loops werden nun überwacht und der entsprechende Link für eine Stunde „aus dem Verkehr gezogen". Das Auftreten und die Häufigkeit wird in der **CONV-C-Tabelle** angezeigt.

(**DCD**)

Das DCD- Kommando zeigt den in der Software erkannten DCD Zustand an.

```
P00 P01 P02 P03 P04 P05 P06 P07 P08 P09 P10 P11 P12 P13 P14 P15
t           rt  rt  rT   T      OFF OFF OFF OFF OFF OFF OFF OFF
```

OFF	=	Port nicht in Betrieb,
r	=	DCD an,
R	=	DCD an und noch Frames im Empfangspuffer der Vanessa,
t	=	PTT an,
T	=	PTT an, noch Frames im Sendepuffer der Vanessa.

(**DOS**) <Kommando>

Zugriff auf COMMAND.COM. Hiermit lassen sich viele Operationen durchführen. Natürlich muss hierzu COMMAND.COM auch den Zugriff auf die entsprechenden Erweiterungsprogramme haben. Doch Vorsicht ist unbedingt angesagt! Bei einem Fehler haben viele zumindest 50 km Autofahrt und evtl. einen kräftigen Fußmarsch gewonnen. Also interaktive Eingaben vermeiden! Falls es doch mal passiert kann man nur hoffen, das der Wachhund greift. Außerdem sollte man nur Programme starten, die sehr schnell wieder beendet sind, weil während der Ausführung des Programms der gesamte Knoten angehalten wird. Programme, die länger als 1 Minute laufen, führen zu einem Rechnerneustart (bei LINUX nur zur Beendung des Programms).

```
Beispiel:    DOS DIR     listet das aktuelle Verzeichnis.
             DOS DIR A:  listet LW A.
```

Unter LINUX heißt das Kommando SHELL (=> Siehe Shell für weitere Möglichkeiten), beim ATARI heißt das Kommando TOS.

(**DX**)CUSTER <call>

setzt das Call des lokalen DX-Cluster. Das Cluster muss nun nicht mehr Bestandteil des eigenen Knotens sein, sondern es kann auch ein beliebiges Cluster eingetragen werden.

(**DX**)CLUSTER -

löscht das gesetzte Call.

(**E**)DIT <filename.ext>

Legt den hinter EDIT angegebenen File-Name im Workpath an. Mit (**E**)dit kann jedoch kein bestehendes File zum Ändern aufgerufen werden. Bestehende Files werden ersetzt!

(**E**)DIT <path\filename.ext>

Auch die Angabe eines Pfades ist möglich.

Ist der Editiermodi bereits durch einen anderen Sysop aufgerufen, so bekommt man die Meldung:

Edit-Mode in use by other Sysop

(**G**)RAPH CLEAR
löscht die Graphdaten. „clear“ muss ausgeschrieben werden.

(**K**)ill <port> <msg>
Unterbricht alle Level-2 Verbindungen auf dem <port> nach Sendung der <msg>.

(**K**)ill <call> <msg>
Unterbricht alle Level-2 Verbindungen zu dem <call> nach Sendung der <msg>.

(**K**)ill * <msg>
Trennt alle Level 2 QSO auf dem Digipeater nach Sendung der <msg>.

(**L3**)MHEARD -
Entfernt Calls aus der Level 3 Statistik, deren Zähler 0 sind und setzt anschliessend die Zählerstände der Level 3-Statistik auf 0, aber löscht nicht die L3MH-Liste.

(**L3**)MHEARD - <Call>
Entfernt das <Call> aus der L3-Statistik.

(**L3**)MHEARD = <anzahl>
Setzt die L3MH-Liste auf die <anzahl> Rufzeichen. Man bedenke aber, dass sie auch gespeichert werden müssen. Die Längenänderung führt ggf. zum Verlust der am Ende der Liste stehenden Calls. Die Anzahl ist auf 5000 begrenzt.

(**L**)INKS +/- <typ> <port> <alias:call[-*]> [<digi1> <digi2>]

Nimmt <typ> mit <alias:call> über <port> in die Linkliste auf. Die Digipeater <digi1> und <digi2> können mit angegeben werden. Es werden nur Nachbarn angenommen die, unter Links eingetragen sind. Hierbei wird auch der eingetragene Port beachtet.

+/- + Eintragen und - Austragen des Links.

Typen:

L = Local
Call und Alias wird mit einer Laufzeit von 4000 ms statisch in die NODES-Liste übernommen, es wird aber keine Laufzeitüberprüfung durchgeführt. In der Routesliste wird er ohne Eintrag in Status geführt.

L+ = Local mit Erreichbarkeitstest
Wie Mode L, jedoch wird die Erreichbarkeit geprüft und die Laufzeit gemessen. Kann die Laufzeit ermittelt werden, so wird dieses Ziel zur besseren Übersicht als conn. im Status geführt.

F = FlexNet
Port arbeitet mit FlexNet Protokoll.

N = NetRom
Aktueller TheNet-Nachbar. Übernahme in Nodesliste mit der ermittelten Laufzeit.

N+ = NetRom Protokoll mit ON5ZS Änderungen
NetRom-Nachbar verwendet das neue Protokoll, wo unerreichbare Ziele sofort abgemeldet werden. Die Übermittlung und Messung geschieht im Level-3 nun protokolliert.

N- = NetRom
NetRom-Nachbar verwendet das alte Protokoll. Nicht erreichbare Ziele fallen auf Grund der Timer aus der Nodesliste. Die Übermittlung und Messung geschieht im Level-2 als UI-Frames und damit NICHT gesichert!

I = InterNode Protokoll
Nodes die in der Linkliste mit „I" eingetragen werden, werden solange keine Verbindung zwischen den Nodes besteht bei der Abfrage der Linkliste mit „N" dargestellt. Das Umschalten auf „I" geschieht nach dem Verbindungsaufbau. Dieser Linkmode sollte bevorzugt werden!!!

#ALIAS
ALIAS - Einträge die mit einem „#" davor gekennzeichnet sind, werden zwar in die örtliche Nodesliste aufgenommen aber nicht als NODE weiter verteilt.

(**LOA**)D <filename.ext> oder (**LOA**)D <path>\<filename.ext>

Lädt <filename.ext> auf die Diskette oder Festplatte beim Knoten. Der <filename.ext> darf maximal 8 Zeichen + 3 ext. lang sein und muss folgendes Format haben:

LOAD TNN179.EXE

Vorsicht bei der Auswahl des Filenamen, damit nicht das bisherige TNN-Programm versehentlich überschrieben wird. Gegebenenfalls einen noch nicht verwendeten Filenamen verwenden und nach dem Laden mit DOS REN ... (bei DOS) bzw. SHELL mv ... (bei Linux) umbenennen.

Nach der Eingabe von „LOAD <filename.ext>" wartet der Rechner auf die Übertragung eines AUTOBIN File.

```
Waiting for AUTOBIN-Transfer...
```

Nach erfolgreichem Einladen gibt der Knoten eine Checksumme (CRC) aus. Diese Checksumme MUSS derjenigen entsprechen, die auch von den o.a. Programmen ermittelt wird.

Siehe hierzu auch noch die Möglichkeiten die STARTCNT.EXE ermöglicht.

(**M**)AILBOX <call>

setzt das Call der lokalen Mailbox. Die Mailbox muss nun nicht mehr Bestandteil des eigenen Knotens sein, sondern es kann auch eine beliebige Mailbox eingetragen werden.

(**M**)AILBOX -

löscht das gesetzte Call.

(**MH**)EARD -

Löscht Calls aus der USER-Statistik, deren Zählerstände 0 sind, und setzt anschließend die Zählerstände der USER-Statistik auf 0, aber löscht nicht die MH-Liste, die bei den Userports benötigt wird.

(**MH**)EARD - <Call>
Löscht <Call> aus der USER-Statistik.

(**MH**)EARD = <anzahl>
Setzt die MH-Liste auf die <anzahl> Rufzeichen. Man bedenke aber, dass sie auch gespeichert werden müssen. Die Längenänderung führt ggf. zum Verlust der am Ende der Liste stehenden Calls. Die Anzahl ist auf 5000 begrenzt.

(**P**)ARMS <nummer>
Zeigt den eingestellten Wert dieses Parameters sowie den einstellbaren Bereich an. Ohne Angabe einer Nummer werden alle Parameter und ihr aktueller Wert angezeigt.

(**P**)ARMS <nummer> <wert>

Beispiel: P 1 20

Setzt Parameter 1 (NoAckBuf) auf 20. Die Änderung wird quittiert mit:

1: NoAckBuf = 20 (7...127)

(**PAC**)SAT <parameter>
zeigt im SYSOP-Mode die eingestellten Parameter an.

```
BROADCAST-Parms:
01:Timer     200    02:Frames    15    03:Diskfree   1000
```

(**PAC**)SAT <parameter> <nummer> <wert>

Beispiel: pac p 1 300

Setzt Parameter 1 (PacSatTimer) auf 300. Die Änderung wird quittiert mit:

1: Timer = 300 (0...10000)

Die Parameter in einzelnen:

01: PacSatTimer : 200
02: PacSatFrms : 15
03: PacMaxMail : 1000

Mit PAC P 01 wird eingestellt, in welchen Intervallen der Knoten die PacSat-Frames sendet, dieser Parameter muss so eingestellt werden, dass der TNC nicht überläuft. Mit PAC P 02 wird die Anzahl der Pakete die, gleichzeitig gesendet werden, eingestellt. Das sollte bei 9k6 20 oder so sein, (pi * Daumen 5 Frames pro Sekunde), sonst lohnt sich das nicht. Mit P 03 kann die max. Anzahl der Mail im PACSAT-Server eingestellt werden. Sind beim S&F mehr als mit PAC P 03 angegebenen Mails im Server, so werden die ältesten Mails gelöscht.

Über die Timersteuerung kann eine optimale Kanalauslastung erreicht werden, die von vereinzelten Aussendungen auf einem User-Einstieg bis zur Dauertastung alles ermöglicht, dabei werden zwei Taktiken unterstützt:

- nur PacSat-Ausgänge:
Dauertastung, für den ISM-Bereich. TNN macht dann 24-Stunden am Tag 365 Tage im Jahr Krach mit Dauerträger. Dazu ist auf dem Port, auf dem PacSat betrieben wird, ein DAMA-EPROM zu verwenden, DAMA für den PORT aber ABZUSCHALTEN. Die DAMA-Bestätigung wird benutzt, um die Dauertastung zu gewährleisten. P 02 bestimmt, wie viele Frames auf einmal zum TNC geschickt werden, der muss immer einen ausreichenden Vorrat haben, damit er die PTT nicht loslässt, das darf aber auch nicht zu viel sein, sonst läuft er über. Je nach Ringbelastung muss dieser Wert so sein, dass der TNC auch ohne, dass er Daten kriegt, eine Weile durchhält. Bei 19k2 Tokenring sollte man den Param p 02 auf 14 einstellen, bei 38k4 kann das weniger sein, bei sehr stark belastetem Tokenring mehr. P 01 gibt an, wie lange gewartet wird, wenn KEINE DAMA-Bestätigung kommt, dieser Parameter sollte auf 1000 stehen, ist aber eigentlich unerheblich (solange die Bestätigungen kommen).

- Gemischte Ein/Ausgänge:
Hier muss man entweder DAMA für den Port einschalten und ein DAMA-Eprom verwenden oder KEIN DAMA-Eprom verwenden und kein DAMA machen, DAMA-EPROM und kein DAMA geht NICHT (dann macht der Knoten Dauertastung)

Dauertastungs-Ausgänge sind unbedingt vorzuziehen, sie setzen 82 MB Daten mit 9k6 am Tag um, das ist das 10-fache eines normalen 9k6-Einstiegs. Außerdem hat jeder User was davon, nicht nur Dauersauger XYZ. Ein Broadcast-Ausgang mit 1k2 ist Unsinn und sollte nicht betrieben werden. Es muss aber unbedingt darauf geachtet werden, dass der Sender auch für Dauertastung geeignet ist. Der Tnet-Mini ist z.B. im Original nicht geeignet!

- Neben BOX kann man jetzt auch mit C PACSATCALL in die Box!

(**PAC**)SAT c <call>
Setzt das Rufzeichen des BROADCAST-Server.

(**PAC**)SAT r
Filesystem neu laden.

(**PAC**)SAT + <port>
schaltet Pacsat auf dem <port> ein.

(**PAC**)SAT - <port>
schaltet Pacsatausgabe aus.

(**PO**)RT <Port Nr.> <Befehl 1> <Befehl 2> .....
Dient zum Einstellen der Port-Parameter und der Schnittstellen.

Als Port-Befehle sind gültig:

OFF	Port abschalten
ON	Port einschalten (nur Linux),
SCC	Port auf USCC-Karte einstellen (DOS/GO32) bzw. keine TNC-Parameter setzen (nur Linux)
Tokenring	Port auf Tokenring schalten
Vanessa	Port auf Vanessa schalten
EXTdrv	Port auf den extern geladenen Treiber schalten
KISS1..4	Port auf Kiss ohne CRC
RKiss1..4	Port auf Kiss mit RMNC-CRC
SMACK1..4	Port auf Kiss mit SMACK-CRC
MODEM1..3	Port auf Modem (TNC3)
HSBUS	Port auf HSBUS (TNC3)
LOOP	Port auf Loopback
Name	Port-Namen vergeben, z.B. Name=User1k2
MODE	Port-Mode setzen, Format: 9600dz Baudrate: (300...4915200) Flags: (c...z) Zugelassene Flags sind: c: DCD bei 1k2-Modem d: Duplex r: ext. Takt (rx) t: ext. Takt (tx) e: ext. Takt beide (Vanessa) m: Multibaud-Kopplung (Vanessa, SCC) z: NRZ statt NRZI
MAXFRAME <wert>	setzen
MAXFRAME <wert>A	Maxframe setzen mit Maxframe-Automatik (s.u.)
MAXCON 0..10	Anzahl der Connects von Usern auf diesem Port begrenzen (0=aus)
TXDelay <wert>	setzen
DAMA 0/1..16	Port auf DAMA Master 1 bis 16 setzen, 0= aus es können mehrere Ports einem DAMA-Master zugeordnet werden
CTEXT on/off	Ctext senden bei Connect
SYSOP on/off	nach Connect nur SYS möglich, gilt nur für Verbindungen zum CCP, NETROM und TCPIP gehen weiterhin
MH on/off	MH-Liste führen
EAXMAXF 1..32	Maxframe bei EAX von 1 bis 32 Frames setzen. 16 Frames sind default
EAXMODE 0..2	Extended-AX.25 setzen.. 0 bis 2. 1 ist default.

(**OF**)f: Port ausgeschaltet.

(**ON**): Nur bei Linux: Port eingeschaltet. Da im File tnn.ini eine feste Zuordnung zwischen den Ports und der Hardware sowie des KISS-Modus eingestellt wird, ist es nicht erforderlich, die Hardware einzustellen. Der Parameter ON ist daher der sicherste, aber KISS ist auch immer gültig zum Einschalten.

(**TO**)kenring: Frames an diesen Port werden auf die Tokenring-Schnittstelle gelegt. Die Tokenring-Schnittstelle liegt default auf COM 1, wenn sie nicht in der START.BAT anders definiert wird. Bei Linux wird die Bezeichnung TOKENRING nur akzeptiert, wenn auch in tnn.ini der Port als Tokenring-Port deklariert ist.

...(**MO**)de: Keine Auswirkung auf den Tokenring, aber auf die L2-Timer !

(**VA**)nessa: TNN ist sowohl für den Tokenring als auch für den gleichzeitigen Einsatz von Vanessakarten geeignet. Für Frames an diesen Port wird, über den in der TNN-Soft implementierten Vanessatreiber, die entsprechende Vanessakarte über den Rechnerbus angesprochen und unter (PO)rts wird der entsprechende Port mit „Vanessa" gekennzeichnet (Jedoch NUR, wenn auch eine VANESSA eingebaut ist). Bei Linux wird die Bezeichnung VANESSA nur akzeptiert, wenn auch in tnn.ini der Port als Vanessa-Port deklariert ist.

...(**MO**)de: <speed>**d**: Vollduplex. Bei der Vanessakarte bleibt der TX nach der letzten Sendung noch für ca. 1 Minute getastet. Weitere Frames werden ohne TXDelay gesendet. Beim TNC im Tokenring ist diese Zeit im Eprom festlegbar.

...(**MO**)de: <speed>**e**: Externer Takt.

...(**MO**)de: <speed>**m**: Dual - Speed - Port (Nur mit und auf den beiden Ports der entsprechenden Vanessakarte möglich).

(**SCC**): Die BayCom USCC-Karte wird bei der DOS- bzw. GO32-Version nun intern unterstützt! Bei Linux bedeutet der Parameter SCC, dass für diesen Port keine Initialisierung der L1-Parameter erfolgt (TXD, Persistence, Slottime, etc.). Dies ist Aufgabe eines Initialisierungsprogramms, das vor TNN aufgerufen werden muss - alternativ gelten die TNC Default-Werte. Der Parameter SCC ist bei Linux nur wirksam, wenn der Port in tnn.ini entsprechend eingetragen ist. Die Mode-Werte werden bei Linux ignoriert.

...(**MO**)de: <speed>**c**: Software DCD für AFSK-Modems.

...(**MO**)de: <speed>**e**: Externer Takt. (Für DF9IC-Modem).

...(**MO**)de: <speed>**z**: NRZ statt NRZI. (Für DF9IC-Modem).

Hinweis: Soll ein DF9IC-Modem an eine Vanessa angeschlossen werden, so benötigt es ein NRZ-GAL! Mit dem NRZI-GAL soll der externe Clock nicht funktionieren.

(KISS1)..(KISS4): Die Frames an diesen Port werden über den in der TNN-Soft implementierten KISS-Treiber, an die in der START.BAT definierte COM-Schnittstelle, über den Rechnerbus geleitet. Unter (**PO**)rts wird der entsprechende Port mit „Kiss1 .. Kiss4" gekennzeichnet und mit (**MO**)de der Speed sowie das CRC-Verfahren festgelegt. Bei Linux kann mit dem Parameter KISS jede Port-Art eingeschaltet werden. Die Zahl für den KISS-Port wird bei Linux ignoriert.

(SMACK1)..(SMACK4): KISSLINK mit SMACK CRC. Bei Linux wird die Bezeichnung SMACK nur akzeptiert, wenn auch in tnn.ini der Port als SMACK-Port deklariert ist. Die Zahl für den SMACK-Port wird bei Linux ignoriert.

(RKISS1)..(RKISS4) : KISSLINK mit RMNC-CRC. Bei Linux wird die Bezeichnung RKISS. nur akzeptiert, wenn auch in tnn.ini der Port als RKISS-Port deklariert ist. Die Zahl für den Kiss-Port wird bei Linux ignoriert.

Anmerkung: Bei EXTERNEN-Treibern wird der TYP der Hardware direkt vorgenommen und ist nach dem Starten der Software bereits unter dem jeweiligen Port sichtbar.

DAMA 0/ 1...16 Port auf DAMA Master 1...16 setzen.

Bei TNN sind mehrere DAMA-Kanäle möglich; dabei wird als DAMA-Kanal die Zusammenschaltung mehrerer DAMA-Ports bezeichnet. Es sind so viele DAMA-Kanäle verfügbar wie Ports, also normalerweise 16. Somit kann man mit TNN auf mehreren Multi-Baud-Einstiegsfrequenzen im DAMA-Modus arbeiten. Für jede Einstiegsfrequenz wird ein DAMA-Kanal verwendet. Beim Port-Befehl wird statt des DAMA-Flags jetzt der DAMA-Kanal angegeben. Die DAMA-Kanäle sind von 1 bis 16 numeriert und mit Kanalnummer 0 wird die DAMA-Funktion des Ports ausgeschaltet. In der Liste der L2-User wird zusätzlich zur Priorität auch die Nummer des DAMA-Kanals angezeigt.

Beispiel:
DAMA-Kanal 1: Port 0 = 1k2 auf 70cm
Port 1 = 4k8 auf 70cm
Port 2 = 9k6 auf 70cm

DAMA-Kanal 2: Port 3 = 9k6 auf 23cm
Port 4 = 19k2 auf 23cm

Bei der Linux-Version funktioniert DAMA NUR mit Tokenring oder Vanessa!

(ACHTUNG DAMA-BIT im EPROM der Tokenring-Software brennen!)

CTEXT on / off Ist CTEXT on wird generell der CTEXT.TXT gesendet. Ist noch zusätzlich ein CTEXT.n (n=Portnr.) vorhanden, so wird auch dieser auf den entsprechenden Port gesendet. Ist CTEXT=0, so werden auch evtl. vorhandene <call>.MSG Files nicht gesendet !!!

SYSOP on / off Setzt den angegebenen Port auf SYSOP-Mode. Diese Funktion soll dem Sysop helfen, seine Wartungsarbeiten durchführen zu können. Der Knoten nimmt ohne eine SYSOP-Priviligierung auf diesem Port keine Kommandos mehr an. Die Umstellung kann auch während des Betriebes erfolgen, ohne dass die User disconnected werden. NetRom und TCPIP gehen weiterhin.

Semiduplex Links können auch auf SYSOP gesetzt werden. Dadurch wird ein unerlaubtes „Einsteigen" auf den Links verhindert. Der Level 3-4 Link zum Nachbarn bleibt unberührt.

MH on / off Das Führen der MH-Liste auf den Interlinks kostet Rechnerzeit und die MH-Liste wird unübersichtlich. Sie wird geführt, um bei mehreren Userzugängen mehr Transparenz über die Zugänge zu erhalten UND damit der KNOTEN weiss, auf welchem Port er einen bestimmen User connecten muss.

EAXMODE 0..2 Steuert die Verwendung von erweitertem AX.25 auf einem Port. Es sind die folgenden Werte möglich:

0: nur AX.25 zugelassen, EAX.25-Verbindungen werden abgelehnt

1: Mode nach je Fähigkeiten der Gegenstation bzw. nach MHeard. (AX.25 und EAX.25) Dieser Mode ist standardmäßig aktiviert

2: nur EAX.25 zugelassen, AX.25-Verbindungen werden abgelehnt.

EAXMAXF 1..32 Maxframe kann zur Zeit 32 Frames sein. Default ist 16 eingestellt.

(**NA**)me: Gibt diesem Port eine spezielle Bezeichnung. Sie darf maximal 10 Zeichen lang sein und dient der Unterscheidung der Ports. Wenn z.B. mehrere Userzugänge oder Baudraten benutzt werden, ist in der MH-Liste ersichtlich, welches Call auf welchem Zugang QRV ist. Weiterhin wird der Portname benutzt, um auf einem bestimmten Port einen Connect auszusenden, unabhängig vom Eintrag in der MH-Liste. DB0EAM hat einen „70cm_1200" und einen „70cm_9600" Zugang sowie nun auf 23cm einen „23cm_9600". Die Port-Namen sind auch so eingestellt. Mit „c DB0XYZ 70cm_9600" kann ein Connect auf dem 9600_Bit/s Port ausgesendet werden. Wird kein Port-Name beim Connect eingesetzt, so wird Port 0 benutzt, ist das Call in der MH-Liste vorhanden mit einem Eintrag auf einen anderen Port als Port 0, so wird dieser Port benutzt.

Maxframe-Automatik:

Um die Zahl der Framewiederholungen für Stationen mit schlechtem Empfänger zu reduzieren, kann der Maxframe-Wert automatisch angepasst werden. Bei eingeschalteter Automatik wird der Maxframe-Wert für den jeweiligen Link jedesmal um 1 reduziert, wenn die Gegenstation ein REJ sendet, wenn von der Gegenstation keine Bestätigung kommt, oder wenn nur ein Teil der gesendeten Frames bestätigt wird. Wenn von der Gegenstation anschliessend 2 mal in Folge alle gesendeten Infos bestätigt wurden, erhöht sich der Maxframe-Wert für den Link wieder um 1. Wer grundsätzlich schlecht empfängt belegt dadurch einen Einstiegskanal mit weniger Wiederholungen, während ein kurzes Empfangsproblem bei einer normalerweise gut funktionierenden Station nur zu einer vorübergehenden Reduktion des Datendurchsatzes führt. Auf Linkstrecken sollte die Automatik besser ausgeschaltet bleiben.

Autoparameter:

Einige Parameter für die Ports werden automatisch von TNN eingestellt, so dass nur noch MaxFrame und TXDelay übrig bleiben. Diese werden mit dem PORT Kommando eingestellt.

T2 = 2888 / (Baudrate / 100)

bei DAMA wird T2*2 genommen (Idee DG3AAH)

IRTT = (T2 + TXDelay) * 2.

Der IRTT wird beim Connect mit der Anzahl der noch zu digipeatenden Stationen multipliziert.

Retry = 10 bei DAMA, sonst 20.

Persistance ist bei Duplex und DAMA 255, ansonsten 255/User. Bei 0 Usern auch 255.

Es werden nur die zum Knoten connecteten User berücksichtigt. Der Knoten nimmt also nicht auf Schwarzfahrer auf der gleichen QRG Rücksicht.

Slottime = TXDelay

(**PR**)OMPT

Ohne Text zeigt die derzeitige Einstellung an:

Prompt: %c de %d (%t) >

(**PR**)OMPT = <text>

Übernimmt den String als Prompt. Der für den Prompt verwendete <text> beginnt direkt hinter dem „=„! Enthält der eingegebene String die Zeichen %c, %d, %t, %0, so werden diese im späteren Prompt wie folgt umgewandelt:

%a	In den Alias des Knotens,
%c	In das Call des User,
%C	In das Call des User mit SSID,
%d	In das IDENT des Knotens,
%D	In das IDENT des Knotens mit SSID,
%r	In ein Return,
%t	In die momentane Uhrzeit in HH:MM,
%s	In das aktuelle Datum,
%0	Verhindert die Aussendung des Promptes.
%l	Anzahl aktiver L2-Links
%p	Portnummer
%P	Pseudo-Name des Ports
%u	Anzahl der User auf dem aktuellen Port
%%	%

PR =%c de %d (%s %t) > ergibt z.B.: DG9FU de DB0EAM (01.04.00 18:30) >

Es hat sich eingebürgert, im gesamten PR-Netz UTC zu verwenden. Dies erspart einem auch das lästige Neusetzen der Sommer-/Winterzeit.

Diese und die anderen Prompt-Makros funktionieren auch in CTEXT.TXT, QUIT.TXT und CTEXT.<portnummer> !!!

(**RE**)AD

Dient zum Lesen eines Files. Der Name muss mit Erweiterung angegeben werden. z.B.:

READ CTEXT.TXT oder READ AKTUELL.TXT.

Hier sind auch Pfadangaben erlaubt. Diese sind notwendig, wenn ein Text vom Laufwerk A: gelesen werden soll und der DOS-Pfad auf ein virtuelles Laufwerk zeigt. z.B.:

READ A:\TNN\AKTUELL.TXT.

(**READB**)IN <filename.ext>
Erlaubt das binäre Herunterladen von Files. Das Verfahren ist das gleiche wie unter LOAD, nur in der Richtung vom Knoten zum Sysop.

(**RES**)ET SYSTEM
Lässt den Knotenrechner einen Kaltstart ausführen. Bei LINUX wird TNN lediglich beendet. Wird hier ein Neustart des Knotenrechners benötigt, muss man den Befehl „SHELL shutdown -r now" verwenden.

(**RES**)ET <port>
Führt einen Reset des Port-TNC bzw. der entsprechenden Vanessakarte aus.

(**RU**)NBATCH <filename.tnb>
Führt das angegebene TNB-File aus und gibt die Bestätigung „ OK " zurück. Der Befehl RUNBATCH kann auch in einem TNB-File eingetragen werden. Er kann beliebig zwar verschachtelt werden, aber hier ist Vorsicht geboten, weil eine unendliche Rekursion nicht verhindert wird - das führt dann zur Blockade des gesamten Rechners, weil die Festplatte mit einem Temporär-File gefüllt wird.

(**SH**)ELL <kommando>
Nur bei Linux: Zugriff auf die System-Shell. Hiermit lassen sich alle verfügbaren Programme starten. Doch Vorsicht ist unbedingt angesagt! Bei einem Fehler hat man evtl. eine längere Autofahrt und einen kräftigen Fußmarsch gewonnen. Allerdings können auch länger laufende Programme gestartet werden, als bei der GO32-Version, weil die Programme als eigener Prozess laufen, der Knoten steht also nicht während der Ausführung. Die Prozessdauer ist allerdings auf 1 Minute begrenzt. Mit einer einfachen Eingabe kann der Prozess auch vorzeitig beendet werden.

Weiterhin ist unter Linux eine Interaktive Shell möglich: Wird bei „SHELL" kein direkt auszuführender Befehl angegeben, so erhält man eine interaktive Shell. Befehle wie z.B. „sh ls –l" werden wie bisher direkt ausgeführt. Der Timeout für nicht-interaktive Befehle beträgt weiterhin eine Minute, die interaktive Shell hat hier fünf Minuten mit Timeout-Warnung.

(**S**)TAT + <call>
Nimmt <call> in die Statistik auf.

(**S**)TAT + <call> <viacall>
Nimmt <call> <viacall> in die Statistik auf. Bei Call wird hier auch ein „*" akzeptiert, wenn alle via Frames protokolliert werden sollen. Beachtet werden muss, dass z.B. „s + db0xx db0yy" VOR „s + * db0yy" eingetragen werden muss!

(**S**)TAT - <call>
Löscht <call> aus der Statistik.

(**SP**)ARM oder besser **S**ave **P**arameter
Save Parameter erstellt ein File PARAMS.TNB, das alle derzeit eingestellten Parameter beinhaltet. PARMS.TNB kann, umkopiert nach TNN179.TNB, zum Einstellen der Parameter von TNN beim Neustart verwendet werden (aber vorher auf Vollständigkeit prüfen!). Als Bestätigung kommt „PARMS.TNB saved ..."

(**STAR**)T <programm>
Startet das angegebene Programm. Der Programmname muss vollständig angegeben werden, z.B.: Start TNNDOS.EXE. Befindet sich das Programm nicht im aktuellen Pfad, so muss dieser selbstverständlich mit angegeben werden. Der Ablauf des Startvorganges wird nun nicht mehr auf dem Bildschirm angezeigt, sondern in eine Datei STARTUP.LOG geschrieben.

(**STAR**)T <programm> funktioniert NICHT bei den DPMI-Versionen!

(**SUS**)PEND + <port> <call> <links>
Beschränkt das Rufzeichen <call> auf Port <port> auf eine Anzahl von <links>.

Beispiele:

SUS + 255 DB2OS 2
DB2OS soll auf dem Knoten generell nur 2 Verbindungen , egal auf welchem Port, aufbauen können. Dann wird unter <port> einfach der fiktive Port 255 angegeben.

SUS + 3 DB2OS 255
Schliesst DB2OS auf Port 3 komplett aus.

SUS + 3 DB2OS
Schliesst DB2OS komplett aus.

Ein Level-4 Ausschluss erfolgt mit:

SUS + 254 DB2OS

Der Connect des Knotens ist erstmal möglich. Jedoch nach dem ersten I-Frame des Users bekommt er das File SUSPEND.TXT zugesandt und danach einen Disconnect.

(**SUS**)PEND - <port> <call>
Gibt dem Rufzeichen <call> auf Port <port> wieder das Benutzen des Knotens frei.

(**SY**)SOP
Gibt eine 5er Zahlengruppe aus für das Einloggen des Sysop.

PASSWORT-Eingaben:
Die Passworteingabe für den User wurde erweitert, so dass auch Passworte wie bei BAYCOM-BOX erlaubt sind. Die alten 5-stelligen Antworten gehen auch. Der Trick besteht darin, das eigentliche Passwort in einem langen String Datenmüll zu verstecken!

BAYCOM-PASSWORT:
Das wichtigste zuerst, die Methode ist kompatibel zur alten, man kann ohne weiteres einfach die 5 Antwortbuchstaben senden. Zusätzlich ist es aber möglich, die Antwort in einer Zufallszeichenkette zu verstecken. Ist die Lösung z.B. 12345, werden folgende Antworten auch als korrektes Passwort akzeptiert:

```
5432153241234522113425423524424241552 6
         -----

12345
-----

12131415234132145132412345
                      -----
```

Da dem Lauscher nicht bekannt ist, wo die Lösung ist, kann er sich das Passwort nicht zusammenbasteln. Es empfehlen sich also Passwörter mit möglichst vielen verschiedenen Zeichen, um die Verschleierung zu erhöhen.

(**TE**)ST <port>
Gibt einen 0/1 Wechsel auf dem angegebenen Port von 4 kByte Länge aus, für Abgleicharbeiten z.B. am Modem. Die Testfunktion kann nur noch der Sysop auslösen. Ruft der User die Testfunktion auf, so bekommt er ein „Invalid command. Try HELP." zurück.

(**TI**)ME
Zeigt Datum und Uhrzeit des Knotens an.

Einstellen von Uhrzeit und Datum ist nur über die DOS-Befehle möglich!

Zeit mit:	**DOS TIME HH:MM**
Datum mit:	**DOS DATE TT.MM.JJ**
oder mit:	**DOS DATE TT.MM.JJJJ**

Doch Vorsicht! Nur „DOS TIME" zeigt die aktuelle Uhrzeit auf dem Monitor des Knotens an und erwartet die neue Uhrzeit per Tastatur. Der Knoten steht dann bis zum RESET. Dieser erfolgt automatisch nach ca. 1 Minute durch den Software-Watchdog von TNN.

Ein wenig Mitdenken gehört auch dazu.

(**TR**)ace <level> <monitor>
Dieser Befehl dient zum erkennen und finden von Fehlern. Im Prinzip besteht der Trace aus 2 Funktionen. Zum einen die Ausgabe von Ereignissen und Fehlern und einem zusätzlich einschaltbaren Monitor. Es stehen folgende Module zur Verfügung. Dazu ist der Level mit einem Wert zwischen 0 (Aus) und 9 (Alles) einstellbar. Der Trace wird, wenn mehr im Monitor zu senden sind als nach Parameter NoAckBuf eingestellt ist, nicht abgebrochen sondern nur unterbrochen. D.h. es können, wenn man zu viel monitoren will, Lücken entstehen. Will man eine komplette Liste ohne Lücken so startet man den Trace von der Knoten - Console aus und lässt den Trace auf die HD schreiben. Mit RUNBATCH MONI_ON.TNB kann man den Trace starten und durch Aufruf einer MONI_OF.TNB ihn wieder beenden. In der MONI_ON.TNB leitet man den Monitor mit esc @E1 in ein File um und startet dann den Trace entsprechend. Mit MONI_OF.TNB wird der Monitor wieder auf esc @E0 gestellt und der Trace mit TRACE 0 MN ausgeschaltet.

Beispiel: (**TR**)ace 9 (ALLE anzeigen, wirklich alles)

9 = reine Information
empfangene Nodes-Info
7 = wichtige Information
fastlearn <call> via <call>
<call> maxtime -> <zeit> ms
5 = wichtiges Ereignis
unreachable source node <call>
own frame to <call> received fm <call> (loop)
frame fm <call> to <call> via <call> flushed (neigb loop)
Max <call> - <call> <alt>-><neu>
3 = nicht kritischer Fehler
invalid node <call> > <call> from <call> (flushed)
invalid alias from <call> (flushed)
invalid ipa from <call> (flushed)
<call>: l3rtt too high: <zeit>
Linkreset <call>
<call>-><call>: too many frames in queue destination <call> received from <call> (ignored)
1 = kritischer (fataler) Fehler
Frame rejected <call> -> <call> .....
send to disc'ed link <call> > <call> via <vialiste>
token lost <call>
INP: frame error received from <call>, len = <n>, left = <n>
0 = aus

Der Monitor erlaubt auch ONLINE ein Call oder einen Port zu monitoren, ohne es zuerst in ein File im Knoten zu schreiben.

MNLUSTIC H/F <port nr.> +/- <call..call>
M = wird ignoriert (Kompatibilität zu älteren Versionen)
C = wird ignoriert und immer als eingeschaltet gekennzeichnet
(Kompatibilität zu älteren Versionen und zum Monitor-Befehl
an der Konsole)

Monitoren von:
N = Monitorfunktion ausgeschaltet.
L = Die Länge des Info-Feldes (abzüglich L3/4-Header) wird
bei I- und UI-Frames angezeigt
U = Unprotokollierten Paketen,
I = Info Paketen,
S = Kontroll Paketen,
T = Timestamp einschalten, also Datum und Uhrzeit
werden mit angezeigt,
H = HEADER, bei I- und UI-Frames nur den Header anzeigen,
der Infoteil der Frames wird unterdrückt,
F = FULL, I- und UI-Frames mit Infoinhalt anzeigen,
es werden die kompletten Frames gemonitort,
<port nr> = Nur der angegebene Port wird gemonitort,
+ <call..call> = Nur diese Call (bis zu 8) monitoren,
- <call..call> = Diese Call im Monitor unterdrücken.
<call> kann mit SSID angegeben werden.
Unter <call> sind auch Eingaben wie QST und LSTAT möglich.

Beispiel: (**TR**)ace 1 MUSICH 9 + DG0XX.
TNN meldet die Einstellung als Quittung zurück.

Beim Trace einer IP-Verbindung ohne den Infoinhalt werden die Frames in dieser Art dargestellt:

```
T15: fm DB0EAM to DB0EAM-10 I52^pid CC TCP/IP
(IPV4 fm 44.130.149.20 to 44.130.27.81 IHL=5 TOS:9 Length=216 ID=15344
Flags:MF=0 DF=1 FraOff=0 TTL=123 TCP)
TCP Packet: SrcP:61005 DstP:11308 SeqN=5a85523 AckN=eeb5911b Wind=8272
DataLen=176 DataOffset=5, CRC=ae17 Urgent=0
Flag: PUSH
Flag: ACK
```

IP-Header

IPV4	= IP Version 4,
IHL	= Internet Header Length ; Länge des IP-Headers in 32 Bit-Worten,
TOS	= Type of Service,
Length	= Länge des IP-Headers und Datenfeld in Bytes,
ID	= ID des IP-Packetes zur Identifikation der Fragmente eines Datenfeldes,
MF	= MoreFollow (1 = es folgenden weitere Fragmente; 0 = es folgen keine),
DF	= Don't Fragment (1 = Packet darf nicht fragmentiert werden; 0 = darf),
FraOff	= Fragment Offset; Position des Fragmentes im Datagramm in 8 Byte Einheiten,
TCP	= Protokoll TCP (Transport Control Protocol).

TCP-Header

SrcP	= Port Nummer der Quelle,
DstP	= Port Nummer des Ziels,
SeqN	= Sequence-Nummer des ersten Datenbytes in diesem Segment,
AckN	= Bestätigungsnummer, nur gültig wenn ACK-Flag gesetzt,
Wind	= Fenstergrösse; Grösse des Puffers, der für diese Verbindung reserviert wurde,
DataLen	= Länge der Daten,
DataOffset	= Anzahl der 32-Bit-Worte im TCP-Header,
CRC	= sagt der Name schon,
Urgent	= Zeiger auf das Ende der dringlichen Daten. Nur gültig bei gesetztem URG-Flag.

Flags:

URG	= Urgent, Dringlichkeitszeiger,
ACK	= Acknowledgement-Number-Feld enthält gültigen Wert,
PSH	= Push, weist den Empfänger des Pakets an, die enthaltenen Daten sofort, an die Anwendungsschicht weiterzuleiten,
RST	= Reset, Verbindung muss unterbrochen werden,
SYN	= Synchronize, zum Abstimmen der Seq-Nummer beim Verbindungsaufbau,
FIN	= Verbindung beenden.

TCPIP Kommandos

(**ARP**) <DestIP> + <publ.> <Port> DG/VC <call[-*]> <[digi1> <digi2>]

bzw.

(**ARP**) <DestIP> - <Port>

Nimmt <destip> mit <publish> <port> als DC/VC als <call> in die Linkliste auf. Die Digipeater <digi1> und <digi2> können mit angegeben werden.

ARP-Einträge macht TNN automatisch. Sie müssen nur für Wege über via-Digipeater und NET/ROM-Verbindungen angeben werden, da hier ARP nicht möglich ist.

ARP-Einträge werden für Gateways gemacht, bzw. für direkt erreichbare Stationen.

DestIP	Ist die IP-Nummer des (IP-) Nachbars auf diesem Port
+/-	Zum Ein- oder Austragen
Publ.	Publish soll dieser Port/Nachbar allen anderen auch bekannt gegeben werden? Ein P steht für „ARP beantworten", wird es weggelassen, werden ARP-Anfragen an dieses Ziel ignoriert.

Port	Portname des Knotens über den die IP-Adresse erreichbar ist. Es kann hier auch wieder NETROM angegeben werden. Vorsicht! Schreibfehler im Portnamen führen evtl. zum Absturz (wir arbeiten noch daran).
DG/VC	Datagram/Virtual Circuit. Soll das ganze Verbindungslos aufgebaut werden (Datagram) oder soll eine Verbindung da sein (Virtual Circuit). Von Digi zu Digi sollte DATAGRAM eingestellt sein, dann gibt es weniger Overhead. Dann findet allerdings auch keine Fehlerkorrektur auf dem Link statt, dies wird aber (zwar etwas langsamer) von den Endstationen (TCP/IP) gemacht. Auf DAMA-Einstieg ist UNBEDINGT VC einzustellen. Bei NETROMs hat dieser Parameter keine Bedeutung.
Call	AX-25-Call, unter dem der Rechner mit der IP-Nummer <destip> erreichbar ist.
Beispiel:	ARP 44.130.82.41 + P Sysop_9k6 VC DB7KG DB7KG-5

Was geht, und was nicht geht:

- Der IP-Router von TNN kann Frames beliebiger Länge durch Fragmentierung übertragen. Die Reassemblierung wird der Endstation überlassen.
- TNN empfängt aus Kompatibilitätsgründen zu FlexNet nur Pakete bis 256 Bytes Info.
- ARP/REVARP-Anfragen werden sofern erlaubt, beantwortet.
- Über INP empfangene Knoten mit IP-Adresse erhalten automatisch einen ARP-Eintrag und zusätzlich einen IPR-Eintrag. Ist die Gegenstelle auch ein TNN-Knoten, so konfiguriert sich das IP-Routing zu diesem Knoten vollkommen automatisch !!! Von Hand gemachte ARP-Einträge haben Vorrang vor automatisch gemachten Einträgen des Routers.

(**IPA**)dress 0.0.0.0/32

Setzt die IP-Adresse des Knotens. Für Subnetting ist die Anzahl der Subnetzbits anzugeben, fehlt diese, so wird 32 angenommen.

(**IPR**)oute <destip>/<maskbits> +/- <interface> <gateip> <metric>

Nimmt <destip>/<maskbits> auf <intercace> <port> <gateip> in die IP-Routenliste auf.

DestIP	Ist die IP-Nummer des (IP-) Nachbars auf diesem Port
Maskbits	Hier wird entweder eine Zahl (Anzahl Bits) oder eine Art Pseudo IP-Nummer eingetragen. Anzahl der Bits heisst: Eine IP-Nummer hat 4 Bytes mit 8 Bit = 32 Bit. Trägt man hier 16 ein, so werden nur die ersten 2 Byte ausgewertet (entspricht für AFU: 44.130; 44=AFu 130=DL). Trägt man 24 ein, wird schon die Region beachtet, also 3 Bytes (44=AFU 130=DL 27=KS) bei einer (Beispiel) Nummer 44.130.27.80. So kann man hierarchisch Routen aussortieren. Bei der anderen Möglichkeit wird für das Byte, was frei bleibt eine 0 eingetragen.

Bsp.: 44.130.27.0 = Kassel.

Kann auch mit Broadcast zusammen so aussehen:

255.255.255.0 oder 44.130.27.255..kommt ganz darauf an

+/-	Zum Ein- oder Austragen
Interface	Hier wird die Schnittstelle benannt, wo die Pakete für <DestIP> herauskommen sollen. Dies kann der NAME eines Ports sein (also User_1k2, User_9k6). Es kann auch „NETROM" angegeben werden. In diesem Fall wird das Packet an ein NETROM-Ziel (das muss natürlich auch ein Gateway [TNN] oder ein TCP/IP-Host [KA9Q] sein) übertragen. Unter Linux existiert das zusätzliche Interface KERNEL, welches eine direkte IP-Anbindung an den Linuxkernel ermöglicht.
GateIP	Wird ein Gateway benannt, so wird hier die IP-Nummer des Gateways eingetragen. Über ihn werden dann alle (IP-)Packets vermittelt.
Metric	Diese Angabe wird in einer späteren Version Bedeutung haben, zurzeit bitte ignorieren.
Beispiel:	IPROUTE 44.130.27.0/24 + User_9k6

ESC/ALT Kommandos

<ALT> X

Bricht das Programm ab und geht auf die DOS-Ebene zurück. Bei Linux geht das nur mit <ESC> QUIT.

Die <ESC>-Befehle (<ESC> steht hier für die entsprechende Taste auf der Konsolentastatur) steuern die Konsolenfunktionen. Einige dieser Befehle können auch per Fernbedienung eingegeben werden (s. Befehl ESC).

<ESC> B

Anzeigen Rounds/s

<ESC> C — Connect von der Tastatur als HOST in den Knoten. Im Terminalmodus ist man automatisch als Sysop privilegiert.

<ESC> D — Disconnect vom Knoten

<ESC> @E1 — Öffnet ein neues Protokollfile YYMMDDHH.PRO. Wenn das File bereits existiert, wird es nicht gelöscht, sondern die Ausgaben werden ans Ende angehängt. Die Monitorausgaben des Batch-Kanals werden in dieses Protokoll-File umgeleitet. ACHTUNG! Die Ausgaben auf dem Konsolenbildschirm werden NICHT mehr in das Protokollfile geschrieben. Das geht nur noch von einem TNB-File aus (in dem der passende TRACE-Befehl eingetragen ist).

<ESC> @E1 Pfad\Filename — Wie bei <ESC> @E1 aber Ausgabe in die Datei Pfad\Filename.

<ESC> @E0 — Schaltet die Ausgabe ins Protokollfile ab und schliesst das File.

<ESC> I <call> — Einstellen des HOST Calls für den aktiven Hostkanal.

<ESC> I <call> <alias> — Setzen des globalen Host-Calls, wenn <call> nicht mit dem TNN-Rufzeichen übereinstimmt. Gleichzeitig mit dem Setzen des Calls wird zusätzlich ein Link-Eintrag für <call> mit <alias> vorgenommen auf dem (virtuellen) Port 16 (bei der 16 Kanal-Version). <call> und <alias> können dann vom User connected werden, entweder mit dem Befehl „C" ohne Parameter oder mit dem entsprechenden Call als Parameter. Soll der Linkeintrag wieder gelöscht werden, wird das TNN-Rufzeichen eingegeben, aber auch hier wird ein <alias> benötigt (der aber ignoriert wird).

<ESC> K — Anzeigen der Einstellung für die Datums-/Zeitanzeige bei Statusmeldungen.

<ESC> K0 — Statusmeldungen werden ohne Datum und Uhrzeit dargestellt.

<ESC> K2 — Statusmeldungen werden mit Datum und Uhrzeit dargestellt.

<ESC> Logout — Verlassen der Konsole und verschließen.

<ESC> L — Zeigt den Status der (30) Host-Kanäle an.

<ESC> M — Zeigt die derzeitige Einstellung an. (Bedeutung der Parameter siehe unten)

ACHTUNG! — Bei den Monitorbefehlen ist, bei stark belasteten Knotenrechnern, Vorsicht geboten. Da der Rechner dann um einiges langsamer wird, bekommt man das Monitoren nur schwer wieder ausgeschaltet, da die Tastatur kaum noch abgefragt wird.

<ESC> MN — Monitor aus. (Je mehr der Monitor anzuzeigen hat, desto langsamer wird der Knoten. Also nach Gebrauch immer den Monitor wieder auf „AUS" schalten!

<ESC> MLUSTIC H/F <port nr.> +/- <call..call> (Siehe oben ESC MLUSTIC H/F <port nr.> +/- <call..call>)

<ESC> QUIT — TNN beenden.

<ESC> R0 — Es erfolgt kein Hinweis auf Token-Recovery.

<ESC> R1 — Bei jedem Token-Recovery wird auf dem Bildschirm der Text
*** Token-Recovery (1) Wed Jan 19 13:45:10 1994 ***
angezeigt. Die Ziffer gibt an, wie viele Recovery nacheinander aufgetreten sind, sowie Datum und Uhrzeit. Dieses ist die default Einstellung.

<ESC> S	Zeigt den Aktuellen Host-Kanal an.
<ESC> Sn	Schaltet zum Kanal n um. (Gültige Kanäle sind 1 - 30)
<ESC> T	Zeigt die momentan auf dem Tokenring eingestellte Baudrate. Dieser Befehl gilt nicht für die Linux-Version.
<ESC> T 9600 / 19200 / 38400	Stellt die entsprechende Baudrate auf dem Tokenring ein.
<ESC> T 57600 / 115200	Stellt die entsprechende Baudrate auf dem Tokenring ein, für diese Baudraten ist jedoch ein FIFO-Baustein (16550) erforderlich.
<ESC> T ?	Gibt eine Hilfe aus.
<ESC> V	Anzeigen der Hostmode-Version.
<ESC> Y0	Kein Connect zum HOST möglich (default).
<ESC> Yn	Verbindungen zum HOST sind auf n Hostkanälen freigegeben (n = 1 bis 30). Der Connect zum HOST ist mit C möglich **OHNE** den Zusatz eines <**call**>, sowie mit C <CALL>, wobei für <CALL> das globale Host-Call gilt.
<ESC> @B	Anzeige der für den Hostmode verfügbaren freien Buffer (1/4 der Buffer am Knoten).
<ESC> @Y	Anzeigen der maximal verwendeten Host-Kanäle.
<ESC> @Y<n>	Einstellen der maximal zu verwendenden Host-Kanäle (für den Terminalmodus oder ein Hostmodeprogramm, das nicht mit den 30 Host-Kanälen arbeiten kann / soll. Hier ist zu beachten, dass der aktive Kanal in dem erlaubten neu zu setzenden Bereich liegt, also wenn der aktive Kanal 20 ist, kann man nur auf 20 .. 30 Host-Kanäle einstellen, nicht aber auf 19. Ist einer der in Zukunft nicht mehr zu verwendenden Kanäle noch connected, wird er automatisch disconnected. Anschliessend werden mit dem Befehl <ESC> L nur noch die verwendbaren Kanäle angezeigt und man kann auch nicht auf einen Kanal mit höherer Nummer umschalten (mit <ESC> S).

(HOST)mode-Befehle

ACHTUNG! Die Mailbeacon-Funktion ist geändert gegenüber TNN 1.77!

	Im Hostmode stehen (fast) die gleichen Befehle zur Verfügung, wie im Terminalmodus. Es wird allerdings das von TF her bekannte Hostmdode-Protokoll verwendet. Folgende Hostmode-Befehle werden von TNN 1.78 unterstützt:
C	Connect zum Knoten. Ausserdem auf dem Monitor-Kanal (0) Anzeigen des Ports und des Zielpfades für UI-Frames im Hostmode (z.B. für MAIL-Beacons). C <port> <Zielcall> <Digicalls> Setzen von Port und Zielpfad für UI-Frames im Hostmode (z.B. für MAIL-Beacons).
D	Disconnect vom Knoten.
E	wird ignoriert (wegen Kompatibilität zu bestehender Software).
G	empfangene Info-, Monitor- und Status-Daten abholen.
I	anzeigen / setzen des Calls für den aktiven Kanal.
K	Statusmeldungen mit / ohne Uhrzeit.
L	Anzeigen der für den aktiven Kanal vorliegenden Infos.
M	Monitor ein / aus (Parameter s. TRACE).
V	Hostmodeversion anzeigen.
Y0	Kein Connect zum HOST möglich (default).

Yn	Verbindungen zum HOST sind auf n Hostkanälen freigegeben (n = 1 bis 30). Der Connect zum HOST ist mit C möglich **OHNE** den Zusatz eines <**call**>, sowie mit C <CALL>, wobei für <CALL> das globale Host-Call gilt.
@B	Anzeige der für den Hostmode verfügbaren freien Buffer (1/4 der Buffer am Knoten).
@S	Anzeigen Linkstatus (für Kompatibilität mit bestehender Software).
@Y	Anzeigen der maximal verwendeten Host-Kanäle.
@Y<n>	Einstellen der maximal zu verwendenden Host-Kanäle (für den Terminalmodus oder ein Hostmodeprogramm, das nicht mit den 30 Host-Kanälen arbeiten kann / soll. Hier ist zu beachten, dass der aktive Kanal in dem erlaubten neu zu setzenden Bereich liegt, also wenn der aktive Kanal 20 ist, kann man nur auf 20 .. 30 Host-Kanäle einstellen, nicht aber auf 19. Ist einer der in Zukunft nicht mehr zu verwendenden Kanäle noch connected, wird er automatisch disconnected. Anschliessend werden mit dem Befehl <ESC> L nur noch die verwendbaren Kanäle angezeigt und man kann auch nicht auf einen Kanal mit höherer Nummer umschalten (mit <ESC> S).

Mailbeacon-Funktion:

Dieser Teil des Hostmode wurde grundlegend geändert und an die TF-Syntax angenähert. Damit sind auf vielfachen Wunsch eines einzelnen Sysops (hallo Charly) auch mehrfache Mailbaken über verschiedene Ports möglich. Der Hostmode-Befehl @C (zum Einstellen des UI-Pfads) entfällt. Statt dessen wird der Pfad nun auf dem Monitor-Kanal mit dem Hostmode-Befehl C eingestellt entsprechend folgender Syntax:

C <port> <zielcall> <digis>

Der L2-Port <port> kann als Port-Name oder Port-Nummer benannt werden, muss aber angegeben werden. Die optionalen <digis> können entfallen. Der UI-Pfad kann ausserdem mit dem TNN-Befehl „ESC C“ angezeigt bzw. gesetzt werden, bei an sonsten gleicher Syntax wie beim Hostmode-C-Befehl. Für die DPBOX muss ein entsprechender Eintrag in system/beacon.box z.B. so aussehen:

QRG TNN 0 MAIL DB0TNN-7

(########.TNB) FILES

....ein Leckerbissen

Ein ganz alltägliches Problem: immer nachts um 03:00 passiert angeblich Fürchterliches am Knoten, wenn der OM AA0BB erscheint. Man müsste mal am Knoten mitmonitoren. Oder man möchte jede Nacht um 23:00 die Statistik auf Disk schreiben. Oder von 16:00 bis 24:00 sollen andere Parameter auf dem Einstieg genommen werden. Alles Beispiele für den Einsatz von TNN-Batch Files.

Die Sache ist ganz einfach. Es gibt zwei Sorten von Batchfiles, die sich durch den Namen unterscheiden: YYMMDDHH.TNB und YYMMWXHH.TNB. Dabei ist YY das Jahr, MM der Monat, DD der Tag, HH die Stunde und X der Wochentag, an dem das File gestartet werden soll. Und damit nicht für jede Aktion ein extra File gemacht werden muss, darf „#" als Platzhalter für eine Ziffer genommen werden.

Verzeichnis:
Die .TNB Dateien gehören in das Verzeichnis, welches in der TNN179.PAS unter Workpath angegeben wurde.

Beispiele:

TNN179.TNB	Wird NUR beim Neustart des Programmes ausgeführt.
########.TNB	Wird beim Neustart sowie zur vollen Stunde ausgeführt.
######23.TNB	Startet jeden Tag um 23:00 Uhr.
####0100.TNB	Startet immer am 1. jedes Monats um 00:00 Uhr.
99040100.TNB	Startet am 01-APR-99 um 00:00 Uhr.
####W116.TNB	Startet immer Montags um 16:00 Uhr (Sonntag ist Tag 0).
99##W011.TNB	Startet an jedem Sonntag im Jahre 1999 um 11:00 Uhr.
NOW.TNB	Startet nach dem nächsten Minutenübergang. Das File wird nach dem Starten gelöscht.

Der Aufbau der Files ist simpel: es steht alles so drin, wie man es auch an der Console eintippen würde. Mit einer einzigen Ausnahme: wenn ein Batch abläuft, ist automatisch ein Login passiert, und nach Ablauf des Files ist der alte Login Status wieder vorhanden.

Damit nun nicht nur der Bildschirm am Knoten gefüllt wird, sondern auch was für die Nachwelt bleibt, wird mit dem Befehl <ESC> @E1 ein Protokoll-File geöffnet bzw. mit <ESC> @E0 wieder geschlossen (siehe oben).

Um eine monatliche Statistik zu erzeugen, ist folgendes File nötig:

```
####0100.TNB
============
;              Dieses File startet am 1. des Monats um 0:00 Uhr.
esc @e1      ; Umlenken der Ausgabe in ein File.
Stat         ; Auslesen der Statistik.
MH 5000 +    ; Auslesen der MH Liste.
Esc @e0      ; Umlenken ausschalten.
Clear        ; Statistik löschen.
MH -         ; USER-Statistik auf „0" Stellen.
```

Ab dem „;" sind Kommentare erlaubt.

Genauso sind auch DOS Kommandos möglich. Bei der Linux-Version sind SHELL Kommandos zwar zulässig, aber sie werden meist nicht ausgeführt. Der Grund hierfür: SHELL Kommandos werden im Hintergrund ausgeführt als eigener Prozess (der normale Knotenbetrieb läuft also weiter). Wird während eines laufenden SHELL Kommandos ein neuer Befehl eingegeben (das ist der nächste Befehl im TNB-File), wird das laufende Kommando abgebrochen und der neue Befehl wird ignoriert.

(RU)NBATCH <filename.tnb>

Hiermit lassen sich aus einer TNB weitere TNB aufrufen. Damit ist es möglich die TNN179.TNB z. B. in eine PARAM.TNB und eine IPNUMMER.TNB zu unterteilen.

Werden mehrere TNB-Files nacheinander aufgerufen so muss eine gewisse Reihenfolge der Abarbeitung beachtet werden.

Beispiel: Jahreswechsel mit zeitgesteuerten TNB-Files aus denen wiederum andere TNB mit runbatch aufgerufen werden.

```
######00.tnb
    runbatch taggraf.tnb
    runbatch tagstat.tnb

####0100.tnb
    runbatch mon_sich.tnb
    runbatch mon_stat.tnb
    runbatch mon_top.tnb

##010100.tnb
    runbatch jahrstat.tnb
```

Die Abarbeitung wird in folgender Reihenfolge vorgenommen:

```
######00.tnb
####0100.tnb
##010100.tnb
    runbatch taggraf.tnb
    runbatch tagstat.tnb
    runbatch mon_sich.tnb
    runbatch mon_stat.tnb
    runbatch mon_top.tnb
    runbatch jahrstat.tnb
```

EXTERNE PROGRAMME/BEFEHLE für den SYSOP:

(**O**)utput (externer Befehl)

OUTPUT.EXE ist ein eigenständiges Programm, das der Sysop im Bedarfsfall aufrufen kann. Der Aufruf erfolgt wie bei einem internen Befehl. OUTPUT.EXE ist, wenn es sich im PATH SYSEXE befindet, NUR vom SYSOP nach der Privilegierung aufrufbar. Bei Eingabe von Output ohne Argument wird der momentane Zustand der Portbits angezeigt.

```
KS:DB0EAM> IO-Port-Status
0  1  2  3  4  5  6  7
1  1  1  1  1  1  1  1
```

Der Befehl Output ist für Fernsteuerungen vorgesehen. Die 8 Datenleitungen des Druckerports (LPT1) des PC können bitweise ein- und ausgeschaltet werden. Zusätzlich wird bei jeder Änderung die Strobe-Leitung getoggelt (400ms).

Der Ausgangszustand nach einem Programmstart ist „**alles eingeschaltet**". Die Basisadresse des Ports wird über das Rechner-BIOS abgefragt. Soll also ein anderer Port als LPT1 verwendet werden, so muss nur bei 40:08 die passende Adresse stehen.

(**O**)utput <port_bit> <ein_aus>

<port_bit> : zu schaltendes Datenbit (0..7).
<ein_aus> : neuer Pegel (0 oder 1).

(**CWER**) (externer Befehl)

Ist ein kleines Programm, welches die Visitenkarten der User, die sie sich selbst mit /P im Conversmode erzeugt haben, anzeigt.

(**CWER**) a

Erzeugt eine Liste mit allen Einträgen.

(**CWER**) l <anzahl>

Zeigt die l=letzten <anzahl> von Einträgen an (die neuen werden immer hinten angehängt).

(**CWER**) <call>

Kann auch in Verbindung den Platzhaltern „*" und „?" aufgerufen werden. Dabei steht „*" für beliebig viele Zeichen und „?" für genau 1 Zeichen.

(**CPERS**) (externer Befehl)

Ist nun nur noch zur Pflege der „Visitenkarten" der Datei CONVERS.PRS. Zum einen gibt es mitunter mal Einträge, die „zufällig" entstanden sind, und die der Sysop gerne entfernen möchte. Oder sie auch nur durch einen anderen ersetzen. Der einfache Aufruf des Programmes entfernt „Leereinträge" ohne Text sowie „unsinnige" Texte, die zum Beispiel nur aus einem Zeichen bestehen. Bei jedem Aufruf des Programmes wird eine Datei CONVERS.ALT angelegt, um zumindest die letzte Änderung der Datei noch einmal rückgängig machen zu können. Da es durch den Aufruf dieses Programmes zu Speicherplatzproblemen (CONVERS.PRS zu gross) kommen kann, überprüft das Programm zuerst, ob die Datei noch in den verbleibenden Speicher geladen werden kann.

(**CPERS**) /sort

Sortiert die Einträge alphabetisch nach dem Call.

(**CPERS**) + <call> <text>

Ersetzt bzw. fügt das Call mit dem <text> an die Datei CONVERS.PRS an. Es wird hierbei **KEIN** Callcheck durchgeführt !

(**CPERS**) **-** <call>

Entfernt das Call und den Eintrag aus der Datei CONVERS.PRS. Auch hier KEIN Callcheck, damit auch die unsinnigsten Einträge bearbeitet werden können.

(**SETCALL**) (externer Befehl)

Die Felder können mit (SETCALL) ausgefüllt oder geändert werden. SETCALL soll dem SYSOP ermöglichen auf die Datensätze zugreifen zu können.

Zu den Programmen SHOWCALL.EXE, SAVECALL.EXE und SETCALL.EXE ist in dem Dir SAVECALL jeweils eine Datei mit der Endung .SXK. Werden diese Sprachdateien in .SPK umbenannt, so werden sie statt der internen Texte benutzt, was jedoch einen zusätzlichen HD-Zugriff bedeutet. Sollte jemand diese Dateien übersetzen, so würde sich Nord><Link über einen Rücklauf freuen.

(**SETCALL**) <call> <feldkenner> <text>

```
      -> SaveCall Version [010696] de DG3AAH <-
SETCALL CALL /N ............................ 30 Zeichen für den Name
SETCALL CALL /Q ............................ 30 Zeichen des Wohnortes
SETCALL CALL /L ......                       6 Zeichen des World-Locators
SETCALL CALL /D .......                      7 Zeichen für Eingabe des DOK
SETCALL CALL /V .........                    9 Zeichen QRV auf ... Digi
SETCALL CALL /M .................           20 Zeichen Mybbs der Mailbox
SETCALL CALL /T .....  . . . . . .  ..... 40 Zeichen für freien Text
          ( ein * als Text löscht diesen Eintrag )

SETCALL CALL /*    : löscht den Inhalt aller Eintrags für Call.
SETCALL CALL /B    : blockiert CALL für ShowCall.
SETCALL CALL /F    : gibt Call für ShowCall frei.
SETCALL CALL       : gibt Infos über Call aus, auch wenn blockiert.
          ( ein B als Call gibt alle blockierten Calls aus )
```

Auf der Diskette ist jeweils ein Grundbestand von Calls vorhanden (der von DB0EAM). Updates davon gibt es nur gegen Zusendung des eigenen Datensatzes und wenn sich jemand findet der eine SORTIERROUTINE sowie die Erzeugung der entsprechenden CALL.IDX dafür schreibt.

(**SYST**)ext Text (externer Befehl)

Systext bietet nun die Möglichkeit, in die Datei SYSOP.PRO eine Zeile Text einzufügen. Damit können zum Beispiel Parameteränderungen für die „anderen" Sysop dokumentiert werden. Damit ist nun auch eine gewisse Historie möglich. Die Datei SYSOP.PRO wird übersichtlicher, wenn die Einträge anders ausgegeben werden. Auch Neustarts (z.Z. mit b-log.exe dokumentiert) sollen noch in der SYSOP.PRO aufgenommen werden.

(**SH**)owSYS (externer Befehl)

SHOW_SYS zeigt die Einträge in der Datei SYSOP.PRO in einer übersichtlichen Form an.

(**SH**)owSYS /l

SHOW_SYS zeigt die Einträge in der Datei SYSOP.PRO in einer übersichtlichen Form an und löscht dabei alle Privilegierungs-Einträge.

(**MSY**) (externer Befehl)

MSY.EXE in SYSEXE ist ein Hilfsprogramm für den Sysop, das zu MSG.EXE in USEREXE gehört.

(**MSY**) C

Lifetime aller Nachrichten wird um 1 verringert. Dieses kann per Hand geschehen oder mit einem solchen TNB-File.

```
######01.TNB
============
;                                  ;Diese File startet täglich um 1:00 Uhr.
MSY C     ;Lifetime der .MSG-File um einen Tag herabsetzen
;                                  ;und bei Lifetime = 0 entfernen.
```

(**MSY**) D

Listet die Header aller Nachrichten auf.

(**MSY**) D -#x o. +#x

Nachrichten, deren Lifetime (-) kleiner o. (+) grösser x-Tage ist.

(**MSY**) D -*x o. +*x

Nachrichten, die (-) jünger o. (+) älter als x-Tage sind.

(**MSY**) D FROM

Zeigt die Nachrichtenliste sortiert nach Absender Calls an.

Die Sortierung kann auch nach :
BYTE, DATE, TIME, LT, RETURN vorgenommen werden.

(**MSY**) E ALL <call>
Löscht alle Nachrichten von oder an <call>.

(**MSY**) G <gruppe> +/- <call>
Fügt/löscht <call> aus <gruppe>. Zugelassen in einer Gruppe sind maximal 50 Call. Existiert eine Gruppe noch nicht so wird sie unter dem Namen <gruppe> neu angelegt und das <call> eingetragen. Mit löschen des letzten call wird auch die <gruppe> gelöscht.

(**MSY**) L
Zeigt alle Nachrichtenfiles.

(**MSY**) S <call> #x <text>
Schreibt <text> an <call> mit Lifetime #<x>. Hier ist eine Lifetime von maximal 99 Tagen möglich. Default Lifetime sind 7 Tage.

(**MSY**) V
Ausgabe von der Versionsnummer und dem Datum.

(**SYSH**)elp (externer Befehl)
Ist ein externer Befehl für eine neue Hilfefunktion. Funktion wie bei (**H**)elp nur wird auf die OHS.TXT Online-Hilfe-Sysop zugegriffen.

(**TOP**) (externer Befehl)
Ist ein externer Befehl, der die MHEARD.TAB in besser lesbarer Form anzeigt. In der CONFIG.TOP müssen allerdings die Port - Namen eingestellt werden. Da nun auch die Level 3 Verbindungen geführt werden, sollte man alle Port - Namen dort eintragen.

```
# Configfile für TOP.EXE.
# Alle Angaben die nicht gelesen werden sollen müssen eine # am Zeilenanfang haben.
#
# Einstieg
P0 = 70cm_1200
# Einstieg
P1 = 70cm_9600
# Einstieg
P2 = 23cm_9600
# Link DB0GOE
P3 = Goettingen
# ... bis
p16 = ???
# Ende des Files
```

(**STARTCNT**) (externer Befehl)
ein sehr hilfreiches Programm.

Wer häufiger neue Testversionen an seinem Digi einsetzt oder mit den TNN179.TNB Konfigurationsdateien spielt, der sollte unbedingt die Utility „STARTCNT.EXE“ verwenden! Dieses Programm wird zusammen mit den TNN-Tools verteilt und stammt von Andreas, DB7KG. STARTCNT ist ein kleines EXE-Programm, welches einen Zähler bei jedem Aufruf herunterzählt. Solange der Zähler nicht null ist, wird immer ein ERRORLEVEL von 1 zurückgegeben. Ist der Zähler abgelaufen, dann ist der ERRORLEVEL 0. Damit lassen sich dann mittels einer Batch-Datei sehr bequem zwei verschiedene Versionen starten. Damit STARTCNT auch bei Stromausfall funktioniert, wird der Zähler natürlich in einer Datei gespeichert.

Wenn man sich als Sysop in den Knoten eingeloggt hat, dann kann man z.B. mit „DOS STARTCNT 5“ diesen Zähler auf 5 setzen. Nun darf der Knoten 5mal abstürzen / resetten oder was auch immer, bis wieder die alte (stabile) Software gestartet wird.

STARTCNT <anzahl>
stellt die jeweilige Anzahl an Starts ein (Durchläufe der START.BAT). Die Anzahl wird um 1 verringert, wenn die STARTCNT.EXE ohne <anzahl> aufgerufen wird.

Die folgenden Zeilen müssen dann jedoch in die START.BAT aufgenommen werden.

```
STARTCNT
IF ERRORLEVEL==1 GOTO OKGO32
ECHO Starte die TNN_ALT.EXE

REM !!!!!!!!!!!!!!! ALTE SOFT HIER STARTEN (TNN_ALT.EXE) !!!!!!!!!!!!!!!!!!
COPY TNN177.DOS TNN177.TNB
TNN_ALT.exe
REM !!!!!!!!!!!!!!! ALTE SOFT HIER STARTEN (TNN_ALT.EXE) !!!!!!!!!!!!!!!!!!

GOTO ENDE
:OKGO32

ECHO Starte die TNN_NEU.EXE

REM !!!!!!!!!!!!!!! HIER WIRD DIE NEUE SOFTWARE GESTARTET !!!!!!!!!!!!!!!!
COPY TNN179.DPI TNN179.TNB
TNN_NEU.exe
REM !!!!!!!!!!!!!!! HIER WIRD DIE NEUE SOFTWARE GESTARTET !!!!!!!!!!!!!!!!
```

HILFSPROGRAMME für den SYSOP zu Hause:

(MAKEDAT.EXE)

Wer Sat-Daten auf dem Digipeater anbietet, muss diese auch pflegen! (Andernfalls solltet Ihr lieber die SAT.EXE in USEREXE löschen.)

DB1HZ hat nun statt der bisher 22 Satelliten 40 Berechnungen ermöglicht in der neuen SAT.EXE. Ausserdem stehen die Daten nun nicht mehr in Einzelfiles, sondern es ist nun nur noch das File SAT.DAT im Verzeichnis TNN zu pflegen. Das Verzeichnis DATEN kann somit entfallen. Ferner wird nun ein evtl. vorhandener CO-Prozessor unterstützend bei der Berechnung hinzugezogen, was die Berechnungszeiten positiv beeinflusst. Die bisher vorhandene Umgebungsvariable SET SATQTH=<Locator> oder <Koordinaten> ist entfallen. Es wird nun die bereits vorhandene Variable QTH mitbenutzt.

Das Programm MAKEDAT dient der Umwandlung der via Packet-Radio verbreiteten Keplerdaten im AMSAT- oder 2LINE-NASA-Format in die von SAT Master erwarteten Datensätze. Nähere Informationen zu MAKEDAT bekommst Du, indem Du das Programm ohne Parameter aufrufst.

Zum Ablauf:
Man öffnet auf seinem Rechner zu Hause ein SAVEFILE z.B. mit dem Namen KEPLER.TXT. Dann kann man mit <read kepler> in der Box alles, was an neuen Daten Texten und Programmen dort vorhanden ist, einlesen. Nach dem Schließen des SAVEFILES wird nun MAKEDAT /l KEPLER.TXT aufgerufen. MAKEDAT aktualisiert nun nur die Einträge der SAT, die in der Datei SAT.FLT eingetragen sind, in der Datei SAT.DAT. Anschließend die Datei SAT.DAT <load sat.dat> wie ein Programm in den Knoten laden .... fertig ist es!

(TNNSET.EXE)

Stammt von IK7NXQ und soll das Erstellen von EPROMs für der Tokenring erleichtern. Einfach mal aufrufen... es erklärt sich selbst.

Befehle für alle User

(**6**)PACK

Mit dem Befehl „6pack“ kann eine Statistik und die aktuelle Zuordnung von TNC zu den Ports abgerufen werden, sofern 6pack verwendet wird. Eine Veränderung von Parametern ist nicht möglich. Die Behandlung von möglichen Fehlern des Rings wird komplett ohne Eingriffsmöglichkeit durchgeführt. Falls ein TNC fehlerhaft sein sollte, so ist dies an den Checksum- und Reset-Zaehlern zu erkennen.

(AX)IPR

Gibt den derzeitigen Status der AXIPR-Einträge aus.

(**BE**)ACON

Zeigt die eingestellten Bakenparameter und Bakentexte an.

(**C**)ONNECT

Connect zum HOST-Interface wenn dieses mit <**ESC**> **Y 1** freigegeben ist. Ist Y auf 0 gesetzt, so wird am Terminal ein CONNECT REQUEST fm <call-ssid> <datum uhrzeit> angezeigt.

(**C**)ONNECT <call>

Mit dem Connect-Befehl wird eine Verbindung zu einem anderen Knoten oder einem anderen Benutzer aufgebaut. Die Eingabe

CONNECT DB0FD

oder auch abgekürzt und klein geschrieben

c db0fd

bewirkt, dass erst in der NODES-Liste (Nodes und Flexnet-Destinations) nach dem Call DB0FD abgesucht wird. Handelt es sich um ein Call, das in der NODES-Liste eingetragen ist, so wird die Meldung:

Interlink setup (via call) ...

ausgegeben und in Klammern wird angezeigt zu welchem direkten Nachbarn der Connect gesendet wird.

Wird dieses Call lokal geführt, so wird ein:

Link setup (Portname)...

Ist kein Eintrag vorhanden, so wird die MH-Liste durchsucht. Ist dort ein Eintrag vorhanden, so wird der Connect auf dem entsprechenden Port ausgesendet. Ist auch dort kein Eintrag vorhanden, so wird eine Fehlermeldung ausgegeben. An den Meldungen ist bereits erkennbar, und somit auch durch Router auswertbar, ob eine Verbindung zu einem TheNet - bzw. TheNetNode-Knoten überhaupt aufgebaut werden kann.

Ist der zu verwendende Port nicht eingeschaltet, so wird angezeigt:

Port not in use

Wurde eine Verbindung zu einem TheNet - oder TheNetNode-Knoten aufgebaut, wird die Meldung mit dem ALIAS ausgegeben, sofern vorhanden:

BS:DB0FC> Connected to H:DB0FD ansonsten nur :
BS:DB0FC> Connected to DB0FD

Ein Benutzer kann auch über Digipeater connected werden. Die Eingabe ist dann:

CONNECT DB3AN via DB0FD

Das Wort via kann entfallen oder beliebig abgekürzt werden. Es sind auch bis zu 8 Digipeater möglich. Die Rufzeichen sind dann durch Leerzeichen zu trennen. Diese Möglichkeit funktioniert aber nur, wenn bei dem entsprechenden Digipeater auch L2-Digipeating eingeschaltet ist.

Mögliche Fehlermeldungen sind:

Failure with ...: Der gesuchte Partner hat sich nicht gemeldet.

Busy from ...: Der Partner hat einen Verbindungsaufbau abgelehnt.

Node not avilabile: Das Ziel ist zwar bekannt aber es gibt zurzeit keine Verbindung dahin.

Die folgenden Meldungen zeigen an, dass der Knoten, an den der Connect-Befehl geschickt wurde, überlastet ist und daher die Verbindung nicht aufgebaut werden konnte:

Node busy
Link table full
Circuit table full

Ein Connect-Befehl kann jederzeit durch eine beliebige Eingabe abgebrochen werden, eine Leerzeile reicht aus.

Im Connect-Befehl kann auch der Port-Name mit angegeben werden. Gültige Port-Namen erhält man mit dem (PO)rt-Befehl. Auf dem Multibaud Zugang von DB0EAM sind unter anderem die Port-Namen „70cm_1200", „70cm_9600" und „23cm_9600" eingetragen. Wird im Connect-Befehl der Port-Name mit angegeben, so wird der Connect auf diesem Port ausgesendet ohne Beachtung eines Eintrages in der MH-Liste. Format: C DB6XYZ 23cm_9600.

Ein Connect zum Digicall mit SSID ist auch in einer verkürzten Schreibweise wie bei Flexnet möglich. Statt „C DB0WHV-12" kann auch „C -12" eingegeben werden.

Wird ein Connect ohne Argument eingegeben, so wird versucht, eine Verbindung zum HOST-Terminal herzustellen. Das ist jedoch nur möglich, wenn der Sysop mit dem Befehl <ESC> y1 Verbindungen zum HOST-Terminal freigegeben hat und davon eine nicht belegt ist, ansonsten gibt es ein:

Busy from <call>.

LOOP DETECTED ist eine Warnung, dass der gewählte Verbindungsaufbau eine Schleife (Loop) bildet. Es wird also eine Interlink-Verbindung in beiden Richtungen über denselben Nachbarknoten und somit unnötig belegt. Im Gegensatz zu anderer Knotensoftware erfolgt jedoch nur eine WARNUNG, die Verbindung kann trotzdem aufgebaut werden.

WARNING: Loop detected (HELP LOOP)

Letzteres ist aber nicht immer sinnvoll und man behindert sich selbst. Bitte mit 2 mal Quit zum vorletzten Knoten zurückgehen und dann die Verbindung direkt aufbauen... Es kann aber auch vorkommen, dass der Router von TNN einen neuen Weg zum Zielknoten gefunden hat, der zu der Schleife führt — dann kann man die Warnung auch einfach ignorieren.

(**CONV**)ers
Schaltet um in den Convers-Modus.

(**CONV**)ers **<n>**
Schaltet um in den Convers-Modus auf Kanal <n>. Es wird der folgende Connect- und Begrüssungstext gesendet:

```
Eingabe: conv 170

conversd @ db0eam PingPong-Release 3.14c (TNN) - Type /HELP for help.
*** (10:34) You are now talking to channel 170. There are 2 users.
*** Personal text and data set.
Hello, Henning, Vellmar JO41RI, Sysop DB0EAM
***
```

(CONV)ers - intern -

Im Convers-Modus stehen folgende Kommandos zur Verfügung (die Kommandos können durch die Verwendung der Grossbuchstaben abgekürzt werden):

/Away [Text]	Markiert Dich als abwesend.
/ALl [Text]	Text an alle User Deines Kanals.
/Beep	Beep-Modus an/aus.
/Channel [n]	Verbindet dich zusätzlich mit dem Kanal n.
/CHARacter	Setzt verschiedene Zeichenwandlungen.
/Destinations	Listet erreichbare ping-pong Hosts.
/Help [Kommando]	Gibt Hilfe-Informationen.
/EXClusiv User Text	Sendet Text an alle auf Deinem Kanal ausser User.
/Filter [Calls]	Setzt Calls, deren Texte gefiltert werden sollen.
/Invite User	Lädt User auf Deinen Kanal ein.
/Links [args]	Listet oder setzt (Sysop) conversd-Partner.
/LISt	Listet alle Kanäle und ihre Themen.
/LEave [Kanal]	Verlässt Kanal oder derzeitigen Kanal.
/Msg User Text	Sendet Text an User oder.
/Msg #Kanal Text	Sendet Text an den angegebenen Kanal.
/ME Text	Sendet einen Aktionstext.
/MOde [Kanal] Optionen	Setzt Kanaloptionen.
/NICKname <Name> oder "@"	setzt den Namen, der @ löscht ihn
/NONickname	löscht den Namen (wie "/nick @")
/NOtify [Calls]	Setzt Calls, deren Erscheinen gemeldet werden soll.
/Personal [Text]	Setzt persönliche Beschreibung.
/Restart	Sysop kann damit den locked Zustand aufheben.
/PRompt abcd	Prompt setzen (a=Query; b=Normal; c=Ctrl-g; d=Ctrl-h).
/Quit	Convers verlassen.
/QUEry [User]	Startet/beendet private Konversation.
/Topic [#Kanal] [Text]	Setzt Thema des Kanals. Thema=@ entfernt Thema.
/UPtime	Wie lange läuft dieses conversd überhaupt schon ?
/Verbose	Laber-Modus ein/aus.
/VERSion	Zeigt Info zu dieser Version.
/Who [*; A; L; Q]	Zeigt User und Ihre Kanäle (*=eigner; A=abwesend; L=lang; Q=kurz).
/WIdth [Wert]	Setzt/zeigt Zeilenbreite.

Die Erklärungen im Einzelnen:

/ALL

Wenn Du im /query Modus bist, wird Text mit vorangestelltem /all behandelt, als würdest Du ohne /query arbeiten.

/AWAY <text>

/away setzt den Abwesendheitstext, den die anderen lesen können. Beim Aufruf ohne Argument wird der Text gelöscht und man gilt wieder als anwesend.

/BEEP

Hiermit wird das Klingelzeichen (CTRL-G), welches vor jeder Mitteilung gesendet werden kann, ein- oder ausgeschaltet. Dieses Kommando ist eigentlich eine Untermenge des /prompt Befehls, siehe dort.

/CHAN

Verbindet Dich zusätzlich mit dem gewünschten Kanal. Im Gegensatz zu älteren Conversd-Implementationen verbleibt man auch noch im vorherigen Kanal, denn es wird eine Mehrfach-Kanal-Verbindung unterstützt. Um einen Kanal zu verlassen, musst Du „/leave" verwenden. Ohne Angabe eines Kanals wird die Info ausgegeben, auf welchen Kanälen Du Dich befindest.

/CHAR

Mit diesem Befehl kannst Du dem Convers mitteilen, welche Zeichensatzwandlung Du haben möchtest. Die Syntax ist: /char [In-Typ [Out-Typ]] wenn Du z.B. mit einem Atari ST arbeitest, könntest Du „/char pc atari" eingeben. Wenn Du einen PC benutzt und Umlaute im TeX-Stil schreiben möchtest, gebe „/char tex pc" ein. „/char ?" listet die möglichen Einstellungen.

Die Einstellung wird mit „/pers" gespeichert (siehe dort).

Der Dank für diese nette Funktion geht an Tommy, <dl9sau@thynet.sub.org> (Internet mail) <dl9sau@db0sao.ampr.org> (AmPR-Net mail). Vorschläge sollten an ihn weitergeleitet werden.

Mögliche Einstellungen mit /char:
iso-8859-1, ansi, 8bit
dumb, ascii, none, us
tex
ibm7bit, 7bit, commodore, c64, digicom
roman8
ibmpc, pc, at, xt
atari,
binary, image

/DEST

Alle Pingpong-Hosts, die miteinander verbunden sind, werden aufgelistet. Die Zahlen zeigen die Antwortzeiten in Sekunden.

/DEST <call>

Fragt den Weg zum <call> ab und zeigt dabei die Übertragungszeiten ab.

/EXCL

Dieses Kommando ist das Gegenteil des /MSG-Befehls. Hiermit sendest Du Text an alle User dieses Kanals ausser dem einen als ersten Parameter angegebenen. Da der Text intern als privater Text an die anderen verschickt wird, werden die Links etwas mehr belastet

/FILT

Wenn Du die Texte bestimmter User nicht lesen möchtest, so kannst Du sie hiermit in eine Liste einfügen. Alle Texte werden dann ausgefiltert, bei persönlichen Texten („/msg") wird eine Rückmeldung an den Absender geschickt. Das Setzen/Löschen geschieht wie bei „/notify", also z.B. „/filter + dc1ik - db4ut" setzt dc1ik und löscht db4ut aus der Liste.

/HELP

Das Hilfekommando kann von zusätzlichen Parametern gefolgt sein. Der Schrägstrich darf hier nicht vor dem fraglichen Kommando stehen, z.B.: /Help Invi. ALLE Hilfstexte können auch ausserhalb des Conversmode mit „Help conversd" als komplette Übersicht gelesen werden.

/INVI

Es wird eine Einladung zum genannten User geschickt. Diese Einladung wird durch das gesamte Netz geleitet. Wenn derjenige auf einem anderen Kanal ist und Dein Kanal als privat eingerichtet ist, so kann er auf Deinen Privatkanal wechseln. Wenn er im Befehlsinterpreter eines Knotens ist, so empfängt er die Einladung, er kann dann aber nicht direkt auf Deinen Privatkanal kommen, weshalb er nochmals einzuladen ist.

/JOIN

Verbindet Dich zusätzlich mit dem gewünschten Kanal. Im Gegensatz zu älteren Conversd-Implementationen, verbleibt man auch noch im vorherigen Kanal, denn es wird eine Mehrfach-Kanal-Verbindung unterstützt. Um einen Kanal zu verlassen, musst Du „/leave" verwenden. Ohne Angabe eines Kanals werden Infos zu den von Dir benutzten Kanälen ausgegeben.

/LEAV

Mit diesem Befehl kannst Du entweder den derzeitigen oder den angegebenen Kanal verlassen. Wenn dieser der letzte ist, so wird conversd verlassen.

/LINK

Der momentane Linkstatus wird angezeigt. Dies sind normalerweise Hostname, Linkstatus, Laufzeiten, Versionskodes und Statuszeit, gefolgt von der Zeit des nächsten Connectversuches und Anzahl der Versuche (auf Disconnecteten oder im Aufbau befindlichen Links), bei bestehender Verbindung werden die Queue-Längen und Bytestatistiken angezeigt.

Wenn Du Sysop bist, kannst Du Verbindungen setzen oder löschen. Es wird dann auch noch zusätzlich in Klammern der Verbindungsweg angezeigt. Syntax: /l [[-] Host [Port [via]]]

/LIST

Alle Kanäle, ihre Themen, Optionen und User werden angezeigt. Die Klammerwerte bedeuten:
(@) = Channel Oparator,
(G) = Mit AWAY abwesend gemeldet,
(!) = User ist im Sysopmodus.

/ME

Dieser Befehl dient dazu, den Usern auf Deinem Kanal eine Tätigkeit anzuzeigen. Wenn du z.B. „/me gähnt" eingibst, bekommen alle User dieses Kanals folgendes angezeigt:

*** dc6iq gähnt

/MODE

Das Modekommando ist eines der Kompliziertesten. Es wird wie folgt benutzt:

„/mo [<Kanal>] <+ ; -> <t ; i ; s ; m ; p ; l ; o <User>>".
Die Optionen bedeuten folgendes:
t = Das Thema des Kanals lässt sich NUR von Kanal-Sysops ändern.
i = Der Kanal wird Usern anderer Kanäle verheimlicht.
s = Der Kanal ist geheim, die Kanalnummer wird nicht mehr angezeigt.
m = Der Kanal ist moderiert, nur Kanal-Sysops dürfen schreiben.
p = Der Kanal ist privat, man benötigt eine Einladung zum Einloggen.
l = Der Kanal ist lokal, Texte werden nicht weiter verteilt.
o <User> = Macht <User> zum Kanal-Sysop (kein - möglich).

Das Plus setzt eine Option, der Strich löscht sie. Es sind Kombinationen erlaubt, so würde z.B. „/mode 69 -s+todc6iq" folgendes bewirken: Kanal 69 ist nicht mehr geheim, aber die Themen dürfen nur vom Kanal-Sysop gesetzt werden. Zusätzlich wird dc6iq ein Kanal-Sysop. Allerdings können auf Kanal 0 keine Modes gesetzt werden.

/MSG

Sendet einen Text an einen speziellen User oder an einen verbundenem Kanal. Wenn der Text an einen Kanal gehen soll, so muss man folgendes eingeben:

„/msg #<Kanal> <text>".

Wenn das Ziel ein User ist, so kann er den Text an den zusätzlichen Sternchen erkennen. Z.B. wenn dc6iq eine Nachricht an dc1ik mit „/m dc1ik Das ist ein Test" schickt, so erhält dc1ik folgendes: „<*dc6iq*>: Das ist ein Test".

/NICKname

Nickname-Unterstützung für den Convers.

/NICKname <Name> oder "@" setzt den Namen, der @ löscht ihn

Es ist die Angabe eines Namens möglich, der zusätzlich vor dem Rufzeichen angezeigt wird. Die Eingabe wird an andere Convers-Hosts weitergeleitet, sofern sie die Nickname-Fähigkeit in ihren Feature-Flags angezeigt haben. Die Umsetzung erfolgte in Anlehnung an die Implementierung im tpp-convers 1.14, jedoch erfolgt die Anzeige des Nicknames nicht so häufig wie im Original.

/NONickname

Löscht den Nickname wieder.

/NOTI

Du wirst informiert, wenn eine bestimmte Person in der Personenliste im Convers erscheint. Z.B. fügt „/notify + dc1ik" dc1ik in die Liste ein, „/notify - db4ut" entfernt db4ut aus der Liste. Das Einfügen/Löschen mehrerer Calls in einem Kommando ist möglich, z.B. bewirkt „/notify + dc1ik db4ut - dc6iq dh2paf +dg3kcr", dass dc1ik, db4ut und dg3kcr eingefügt werden sowie dc6iq und dh2paf entfernt werden. Das Entfernen von Calls, die nicht in der Liste stehen, wird ignoriert.

/PERS

Es kann eine kurze Beschreibung zu Deiner Person gesetzt werden, den die anderen User mit „/who" sehen können. Z.B: „/pers Fred, Buechig, JN49fb".

Ohne Text wird die Beschreibung gelöscht. Diese Implementation merkt sich bis zu 118 Zeichen der Beschreibung und setzt diese dann automatisch beim Einloggen (die „/char" und „/width" Einstellungen werden dann auch gespeichert und beim Einloggen gesetzt).

/RESTART

Ermöglicht dem Sysop einen „Disc./locked" Zustand auch von Hand aufzuheben.

/PROM

Das Prompt-Kommando nimmt vier Argumente in einer zusammenhängenden Zeichenkette. „/prompt abcd" setzt
a = Als „/query"-Prompt.
b = Für den normalen Prompt.
d = ist ein Zeichen, um den Prompt zu löschen (normalerweise Backspace
c = Ist ein Zeichen, welches vor jedem Text, den Du empfängst, gesendet wird (normalerweise also CTRL-G).

/QUER

Der angegebene User ist in Zukunft der einzige Empfänger für alle Texte, die Du eingibst. Diese werden dann als private Texte an den User geschickt, wie bei „/m". Zum Ausschalten ohne Argument aufrufen, danach geht alles wieder wie gewohnt an den Kanal. Sozusagen ein Privatmodus.

/QUIT

Wenn Du das eingibst, verlässt Du diesen wunderbaren Ping-Pong-Convers. Wir hoffen, es gefiel Dir.

/TOPI

Hiermit kann für den Kanal ein Thema gesetzt werden. Die anderen User können dieses sehen, wenn sie „/who quick" oder „/list" eingeben. Wenn keine Kanalnummer angegeben wird, so wird das Thema des aktiven Kanals gesetzt. Wird eine Nummer angegeben, so muss Du auch auf diesem Kanal eingeloggt sein. Um das Thema zu löschen, ist als Thema ein „@" einzusetzen.

/UPTI

Dieser Befehl zeigt an, wie lange conversd schon aktiv ist.

/VERB

Schaltet die Laber-Option ein/aus. Du bekommst dann viele Informationen über Aktionen der User (Einloggen/Ausloggen/Texte setzen/...), auch wenn diese nicht auf Deinem Kanal sind.

/VERS

Zeigt etwas Text zu dieser Version (in Englisch).

/WHO

Dieser Befehl hat 4 Optionen.

a = Zeigt alle User und ihre Abwesendheitstexte, wenn gesetzt.
l = Generiert eine LANGE Liste mit Personenbeschreibung, Abwesendheitstexte und Queue-Informationen.
q = Gibt eine kurze Auflistung aus
* = Zeigt alle User Deines Kanals.

Wenn Du Informationen über bestimmte User brauchst, kannst Du die „/who u Userliste" Variante benutzen.

/WIDT

Macht conversd Deine Bildschirmbreite (Zeichen/Zeile) bekannt. Die Meldungen der anderen werden dann auf diese Breite gebracht. Voreingestellt ist 80. Die Einstellung bei „/pers" gespeichert (siehe dort).

(CONV)ers C**(stat)**

Zeigt die bestehenden Verbindungen, Laufzeiten, Datenmengen usw. an.

```
Host      State       Quality Revision  Since NextTry Tries Queue  RX    TX
db0goe    Connected    1s/1s  pp-3.06t  17:55                0  136K  267K
(DB0GOE on port 6)
db0ii     Connected   10s/7s  pp-3.06t   7:07                0    1K   48K
(DB0II on port 9 via DB0BID)
db0kh     Connected    1s/1s  pp-3.06t   6:13                0   17K   65K
(DB0KH on port 11)
db0gso    Disc./locked   ---             9:22  10:22         0
(DB0GSO on port 9 via DB0BID)
 1 loops detected.
dg9fu     Disconnected   ---             18:47
 (trusted host)
DB0KV     (dw-0.84k)  2m   DB0NOE  (dw-0.82b)  5m   db0acc  (pp-3.12f) 25s
db0ais    (pp-3.12f)  3m   db0ber  (pp-3.06t)  3m   db0bhv  (pp-2.93t)  1m
db0bid    (pp-3.06t)  9s   db0bro  (pp-3.06t) 12s   db0cl   (pp-2.93t)  2m
```

Wird nun ein Convers-LOOP bemerkt, so wird der ermittelte Link für eine Stunde aus dem Verkehr gezogen. Trotzdem sollten LOOP vermieden werden. Der Sysop kann mit /RESTART den „Disc./locked" Zustand auch von Hand aufheben. Die Anzeige der Einträge wie (DB0GOE on port 6) sind erst nach der SYSOP-Priviligierung sichtbar.

(**CONV**)ers **O**(nline) [q ; l ; a]

Zeigt die Benutzer das Convers an. Zusätzliche Optionen wie [q ; l ; a], sind möglich.

(**!**) <TheNetNode-Befehl>:

Mit einen vorangestelltem „!" ist es möglich, die TheNetNode-Befehle auch vom Convers aus aufzurufen.

(**CQ**) <text>

Durch Eingabe von CQ kann über jeden TheNetNode-Digipeater ein CQ-Ruf gestartet werden. CQ Text... (Text bis zu 75 Zeichen). TheNetNode kann mehrere CQ-Rufe gleichzeitig verwalten, jedoch nur einen je Uplink bzw. Circuit.

Wie starte ich einem CQ-Anruf?

Angenommen, DB2OS in Hannover möchte beim Knoten in Kassel einen allgemeinen CQ-Ruf absetzen. Zunächst connected er KS:DB0EAM und gibt dann dort den CQ-Befehl ein.

cq cq de DB2OS HANNOVER JO42VG/EM60G VIA KS -- PSE CONNECT DB2OS via DB0EAM

TNN antwortet mit „KS:DB0EAM> Waiting ...“ und ab sofort kann DB2OS connected werden. Ausserdem sendet TNN auf allen eingeschalteten Ports, auf denen eine MHEARD-Liste geführt wird, ein UI-Frame mit folgendem Inhalt:

fm DB2OS to CQ via DB0EAM* ctl UI^ pid F0
cq de DB2OS HANNOVER JO42VG/EM60G VIA KS -- PSE CONNECT DB2OS via DB0EAM

WICHTIG: Durch einen nachfolgenden Befehl oder ein RETURN wird der CQ Zustand abgebrochen!

VARIANTE A:

OM Karl, DK7AL, ist zur gleichen Zeit mit dem Knoten KS:DB0EAM connected und sieht nun bei der Eingabe des USER-Befehls folgende Liste:

```
KS:DB0EAM>  TheNetNode (GO32), 1.78 (12401)
 Circuit(H:DB0FD DB2OS)          <..>  CQ(DB2OS)
Uplink (DK7AL)
```

„<..> CQ (DB2OS)” zeigt nun an, dass DB2OS (vom Knoten H:DB0FD kommend) eine Verbindung in den Raum Kassel sucht und den CQ-Befehl eingegeben hat. DK7AL muss an dieser Stelle nur „C DB2OS” eingeben und ist SOFORT mit DB2OS verbunden!!! Das mühsame Zurückverfolgen des Verbindungsweges entfällt komplett bzw. ist nicht erforderlich.

VARIANTE B:

OM Wolfgang, DB3AN, monitort die Frequenz und sieht plötzlich folgendes Paket auf dem Bildschirm (NORD><LINK Firmware):

```
fm DB2OS to CQ via DB0EAM ctl UI^
CQ de DB2OS HANNOVER JO42VG/EM60G via KS -- PSE CONNECT DB2OS via DB0EAM
```

```
!        !
!        +----------------------CQ CQ CQ...ggf. mit Text...
+-----------------------------Absender, OM Peter, DB2OS
```

Dieses Paket wurde von KS:DB0EAM unmittelbar nach dem Empfang des CQ-Befehl (mit Text dahinter) als UI-Paket abgestrahlt.

Um diesen CQ-Ruf zu beantworten, muss Wolfgang nicht erst den Knoten KS connecten, sondern er gibt seinem eigenen TNC den Befehl zum Aufbau einer Verbindung mit DB2OS (Connect DB2OS via DB0EAM), als ob DB2OS direkt in der Nachbarschaft wohnt.

Nachdem KS:DB0EAM das SABM-Paket von DB3AN empfangen hat, ist SOFORT die Verbindung mit DB2OS hergestellt, und bei DB2OS erscheint die Meldung „KS:DB0EAM> Connected to DB3AN”.

Wie man sieht, kann man also auf der Benutzerebene des Knotens, oder direkt, den Verbindungsaufbau nach dem „Sichten” des CQ-Ruf einleiten.

(**D**)EST

Zeigt die Einträge der NODES-Liste in der bei den RMNC üblichen Weise an.

```
Destinations (5):
DB0BID  0-0    DB0EAM  3-3    DB0EAM  4-4    DB0NHM  0-0    DB0NHM  4-4
  ^1.)   ^2.)
```

1.) Erreichbare Ziele (Destinations),
2.) SSID-Bereich des Calls.

(**D**)EST <*>

Liste wie oben jedoch zusätzlich mit Laufzeiten.

(**D**)EST <call*>

Hierbei muss das Call nicht vollständig sein. Die Eingabe von (**D**)est HB9* zur Anzeige aller HB9.. Destinations. Sie werden mit Call, SSID-Bereich, Laufzeit und Port, auf dem sie erreichbar sind, angezeigt.

(**D**)EST <call>

Zeigt die Liste wie bei (N)odes <call> an.... Eben nur für die FlexNet Liebhaber.

(**D**)EST < <Nachbar Call>

Zeigt die Nodes / Destinations an, die von diesem Nachbarn mit der besten Laufzeit gemeldet wurden.

(**DX**)CLUSTER

Wurde ein lokales Cluster eingetragen, so reicht die Eingabe von <dx> zum Connect dieses Clusters. Sinn dieses Befehles ist es, nicht immer auf einem Knoten nach dem nächstgelegenen Cluster suchen zu müssen.

(**G**)RAPH

Graphische Ausgabe einiger statistischer Werte. Zwei Typen, Systemgraph und Portgraph stehen zur Verfügung. Es sind Darstellungen der letzten Stunde, der vergangenen 24 Stunden und der vergangenen 7 Tage möglich. Das „A“ in der Zeitachse zeigt an, welcher Spalte zurzeit aktualisiert wird. Ein „N“ zeigt den letzten Neustart an. Die Kommandos gliedern sich wie folgt:

```
SYSTEMGRAPH:
(G)raph   (H)our   (B)aud   (”#*+”) eig. Darstellung  (+) expand
          (D)ay    (C)ircuits
          (W)eek   (F)ree buffers
                   (L)2-Links
                   (N)odes
                   (R)ounds
                   (*)Für die Ausgabe aller Statistiken.
```

- Der erste Parameter (H, D, W) gibt die zeitliche Darstellungsform an, wird dieser Parameter vergessen, so wird die Stundendarstellung verwendet.

- Der zweite Parameter (B, C, F, L, N, R, *) spezifiziert die Datenart, die der User angezeigt bekommen möchte. Hierbei ist klar, das in Analogie zu anderen TNN Befehlen die Möglichkeit besteht, mit Hilfe von „*“, alle Datenarten anzeigen zulassen. Wird der zweite Parameter weggelassen oder ist er falsch, so erfolgt ein Hinweis auf unzureichende Spezifizierung.

- Mit dem dritten optionalen Parameter („#*+”) können die Darstellungszeichen des Graphen verändert werden. Bei der Eingabe ist darauf zu achten, dass die drei Darstellungszeichen durch zwei Anführungszeichen eingeschlossen sind. Bei Fehlen der Angabe werden die Maximalwerte durch „+“ , Mittelwerte durch „*“ und Minimalwerte durch „#“ gekennzeichnet.

- Der vierte Parameter (+) ist auch optional, und gibt an, ob die Daten „expandiert“ werden sollen. Das heisst, der Zeilenraum für die Darstellung der Werte zwischen Null und Minimum, wird zu Gunsten der Werte zwischen Minimum und Maximum weggelassen. Dadurch wird die Darstellung zwischen Minimum und Maximum präziser.

```
PORTGRAPH:
(G)raph <PortNr> (H)our (I)nfo frames    (”TR”) eigene Darstellung
                 (D)ay  (R)eject frames
                 (W)eek (F)rmr frames
                        (S)abm frames
                    dis(C) frames
                      d(M) frames
                       (*) All
```

- Der erste Parameter (<PortNr>) erwartet eine Portnummerangabe. Dadurch wird zwischen Systemgraph und Portgraph unterschieden.

- Der zweite Parameter (H, D, W) gibt die zeitliche Darstellungsform an, wird dieser Parameter vergessen, so wird die Stundendarstellung verwendet.

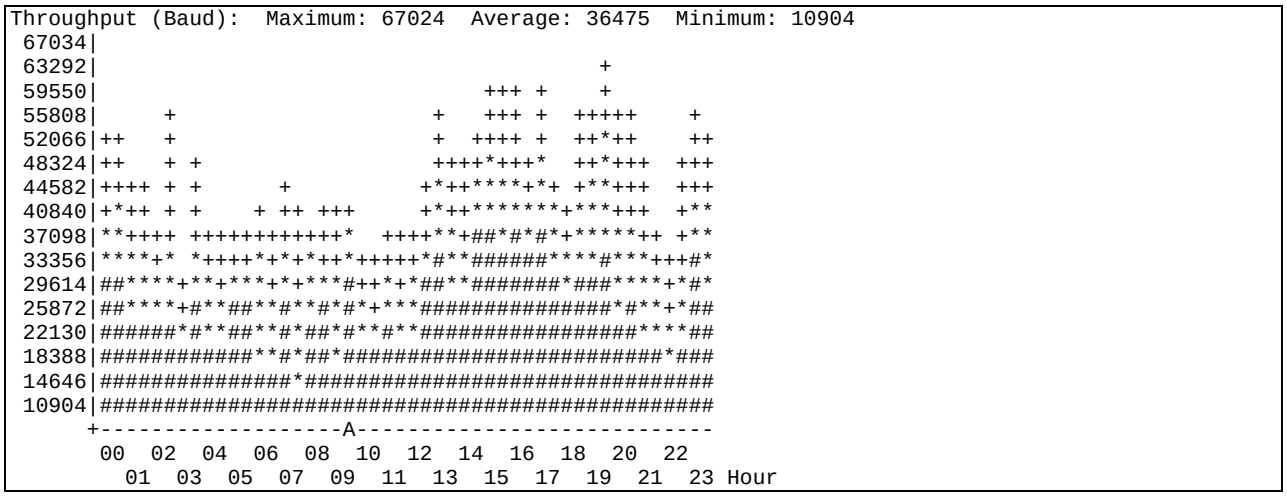
- Der dritte Parameter (I, R, F, S, C, M, *) spezifiziert die Datenart, die der User angezeigt bekommen möchte. Hierbei ist klar, das in Analogie zu anderen TNN Befehlen die Möglichkeit besteht, mit Hilfe von „*“, alle Datenarten anzeigen zulassen. Wird der dritte Parameter weggelassen oder ist er falsch, so erfolgt ein Hinweis auf unzureichende Spezifizierung.

- Mit dem vierten optionalen Parameter („TR”) können eigene Darstellungszeichen angegeben werden. Es ist darauf zu achten das die beiden Darstellungszeichen durch zwei Anführungszeichen eingeschlossen werden. Bei Fehlen der Angabe wird die Aussendung mit „T” und der Empfang mit „R” gekennzeichnet.

Eingabebeispiele:

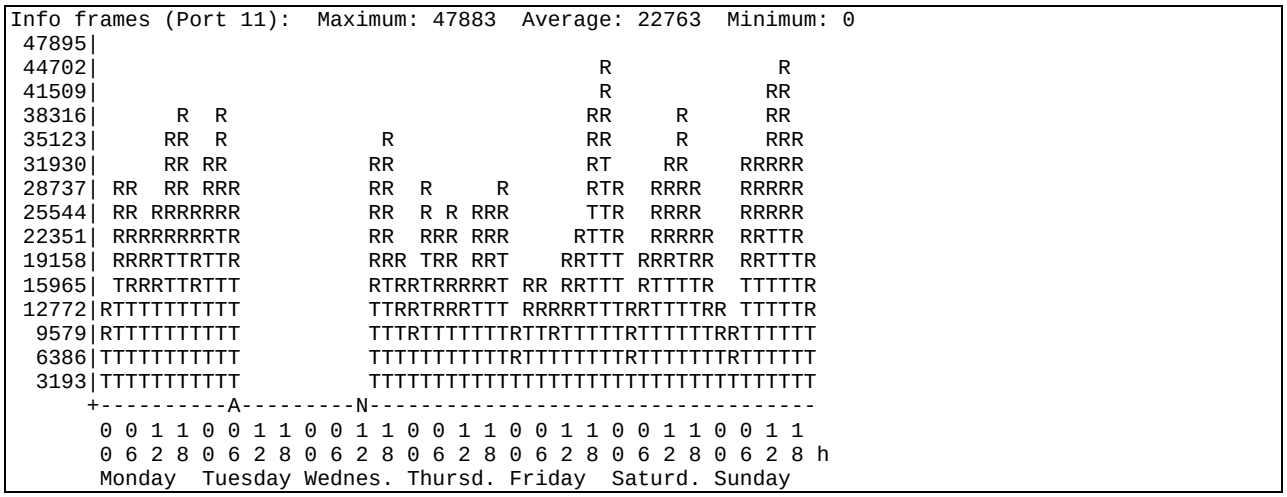
G	interner Hilfetext,
G H *	alle System-Graphen der letzten Stunde,
G D * +	alle expandierten System-Graphen des letzten Tages,
G D * „:-*“ +	wie zuvor, jedoch eigene Darstellungsweise (Die Anführungszeichen müssen mit eingegeben werden),
G w *	Wochenüberblick aller System-Graphen,
G d * +	Tagesüberblick aller System-Graphen in expandierter alternativen Form.
G 8 D * +	Tagesüberblick aller Port-Graphen für Port 8.

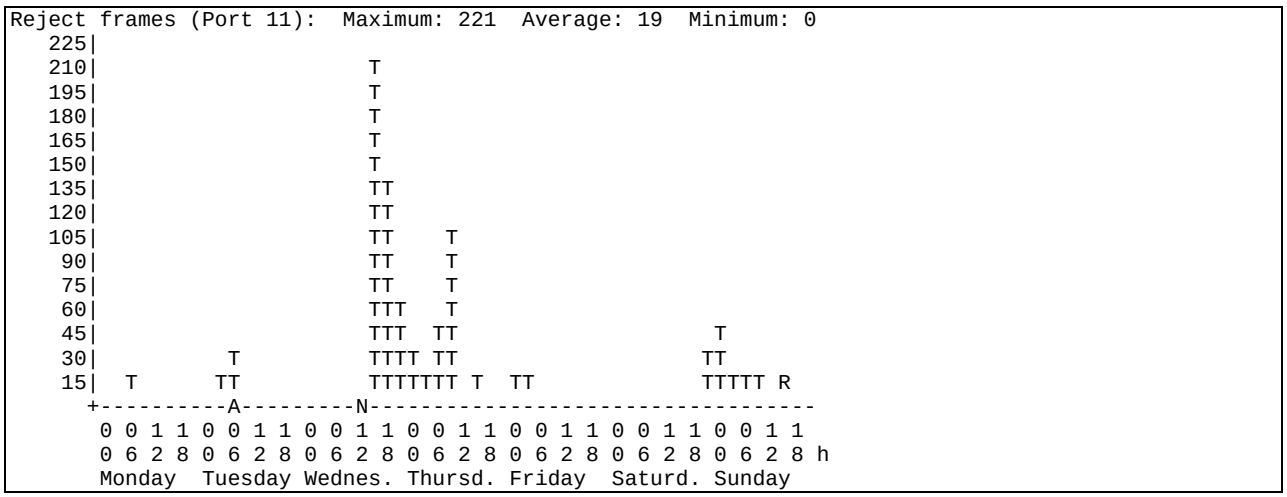
Beispiel: (**G**)RAPH (**D**)ay (**B**)AUD (+) zeigt die Entwicklung der letzen 24 Stunden an:



+	Zeigt den Bereich des maximalen Wertes an,
*	Zeigt den Bereich des Durchschnittwertes an,
#	Zeigt den minimalen Wert in dem jeweiligen Zeitintervall an.

Mit dem Portgraph ist nun auch z.B. eine zeitliche Gegenüberstellung von Info-Frames und Reject-Frame möglich.





R Zeigt die Anzahl der empfangenen Frames,
T Zeigt die Anzahl der gesendeten Frames an.

Man kann hier nun deutlich sehen, dass es keinen Zusammenhang zwischen Datenmenge und Reject gibt. Diesen Link Reject frei zu bekommen wird wohl schwierig.

Nach Starten des Knotens wird eine Minute mit der Erfassung der Daten gewartet, da einige Werte, z.B. RPS, erst errechnet werden müssen.

(**H**)ELP

Zu den Befehlen in TheNetNode gibt es jeweils auch eine Erklärung bzw. Hilfe. Mit der Eingabe (**H**)ELP wird eine Übersicht über die möglichen Hilfen ausgegeben sowie auch die Anzahl der Bildschirmseiten. Mit **(H)**ELP **(N)**ODES werden die ersten 22 Zeilen ausgegeben. Die Zeilen 23-44 kommen nach der Eingabe **(H)**ELP **(N)**ODES 2 . Wer zurückblättern kann, kann auch alle Seiten auf einmal übermittelt bekommen durch das anhängen eines „*”. Beispiel: **(H)**ELP **(N)**ODES *

```
Programm-1.0 vom Feb 29 2000 by DL1XAO
TNN-Doku-Version 1.79 vom 18. Oktober 2006 NORD><LINK e.V.

Folgende Befehle sind laut Dokumentation verfügbar :

Befehl              Hilfe-Seiten      Befehl              Hilfe-Seiten
(A)KTUELL      (extern)  1           (BE)ACON                 1
(C)ONNECT                6           (CONV)ers               19
(CQ)                     1           (D)EST                   2
(DX)CLUSTER              1           (G)RAPH                  5
(H)ELP                   3           (HA)RDWARE     (extern)  1
(I)NFO         (extern)  1           (L)INKS                  1
(L3)MHEARD               3           (LOOP)                   1
(M)AILBOX                1           (MAP)          (extern)  1
(MH)EARD                 3           (MSG)          (extern)  3
(N)ODES                 12           (P)ARAMETER              6
(PAC)SAT                 1           (PI)NG                   1
(PO)RT                  11           (Q)UIT                   1
(QTH)          (extern)  2           (R)OUTES                 9
(S)TAT                  14           (SAT)          (extern)  1
(SAV)ecall     (extern)  1           (SH)owcall     (extern)  2
(SO)FTWARE     (extern)  1           (TA)LK                   1
(TI)ME                   1           (TOP)          (extern)  5
(U)SER                  12           (V)ERSION                1

Externe Befehle sind nicht bei jedem Digi vorhanden !
```

(L3)MHEARD

Gibt eine aktuelle Rufzeichenliste der letzten 10 gehörten L3-Calls mit Datum, Uhrzeit, Port-Name, RX-Byte, TX-Byte, L3 Frame von Call und L3 Frame an Nachbar geroutet. Die L3MH-Liste wird wie die Statistik im aktuellen Laufwerk gesichert. Sie dient dazu, die L3-Verbindungen die über den Knoten laufen, beurteilen zu können. Die SSID wird hierbei beachtet.

```
KS:DB0EAM> MHEARD (97/500)
30.01.98 17:29 P 6 [   6095559,  12686304] DB0GOE    DB0GOE
30.01.98 17:29 P 6 [     67218,    153548] DB0MAR    DB0GOE
30.01.98 17:29 P 8 [    421886,  15142753] DB0LIP    DB0LIP
30.01.98 17:29 P11 [   5535466,  17842422] DB0KH     DB0KH
30.01.98 17:29 P 7 [     24644,   1217402] DB0AX-1   DB0AX-1
30.01.98 17:29 P 6 [     62988,    519602] DB0PDF    DB0GOE
```

MHEARD (<anzahl>/<anzahl>)
Die beiden Ziffern in der ersten Zeile geben an:
1. <anzahl> = Länge der geführten MH-Liste,
2. <anzahl> = Eingestellte Länge.

(L3)MHEARD <anzahl>

Gibt eine aktuelle Rufzeichenliste der letzten <anzahl> gehörten Calls mit Datum, Uhrzeit, RX-Byte, TX-Byte, L3 Frame von Call und L3 Frame an Nachbar geroutet aus.

(L3)MHEARD <call>

Listet wann und unter welcher SSID der Knoten mit dem <call> zuletzt ein L3-Frame über diesen Digi gesendet hat. Weiterhin werden die RX-Byte und TX-Byte (aus der Sicht des Knotens) mit angezeigt, die seit dem letzen Löschen der L3MHEARD-Liste oder Verändern der Anzahl der Listeneinträge aufgelaufen sind.

Im <call> können auch Wildcards verwendet werden. Dabei steht „*” für beliebig viele (oder keine) Zeichen und „?” steht für genau 1 Zeichen.

(**L3**)MHEARD df6ln = Einträge mit dem Rufzeichen DF6LN
(**L3**)MHEARD df* = Einträge von DF-Stationen
(**L3**)MHEARD *b? = Stationen mit „B” als vorletztem Buchstaben
(**L3**)MHEARD *b* = Stationen mit „B” im Rufzeichen

(**L**)INKS

Zeigt die eingetragenen Rufzeichen, die den Links zugeordnet sind.

```
Type-Port-ALIAS:CALL------Route---------
  L+  4   KSDXC:DB0EAM-4
  L+  5   KSBOX:DB0EAM-3
  I   6    GOE7:DB0GOE
  F   9     BID:DB0BID
  N   9   SARTG:HB9AK     via DB0BID
```

Typ:

I	= Nachbar arbeitet mit dem InterNode - Protokoll
L	= Localer Eintrag
L+	= Localer Eintrag wird gemessen
F	= FlexNet-Nachbar
N-	= NetRom-Nachbar mit altem Protokoll und Übermittlung und Messung geschieht im Level-2
N	= NetRom-Nachbar mit Level-3 Protokoll alter Art, aber Meßframes unprotokolliert
N+	= NetRom-Nachbar mit ON5ZS Protokoll und Übermittlung und Messung protokolliert.

(LOOP)

LOOP DETECTED ist eine Warnung, dass der gewählte Verbindungsaufbau eine Schleife (Loop) bildet. Es wird also eine Interlink-Verbindung in beiden Richtungen und somit unnötig belegt. Im Gegensatz zu anderer Knotensoftware, erfolgt jedoch nur eine WARNUNG, die Verbindung kann trotzdem aufgebaut werden.

Letzteres ist aber nicht immer sinnvoll und man behindert sich selbst.. Bitte mit 2 mal Quit zum vorletzten Digi zurück gehen und dann die Verbindung direkt aufbauen... Es kann aber auch vorkommen, dass der Router von TNN einen neuen Weg zum Zielknoten gefunden hat, der zu der Schleife führt — dann kann man die Warnung auch einfach ignorieren.

(**M**)AILBOX

Wurde eine locale Mailbox eingetragen, so reicht die Eingabe eines <m> zum Connect dieser Mailbox. Sinn dieses Befehles ist es, nicht immer auf einem Knoten nach der nächstgelegenen Mailbox suchen zu müssen.

(**MH**)EARD <erweiterung>

Gibt eine aktuelle Rufzeichenliste der letzten 10 gehörten Calls mit Datum, Uhrzeit, Port-Name, RX-Byte und TX-Byte aus. Die MH-Liste dient nun auch dazu, einen User auf dem Port zu connecten, auf dem er zuletzt gehört wurde. Die SSID wird hierbei beachtet. Es ist also möglich, mit dem Call DB0XY-1 auf dem Port 0 und mit dem Call DB0XY-2 auf einem anderen Port QRV zu sein. Wer selten QRV ist, fällt nun, je nach Länge der Liste, irgendwann aus ihr raus.

<erweiterung>
Als Erweiterung kann ein „+“ eingegeben werden. Es wird dann eine erweiterte User-Statistik ausgegeben. Sie besteht aus den vom User empfangenen Rej = (r) an den User gesendete Rej = (t) sowie die Anzahl der DAMA-Verstösse = (d).

```
KS:DB0EAM> MHEARD (400/500)
19.03.98 13:34 P 9 [   927365,  1932892] DG9FU      126r   119t     0d
19.03.98 13:33 P 0 [    59149,   407445] DG2ACM      45r    28t     0d
19.03.98 13:33 P 0 [    24821,   856196] DK7VW       42r    23t     0d
19.03.98 13:33 P 0 [    40804,   231704] DG2ACM-1    49r    44t     0d
19.03.98 13:33 P 0 [      880,    43047] DK3ZL        4r     0t     0d
DG9FU de DB0EAM >
```

MHEARD (<anzahl>/<anzahl>)
Die beiden Ziffern in der ersten Zeile geben an:
1. <anzahl> = Anzahl der geführten Calls in der MH-Liste,
2. <anzahl> = Länge der MH-Liste.

(**MH**)EARD <anzahl> <erweiterung>

Gibt eine aktuelle Rufzeichenliste der letzten <anzahl> gehörten Calls mit Datum, Uhrzeit, RX-Byte und TX-Byte aus.

(**MH**)EARD <call> <erweiterung>

Listet wann und unter welcher SSID der User mit dem <call> den Knoten zuletzt benutzt hat. Weiterhin werden die RX-Byte und TX-Byte (aus der Sicht des Knotens) mit angezeigt, die seit dem letzen Löschen der MHEARD-Liste oder Verändern der Anzahl der Listeneinträge aufgelaufen sind.

Im <call> können auch Wildcards verwendet werden. Dabei steht „*“ für beliebig viele (oder keine) Zeichen und „?“ steht für genau 1 Zeichen.

(**MH**)EARD df6ln	= Einträge mit dem Rufzeichen DF6LN
(**MH**)EARD df*	= Einträge von DF-Stationen
(**MH**)EARD *b?	= Stationen mit „B“ als vorletztem Buchstaben
(**MH**)EARD *b*	= Stationen mit „B“ im Rufzeichen

(**N**)ODES

Zeigt alle momentan bekannten Ziel-Knoten an. Das sind sowohl solche, die das NET/ROM- bzw. TheNet-L3-Protokoll verwenden, als auch Flexnet-Ziele. Die ausgegebene Liste wird regelmäßig über so genannte Rundspruchsendungen (Broadcast) der Nachbarknoten auf dem neuesten Stand gehalten. Die Liste ändert sich also, wenn Links ausfallen oder neue Links hinzukommen. Außerdem ändern sich die Laufzeiten der einzelnen Einträge in Abhängigkeit von der Linkbelastung und bei schlechten Ausbreitungsbedingungen. Beispiel:

```
 1    2             3    4
 /    /             /    /
KS:DB0EAM > Nodes (139/1009):

SH9600:DB0AZ    HUSUM:DB0HES    HHWEST:DB0HHW   HL:DB0MAR
KIELMB:DB0OQ       SL:DB0SUE      SYF7:OZ3DIJ-7
  /      /
 5      6
```

1.) Alias dieses Netzknoten.
2.) Rufzeichen dieses Netzknoten.
3.) Anzahl der bekannten Knoteneinträge.
4.) Maximal mögliche Anzahl an Einträgen
5.) Alias des Endknoten.
6.) Rufzeichen des Endknoten.

ALIAS ist dabei eine maximal 6-stellige Abkürzung zur besseren geographischen Einordnung des Knotenstandortes. In unserem Raum werden normalerweise die üblichen Autokennzeichen als ALIAS verwendet oder aber kurze Ortsnamen auch ausgeschrieben (wie z.B. bei KS:DB0EAM oder KIEL:DB0IL). Zu den in der Liste aufgeführten Endknoten kann in der Regel mit dem Connect-Befehl eine Verbindung hergestellt werden. Endknoten, deren ALIAS mit „BOX" oder „MB" endet, sind damit als Mailbox erkennbar; DX-Cluster sind mit „DX" oder „DXC" am Ende des ALIAS erkennbar.

Die Anzahl der bekannten Netzknoten wird hinter „Nodes" in Klammern angezeigt.

Der Nodes-Befehl kann bis auf den ersten Buchstaben abgekürzt werden und auch beliebig groß und / oder klein geschrieben werden. Der Nodes-Befehl kann außerdem mit Parametern aufgerufen werden, um Informationen zu einzelnen Endknoten oder Gruppen von Endknoten zu bekommen.

Mit dem Befehl:

(**N**)ODES <call> oder (**N**)ODES <alias>

bekommt man eine Auflistung der bekannten Wege zu dem mit <call> bzw. <alias> angegebenen Endknoten. So erhält man z.B. mit „(N)odes DB0FC". Weiterhin wird hiermit der NetRom - Route - Rekord (NRR) ausgelöst oder Flexnet-Routentest.

Die Abfrage wird auf dem Weg ausgesendet, über den auch ein Connect erfolgen würde. Zur Erklärung: TheNetNode ist nicht dafür ausgelegt, Flexnet-SSID-Bereiche im Netz weiterzuleiten. Ausnahmen hierbei sind mit L + eingetragene Ziele die auch erreichbar (QRV) sind und auf das Messframe reagieren. Nicht weitergereicht werden SSID-Bereiche von FlexNet Knoten. So ergeben auf DB0AX-1 die Abfragen:

```
n db0bid

Routes to BIDFLX:DB0BID
---T[ms]----RxT----TxT--LT-Mode-Obc-----RTT-Po-Route---------
>   3060   2480      0   3   DG   0     580  1 DB0EAM
DG9FU-4 de DB0AX-1 (18:57) >

PB:DB0AX-1>
Route (DG): DB0AX-1 DB0EAM DB0BID-7*<flexgate> DB0EAM DB0AX-1

n db0bid-1

PB:DB0AX-1>
No entry for: DB0BID-1
DG9FU-4 de DB0AX-1 (18:57) >
```

Auf einem Knoten mit FlexGate wäre hier nun die Abfrage nach DB0BID-1 folgerichtig auf dem FlexGate ausgesendet worden, da nur dort ein einstufiger Connect nach DB0BID-1 möglich ist. User auf DB0AX-1 können DB0BID-1 nur erreichen wenn sie erst DB0BID und dann DB0BID-1 connecten. Ein SSID-Bereich lässt sich nicht in eine SSID zwängen.

(**N**)ODES <call> * oder (**N**)ODES <alias> *

Ein zusätzliches „*" hinter <call> zeigt alle Wege an.

```
              1   2
             /   /
Routes to   BRO:DB0BRO
---T[ms]----RxT----TxT--LT-Mode-Obc-----RTT-Po-Route--------------------
>   4330   3910      0   2   DG   0     420  6 DB0GOE
   16440   7000   4870  10   DG  19    9440  9 HB9AK via DB0BID
   30970  17560   4870   5   DG   0   13410  9 DB0LBG-7 via DB0BID
  600000 599990   4870   4   DG   0    1550  9 DB0KH via DB0BID
>   4800   4800      0  10   VC   0     910 10 DB0NHM
    5970     10      0   1   DG   0    5960 10 DB0BRO via DB0NHM
   25200   3480      0   2   DG   0   21720 10 DB0GOE via DB0NHM
  600000 599990   4870   4   DG   0     520 11 DB0KH
^   /     /         /   /    /   /       /  /    /
3  4     5         6   7    8   9       10 11   12
```

1.) ALIAS-Kennzeichen des gefragten Endknoten (nur wenn es bekannt ist).
2.) Rufzeichen des gefragten Endknoten.
3.) > = Der Weg wird derzeit verwendet für eine Verbindung zu dem Endknoten.
+ = Zeigt eine alternative Route an, über die auch Datenübertragung stattfindet.
- = Diese Route ist abgemeldet und wird nicht mehr benutzt.
* = Es wird kein Obs mehr geführt, da der Nachbar bereits differenziellen Broadcast unterstützt.
4.) Gemessene Laufzeit in ms des jeweiligen Weges.
5.) RxT Laufzeit die vom dem Nachbar über das Ziel DB0BRO gemeldet wurde.
Bzw. 10 wenn es ein direkter Nachbar ist.
6.) Laufzeit mit der das Ziel weitergemeldet wird. (Errechnet sich aus RxT und gemessener Laufzeit zum Nachbarknoten).
7.) Lifetime des Knotens. Wenn LT=0 wird der Knoten nicht mehr weiter verbreitet.
8.) Übertragungsmode auf diesem Link DG = Datagramm (Höhere Protokollebene die das Umrouten ermöglicht) ; VC = Virtual Circuit (Unterste Protokollebene).
9.) Obc Obsolentcounter (Veraltenszähler) für Ziele mit altem Protokoll.
10.) Round - Trip - Timer oder Laufzeit zum Nachbarknoten.
11.) Port über den die Verbindung geroutet wird.
12.) Rufzeichen des Nachbarknotens für den jeweiligen Weg.

Bei einer Laufzeit 600000 ms wurde der Weg über Fastlearn aufgenommen.

NetRom - Route - Rekord (NRR) oder die Ermittlung des Übertragungsweges zwischen 2 Knoten die dieses Protokoll untersützen. Es wird an das Node ein NRR - Frame gesendet. Auf dem Hin- und Rückweg hängen alle Knoten Ihr Call in dieses Frame ein. Der Knoten, der das Frame an den Absender zurücksendet kennzeichnet sich selbst mit einem „*". Ein „DB0EAM*<flexgate>" oder „DB0EAM*<local>" bedeutet, dass dieser Digipeater das abgefragte Node über ein FlexGate oder einen local eingetragenen Link auf Level2 umsetzt. Digipeater die NRR nicht unterstützen, jedoch Level3-Frames transportieren, erscheinen in der Liste mit einem „?". Unterstützt das abgefragte Node jedoch NRR nicht, so kommt keine Antwort zurück.

KS:DB0EAM> Route (DG): DB0EAM DB0GOE DB0BRO* DB0EAM

Wenn die INP-Optionen durchgängig weitergereicht werden, so könnte auch die IP-Adresse mit Subnet-Maske noch hinter dem Knoten-Call erscheinen: Routes to SH9600:DB0AZ (44.130.5.129/26)

Folgende Konstellation bei DB0SHG:

```
 Routes to     BS:DB0FC
---T[ms]----RxT----TxT--LT-Mode-Obc-----RTT-Po-Route----------
-  33800  33800  33900  10   VC   0   12880  4 DB0HSK
>  17700  17700      0  10   VC   0   77830  6 DB0HW
   43360     10      0   1   DG  21   43350  6 DB0FC via DB0HW
```

bedeutet nichts anders, als das ein User der DB0FC connecten möchte, den FlexNet-Weg über DB0HW vorgeschlagen bekommt, weil sonst keine wesentlich schnellere Alternative zur Verfügung steht. Eine solche Verbindung kann natürlich NICHT umgeroutet werden, selbst wenn zwischendurch ein besserer NETROM-Weg zur Verfügung steht!

Umgedreht wird nie eine NETROM-QSO auf FlexNet-Weg umgeroutet. Hin- und Rückweg können bekanntlich unterschiedlich Wege nehmen.

Um erweiterte Informationen über eine Gruppe von Endknoten zu bekommen, gibt man den Befehl ein:

(**N**)ODES <laufzeit> <name>

Für <laufzeit> wird dabei der Wert für die minimale Laufzeit der Endknoten angegeben. Soll nicht eine minimale, sondern eine maximale Laufzeit angegeben werden, wird vor der Laufzeit ein Minus geschrieben. Für <name> kann ein Rufzeichen oder ein ALIAS eingesetzt werden, bei dem „*" und „?" als Platzhalter verwendet werden können. Dabei steht „*" für beliebig viele (oder keine) Zeichen und „?" steht für genau 1 Zeichen. Ausserdem kann man die Abfrage auf den ALIAS-Teil oder auf den Rufzeichen-Teil des Knoteneintrages beschränken, indem man ein „:" vor oder hinter <name> setzt. Mit „<name>:" begrenzt man die Abfrage auf den ALIAS-Teil, und mit „:<name>" begrenzt man die Abfrage auf den Rufzeichen-Teil der Knoteneinträge. Wird nur eine <laufzeit> angegeben, so werden alle Knoten mit passender Laufzeit angezeigt. Wird nur ein <name> angegeben, so wird die Laufzeit bei der Auswahl nicht berücksichtigt. Die ausgegebene Liste an Endknoten sieht dann z.B. so aus:

```
KS:DB0EAM > Nodes: (7)

  HHS:DB0HBS  176/11 HHSBOX:DB0HBS-1  69/11
   /     /    /  /
  1     2    3  4
```

1.) ALIAS des Endknoten.
2.) Rufzeichen des Endknoten.
3.) Laufzeit in 10 ms Schritten für den besten bekannten Weg.
4.) Port für Nachbarknoten.

Beispiele:

```
Befehl        ! Antwort
==============!=========================================================
NODES 2000    ! Alle Einträge mit Laufzeiten min. 2000.
NODES -60     ! Alle Einträge mit Laufzeiten max. 60.
NODES 100 d*  ! Rufzeichen oder ALIAS beginnt mit „D", Laufzeit min. 100.
NODES -100 *d ! Rufzeichen oder ALIAS endet mit „D", Laufzeit max. 100.
NODES *d*     ! Rufzeichen oder ALIAS enthält ein „D", Laufzeit egal.
NODES ?????   ! Rufzeichen oder ALIAS ist genau 5 Zeichen lang.
NODES :d*     ! Rufzeichen beginnt mit „D".
NODES :?b*    ! Rufzeichen mit „B" als 2. Buchstaben (DB- u. HB9-Stationen).
NODES :hb9*   ! HB9-Rufzeichen.
NODES :*l     ! Rufzeichen endet mit „L".
NODES k*:     ! ALIAS beginnt mit „K".
NODES *box*:  ! ALIAS enthält „BOX".
NODES *dx:    ! ALIAS endet mit „DX".
NODES *u?:    ! ALIAS mit „U" als vorletztem Buchstabe.
NODES *       ! Zeigt die gesamte Liste an.
```

(**N**)ODES < <Nachbar Call>

Zeigt die Nodes / Destinations an, die von diesem Nachbarn mit der besten Laufzeit gemeldet wurden.

(**P**)ARAMETER

Ausgabe der Parameterliste.

```
KS:DB0EAM> Parms:
01:NoAckBuf        20   02:L3-MaxTime       0   03:SaveConfig       3
04:DAMA-Speedf   4800   05:DAMA-MaxPri     10   06:DAMA-MaxPol      5
07:DAMA-Tout       50   08:CommandLog       0   09:SysopLog         1
10: TestSSID       15   11:ConvSSID         1   12 AutoIPR          3
DG9FU de DB0EAM (13:12)>
```

01: NoAckBuf

Überfüllungsgrenzwert in Anzahl Frames. Anzahl der Pakete, die auf der Transport-Layer-Ebene zwischengespeichert werden, bis eine Choke-Nachricht zum vermittelten Knoten geschickt wird. Gleichzeitig die Anzahl Frames, die im Link-Layer zwischengespeichert werden, bevor das Link-Layer in den Busy-Zustand geht. Dieser Grenzwert** verhindert den Überlauf eines TheNet-Knoten, falls über das Transport-Layer zu viele Pakete einlaufen, oder falls eine Station in einem Link zu viele Pakete auf einmal senden will.

02: L3-MaxTime

Über diesen Parameter teilt der Nachbarknoten mit, welche MAX-Laufzeit die Zielknoten haben dürfen die er gemeldet haben möchte. Dieses entlastet das Meldeverhalten. Eingestellt wird in 10mS Schritten 0...60000 was einer Zeit von 0...600.000mS entspricht.

03: SaveConfig

Zeitraum in 10 min-Schritten, in dem TNN179.STA und MHEARD.TAB mit den aktuellen Statistikdaten auf Disk oder HD gespeichert wird.

04: Dama-Speedf

Bei TNN-Digis mit Multibaud (z.B. 1200 und 9600 Baud) bestand ebenfalls ein weiter Wunsch, einen Anreiz für 9k6-Betrieb zu schaffen. Im Normalfall werden 1200- und 9k6-User gleichbehandelt, und man kann lediglich über MAXFRAME den 9k6-Betrieb etwas bevorzugen. Üblicherweise sollte man MAXFRAME 7 auf 9k6 benutzen und nicht mehr als MAXFRAME 2 auf 1200 Baud. Aber einigen Sysop war dies noch nicht genug. Was der neue DAMA-Speedfaktor bewirkt, soll nun am folgenden Beispiel erläutert werden. Angenommen, wir haben einen 1200-Baud- und einen 9600-Baud-Einstieg, und wir wollen die 9600er User deutlich bevorzugen. Steht DAMA-Speedf jetzt auf 9600, dann werden User auf einem 1200-Baud erst in jeder 8. Runde bedient. Auf den ersten Blick scheint dies viel zu langsam, aber bei 9k6 sind die Frames ja so kurz, so dass es also NICHT gleich um Faktor 8 langsamer geht. Sind überhaupt keine 9k6-User da, dann ist alles wie bisher. Wichtig: Die korrekte Baudrate muss im Speed-Parameter für den jeweiligen Port richtig eingetragen sein.

Die „Verlangsamung" berechnet sich so:

$$\text{Verlangsamung} = \frac{\text{DAMA-Speedf}}{\text{Speed[Port]}} = \frac{9600}{1200} = 8$$

Bei dem 9k6-Port ergibt sich entsprechend ein Faktor von 1. Hat man gleichzeitig noch einen 2k4-Einstieg, dann ist die Verlangsamung dort 9600/2400 = 4. Möchte man den 1200er Einstieg nicht gleich um Faktor 8 beeinflussen, kann man natürlich einen beliebigen Wert zwischen 1200-9600 (oder mehr!) bei diesem Parameter eintragen. Ein Wert kleiner als die niedrigste Baudrate bzw. ein Wert von 0 schaltet diese Funktion ab.

05: DAMA-MaxPri

Maximaler Zählerstand, der erreicht werden kann bei geringer Aktivität eines User. Dieser User wird dann seltener zu Gunsten der anderen User abgefragt, ob er Daten zu senden hat.

06: DAMA-MaxPol

0 = ausgeschaltet.
x = Nach x Verstößen erfolgt ein Abwurf vom DAMA-Master.

Die TNN-Software bietet für den SYSOP eine **abschaltbare** DAMA-Kontrolle. Alle User, die sich nicht an das DAMA-Protokoll halten, bekommen bei jedem DAMA-Verstoss eine entsprechende WARNUNG mitgeteilt. Ist eine, vom Sysop einstellbare, Anzahl überschritten, dann erfolgt ein Abwurf des Users vom Digi! Die „Meckermeldungen" werden nicht mehr nutzlos als UI-Frame (Bake) ausgesendet werden, sondern sie werden jetzt mitten in das betreffende QSO eingehängt. Die Warnung gibt die Anzahl der aktuellen Verstöße an sowie die Anzahl der Verstöße, bis ein Disconnect seitens des DAMA-Master erfolgt (mit entsprechender Meldung).

07: DAMA-Tout

Zeit in 10 ms, die gewartet wird, wenn auf ein Paket an einen User keine Bestätigung kommt, bis zum Senden an den nächsten User in der Reihenfolge.

08: CommandLog
Schafft nun die Möglichkeit, eine Logdatei COMMAND.LOG im Verzeichnis TNN zu führen. Es werden alle Eingaben im Knoten mit CALL, DATUM, UHRZEIT und COMMAND in der Datei abgespeichert. Es sind folgende Einstellungen möglich:

0 = Keine Protokollierung.
1 = Nur Sysop-Eingaben werden protokolliert.
2 = Alle Eingaben werden protokolliert (Vorsicht, die Logdatei wächst schnell!).

09: SysopLog
Speichert wer sich als SYSOP beim Knoten privilegiert.

10: TestSSID
Legt die SSID fest, mit der dieser Knoten die Laufzeitmessung durchführt bei einem mit L+ eingetragenes Call.

11: ConvSSID
Einstellung der SSID die der Convers benutzen soll.

12: AutoIPR
Automatisches Erstellen von IPR und ARP-Einträgen.
Wobei Stufe 3 unbedingt zu empfehlen ist.

0 = Automatik ausgeschaltet
1 = Automatik eingeschaltet. Keine Prüfung auf Richtigkeit der IP-Adresse
2 = Unmögliche IP-Adressen werden ignoriert.
3 = Automatische Ein/Austragung, es werden nur IP-Adressen beruecksichtigt, die im GLEICHEN Netz und gültig sind. (default) (Bsp: hat man die IP 44.1.2.3 werden nur andere 44.x.x.x-IPs eingetragen)

(**PAC**)SAT
Zeigt Call und Message-Pool des BROADCAST-Server an.

```
Server call: XX0XX-2
Message pool: 1-9 (9 Messages)
```

(**PI**)NG <ipadd>
Führt eine Laufzeitabfrage zu der IP-Nummer durch.

(**PO**)RT
Gibt eine Liste mit den aktuellen Porteinstellungen aus: (Diese Darstellung ist ein MISCHMASCH aus DOS/GO32- und LINUX- Version und dient hier nur zur Verdeutlichung.)

```
TEST:DB0XX> Link-Interface Ports:
-#-Name----------Speed/Mode-Max-TXD-DA---------------Hardware--
 0 70cm_1200       1200m      2a  20  1 CTEXT       MH VAN v4.04E
 1 70cm_9600       9600m      7a  15  1 CTEXT       MH VAN v4.04E
 2 23cm_9600       9600       7a  10  2 CTEXT       MH VAN v4.04E
 3 Lohfelden       9600d      4   10    CTEXT          TOKENRING/dev/ttyS0
 4 Mailbox        64000d      7    0                   TNN-ETH
 5 HHW             9600       7    5    CTEXT SYSOP    SMACK /dev/ttyS5
 6 Fyn             1200       4   25    CTEXT          SMACK /dev/ttyS3
 7 SEG             9600       7    5    CTEXT          SMACK /dev/ttyS1
 8 Cluster        19200       7  200                   KISS /dev/ptyz3
```

#:
Gibt den Port des Knotens an.

Name:
Zur leichteren Übersicht.

Speed/Mode:
Speed zeigt die eingestellte Baudrate sowie zusätzlich gesetzte Flags an.

d : Duplex.
c : CRC bzw. DCD bei 1k2-Modem.
r : ext. Takt (rx).
t : ext. Takt (tx).
e : ext. Takt beide (Vanessa).
m : Multibaud-Kopplung (Vanessa, SCC).
z : NRZ statt NRZI.

MAX:
eingestelltes MaxFrame. Das „a" bedeutet das die Frameautomatik eingeschaltet ist.

TXD:
eingestelltes TXDelay.

DA:
Der Kanal ist dem DAMA-Master 1..16 zugewiesen.

CTEXT:
Auf diesem Port werden die Connect-Texte ausgesendet.

SYSOP:
Auf diesem Port ist der SYSOP-Mode aktiviert. Ein L2-Connect ist nur mit Passwort möglich.

MH:
Auf diesem Port wird die MH-Liste geführt.

Hardware:

Tokenring:	Serielle Übertragung auf dem Tokenring.
Van 4.04e:	Parallele Übertragung zum Frontendrechner Vanessa mit der Softwareversion.
Kiss1:	Übertragung über die eingestellte serielle Schnittstelle ohne CRC.
SMACK1:	Wie KISS1 jedoch mit SMACK-CRC.
RKISS1:	Wie KISS1 jedoch mit RMNC-CRC.
IPX_PD:	Hier ist eine Ethernet-Karte im Einsatz mit IPXPD-Treiber.
TNN-ETH:	Ethernet Packet Treiber.
6PACK:	6PACK Treiber.

Bei Linux wird zusätzlich die Device-Bezeichnung des Port angegeben.

(**PO**)RT ***** oder (**PO**)RT **+**
Gibt eine Liste mit den aktuellen Port Parametern aus:

```
WHV:DB0WHV> Port Parameters:
              TX-                   Max-  L2-          Max EAX- EAX-
-#-Port-------Delay-Pers-Slot-IRTT-Frame-Retry-Timer2-Con-MaxF-Mode
 0:9k6          10   255  10   140   7a    10     60    0  16    1
 1:1k2          20   255  20  1000   5a    10    480    0  16    1
 2:Link-BHV     12   200  12    84   7     20     30    0  16    1
 3:Link-LER      0   255   1    30   7     20     15    0  16    1
 4:Link-TNC      0   255   1    30   7     20     15    0  16    1
 8:Funkruf       0   255   1    30   7     20     15    0  16    1
 9:Box-Link      0   255   1    10   7     20      5    0  16    1
10:ServerLink    0   255   1    30   7     20     15    0  16    1
```

Nur TXDelay und MaxFrame werden noch vom Sysop eingestellt. Die anderen Parameter stellt TNN selbst ein.

TXDelay:
Sendervorlaufzeit nach dem Hochtasten des Senders bis zur Aussendung des ersten Datenpaketes in 10ms.

Pers:
Persistence Wert (0-255).

Slot:
Zeitschlitz-Intervall (Slot time intervall) in 10ms. Dieser Parameter gibt die Dauer des Zeitschlitzes für die Persistance-Steuerung an. Jedes Mal, wenn der TNC ein Paket ausstrahlen wollte und die unter Slot-Time beschriebene Zufallszahl außerhalb des Persistance-Bereichs lag, wird dann für die Dauer des Zeitschlitzes gewartet und anschliessend die Persistance-Prozedur erneut durchlaufen.

IRTT:
Bedeutet Initial-Round-Trip-Time und ist der Anfangswert für die RTT-Berechnung, also der Wert, der für die erste Berechnung anstelle von SRTT benutzt wird, wenn noch keine Messung erfolgen konnte.

Der Timer 1 (wann soll der TNN Nachfragen, wenn er Deine Antwort nicht gehört hat) berechnet sich wie folgt:

Für RETRIES kleiner gleich 3:	T1 = SRTT * 3
für RETRIES grösser 3:	T1 = SRTT * (RETRIES + 4)

Beim Connecten wird SRTT aus dem IRTT berechnet (siehe Parms):

SRTT = (DIGI*2 + 1) * IRTT

Bei einem direkten Connect entspricht SRTT dann dem IRTT, mit Anzahl DIGI wird es dann entsprechend größer. Wenn nun auf einem Ports ein IRTT = 500 = 5s aus der Baudrate errechnet wurde, bedeutet das:

Retry 1..3 --> T1 = 5 * 3 = 15s bis zur Nachfrage !

z.B. bei Retry 6 --> T1 = 5s * (6 + 4) = 50s!!!

Bei laufendem Betrieb bewegt sich der SRTT dann dynamisch.

MaxFrame:
Anwender-Link MaxFrame in Anzahl der Frames. Anzahl der Infopakete auf Layer2-Ebene, die ohne Erhalt einer Bestätigung hintereinander ausgesendet werden dürfen. Das „a" zeigt die eingeschaltete Frameautomatik an.

L2Retry:
Bestimmt die Anzahl der Versuche, um auf Layer2-Ebene Kontakt zu einer anderen Station zu bekommen (Antwort auf Kommandos und Poll). Nach dieser Anzahl von Versuchen wird der Link als defekt gemeldet, falls keine Antwort erfolgt.

Timer2:
Anwender-Link T2 in 10ms. Dieser Timer bestimmt die Wartezeit, nachdem ein eingehendes Informationspaket bestätigt wird mit einem RR/REJ/RNR-Paket. Einerseits ist diese Verzögerung zur Durchsatzsteigerung da, weil man in diesem Intervall anderen eine Chance zum Senden gibt, andererseits wird dem Sende-Layer die Chance gegeben, eine Bestätigung in ein zu sendendes Infopaket zu packen und somit ein Link-Layer-Paket einzusparen. (Das Ergebnis ist in der Statistik abzulesen.)

EAX-MaxF:
MaxFrame für EAX.25 in Anzahl der Frames. Anzahl der Infopakete auf Layer2-Ebene, die ohne Erhalt einer Bestätigung hintereinander ausgesendet werden dürfen. Das „a" zeigt die eingeschaltete Frameautomatik an. EAXMAXF kann max. 127 sein, sollte jedoch 32 nicht überschreiten, default ist 16. Bisher ist ein Maxframe größer 7 noch nicht mit allen Hardwareinterfaces getestet!!! Hier muss experimentiert werden. (auf VANESSA funktioniert es einwandfrei)

EAX-Mode:
Einstellung für das Verhalten von EAX.25.

Mode 0:	nur AX.25
Mode 1:	AX.25 und EAX.25, Connects nach MHeard (default)
Mode 2:	AX.25 und EAX.25, ausgehende Connects zuerst immer in EAX.25, erfolgt nach zwei SABME keine Antwort, dann Rückfall auf AX.25
Mode 3:	nur EAX.25 erlaubt

(**Q**)UIT

Verbindung wird vom Knoten aufgelöst (DISCONNECT). Dadurch ergibt sich die Möglichkeit, zum vorher connecteten Knoten reconnected (zurückverbunden) zu werden. Wenn vorhanden, wird der Text QUIT.TXT vor dem Auflösen der Verbindung gesendet.

(**R**)OUTES

Zeigt die bestehenden Routen zu den Nachbarknoten an.

```
Routes of WHV:DB0WHV (10/16)
Node------SSID--Typ-Po--Dst/Rou---L3SRTT[ms]---MaxT[ms]-State--Route----------
DB0BHV      0    N   2    0/0       390/370           0 conn
DB0LER     0-9   F   3   11/7       310/200      300000 conn.
DB0PDF      0    I   3   48/17     2270/2900     100000 conn.  DB0LER
DB0FHO      0    I   4  531/523     350/350           0 conn.  DB0TNC
DB0TNC     0-14  F   4    4/3       270/200      300000 conn.
DB0WHV-6    6    L+  8    1/1       100/----             active
DB0WHV-7    7    L   9    1/1
```

Routes of WHV:DB0WHV (10/16)

Die Routesliste ist auf das wesentliche gekürzt! Die (10/16) besagt, dass in der Linkliste 10 von 16 möglichen Einträgen benutzt sind. Die Länge der Linkliste kann ggf. mit SET TNNCFG=xxxx,xx angepasst werden. (siehe: START.BAT)

NODE :

Anzeige des Rufzeichens des benachbarten Knotens.

Type :

Gibt an, um welchen Typ es sich bei diesem Eintrag handelt. Möglich sind:

Type **I** :

Nachbar arbeitet mit dem neuen InterNode - Protokoll, in dem TheNetNode nur Laufzeiten austauscht.

Type **L** :

Local Call und Alias wird mit Laufzeit 4000 ms eingetragen. Es wird keine Laufzeitüberprüfung durchgeführt.

Wird ein L+ eingesetzt, so wird das Ziel alle 5 Minuten connected. Ist das Ziel QRV, so wird dieses mit der Laufzeit in die Routesliste eingetragen und weitergemeldet.

Type **F** :

FlexNet Port arbeitet mit FlexNet Protokoll. FlexNet-Knoten werden nun über TheNetNode wie ein TheNetNode-Knoten als Node verteilt. Ein Connect zu einem FlexNet-Knoten wird an einem TheNetNode-Knoten mit FlexGate umgesetzt auf Level 2. Der Weg bis zum FlexGate ist dabei wieder Level 3-4 und damit umzurouten. Sind an einem Knoten mehrere FlexGate in Betrieb so passt sich TheNetNode wie ein FlexNet-Knoten in das Netz ein.

Type **N** :

NetRom-Nachbar kann das Level-3 Protokoll alter Art, sendet jedoch die Messframes noch unprotokolliert zurück.

Type **N+** :

NetRom-Nachbar verwendet das ON5ZS-Protokoll, wo unerreichbare Ziele sofort abgemeldet werden. Die Übermittlung und Messung geschieht im Level-3 nun protokolliert.

Type **N-** :

NetRom-Nachbar verwendet das alte Protokoll. Nicht erreichbare Ziele fallen auf Grund der Timer aus der Nodesliste. Die Übermittlung und Messung geschieht im Level-2 als UI-Frames und damit NICHT gesichert!

Po :

Port, über den die Verbindung läuft.

Dst / Rou :

Anzahl der Ziele die von dem Nachbarn gemeldet wurden / Anzahl der Ziele mit kürzester Laufzeit über diesen Nachbar.

L3SRTT[ms] :

Zeigt die zu diesem Node ermittelte Laufzeit in Millisekunden an, sowie die der Gegenseite.

MaxT[ms] :

Maximale Laufzeit in Millisekunden, die an diesen Nachbarn noch gemeldet wird. 0 bedeute keine Grenze.

State setup :

Es wird versucht eine Verbindung zum Nachbarn aufzubauen.

State conn. :

Die Verbindung zum Nachbarn ist erfolgreich aufgebaut worden und bleibt von nun an bestehen. NUR wenn diese Level2 Verbindung besteht, werden auch die NODES / DESTINATIONS von diesem Nachbarn in die NODES-/DESTINATION-Liste übernommen. Damit werden nun Zielknoten aus dem Netz entfernt, die nicht sicher zu erreichen sind oder zu denen der Link nur in einer Richtung besteht.

Route :

Digipeater, die nicht das TheNet-Protokoll benutzen bis zum eingetragenen TheNet-Knoten.

(**R ***) oder (**R +**) = Routes Version :

Zeigt zusätzlich den Softwarestand der Nachbarknoten mit an und was für ein Broadcast dahin gemacht wird.

UI-Broadcast	= Alter TheNet Broadcast,
Info-Broadcast	= Broadcast wird gesichert im Level 3 übertragen,
Differential-Broaccast	= Es werden nur die Äderungen übertragen.

(S)TAT :

Gibt die Statistik, die im Knoten geführt wird, aus. Mit (S)TAT werden jedoch nur noch die Kopfdaten ausgegeben.

Weitere Statistiken:

(**S**)tat (*****)	Ausgabe aller Statistiken;
(**S**)tat (**E**)rror	Kopf und Error-Statistik;
(**S**)tat (**H**)ardware	Kopf und Hardware-Statistik;
(**S**)tat (**I**)p	Kopf und TCPIP-Statistik;
(S)tat (**K**)ernel	Kopf und Kernelinterface-Statistik (Linux)
(**S**)tat (**L**)ink	Kopf und Link-Statistik;
(**S**)tat (**P**)orts	Kopf und Port-Statistik.

Die Kopfdaten....

```
 System Statistics: 01.05.05 00:00:13 - 25.05.05 08:45:18
           Startup: 21.05.05 16:29:58
            Uptime:        3/16:15

                       (min)    (now)    (max)
       Rounds/sec:       337      737     1487
     Free Buffers:      1747     2369     8123
Overall Throughput:             32400    58208 Baud
   Active L2-Links:               109      179
   Active Circuits:                25       53
      Active Nodes:               781     1003

          TNN-Load: 0%
           Sysload: 0.94%
           Loadavg: 0.00 0.00 0.00
     CPU time used: user 198163ms, system 37882ms
      Buffer usage: 12%
      Network Heap: 376352 Bytes
```

System Statistics:

Datum und Uhrzeit des letzten Löschens der Statistik sowie das Auslesedatum und - Uhrzeit. Wichtig bei automatischen Abrufen der Statistik für Diagramme.

Startup: 21.05.05 16:29:58

Letzter Neustart der Software.

Uptime = 3/16:15

Laufzeit des Programmes in Tagen/Stunden:Minuten.

Die folgenden Angaben sind jeweils in dieser Reihenfolge aufgeführt: minimaler Wert aktueller Wert maximaler Wert

Rounds/sec: <anzahl> <anzahl> <anzahl>

Gibt die Hauptschleifendurchläufe des Programmes an. Bei der DOS/GO32-Version sind die angezeigten Werte abhängig von der Rechnerperformance. Bei Linux wird als aktueller Wert üblicherweise ca. 50 angezeigt — mit Tokenring ca. 100.

Free Buffers: <anzahl> <anzahl> <anzahl>

Zeigt die freien Buffer des Knotens an.

Ein Buffer sind 64 Info-Byte und 8 Listenzeiger-Byte, insgesamt also 72 Byte. Jede zwischengespeicherte Information und auch jeder Eintrag in der MHEARD-Liste belegt Buffer. Wenn TheNetNode keinen freien Buffer mehr zur Verfügung hat, wird ein Reset ausgelöst, der alle bestehenden Verbindungen mit totalem Informationsverlust löscht. Im Normalbetrieb ist dieser Fall aber nahezu ausgeschlossen, da bei normal gesetzten TheNetNode-Parametern eigentlich nie zu wenig Buffer vorhanden sein können (man kann nicht beliebig viele Pakete hintereinander im Knoten „abladen").

Overall Throughput: <anzahl> <anzahl> <anzahl>
Zeigt den Datendurchsatz des Knotens an. Hierbei handelt es sich um reine Infodaten ohne Protokoll- und Overheadbytes!

Aktive L2-Links: <anzahl> <anzahl>
Anzeige der Anzahl der aktiven und maximal gleichzeitig vorgekommenen Level-2-Verbindungen.

Active Nodes: <anzahl> <anzahl>
Hier wird die Anzahl der derzeit bekannten Nodes angezeigt. Die zweite Anzahl gibt die bisher maximal eingetragenenNodes an.

Aktive Circuits: <anzahl> <anzahl>
Anzeige der Anzahl der aktiven und maximal gleichzeitig vorgekommenen Circuit-Verbindungen. Hier werden natürlich nur die Verbindungen angezeigt, die auf diesem Knoten beginnen oder enden. Alle anderen Verbindungen werden ja virtuell durch das Netz geleitet und brauchen nicht auf jedem Knoten verwaltet zu werden.

CPU load: 39%
Zeigt die Auslastung der CPU an. Hier wird das Verhältnis aus maximaler und aktueller Zahl an Rounds/sec als Maß für die CPU-Auslastung berechnet.

TNN-Load: 0%
Die CPU-Auslastung kann unter Linux nicht aus /proc/loadavg ermittelt werden, da dort etwas ganz anderes ausgesagt wird, nämlich die IO-Last. Die echte CPU-Last wird nun intern selbst aus dem Verhältnis von Uptime zu Idletime der letzten zehn Sekunden berechnet.

Sysload: 5.00%

Loadavg: 0.01 0.05 0.06
Linux: Die Werte aus /proc/loadavg werden zur Information weiterhin ausgegeben.

CPU time used: user 7819s, system 1974s

Buffer usage: 14%
Verhältnis zwischen belegten und zur Verfügung stehenden Buffern in %.

Network Heap: <anzahl>
Sind die durch Routing - Infos belegten Bytes.

Die Hardware - Statistik....

```
Tokenring-Statistics:

       Tokens/sec:       92      161      180
   TOKENRING load: 11%

  Token ring speed: 38400 Bit/s.
  Token-Recoveries: 1
       Bad-Frames: 0
```

Tokens/sec: <anzahl> <anzahl> <anzahl>
Zeigt die Aktivitäten des Tokenringes an.

TOKENRING load: 36%
Zeigt die Auslastung des Tokenring an.

Tokenring speed: 38400 Bit/s.
Anzeige der eingestellten Baudrate des Tokenringes.

Token-Recoveries: <anzahl> <datum>
Sind Abfragen an die TNC, die nicht wieder zum Rechner zurückkommen.

BAD Frames:
Defektes oder falsches Token. Diese Fehler werden auf Port 0 gezählt, da nicht festgestellt werden kann, auf welchem Port der Fehler entstanden ist.

```
KISS-Statistics:
  KISS1      RxCRC:     0  RxErr:     0
```

KISS1 RxCRC: 0 RxErr: 0
Fehler dieser Kiss - Schnittstelle.

RxCRC Fehler:
Ein Frame mit falscher CRC wurde auf einem SMACK- oder RMNC-Kisslink empfangen. Diese Fehler werden erkannt und sind unkritisch, weisen aber auf eine schlechte Übertragung hin.

```
External-Statistics:
  TNN-ETH    TxErr: 0  RxOvr: 0  OFlow: 0  IOErr: 0
```

Fehler die durch externe Treiber entstehen.

Die Error - Statistik....

```
Error-Statistics:

                 RxID  RxLen  RxCtl Resets
 0:70cm_1200      26      0      0      0
 1:70cm_9600     347      0      0      2
 2:23cm_9600     383      3      0      0
 3:Lohfelden     170      3      0      2
 4:Cluster         0      0      0      0
 5:Mailbox         0      0      0      2
 6:Goettingen      0      0      0      0
```

Resets:
Anzahl der Resets der entsprechenden Port Hardware (Vanessakarten oder TNC im Tokenring).

Die IP - Statistik....

```
IP-Gateway-Statistics:

ipForwarding:      1      ipDefaultTTL:      32     ipInReceives:      0
ipInHdrErrors:     2      ipInAddrErrors:    0      ipForwDatagrams:   2
ipInUnknownProtos: 0      ipInDiscards:      0      ipInDelivers:      0
ipOutRequests:     1      ipOutDiscards:     0      ipOutNoRoutes:     2
ipReasmTimeout:    30     ipReasmReqds:      0      ipReasmOKs:        0
ipReasmFails:      0      ipFragOKs:         0      ipFragFails:       0
ipFragCreates:     0
```

Die Kernel-Interface Statistik....

```
Kernel-Interface statistics:
Bytes received : 429631
Bytes sent     : 260492
```

Die Link - Statistik....

```
Link to DB0GOE                    10.10.99 09:38:46 - 16.10.99 16:27:18
Frames:      I        UI        RR      REJ     RNR  SABM/UA  DISC/DM FRMR
RX:     518222         0    205289      242       0        0        0    0
TX:     838835         0    176972        8       0        0        0    0
Bytes:   Total      Info    Header Overhead     %I     %RR    %REJ    %RNR
RX:   48160406  36785889  11374517    23.6%  71.6%   28.3%    0.0%    0.0%
TX:  135351694 119275634  16076060    11.8%  82.5%   17.4%    0.0%    0.0%
TX:      Once: 119087365 Repeated:   188269 IQual:   99.8%  TQual:   87.9%
```

Link to <call> <datum> - <datum>.
Diese Rufzeichen-Statistik zeigt an, wann die Statistik zuletzt gelöscht und wann das letzte Byte empfangen worden ist.

Frames: I UI RR REJ RNR SABM/UA DISC/DM FRMR

RX bzw. TX:

<anzahl> I	Information Frame,
<anzahl> UI	Unnumbered Information Frame,
<anzahl> RR	Receive Ready Frame,
<anzahl> REJ	Reject-Frame,
<anzahl> RNR	Receive Not Ready Frame,
<anzahl> SABM/UA	Set Asynchronous Balanced Mode Frame,
<anzahl> DISC/DM	Disconnect Mode/ Disconnect Frame,
<anzahl> FRMR	Frame Reject.

Bytes: Total Info Header Overhead %I %RR %REJ %RNR

RX bzw. TX:

<anzahl> Total	Byte insgesamt,
<anzahl> Info	Info-Byte ,
<anzahl> Header	die benötigten Protokoll-Byte,

Die folgenden % Angaben sind das Verhälnis von:

<anzahl> Overhead	Protokoll-Bytes zu Total Bytes,
<anzahl> %I	I Frames zu Total Frames,
<anzahl> %RR	RR Frames zu Total Frames,
<anzahl> %REJ	REJ Frames zu Total Frames,
<anzahl> %RNR	RNR Frames zu Total Frames.

TX:

Once: <anzahl>	effektiv gesendete Nutzbytes (einmal gesendete I-Bytes),
Repeated: <anzahl>	wiederholt gesendete I-Bytes,
IQual:	Verhältnis in Prozent von einmal gesendeten I-Bytes zu gesamt gesendeten I-Bytes,

Beispiel: Bei 100 Prozent ging kein einziges Byte bei der Übertragung verloren,

TQual:	Verhältnis in Prozent von eff. Nutzbytes zu TX-TotalBytes.

Hier wird der Overheadanteil des AX.25 Protokolls deutlich sowie auch die Unterschiede des TheNetNode Protokolls (siehe Link-Statistik DB0GOE) und FlexNet (siehe Link-Statistik DB0BID)

Die Link – Statistik wurde erweitert um die getrennte Anzeige von Frame vom Nachbar und Frame via Nachbar.

```
Link to DB0BID                  10.10.99 09:38:46 - 16.10.99 16:27:14
Frames:      I        UI        RR       REJ      RNR  SABM/UA  DISC/DM FRMR
RX:      59244         0     53873         9       14     4222      407    0
TX:      49337         0     31976      1622      153     2540     2899    0
Bytes:   Total      Info    Header Overhead      %I     %RR    %REJ    %RNR
RX:    9975854   8019756   1956098    19.6%  50.3%   45.7%    0.0%    0.0%
TX:    3969290   2483254   1486036    37.4%  55.7%   36.1%    1.8%    0.1%
TX:      Once:   2480673 Repeated:     2581 IQual:   99.8%  TQual:   62.4%

Link to * via DB0BID            10.10.99 09:38:46 - 16.10.99 16:27:18
Frames:      I        UI        RR       REJ      RNR  SABM/UA  DISC/DM FRMR
RX:     799684     34986    944581    37922    8172    23545     9870    0
TX:    1238721      2599    726171    31516     973    27056     7021  110
Bytes:   Total      Info    Header Overhead      %I     %RR    %REJ    %RNR
RX:  164043431 115333128  48710303    29.6%  43.0%   50.8%    2.0%    0.4%
TX:  210914029 155435715  55478314    26.3%  60.8%   35.6%    1.5%    0.0%
TX:      Once: 117943369 Repeated: 37492346 IQual:   75.8%  TQual:   55.9%
```

Die Port - Statistik....

```
              Links       RxB      TxB   RxBaud   TxBaud  RxOver TxOver
 0:70cm_1200    6/5      5387k   52340k       64      208     83%     8%
 1:70cm_9600    4/3     27434k  576083k       56      104     42%     1%
 2:23cm_9600    4/1     10524k   60302k      392      248      6%    12%
 3:Lohfelden    5/2     20441k  119742k       48       48     46%     9%
 4:Cluster     23/1     68199k   31129k      128       48     10%    59%
 5:Mailbox     21/1      2011M  588261k     4720     1616      2%    17%
 6:Goettingen   1/1    176951k  424651k      496     1216      8%     2%

Total = 922,987,789 Bytes
```

In der Statistik werden nur die Info-Byte erfasst!

<Portnummer>:<Portname>

Port Nummer und Portname.

Links x/y
x = Anzahl der L2 Verbindungen
y = Anzahl unterschiedlicher Call.
Links wird für die Einstellung der automatischen Parameter gebraucht.

RxB ... TxB
Empfangene ... gesendete Datenmenge in Byte; k = Kilobyte; M = Megabyte; G = Gigabyte.

RxBaud ... TxBaud
Momentane Empfangsseitige ... Sendeseitige Linkbelastung.

RxOver ... TxOver
Prozentuales Verhältnis von Nutzdatenübertragung zu Protokolldatenübertragung.

Total = <anzahl> Bytes.
Summe aller empfangenen und gesendeten Byte.

(**TA**)LK <call>
Schaltet eine TALK-Verbindung zu dem User mit dem <call>, wenn <call> mit dem Kommandointerpreter verbunden ist. Als Antwort wird dieser Text ausgegeben: You are now talking with <call>. Leave this mode with /q. Alle weiteren Eingaben bekommt der User als: Msg from <call>: text..... Dieses ist jedoch nur eine einseitige Verbindung.

(**TA**)LK <call> <text>
Mit TALK <call> <text> kann man einem User eine Textzeile zusenden, wenn dieser mit dem Kommandointerpreter des Knotens verbunden ist. Besteht zu dem User keine Verbindung vom Kommandointerpreter, so wird die Textzeile bis zu einem Reconnect aufgehoben. Wird die Verbindung durch einen Disconnect vom User aufgelöst, wird die Textzeile .....fachgerecht und gebührenfrei entsorgt.

(**TI**)ME
Gibt das Systemdatum und die Uhrzeit aus.

(**U**)SER
Nach der Eingabe von U (für USER) erscheint z.B.:

```
KS:DB0EAM> TheNetNode (GO32) V1.78
Uplink(DK3NZ-5)                     <-->  Downlink(DK3NZ-5 DB0EAM-3)
 Uplink Northeim via DB0NHM
Circuit(LOH:DB0VFK DL2FAI-1)        <..>  Downlink(DL2FAI-1 DB0EAM-3)
Uplink(DB0OVA)                      <-->  Downlink(DB0OVA DB0EAM-4)
 Uplink Knuell via DB0KH DB0SHI
Uplink(DL8FAJ)                      <-->  Downlink(DL8FAJ DB0EAM-4)
Uplink(DD1SI)                       <-->  Downlink(DD1SI DK5ZK)
 Uplink Knuell via DB0KH DB0SHI DB0DQ
 Downlink 9600_Bit/s
Convers(DB0EAM-1) Host              <-->  Circuit(GOE7:DB0GOE DB0EAM-1)
Host(DB0EAM)
```

KS Ist der „ALIAS" des Knotens, eine Art Pseudonym. Zum einen soll dieser ALIAS eine geographische Information bieten, die einfacheres Zuordnen unbekannter Calls zu einem Gebiet ermöglicht. Zum anderen ermöglicht, im Gegensatz zum IDENT (Rufzeichen), der ALIAS ein mehrfaches Connecten eines Knotens, denn man kann z.B.:

KS, KS-1, KS-2,

etc. gleichzeitig connecten, aber nur einmal DB0EAM.

Die andere Möglichkeit, Multiconnect an einem TheNet-Knoten durchzuführen, ist die Benutzung des eigenen Rufzeichens mit verschiedenen SSID gleichzeitig. Das kann derzeit aber nicht jede Software. Als ALIAS wird in DL häufig des Autokennzeichens des Haupteinzugsbereiches verwendet, kurze Ortsnamen können auch vorteilhaft ausgeschrieben werden, wodurch der Erkennungswert weiter zunimmt. Andere Alternativen wie z.B. Postleitzahlen oder WW-Locatoren sind nicht sehr einprägsam.

Die Verwendung des ALIAS, statt des Rufzeichens, ist K E I N Rufzeichenmissbrauch, da der ALIAS ja gar keinen den internationalen Normen für Amateurfunkrufzeichen entsprechenden Aufbau hat (haben sollte...). Genausowenig wie ja auch ein Ruf an CQ oder Test kein Rufzeichenmissbrauch ist oder das 4-Zeichen Call bei Amtor. Die geforderte Rufzeichen-Nennung alle 10 Minuten wird vom Netzknoten mit einer Bakenaussendung an ID ausgeführt, in der auch der verwendete ALIAS mitgeteilt wird. Ausserdem bleibt die Verwendung des ALIAS im AX.25-Adressfeld auf Endanwender begrenzt, zwischen den Knoten selbst werden die offiziellen IDENT (Rufzeichen) benutzt. Auch lässt sich jeder TheNet-Knoten anhand des INFO-, USER- und NODES-Befehles identifizieren. Eine einheitliche Verwendung von Autokennzeichen in Knotenlisten erhöht die Transparenz der Netze wesentlich.

DB0EAM
: Ist das offizielle IDENT (Rufzeichen) des TheNet-Knotens.

Uplink <call>
: Zeigt an, dass der Benutzer mit dem Rufzeichen <call> hier am Knoten in das Netz eingestiegen ist. Etwa vorhandene Digipeater werden in der nächsten Zeile angezeigt.

Downlink <call>
: Zeigt an, dass der Benutzer hier mit dem Rufzeichen <call> das Netz verlässt. Etwa vorhandene Digipeater werden in der nächsten Zeile angezeigt.

Circuit <alias:ident>
: Auf der linken Seite bedeutet, dass der Benutzer von dem Knoten mit dem <id> aus diesen Knoten connected hat und dort mit dem Rufzeichen <call> connected ist.

Circuit <alias:ident>
: Auf der rechten Seite bedeutet, dass die Verbindung zu dem Knoten mit dem <id> geschaltet ist und der Benutzer dort das Rufzeichen <call> führt.

Host
: Bedeutet eine Verbindung zum Bedienerterminal des Knotens oder auch zu einer direkt am Knoten angeschlossenen Mailbox.

<-->
: Zeigt eine bestehende Verbindung an.

<..>
: Zeigt eine im Aufbau befindliche Verbindung an.

: Ein Eintrag ohne „rechte Seite" bedeutet, dass die Verbindung hier zur Zeit endet und der Benutzer mit dem Befehlsinterpreter des Knotens verbunden ist.

Uplink @ DB0FD
: Zeigt den Einstiegsknoten des Users an, sofern dieser nicht aus der Circuit-Zeile bereits hervorgeht. Dieses ist immer dann der Fall, wenn mehrere Level4-Verbindungen (Circuits) nacheinander aufgebaut werden.

Downlink Port_5 via DB0XYZ oder Uplink Port_5 via DB0XYZ
: Zeigt den Downlink-Weg an, den der User genommen hat. >Port_5< steht in diesem Fall für Port-Namen des entsprechenden Port (siehe (**PO**)rt Befehl) und ist ein Level2-Link. Ein Uplink- oder Downlink-Port wird nur angezeigt, wenn es sich dabei nicht um einen User-Port handelt, auf dem eine MHEARD-Liste geführt wird.

(**U**)ser (**H**)ost

Wie (U)ser, zeigt jedoch nur die Verbindungen zum Host an. Wenn das Host - Interface zum Anschuss einer Box oder Cluster benutzt wird zusätzlich diese Liste ausgegeben.

```
 CH Call      F     NT    RX    TX    ST     RxB     TxB    Baud   ConTime
 ------------------------------------------------------------------------
 03 DB0IL-3   CD  7199     1     0     0      65    1494     120   0:02:05
 05 DB0IL-5   C   7086     0     0     0      51     595      56   0:02:05
 06 OZ5BBS-1  C   7068     0     0     0    7620    1598      96   0:09:37
  /  /        /     /     /     /     /       /       /       /         /
 1) 2)       3)     4)    5)    6)    7)      8)      9)      10)       11)
```

1) Hostmode-Kanal
2) Rufzeichen des Users (Connect vom User) oder eigenes Call (Connect vom Host zum Knoten)
3) Hostmode-Flags
   C = Connected
   D = Disconnecten wenn Info übertragen
4) Noactivity-Timer (in Sekunden)
5) empfangene Frames in Warteschlange
6) zu sendende Frames in Warteschlange
7) Statusmeldungen in Warteschlange
8) Anzahl empfangender Bytes seit Bestehen des Links
9) Anzahl gesendeten Bytes seit Bestehen des Links
10) Aus 8) + 9) errechnete effektive Baudrate für diesen Link
11) Connectzeit

(**U**)ser +

Zeigt in Tabellenform die Level2-Verbindungen an.....

```
L2 - User:
PO SrcCall   DstCall  LS  Rx Tx Tr SRTT   RxB     TxB  Baud  ConTime  Pr Da
-------------------------------------------------------------------------
 1 DB0EAM    DB0GOE   IXF  0  9  0  79 109928k 685912k 1413  38:59:23  -
 3 DG9ABF-1  DB0EAM-3 IXF  0  0  0 755   4583       9   253   0:02:25  -
 3 DL1DAQ-1  DB0EAM-3 DBS 10  0  0 674 259713     188   200   2:52:29  -
 4 DB0RDX-1  DB0EAM-4 DBS  8  0  0 172  50776    6325    15   8:06:15  -
 5 DB0EAM    DB0AX    IXF  0  0  0  62 273953k 169538k 1330  38:59:17  -
 6 DB0EAM    DG9XYZ   IXF  0  4  0 370 451216  189109   377   3:46:25  0  1
 6 DB0EAM    DG9XYZ-1 IXF  0  0  0  96 104530    2955    33   7:11:00 20  1
/    /        /     /   /  /  /   /     /        /     /      /    /   /
1)  2)       3)     4)  5) 6) 7)  8)    9)       10)   11)    12)  13) 14
```

1) Benutzter Port zur einfacheren Übersicht.
2) Quellrufzeichen des L2-QSOs.
3) Zielrufzeichen des L2-QSOs.
4) L2-Link-Status:
   SET = Link-Setup,
   FMR = Frame Reject,
   DRQ = Disconnect Request,
   HTH = Wartet auf Connect eines Digipeating-Zieles,
   IXF = Info Transfer,
   REJ = REJ Send,
   WAK = Waiting Acknowledge,
   DBS = Device Busy,
   RBS = Remote Busy,
   BBS = Both Busy,
   WDB = Waiting Ack and Device Busy,
   WRB = Waiting Ack and Remote Busy,
   WBB = Waiting Ack and Both Busy,
   RDB = REJ Send and Device Busy,
   RRB = REJ Send and Remote Busy,
   RBB = REJ Send and Both Busy.
5) Anzahl der empfangenen Frames in der Warteschlange für diesen Link.
6) Anzahl der noch zu sendenden Frames in der Warteschlange für diesen Link
   Wenn ein Link zu einem Nachbarn zusammenbricht, wird dieser bei mehr als 100 ausstehenden Paketen im L2 einfach abgebrochen.
7) Anzahl Retries.
8) Stand des 'Smoothed Round Trip Timers'(in mal 10 ms).
9) Anzahl empfangender Bytes seit Bestehen des Links.
10) Anzahl gesendeter Bytes seit Bestehen des Links.
11) Aktuelle effektive Baudrate für diesen Link.
12) Connectzeit in HH:MM:SS und bei über 23:59:59 Stunden dann nur noch als TTT/HH:MM.

13) Bei DAMA-Netzeinstiegen:
Aktuelle Priorität des User (0: = höchste Priorität).
14) Port wird unter dem DAMA-Master entsprechend der Nr. verwaltet.

.....und auch die Level4-Verbindungen.

```
L4 - User:

Call      Node       S  Rx  Tx Tr Win SRTT     RxB     TxB    Baud   ConTime
-------------------------------------------------------------------------------
DL3EAG    DB0II      I   0   0  0 10  7698       0   12504      48   2:33:55
DB0EAM-1  DB0LIP     I.  0   0  0 10   914   68693    1400k    152   3/14:24
DB0KG     DB0GOE    .I   0   0  0 10   998     257   15781     456   0:02:56
DJ9KG     DB0GOE     I   0   0  0 10  7161      47   20079      48   2:39:36
/           /       /    /   /  / /    /        /      /        /      /
 1)        2)     3)    4)  5) 6) 7)  8)       9)     10)      11)     12)
```

1) Rufzeichen des User.
2) Call des Knotens mit dem der User verbunden ist.
3) L4-Circuit-Status.
S = Circuit-Setup,
.I = Eigener Knoten steht im Choke (Datenübertragung angehalten),
I. = Der Zielknoten steht im Choke,
I = Normaler Info-Transfer,
D = Disconnect-Request.
4) Anzahl der empfangenen Frames in der Warteschlange für diesen Circuit.
5) Anzahl der noch zu sendenden Frames in der Warteschlange für diesen Circuit.
6) Anzahl Transport-Retries.
7) Transport Fenstergrösse.
8) L4 SRTT.
9) Anzahl empfangender Bytes seit Bestehen des Circuits.
10) Anzahl gesendeter Bytes seit Bestehen des Circuits.
11) Aktuelle effektive Baudrate für diesen Circuit.
12) Connectzeit in HH:MM:SS und bei über 23:59:59 Stunden dann nur noch als TTT/HH:MM.

Zur Messung des L4SRTT:

Durch den Aufbau des NET/ROM Layer 4 kann man Grundsätzlich nur in Senderichtung den SRTT messen, d.h. immer wenn wir Info senden, messen wir die Laufzeit. Von dem L4SRTT wird hauptsächlich das Acknowledge - Timeout und das Requery - Timeout abgeleitet.

Acknowledge - Timeout:

Der L4 kann Daten Senden, bis ein so genanntes Sendefenster voll ist. Dieses umfasst in der Regel 10 Frames. Das Sendefenster wird beim Verbindungsaufbau zwischen den Partnern vereinbart und ist auf beiden Seiten gleich. Danach wird auch das Acknowledge- (Bestätigungs-) verhalten ausgerichtet. Sobald das Empfangsfenster halb voll ist, wird sofort eine Bestätigung gesendet, spätestens aber nach der einfachen L4SRTT-Zeit. Damit wird einerseits ein ständiger Fluss sichergestellt, andererseits werden unnötige Bestätigungen vermieden.

Requery - Timeout:

„Requery“ ist hier eigentlich nicht richtig, denn NET/ROM hat keine Kommandos in Empfangsrichtung. Dies bedeutet, dass nach einer bestimmten Zeit das Frame vom Sender wiederholt werden muss, der Empfänger kann nicht mitteilen, dass er das Frame nicht bekommen hat. Man spricht von Requery-by-Timeout, man könnte auch sagen Retransmission-Timeout. Dieser Timeout wird wieder vom L4SRTT abgeleitet, mit Faktor 3.

Was sagt uns das?

Ein sehr hoher SRTT-Wert (sagen wir 100s) bedeutet nicht, dass diese Verbindung wirklich so lahm ist, sondern einfach das dort wenig los ist, und beide Seiten sehr selten ein ACK schicken. Da wir sowieso beim halben Fenster ein ACK schicken, kann die Verbindung jederzeit schnellen Datentransfer aufnehmen. Andererseits ist es richtig, dass solche Links mit sehr langen Requery - Timeout belegt werden, schliesslich könnte es sein, das der Partner sich einfach lange Zeit lässt für die Bestätigung (hoher L4SRTT). Die hier vorgestellten Abläufe sind TCP/IP entliehen und den besonderen Anforderungen von NET/ROM angepasst. Da TCP/IP mit Antwortzeiten im Millisekunden - Bereich arbeitet, wir aber bei einigen Sekunden, wird sich erst zeigen, ob das alles so richtig hinhaut. Durch Begrenzungen nach unten und oben ist im Programm sichergestellt, das nicht allzuschiefe „automatische“ Timerwerte generiert werden können.

(**U**)ser (**L**)inks
Zeigt in Tabellenform NUR die Level2-Verbindungen an.

(**U**)ser (**C**)ircuit
Zeigt in Tabellenform NUR die Level4- (Circuit -) Verbindungen an.

(**U**)ser <Port Nr.>
Zeigt in Tabellenform NUR die Level2-Verbindungen des angegebenen Ports an.

(U)ser <call>
Zeigt alle Informationen zu einem bestimmten User oder einer Gruppe von Usern an. Dabei sind die folgenden Abfragen möglich:
u db7*;
u db7kg* zeigt aber auch db7kga, db7kgb, db7kgc usw. aber auch alle SSID von DB7KG;
u db7kg zeigt db7kg-0 an;
u db7kg-15 zeigt db7kg-15 an;
u db7kg-* geht NICHT

```
L2 - User:

Po SrcCall   DstCall   LS  Rx Tx Tr SRTT   RxB    TxB   Baud   ConTime Pri
-------------------------------------------------------------------------
 2 KS-5      DG9FU-5   IXF  0  0  0   95  2624  96563     96   2:13:33   -
 2 KS-6      DG9FU     IXF  0  0  0   85   176   5367     72   1:07:41   -

Uplink(DG9FU-5)                    <-->  Convers(DG9FU-5) Ch 170
Uplink 23cm_9600
Uplink(DG9FU)
Uplink 23cm_9600

DG9FU de DB0EAM (16:52) >
```

(**V**)ERSION
Gibt die aktuelle Versionsnummer der Software aus.

(**V**)ERSION * oder (**V**)ERSION +
Zeigt die Version und die Compiler Switches an.

```
TheNetNode (Linux), Version 1.79 (Oct 14 2002)
     Copyright by NORD><LINK, free for non-commercial usage.
         See www.nordlink.org for further information.
  This version compiled for 16 Ports, 400 L2-Links and 200 Circuits.

Compiled options:
   * IP-Router
   * PACSAT Server
   * Local-Hosting
   * Graph
   * User-Password
   * User-MaxConnection
   * DAMA-Slave
   * Kernel-Interface
   * Command-Aliasing
   * Extended-AX.25

Hardware:
   * KISS-Protocols: TOKENRING KISS SMACK RKISS TF
   * VANESSA
   * IPX
   * AX25IP
   * Kernel-AX.25
   * IP-Tunnel
```

Zurzeit sind folgende Compiled options möglich:

IP-router	IP-Router eingebunden,
PACSAT	PacSat Server eingebunden,
SYSPORT	Port Mode „Sysop" ist freigegeben,
Local-hosting	FlexNet-Ziele werden weitergeleitet als Nodes,
Graph	Graphische Darstellung der Statistik,
User-Password	Auch für USER kann ein Passwort eingerichtet werden.
Command-Aliasing	ALIAS-Kommando eincompiliert
Extended AX.25	Extended-AX.25 eincompiliert
User-MaxConnection	Möglichkeit den USER auf eine maximal Connectzahl zu beschränken
DAMA-Slave	DAMA-Slave Modus eincompiliert
Kernel-Interface	Linux Kernel-Interface Unterstützung mit eingebunden

Und an Hardware wird unterstützt:

LOOPBACK
KISS-Protocols: TOKENRING KISS SMACK RKISS TF
6PACK
VANESSA
EXTDEV
SCC
AXIPX
AX25IP
Kernel-AX.25
IP-Tunnel

EXTERNE Befehle für alle User

Im Prinzip können die Filenamen frei gewählt werden. Textnamen und Programmnamen müssen sich jedoch MEHR als durch die Extension unterscheiden. Ein Programm MSG.EXE und ein Textfile MSG.TXT sind nicht möglich, es wird in diesem Fall nur das EXE-File gefunden.

Unter externe Befehle fallen alle Texte, die sich im PATH TEXTCMD befinden. Durch das Editieren der CTEXT.*** können den Usern weitere Texte angeboten werden. Zur Vereinheitlichung sollten bitte die folgenden Bezeichnungen beibehalten werden:

(A)KTUELL (externer Befehl)
Gibt aktuelle Informationen aus.

(HA)RDWARE (externer Befehl)
Gibt eine Hardwarebeschreibung des Knotens aus.

(I)NFO (externer Befehl)
Ausgabe eines Info-Textes.

(MAP) (externer Befehl)
Zeigt eine kleine Karte der direkten Umgebung des Knotens.

(SO)FTWARE (externer Befehl)
Gibt eine Softwarebeschreibung aus.

EXTERNE Programme für alle User

Hier stehen im Moment die Programme:

Digimail:

(**MSG**) (externer Befehl)

(**MSG**) S <ZIELCALL/GRUPPE> #<lifetime> <text>
Zum Erstellen einer Digimail:

Der MSG-Befehl ermöglicht es, einem Benutzer des Knotens eine kurze (!) Nachricht in den Connect-Text zu schreiben. Beim nächsten Connect wird diese Zeile dann bei ihm im CTEXT erscheinen. Statt des Zielcalls kann auch eine Zielgruppe angegeben werden. Diese ist eine Art Verteilerliste. Jedes Mitglied der Zielgruppe bekommt eine Kopie der Nachricht.

Die Lifetime kann, muss aber nicht, angegeben werden. Sie kann von #1 = einem Tag bis #30 = dreissig Tagen angegeben werden. Wird keine Lifetime angegeben, so wird sie auf default = 7 Tage gesetzt. Das Herabsetzen der Lifetime geschieht mittels MSY C. Siehe „Externe Programme für den SYSOP".

Abhängig davon, ob das System mit einer Festplatte oder einer RAMDISK arbeitet, kann die Nachricht bei Absturz oder Reset verloren gehen.

Beispiel: MSG S DL9GYA #10 Roland, bitte connecte mich, wenn Du zurück bist!

MSG S SYSOP Link nach ..... defekt?!

Letzteres würde die Nachricht an DL1KWS und DL9GYA weiterleiten. Die Definition einer neuen Gruppe geschieht nur durch den Sysop. Bitte danach fragen.

(**MSG**) R
Liest die eigenen Nachrichten aus.

(**MSG**) R <call>
Liest die an <call> gerichteten Nachrichten aus.

(**MSG**) R <call> 1 oder 1-3
Zusätzlich kann noch eine Numerierung mit angegeben werden. Also Lesen der Nachrichten 1 oder 1-3.

(**MSG**) L
Listet die eigenen Header auf.

(**MSG**) E
Der MSG E Befehl löscht alle eigenen Digimail-Nachrichten im eigenen CTEXT.

(**MSG**) G
Zeigt alle vorhandenen Verteilergruppen mit Call an.

(**MSG**) G <gruppe>
Zeigt nur die Call in dieser <gruppe> an.

Neue Gruppen können nur vom Sysop angelegt werden!

(**MSG**) V
Gibt die Versionsnummer und - Datum aus.

Locatorberechnung:

(**QTH**) (externer Befehl)

(**QTH**) <locator> oder (**QTH**) <locator> <locator>
QTH berechnet Entfernung und Richtung zwischen zwei QTH-Kennern. Wird nur ein QTH-Kenner angegeben, so wird die Entfernung zum Knoten berechnet! Ausserdem ermittelt QTH alle gültigen Angaben für die Standorte, z.B. zur Ermittlung der neuen weltweiten QTH-Kenner aus der geografischen Koordinate. QTH akzeptiert sowohl den neuen weltweiten als auch den alten QTH-Kenner. Ausserdem können Längen- und Breitengrade in Dezimalform oder auch in Grad:Minuten:Sekunden angegeben werden.

(**QTH**) <locator>
Berechnung der Entfernung und Richtung zwischen dem Standort des Knotens und <QTH>.

(**QTH**) <locator1> <locator2>
Berechnung der Entfernung und Richtung zwischen <locator1> und <locator>.

Gültige Eingabeformate:
Alter QTH-Kenner : FM04C oder FM04C/2 (Ohne Angabe gilt Feldraster /2)
Neuer QTH-Kenner : JO52JW

Geogr. Koordinate: GGG:MM:SS/GG:MM:SS z.B. 10:47:30/52:56:15
(östl.Länge/nördl.Breite in Grad:Min:Sek)

Geogr. Koordinate: GGG.GGG/GG.GGG z.B. 10.792/52.937
(östl.Länge/nördl.Breite in Dezimalform)

Es ist zu beachten, dass innerhalb einer geografischen Koordinate nur in Grad:Minuten:Sekunden oder mit Gleitkommazahlen (Realzahlen) gearbeitet werden kann. Ein Mischen beider Formate ist unzulässig.

SATELLITEN-Standort-Berechnung:

(**SAT**) (externer Befehl)
Zeigt die derzeit 40 möglichen Satelliten sowie eine kurze Hilfe an.

(**SAT**) <**nr**>
Berechnet die Satellitenposition zum Zeitpunkt der Abfrage, bezogen auf den Standort des Knotens. Ausgegeben werden nicht nur die Positionsdaten, sondern auch die nötigen Antennen-Einstellungen.

(**SAT**) <**nr**> <**locator**> oder (**SAT**) <**nr**> <**koordinaten**>
Wie zuvor, jedoch wird nicht der Locator des Knotens, sondern der angegebene verwendet.

Gültiges Eingabeformat:
Geogr. Koordinate: GGG.GGG GG.GGG z.B. 10.792 52.937
(östl.Länge/nördl.Breite in Dezimalform)

User-Statistik:

(**TOP**) (externer Befehl)
Dieser Aufruf wertet die MHEARD.TAB aus und erzeugt eine „Hitliste". Die Auswertung ist nicht immer auf dem neuesten Stand. Die Datei MHEARD.TAB wird in Abhängigkeit des Parameters 3 aktualisiert. Der aktuelle Stand kann an der Uhrzeit im Kopf der Tabelle abgelesen werden.

(**TOP**) <**anzahl**> *****
Gibt eine Liste mit der Länge bis maximal <anzahl> aus. Mit der Option '*****' werden auch Calls ausgegeben, die nix übertragen haben.

(**TOP**) -<**port**> *****
Gibt eine Liste von 10 Calls, die auf dem <port> bekannt sind, aus.

(**TOP**) <**anzahl**> -<**port**> *****
Gibt eine Liste mit der Länge bis maximal <anzahl>, die auf diesem <port> bekannt sind, aus.

(**TOP**) <**call**>
Gibt nur dieses Rufzeichen, aber mit den verschiedenen registrierten SSID aus.

(TOP) <ca*>
Z.B.: TOP DG8B* . Gibt alle Rufzeichen aus, die mit DG8B anfangen.

(**TOP**) **-L3**
Wertet statt der MHEARD.TAB die L3HEARD.TAB aus. In der werden die Links gespeichert. Der Parameter -l3 kann mit -port und anzahl verknüpft werden.

(TOP) -h
Gibt eine kurze Hilfe aus.

(**TOP**) **-v**
Zeigt den Versionsstand an.

```
      MHEARD-Auswertung vom 01.10.02 00:00 UTC bis 15.10.02 20:39 UTC

Nr. Call   Port              Rx         Tx      Summe     % RxRej TxRej Dama
  1 DG8BR  70cm          5464816    2015963    7480779 34.76     0    20    0
  2 DL1BMF 70cm          4153277     774068    4927345 22.89     0   289    1 #
  3 DL1BH  70cm          2454386      61482    2515868 11.69     0   269    0
  4 DG5BD  70cm          1670080      42713    1712793  7.95     0     2    0
  5 DL1BJG 70cm           911249      38245     949494  4.41     0    10    0
  6 DL8BED 70cm           563204     123884     687088  3.19     0     0    0
  7 DL2NO  70cm           436010      71586     507596  2.35     0     8    0
  .  Hat Call empfangen -------!          !          !     !     !     !    ! !
  .  Hat Call gesendet --------------------!          !     !     !     !    ! !
  .  Hat Call über den Digi übertragen ---------------!     !     !     !    ! !
  .  Hat Call Anteil an Digiauslastung ---------------------!     !     !    ! !
  .  Hat Call gesendet, weil schlecht gehört ----------------------!     !    ! !
  .  Hat Call empfangen, weil Digi schlecht gehört ---------------!     !      !
  .  Hat Call gegen DAMA verstossen ----------------------------------!      !
  .      Call ist   # = mit unterschiedlichen SSID auf mehreren Ports QRV ----!

 22 DG4BRT 70cm            19115       3067      22182  0.10     0    12    0
 23 DL3BCU 70cm            16536       2865      19401  0.09     0     3    0
 24 DL6BBW 70cm             1755        207       1962  0.00     0     0    0
 25 DL2BKH 70cm               30         60         90  0.00     0     0    0

Diese User haben auf allen Ports 21.518.562 Bytes = 100.00 % uebertragen.
Insgesamt wurden 21.518.562 Bytes von 25 Benutzer bewegt.
```

Falls sich jemand über die Differenz zwischen Plätzen und Usern wundert: Sie entsteht wenn ein Benutzer auf mehreren Ports aktiv ist. Z.B.: 70cm und 23cm. Oder so... Tatsache ist, es wird das Rufzeichen nur einmal gezählt.

Beispiel für eine Einzelausgabe:

```
Datum          Port              Rx        Tx RxRej TxRej   D Call

15.10.02 17:00 70cm          154961     26042     0     0   0 DG8BR-11
15.10.02 18:14 70cm         3607302    120383     0     0   0 DG8BR-10
15.10.02 20:38 70cm          112773     47030     0     0   0 DG8BR-1
15.10.02 20:39 70cm         1589780   1822508     0    20   0 DG8BR

Stand: 15.10.02 20:39 UTC
```

Die Liste erklärt sich wohl von selbst.

Unter D stehen die DAMA-Verstösse.

Visitenkarte:

(**SAV**)ecall (externer Befehl)

(**SAV**)ecall <feldkenner> <text>

Die Felder können mit (**SAV**)ecall ausgefüllt oder geändert werden.

```
SAVECALL /N .... . . . . . ..... 30 Zeichen für den Namen
SAVECALL /Q .... . . . . . ..... 30 Zeichen des Wohnortes
SAVECALL /L ......               6 Zeichen des World-Locators
SAVECALL /D .......              7 Zeichen für Eingabe des DOK
SAVECALL /V .........            9 Zeichen QRV auf ... Digi
SAVECALL /M ................... 25 Zeichen Mybbs der Mailbox
SAVECALL /T .... . . . . . ..... 40 Zeichen für freien Text
            ( ein * als Text löscht diesen Eintrag )
SAVECALL /*     ==> Vorsicht !! löscht den ganzen Eintrag !
```

DOK Bitte so eintragen: F73 oder F73/Z25
.....sonst klappt es mit dem Suchen nachher nicht.

(**SH**)owcall (externer Befehl)

(**SH**)owcall <call>

Ist ein kleines Programm, welches die Visitenkarten der User, die sie sich selbst SAVECALL erzeugt haben, ausliest und anzeigt. Muster:

```
== ShowCall ================================== 01.09.95 ===
=  Call   :  ......     Name : ............................ =
=  Loc    :  ......     QTH  : ............................ =
=  QRV on :  ......... MyBBS : ...................          =
=  Dok    :  .......  Update : JJMMTTHH CALL                =
=                                                           =
=  Free MSG    :  ....................................    =
============================================== de DG3AAH ==
```

Die Liste lebt von der Beteiligung der User. Deshalb darf nur der darin lesen und suchen der selbst zumindest die Felder Name, Locator und QTH eingetragen hat. Das <call> kann auch in Verbindung den Platzhaltern „*" und „?" aufgerufen werden. Dabei steht „*" für beliebig viele Zeichen und „?" steht für genau 1 Zeichen. Es wird eine Liste der Calls aus, bestehend aus :
- Call, Name, Wohnort und Locator.
Die ersten 3 Zeichen müssen jedoch angegeben werden, damit nicht unendliche Listen ausgelesen werden.

(**SH**)owcall W
Zeigt die Anzahl der Einträge an.

Onlinehilfe:

(**H**)elp (externer Befehl)
Ersetzt die alte Hilfefunktion 'H'. Es wird nur noch in einer Datei nach dem eingegebenen Befehl gesucht. Zeigt alle möglichen Befehle an.

(**H**)elp <befehl>
Zeigt die erste Seite (20 Zeilen) der Hilfe zum <befehl> an.

(**H**)elp <befehl> 2
Zeigt die zweite Seite der Hilfe zum <befehl> an.

(**H**)elp <befehl> *
Zeigt alle Seiten der Hilfe zum <befehl> an.

ANHANG A:

Die verschiedenen Textdateien und ihre Bedeutung

Um die Gestaltung der Info - oder Hilfstexte möglichst einfach zu handhaben, haben wir den Weg gewählt, diese Texte als einzelne Dateien auf Diskette, RAM-Disk oder Hard-Disk abzulegen. Jeder einzelne Text kann vom SYSOP mit dem EDIT-Befehl editiert und mit dem READ-Befehl gelesen werden. Ausserdem sind fast alle Texte auch anderen Benutzern über die einzelnen Kommandos zugänglich.

Folgende Texte sollten vorhanden sein:

AKTUELL.TXT (ins SUB-DIR: TEXTCMD)
: Aktuelle Informationen zum Knoten, Veranstaltungen, usw.
(Alte AKTUELL.TXT dürfen ruhig auch mal entfernt werden.)

CTEXT.TXT (ins TNN-Hauptverzeichnis)
: Connect-Text (wird nur ausgesendet, wenn freigegeben).
Im CTEXT.TXT kann der Hinweis auf einen vorhandenen AKTUELL.TXT vorhanden sein. Soll dessen Datum automatisch mit angegeben werden, so ist dieses mit einem %A auszugeben. Beispiel für einen CTEXT:
(H)elp (A)ktuell (vom %A)

CTEXT.n (ins TNN-Hauptverzeichnis)
: Portabhängiger Ctext. CTEXT.n wird zusätzlich zum CTEXT.TXT auf dem jeweiligen Port gesendet. Also wird CTEXT.3 nach dem CTEXT.TXT auf Port 3 gesendet.

HARDWARE.TXT (ins SUB-DIR: TEXTCMD)
: Informationen zum Knotenrechner, usw. (HARDWARE-Befehl).

INFO.TXT (ins SUB-DIR: TEXTCMD)
: Informationen über Linkstrecken, usw. (INFO-Befehl).

MAP.TXT (ins SUB-DIR: TEXTCMD)
: Kleine Karte der Nachbarknoten (MAP-Befehl).

QUIT.TXT (ins TNN-Hauptverzeichnis)
: Wird beim QUIT-Befehl ausgegeben, falls vorhanden.

SOFTWARE.TXT (ins SUB-DIR: TEXTCMD)
: Informationen zur Software (SOFTWARE-Befehl).

SUSPEND.TXT (ins TNN-Hauptverzeichnis)
: Siehe dazu beim Suspend Befehl!

Eine Übersicht über alle vorhandenen Texte bekommt man eventuell mit dem „HELP INDEX“-Befehl. Dieser Befehl listet alle vorhandenen Texte in der Reihenfolge auf, in der sie auf Diskette stehen. Falls die (externe) Onlinehilfe installiert ist, steht der Befehl „HELP INDEX” nicht zur Verfügung.

ANHANG B:

(TNC2c)

TNC2c und TheNetNode

Taktrate des SIO-SyncB-Taktes

Wenn der TNC mit 2.4576 MHz betrieben wird, dann unbedingt die Brücke BR1 durchkratzen und SyncB (SIO Pin 29) an Pin 1 des 74HC4060 legen. Diese Änderung ist auch in der TNC2c-Anleitung beschrieben. Soll der TNC mit 4.9152 MHz betrieben werden, dann sind folgende Änderungen erforderlich:

Quarz tauschen (... klar)

Brücke BR1 durchkratzen und SyncB (SIO Pin 29) - das ist der mittlere Anschluss von BR1 - an Pin 2 (Pin 2!!) des HC4060 legen. Der Pin 2 ist nicht an BR1 herangeführt, aber die Änderung ist notwendig, damit der SyncB-Takt 600 Hz beträgt.

Das EPROM muss eine Zugriffszeit von höchstens 200ns haben!

Soll der TNC mit 9600Bd auf der HDLC-Seite betrieben werden, so ist mindestens eine Taktfrequenz von 4.9152 MHz notwendig, besser sind 9.8304 MHz (Bei Vollduplex sind 9.8304 MHz zwingend erforderlich). Für den Umbau auf 9.8304 MHz werden ein paar Bauteile benötigt, die nicht jeder in der Bastelkiste liegen hat. Für einen Anwender-TNC mögen ausgesuchte Standardbausteine ausreichen, die jenseits ihrer Spezifikation betrieben werden. Der Netzknoten soll jedoch im Dauerbetrieb laufen, meist an einem nicht beliebig schnell erreichbaren Standort. Deshalb sollte man kein Risiko eingehen und die Bauteile nur innerhalb ihrer Spezifikation betreiben. Welche Art von Programmabstürzen oder auch nur Programmungenauigkeiten bei nicht spezifiziertem Betrieb auftreten, ist nicht vorherzusehen. So war z.B. bei einem TheNet-Netzknoten die Laufzeitermittlung falsch.

Für die CPU und die SIO sind 10 MHz-Bausteine zu verwenden (z.B. Zilog Z84C0010 und Z84C4010).

Das RAM benötigt eine Zugriffszeit von 80ns oder weniger.

Das EPROM benötigt eine Zugriffszeit von 100ns oder weniger.

Die Bausteine 74HC132 und 74HC139 sind unbedingt gegen 74AC132 und 74AC139 zu tauschen!

Vom 74AC139 (5) kann, wenn der TNC ohne AKKU betrieben wird, eine direkte Verbindung zum 62256 (20) hergestellt werden. Dadurch entfällt die Laufzeit über die beiden Gatter des 74HC132.

Die Brücke BR1 wird so verbunden, dass der SyncB-Anschluss der SIO (Pin 29) mit dem 74HC4060 Pin 3 verbunden ist.

KISS-Software für den TNC im Tokenring und Watchdog:

Aktuell ist das auf der Diskette befindliche KISS17.BIN.
Es enthält den EPROM-Inhalt für die TNC.

Adr.	Zelleninhalt	Beschreibung (Alle Werte in HEX):
5C	**00..08..0F**	**Ringadresse je nach Prg. 8 ... 16 Kanäle,**
5D	00..FF	TXDelay in 10 ms Schritten HEX 14 = 200 ms,
5E	00..FF	PERSIST,
5F	00..FF	SLOTTIME,
60	00..FF	TXTAIL,
61	00..01	00 = Simplex; 01 = Vollduplex,
62	00..FF	PTT Haltezeit (Low-Byte),
63	00..FF	PTT Haltezeit (High-Byte),
64	**00..01**	**00 = DAMA aus; 01 = DAMA ein.**

Die Adressen 5C und 64 **müssen** beim Brennen des EPROM angepasst werden. Die Adresse 62 und 63 ist bei einem VOLLDUPLEX-Link relevant. Die anderen Adressen werden von TNN aus eingestellt und dienen hier nur der vollständigen Information.

Adr. 5C: Jeder TNC im Ring muss seine eigene Adresse haben, die im EPROM stehen muss.

Adr. 5D..61: Diese Speicherzellen werden im RAM von TNN aus nach einem Neustart des TNC gemäss den eingestellten Parametern geändert.

Adr. 62 + 63: Die PTT Haltezeit ist nur bei der Einstellung VOLLDUPLEX aktiv. Ist der TX inaktiv, so wird er mit dem eingestellten TX-DELAY getastet. Nach Ende der Tastung läuft nun dieser Timer ab. Ist ein weiteres Paket in dieser Zeit zu senden, so wird es mit TX-DELAY = 0 ms ausgesendet und der Timer neu gesetzt. Hierdurch wird viel Sendezeit eingespart, die nun der Datenübertragung zur Verfügung steht. Soll diese Funktion ausgenutzt werden, so muss noch der 22 µF Kondensator am PIN 11 (SIO RTS NOT) des Modem-Disconnect auf der TNC2c Platte überbrückt werden. Die „Wachhund-Schaltung" für die Sendertastung ist dann natürlich außer Betrieb.

Zu beachten ist, dass der TNC2 immer auf 32 KByte RAM aufgerüstet sein muss, und es sollten für 9600 Baud-Vollduplex-Stecken nur solche TNC eingesetzt werden, die mit einem CPU- Takt von > 6 MHz arbeiten!

Siehe hierzu eine Umbauanleitung unter ANHANG B.

Beispiel: Adr. 62 = Hex 50 ; Adr. 63 = Hex 46

Gesamt: Hex 4650 = 18000 ms = 3 Minuten Haltezeit.

Adr. 64: Ist das DAMA-Bit gesetzt, so teilt der TNC dem Rechner mit, wann das Paket gesendet wurde. Hiermit wird der DAMA-Tout (22) Timer gestartet. Es ist also zur Entlastung des Tokenringes und des Rechners nur bei dem TNC zu programmieren, der auch als DAMA-TNC eingesetzt werden soll.

Watchdog, Version mit VANESSA:

Die Vanessakarte bietet mit dem PIN X38 auch eine RESET-Leitung an. Dieser Reset wird immer dann aktiviert, wenn 10 Minuten lang von der Karte (beide Ports) KEIN Frame gesendet wurde. Ist die Karte auf einem Interlink installiert, so wird dort ja alle 3 Minuten eine Bake gesendet und der Timer somit zurückgesetzt.

Der wesentliche Vorteil liegt darin, dass der Watchdog wesentlich günstiger (ca. 10.-DM) ist als die alte Version, die pro TNC/Rechner einen Aufwand von ca. 25.-DM erfordert. Der alte Watchdog soll manchmal nicht reagiert haben, da das Toggeln des RTS-Pins vorhanden war, obwohl sich der TNC aufgehängt hat.

Als Nachteil ist zu bewerten, dass man zum Knoten fahren muss, wenn tatsächlich mal ein TNC ausfällt, da dann der ganze Knoten steht und nicht nur ein Link betroffen ist. Aber nach Murphy trifft es eh immer den Link, der niemals ausfallen darf. Das ganze ist wahrscheinlich eine Glaubenssache, aber das Geld, das man spart, kann man besser in einen Link stecken.

Wenn man noch einen „Not-Reset" für den gesamten Netzknoten z.B. per DTMF-Steuerung vorsieht, kann man auch einen gezielten Reset durchführen und so einige Ausflüge zum Knoten sparen.

Und nach dem Neustart...

Nach dem Neustart des TNC, hängt er an das nächste Token seine Neustartmeldung an. Daraufhin bekommt er vom Rechner seine ihm zugedachten Parameter wie Halb- oder Vollduplex, TXDelay usw. mitgeteilt. Gleichzeitig wird der Vorfall in der Statistik des Programms aufgenommen, um solchen Störenfrieden auf die Schliche zu kommen.

Watchdog am LPT-Port

An einem LPT Port des TNN-Rechners kann ein zusätzlicher Watchdog angeschlossen werden. Der Port wird ähnlich TOKENCOM gesetzt: In der Umgebungsvariable muss mit „set TOGGLEPORT=x" ein Eintrag gesetzt
werden. Dabei steht „x" für die Nummer des LPT-Ports, zugelassen sind 1-3.

Es wird jede Minute einmal die Resetleitung (Pin 16) der LPT getoggelt. Es darf auch eine LPT auf einer Videokarte sein.

ANHANG C:

VANESSA

Installationsanleitung:

Inhalt:

1. Allgemeines.
2. Spezifikation.
3. Installation und Konfiguration.
4. Probleme und Fehlersuche.

1. Allgemeines:

SEPRAN ist ein Akronym und steht für

SWISS EXPERIMENTAL PACKET RADIO AMATEUR NETWORK.

Unter dieser Projektbezeichnung fallen verschiedene Aktivitäten, die im Rahmen der SWISS-ARTG (Swiss Amateur Radio Teleprinter Group) ausgeführt werden. Die Arbeiten umfassen Tätigkeiten auf verschiedenen Ebenen wie Computer, Software, Antennen oder Hochfrequenz-Baugruppen um ein schnelles PR System verwirklichen zu können.

1.1 Konzept PR-Knoten.

Es gibt viele Ideen, wie eine leistungsfähige Hardware aussehen müsste. Insbesondere kann man sich auch an am Markt etablierten X.25-Produkten (Multi-PAD, Multiplexer) orientieren. Die Merkmale eines guten Vermittlungssystems sind sowohl die hohe Datenrate als auch der grosse Datendurchsatz. Der kritische Punkt des Systems wird immer erreicht, wenn sehr viele und kurze Datenpakete zu verarbeiten sind. Eine Aufteilung der Aufgaben in Routing einerseits und Unterhalt der Links anderseits wird als notwendig erachtet, um die Ziele zu erreichen.

Ideal erscheint ein Mehrprozessorkonzept, in dem ein Prozessor die I/O-Last (FP VANESSA) abfängt und ein zentraler Prozessor (CP) die höheren Aufgaben für mehrere Kanäle erfüllt.

Diese Überlegungen führten zu folgendem Konzept:

Frontendprozessor VANESSA-2 mit zwei unabhängigen AX.25 Kanälen, den notwendigen Peripheriebausteinen und eine Z80-280 CPU von ZILOG.

IBM PC (XT oder AT kompatibel) als Routingprozessor (CP) für mehrere Frontendprozessoren.

DUAL-PORT-RAM als gemeinsames Interface zwischen dem CP und FP.

Warum ein PC als Routingprozessor ?

Der Einsatz eines PC als Routingprozessor bietet viele Vorteile. Betrachten wir einen Knoten mittlerer Grösse, so muss zumindest ein Gehäuse und eine Stromversorgung für den Digitalteil (5 VDC) beschafft werden. Für Tests ist temporär eine Eingabeeinheit und ein Bildschirm notwendig.

Daneben sollten bei einem komfortablen Knoten Helptexte für die USER, Parameter, Konfigurationstabellen ect. gespeichert werden. Auch bei einem Stromausfall sollten diese Daten nicht verloren gehen, so dass der SYSOP die Daten nicht jedesmal neu eingeben respektive neu laden muss. Dazu bietet sich ein PC mit einem Floppy-Laufwerk geradezu an. Die Daten werden beim Start von der Floppy ins RAM gelesen und stehen dann für die USER zur Verfügung.

Betrachten wir die Kostenseite, so sind die Aufwendungen für die Beschaffung eines (gebrauchten) PC sehr bescheiden, für einen mittleren Knoten genügt ein 286 Typ, kleinere Knoten benötigen einen XT oder AT mit 10 MHz. Eine Harddisk ist nicht notwendig, eine Floppy von 720 KByte genügt vollkommen. Und das sollte heutzutage gar kein Problem mehr darstellen.

2. Spezifikation:

Der Frontendprozessor ist auf einer IBM-PC-Steckkarte aufgebaut und unterstützt zwei AX.25 Datenkanäle mit hoher Geschwindigkeit.

CPU

Typ	ZILOG Z80 280,
Adressraum	64 KByte aus 16 MByte,
Run-Modi	System / User,
Cache	256 Byte,
MMU	eingebaut,
DMA	4 Kanäle,
Clock	9,8 MHz, 19,6 MHz optional,
Timer	3 getrennte Timer mit je 16 Bit.

Memory

RAM	32 KByte, 128 KByte optional,
EPROM	32 KByte, 128 KByte optional,
DP-RAM	8 KByte.

Interface

AX.25 Daten	ZILOG SCC 85C30, mit DMA,
RS-232C	Diagnose Terminal,
Opt. Anzeigen	5 LED,
Modem	2 Stück. AM-7911 1200bps,
Externe Modems	RS-422 Treiber,
Anschluss	Sub-D 37 poliger Stecker.

Datenrate

interne Modem	2 x 1200 bps Full Duplex,
externe Modem	2 x > 38400 bps Full Duplex.

2.1 Prozessor:

Als CPU wird der ZILOG Prozessor Z80-280 eingesetzt. Er ist binärkompatibel zum bekannten Z80 Prozessor, erlaubt jedoch durch seinen erweiterten Befehlssatz eine effiziente Speicherverwaltung und erspart durch die integrierten Timer DMA und UART einige externe Bausteine.

Die wichtigsten Merkmale sind:

Getrennte Stack - und Speicherbereiche für SYSTEM, USER, DATEN und PROGRAMMCODE,

Virtueller Speicherbereich von 16 MByte,

On-Chip-Peripherals: vier DMA Kanäle, drei 16 Bit Zähler, ein UART sowie Cache Memory.

2.2 Speicher:

Der verwaltbare Speicherbereich umfasst 16 MByte. Die eingebaute MMU ermöglicht je nach Programmierung bis 4 x 64 KByte Speicher im Direktzugriff zu benutzen. Die minimale Konfiguration umfasst je 32 KByte EPROM und RAM für Code respektive Daten. RAM und EPROM können je bis zu 128 KByte ausgebaut werden. Für zukünftige Speichererweiterungen wurden 32 Pin Byte-Wide-Sockel vorgesehen. Zusätzlich ist der Einsatz von 8 KByte Dual-Port-RAM [DP-RAM] vorgesehen. Ein Arbiter verhindert Zugriffskonflikte. Bei gleichzeitigem Zugriff beider Prozessoren auf das DP-RAM wird der eine Bus mit einem Wait State blockiert. Die maximale Verzögerung ist gegeben durch die Zykluszeit der CPU, die gerade einen Zugriff durchführt, und beträgt maximal 70 ns.

2.3 Memory Decoder:

Der Memory-Decoder fällt einfach aus, da durch den großen Speicherbereich die oberen Adressleitungen zu Decodierzwecken herangezogen werden können. Die Programmierung der MMU (Memory Management Unit) erlaubt eine beliebige Anordnung der physikalischen Speicherblöcke innerhalb der CPU Adresssegmente und somit die vollständige Auffüllung des Adressraumes im Direktzugriff mit Speicher.

2.4 Watchdog Timer:

Ein Watchdog Timer soll die Zuverlässigkeit gegenüber Hardwareproblemen erhöhen. Keinesfalls sollen damit Unzulänglichkeiten der Software überspielt werden. Der Watchdog wird innerhalb der Timer-Interrupt-Routine neu getriggert, so dass ein unzulässiger Programmzyklus einen Restart der Z80-280 CPU auslöst. Falls VANESSA je in einem Satellit auf intergalaktischen Flügen eingesetzt werden sollte, ist es beruhigend zu wissen, dass ein Fehler vom Watchdog-Timer erkannt und zum Systemrestart führen wird.

2.5 Status Anzeigen:

Zur Anzeige des Systemzustandes sind fünf LED Anzeigen vorhanden.

2.6 Modem:

Als Modem wird der AM-7911 eingesetzt. Dieses Single-Chip Modem erlaubt eine Baudrate von 1200 bps gemäss BEL-202 Norm. Für grössere Baudraten werden alle notwendigen Signale über RS-422 Treiber auf ein externes Modem zu führen.

2.6 RS-422 Treiber:

Die Signale vom und zum Modem werden ausschließlich über RS-422 Leitungstreiber geführt. Dadurch kann einerseits eine praktisch beliebige Leitungslänge erreicht werden, anderseits werden durch die paarverdrillten Leitungen Einstreuungen und Erdschleifen vermieden.

2.7 I/O Funktionen:

Für die Steuerung der Peripherie wurden zwei Output-Ports realisiert.

2.8 RS-232 Serial Port:

Für die Inbetriebnahme eines PR-Knotens, Hardwaretests sowie Weiterentwicklung von Hard - und Software ist am Datenstecker eine serielle Schnittstelle RS-232 für ein Diagnoseterminal vorhanden. Über dieses Terminal können interne Testprogramme aufgerufen sowie Betriebsparameter überwacht und geändert werden. Für den Normalbetrieb ist kein Terminal notwendig.

2.9 Bemerkungen zum Hardwarekonzept:

Durch die parallele Datenverarbeitung auf den Bus des Rechners hat es einen mächtigen Ruck im Datendurchsatz gegeben. Die Verarbeitungsgeschwindigkeit des Knotens wird nun durch seine Links (Baudrate und das Übertragungsverfahren (Semi - oder Vollduplex)) bestimmt. Die Laufzeiten innerhalb des Knotens sind durch die parallele Verarbeitung der Daten so gering geworden, dass sie auch bei einem 64 Kbit/s Vollduplexlink nicht ins Gewicht fallen. Beobachtungen des Knotens DB0NHM (der bereits jetzt von DB0EAM über eine Vanessakarte versorgt wird) zeigen deutlich, dass die Antwortzeiten, die der RMNC-Knoten ermittelt, nur ca. 50 % von seinem RMNC-Nachbarn DB0NID betragen (beide Funkstrecken 9600 Bit/s Semiduplex).

DUAL PORT RAM (DP-RAM):

Bei konventionellen Lösungen (TheNet, TheNetNode) geht der ganze Datenverkehr vom Host-Prozessor zum TNC und umgekehrt über die serielle RS232-Leitung. Sind nun mehrere Kanäle angeschlossen, konzentriert sich das ganze auf eine oder zwei Schnittstellen. Inzwischen sind Bitraten von 9600 bps pro Link keine Seltenheit mehr, so dass diese Schnittstelle der eigentliche „Flaschenhals" im System wird. Dazu kommt, dass serielle Schnittstellenkarten den PC - Bus stark belasten. Als Alternative besitzt die VANESSA ein DUAL PORT RAM. Damit ist ein schneller und einfacher Datentransfer vom und zum PC gewährleistet. Der PC und die VANESSA CPU können quasi gleichzeitig auf den Speicher zugreifen und die Übertragung erfolgt blockweise.

RS 422 Leitungstreiber:

An einem Knoten sind komplexe Erdverhältnisse vorhanden. Verschleppung von Kriechströmen führt unweigerlich zur Verschlechterung der Datenübertragung. Neben der Antennenerde (Blitzschutz) sind noch Schutzerde (Netzteil) und Ground der digitalen Signale zu verknüpfen, ohne dass Erdschleifen auftreten. Modemschaltungen für schnelle Links müssen möglichst dicht beim Modulator / Demodulator angeordnet sein, da sonst Einstreuungen und Leitungskapazitäten ihr Vorhandensein bemerkbar machen. Das bedeutet, dass einerseits der Aufbau des Modems möglichst dicht beim TRX (max. 30 cm Leitungslänge), anderseits der geordnete Aufbau in einem Rack verlangt wird. Zur Entkopplung sind an der VANESSA alle Signale zum oder vom Modem über RS 422 Leitungstreiber und paarverdrillte Leitungen geführt. Damit sind Leitungslängen von mehr als 500 m mit einfachem Signalkabel möglich, das System ist entkoppelt.

2.10 VANESSA Firmware:

KISS MODE (Keep lt. Simple and Stupid):

Für den Betrieb der VANESSA als TNC-Ersatz beim Knoten wurde eine KISS-Mode-Firmware entwickelt. Dabei übernimmt die VANESSA die Daten aus dem DP RAM paketweise und gibt sie über den SCC 85C30 (HDLC Controller) an die Modems weiter. Die Sendersteuerung (Slottime, DCD/Persistance Algorithmus, TX Delay ect.) erfolgt unabhängig vom Hostrechner durch die VANESSA. Empfangene Daten werden nach einem Plausibilitätstest via das DP RAM an den Hostrechner zur Verarbeitung weitergereicht. Damit ist der Hostrechner von den zeitkritischen Aufgaben weitgehend entlastet. Neben den eigentlichen KISS Routinen sind noch verschiedene Testroutinen für Senderabgleich und Hardwaretest (VANESSA Peripherie, Speicher LED etc.) vorhanden, die bei Bedarf von einem externen Terminal aufgerufen werden können.

HOST MODE:

Im Gegensatz zum KISS-Mode übernimmt im Host-Mode der FP VANESSA die komplette AX.25-Layer2-Protokollsteuerung. Zwischen Host und VANESSA werden nur die Infodaten sowie der Status der Links (Connected, Info, Busy, Disconnected etc.) übertragen. Der Hostrechner besitzt die Masterfunktion, d.h. der FP wird aufgefordert, Daten oder Status an den CP zu senden.

Damit ist die Verwendung der VANESSA als Hochleistungs-TNC für DieBox oder Terminalprogramme wie SP, GP oder THP möglich. Bei Bedarf werden bis zu 30 Kanäle pro VANESSA unterstützt.

Der weiteren Entwicklung für PR oder verwandte Betriebsarten sind (noch) keine Grenzen gesetzt. Die Rechenzeitreserven sind noch nicht ausgeschöpft, Speicherplatz ist ebenfalls genügend vorhanden. Dank der universellen RS 422 Schnittstellen kann jedes beliebige Modem mit internem oder externem Clock adaptiert werden.

3 Installation:

Der Aufbau der VANESSA Hardware wurde bewusst universell ausgelegt, um individuelle Konfigurationen zu ermöglichen. Die einzelnen Funktionen sind aus dem Schema ersichtlich, die wichtigsten werden hier kurz erklärt.

3.1 Grundkonfiguration Jumper:

2 x 1200 bps, BEL 202, Port 0/1. (Vanessakarte so gehalten, dass die SUB-D 37 nach rechts zeigt.)

```
X1  links        X2  links        X23 links        X21 links
X18 oben         X17 unten        X16 unten        X15 oben
X14 unten        X9  unten        X10 unten        X11 oben
X12 oben         X13 oben         X8  unten        X7  oben
X6  oben         X5  unten        X4  unten
```

X24, X25, X28, X29, X27 und X26 bleiben geschlossen.

DIP Schalter SW 1 alle Switches auf ON.

IC 13 und IC 14 bestückt (Modem AM-7911). IC 33 bis 36 leer (RS422 Schnittstelle).

3.2 Einstellung Port-Adresse:

Die Port Adresse wird mit dem DIP Switch SW 1 definiert. Damit wird die Adresse des DUAL PORT RAM festgelegt. x dont care

1	2	3	4		Dip-Switch	
x	ON	ON	ON		Port Nr. 0/1	Karte 1
x	OFF	ON	ON		Port Nr. 2/3	Karte 2
x	ON	OFF	ON		Port Nr. 4/5	Karte 3
x	OFF	OFF	ON		Port Nr. 6/7	Karte 4
x	ON	ON	OFF	*	Port Nr. 8/9	Karte 5
x	OFF	ON	OFF	*	Port Nr. 10/11	Karte 6
x	ON	OFF	OFF	*	Port Nr. 12/13	Karte 7
x	OFF	OFF	OFF	*	Port Nr. 14/15	Karte 8

Bemerkung:

Vanessa Port 8-15: Da es für die Adressierung der LED Adresskonflikte mit einigen HD-Controllern gab, wurden die Adressen verschoben. Dadurch muss leider eine Hardwareänderung auf der Vanessakarte durchgeführt werden. Am IC29 muss der Pin 16 (bei Port 0-7 auf +5 Volt) auf GND umgelegt werden. Das kann am einfachsten mit einem Zwischensockel und einer Drahtbrücke gemacht werden, z.B. nach Pin 18.

3.3 Verwendung Modem nach G3RUH:

Bei Einsatz von Modems nach G3RUH entfallen IC13 (Port A) und IC14 (Port B). Die Jumper X24 und X28 (Port A) resp. X25 und X29 (Port B) sind offen. Anstelle der internen Modembausteine AM-7911 werden die entsprechenden Leitungstreiber IC33 bis IC36 gesteckt.

Achtung: CTSa (b) und RTXCa (b) (IC35/36 Pin 11 und 13) werden vom G3RUH Modem nicht unterstützt, und DÜRFEN NICHT in die Sockel gesteckt werden!

Konfiguration Jumper:

Port A	Port B	X24	X25	X28	X28
G3RUH	1200 bps	OFF	ON	OFF	ON
1200 bps	G3RUH	ON	OFF	ON	OFF
G3RUH	G3RUH	OFF	OFF	OFF	OFF

Die Vanessakarte stellt an ihrem Ausgang TRXCa (b) den 32-fachen Takt der eingestellten Baudrate zur Verfügung. Für das G3RUH-Orginal Modem wird jedoch nur der 16-fache Takt benötigt. Deshalb muss dieser noch mal durch 2 geteilt werden, z.B. mit einem 74HC74, was der Flankenqualität zugute kommt.

3.4 Modem mit eigenem Clock Generator:

Die DF9IC-Modem bieten im Gegensatz zu den Orginal-G3RUH-Modem Ausgänge für den RX - Takt und den TX - Takt. D.h. das Modem erzeugt die entsprechenden Takte intern.

Zum besseren Verständnis sollte man die VANESSA - Pläne, Seite 5 und Seite 2, zur Hand haben.

Die Beschreibung bezieht sich nur auf Port A der VANESSA, der Port B wird ähnlich verdrahtet.

Der RX - Takt wird an IC35 Pin 14/15 (RS422-Seite) gelegt und gelangt über Pin 13 dann auf den - RTXCa.

Für den TX - Takt muss ein 26LS32 als Eingang missbraucht werden, es bietet sich für Port A IC35 Pin 9/10 an. Der Pin 11 von IC35 darf nicht mehr in den Sockel gesteckt werden, sondern wird mit einer kleinen Drahtbrücke auf den Anschluss - TRXCa an IC34 gelegt. Der entspr. Treiber auf IC34 sollte durch herausbiegen des IC - Beinchen ausser Betrieb genommen werden (Verbraucht sonst nur Strom). Werden auf beiden Vanessa - Ports DF9IC-Modem eingesetzt, kann IC34 auch komplett eingespart werden.

Da der Anschluss - CTSa jetzt offen liegt, muss dieser auf Masse Potential gelegt werden!

Zum Schluss muss noch der Taktteiler IC27 Port A bzw. IC32 Port B von der Vanessa entfernt werden. Da die VANESSA bei externen Takt - Modus **NUR** NRZ - Codierung verarbeiten kann, müssen die Modems mit NRZ - Gal bestückt sein. (Müssen extra bestellt werden!)

Noch eine Warnung zum Schluss:

Verwendet unbedingt die RS422-Schnittstellen, ohne die Treiber gibt’s bereits bei 20 cm Kabellänge bei 38k4 schon erheblich Probleme und der SCC wird durch Potential - Unterschiede unnötig gefährdet!

Parametereinstellung:

Als Parameter muss im Mode dann ein „e" wie Externer Takt angefügt werden. Für einen 38k4 Vollduplex - Link währe der richtige Eintrag „mode=38400de". Die Baudratenangabe ist zwar dann unwichtig, da der SCC auf der VANESSA ja vom Modem getaktet wird, aber die richtige Baudrate ist informativer.

3.5 Test der Konfiguration:

Für den Funktionstest der VANESSA wird ein Terminal am 37-Sub-D-Stecker angeschossen. Der RS232-Anschluss wird mit 4800 Bd., 8 Bit, no parity betrieben.

Eine Beschreibung der FIRMWARE-Testschritte ist im VANESSA USER Manual enthalten. Beim Start des PC wird auf der RS 232 Schnittstelle ein Logon Menü ausgegeben, die beiden unteren LED erlöschen.

Das Interface VANESSA zum PC kann nun mit dem Hilfsprogramm VANUTIL.EXE ausgetestet werden. Nachfolgend eine kurze Erklärung der Testschritte:

1. Show Konfiguration
   Anzeige der eingestellten Adresse des DP-RAM und der FIRMWARE Version.

2. PC I/O Port Test
   Test der beiden oberen LED an der Steckerplatte.

3. Dual Port RAM Test
   Statischer Test des Dual-Port-Ram. Dabei wird ein Testpattern eingelesen und anschliessend verglichen.

4. Vanessa Reset
   Test der RESET-Leitung zur VANESSA, die unteren LED werden ebenfalls angesteuert.

5. Dynamic RAM Test
   Dies ist der wichtigste Test! Dabei werden Daten ins Dual-Port-Ram geschrieben, durch die Vanessa an eine andere Adresse verschoben und durch den PC wieder ausgelesen und verglichen. Dabei wird das Timing auf dem PC-Bus sowie die korrekte Funktion des Dual-Port-Ram geprüft. Treten hier Fehler auf, so besteht die Gefahr, dass Daten korrupt übertragen werden. Hinweise zur Fehlersuche im folgenden Abschnitt.

3.6 Dual Port Mode:

Mit Hilfe des Programmes „VANCONF.EXE" lassen sich leicht Multi-Baud-Userzugänge einrichten. Die beiden verknüpften Ports müssen jedoch auf derselben Karte liegen. Unter Dual-Port-Mode werden die PTT und DCD der beiden Ports gegeneinander softwaremässig verriegelt. Der Dual-Port-Mode muss beim Starten des Rechners mit Hilfe der BAT-Datei bereits festgelegt werden. Soll der Port 0 + 1 verwendet werden, so ist das Programm „VANCONF 0 D" aufzurufen. 0 = Port 0 + 1 D = Dual-Port-Mode. Die Baudraten brauchen nicht gesetzt zu werden, da sie durch TNN eingestellt werden.

4. Probleme, Fehlersuche:

VANUTIL.EXE erkennt keine Vanessa-Fehler im Dual-Port-RAM oder Adressierlogik zum PC-Bus. VANESSA-Firmware, CPU und RAM kontrollieren mit DEBUG und Terminal an der RS232. Erlöschen die beiden unteren LED beim Start? Kontrolle der Adressmodifikation bei Port Nummer 8 und grösser. Fehler im Dual-Port-RAM Bereich können mit dem DEBUG-Programm ebenfalls verifiziert werden. Die Adresse des Dual-Port-Ram kann wie folgt berechnet werden: Startadresse: 0xD000:0000 + (n * 0x1000), d.h. 0xD000:0 beim ersten Port, D100:000 beim zweiten Port usw. In diesem Bereich (0xD000:0 - 0xD000:fff können mit DEBUG Daten eingegeben und wieder ausgelesen werden (n = Port Nr. 0 ...15).

Datenverluste:

Mit einem schnellen Rechner wurden verstümmelte Frames oder gar Datenverlust an der Vanessa-Schnittstelle beobachtet. Mit einem XT-Board waren keine Probleme zu beobachten, aber ab ca. 16 MHz traten diese Störungen auf. Das Dual-Port-Ram auf der Vanessa muss deutlich schneller als 100 ns sein, mit 70ns-Rams konnten bei 33 MHz Rechnern keine Probleme mehr beobachtet werden. Bei 386er oder 486er sollte das Mainboard-Setup auf folgende Werte gesetzt werden, da der AT-Bus-Clock mit 8 MHz spezifiziert ist:

- DRAM READ 1 WAIT STATE,
- DRAM WRITE 1 WS,
- EXTRA AT CYCLE WS 1,
- AT BUS CLOCK CLK / x (so dass der Bus-Clock ca. 8 MHz ist).

Versuche auf dem DB0EAM-Rechner haben ergeben, dass die Vanessakarten auch bei 12 MHz Bus-Clock mit VANUTIL8.EXE keinerlei Fehler zeigten.

Alternativ kann die Bestückung der VANESSA geändert werden. Beim Einsatz von ALS-Typen anstelle der HC- oder HCT-Typen im Bereich des Dual-Port-RAM konnten auch bei schnellerem Bus-Clock keine Fehlfunktionen mehr festgestellt werden.

Anstelle der HC/HCT-Typen werden IC 20, 21, 22, 23 24, 25 und IC 28 gegen ALS-Typen ausgetauscht.

LED

Zu den LED kommen auch immer wieder Fragen. An der Sub-D-37-Leiste sitzen die grünen LED. Sie werden von dem Z280 Prozessor gesteuert und zeigen das Senden eines Frames an (nicht zu verwechseln mit der PTT, die bei Duplex-Betrieb ja für 3 Minuten nach dem letzten Frame getastet bleibt). Diese LED werden deshalb auch beim VANUTIL-TEST nicht angesteuert. Ist externer Clock bei der Vanessa eingestellt, aber KEIN Modem angeschlossen, so können diese LED leuchten, da der SCC dann einen Takt erwartet.

Die darüberliegenden gelben LED zeigen den Empfang von Daten via dem Dual-Port-RAM, also über den Rechnerbus, an. Die Funktion der LED lässt sich mit dem VANUTIL-Test prüfen.

Die einzelne rote LED zeigt die Funktion des karteneigenen Prozessors an. Sie leuchtet, wenn der Prozessor steht (was eigentlich nie passieren darf..).

Die beiden linken LED (gelb und grün) sind für den Port A der Karte zuständig und somit die rechten beiden für den Port B.

Die oberen LED können auch mit dem Debug-Programm via PC-Port gesteuert werden. Die Adressberechnung ist wie folgt: Linke LED Adresse 0x306 + (n * 0x8), also 0x306, 0x30E ... Rechte LED 0x307 + (n * 0x8), also 0x307, 0x30F usw.

Daraus ergibt sich die folgende Adressenliste:

```
Port  0 :  o 306 00 --> LED leuchtet (oben links)       o 306 01 --> LED aus
Port  1 :  o 307 00 --> LED leuchtet (oben rechts ...)
Port  2 :  o 30e 00
Port  3 :  o 30f 00
Port  4 :  o 316 00
Port  5 :  o 317 00
Port  6 :  o 31e 00
Port  7 :  o 31f 00
Port  8 :  o 126 00 ..Ist die Adresse nicht geändert:   o 326 00
Port  9 :  o 127 00                                     o 327 00
Port 10 :  o 12e 00                                     o 32e 00
Port 11 :  o 12f 00                                     o 32f 00
Port 12 :  o 136 00                                     o 336 00
Port 13 :  o 137 00                                     o 337 00
Port 14 :  o 13e 00                                     o 33e 00
Port 15 :  o 13f 00                                     o 33f 00
```

Die unteren beiden LED lassen sich genauso ansprechen unter den Adressen:

```
Port  0 +  1  o 300 00 --> LED Leuchten                 o 300 01 LED aus
Port  2 +  3  o 308 00
Port  4 +  5  o 310 00
Port  6 +  7  o 318 00
Port  8 +  9  o 120 00 ..Adresse nicht geändert:        o 320 00
Port 10 + 11  o 128 00                                  o 328.00
Port 12 + 13  o 130 00                                  o 330 00
Port 14 + 15  o 138 00                                  o 338 00
```

Offene Gatter auf der Vanessakarte: Offene C-MOS Eingänge an Gattern neigen zum Schwingen. Deshalb sollten sie besser auf einem logischen Pegel liegen. Betroffen sind die folgenden IC: (Klammerwerte = IC-Eingang)

IC 9 (13); IC 20 (9-10) und (12-13); IC 25 (9-10) und (12-13);
IC 26 (10) und (12) und IC 12 (4-5), (9-10) und (12-13).

SWISS ARTG
Swiss Amateur Radio Teleprinter Group
Geschäftsstelle Arturo Dietler, HB9MIR,
Blauenweg 8
CH-4335 Laufenburg
Autor: HB9PAE @ HB9AJ

ANHANG D:

(Ethernet) ...Was muss ich tun ?

Ethernet gibt es auf 2 unterschiedlichen Kabeln.

Zum einem ist der Betrieb auf dem uns sehr bekannten RG58 möglich. Hierbei MUSS man unbedingt beachten, dass BEIDE Enden des Kabels mit je einem 50 Ohm Abschlusswiderstand versehen werden müssen.

Das Gegenstück zum RG58 ist Twisted-Pair. Es besteht aus einer Doppelader und benötigt keinen Leitungsabschluss.

Wie geht man nun vor, wenn man bei Ethernet noch unsicher ist?

1.) Zuerst muss man sich auf seinen Rechner einen freien IRQ suchen, den die Ethernet-Karte belegen kann.
2.) Dann ist der Adressbereich der Ethernet-Karte im Bereich 0x2e0 - 0x320 festzulegen. Hier kann es zu Überschneidungen mit den Vanessakarten kommen, bitte beachten.
Die Vanessakarten benutzen die Adressen 0x300...0x33f (je nach Anzahl der Vanessakarten und verwendetem Port.) Ausserdem wird default der Adressbereich D000..Dxxx benutzt. Auch dieser muss bei gleichzeitiger Verwendung von Vanessakarten und Ethernet-Karten bedacht werden.
3.) Nach dem Einbau in den Rechner kann nun die Ethernet-Karte auf Brauchbarkeit und Funktion mit dem Programm TESTDRVR.EXE überprüft werden. Dazu muss der entsprechende Kartentreiber noch aufgerufen werden. Z.B.: NE2000.COM mit Angabe des Soft/Hardware IRQ und Adressbereich der Karte. NE2000 0x60 0x300.
4.) Meldet nun TESTDRVR keine Probleme auf den Rechnern, so kann man auf dem einen Rechner SEEPKT.EXE starten (dieses ist ein Ethernet-Monitor) und auf dem anderen Rechner SENDPKT.EXE (dieses ist eine Bake, die ständig Daten sendet). Die Daten, die SENDPKT.EXE erzeugt, sollen mit SEEPKT.EXE lesbar sein. Dann steht eigentlich einer Verbindung der 2 Rechner über Ethernet nichts mehr im Wege, sofern das in beiden Richtungen funktioniert.

Aber erst mal zum Testen der Schnittstellenkarte:

TESTDRVR - Ein Utility zum Testen von FTP-Packet Treibern:

Vorwort:

Leider halten nicht alle mit den Netzwerkkarten gelieferten Treiber, was das Handbuch verspricht. Dies mag zum Teil an Mehrdeutigkeiten in den Spezifikationen liegen. Teilweise liegt es aber mit Sicherheit an falsch verstandenem Schutz vor „Datenspionen" im Netz. Und einiges können wir auch auf das Konto „Schlampereien" buchen.

Man kann nun den Treiber mit der Anwendungssoftware testen. Das ist oft unbequem und die Fehlermeldungen sind sparsam oder nichts sagend - wenn überhaupt welche kommen. TESTDRVR soll hier für etwas Bequemlichkeit sorgen.

Wenn der mit der Karte gelieferte Treiber nicht richtig will, kann man bei den Standardkarten immer die Treiber aus dem FTP-Paket nehmen. Für NE2000 und WD (SMC) kompatible Karten sind Treiber in diesem Paket enthalten.

FTP-Packet Treiber sind meines Wissens die einzigen Treiber, die verschiedene Netzwerkprotokolle auf einer Karte gleichzeitig gestatten. Hinzu kommt, dass die Schnittstelle hervorragend dokumentiert ist und die gesamte Software dafür im Quelltext frei verfügbar ist.

In dieser Anleitung wird nicht viel über Netzwerkgrundlagen geschrieben. Das soll in einer getrennten Dokumentation geschehen.

Voraussetzungen:

Hardware

Netzwerkkarten bekommt man heute „NE2000" kompatibel für weniger als € 10,-. Üblich ist sind „Cheapernet" = BNC Anschluss (10Base2) und/oder UTP (RJ45) Anschluss (Twisted-Pair). Der BNC Anschluss hat den Vorteil, dass man ohne weitere Hardware einige Rechner koppeln kann. Wer einen Rechner „solo" testen will, beachte bitte, dass der Netzwerkanschluss an beiden Seiten mit jeweils 50 Ohm abgeschlossen sein muss. Sonst verweigern die meisten Karten die Sendung. Also ein T-Stück mit zwei 50 Ohm Abschlüssen aufstecken.

Moderne Karten werden nicht mehr über Jumper konfiguriert, sondern per Software. Das kann Probleme machen, wenn die vom Kartenhersteller gewählten Defaults im Rechner bereits belegt sind. Dann muss man unter Umständen erst andere Karten ausbauen, um die Netzwerkkarte umkonfigurieren zu können. Eine NE2000 kompatible Karte braucht einen freien Interrupt und eine freie IO-Adresse. **ACHTUNG: Werden Vanessakarten eingesetzt, so sind die IO-Adressen je nach Port zwischen 0300-0338 bereits belegt!!**

Software

Wer schon einen Netzwerkanschluss an seinem Rechner hat, bedenke bitte, dass eine Netzwerkkarte nicht zwei Herren gleichzeitig dienen kann. Man muss also die vorhandene Netzwerksoftware (Novell, WfW usw.) totlegen, um mit FTP-Packet Treibern experimentieren zu können. Anschliessend kann man dann versuchen, die „alte" Software auf der Basis des Packet-Treiber zu installieren. Nur in dieser Richtung geht es, nicht anders herum. Und ein Packet-Treiber braucht nie mehr als einen Aufruf zum Installieren (Beispiel „EP3000PD 0x60"). PROTMAN.SYS und ähnliche Programme sind hier definitiv falsch.

Bei der Installation muss im Allgemeinen ein Software-Interrupt für den Treiber angegeben werden. Dieser Interrupt kann per Definition im Bereich 0x60 bis 0x80 liegen. Es spricht nichts dagegen, 0x60 zu verwenden.

Testablauf

Als erstes wird die Signatur des Treibers gesucht und so der verwendete Software-Interrupt gefunden. Natürlich könnte man auch den Software-Interrupt im Aufruf angeben. Da aber fast alle Programme den Treiber über seine Signatur suchen, macht es schon Sinn, hier auch so vorzugehen.

Dann müssen wir dem Treiber mitteilen, was wir von ihm verlangen - den Netzwerktyp, welche Art Pakete usw. Außerdem müssen wir ihm sagen, wo er die Pakete abliefern soll, die „für uns" eingegangen sind. Der Treiber liefert uns als Ergebnis ein „Handle", das zukünftig bei der Kommunikation zwischen dem Treiber und uns den Briefkasten bezeichnet. Ohne gültiges Handle wird nicht gearbeitet.

Nun fragen wir den Treiber, welchen Befehlssatz er versteht. Es gibt drei Ausbaustufen: Basisdienste, erweiterte Dienste und highperformance Funktionen. Wir brauchen die erweiterten Dienste. Die ersten Versionen von TFNETR und TNN liefen auch mit den Basisdiensten. Um zu PE1CHL und G8BPQ kompatibel zu werden, haben wir das Protokoll aber noch einmal geändert.

Danach fragen wir die Hardwareadresse der Karte ab. Diese Adresse ist eindeutig, jede Karte hat eine andere Adresse. Wir sollten sie nie ohne zwingende Grunde ändern. Bei dieser Abfrage erleben wir die Schönheiten der Segmentierung des 80X86 Prozessors. Wenn der Programmierer des Kartentreibers nicht sorgfältig war, kommt hier als Adresse 00:00:00:00:00:00. Und dann sind wir ziemlich sicher, dass die gleiche Schlamperei auch in der Empfangsroutine drin ist. Damit sind dann Abstürze vorprogrammiert.

Die Kommunikation zwischen zwei Netzwerkkarten kann auf mehrere Arten erfolgen:

-- Im Paket werden die Hardwareadressen der beteiligten Karten angegeben. Das ist kompliziert. Denn ich muss vor der ersten Kontaktaufnahme die Adresse meines Partners kennen.

-- Die Pakete werden „an Alle" gesendet (Broadcast). Für diese Zwecke ist die Zieladresse FF:FF:FF:FF:FF:FF definiert. Dieses Verfahren stört in größeren Netzen sehr, weil wir dann mit unseren Paketen das gesamte Netz überschwemmen und jeder unsere Pakete hören und analysieren muss.

-- Die Pakete werden an eine Gruppe gesendet (Multicast). Nur die Gruppe hört auf diese Pakete. Dieses Verfahren werden wir verwenden. Wir kontrollieren nun, ob unser Treiber dieses Verfahren empfangsseitig versteht.

Der nächste Test erfolgt, um zu sehen, ob das Programm NETWATCH laufen wird. NETWATCH ist ein universeller Netzwerkmonitor. Leider haben viele Programmierer den dazu notwendigen „promiscuous mode" zugenagelt. Aber es gibt ja die FTP-Treiber.

Nun setzen wir den Treiber sendeseitig in den Multicast-Mode. Dazu definieren wir die „BPQ-Adresse.

Als letzten Test schicken wir ein Paket (mit zufälligem Inhalt) an unsere Gruppe. Wenn der Abschlusswiderstand nicht steckt, bleiben einige Karten hier hängen. Andere kommen mit einer Fehlermeldung.

Zum Schluss geben wir das Handle wieder frei. Wenn der Test bis hierher durchgelaufen ist, besteht Hoffnung, dass der Treiber auch mit der Anwendersoftware laufen wird.

Ein letzter Hinweis zum Schluss: ein Rechnerabsturz beim Empfang oder Senden eine Paketes in Verbindung mit einer WD (SMC) Karte ist meist auf einen Fehler in der Konfiguration des Speichermanagers zurückzuführen. Diese Karten nutzen nämlich Teile des Hauptspeichers zur Kommunikation mit dem Treiber.

Fehlermeldungen:

Einige Fehlermeldungen sollten nie vorkommen. Sie sind mit einem??? markiert. Treten sie dennoch auf, so deutet das auf eine kapitale Macke im Treiber hin.

kein Fehler ???

Der Treiber hat einen Fehler gemeldet und erklärt bei Nachfrage, es sei nichts gewesen.

ungültiges Handle???
: Der Treiber erinnert sich nicht an sein eigenes Handle.

ungültige Klasse
: Der Treiber unterstützt die Klasse „ETHERNET” nicht.

ungültiger Typ???
: Wir haben „irgendeinen” Typ verlangt.

ungültige Nummer???
: Wir haben „die erste Karte” verlangt.

ungültiger Packet Typ???
: Wir haben „alle verfügbaren” Typen verlangt.

Multicast unzulässig
: „Unser” Modus wird nicht unterstützt, Treiber unbrauchbar.

TERMINATE unzulässig???
: Wir verlangen nicht, dass der Treiber sich entfernt.

unzulässiger RX-Modus
: „Unser” Modus wird nicht unterstützt, Treiber unbrauchbar.

nicht genug Speicher
: Karte ist überfordert, nach RESET des Rechners noch einmal versuchen. Ist noch andere Netzwerksoftware aktiv?

Type ist belegt
: Andere Software hat die Karte undefiniert hinterlassen.

unzulässiger Befehl
: Ein Standardbefehl wird nicht erkannt, Treiber unbrauchbar.

(Hardware) Fehler beim Senden
: Abschlusswiderstand nicht gesteckt oder falscher Anschluss.

kann Hardwareadresse nicht setzen???
: Wir fummeln da nicht dran.

falsches Hardware-Adressenformat???
: Der Treiber ist defekt.

RESET nicht möglich???
: Da wurde ein Befehl missverstanden.

Auf der TNN-Seite ist nun folgendes nötig:

Es muss in der Start.bat ein Ethernetkarten-Treiber aufgerufen werden. Treiber einfach mal aufrufen, meist sagen sie dann, was sie gerne für Parameter hätten. Hierzu ist auf der Diskette ein DIR TESTDRVR mit ein paar Hilfsprogrammen.

```
tnl1.exe -v
nwpd 0x68 10 0x340
ipxpd.exe /i=0x68

TNNGO32.exe

tnl1.exe -u
nwpd -u
```

ANHANG E:

DER PACSAT-BROADCAST-SERVER von TNN

ABLAUF:

TNN macht simplen DieBox-S&F mit einer Mailbox und speichert die Nachrichten auf der Platte und macht dann mit den Daten seinen Broadcast. Der Befehl „BOX" schaltet TNN in den S&F-Modus. Die Nachrichten werden so, wie sie über den S&F kommen, mit aufsteigenden Nummern gespeichert. Beim Start von TNN werden die 1. und die letzte Nummer ermittelt.

Der Broadcast:

Das Protokoll ist in PACSAT.TXT dokumentiert, er ist 100% PACSAT kompatibel. Der PACSAT-Server benutzt optional und extended header und überträgt im RLM Message Format. Damit sollte auch z.B. WISP (bei AMSAT zu bekommen) klarkommen und der User kriegt die Nachrichten automatisch in Rubriken sortiert. Ein von PE1HCL umgestrickter Headergenerator heisst jetzt PFHADD.EXE (V3.3 vom 2.6.95), er muss im PACSAT-Verzeichnis stehen. Damit WiSp mit dem TNN-PACSAT-BROADACST zurecht kommt, wird vor dem eigentlichen File noch der DIR-Eintrag ausgesendet.

Bedienung:

- Nachrichteneingang durch DieBox-1.8-Kompatiblen S&F (keine BIN-Mail!).

- User&Sysop-Schnittstelle (Befehl PACSERV, PAC*), dort stehen zur Verfügung:

Für den Sysop:

+ Port

Broadcast für einen Port aktivieren.

- Port

Broadcast deaktivieren.

R Reload, das Filesystem nach einer externen Veränderung neu einlesen.

C PacSat-Boxcall setzen, C call, sollte das Rufzeichen des Knotens sein oder ein Fantasie-Call, es muss aber mit dem Rufzeichen übereinstimmen, mit dem man bei der Box eingetragen ist. Dieses Rufzeichen wird niemals HF-mässig gesendet. Es wird eigentlich nur benötigt, da bei DieBox alle Partner unterschiedliche Calls haben müssen. DB0HRO-5 (Box) macht mit DB0HRO-7 (Cluster) S&F, somit kann man dort nicht mehr DB0HRO (Knoten) als S&F-Partner hinzufügen, also trägt man dort den Knoten als BR0CST (oder was auch immer) ein und stellt M auf M BR0CST. Dann sollte es klappen.

P Zum Einstellen verschiedener Parameter

Ablauf des Broadcast:

TNN hat eine Prioritätsschaltung, welche die letzten 200 eingegangenen Nachrichten im 10-15 Minutentakt wiederholt, damit User nur kurze Zeit einzuschalten brauchen und sofort die wichtigsten Nachrichten haben. Für den Rest der Zeit werden die sonstigen Nachrichten ausgesendet.

Broadcast Format:

Ist 100% kompatibel zu dem, was unsere künstlichen Erdtrabanten machen, inkl. DIRECTORY-BROADCAST.

Installation:

In TNN179.PAS muss der PACSATPATH angefügt werden, er sollte auf C:\TNN\PACSAT\ stehen. Dort werden alle Nachrichten gespeichert. In dieses Verzeichnis muss auch die Datei „PFHADD.COM" kopiert werden, sie wird intern benötigt. Dann muss man den Sysop der nächsten Mailbox beknien, damit der S&F einträgt. Die Box muss, wenn sie den Knoten connected hat, „BOX" eingeben, um in den S&F-Modus zu kommen. Sobald die 1. Nachricht eintrifft, geht das Broadgecaste los, man kann auch auf mehreren Ports gleichzeitig broadcasten (es geht auch auf einem mit 9k6, auf dem anderen mit 19k2, d.h. die Abfolgen sind unabhängig, der eine Port wartet nicht auf den anderen). Der Knoten benötigt (natürlich) eine Festplatte, sie sollte schon 80MB oder so haben, mehr ist auch nicht schlecht.

SONSTIGES:

Der User:

WISP ist ein Programm, das automatisch die Mail dekodiert und in Rubriken sortiert.

Wartungsaufwand für den Sysop:

Wenig. Wenn der Broadcast erstmal installiert ist, läuft er von selbst und braucht nicht gewartet zu werden. Wenn auf der Platte ein Lesefehler auftritt, kann der Sysop eine Freifahrt haben, das ist bei Boxen aber auch so, ist leider nicht zu ändern.

Wichtig:

TNN muss pro Sekunde etwa 1,5kb von der Platte lesen, deshalb ist SMARTDRV mit NUR-LESE-CACHE zu empfehlen, es schont die Platte. Der VORLESESPEICHER sollte von standardmässig 8kb auf 32kb gestellt werden (damit mal experimentieren).

Vanessa ist Tokenring vorzuziehen, der Tokenring muss bei 9k6 Broadcast 19k2 haben, 38k4 ist besser, auf vollen Digi (viele Links, z.B. DB0FC) kann es passieren, dass Dauertastung nicht geht, das ist nicht zu beheben. Mit Vanessa sollte es nicht passieren.

Doppelte-Nachrichten:

Werden nicht überprüft. User, die über Box-Software verfügen (WinGT), die S&F kann, können auch in den Server einspielen, das geht dann auch auf dem Ausgang raus. Der Sysop sollte das ein wenig überwachen, damit die Leute keinen Unfug machen.

Request:

(Also anfordern von fehlenden Teilen durch UI-Frames) ist für unsere Verwendung nicht sinnvoll, wird es auch erstmal nicht geben.

Spezial:

In einigen Gebieten ist ein PR-BROADCAST-Ausgang im ISM-Bereich wegen vorhandener ATV-Belegung nicht möglich. Aus diesem Grund wird z.B. in Köln BROADCAST auf der Eingabe (!) eines FM-Relais gemacht. Dazu werden immer nur 2 Frames gesendet, um dann zu prüfen, ob das Relais geöffnet wurde. Ist das Relais noch geschlossen, so werden wieder 2 Frames gesendet. Wurde das Relais aber geöffnet, so wird über eine kleine Schaltung die PTT gesperrt. Der Durchsatz ist natürlich geringer als bei einem DAUER-TX-Ausgang, aber könnte durchaus eine gute Alternative darstellen. Zu diesem Zweck wurde eine kleine Schaltung entwickelt. Bei Fragen bitte wenden an: (DG1KWA / Andreas @DB0KOE).

ANHANG F:

Aufbau der Layer 3 / 4 Frames bei TheNet, TheNetNode und TheBoxware.

Da für die 3 Programme TheNet, TheNetNode und TheBoxware dasselbe Layer 3/4-Protokoll gilt, wird im Folgenden nur von TheNet die Rede sein, aber es sind alle 3 Programme gemeint (wenn nicht ausdrücklich anders vermerkt). Hier werden nur die Frames mit erweitertem AX.25-Format behandelt, Identifikations-Frames etc. bleiben unberücksichtigt.

Alle speziellen Layer 3/4 Frames von TheNet werden mit dem PID „CF" ausgesendet, wodurch eine Unterscheidung von normalen Layer 2 Frames gewährleistet ist (bei denen ist der PID „F0"). Dies gilt sowohl für I-Frames als auch für UI-Frames.

Das zusätzliche Protokoll im Infofeld des AX.25-I-Frames besteht grundsätzlich aus 3 Blöcken: der Netzwerk-Kontrollblock (Layer 3) besteht aus 15 Bytes, der darauf folgende Transport-Kontrollblock (Layer 4) besteht aus weiteren 5 Bytes, und schliesslich bleibt noch der Informationsteil, für den eine Länge von 0 bis 236 Bytes vorgegeben ist.

Netzwerk Layer Frames.

Der Router von TheNet bestimmt, auf welchem Weg Nachrichten von einem Zielknoten zum anderen übertragen werden. Das Routing erfolgt automatisch auf dem vermutlich günstigsten Weg. Der während einer Verbindung verwendete Weg kann sich ändern, wenn z.B. ein Netzknoten ausfällt; dann wird automatisch ein alternativer Weg verwendet (wenn vorhanden) - der Enduser merkt von der Umleitung nichts.

Für das Routing bei TheNet werden zunächst Informationen über die erreichbaren Zielknoten mit einer Broadcast-Aussendung ausgetauscht. Bei TheNetNode wird die Laufzeit zum Nachbarknoten gemessen.

Der höhere Transport-Layer baut auf dem Netzwerk-Layer auf, so dass alle TheNet-I-Frames einen Netzwerk-Kontrollblock enthalten, der immer gleich aufgebaut ist. Er besteht aus insgesamt 15 Bytes. Als erstes kommt das Rufzeichen des Absenderknotens (7 Bytes), gefolgt vom Rufzeichen des Empfängerknotens (7 Bytes). Schliesslich wird die Lifetime dieses Frames übertragen (1 Byte). Die Lifetime wird bei jeder Weitergabe an einen Nachbarknoten um 1 vermindert, bis entweder der Zielknoten erreicht ist, oder aber der Wert 0 wird. Falls der Wert 0 erreicht wird, ist das Frame zu vernichten, damit ein Frame nicht durch eine zufällig entstandene Schleife des Netzes endlos lange kreist und dadurch das Netz blockiert.

Laufzeitmessung bei TheNetNode.

Um die Laufzeit für Layer 3 Frames zu einem Nachbarknoten und zurück zu messen, werden von TheNetNode regelmässig Messframes an alle connecteten Nachbarknoten geschickt. Dabei handelt es sich um I-Frames mit dem PID „CF". Im Netzwerk-Kontrollblock ist als Absender das Rufzeichen des Knotens eingetragen, der die Laufzeitmessung vornimmt. Als Zielknoten ist das Phantom-Rufzeichen L3RTT (wie „Layer 3 Round Trip Time") eingetragen. Der Transport- Kontrollblock ist mit 5 Bytes „00" besetzt. Im Informationsteil sind weitere Daten wie z.B. die Systemzeit beim Erzeugen des Frames enthalten. Da diese Daten nur beim Absenderknoten ausgewertet werden, ist das genaue Datenformat nicht wichtig. Ausserdem ist dieser Teil des Protokolls noch in der Entwicklungsphase, so dass Änderungen hier nicht auszuschliessen sind.

Automatisches Routing (alter Art).

Diese Art des Routings wird noch für Knoten bereitgestellt, die noch nicht das Protokoll von ON5ZS unterstützen.

Für das automatische Routing werden regelmässig von den Netzknoten Informationen über den Inhalt der Zielknotenliste ausgesendet. Dies erfolgt durch UI-Frames mit dem TheNet-PID „CF". Das Absender-Rufzeichen dieser Routingframes (Broadcast) ist das Rufzeichen des absendenden Netzknotens, und als Zielrufzeichen wird das Phantom-Rufzeichen „NODES" verwendet. Für TheNetNode ist das Zielrufzeichen nicht mehr vom Programm vorgegeben, um auch über spezielle Layer 2 Digipeater (z.B. FlexNet) routen zu können.

Im Info-Feld des Broadcast-Frames wird zuerst als Kennung ein Byte „FF" gesendet, gefolgt von dem Ident des Absenderknotens (6 Bytes). Schliesslich folgen bis zu 11 Einträge der Zielknotenliste.

Jeder Zielknoteneintrag besteht aus 4 Datenblöcken. Der erste Datenblock enthält das Rufzeichen des Zielknotens (7 Bytes). Der zweite Datenblock beinhaltet den Ident des Zielknotens (6 Bytes). Im dritten Datenblock folgt der Nachbarknoten, über den der Zielknoten mit der besten Laufzeit bekannt ist (7 Bytes). Schliesslich wird im vierten Datenblock die Laufzeit des besten Weges zum Zielknoten angegeben (1 Byte).

Alle angegebenen Idents werden als normale ASCII-Zeichen übertragen, während für alle Rufzeichen das AX.25-Format (ASCII um 1 Bit nach links geschoben) verwendet wird.

Bei neueren TheNet-Versionen ist die Aussendung der Zielknotenliste erweitert um die Layer 3 Lifetime der einzelnen Zielknoten. Damit diese Erweiterung kompatibel zu den älteren Versionen bleibt, wurde der zusätzliche Teil an die bisherige Aussendung angehängt. Als Kennung für den Beginn des Lifetimeblocks wird ein Byte „00" gesendet. Diese Kennung verhindert gleichzeitig eine weitere Auswertung des Frames durch ältere Programme, da sonst statt der Lifetimes ein weiterer Zielknoten folgen müsste, beginnend mit einem Rufzeichen. Rufzeichen können aber nicht mit „00" anfangen.

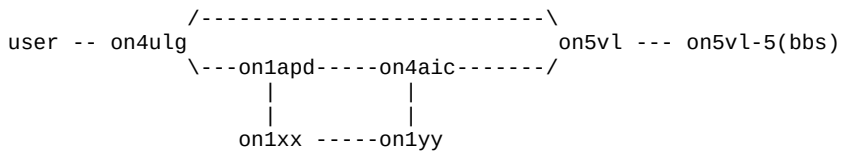
Darauf folgt die eigene Layer 3 Lifetime des Absenderknotens. Anschliessend werden die Lifetimes der in dem Frame enthaltenen Zielknoten in der Reihenfolge ausgesendet, in der vorher die Zielknoten ausgesendet wurden.

Für die Lifetime wird jeweils 1 Byte gesendet und zwar diejenige Lifetime, die auch in einem Frame des entsprechenden Zielknotens eingetragen wäre. Ist die Lifetime unbekannt, wird statt dessen „00" gesendet.

Neues Routing in TheNetNode von ON5ZS:

Routeränderungen:
Der BROADCAST wird nun im sicheren Level 3 übertragen. So wie ein Node verschwindet, wird seine Qualität auf „1" gesetzt. Die Nachbarn werden SOFORT durch den BROADCAST darüber verständigt. Diese verständigen auch sofort ihre andere Nachbarn und so weiter. Damit wird das gesamte Netz davon benachrichtigt. Dauer der Ausbreitung: zwischen 0 und 4 Sekunden pro Hop.

Ein neues System:
1 Strecke und 2 Wege. Das System ist jetzt soweit klar.



Die Lösung war ziemlich einfach...

Damit on5vl den Weg on5vl, on4aic, on1apd nimmt, muss bei on4aic der beste Weg nach on4ulg über on1apd eingetragen sein. Durch die Art der Berechnung der Qualität, die Qualität von on4aic bei on5vl ist gleich der Qualität von on5vl bei on4aic, kann on5vl sehr einfach die Qualität der Strecken von on4aic berechnen (und entscheiden, ob on4ulg direkt angesprochen wird oder nicht).

a) Wenn es möglich ist (kein ping-pong -> Berechnung der Qualitäten des Nachbarn).

b) Der alternative Weg wird gewählt, wenn die Übertragungszeit eines Frames auf der alternativen Strecke kürzer war.

c) Wenn ein Frame vom Level 4 kommt (on4aic -> on1yy wird niemals gebraucht).

Die alternative Strecke wird beansprucht ab dem Augenblick, wo eine bestimmte Anzahl (*) von Frames im tx_buffer level2 (-> tosend) bei on5vl -> on4ulg vorhanden ist.

(*) Diese Anzahl verändert sich nach der Qualitätsdifferenz zwischen Hauptstrecke und alternativer Strecke in der Art, wie die Frames in etwa in der richtigen Reihenfolge bei on4ulg ankommen (TNN gibt sie in der richtigen Reihenfolge weiter).

Es gibt kein „Selective retransmission request" von on4ulg.

Transport Layer Frames.

Zwischen den Endknoten eines Circuit wird ein spezielles Übertragungsprotokoll verwendet, um eine transparente, fehlerfreie Datenübertragung zu ermöglichen. Dieses zusätzliche Protokoll wird im Transport-Kontrollblock und im Informationsteil des Frames übertragen.

Der Transport-Kontrollblock besteht aus 5 Bytes. Die Bedeutung der ersten 4 Bytes ist abhängig vom 5. Byte, dem Kontrollbyte. Das Kontrollbyte selbst besteht aus dem Opcode sowie 3 Flags.

Die Bits 0-2 im Kontrollbyte bilden den Opcode, aus dem die Frameart abzuleiten ist. Von den 8 möglichen Opcodes werden derzeit nur 6 verschiedene Möglichkeiten verwendet:

1 = Connect-Request,
2 = Connect-Acknowledge,
3 = Disconnect-Request,
4 = Disconnect-Acknowledge,
5 = Information-Transfer,
6 = Information-Acknowledge.

Die Bits 3 und 4 werden derzeit (noch) nicht verwendet, sie werden daher immer auf 0 gesetzt.

Bit 5 ist das More-Follows-Flag. Dieses Flag zeigt an, dass der Informationsteil auf mehrere Frames aufgeteilt wurde. Um auch Frames des Enduser mit mehr als 236 Byte Informationsteil transparent zu übertragen, werden solche Frames aufgeteilt. Im ersten Teil werden die maximal möglichen 236 Bytes Daten übertragen, im nächsten Frame folgt der Rest. Bei älteren Versionen von TheNet wurden Informationen des Knotens selbst (Userliste, Nodesliste) zur Übertragung innerhalb des Netzes teilweise auch mit wesentlich grösserer Anzahl an Datenbytes übertragen. Dafür wurden dann entsprechend mehrere Frames mit gesetztem More-Follows-Flag nacheinander gesendet. Beim Endknoten wurde dann die Unterteilung in AX.25-gerechte 256-Byte-Frames vorgenommen.

Bit 6 heisst NAK-Flag. Dieses Bit wird nur bei den Frametypen Information-Transfer und Information-Acknowledge verwendet und zeigt an, dass ein Frame wiederholt werden soll.

Bit 7 ist das Choke-Flag. Dieses Flag wird bei den Frametype Information-Transfer, Information-Acknowledge und Connect-Acknowledge verwendet. Die unterschiedliche Bedeutung dieses Flags ist bei den entsprechenden Frametypen erläutert.

Connect-Request.

Mit einem Connect-Request-Frame wird ein neuer Circuit angefordert. Die ersten beiden Bytes des Transport-Kontrollblocks spezifizieren den Circuit des Absenderknotens. Dabei gibt das erste Byte einen Index in der Circuit-Tabelle an (Circuit-Index). Das 2. Byte wird als zusätzliche Kennzeichnung für den jeweiligen Circuit verwendet, um Mehrdeutigkeiten zu verhindern (Circuit-ID). Die folgenden 2 Bytes werden nicht verwendet und daher auf „00" gesetzt. Auch die Flags im Kontrollbyte werden nicht verwendet und daher auf 0 gesetzt.

Im Info-Teil des Connect-Request-Frames wird zuerst die vorgeschlagene Fenstergrösse für den Transport-Layer übertragen (1 Byte). Darauf folgt das Rufzeichen des Users, der den neuen Circuit aufzubauen wünscht. Schliesslich folgt das Rufzeichen des Absenderknotens. Zusammen sind das also 15 Bytes im Info-Teil des Connect-Request-Frames.

Bei neueren Versionen von TheNet wird zusätzlich übertragen, bei welchem Netzknoten der User in das Netz eingestiegen ist, und über welchen Digipeaterweg er dort zu erreichen ist. Zuerst wird hierfür das Rufzeichen des Netzknotens übertragen, bei dem der User in das Netz eingestiegen ist. Danach folgt die eventuell vorhandene Digipeaterliste beim Uplink, also maximal 8 weitere Rufzeichen. Zum Abschluss folgt 1 Byte „00". Für diese Erweiterung werden also minimal 8 und maximal 64 Bytes zusätzlich im Informationsteil übertragen.

Für alle beim Connect-Request übertragenen Rufzeichen wird das um 1 Bit nach links geschobene AX.25-Rufzeichen-Format verwendet.

Connect-Acknowledge.

Auf ein Connect-Request-Frame wird mit einem Connect-Acknowledge-Frame geantwortet. Die ersten beiden Bytes des Transport-Kontrollblocks bezeichnen den Circuit des Knotens, der den Connect-Request gesendet hat. Sie werden direkt aus dem Connect-Request übernommen. Die darauf folgenden beiden Bytes bezeichnen den Circuit des Knotens, der den Connect-Acknowledge-Frame sendet, wieder in der Reihenfolge Circuit-Index - Circuit-ID, wie beim Connect-Request-Frame.

Wenn der Connect-Request akzeptiert wurde, sind alle Flags im Kontrollbyte gelöscht. Falls kein Circuit aufgebaut werden kann, ist das Choke-Flag (BIT 7) gesetzt.

Mit einem Connect-Request-Frame wird ein neuer Circuit angefordert. Die ersten beiden Bytes des Transport-Kontrollblocks spezifizieren den Circuit des Absenderknotens. Dabei gibt das erste Byte einen Index in der Circuit-Tabelle an (Circuit-Index). Das 2. Byte wird als zusätzliche Kennzeichnung für den jeweiligen Circuit verwendet, um Mehrdeutigkeiten zu verhindern (Circuit-ID). Die folgenden 2 Bytes werden nicht verwendet und daher auf „00" gesetzt. Auch die Flags im Kontrollbyte werden nicht verwendet und daher auf 0 gesetzt. Im Info-Teil des Connect-Request-Frames wird zuerst die vorgeschlagene Fenstergrösse für den Transport-Layer übertragen (1 Byte). Darauf folgt das Rufzeichen des Users, der den neuen Circuit aufzubauen wünscht. Schliesslich folgt das Rufzeichen des Absenderknotens. Zusammen sind das also 15 Bytes im Info-Teil des Connect-Request-Frames.

Im Info-Teil des Connect-Acknowledge-Frames wird 1 Byte übertragen für die zu verwendende Fenstergrösse im Layer 4.

Disconnect-Request.

Wenn ein Information-Transfer-Frame empfangen wird und keine Information für die Gegenstation vorliegt, wird der Empfang mit einem Information-Acknowledge-Frame bestätigt. Die ersten beiden Bytes des Transport-Kontrollblocks bezeichnen wieder den Circuit des Knotens, an den der Information-Acknowledge-Frame gesendet wird, in der Reihenfolge Circuit-Index - Circuit-ID. Das dritte Byte des Information-Acknowledge-Frames ist nicht verwendet und wird daher auf „00" gesetzt. Das 4. Byte des Transport-Kontrollblocks ist die Empfangs-Sequenz-Nummer und gibt die als nächstes von der Gegenstation erwartete Sendesequenznummer an.

Ein Disconnect-Request-Frame enthält keinen Info-Teil.

Disconnect-Acknowledge.

Als Antwort auf ein Disconnect-Request-Frame wird ein Disconnect-Acknowledge-Frame gesendet. Die ersten beiden Bytes bezeichnen wieder den Circuit des Knotens, an den der Disconnect-Acknowledge-Frame gesendet wird, in der Reihenfolge Circuit-Index - Circuit-ID. Die folgenden beiden Bytes werden nicht verwendet und deshalb auf „00" gesetzt. Auch die Flags im Kontrollbyte werden nicht verwendet und daher gelöscht.

Ein Disconnect-Acknowledge-Frame enthält keinen Info-Teil.

Information-Transfer.

Ein Information-Transfer-Frame überträgt die Informationen des Users. Die ersten beiden Bytes bezeichnen den Circuit des Knotens, an den der Information-Transfer-Frame gesendet wird, in der Reihenfolge Circuit-Index - Circuit-ID. Darauf folgt ein Byte für die Sende-Sequenz-Nummer dieses Frames. Das 4. Byte des Transport-Kontrollblocks ist die Empfangs-Sequenz-Nummer und gibt die als nächstes von der Gegenstation erwartete Sendesequenznummer an. Damit werden gleichzeitig mit der Informationsübertragung die von der Gegenstation empfangenen Frames bestätigt.

Ist das More-Follows-Flag gesetzt, bedeutet dies, dass nicht die vollständige Information eines User-Frames auf einmal übertragen werden konnte, da im Information-Transfer-Frame höchstens 236 Bytes Info enthalten sein können (die restlichen 20 Bytes werden für Netzwerk - und Transport-Kontrollblock benötigt). Im nächsten Frame ist die Fortsetzung der User-Information enthalten.

Das NAK-Flag wird gesetzt, wenn ein empfangenes Information-Transfer-Frame nicht die erwartete Sendesequenznummer hatte. Damit wird die Wiederholung des fehlenden Frames angefordert.

Ist das Choke-Flag gesetzt, bedeutet dies, dass vorerst keine weitere Information für diesen Circuit angenommen werden kann.

Im Info-Teil des Information-Transfer-Frames wird die User-Information übertragen (max. 236 Bytes je Frame).

Information-Acknowledge.

Wenn ein Information-Transfer-Frame empfangen wird und keine Information für die Gegenstation vorliegt, wird der Empfang mit einem Information-Acknowledge-Frame bestätigt. Die ersten beiden Bytes des Transport-Kontrollblocks bezeichnen wieder den Circuit des Knotens, an den der Information-Acknowledge-Frame gesendet wird, in der Reihenfolge Circuit-Index - Circuit-ID. Das dritte Byte Des Information-Acknowledge-Frames ist nicht verwendet und wird daher auf „00" gesetzt. Das 4. Byte des Transport-Kontrollblocks ist die Empfangs-Sequenz-Nummer und gibt die als nächstes von der Gegenstation erwartete Sendesequenznummer an.

Das NAK-Flag wird gesetzt, wenn ein empfangenes Information-Transfer-Frame nicht die erwartete Sendesequenznummer hatte. Damit wird die Wiederholung des fehlenden Frames angefordert.

Ist das Choke-Flag gesetzt, bedeutet dies, dass vorerst keine weitere Information für diesen Circuit angenommen werden kann. Das More-Follows-Flag wird nicht verwendet und ist daher gelöscht. Ein Information-Acknowledge-Frame enthält keinen Info-Teil.

DF6LN (23.7.93); erweitert um ON5ZS-Router ON5ZS/DG9FU (9.9.96)

FlexNet-Interface:

Allgemeines:

Die vorliegende Beschreibung dokumentiert die Implementation der FlexNet-Gateway-Routinen in TNN. Bei der Erarbeitung der Grundlagen bestanden keinerlei Absichten, den FlexNet-Router zu kopieren, zu verbessern oder zu übernehmen. Ziel ist und war, eine engere Verknüpfung der TNN-Digipeater und FlexNet-Digipeater auf Systemebene zu erreichen und Network-Informationen gegenseitig auszutauschen.

Bis dato konnten die FlexNet - und Nord><Link Gruppen kein gemeinsames Interlink-Protokoll ausarbeiten. Aus diesem Grunde haben wir nun versucht, die Interlink-Kommunikation der FlexNet-Schnittstelle teilweise nachzubilden und damit eine vereinfachte Gateway-Schnittstelle zu realisieren.

Grenzen der Implementation:

Da die FlexNet-Schnittstelle nicht (öffentlich) dokumentiert ist, übernehmen wir keinerlei Funktionsgarantie. Aus den oben erwähnten Gründen bestand sogar die Absicht einer eingeschränkten Emulation. Bei Bedarf (und weitergehenden Informationen) kann die Schnittstelle zu einem späteren Zeitpunkt erweitert werden.

Realisiert wurden folgende Funktionen:

- Initialisierung,
- Laufzeitmessung zum Nachbarn,
- Beantwortung der Laufzeitmessung des Nachbarn,
- Auswertung der Destinations-Informationen,
- Rückmeldung bekannter Destinations,
- Find Destinations,.

Nicht realisiert sind folgende Funktionen:

- Header Kompression.

Diese Version kann nun beliebig viele FlexNet-Nachbarn verwalten. Laufzeitmessung und Destinations-Übernahme erfolgen dynamisch. Die empfangenen FlexNet-Ziele werden in der gleichen Tabelle gespeichert wie die Nodes, jedoch keine Nodes an FlexNet gemeldet. Verfügt ein TNN-Knoten allerdings über mehrere FlexNet-Nachbarn, so werden zwischen diesen Destinations ausgetauscht.

ANHANG G:

Die GO32-Version von TNN

Einleitung:

Unter MS-DOS hatten wir bisher immer damit zu kämpfen, dass der verfügbare Speicher auf 1 MB (+ HMA) beschränkt war. Diesen Speicher mussten sich alle Programme teilen. Damit waren Programme wie TNN in einen Speicherbereich eingezwängt.

Mit der GO32-Version steht nun der gesamte Speicher zur Verfügung. Aber warum noch MS-DOS unterstützen, wo es so gute Betriebssystem wie OS/2, LINUX oder WINDOWS gibt? Alle diese Betriebssysteme haben einen recht hohen Speicherbedarf, außerdem schleppt man meist eine für unsere Zwecke unnötige Oberfläche mit. Deshalb die weitere Unterstützung von MS-DOS.

Was bedeutet DPMI?

Die ersten Prozessoren der Firma INTEL konnten 1 MB Speicher adressieren, dies war zu damaligen Zeiten eine Menge. Mit dem Boom der Computertechnik wurde schnell erkannt, dass diese Grenze schnell gesprengt würde. Deshalb implementierte INTEL eine CPU-Betriebsart, den Protected Mode. Mit dieser Methode konnte man auf einem 286'er (damals) unglaubliche 16 MB adressieren. Leider war man nicht recht konsequent mit der Umsetzung, so konnte man die CPU ohne weiteres in den PM (Protected Mode) schalten, aber zurückschalten in den Real Mode, so bezeichnet man den 8086 kompatiblen Modus, war nur mit großem Aufwand möglich. Erst mit dem 386'er wurde dieses Manko behoben. Außerdem wurde der Protected-Mode noch mal erweitert, so dass bis zu 4 GByte (32 Bit Selektor und 32 Bit Offset) adressieren kann.

Leider hatte dies keine Vorteile für ein Betriebssystem wie MS-DOS, durch die spezielle Art des PM würde es keine Abwärtskompatibilität geben. Um trotzdem auf den Speicher über 1 MB zugreifen zu können, wurden Krücken wie LIM-EMS oder XMS eingeführt. Hierbei wird ab dem 386'er ein weiterer Modus des 386'er ausgenutzt, der Virtuelle-8086-Modus, auf diesen soll hier aber nicht weiter eingegangen werden.

Um dieses Problem zu beheben, wurde das Dos-Protected-Mode-Interface definiert. Damit ist es möglich, Protected-Mode-Programme auch unter MS-DOS auszuführen, ohne gleich ein neues Betriebssystem implementieren zu müssen.

Der DPMI-Server schalten den Prozessor (ab 386'er) immer dann in den Real-Mode zurück, wenn z.B. eine DOS-Routine oder ein BIOS-Aufruf stattfinden. Danach wird die CPU wieder in den PM versetzt. Außerdem stellt der DPMI-Server eine gewisse Anzahl an Funktionen zur Verfügung, die das Programmieren im PM erleichtern.

Warum Protected?

Adresse zu bilden.

Der Protected-Mode ist eigentlich für Multitasking-Systeme gedacht. Durch diesen Modus bietet die CPU Möglichkeiten, einzelne TASK voreinander zu schützen d.h. ein Programm kann nicht in ein anderes eingreifen. Dies ist aber nur durch einen gewissen Aufwand zu erreichen. So zeigt ein Pointer nicht auf eine physikalische Adresse, sondern enthält einen Selektor und einen Offset. Zum Generieren einer physikalischen Adresse im PM wird zum Selektorteil, der logischen Adresse, als Index in eine Deskriptortabelle gesprungen. Der Eintrag in die Deskriptortabelle enthält eine 24-Bit Basisadresse, welcher der Offset-Teil der logischen Adresse hinzuaddiert wird, um die physikalische Adresse zu bilden.

Im DPMI 32-bit ist dies ähnlich, allerdings werden dort 32-Bit-Offsets verwendet (4 GByte), die einen Bezug zu einem 4G-langen-Segment herstellen können. Versucht jetzt einen Anwendung auf Daten jenseits einer Segmentgrenze zuzugreifen, so gibt es eine allgemeine Schutzverletzung (GBP oder General Protection fault), daher Protected Mode. In DPMI-Version von TNN wird der 16-Bit-DPMI verwendet.

Was ändert sich für den Nutzer?

Erstmal nichts, die DPMI-Version bietet erstmal die gleiche Funktionalität wie die „normale" Version. Allerdings ist die Systemanforderung etwas höher. Durch das Umschalten zwischen PM und Real-Mode ist eine gewisse Geschwindigkeitseinbusse zu bemerken. Auch sollten min. 3-4 MB-RAM verfügbar sein. Für DOS-Aufrufe stehen dann allerdings mehr als 400 KByte zur Verfügung und die Buffer können etwas großzügiger bemessen werden. Auftretende Fehler (Exceptions) werden auf der Festplatte im TEXTCMD-Pfad verewigt und sollten an eine NORD><LINK-Service-Stelle (z.B. DG9OBU oder DG8BR) weitergeleitet werden, damit wir die Möglichkeit haben, den Fehler ausfindig zu machen.

Übrigens verträgt sich die DPMI-Version bestens mit OS/2 oder WIN 3.1x / WIN 95.

GRENZEN / PROBLEME:

Bedingt durch den Aufbau des DPMI gibt es einen ganz großen Haken. Wie oben bereits beschrieben muss die CPU fast laufend zwischen dem Protected Mode und dem Real Mode hin und herzuschalten. Dieses kostet natürlich Zeit.

Deshalb sind einige Sachen zu beachten:

KISSLINK

Nur mit Fifo, ohne hat es nicht viel Sinn!!

BAYCOM-USCC
: Bedingt durch den Aufbau der Karte werden sehr viele Interrupts erzeugt! Vier 9k6-Links entsprechen ca. einem 38k4 KISSLINK ohne Fifo! Die USCC-Karte verfügt zwar auch über einen kleinen Fifo, ob der aber reicht ist fraglich!

VANESSA
: Super für DPMI, da die VANESSA keine Interrupts erzeugen!

ETHERNET
: Macht natürlich IRQ ...ist aber bei heutigen Rechnern nicht kritisch.....

: Einen 486DX2-66 oder höher sollte aber wärmstens empfohlen werden.

DEBUG.TXT:

Bei Abstürzen / Exceptions:
Wird AUF DEM BILDSCHIRM eine Debug-Ausgabe erzeugt. SOFORT NACH der TNN17x.EXE sollte das Programm „SYMIFY -O DEBUG.TXT TNN.EXE" aufgerufen werden, dieses wertet den Stack aus und generiert die eigentliche Debug-Meldung, diese bitte an eine TNN-Service-Stelle weiterleiten. Der Sysop ist für einen Reset danach, selbst verantwortlich! Dies kann entweder ein Hardware-Watchog sein oder eine Datei in der START.BAT die nach dem Aufruf der TNN-EXE und SYMIFY steht, z.B. RESET.COM. Leider war eine andere Lösung bis jetzt nicht möglich!

: Mögliche Exceptions:

: 08 = Doppelfehler,
: 0A = Ungültiges TSS,
: 0B = Segment nicht vorhanden,
: 0C = Stack-Fehler,
: 0D = Allgemeine Schutzverletzung,
: 0E = Seitenfehler.

: Weitere Hinweise zur GO32-Version:

- die TNN179.STA/MHEARD.TAB/CONVERS.PRS der 32bit-Version unterscheiden sich durch das DWORD-Alignment des GNU-Compilers zu den DOS/DPMI-Versionen
- die 32bit-Version generiert keine Resets sondern verlässt das Programm, in einer Batch-Datei ist dann nach der TNN.EXE ein Programm aufzurufen, das den Reset durchführt. Zwischen TNN.EXE und RESET.EXE ist das „SYMIFY" einzufügen (s.u.)

Compiler:

Als Compiler kommt der DJGPP/GNU zum Einsatz. Dieser Compiler ist frei verfügbar.

Weiter sind die folgenden Programme erforderlich:

SYMIFY.EXE
: Zum Auslesen des Speichers bei einem Programmfehler.

CWSDPMI.EXE
: Der nötige DPMI-Handler incl. CWSDPMI.DOC, CWSDMP0.EXE CWSPARAM.EXE.

RESET.EXE
: damit die Software auch sofort wieder gestartet wird.

Die START.BAT könnte nun so aussehen:

```
set tz=UTC0
set conversd=P:1
set tnn32buffers=9000
set tnnpath=c:\tnn\
set tnncfg=1009,32
set msgpath=c:\tnn\msg\
set confpath=c:\tnn\
set digicall=DB0EAM
set satqth=JO41PI
set qth=9:16:12/51:21:10
set tokencom=0
set kiss1=1
c:\b-log.exe c:\tnn\sysop.pro -m

STARTCNT
IF ERRORLEVEL==1 GOTO OKGO32
ECHO Starte alte TNN178...

REM !!!!!!!!!!!!!! ALTE SOFT HIER STARTEN (TNN178.EXE) !!!!!!!!!!!!!!!!!!
COPY TNN177.DOS TNN178.TNB
TNN_ALT.exe
REM !!!!!!!!!!!!!! ALTE SOFT HIER STARTEN (TNN178.EXE) !!!!!!!!!!!!!!!!!!

GOTO ENDE
:OKGO32

ECHO Starte neue TNNGO32.EXE

REM !!!!!!!!!!!!!! HIER WIRD DIE NEUE SOFTWARE GESTARTET !!!!!!!!!!!!!!!!
COPY TNN179.DPI TNN179.TNB
TNNGO32.exe
SYMIFY -o DEBUG.ERR TNNGO32.EXE
TYPE DEBUG.ERR >> C:\TNN\TEXTCMD\DEBUG.TXT
REM !!!!!!!!!!!!!! HIER WIRD DIE NEUE SOFTWARE GESTARTET !!!!!!!!!!!!!!!!

:ENDE
RESET.EXE
```

Weitere Fragen ?

Dann sind zu empfehlen:

- PC Intern,
- Die 386/486’er-CPU,
- http://www.delorie.com.

ANHANG H:

D A M A - ein neues Verfahren für Packet-Radio !

---

von DK4EG Detlef J. Schmidt,
Stenbrechstr. 22,
D-38106 Braunschweig

NORD><LINK e.V.
c/o Peter Gülzow, DB2OS
Allensteiner Str. 5
D-30880 LAATZEN
Germany

Immer wieder haben Packet-Radio-Amateure Probleme damit, dass sie vermeintlich nicht an 'ihren' Digipeater herankommen oder meinen, der Digipeater würde sie nicht verstehen. Und das, obwohl sie den Digi mit lautem gutem Signal aufnehmen können. Der Schluss, dass der Empfänger des Digi wohl taub sein müsse, scheint sich aufzudrängen.

Den Fall gibt es natürlich auch, er soll aber hier nicht betrachtet werden.

Wahrscheinlicher ist es, dass der Digipeater nicht 'nichts hört', sondern dass er 'viel zu viel hört'. Er hat ja typischerweise den 'besten' Standort aller beteiligten Stationen. Das führt dann zu Kollisionen beim Digi mit mehreren Stationen, die sich untereinander nicht hören ('Hidden-Stations-Problem'). Nur wenn eines der gleichzeitig empfangenen Signale deutlich stärker beim Digi ankommt als alle anderen, hat es Chancen 'durchzukommen'. Für weiter entfernt liegende Stationen ist es dann oft unmöglich, mit ihren Packets zum Digi durchzudringen.

Verschiedene Versuche sind schon unternommen worden, um dieses Problem auf den Amateurbändern zu lösen. Ein Lösungsansatz ist zum Beispiel der Einsatz von Duplex-Digis (BTMA). Aber auch dieses System hat etliche Nachteile. Der Hardwareaufwand eines Duplex-Digis ist um einiges höher als bei einem Simplex-Digi. Ausserdem belegt er z w e i Kanäle, bringt aber nur den Durchsatz e i n e s kollisionsfreien Kanals. Eine Durchsatzerhöhung tritt nur statistisch durch Kollisionsreduzierung auf.

Eleganter wäre es aber wohl, wenn man z.B. durch einfachen EPROM-Tausch oder noch besser nur durch eine andere Parametereinstellung das Problem lösen könnte.

Dieses Problem der Packet-Radio-Amateure ist nun aber gar nicht so neu. Es gibt auch andere Funkdienste, die das gleiche Problem haben, z.B., wenn von Schiffen auf hoher See ein Kommunikationssatellit angesprochen werden soll.

Ein etabliertes Verfahren, das dieses Kollisionsproblem löst, ist z.B. das DAMA-Verfahren. Es ist die Abkürzung von: Demand Assigned Multiple Ales, also etwa anforderungsbezogener Vielfachzugriff (auf den Übertragungskanal). Vereinzelt taucht in der Literatur auch der treffendere Begriff CODAC dafür auf (Carrier Oriented and Demand Assigned Control). Der Begriff ist aber wohl wegen der phonetischen Verwechslungsmöglichkeit mit einem Firmennamen aufgegeben worden.

Der Ablauf dieses etablierten DAMA ist etwa folgender: In einem Connection oriented protokoll wird zunächst nach dem Slotted-Aloha-Verfahren versucht, den Satelliten zu connecten. Kollisionen sind selten, werden aber in dieser Phase in Kauf genommen. Danach kennt der Satellit die Station und nimmt sie in seine Polling-Liste auf. Es werden also alle Stationen reihum vom Satelliten aus aufgerufen (gepollt), wobei der Aufruf gleichzeitig die Bestätigung der empfangenen Packets ist.

Ist die Station einmal connected, sendet sie ihre Datenblöcke (I-Frames) nur noch nach Aufforderung durch einen Poll. Es können auch mehrere Frames in einem Block gesendet werden. Antwortet der Endknoten (User) nicht unmittelbar innerhalb einer vorgegebenen Zeit, wird angenommen, dass der Poll nicht angekommen ist, und im nächsten Durchlauf wird sofort der Poll wiederholt.

Desgleichen wird nach Empfang von Nutzdaten-Blöcken (I-Frames) in der nächsten Runde (also wenn alle anderen Stationen der Connect-Liste abgearbeitet sind) mit dem Poll die Bestätigung geschickt. Kommt dagegen vom Endknoten nur eine leere Bestätigung an (Receive Ready/Final), so wird er in der nächsten Runde übergangen.

Mit zunehmender Belegung des Übertragungskanals kann eine momentan nicht aktive Station noch weiter in der Poll-Priorität heruntergesetzt werden, erlangt aber sofort wieder die höchste Priorität, wenn sie mal mit einem (oder mehreren) I-Frames antwortet.

Liest man sich die vorige Protokollbeschreibung durch, glaubt man fast, das AX.25-Level-2-Protokoll darin zu erkennen. Und darin liegt wohl auch die Chance und die Möglichkeit des DAMA-Verfahrens für Amateur-Packet-Radio. AX.25-L2 bietet unmittelbar alle Elemente, die für DAMA notwendig sind. Es müssen keine neuen Syntaxelemente eingeführt werden. Viele Funktionen lassen sich bereits durch Verstellen der Betriebsparameter erreichen. Der Rest ist zum Teil nur eine minimale Änderung der Statetable in der Firmware.

Wie könnte nun eine DAMA-Version für Amateur-Packet-Radio aussehen?

Da bereits alle Syntaxelemente vorhanden sind, müssen auch keine neuen eingeführt und erklärt werden. Wir bleiben daher für die Beschreibung in der bisherigen Terminologie von AX.25.

Die verschiedenen Phasen des Protokolls sollen hier einzeln betrachtet werden.

Verbindungsaufbau

Soll via Digi ein Endknoten connected werden, so wird die neue Station direkt in die Poll-Liste aufgenommen und mit SABM-Frames angepollt. Erfolgt innerhalb einer vorgegebenen Anzahl von SABM kein UA des angesprochenen Endknotens, wird die Station wieder aus der Poll-Liste gestrichen. Geht die Initiative umgekehrt vom Endknoten aus, wird wie bisher im CSMA-Verfahren ein SABM an den Netzknoten gesendet. In dieser Phase sind Kollisionen möglich, es ist also eventuell notwendig, nach einem vorgegebenen Timeout wie bisher weitere SABM zu senden, bis der Netzknoten mit einem UA antwortet.

Damit ist der Endknoten in die Polling-Liste (Userliste bei TheNet) aufgenommen und wird ab diesem Zeitpunkt koordiniert. Der Endknoten antwortet mit einem RR count 0 als Signal an den Digi, dass sein UA aufgenommen wurde.

Idle

Solange kein Informationstransfer zwischen Netz- und Endknoten stattfindet (Ideln), sendet der Netzknoten seine Polls mit RR und dem zugehörigen aktuellen Count. Kommt als Antwort des Endknotens beim Digi ein RR mit dem zugehörigen Count an, dann wird die Zeit bis zum nächsten Poll an denselben Endknoten verlängert, um den Kanal nicht unnötig mit Polls zu belasten. Wird das RR des Endknotens nicht beim Digi empfangen (weil entweder der Poll oder die Antwort verloren ging), dann wird derselbe Endknoten beim nächsten Durchlauf (also wenn alle anderen Uplink-Stationen einmal abgearbeitet wurden) sofort wieder angepollt usw.

Wenn nach einer vorgegebenen Anzahl von Polls keine Antwort empfangen wird, kann der Endknoten aus der Poll-Liste gelöscht werden. Dieses Verfahren stellt nur auf den ersten Blick mehr Transferaufwand dar als in der bisherigen Version, da auch nun schon die Gegenstation routinemäßig angepollt und irgendwann abgehängt wird. Der Unterschied ist nur, dass in der neuen Version der Endknoten früher aus der Userliste gestrichen wird, wenn er nicht mehr antwortet, bei gleich bleibender Anzahl von Versuchen.

Nutzdaten-Transfer Netzknoten-Endknoten

Dieser Fall unterscheidet sich in nichts vom ‘normalen’ CSMA-Verfahren. Da die Initiative ohnehin beim Digi liegt (Master), kann er auch zu jeder Zeit statt eines Polls ein oder mehrere I-Frames an den Endknoten (Slave) senden. Der Endknoten bestätigt wie gehabt mit RR und dem zugehörigen Count, er kann aber auch direkt mit I-Frames und den korrespondierenden Counts bestätigen. Auch die Anwendung des Poll-/Final-Bits bleibt unverändert.

Nutzdaten-Transfer Endknoten-Netzknoten

Wie schon zuvor skizziert, sendet der Netzknoten reihum Polls an alle in der Userliste verzeichnete Endknoten. Der Endknoten wartet also, bis er vom Digi mit einem RR angepollt wird oder ihm ein I-Frame gesendet wird und antwortet dann unmittelbar mit einem oder mehreren I-Frames zum Netzknoten. Dieses Warten auf den Poll macht den entscheidenden Aspekt der Kollisionsverhütung aus im Gegensatz zum bisherigen Verfahren, bei dem mehrere Endknoten quasi gleichzeitig senden können, weil sie sich untereinander nicht hören und nicht koordiniert sind.

Zusätzlich wird das Problem behoben, dass zwei Endknoten senden, die sich eigentlich hören könnten, aber in der jeweils eigenen Totzeit zwischen dem Abschalten des Empfängers und dem Abstrahlen des modulierten Trägers die andere Station nicht ‘hören’. Dieser Fall ist gar nicht so selten, da verschiedene sendebereite Stationen ja von dem Träger des Digi untereinander synchronisiert werden.

Der Digi wird die gesendeten I-Frames nicht sofort bestätigen, sondern erst in der nächsten Runde, nachdem alle anderen Stationen der Liste bedient wurden. Dies ist dann auch gleichzeitig der nächste Poll. Stehen keine I-Frames zum Senden an, wird vom Endknoten wieder nur ein RR mit Count geschickt.

Verbindungsabbau

Es kommen wieder dieselben Syntaxelemente zum Einsatz. Will der Netzknoten (Digi) den Endknoten (User) abwerfen, so schickt er einfach wie bisher ein DISC-Frame. Der Endknoten antwortet darauf unmittelbar mit einem UA (Finalbit gesetzt). geht das UA verloren, antwortet der Endknoten beim nächsten eintreffenden DISC-Frame wie bisher mit einem DM-Frame. Will im umgekehrten Fall der Endknoten ‘aussteigen’, so wartet er den nächsten Poll ab und sendet daraufhin sein DISC-Frame an den Netzknoten. In diesem Falle ist es unerheblich, ob der Digi sofort oder erst in der nächsten Runde mit UA bestätigt.

UI-Frames

Eine gewisse Sonderstellung nehmen UI-Frames ein (Unnumbered Information), sowohl im normalen CSMA-Verfahren als auch im DAMA-CSMA-Verfahren. Sie sind ursprünglich einmal dafür kreiert worden, einen Informationsaustausch außerhalb des normalen Protokollablaufes zu realisieren.

Werden UI-Frames vom Endknoten gesendet, so kann dies normalerweise nur außerhalb des Betriebs mit dem Digi auftreten. Eigentlich sollte es diese Fälle ja nicht geben, da in diesem Falle der Direkt-QSO nicht die Digi-Einstiegsfrequenz benutzt werden sollte. UI-Frames des Digi stellen ohnehin wieder kein Problem dar, da er ja von allen Stationen auf der Frequenz gehört wird.

Sonstige Protokollelemente

Somit wäre eine idealisierte Session vom Verbindungsaufbau bis zum -abbau komplett beschrieben.

Es wurden nicht alle AX.25 möglichen Protokollelemente dargestellt. Dies ist aber in diesem Kontext auch nicht zwingend notwendig, da sämtliche Elemente ihre logische Bedeutung behalten. DM, RNR, REJ werden in dem gleichen Sinne benutzt wie bisher. Der Unterschied zur reinen CSMA-Version ist lediglich, dass der Endknoten sie wie die anderen Protokollelemente nur nach einem Poll des Digi sendet.

Netzknotenseitig gibt es nur den Unterschied, dass diese entsprechenden Frames nicht sofort oder zu beliebiger Zeit, sondern immer erst dann gesendet werden, wenn alle anderen registrierten Stationen (im gleichen Sinne) einmal abgearbeitet wurden. und genau der im Frame adressierte Endknoten angesprochen werden soll.

Verträglichkeit der Protokollversionen

Eine mögliche Einführung des DAMA-CSMA-Verfahrens könnte sich kontinuierlich vollziehen. Natürlich wird die Durchsatzsteigerung umso größer, je mehr Stationen auf das neue Verfahren einsteigen. Aber selbst Stationen, die nicht auf die neue Kanal-Zugriffsmethode einsteigen, können teilweise zur Durchsatzsteigerung beitragen, indem sie ihre Parameter anders einstellen. Dazu müsste der Delay zwischen Empfang eines an sie gerichteten Frames und der zugehörigen Bestätigung (T2 oder DWAIT) auf einen Minimalwert gesetzt werden

(weniger als 1 sec).

Die Zeit bis zur Pollaussendung auf ein eigenes ausgesendetes I-Frame (um eine Bestätigung ‘anzumahnen’) wird auf einen relativ hohen Wert gesetzt, der deutlich größer ist als normalerweise der zeitliche Abstand zwischen zwei Polls des Netzknotens.

Um die vollen Möglichkeiten des Verfahrens auszuschöpfen, müssen beide Seiten die neue Protokollversion fahren. Dazu muss der Endknotenseite irgendwie mitgeteilt werden, dass sie sich nun auf die neue Protokollvariante einzustellen hat. Mehrere Methoden bieten sich an:

- Einführung eines UPLINK-Kommandos zusätzlich zum CONNECT-Kommando;

- Einführung eines kanalspezifischen Parameters im TNC, der die Protokollvariante steuert;

- automatische Erkennung der Protokollvariante mittels Protokoll-Identifier-Bytes vom Netzknoten;

- eingeschränkte Nutzung des Verfahrens durch Änderung der TNC-Timer (wie zuvor skizziert);

- Daneben ist natürlich auch noch die Einführung eines neuen Protokollelements möglich, mit der die jeweilige Variante gesteuert wird. Das würde z.B. wie bei X.25 die Funktion eines SABM-Frames darstellen. Auch damit wäre weiterhin Kompatibilität zu dem bisherigen Protokoll gewährleistet, da die ‘alten’ Protokollversionen auf unbekannte Frames nicht oder mit FRMR reagieren sollen

Zusammenfassung

Das skizzierte Verfahren kann den Durchsatz auf einem AX.25-Kanal deutlich erhöhen. Es hat die entscheidenden Vorteile, dass es kein Netzzusammenbruch durch Überlastung des Kanals gibt. Die Übertragungkapazität des Kanals steigt kontinuierlich bis zum Maximum, es gibt keinen Umkippunkt wie bei reinem CSMA, bei dem die Netto-Übertragungsrate oberhalb einer gewissen Schwelle (ca. 65 bis 85 % bei CSMA pur) wieder abnimmt.

Nur bei geringer Kanalbelastung werden bei DAMA mehr Übertragungen notwendig sein (Overhead). Dies stört aber nicht weiter, solange die Kanalkapazität nicht ausgeschöpft wird. Mit zunehmender Kanalauslastung geht der Overhaed-Anteil dann weiter zurück bis auf den minimalen Wert des idealen CSMA-Verfahrens ohne Kollisionen.

Als entscheidende ‘soziale’ Komponente gibt DAMA auch entfernteren Stationen die Möglichkeit, stabil über den Digi zu arbeiten, ohne von den näher liegenden Stationen ‘niedergebrüllt’ zu werden.

Es ist nicht notwendig, dass alle beteiligten Stationen gleichzeitig auf die neue Protokollvariante umsteigen. Alle Protokollelemente behalten ihre ursprüngliche Bedeutung, so dass die Endknoten die alte und die neue Variante nebeneinander benutzen können. Der Kanaldurchsatz wird dabei umso besser, je mehr Stationen anteilig die neue Variante benutzen.

Literatur

div. : Normblatt X.25 CCITT.
div. : Transaction on communication IEEE.
Fox, T. : AX.25 level 2 protocol specification AMRAD.
Kauffels, F.J. : Lokale Netze . R.-Müller-Verlag.
Schmidt, D.J. : Synchrone DFÜ-Protokolle mit 6809-Micro-Computern in heterogenen Sternnetzen. TU-Script BS’81.
Tanenbaum, A. : Computer Networks. Prentice Hall Verlag.
Schmidt,D.J. : DAMA, ein neues Verfahren für Packet-Radio cq-DL, 4/89

Glossar

DM	: Disconnect Mode
DISC	: Disconnect Frame
FRMR	: Frame Reject
I	: Information Frame
REJ	: Reject Frame
RNR	: Receive Not Ready
RR	: Receive Ready
SABM	: Set Asynchronous Balanced Mode
SARM	: Set Asynchronous Response Mode
UA	: Unnumbered Acknowledge
UI	: Unnumbered Information Frame
BTMA	: Busy Tone Multiple Access
CSMA	: Carrier Sense Multiple Access

Connection oriented protocoll:

Alle Knoten der Strecke kennen alle Stationen, welche die Strecke benutzen, zumindest kurzzeitig. Im Gegensatz dazu steht das Connectionless protocoll, bei dem alle Daten-Packets einfach nur weitergereicht werden (also z.B. einfaches Level-2-Digipeaten bei PR). Connection oriented Protokolle sind typischerweise aufwendiger in der Realisierung und benötigen mehr Rechnerleistung in den Netzknoten, dafür sind sie aber speziell auf Übertragungskanälen mit geringer Kapazität überlegen, da nicht in jedem Frame die gesamte Routing-Information mittransportiert werden muss, sondern nur einmal während des Verbindungsaufbaus.

Definition des DAMA-Slaves nach DG2FEF

Im Folgenden hier in Stichpunkten die Definition des DAMA-Slaves, wie er zurzeit in TF2.7b implementiert ist. Diese Definition soll in Zukunft verbindlich für die Programmierer von Layer-2-Software sein, um einen einheitlichen Standard zu schaffen, auf dessen Basis die Betriebsarten DAMA und OPTIMA (DAMA für Duplex-Digis) weiterentwickelt werden können.

1.) Als Master-Poll (bzw. DAMA-Poll) ist jedes an den Slave adressierte Frame zu verstehen, in dessen Adressfeld zugleich das DAMA-Bit, das Poll/Finalbit und das Command/Response-Bit gesetzt ist.

2.) Eine Station soll in den Betriebszustand „DAMA-Slave“ wechseln, wenn sie ein an sich adressiertes Frame mit gesetztem DAMA-Bit empfängt. Als DAMA-Slave soll sie einen globalen Timer führen, der mit einem geeigneten Wert (der vorgeschlagene Mindestwert beträgt 4 Minuten) zu initialisieren ist. Der Timer soll mit jedem an sie gerichteten Frame mit gesetztem DAMA-Bit neu auf den Startwert initialisiert werden. Die Station soll den Betriebszustand „DAMA-Slave” verlassen, wenn entweder der globale DAMA-Timeout ausläuft, oder sie alle Verbindungen auf dem Kanal des DAMA-Masters aufgelöst hat.

3.) Der Slave darf für ein QSO mit dem oder über den Master niemals ein Frame ausstrahlen, bei dem zugleich das Poll/Final- und das Command/Response-Bit gesetzt sind. Das gilt auch für so genannte „Unproto-Polls”.

4.) Es darf keine vom Slave motivierten Aktionen geben. Jede Aktion des Slave darf nur als Reaktion auf den Master-Poll und die in ihm übermittelten Informationen erfolgen. Eine Reaktion auf den T3 ist zulässig, in diesem Zusammenhang auch die Aussendung eines Polls, jedoch sollte der T3 grundsätzlich auf eine Zeit nicht kleiner als 3 Minuten initialisiert sein. Auch sollte dieser Parameter nicht für den User konfigurierbar sein.

Eine Ausnahme bildet das Auf- oder Abbauen eines neuen Links, hier findet natürlich eine vom Slave motivierte Aktion statt. Der T1 darf in diesem Fall als Kriterium für die Erzeugung eines SABM oder DISC benutzt werden, die Aussendung muss innerhalb der Reaktion auf einen Master-Poll erfolgen.

5.) Die Reaktion des Slave auf einen Master-Poll hat unverzüglich und ohne Beachtung der DCD zu erfolgen.

6.) Die Reaktion des Slave auf einen Master-Poll muss aus einem einzigen, zusammenhängenden Sendedurchgang bestehen. Ein „Flackern” des Senders darf unter keinen Umständen auftreten.

7.) Bei Multi-Connect-Verbindungen, sendet der Slave bei einem DAMA-Poll auf einem beliebigen Link alle anstehenden, ungesendeten Frames aller bestehenden Links aus. Schon mal gesendete Frames eines bestimmten Links werden dagegen nur dann wiederholt, wenn dieser Link explizit angepollt wird.

ANHANG I:

TheNetNode 1.79 unter Linux

Zunächst ein wichtiger Hinweis: Die Umstellung eines Digipeaters von der DOS- oder GO32-Version zur Linux-Version von TNN ist nicht in 5 Minuten erledigt! Man sollte sich vorher eingehend mit dem Betriebssystem Linux beschäftigen.

Hier sollen nur die Teile von TheNetNode beschrieben werden, die gegenüber der DOS-Version geändert sind. Die allgemeine Bedienung ist dem vorderen Teil der TNN- Beschreibung zu entnehmen.

Inhalt:

1. Was ist neu seit Version 1.78?
2. Welche Möglichkeiten bietet die Software?
3. Was geht nicht?
4. Anforderungen an die Hardware
5. Installation von Linux
6. Serielle Schnittstellen unter Linux
7. Zugriffsrechte
8. Verzeichnisse und Dateien für TNN
9. Installation von TNN
10. Die Datei tnn.ini
11. Die Datei tnn179.pas
12. Die Datei tnn179.tnb
13. Befehle der Linux-TNN
14. Automatischer Betrieb von TNN
15. Externe Programme
16. TNN und DPBOX auf einem Rechner
16.1. Verbindung zum Hostmodus
16.2. Verbindung über Pseudoterminals
17. TNC-EPROMs
18. Kritik, Probleme, Fehlermeldungen
19. Literatur

1. Was ist neu seit Version 1.78?

- OUTPUT Befehl nun auch für Linux-Version verfügbar (externes Programm)
- In der Statistik wird die verbrauchte CPU-Zeit angezeigt
- Zusätzliche Socket-Optionen bei AX25IP
- Zugriffsrechte der von TNN generierten Dateien können festgelegt werden
- Die Zahl der Rounds/s kann vorgegeben werden
- TNN läuft jetzt auch auf MIPS-Systemen wie dem MeshCube oder dem WRT54G(S)
- Neues Kernelinterface. Befehl ist KERN
- Neues L1-Interface: 6PACK
- Interaktive Linux-Shell

2. Welche Möglichkeiten bietet die Software?

- Keine Speicherprobleme - die Zahl der zu verwendenden Puffer wird in einer INI-Datei festgelegt.
- Es werden bis zu 16 serielle Schnittstellen bedient (braucht jemand mehr?) - also je Port kann eine eigene Schnittstelle verwendet werden.
- Die seriellen Schnittstellen können die hohen Interrupts verwenden, sowie sich Interrupts teilen.

- Eine der seriellen Schnittstellen kann mit dem Tokenring-Protokoll arbeiten - für DAMA muss der Tokenring oder eine Vanessa-Karte verwendet werden.
- Alle Schnittstellen können mit KISS-, RMNC-KISS oder SMACK-Protokoll arbeiten.
- Ein vom Sysop (mit dem SHELL-Befehl) gestartetes externes Programm läuft im Hintergrund und kann abgebrochen werden.
- Eine Mailbox wie z.B. DPBox kann auf demselben! Rechner laufen.

3. Was geht nicht?

- Ein Watchdog, der den Rechner resettet oder die Stromversorgung des Rechners einfach ausschaltet, sollte AUF KEINEN FALL installiert werden! (Watchdogs für die TNCs sind unproblematisch.)

4. Anforderungen an die Hardware

Die Anforderungen an die Hardware werden hauptsächlich vom Betriebssystem Linux bestimmt:

- Prozessor ab 386SX
- Hauptspeicher min. 8MB - besser mehr
- Festplatte > ca. 80MB (abhängig von installierter System-Software bzw. Linux-Distribution)

5. Installation von Linux

Hier soll keine Anleitung zur Installation von Linux gegeben werden. Dies ist schon deshalb (fast) unmöglich, weil bei den verschiedenen Distributionen Unterschiede in der Installation sowohl im Umfang, als auch in der Vorgehensweise bestehen. Es ist auf jeden Fall die mit der Linux- Distribution gelieferte Dokumentation zu Rate zu ziehen. Wer aber nicht nur die ersten beiden seriellen Schnittstellen (/dev/ttyS0 und /dev/ttyS1- entsprechend COM1 und COM2 unter DOS) verwenden will, benötigt auf jeden Fall auch das Programm „setserial“. Die AX.25-Optionen des Kernels werden für TNN genauso wenig benötigt wie die AX.25-Utilities (TNN kann das AX.25 Protokoll alleine verarbeiten). Weiterhin werden zum Übersetzen des Programms der C-Compiler GCC sowie das Programm MAKE benötigt. Das Makefile wird sowohl für die GO32 als auch für die Linux-Version verwendet.

6. Serielle Schnittstellen unter Linux

Bei TNN unter Linux werden die seriellen Schnittstellen vom Betriebssystem bedient, d.h. TNN greift nicht selbst auf die Hardware zu. Die Einstellungen der Hardware (I/O-Adressen, Interrupts) müssen daher dem Betriebssystem mitgeteilt werden, nicht aber TNN. Dafür gibt es das Programm „setserial“. Z.B. wird mit dem Befehl

```
setserial -a /dev/ttyS2 port 0x3E8 irq 5 autoconfig
```

die 3. serielle Schnittstelle auf die Port-Adresse 3E8 und den Interrupt 5 eingestellt. Mit dem Parameter „autoconfig“ wird vom Kernel der UART-Typ bestimmt. Das Programm meldet bei o.a. Aufruf sämtliche zu dem angegebenen Port passenden Parameter. Insbesondere sollte man beachten, welcher UART- Typ gemeldet wird - der Kernel erkennt die folgenden UART-Typen: none (kein UART gefunden), 8250, 16450, 16550, und 16550A. Nur wenn ein 16550A gefunden wird, wird auch der FIFO des UART aktiviert. Statt einen Interrupt anzugeben, kann man das auch dem Programm setserial überlassen. Das könnte dann für das oben angeführte Beispiel so aussehen:

```
setserial -a /dev/ttyS2 port 0x3E8 auto_irq autoconfig
```

Mehrere serielle Schnittstellen können sich bei Linux einen Interrupt teilen. Allerdings muss die Hardware das unterstützen. Notfalls werden die Interrupts mehrerer Schnittstellen über eine ODER-Verknüpfung verbunden (Info dazu auf Anfrage erhältlich). Allerdings wird von Linux für die Interrupts der 3. und 4. Schnittstellen angenommen, dass dieselben Interrupts verwendet werden, wie für die ersten beiden (IRQ 4 und 3). Hier könnten alternativ die IRQs 5 und 7 eingestellt werden, die sonst für die Parallelports üblich sind. Dann sollte man aber auch sicherstellen, dass nicht noch ein Parallelport auf diese IRQs eingestellt ist. Wer eine passende I/O-Karte (z.B. 16-Bit ISA) hat, kann auch mit den hohen Interrupts arbeiten.

Grundsätzlich ist die Verwendbarkeit von I/O-Adressen und Interrupts VOR der Installation zu prüfen (insbesondere auch VGA-Karten und Printer-Ports beachten). Die Seriellen Schnittstellen haben bei Linux die Bezeichnung „/dev/ttyS*“, wobei das Zeichen „*“ durch die Nummer der Schnittstelle zu ersetzen ist. Die Numerierung beginnt mit „0“, nicht mit „1“, wie bei DOS. Die folgende Tabelle soll die Zuordnung verdeutlichen:

Nummer der Schnittstelle	Adresse hexadezimal	Bezeichnung bei DOS	Bezeichnung bei Linux
1	3F8	COM1	/dev/ttyS0
2	3E8	COM2	/dev/ttyS1
3	2F8	COM3	/dev/ttyS2
4	2E8	COM4	/dev/ttyS3

Wenn eine serielle Schnittstelle nicht so funktioniert, wie erwartet, kann es diverse Ursachen haben. Insbesondere bei späterer Installation von zusätzlichen I/O-Karten in einen Rechner muss man sehr sorgfältig prüfen, ob die neu installierten seriellen Schnittstellen auch ohne Konflikt mit den anderen bereits vorhandenen Komponenten des Rechners zusammenspielen. Hier sollen einige mögliche Fehlerursachen aufgeführt werden. Wer weitere Fehlerursachen herausfindet, sollte diese bitte weitergeben - ich nehme die dann gerne in eine spätere Ausgabe dieser Beschreibung auf.

- Die verschiedenen Hersteller von I/O-Karten haben leider keine einheitliche Norm für die Kabel zwischen den Pfostensteckern und SUB-D-Steckern. Wenn man nun ein Kabel aus der Bastelkiste nimmt, wird man voraussichtlich eines mit falscher Pin-Belegung erwischen (Murphy).
- Auf dem Mainboard des Rechners installierte serielle und parallele Schnittstellen können mit den Adressen und IRQs der neu installierten I/O-Karten kollidieren. Wenn das Programm setserial für die gewünschte Schnittstelle bei dem Befehl „setserial -g /dev/ttyS*“ entweder „UART: unknown“ bzw. „IRQ: 0“ meldet, liegt entweder ein Adressen- bzw. IRQ-Konflikt vor, oder aber die Hardware ist defekt. Zur Fehlereinkreisung sollte man ALLE Schnittstellen erst einmal ausschalten und dann jede einzeln einschalten und mit setserial die Erkennung prüfen.
- Wenn mit dem Programm setserial ein IRQ vorgegeben wird, wird dieser nicht geprüft. Wenn man dann 2 IRQs verwechselt, kann es vorkommen, dass zunächst scheinbar alles richtig funktioniert. Erst nach einiger Zeit hängt dann TNN. Es ist dann nicht einmal gesagt, dass TNN abstürzt, möglicherweise funktioniert dann einfach nur keine der Schnittstellen mehr.
- Beim Booten des Rechners werden mehrere Scripte gestartet. Welche das sind und in welcher Reihenfolge sie gestartet werden, ist abhängig von der jeweiligen Distribution. Es muss unbedingt darauf geachtet werden, dass TNN erst gestartet wird, wenn ein eventuell notwendiges Script zum Initialisieren der seriellen Schnittstellen beendet ist.

7. Zugriffsrechte

Üblicherweise wird TNN bei einem Netzknoten auf einem eigenen Rechner installiert. Dann kann man das Programm einfach unter dem Benutzernamen „root“ starten. Damit hat man den gesamten Rechner auch per Fernsteuerung im Zugriff (über den Sysop-Befehl SHELL). Wird die Vanessa-Karte verwendet, muss wegen des direkten Hardwarezugriffs als Benutzer „root“ gestartet werden. Dies ist allerdings eine Sicherheitslücke, wenn externe Programme verwendet werden sollen, oder wenn außer TNN auch andere Programme auf demselben Rechner laufen sollen (Mailbox, TCP/IP, etc.). Soll TNN nicht unter dem Benutzer „root“ gestartet werden, so muss man dafür sorgen, dass der Benutzer, unter dessen Namen TNN gestartet wird, auch wirklich Schreib- und Leserechte für die zu verwendenden seriellen Schnittstellen hat. Das wird z.B. mit dem Befehl „chmod“ oder „chown“ erreicht. Alternativ kann aber auch der zu verwendende Benutzer in die Gruppen eingetragen werden, die Eigentümer der zu verwendenden Schnittstellen sind. Ausserdem muss man dafür sorgen, dass andere Benutzer des Rechners keinen Zugriff auf die Datei tnn179.pas erhalten, weil darin das Knotenpasswort steht. Durch einen Eintrag in der Datei TNN.INI (s.u.) kann ein Zugriffsmodus für Dateien festgelegt werden, die TNN anlegt. Im Folgenden wird allerdings davon ausgegangen, dass TNN vom Benutzer „root“ verwendet wird, und dass keine weiteren Benutzer auf dem Rechner existieren.

8. Verzeichnisse und Dateien für TNN

Zunächst sollte man die von TNN zu verwendenden Verzeichnisse festlegen. Hier wird davon ausgegangen, dass die folgenden Verzeichnisse verwendet werden, auch wenn diese nicht dem aktuellen Filesystem-Standard entsprechen:

```
/usr/local/tnn                - Arbeitsverzeichnis (auch Hilfstexte)
/usr/local/tnn/textcmd        - Texte als User-Befehl
/usr/local/tnn/msg            - Digimail-Texte (MSG-Befehl)
/usr/local/tnn/userexe        - Programme als User-Befehl
/usr/local/tnn/sysexe         - Programme als Sysop-Befehl
/usr/local/tnn/pacsat         - Dateien für PACSAT-Broadcast
```

Die Namen der von TNN benötigten Dateien werden klein geschrieben. Für den Betrieb von TNN werden die folgenden Dateien benötigt:

- tnn - Das Programm selbst
- tnn179.pas - Diese Datei wird von TNN generiert, wenn nicht schon vorhanden. Hier wird das Passwort für den SYSOP-Befehl definiert, sowie das Call des Digipeaters, aber auch die verschiedenen Pfade. Die Datei wird in dem Verzeichnis gesucht, in dem TNN gestartet wird.
- tnn.ini - Definition der Schnittstellenparameter etc.
- tnn179.tnb - Starteinstellung der Ports, Links, Parameter etc.

9. Installation von TNN

Zur Installation von TNN wird zunächst der Quelltext ausgepackt. Dazu wird im Verzeichnis "/" der folgende Befehl eingegeben:

```
bunzip2 -c tnn179.tar.bz2 | tar -xv
```

Damit wird das Verzeichnis /usr/local/src/tnn/tnn179 angelegt, in dem die Sourcen dann zu finden sind - dorthin wird daher auch gewechselt. Anschließend wird das Programm übersetzt mit dem Befehl „make tnn“. Mit dem Befehl „make install“ kann das Programm in das Verzeichnis „/usr/local/tnn“ kopiert werden. Dorthin werden dann auch die anderen benötigten Dateien (Ini-Dateien, Hilfetexte) kopiert, man wenn eingibt

```
make baseinstall
```

Wenn ein anderes Arbeitsverzeichnis verwendet werden soll, muss man das Makefile anpassen oder die entsprechenden Dateien von Hand kopieren. Das Programm „tnn“ steht im Unterverzeichnis „bin“ des Quellverzeichnisbaums, die Startdateien und Hilfstexte im Verzeichnis "os/linux/ini". Anschliessend werden die Dateien „tnn.ini“, „tnn179.pas“ und „tnn179.tnb“ mit einem Texteditor angepasst. Die Quelltexte sind für 16 Kanäle eingestellt. Ebenso sind die Beispieldateien für „tnn.ini“, „tnn179.pas“ und „tnn179.tnb“ für 16 Kanäle eingerichtet - alle 16 Kanäle am Tokenring an der seriellen Schnittstelle /dev/ttyS0 (entsprechend COM1 unter DOS) mit 38400 Baud.

Die Quelltexte sind nicht nur für die Linux-Version, sondern ebenso für die GO32-Version (für MSDOS) verwendbar. Mit Hilfe des Makefiles kann aber nur die Linux-Version installiert werden. Die MSDOS-Version ist nach dem Übersetzen im bin-Verzeichnis als „tnngo32.exe“ zu finden.

10. Die Datei "tnn.ini"

Die Datei „tnn.ini“ wird in dem Verzeichnis gesucht, in dem „tnn“ gestartet wird. Für die Initialisierung der Hardware wird die Datei „tnn.ini“ benötigt. Änderungen an dieser Datei sind nur notwendig, wenn die Hardware geändert wird, also die Zahl der TNCs oder die Baudraten zu den TNCs, oder auch die Verbindung eines TNC zu einer seriellen Schnittstelle.

WARNUNG! - WARNUNG! - WARNUNG! - WARNUNG! - WARNUNG! - WARNUNG! - WARNUNG!

Die Angaben in dieser Datei werden (in gewissen Grenzen) auf Plausibilität geprüft. Bei fehlerhafter Datei wird das Programm mit einer Fehlermeldung beendet. Da zu viele verschiedene Konfigurationsmöglichkeiten bestehen, ist eine Vorgabe für einen „Notfallmodus“ leider nicht möglich. Eine Änderung dieser Datei per Fernsteuerung ist daher sehr gefährlich - der Digipeater kann möglicherweise dadurch lahm gelegt werden.

In der Datei „tnn.ini“ wird die zu verwendende Hardware (serielle Schnittstellen, Vanessa- Karten, etc.) festgelegt. Hier wird auch die Zuordnung zu den logischen Ports und die Art des KISS-Modus definiert.

Die Datei besteht aus mehreren Blöcken - je ein Block pro serieller Schnittstelle, sowie ein Block mit allgemeinen Angaben. Die Reihenfolge der Angaben innerhalb eines Blocks MUSS eingehalten werden!

In dem ersten Block befinden sich allgemeine Daten. Zunächst wird angegeben, wie viele Buffer „tnn“ verwenden soll. Dazu wird nach dem Kennwort „buffers“ die gewünschte Anzahl angegeben. Wird diese Angabe weggelassen, werden als Voreinstellung 10000 Buffer verwendet. Als nächstes wird der Name für die Datei angegeben, in der die Prozessnummer des Programms hinterlegt wird. Diese Zeile darf entfallen - dann wird als Dateiname die Vorgabe „tnn.pid“ verwendet. Diese Datei wird z.B. für das Programm „boxstart“ von DL4YBG benötigt, mit dem eine automatische Überwachung von TNN möglich ist. Der Name der Datei folgt auf das Kennwort „tnn_procfile“. Die Datei wird im TNN-Arbeitsverzeichnis abgelegt. In einer zusätzlichen Zeile können die Zugriffsrechte der von TNN generierten Dateien festgelegt werden. Auf das Kennwort „perms“ folgt eine Zahl für die Berechtigungsbits. Die Zahl ist eine Oktalzahl. Die Berechtigungsbits werden angegeben, wie für den Befehl umask üblich. Mit z.B. „perms 077“ werden die Dateien nur noch lesbar und schreibbar für den User, der TNN gestartet hat.

Parameter “rounds” in tnn.ini:

Mit diesem Parameter kann man die Anzahl der Hauptschleifendurchläufe pro Sekunde einstellen, per Default, bzw. wenn dieser Parameter fehlt, werden wie bisher 100 Runden pro Sekunde absolviert. Durch eine Erhöhung dieses Parameters kann eine Steigerung des Datendurchsatzes erreicht werden, jedoch geschieht dies zu Lasten des Rechenzeitverbrauches. In den meisten Fällen ist die Standarddrehzahl ausreichend.

Die tatsächlich absolvierte Drehzahl hängt zusätzlich noch von der verwendeten Hardware (Tokenring, Vanessakarte etc.) ab, sie kann mal höher mal niedriger ausfallen. Maximal können 65500 Runden eingestellt werden, Werte unterhalb von 100 sind nicht möglich. Es ist nicht empfehlenswert, Werte größer als 5000 einzustellen da der Rechenzeitverbrauch erheblich ansteigt. Ist der verwendete Rechner zu langsam wird unter Umständen die eingestellte Drehzahl nicht erreicht.

Beispiel: rounds 1000

Nun folgen die Blöcke für die Beschreibung der seriellen Schnittstellen.

In jedem dieser Blöcke wird zunächst die Bezeichnung der seriellen Schnittstelle angegeben hinter dem Kennwort „device“. Für die Vanessa- Karte wird hier der Name „vanessa“ angegeben. Ähnliche Devices wie z.B. Pseudoterminals und SCC-Karten können ebenfalls angegeben werden. In der nächsten Zeile folgt nach dem Kennwort „tnn_lockfile“ der Dateiname für das zur Schnittstelle gehörende Lock-File mit dem kompletten Pfad. Diese Zeile ist optional - entfällt diese Zeile, wird kein Lock-File verwendet. Mit den Lock-Files soll verhindert werden, dass mehrere Programme auf dieselbe Schnittstelle gleichzeitig zugreifen. Für die Lock-Files ist das Verzeichnis „/var/lock“ üblich. Man kann beim Aufruf von „tnn“ mit dem Parameter "-u" festlegen, dass die Lock-Files ignoriert (überschrieben) werden. Nach dem Programmende (aber nicht nach einem eventuellen Programmabsturz) werden die Lock-Files gelöscht.

Als nächstes folgt auf das Kennwort „speed“ die Baudrate. Mögliche Baudraten sind 9600, 19200, 38400, 57600 und 115200. Ob das ganze insbesondere bei 57600 und 115200 Baud funktioniert, hängt natürlich auch vom Prozessortakt ab und dürfte ohne FIFO-Baustein nicht gehen. Für eine Vanessa-Karte ist hier 0 anzugeben (die HF-Baudrate der Vanessa wird mit dem Port-Befehl in tnn179.tnb definiert). Soll die Verbindung zu einem anderen Programm über andere Devices erfolgen (pty, tty, scc), wird hier ebenfalls 0 eingesetzt. Statt „speed 0“ anzugeben, darf die Zeile auch entfallen.

Schließlich folgt auf das Kennwort „kisstype“ die Art des zu verwendenden KISS-Modus. Möglich sind 0 = KISS (ohne CRC), 1 = SMACK, 2 = RMNC-KISS, 3 = Tokenring, 4 = Vanessa, 5 = SCC, 6 = TF, 7 = IPX, 8 = AX.25IP, 10 = Kernel-AX.25, 11 = DG1KJD Kernel-AX25, 12 = 6PACK. Allerdings kann nur EIN Tokenring verwendet werden, und dieser MUSS im ersten Block deklariert werden. Für eine Vanessa-Karte ist als kisstype 4 anzugeben. Ein DAMA-Port kann nur mit Tokenring oder Vanessa verwendet werden. Der KISS-Modus 5 (SCC) entspricht KISS-Modus 0 mit dem Unterschied, dass keine Port-Parameter zur Schnittstelle übertragen werden - dies ist Aufgabe eines getrennten Programms, das die SCC-Karte(n) vor dem Start von TNN initialisiert. Für die Verwendung von AX.25IP und IPX kann jeweils nur 1 Port deklariert werden. Weitere Informationen zu diesen beiden Interfaces sind in der Datei „ipxaxip.doc“ enthalten.

Zum Abschluss des Blocks wird nach dem Kennwort „port“ der an dieser Schnittstelle zu verwendende Port festgelegt. Nur für den Tokenring sind mehrere Port-Zeilen zulässig / notwendig. Wenn beide Ports einer Vanessa- Karte verwendet werden sollen, müssen 2 Definitionsblöcke mit je einer Port-Zeile verwendet werden. ACHTUNG! Die Port-Nummern werden nicht auf mehrfache Verwendung überprüft - es dürfen aber nur Port-Nummern angegeben werden, die von der Software-Version unterstützt werden.

In der Datei „tnn.ini“ darf überall eine Kommentarzeile (oder mehrere) eingefügt werden. Diese muss mit einem „#“ beginnen.

Soll die Datei "tnn.ini" einen anderen Namen erhalten oder nicht aus dem aktuellen Verzeichnis gelesen werden, so kann beim Aufruf von „tnn“ der Pfad zur Datei angegeben werden mit dem Aufrufparameter „-i <Filename>“.

In der Datei „os/linux/ini/tnnini.all“ ist für alle verfügbaren Schnittstellenkarten ein Beispiel angegeben.

11. Die Datei „tnn179.pas“

In der Datei „tnn179.pas“ sind das Rufzeichen des Netzknotens, das Passwort für den SYSOP-Befehl sowie die Pfade für externe Befehle abgespeichert. Der Aufbau der Datei, d.h. die Reihenfolge der Infozeilen, darf keinesfalls geändert werden! Lediglich die Kommentarzeilen (beginnend mit „;“) dürfen sich ändern. Wenn die Datei fehlt, wird sie von TNN neu angelegt. Es werden dann die oben angegebenen Standart-Pfade verwendet.

12. Die Datei „tnn179.tnb“

In dieser Datei werden die Port- und Linkeinstellungen beim Start vorgenommen. Die Datei ist wie alle anderen tnb-Dateien auch aufgebaut. Wichtig ist hier vor allem, dass die verwendeten Ports eingeschaltet werden. Per Voreinstellung sind nämlich alle Ports ausgeschaltet. Auch die Linkeinträge müssen hier vorgenommen werden, sowie Parameter, die gegenüber der Vorgabe abweichen.

13. Befehle der Linux-TNN

Bei der Linux-Version von TNN werden im Allgemeinen dieselben Befehle verwendet, wie bei der DOS-Version. Es ist aber folgendes zu beachten:

GANZ WICHTIG: Zum Verlassen gibt es nur den Befehl <ESC>QUIT

mit ALT-X geht das nicht! Außerdem muss man im Knoten eingeloggt sein, wenn man den Befehl <ESC>QUIT eingeben will.

Es gibt den Befehl <ESC T> zum Einstellen der Baudrate beim Tokenring NICHT! Die Tokenring-Baudrate wird nur in der Datei „tnn.ini“ festgelegt.

Da die Hardware in der Datei „tnn.ini“ deklariert wird, gibt es beim PORT-Befehl die Parameter „ON“ und „OFF“, nicht aber „TOKENRING“, „KISSLINK“ oder „VANESSA“. Allerdings werden die Parameter „TOKENRING“, „VANESSA“ und „KISS“ auch statt des Parameters „ON“ akzeptiert, damit die Datei „parms.tnb“ nach dem Umbenennen zu „tnn179.tnb“ ohne zusätzliche Bearbeitung verwendet werden kann. Dies geht natürlich nur, wenn keine zusätzlichen Befehle beim Start benötigt werden. Auf freien Ports, die also nicht in der Datei tnn.ini deklariert sind, kann auch die Einstellung „LOOP“ verwendet werden.

Bei externen Befehlen für Textausgabe (aktuell, info, etc.) wird der Text im Linux-Format benötigt, also als Zeilentrennung nur <LF> und kein <CR>. Andernfalls werden zusätzliche Leerzeilen ausgegeben. Mit dem EDIT-Befehl erhält man das gewünschte Dateiformat.

Bei LOAD, READ und auch bei READBIN können Dateinamen mit absolutem Pfad angegeben werden. Es ist aber zu beachten, dass der Dateiname IMMER in Kleinschrift umgewandelt wird. Weiterhin ist die Länge der Eingabezeile auf 128 Zeichen begrenzt - superlange Pfade können also nicht verwendet werden.

Der SHELL-Befehl entspricht dem Befehl DOS (den gibt es auch noch und er hat die gleiche Funktion). Man beachte, dass die übergebene Befehlszeile von der Shell (bash) ausgewertet wird. Daher sind Dateinamen wie z.B. „########.tnb“ in Anführungszeichen zu setzen. Das aktuelle Verzeichnis beim SHELL-Befehl ist das TNN-Arbeitsverzeichnis. Die Ausführung des SHELL-Befehls erfolgt in einem Hintergrundprozess, so dass der Knoten nicht während der Ausführung steht, wie bei der DOS-Version. Daher können problemlos auch längere Programme gestartet werden (z.B. neue TNN-Version übersetzen). Allerdings wird beim SHELL-Befehl ein Timer gestartet, der die Ausführungszeit auf 1 Minute begrenzt. Danach wird das Programm abgebrochen. Falls versehentlich ein falsches aber sehr lang laufendes Programm gestartet wurde, kann durch eine Eingabe die Befehlsausführung abgebrochen werden; die Eingabe wird nicht mehr weiter ausgewertet. Eine interaktive SHELL gibt es bei TNN nicht. Innerhalb von tnb-Files kann der SHELL-Befehl allerdings nicht verwendet werden, weil er automatisch von dem folgenden Befehl beendet wird. Statt des SHELL-Befehls bleibt nur die Verwendung eines Script-Programms im sysexe-Verzeichnis, dass das gewünschte Programm mit passenden Parametern aufruft.

Bei Eingabe des Befehls RESET SYSTEM wird TNN beendet. Der bei früheren Versionen verwendete Rechnerneustart ist bei Linux nicht gerade sinnvoll. Wird tatsächlich ein Rechnerneustart gewünscht, so muss man den TNN-Befehl „shell /sbin/shutdown -r now“ verwenden. Von der Verwendung eines Hardware- Watchdog, der den Reset- Knopf des Rechners betätigt oder die Stromversorgung unterbricht, ist DRINGEND abzuraten! Ebenso problematisch ist der Kernel- Watchdog von Linux Version 2.0.0 (und evtl. neueren Versionen?), da auch hier das Betriebssystem nicht ordnungsgemäß heruntergefahren wird. Daher ist die Verwendung des Kernel-Watchdog auch nicht in TNN implementiert

KERN INIT: Initialisiert das Interface, es wird geprüft ob der Kernel die benötigten Funktionen bereitstellt. Unterstützt werden die Kernel 2.0.x, 2.2.x und 2.4.x. Bei 2.0.x und 2.2.x muss die benötigte Funktionalität über ein Zusatzpaket hinzugefügt werden, bei 2.4.x ist nur eine Option beim Kernelcompilieren entsprechend zu setzen. Kernel 2.0.x/2.2.x: es sind zusätzliche Programmpakete erforderlich. Kernel 2.4.x: bei „Network device support“ der Unterpunkt „Universal TUN/TAP device driver support“, möglichst fest in den Kernel bauen, als Modul bisher nicht getestet. Das Interface initialisiert sich NICHT von selbst, dies muss explizit mit INIT erfolgen!!! Es ist sonst nicht nutzbar und zeigt sich ziemlich bockig. Es erfolgt ein Warnhinweis, falls zu diesem Zeitpunkt mit IPA noch keine Node-IP festgelegt wurde. Die Interfacestatistik wird gelöscht.

KERN STATUS: Zeigt den momentanen Status und die Statistik des Interfaces an.

KERN CLEAR: Löscht die Statistik, dies passiert auch bei „CLEAR“ für die Gesamtstatistik.

KERN SETKIP: Setzt die IP des Linuxkernels. Hier kann entweder eine IP-Adresse oder ein Hostname angegeben werden, letzterer wird entsprechend aufgelöst *wenn* ein Nameserver verfügbar ist. Die Angabe einer IP-Nummer ist deshalb zu bevorzugen!

Beispiel: „KERN SETKIP 44.130.13.110“ oder „KERN SETKIP db0uhi.ampr.org“ Dies muss passieren, bevor das Interface geUPt wird, ist dies nicht geschehen kann das Interface nicht benutzt werden!!! Die IP von TNN wird wie üblich mit dem IPA-Kommando gesetzt, dies muss ebenfalls vor dem UPpen passiert sein.

KERN UP: Aktiviert das konfigurierte Interface. Es wird im Kernel ein Interface mit dem Namen „tnn“ erzeugt (mit ifconfig prüfbar) und eine Route für tnn eingetragen (route). Dies passiert alles automatisch auf der Basis der gesetzten IPs. Im tnn-IP-Router wird ebenfalls ein passender Routeneintrag erzeugt. Das Interface lässt sich nur UPpen wenn folgendes vorher passiert ist: IPA, KERN INIT, KERN SETKIP <ip/hostname>.

KERN DOWN: Deaktiviert das Interface zum Kernel. Das eingetragene Interface und die automatisch im Kernel und tnn eingetragenen IP-Routen werden gelöscht, sonstige Routen im tnn-Router die auf das Interface zeigen, bleiben eingetragen, sie funktionieren aber natürlich nicht mehr. Daten, die bei deDOWNtem Interface durch noch bestehende Routeneintraege auf ein solches geroutet werden, werden in den Biteimer befördert.

Allgemeines: etliche Kommandos sind nur nach vorheriger, erfolgreicher Ausführung anderer Kommandos möglich oder hängen vom derzeitigen Zustand des Interfaces ab. Wenn man etwas machen will was derzeit nicht geht, erhält man (mehr oder weniger) ausführliche Meckermeldungen. Die Konfiguration eines Interfaces kann nur geändert werden wenn es DOWN ist. Ausnahme: IPA-Änderungen, aber die werden bei aktivem Interface (noch) nicht an den Kernel durchgereicht. Das Resultat ist denkbar einfach: das Interface geht dann nicht mehr.

Ablaufbeispiel:

- IPA 44.130.13.100
- KERN INIT
- KERN SETKIP 44.130.13.102 bzw. KERN SETKIP db0uhi.ampr.org
- (KERN STATUS zum gucken)
- KERN UP
- (eventuell zusätzliche Routeneinträge mit IPR)
- KERN DOWN

STAT zeigt auch die Statistik des Kernelinterfaces mit an, diese kann auch direkt mit „STAT K“ abgefragt werden wenn eincompiliert.

= Versionsbefehl zeigt Feature und Kernel als Interface an wenn eincompiliert

14. Automatischer Betrieb von TNN

Um ein automatisches Starten von TNN nach dem Booten oder Absturz zu ermöglichen, gibt es 3 Möglichkeiten. Man kann einen Eintrag in der Datei „/etc/inittab“ vornehmen, der die entsprechende Aufgabe übernimmt. Dies könnte z.B. so aussehen:

```
8:2:respawn:/tnn/tnn -i /tnn/tnn.ini -u </dev/tty8 >/dev/tty8
```

Damit ist TNN auch direkt über die Tastenkombination Alt-F8 zu erreichen. Allerdings muss die obige Beispielzeile an die jeweilige Distribution angepasst werden (unterschiedliche Verwendung der Runlevel).Alternativ hat sich auch das Programm „boxstart“ von DL4YBG bewährt, das zusammen mit der Mailbox DPBOX verteilt wird. Sollen mehrere Programme auf demselben Rechner laufen (z.B. TNN + DPBOX), so ist dieses Programm vorzuziehen (siehe weiter unten sowie die Doku zu „boxstart“).

Schließlich kann man über ein Script in einer Endlosschleife TNN nach dem Start des Rechners bzw. nach einem Absturz neu starten. Dieses Script könnte folgendermaßen aussehen:

```
#!/bin/sh
#
 while [ " " ]   # Endlosschleife
  do
  #
  echo "Gleich wird tnn neu gestartet. Abbruch mit CTRL-C."
  #
  # so viel Zeit werden wir wohl noch übrig haben:
  sleep 5
  # TNN starten im Verzeichnis /usr/local/tnn
  (cd /usr/local/tnn ; ./tnn -u)
  #
 done
```

WICHTIG! - WICHTIG! - WICHTIG! - WICHTIG! - WICHTIG! - WICHTIG! - WICHTIG!

Unabhängig von der Vorgehensweise ist bei einem automatischen Betrieb von TNN immer folgendes zu achten: Wenn nicht nur die seriellen Schnittstellen /dev/ttyS0 und /dev/ttyS1 (entsprechend COM1: und COM2: unter DOS) mit den Standart-Adressen und -Interrupts verwendet werden sollen, muss die Einstellung der Schnittstellen mit dem Programm „setserial“ VOR dem ersten Start von TNN erfolgen.

15. Externe Programme

Die in der TNN-Doku beschriebenen externen Programme zur DOS-Version sind für die Linux-Version von TNN nicht alle verfügbar. Die Programme MSG / MSY und OUTPUT werden mit dem Quelltext verteilt, einige weitere sind verfügbar. Wie bei der DOS-Version auch, können im Verzeichnis „userexe“ Programme abgelegt werden, die vom TNN-User aufgerufen werden können. Es können hier auch Script-Programme verwendet werden, wenn sie ausführbar sind. Dem Programm wird als Parameter der Rest der User-Zeile beim Aufruf übergeben sowie als letzten Parameter zusätzlich das Rufzeichen des Users (in Kleinschrift!). WICHTIG: Es dürfen keine interaktiven Programme verwendet werden - sonst steht der Digi!

16. TNN und DPBOX auf einem Rechner

Da offenbar sehr großes Interesse besteht, TNN und DPBOX auf einem Rechner zu betreiben, folgt hier eine etwas ausführlichere Anleitung, wie man die beiden Programme verbinden kann. Für den Betrieb der Mailbox benötigt man außer dem Programm DPBOX von DL8HBS zusätzlich das Programm TNT von DL4YBG und evtl. auch noch das Programm TFKISS von DL4YBG, da in TNN kein direktes Interface zu DPBOX existiert. Zum automatischen Betrieb wird weiterhin das mit DPBOX / TNT verteilte Programm BOXSTART von DL4YBG verwendet.

Grundsätzlich gibt es dabei 3 Möglichkeiten. Zunächst die Verbindung über 2 serielle Schnittstellen, wobei eine an TNN und die andere über TFKISS an TNT angebunden ist. Wie man das installieren muss, soll hier nicht erläutert werden, aber das dürfte auch ohne zusätzliche Anleitung klar sein. Weiterhin gibt es den Hostmodus, wie er bei einem TNC verwendet wird. Dieser Hostmodus wird über eine Socket-Schnittstelle angesprochen, wie sie bei TNT eingebaut ist. Diese Art der Verbindung zwischen TNN und TNT ist auf jeden Fall vorzuziehen, wenn man mit 30 Hostmode-Kanälen auskommt. Die 3. Methode verwendet statt der seriellen Schnittstellen sogenannte Pseudoterminals. Dieses Verfahren ist relativ kompliziert, soll aber trotzdem dargestellt werden, weil es auch für andere Programme (z.B. Wampes) verwendet werden kann, die über den KISS-Modus verfügen. Beide hier beschriebenen Methoden übertragen die Daten zwischen TNN und TNT bzw. TFKISS über den Speicher des Betriebssystems und nicht über eine Drahtleitung.

16.1. Verbindung zum Hostmodus

Das Hostmode-Interface von TNN ist abgestimmt auf den Hostmodus von TNT über eine Unix-Socket-Schnittstelle. An TNT wiederum ist dann DPBOX angeschlossen. Um das Hostmode-Interface zu aktivieren, wird am Dateianfang von „tnn.ini“ eine Zeile eingefügt, in der ein Dateiname für den Socket angegeben wird - z.B.:

```
tnn_socket /usr/local/tnn/tnnsocket
```

Wichtig! Diese Zeile ist keine Zuweisung für ein Device, wie für die Linkports. Für den Host-Mode wird kein Port zugewiesen.

Da bei Verwendung der Socket-Schnittstelle eine Bedienung von TNN über die Konsole nicht mehr möglich ist, wird TNN als Hintergrundprozess gestartet, wenn in „tnn.ini“ ein Dateiname für einen Socket angegeben ist. In der Datei „tnn179.tnb“ wird das Call gesetzt, mit dem der Host-Mode beim Connect-Befehl und mit dem Mailbox-Befehl erreichbar sein soll, und außerdem der Alias für den Eintrag in die Link-Liste:

```
ESC I DB0XYZ-7 TESTMB

M DB0XYZ-7
```

TNN stellt damit einen Local- Link- Eintrag her auf Port 16. Dieser Eintrag und auch das Call für die Konsole / den Host-Mode können nur geändert bzw. gelöscht werden, wenn ein neues Call MIT Alias mit dem ESC-I-Befehl eingegeben wird. Soll der Eintrag gelöscht werden, muss das Knoten-Call eingegebenen werden (ein Alias ist hier auch notwendig!).

Für die Verbindung zu TNT wird in der Datei „tnt.ini“ folgendes eingetragen:

```
soft_tnc 1
device /usr/local/tnn/tnnsocket
tnc_channels 30
```

Die erwähnten Programme sollen nun automatisch gestartet werden. Dafür wird zusätzlich das Programm BOXSTART von DL4YBG verwendet. Die Reihenfolge zum Starten der verschiedenen Programme wird mit einem Shell- Script festgelegt, das als /usr/local/boxstart/startall gespeichert wird:

```
#!/bin/sh
#
# Script zum Starten von TNN, TNT und DPBOX.
#
# zuerst tnn starten
cd /usr/local/tnn
/usr/local/boxstart/boxstart -i /usr/local/tnn/start_tnn
#
# als nächstes dpbox starten
cd /usr/local/box
/usr/local/boxstart/boxstart -i /usr/local/box/start_box
#
# fehlt nur noch tnt
cd /usr/local/tnt
/usr/local/boxstart/boxstart -i /usr/local/tnt/start_tnt
```

In jedem der Verzeichnisse gibt es eine passende Datei, aus der BOXSTART die nötigen Informationen über das zu startende Programm findet (siehe auch die Dokumentation zu BOXSTART). Als Beispiel folgt hier die Datei /usr/local/tnn/start_tnn:

```
/usr/local/tnn/tnn.pid
/usr/local/tnn/tnn -u
```

Für die anderen Programme sieht die Startdatei entsprechend aus. Damit haben wir zunächst die Möglichkeit, alle verwendeten Programme mit einem einzigen Befehl zu starten. Aber für automatischen Betrieb muss dieses Script regelmäßig aufgerufen werden, falls eines der Programme einmal abgestürzt ist. Daher wird zusätzlich mit dem Befehl „crontab -e“ ein Eintrag im Terminkalender erstellt:

```
* * * * * /usr/local/boxstart/startall
```

Damit wird jede Minute das Start-Script aufgerufen und die einzelnen Programme neu gestartet, falls sie nicht mehr laufen.

16.2. Verbindung über Pseudo-Terminals

Die Verbindung zu anderen Programmen kann von TNN über Pseudo-Terminals hergestellt werden, wobei die Verbindung zwischen den Programmen über das Betriebssystem vorgenommen wird. Eine Pseudo- Terminal- Schnittstelle hat den großen Vorteil, dass sie sehr ähnlich wie eine serielle Schnittstelle programmiert wird, dadurch ist kein zusätzlicher Programmaufwand bei TNN nötig. Außerdem spart man 2 serielle Schnittstellen und die dazu gehörenden Interrupts ein. Hier soll die Anbindung der Mailbox DPBOX über diese Schnittstelle dargestellt werden.

Die Pseudo-Terminal-Schnittstelle besteht aus je 2 über das Betriebssystem verbundenen Teilen, dem Master /dev/ptyXX und dem slave /dev/ttyXX, wobei das „XX“ in der Bezeichnung jeweils durch eine von (bei der Linux-Kernel Version 2.0) 256 möglichen Zeichenkombinationen zu ersetzen ist. Die Zeichenkombination besteht aus einem der Buchstaben [p-z] oder [a-e] an erster Stelle und einer Ziffer [0-9] oder einem Buchstaben [a-f] an zweiter Stelle (Kleinschrift bei den Buchstaben beachten!). Für die Verbindung von TNN und TFKISS wird eine dieser Schnittstellen ausgesucht, im Folgenden wird von /dev/ptyz0 und /dev/ttyz0 ausgegangen. Weiterhin wird von den folgenden Pfaden zu den einzelnen Programmen ausgegangen:

tnn	in /usr/local/tnn
tfkiss	in /usr/local/tfkiss
tnt	in /usr/local/tnt
dpbox	in /usr/local/dpbox
boxstart	in /usr/local/boxstart

Wichtig bei der Verwendung der pty/tty-Schnittstelle ist die Reihenfolge, in der die Programme die Schnittstellen initialisieren. Zuerst MUSS immer die Master-Seite geöffnet werden, also hier /dev/ptyz0. Erst danach kann / darf die Slave-Seite geöffnet werden. Dies ist zu gewährleisten, indem die Programme in der richtigen Reihenfolge gestartet werden. Problematisch wird es nur, wenn das Programm an der Master-Seite beendet wird (egal ob durch den Benutzer gewollt, oder durch Programmabsturz). Das Programm an der Slave-Seite MUSS dann auf jeden Fall ebenfalls beendet werden. Wie Versuche gezeigt haben, ist es auch möglich, nach einem Absturz zuerst die Master-Seite neu zu öffnen und erst danach die Slave- Seite zu schließen und wieder zu öffnen. Man kann also auch im File tnn179.tnb als ersten Befehl einen entsprechenden kill-Befehl aufnehmen, um TFKISS zu beenden - z.B.:

```
shell killall -9 tfkiss
```

Für TNN verwenden wir die Master-Seite der Schnittstelle, also im File tnn.ini „device /dev/ptyz0“. Ein Lock-File wird nicht benötigt, weil es von anderen Programmen sicherlich nicht beachtet würde. Eine Angabe der Baudrate ist für TNN nicht notwendig, hier kann „speed 0“ verwendet werden; dann versucht TNN auch nicht, die Baudrate zu programmieren, oder andere Einstellungen wie Stopbits oder Parität.

Bei TFKISS wird die Slave-Seite verwendet, also in tfkiss.ini steht an passender Stelle „device /dev/ttyz0“. Bei TFKISS muss eine Baudrate von 9600 oder 19200 angegeben werden, bei 38400 oder mehr funktioniert es nicht. Die Verbindung zu TNT und DPBOX wird entsprechend der zugehörenden Dokumentation vorgenommen. Um die Programme TNN, TFKISS, TNT und DPBOX automatisch zu starten wird (wie bei der Hostmode-Verbindung beschrieben) ein Start-Script verwendet, diesmal etwas erweitert.

```
#!/bin/sh
#
# Script zum Starten von TNN, TNT und DPBOX.
#
# zuerst tnn starten
cd /usr/local/tnn
/usr/local/boxstart/boxstart -i /usr/local/tnn/start_tnn
#
# als nächstes dpbox starten
cd /usr/local/box
/usr/local/boxstart/boxstart -i /usr/local/box/start_box
#
# erstmal etwas warten, damit tnn die pty/tty-Schnittstelle geöffnet hat
sleep 15
#
# nun darf tfkiss gestartet werden
cd /usr/local/tfkiss
/usr/local/boxstart/boxstart -i /usr/local/tfkiss/start_tfkiss
#
# fehlt nur noch tnt
cd /usr/local/tnt
/usr/local/boxstart/boxstart -i /usr/local/tnt/start_tnt
```

Weitere Informationen sind bei der Beschreibung der Hostmode-Verbindung zu finden.

17. TNC-EPROMs

An die seriellen Schnittstellen für TNN können TNCs mit verschiedenen EPROMs angeschlossen werden. Für den Tokenring wird die Tokenring- Kiss- Software verwendet, die auch bei der GO32/DOS-Version verwendet wird. Für den TNC2 gibt es auch ein EPROM nur für SMACK/KISS. Beide Softwareversionen sind schon seit Jahren erfolgreich in Betrieb u.a. bei DB0IL.

Die Unterstützung für ein EPROM mit TheFirmware 2.7b wurde für Versuchszwecke eingebaut, damit man nicht regelmäßig das EPROM seines TNC wechseln muss. Dafür ist eine Umschaltung vom Terminal- in den KISS-Modus eingebaut. Wenn TNN beendet wird, wird der TNC auch wieder in den Terminal-Modus zurückgeschaltet. Hierbei gibt es aber folgende Probleme:

- Der TNC muss im Terminal- oder KISS-Modus laufen. Im Terminal-Modus dürfen noch keine Zeichen im Eingabepuffer des Programms vorhanden sein. Dies ist gewährleistet, wenn man den TNC neu einschaltet, oder wenn man ein Hostmode-Terminal-Programm beendet.
- Es gibt keine Möglichkeit, einen Port-Reset auszuführen.
- Wenn das TNC-Programm abstürzt oder die Stromversorgung unterbrochen wird, kann TNN das nicht feststellen, und es wird evtl. Schrott als UI-Frames statt verwertbarer Daten gesendet. Empfangen kann man dann natürlich auch nichts mehr. Da hilft dann nur, TNN zu beenden, den TNC zu resetten und TNN neu zu starten.
- Bei einem Programmabsturz von TNN bleibt der TNC im KISS-Modus. Wenn danach wieder TNN gestartet werden soll, ist das kein Problem, aber nicht alle Hostmode-Terminalprogramme können den TNC aus dem KISS-Modus zurückschalten. Dann muss man den TNC resetten. An einem Digipeater sollte also kein TF2.7b EPROM verwendet werden. Für Versuche zu Hause ist der Betrieb aber gut geeignet.

18. Kritik, Probleme, Fehlermeldungen

Wenn mit der Linux-Version von TheNetNode Probleme auftreten - egal, ob Programmabstürze oder „nur“ Schwierigkeiten bei der Installation, sollte man sich im DL-Convers auf Kanal 170 melden. Dies gilt natürlich auch für Änderungswünsche oder Kritik. Selbstverständlich würden wir uns auch über Rückmeldungen freuen, wenn irgendeine in dieser Beschreibung nicht erwähnte Hardware angepasst werden konnte.

Die Software wurde getestet mit Linux Kernel Versionen 2.0 bis 2.6 sowie den Compiler-Versionen gcc 2.7.2 bis 3.3.4 und läuft damit auch unter anderem bei DB0IL, DB0CEL, DB0MAR und DB0HHW jeweils auf einem Rechner zusammen mit DPBOX. Bei vielen weiteren Digipeatern läuft TNN zusammen auf einem Linux-Rechner mit der Open-BCM ebenfalls problemlos.

19. Literatur

- The Linux Serial HOWTO by Greg Hankins, greg.hankins@cc.gatech.edu (als Textdatei bei einigen Linux-Distributionen enthalten).
- Die Linux-Befehle sind in Deutsch beschrieben im „Linux Anwenderhandbuch“ von Sebastian Hetze, Dirk Hohndel, Martin Müller und Olaf Kirch. Dieses Buch ist auch bei einigen Linux-Distributionen als Text-Datei enthalten.

73, Nils (DF6LN @ DB0IL).

Die Ethernet-Interfaces von TNN-LINUX

Das IPX-Interface

Das IPX-Protokoll wird hauptsächlich in Novell-Netzwerken verwendet. Um dieses Protokoll mit TNN zu verwenden, muss der Kernel mit IPX-Option compiliert werden. Ausserdem sollten die IPX-Konfiguration-Utilities installiert werden. Des weiteren muss erfolgreich eine Netzwerkkarte installiert worden sein. Ist der Kernel ohne IPX-Unterstützung compiliert, gibt es beim Starten von TNN mit IPX-Interface eine Fehlermeldung „cannot create socket“. Vor dem Start von TNN muss dann ein IPX-Interface aktiviert werden, dies wird mit dem Befehl:

ipx_interface add eth0 -p EtherII

in der shell gemacht. Eth0 ist die Netzwerkkarte, -p definiert das Interface als Primäres IPX-Interface und EtherII ist das Frameformat. Das Frameformat ist kompatibel mit dem des PC/FlexNet-Treibers IPXPD und dem von TFX_IPX. Ist kein IPX-Interface konfiguriert, beendet sich TNN mit der Fehlermeldung „cannot bind address“.

Ein Beispiel für die TNN.INI

```
# *****************************************************
# device 5
# *****************************************************
device ipx
# lockfile for device 5
#tnn_lockfile /usr/spool/uucp/LCK..cua4
# speed on device 5
#speed 38400
# type of KISS on device 5: 0 = KISS, 1 = SMACK, 2 = RMNC-
KISS
kisstype 7
# L2-Port associated with device 5
port 5
```

Dieser Eintrag definiert das IPX-Device auf Port 5. In der TNB kann der Port jetzt mit PO 5 ON eingeschaltet werden. Ansonsten ist keine weitere Konfiguration notwendig.

Folgende Einschränkung ist zu beachten: IPX wird als Broadcast ins Netzwerk gesendet. Irgendwelche Router/Switches im Netzwerk werden die IPX-Frames mit grosser Sicherheit NICHT durchlassen!

Das AXIP-Interface

Vorwort

Das AXIP-Format stützt sich auf wichtige Grundregeln des IP-Routings. Aus diesem Grund sind Grundkenntnisse für den Einsatz des AXIP-Interfaces notwendig. Ausserdem sollte man in der Lage sein, eine Netzwerkkarte unter Linux zu installieren und zu konfigurieren. Eine Einführung in diese komplexe Materie würde diese Anleitung sprengen.

Konfiguration

Das AXIP-Format wurde in der RFC 1226 (IP Encapsulation of AX.25 Frames, May 1991) beschrieben. Eingeschaltet wird das Interface in der TNN.INI mit (Beispiel):

```
# *****************************************************
# device 6
# *****************************************************
device ax25ip
# lockfile for device 6
#tnn_lockfile /usr/spool/uucp/LCK..cua5
# speed on device 6
#speed 38400
# type of KISS on device 6: 0 = KISS, 1 = SMACK, 2 = RMNC-KISS
kisstype 8
# L2-Port associated with device 6
port 6
```

Bei AXIP (=AX25IP) werden die Frames nicht als Broadcast ins Netzwerk gesendet, sondern gerichtet an den entsprechenden Empfänger. Damit dies funktioniert muss im Kernel ein entsprechender Routen-Eintrag zum Zielrechner vorhanden sein.

Nehmen wir mal an, wir möchten eine Verbindung zwischen DB0KOE und DB0GSO aufbauen. Der Rechner von DB0GSO hat die IP-Adresse 192.100.100.1. Mit 'route add 192.100.100.1 eth0' sage ich dem Kernel wie die IP-Adresse von DB0GSO zu routen ist.

Ausserdem muss man die Datei ax25ip.cfg entsprechend editieren:

```
# Sample thenetnode/ax25ip configuration file
#
# First select active socket types
# syntax: socket <ip/udp> [udp-port]
#
socket ip
#socket udp 10093
#
#
# loglevel 0 - no output
# loglevel 1 - config info only
# loglevel 2 - major events and errors
# loglevel 3 - major events, errors, and AX25 frame trace
# loglevel 4 - all events
#
loglevel 2
#
# Define some routes.  One example routes all traffic for callsign db0zzz
# to a host named nexthost.bla.blub.  You can define as many as
# required.
# syntax: route <callsign> <hostname> [udp] [udp-port]
#
#
route db0gso 192.100.100.1
#
#
# A catch-all is provided: this line sends all calls not specifically
# noted in the routing tables to otherhost.bla.blub.  Use this feature
# with great care -- the host on the other end may not appreciate all the
# traffic!
# syntax: route default <hostname> [udp] [udp-port]
#
#route default 192.100.100.50
#
```

Mit dem Eintrag 'route db0gso 192.100.100.1' teilt man TNN mit, dass das Call DB0GSO-0 (SSID beachten!) unter der IP-Adresse 192.100.100.1 zu erreichen ist. Das bedeutet, dass der AXIP-Treiber von TNN bei jedem zu sendenden Frame die Empfänger-IP-Adresse ermitteln muss! Der große Vorteil bei diesem Verfahren ist, dass die Pakete durch ein komplettes Netz gerouted werden können, da es sich um keinen Broadcast handelt. Es ist damit also möglich zwei TNNs über ein größeres IP-Netzwerk zu verbinden.

Die Tabelle, in der IP-Adresse und Call gespeichert werden, ist z.Zt. auf 128 Einträge beschränkt. Sollten mehr gebraucht werden, ist die Variable TABLE_SIZE in der Datei AX25IP.C zu ändern. Ein automatisches Eintragen von Zielen ist im Moment nicht vorgesehen. Steht also ein Call nicht in der Tabelle, so ist KEIN Connect möglich!

Für die Fehlersuche können vier verschiedene Log-Level aktiviert werden, allerdings ist zur Zeit nur der Level 2 teilweise eingebaut! Sehr hilfreich ist auch das Linux-Tool TCPDUMP. Sollte der Connect zu einem Rechner nicht möglich sein, so sollte man in einer shell zuerst mit dem Ping-Befehl die Erreichbarkeit des Rechners überprüfen!!

Bei Fragen und Problemen stehe ich gerne zur Verfügung.

vy 73/55 de DG1KWA / Andreas

Beschreibung der Option „socketoption“ in ax25ip.cfg

Mit Hilfe der Option „socketoption“ können bestimmte Socketoptionen des Kernels und des IP-Protokollstacks gesetzt werden. Da nicht alle Kernel alle Optionen verarbeiten, kann es leicht vorkommen, dass beim Start von TNN eine Fehlermeldung angezeigt wird. Die entsprechende Option konnte dann nicht gesetzt werden und der Programmstart erfolgt dementsprechend ohne die Option. Es ist sehr wichtig, dass jede Option in einer eigenen Zeile der Konfigurationsdatei steht.

Hier ein Bespiel:

```
# Now set some socket options
# available options are : IPTOS_THROUGHPUT SO_KEEPALIVE
# important ! use multiple socketoption-lines, one for each option
#
socketoption IPTOS_THROUGHPUT
socketoption SO_KEEPALIVE
```

Bisher sind folgende Optionen realisiert:

IPTOS_THROUGHPUT:	stellt die Verbindung auf den maximal möglichen Durchsatz ein
SO_KEEPALIVE:	aktiviert auf Socket-Ebene einen Linkcheck

Eine genaue Übersicht, welche Optionen zur Verfügung stehen, findet sich in der jeweiligen Beispiel-ax25ip.cfg der TNN-Version.

ANHANG J:

TCP/IP und TNN

Eine Einführung zur Einstellung der Routen in TheNetNode

Es hat sich gezeigt, dass viele Sysop von TNN Dingis bisher nichts oder nur wenig mit TCP/IP zu tun hatten. Aus diesem Grund versuche ich hier eine kleine Einführung mit Beispielen zu schreiben.

Es sollen keine Grundlagen des TCP/IP Protokolls vermittelt werden, sondern es soll nur dazu dienen einen TNN Digi TCP/IP mäßig richtig zu konfigurieren. Die TCP/IP Grundlagen sind an anderer Stelle nachzulesen.

IP Adresse:

Zuerst muss der Digi eine IP Adresse bekommen.

IP Adressen werden weltweit international vergeben, d.h. eine IP Adresse ist nur EINMAL auf der Welt vorhanden bzw. zugeteilt. Man kann sich also nicht einfach eine Adresse aussuchen, sondern muss sich an den entsprechenden IP Adressverwalter wenden.

Weltweit macht das Brian Kantor (Rufzeichen gerade nicht zur Hand). Weiterhin gibt es in jedem Land einen weiteren IP Adressverwalter. In DL ist das im Moment DH9KAE, der wiederum weitere Adressverwalter hat, welche die einzelnen Regionen verwalten.

Für den Hamburger Raum macht das z.B. Eddi dk3uz. Übrigens hat Brian Kantor das gesamte 44er Netz und muss dafür bezahlen! Ich weiß nicht wie viel, aber solche Netze sind gebührenpflichtig; das nur kurz zur Info.

Es gibt eine Ausnahme:

44.128.0.0 - 44.128.255.255 sind Adressen die jeder mal zum testen verwenden darf, denn diese Adressen werden offiziell nicht zugeteilt.

Wie sieht so eine IP Adresse aus?

Beispiel von db0hht.ampr.org:

```
44.130.0.100
 |  |  |  +---- die 100 ist die Adresse von db0hht.ampr.org
 |  |  +------- die   0 steht für das Netz 0
 |  +---------- die 130 steht für DL
 +------------- die  44 steht für die Domain ampr.org, also Amateurfunk
```

Noch ein weiteres Beispiel von db0nds:

```
44.130.150.100
 |  |   |   +-- die 100 ist die Adresse von db0nds.ampr.org
 |  |   +------ die 150 steht für das Netz 150
 |  +---------- die 130 für DL
 +------------- die  44 steht für die Domain ampr.org, also Amateurfunk
```

In diesem Beispiel unterscheiden sich die Adressen an einem Punkt, d.h. einmal steht da die 0 für Netz 0 und einmal die 150 für Netz 150!

Die IP Adresse wird bei TNN jetzt mit dem Kommando IPA <Adresse> eingestellt.

Einstellen der tcp/ip routen:

Wie erreichen wir jetzt, dass IP Pakete von z.B. db0nds aus in das Netz 0 richtig geroutet werden?

Für das Netz 0 ist db0hht.ampr.org das Gateway, d.h. alle Pakete sollten über db0hht.ampr.org geschickt werden und zwar die Adressen 44.130.0.0 bis 44.130.0.255, also die 255 Adressen für das entsprechende Netz. DB0HHT.ampr.org reicht diese Adressen dann richtig auf dem Einstieg weiter. Wir müssen also ein sog. Clusterrouting einstellen und das können wir z.B. bei db0nds so machen:

Zuerst wird die Route zu db0hht.ampr.org eingestellt (44.130.0.100)

IPR 44.130.0.100 + NET/ROM

Dann werden alle Adressen des Netzes 0 über db0hht.ampr.org geroutet:

IPR 44.130.0.0/24 + NET/ROM 44.130.0.100

Jetzt können alle Rechner im Netz 0 von db0nds aus über db0hht.ampr.org erreicht werden. Wenn da nicht noch etwas wäre.

Woher weiß db0nds denn wie db0hht mit AX.25 erreicht werden kann?

Dafür ist der ARP Eintrag vorgesehen.

ARP 44.130.0.100 + NET/ROM db0hht

Damit wird der Software mitgeteilt, dass TCP/IP Pakete, welche die Adresse 44.130.0.100 haben, an die AX.25 Adresse db0hht per NetRom zu schicken sind!

Wir müssen hier nämlich genau zwischen „Hostname" und „AX.25-Adresse" unterscheiden! Das ist ein ganz wichtiger Punkt, der gerne Verwirrung stiftet:

Es kann z.B. folgendes sein:

Hostname : dk3hg.ampr.org

AX.25-Call: dk3hg

Username : dk3hg

Hier taucht immer dieser „dk3hg" auf, aber alle drei Namen, die zwar gleich lauten, haben unterschiedliche Bedeutung!!

Noch ein Wort zu den „routen":

Es macht wenig Sinn einzelne Hosts zu routen, ohne das Clusterrouting anzuwenden, denn dann erreicht man nur diesen EINEN Hostnamen und weiter nichts. Das ist aber nicht der Sinn eines Routers, wie ihn TNN darstellt.

Außerdem sollten die Sysop sich untereinander absprechen „wer wie was" routet, damit es nicht zum Durcheinander kommt. Ein guter Anfang ist eigentlich immer sich an die Sysop der eigentlichen TCP/IP Digi zu wenden, wenn es um solche Fragen geht.

Einen ganz besonderen Vorteil möchte ich noch herausstellen. Bisher war es vor allem für den Norden (Netromland) sehr problematisch TCP/IP in den Süden (FlexNet-Land) zu machen, weil die Digi im FlexNet-Land immer ihren Broadcast über FlexNet nach Norden schicken mussten. Das kann man jetzt umgehen, wenn die TNN, die an FlexNet angeschlossen sind, ihre routen richtig eingestellt haben. Dann ist es möglich von z.B. db0hht aus auch alle möglichen anderen TCP/IP Digi zu erreichen, die NUR FlexNet machen.

Wer wissen möchte welcher Subnetzkoordinator für seine Region zuständig ist, kann in den Boxen nach „Subnet_Koordinator_Liste" suchen. Dort sind alle Koordinatoren für DL aufgeführt. Ich schicke diese Liste auch gerne auf Anforderung per Mail zu, sollte sie nicht mehr in den Boxen zu finden sein.

Wer in seiner Region keinen Subnetzkoordinator hat, muss sich dann an DD9QP, DL3SBB oder DL9SAU wenden, um ein neues Subnetz zu beantragen. Gleichzeitig sollte ein OM in dieser neuen Region die Verwaltung dieses Netzes übernehmen. Es empfiehlt sich dann eine IP Adresse für den TNN Router zu nehmen, die mit einer 1 anfängt, wie z.B. 44.130.neues_Netz.1. Das hat dann den Vorteil dass man dieses Netz bei Bedarf noch weiter aufteilen kann.

Eine aktuelle Liste der IP Adressen ist bei db0hht oder bei anderen UNIX Digi die WAMPES verwenden, im Verzeichnis /tcp zu finden. Je nach Sysop mehr oder weniger aktuell :-). Wer die gesamte weltweite Liste haben möchte und einen Internetanschluss hat, kann sich bei ftp://ftp.ucsd.edu/hamradio mal umschauen.

Im Anhang 1 ist die Aufteilung eines Netzes in weitere Subnetze zu finden. Im Anhang 2 ist ein unvollständiges Beispiel wie wir es bei db0nds eingestellt haben.

Anhang 1

Man kann auch ein Netz von z.B. 44.130.0.0 bis 44.130.0.255 noch weiter in einzelne Subnetze aufteilen. Das sieht dann z.B. so aus:

```
Netmask /Bits    Adressen  Hosts
================================================================================
44.130.0.0/24       256     254
44.130.0.128/25     128     126
44.130.0.192/26     64      62
44.130.0.224/27     32      30
44.130.0.240/28     16      14
44.130.0.248/29     8       6
44.130.0.252/30     4       2
--------------------------------------------------------------------------------
```

Anhang 2

```
ipa 44.130.150.100

ipr 44.130.0.100   + netrom
arp 44.130.0.100   + db0hht
ipr 44.130.0.0/24  + netrom 44.130.0.100

ipr 44.130.82.101  + Ruhner-B.
arp 44.130.82.101  + Ruhner-B. VC db0oca-11 db0lwl-2
ipr 44.130.83.0/24 + Ruhner-B. 44.130.82.101

ipr 44.130.20.0/24 + Ruhner-B. 44.130.82.101
ipr 44.130.42.0/24 + Ruhner-B. 44.130.82.101
ipr 44.130.56.0/24 + Ruhner-B. 44.130.82.101
ipr 44.130.60.0/24 + Ruhner-B. 44.130.82.101
ipr 44.130.80.0/24 + Ruhner-B. 44.130.82.101
ipr 44.130.88.0/24 + Ruhner-B. 44.130.82.101
ipr 44.130.91.0/24 + Ruhner-B. 44.130.82.101

--------------------------------------------------------------------------------
```

IP-Router

Diese Anleitung richtet sich an Sysop und User, die etwas über IP, IP-Routing und IP-Nutzung mit TheNetNode erfahren möchten. Alle Beschreibungen sind auf den Kern reduziert und genügen sicher nicht erfahrenen Gurus.

1. WAS IST IP?

IP ist ein bestimmtes Frameformat, das Datenaustausch über ein Medium ermöglicht. So ein Medium kann z.B. ETHERNET (Netzwerkkarte), ARCNET (dito) oder aber auch PACKET RADIO sein. Dies ist ein wichtiger Aspekt. Sowohl AX.25 als auch NET/ROM sind für IP nur Transportmedien. Deshalb ist TCP/IP auch nicht zum Open Systems Interconnection (OSI) Normvorschlag konform.

DIENSTE TELNET, FTP, HTTP

TRANSPORT TCP, UDP

ROUTER IP

MEDIUM ETHERNET, ARCNET, SLIP, PACKET-RADIO

TheNetNode kennt nur ROUTER und MEDIUM (und noch ROUTER-Dienste, aber das ist hier erstmal egal).

2. IP-ROUTER

Ein IP-Router analysiert die Zieladresse von IP-Frames und versucht anhand dieser Adresse, Frames zuzustellen. Hierzu benutzt TNN die sogenannte IP-Router-Tabelle, die mit dem Befehl IPR konfiguriert wird.

IPR Adresse/Maske [+/-] [Flags] [Interface] [Gateway]

Adresse/Maske definiert eine Adresse (Maske=32, also alle Bits gelten) oder einen Adressbereich, für den dieser IPR-Eintrag gilt.

Für Flags kann D wir „dynamisch" angegeben werden. Diese Adressen haben eine besondere Bedeutung, die unten näher erklärt wird.

Als Interface kann ein Portname oder „NETROM" angegeben werden. Frames für diese Adresse(n) werden IMMER an dieses Interface gesendet.

Gateway wird im Zusammenhang mit ARP erklärt. Es ist die IP-Adresse des nächsten Routers, wenn diese Adresse nicht direkt erreicht werden kann oder soll.

3. ARP

ARP bedeutet ADDRESS RESOLUTION PROTOCOL. Wenn ein Frame anhand der IP-Routen-Tabelle einem Interface(=Port) zugeordnet wurde, wissen wir immer noch nicht die AX.25-Addresse (=Rufzeichen) der Station, die dieses Frame bekommen soll, wir brauchen also eine Zuordnung IP-Adresse und Rufzeichen.

ARP DestIP [+/-] [Publ.] [Port] [DG/VC] [Call] [Digi1[Digi2]]

DestIP ist die Adresse (hier wird nur EINE Adresse angegeben, eine AX.25-Addresse hat auch nur eine IP Nummer). Ist DestIP ein Gateway, dann wird die Adresse hier angegeben und in den IP-Routen immer auf diese Adresse als Gateway verwiesen. Dadurch können Adressbereiche einzelnen Gateways zum routen zugeordnet werden.

Für Public kann ein „P" angegeben werden, dies bedeutet, dass dieser Eintrag öffentlich ist, und Anfragen von anderen Stationen über diesen Eintrag beantwortet werden.

DG/VC bestimmt, ob die Verbindung Connected oder mit UI-Frames passieren soll.

Call/Digi gibt den Connectweg an.

Für NET/ROM wird für Port wieder „NETROM" angegeben.

4. Automatisches ARP

Das Eintragen der ARP-Daten kann automatisch erfolgen, wenn das Ziel direkt vom Knoten gehört wird, also alle lokalen User usw. Es geht nicht für via-Verbindungen und NET/ROM.

Wenn TNN ein Frame auf einem Port zustellen will, und keinen passenden ARP-Eintrag hat, wird eine UI-Anfrage generiert. Wird diese beantwortet, dann werden künftige Frames an diese Adresse geschickt. Der Eintrag gilt für 60 Minuten.

5. Dynamische Adressen

Ein User kann mit „IPR +" für die aktuelle Verbindung eine IP-Adresse beantragen, wenn für diesen Port dynamische Adressen eingetragen sind. Die Adresse ist für eine Stunde vergeben (ab der letzten Nutzung).

Ein Skript zum IP-Betrieb könnte also so aussehen:

IPR +

[IP-Adresse wird als Text ausgegeben, diese lesen]

ARP +

[Verbindung ist umgeschaltet]

[LOKAL auf IP schalten und loslegen]

Hier noch ein paar Begriffserklärungen verfasst von DL8XAS:

Internetprotokolle

>From ampr.bbs.tcpip Tue Dec 12 17:12:25 1995
>From: dl8xas@db0hb.ampr.org
Date: Mon, 11 Dec 95 19:34:13 GMT
Newsgroups: ampr.bbs.tcpip
Subject: Internetprotokolle
Message-ID: 11C50FDB0HB@bbs.net
Path: dk3hg!db0hht!db0hb!dl8xas

TCP/IP
ist eine Abkürzung, bzw. die beiden wichtigsten Protokolle der sogenannten Internet Protocol Suite. Dies ist im Grossen und Ganzen eine Sammlung von Protokollen, die im ehemaligen ARPA-Net (Advanced Research Project Agency) des Department of Defense (auch kurz „Pentagon“) entwickelt wurde. Ziel der Entwicklung war es, ein funktionierendes, zuverlässiges Netzwerk zu errichten und betreiben. Der Grundstein Ufer diese Entwicklung wurde schon Ende der 60er Jahre gelegt, die heute verwendeten Protokolle sind aber alle neueren Datums.

IP (Internet Protocol)
Dieses Protokoll ist das Herzstück des ganzen Systems. Seine Aufgabe ist das Routen von einzelnen Paketen, im Internet-Sprachgebrauch auch Datagramme genannt. Jedes IP-Datagramm hat unter anderem eine Absenderadresse, eine Empfängeradresse (die sog. IP-Adressen), einen Protokoll-Identifier, div. weitere Verwaltungsinformationen und einen Datenteil. IP arbeitet nur als untergeordnetes Protokoll für höher liegende Protokolle und ist ausschliesslich dafür zuständig, dass ein IP-Datagramm zugestellt wird. Folglich ist es die Aufgabe der höher liegenden Protokolle, zu kontrollieren ob ein Paket auch beim Empfänger angekommen ist, ob es mehrfach angekommen ist, oder ob eine Sequenz von Paketen auch in der richtigen Reihenfolge dort gelandet ist. IP sorgt lediglich für das korrekte Routing durch diverse Rechner und Gateways hin zum Empfänger.

ICMP (Internet Control Message Protocol)
Über dieses Protokoll werden Fehlermeldungen zwischen den IP-Prozessen der einzelnen Rechner ausgetauscht. Beispielsweise wenn ein Host ein Paket nicht korrekt routen kann, oder wenn das Time-To-Live-Feld im IP-Datagramm abgelaufen ist und das Paket so weggeworfen werden muss. Aber auch die Erreichbarkeit eines Rechners lässt sich per ICMP-ECHO-REQUEST feststellen, der übliche Befehl dafür nennt sich PING.

ARP (Address Resolution Protocol)
Hierfür am besten ein Beispiel: DB0HHS erhält ein IP-Datagramm mit der Empfänger-IP-Adresse 44.130.0.100. Nun schaut der Rechner in seine Routing-Tabelle, und stellt fest, dass er das Paket auf Schnittstelle /dev/sio01 wieder rausschicken soll, das der Empfänger dort direkt erreichbar ist, und aus seiner Konfiguration weiss er auch, dass auf dieser Schnittstelle AX.25-Protokoll gefahren wird. Doch wie heisst der Empfänger? Nun fragt er einfach nach. Er schickt einen ARP-REQUEST an das Rufzeichen QST mit folgendem Inhalt: „Hier ist 44.130.2.102 mit dem AX.25-Rufzeichen DB0HHS-9, wer ist denn hier 44.130.0.100?“ Der Nachbarrechner hört das und antwortet: „Hier ist 44.130.0.100, mein Rufzeichen lautet DB0HHT.“ ARP wandelt also IP-Adressen in AX.25-Rufzeichen um. Ursprünglich wurde es für Ethernet entwickelt, wurde aber so weitsichtig ausgelegt, dass man es bei anderen Netzwerktechniken wie Token-Ring oder AX.25 ebenfalls verwenden kann.

TCP (Transmission Control Protocol)
Da in IP-Netzen es ja durchaus vorkommen kann, das ein Datagramm verloren geht, doppelt zugestellt wird oder die Reihenfolge von Paketen vertauscht wird, ist IP als Protokoll für viele Anwendungen nicht direkt benutzbar. Um eben diese Probleme zu beheben, gibt es TCP. TCP arbeitet mit virtuellen Verbindungen, d.h. es wird eine Verbindung vor dem ersten Datenaustausch erst einmal aufgebaut, und wenn der Datenaustausch beendet ist, wird diese auch wieder korrekt abgebaut. Um die Verbindungen auf die einzelnen Programme aufteilen zu können, arbeitet TCP mit Portnummern. Jedes Programm belegt eine bestimmte Portnummer wenn es eine Verbindung aufbauen möchte, diese liegt in der Regel über 1000. Es gibt ja aber auch Serverprozesse. Diese belegen in der Regel eine Portnummer <1000, und legen sich dann „schlafen“, d.h. sie legen ihre Seite einer Verbindung an, und warten dann auf Verbindungsanforderungen. Diese Portnummern sind genormt, z.B. das Mailprotokoll SMTP benutzt Port 25. Wenn z.B. DB0HHT eine Mail an DB0HHS schicken möchte, baut der Mail-Prozess bei DB0HHT eine TCP-Verbindung zu 44.130.2.100:25 auf.

UDP (User Datagram Protocol)
Nicht für jede Verbindung wird eine Flusskontrolle, wie TCP sie bietet, benötigt. Protokolle, die eine einzelne Anfrage stellen und auch nur eine einzige Antwort benötigen, sind auf den Aufwand, den TCP treibt, nicht unbedingt angewiesen. Wenn eine Anfrage oder eine Antwort verloren geht, so fragt man halt kurz später noch mal nach. Um aber trotzdem die Datagramme den einzelnen Programmen zuordnen zu können, benötigt man halt Portnummern wie bei TCP. Für diese Anwendung wurde UDP geschaffen. Es verpackt ein einzelnes Paket eines Benutzerprogramms in ein IP-Datagramm, versieht es mit den benötigten Portnummern, und schickt es ab. UDP ist genau so verlässlich wie IP - man sollte sich nicht drauf verlassen!

TELNET
Um eine Terminalverbindung mit einem anderen Rechner aufzubauen, benutzt man TELNET. Nachdem der TELNET-Prozess auf der lokalen Maschine sich mit dem TELNET-Server auf der entfernten Maschine verbunden hat, tauschen beide Parameter über die Verbindung aus wie z.B. ob Zeichen lokal oder per remote geechot werden. Danach ist man mit dem anderen Rechner verbunden, so als ob man dort direkt sitzen würde. Um eine einwandfreie Verbindung zu gewährleisten, benutzt TELNET TCP als Transportprotokoll, die Portnummer ist 23.

FTP (File Transfer Protocol)
Mit diesem Protokoll werden Dateien zwischen einzelnen Rechnern übertragen, und ggf. auch gewandelt. Textdateien z.B. zwischen DOS- (CR/LF) und Unix- (LF) Konvention. Für Befehle und Daten werden getrennte TCP-Verbindungen benutzt. Die Portnummern lauten 20 für Daten und 21 für Kommandos.

SMTP (Simple Mail Transfer Protocol)
Über dieses Protokoll wird Post verschickt. Das S für Simple ist eigentlich nicht gerechtfertigt, es gibt kaum eine Implementation, die alle Protokollmöglichkeiten ausnutzt. Aber für den normalen Hausgebrauch reichen auch die Minimalimplementationen wie sie allgemein üblich sind. Der verwendete TCP-Port ist 25.

DNS (Domain Name System)
Um als normaler Benutzer nicht ständig mit den schwer merkbaren IP-Nummern hantieren zu müssen, haben alle Rechner auch einen Namen und eine Domain zu der sie gehören. Z.B. db0hhs.ampr.org ist der Rechner db0hhs in der Domain ampr (AMateur Packet Radio), welche eine Unterdomain von org (Non-Profit-Organisationen) ist. Um nun aus diesem Namen die IP-Nummer 44.130.2.100 zu gewinnen, kann man entweder alle Rechner, die man braucht, in einer Datei speichern (z.B. hosts.net oder /etc/hosts), was aber bei maximal 2^32 Hosts (über 4 Milliarden) etwas umständlich ist, oder man legt eine zentrale Datenbank an, die so etwas verwaltet. Gelöst hat man das nun durch mehrere Datenbanken, die jeweils eine so genannte Zone verwalten. Als Beispiel aus den Drahtnetzen: ein Rechner möchte eine Verbindung aufbauen und braucht die IP-Nummer von rzdspc5.informatik.uni-hamburg.de. Er fragt den nächstgelegenen Nameserver nach der Nummer, der sagt ihm, .de-Adressen kenne er nicht, aber er solle doch mal bei unido.de mit der IP-Nummer a.b.c.d (wie auch immer) nachfragen. Der wiederum erzählt ihm, dass er für UNI-Hamburg nicht zuständig währe, und er möchte sich doch an den dortigen Nameserver wenden. Und der wird ihn dann noch an den Nameserver des Fachbereichs Informatik weiterverweisen. Das alles hat zur Folge, das jedes Institut, Unternehmen, Verein oder sonst was seine IP-Nummern selbst verwaltet, der Rest der Welt fragt schon nach, wenn er es wissen will. Und dafür braucht man schlicht und einfach das DNS-Protokoll. Da es nur kurze Anfragen sind, mit einer Antwort, und die Verbindung danach nicht mehr gebraucht wird, benutzt man hierfür das UDP-Protokoll, Portnummer 53. Wenn eine Anfrage oder eine Antwort verloren geht, dann fragt man halt noch mal nach.

NNTP (Network News Transfer Protocol)
Was SMTP für private Nachrichten, ist NNTP für öffentliche. Der Unterschied zu „normalem“ S&F von DieBox ist, dass NNTP strukturierte Sparten hat, z.B. wird TCPIP aus DieBox in ampr.bbs.tcpip abgebildet, es gibt aber auch völlig andere News-Gruppen, aus den Kabelnetzen z.B. comp.os.unix oder rec.pyrotechnic. NNTP benutzt TCP-Port 119.

POP (Post Office Protocol)
SMTP versucht in regelmässigen Abständen, Post zuzustellen. Der verwendete Mechanismus geht aber davon aus, dass der nächste Rechner auf dem Weg eigentlich ständig zu erreichen ist. Bei privat betriebenen Rechnern ist das aber nicht unbedingt vorauszusetzen. Dafür wurde POP geschaffen, es dreht den Mechanismus um, man fragt also bei einem Rechner nach, ob für einen Post da ist, und holt sie sich dann ggf. ab. Verwendet werden z.Zt. zwei Protokollversionen, POP2 auf TCP-Port 109, und POP3 auf TCP-Port 110.

Weitere, für den Amateurfunk weniger wichtige Protokolle:

ECHO
Schickt alles zurück, was es empfängt (nur für Tests). TCP- und UDP-Port 7.

DISCARD
Wirft alles weg (nur für Testzwecke sinnvoll). TCP- und UDP-Port 9.

DAYTIME
Gibt die Uhrzeit aus (sinnvoll für was?). TCP- und UDP-Port 13.

QUOTD
Gibt den täglichen Spruch aus (Quote of the Day). TCP-Port 17.

CHARGEN
Erzeugt eine künstliche Netzbelastung (Character Generator). Ebenfalls nur für Testzwecke sinnvoll. TCP- und UDP-Port 19.

TIME
Timeserver, wird benutzt um die Uhren von mehreren Rechnern zu synchronisieren. TCP- und UDP-Port 37.

BOOTP
Über dieses Protokoll können Rechner booten. UDP-Port 67.

TFTP
Das Trivial File Transfer Protocol, weitaus einfacher aufgebaut als FTP, wird hauptsächlich zum booten benutzt (wie BOOTP). UDP-Port 69.

FINGER
Liefert Informationen über Benutzer, wie z.B. Name, Bürotelefonnummer oder andere Selbstdarstellungen. TCP-Port 79.

SUNRPC
Protokoll um Unterprogramme auszuführen, die auf anderen Rechnern laufen. Ursprünglich von der Firma Sun entwickelt (SUN Remote Procedure Call). Über dieses Protokoll werden noch weitere Unterprotokolle abgewickelt, das bekannteste davon dürfte NFS (Network File System) sein, mit welchen man sich Festplatten oder anderes netzweit teilt. Auch NIS (Network Information Service), ehemals YP (Yellow Pages) läuft hierüber. Mit NIS kann man Verwaltungsinformationen von Rechnern, wie z.B. die Passwort-Datei, zentral verwalten. Man trägt einen Benutzer z.B. nur einmal zentral ein, und er kann dann alle Rechner eines Netzes nutzen. Und über NFS erhält er auch seine Dateien an jedem Rechner dann. TCP- und UDP-Port 111.

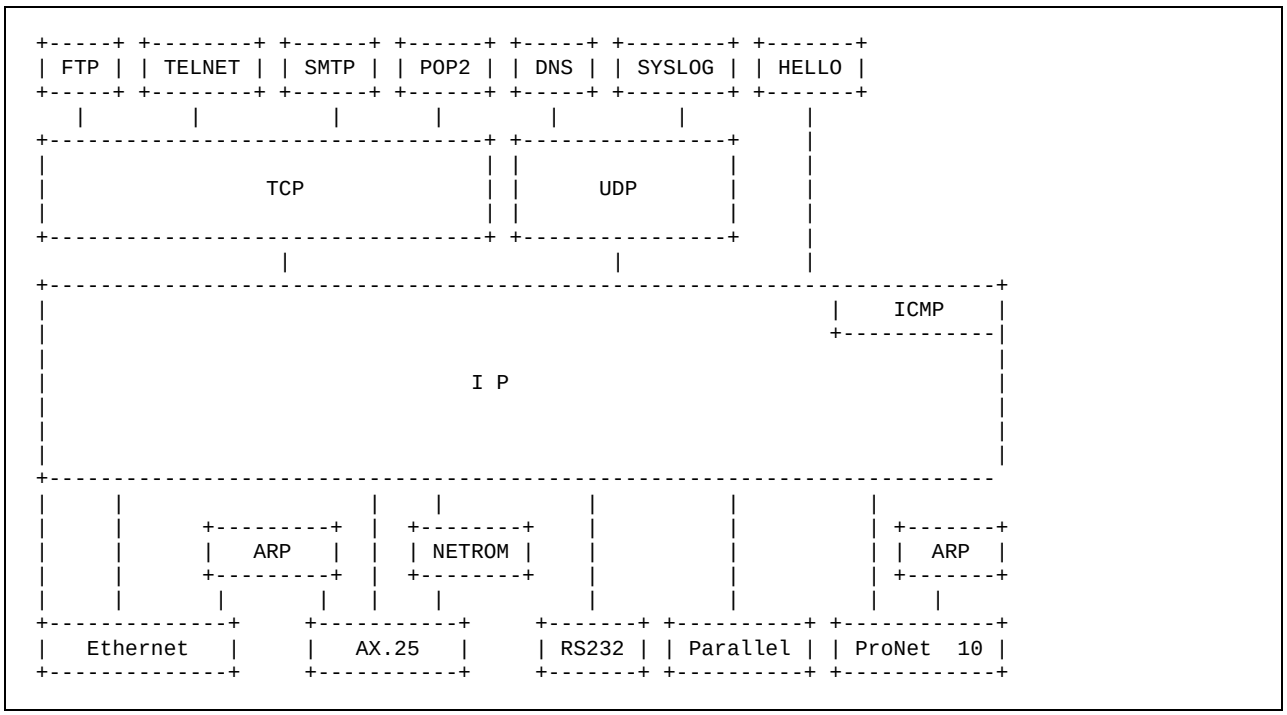
NTP
Das so genannte Network Time Protocol. Im Gegensatz zu TIME hat NTP keine klare Trennung zwischen Master und Slave, hier versuchen sich die Rechner gegenseitig zu einigen, das hat zur Folge, das Schwankungen in den Ganggenauigkeiten der Uhren sich (hoffentlich) gegenseitig ausgleichen. UDP-Port 123.

SNMP
Mit Hilfe des Simple Network Management Protocols kann man andere Rechner im Netz konfigurieren. Man kann z.B. die Routingtabellen ändern oder sich ansehen, wieviele kaputte und wieviele heile Pakete in einem gegebenen Zeitabschnitt man auf einem bestimmten Netzwerkinterface empfangen hat. TCP-Port 161.

Es gibt auch Betriebssystemspezifische Protokolle, wie z.B. PRINTER (TCP-Port 515) mit dem man unter Unix Druckausgaben von einen Rechner zum anderen (oder Printserver) schickt, oder SYSLOG (UDP-Port 514), über das man z.B. Betriebssystemfehlermeldungen (z.B. kaputter Plattensektor) netzweit zentral sammeln kann.

Weiterhin gibt es auch noch Routingprotokolle, die man in Interne oder Externe Routing Protokolle einteilt. Interne Protokolle versuchen in einem Netz den optimalen Pfad von einem Rechner zu einem anderen zu finden. Ein Vertreter dieser Gruppe ist z.B. RIP (Routing Information Protocol) auf UDP-Port 520, oder HELLO. Letzteres benutzt weder TCP noch UDP, doch davon später mehr. Externe Routingprotokolle tauschen nur Routinginformationen zwischen den Netzen unterschiedlicher Verwaltungen aus, da währen z.B. EGP (Exterior Gateway Protocol) oder BGP (Border Gateway Protocol). Letzteres arbeitet auf TCP-Port 179. EGP und HELLO benutzen weder TCP noch UDP, sondern sind sozusagen ihre eigenen Transportprotokolle. IP hat ein Feld, in dass das darüber liegende Protokoll eingetragen wird. TCP z.B. hat die Kennung 6, UDP die Kennung 17 und ICMP die Kennung 1. Nun kann man hier natürlich weitere Tansportprotokolle schaffen, und dies wird z.B. bei EGP (Protokollnummer 8) und HELLO (Protokollnummer 63) gemacht. Auch hier gibt es noch weitere Protokolle, die aber kaum eine Praktische Bedeutung haben, da währen z.B. das Host-Monitoring-Protocol HMP (Protokoll 20) oder das Reliable Datagram Protocol RDP (Protokoll 27).

Man beachte:
NETROM kam bisher nicht vor, auch AX.25 nicht. IP selber übernimmt das Routing, ist aber selber unabhängig von der darunter liegenden Netztechnik. Ob es sich um ein Ethernet, eine Funkverbindung, eine Telefonleitung oder eine RS232-Verbindung handelt, ist IP egal. Hauptsache, da ist etwas, was Datagramme entgegen nimmt und verschickt. Wie, ist nebensächlich. Nun haben AX.25-Frames (allgemein auch Pakete genannt), einen PID, oder auch Protocol Identifier. Für normale Textverbindungen ist dieser 240, oder auch Hexadezimal $F0. Anhand dieser Kennung erkennt der Empfänger, dass kein weiteres Protokoll verwendet wird. Bei NETROM ist der PID z.B. $CF, bei IP in der Regel $CC. Der PID kann aber auch andere Werte haben, die beiden oberen Bits des PID zeigen an, ob das Datagramm fragmentiert wurde und so aus mehreren AX.25-Frames besteht. IP-Datagramme können bis zu 64kB gross werden, ein Wert, denn man bei AX.25-Frames vermeiden sollte. Da die höheren Protokolle, z.B. TCP, für eine Flusskontrolle sorgen, werden IP-Datagramme meistens in UI-Paketen versand, d.h. die Gegenstation sendet keine Bestätigung für den Empfang eines AX.25-Frames. Die Bestätigung wird dann als TCP-Steuerpaket in ein IP-Datagramm verpackt und so als AX.25-Frame verschickt. Da TCP das sowieso macht, spart man sich so zwei Frames, nämlich auch die AX.25-Bestätigung für den Empfang der TCP-Bestätigung. Für schlechte Verbindungen kann man aber auch eine Verbindung im „connectet mode“ fahren, d.h. es wird eine normale AX.25-Verbindung aufgebaut, nur das dort halt Frames mit einem anderen PID als normal benutzt werden. Um nun solche Verbindungen auch über NETROM-Knoten zu benutzen, kann man NETROM noch dazwischen packen. Da NETROM aber selber im „connectet mode“ fährt und für sich auch wieder jedes Paket bestätigt, wird dadurch allerdings die Frequenz arg belastet. Die Aufschichtung der einzelnen Protokolle kann man sich dann in etwa folgendermaßen vorstellen:



Das obige Beispiel ist natürlich stark konstruiert. Bis auf wenige Wampes, bzw. Unix-Rechner wird wohl kaum jemand den SYSLOG benutzen. Und HELLO zu verwenden macht auch wenig Sinn. Ein ProNet-10-Interface ist an einer Amateurfunkstation auch nur schwer vorstellbar. Hier sollen aber eher die verschiedenen Möglichkeiten aufgezeigt werden, als nur der „Durchschnitts-TCP/IP-Rechner“. Man lasse sich nicht dadurch verwirren, dass zwei ARP für drei Interfaces eingetragen sind. Für jede Netzwerkhardware braucht man eine ARP-Tabelle, aber alle Architekturen lassen sich mit einer ARP-Implementation abdecken. Mehrere Interfaces gleicher Art, z.B. zwei AX.25-Links, brauchen natürlich nur eine

ARP-Tabelle.
Wer mehr über die einzelnen Protokolle wissen möchte, sollte sich nicht scheuen, in die entsprechenden RFC's zu sehen. Zu erhalten sind sie auf einigen Wampes-Rechnern, im Internet (z.B. ftp.informatik.uni-hamburg.de), oder bei mir, dl8xas@db0hht.ampr.org. Kurze RFC's (so bis ca. 30KB) verschicke ich noch als Mail, größere nur gegen Einsendung einer Diskette mit SASE. Vorher aber erst eine Mail an mich schicken (s.o.), denn alle RFC's habe ich auch nicht. Aber die meisten ab ca. 680 bis ca. 1820 (Stand 12/95) mit einigen Lücken habe ich da. Ansonsten gibt es auch durch den allgemeinen Internet- und WWW-Boom ein Haufen Bücher die sich damit beschäftigen.

Hier noch eine kurze Zusammenstellung der wichtigsten RFC's:

rfc791	Internet Protocol (IP)
rfc792	Internet Control Message Protocol (ICMP)
rfc793	Transmission Control Protocol (TCP)
rfc768	User Datagram Protocol (UDP)
rfc855	Telnet (auch 856-860 und diverse andere RFC’s)
rfc959	File Transfer Protocol (FTP)
rfc821	Simple Mail Transfer Protocol (SMTP)
rfc977	Network News Transfer Protocol (NNTP)
rfc937	Post Office Protocol - Version 2 (POP2)
rfc826	Address Resolution Protocol (ARP)
rfc1058	Routing Information Protocol (RIP)

Wer eine Protokollbeschreibung zu AX.25 sucht, siehe doch bitte bei der nächstgelegenen Box unter AX.25 nach einen Text von DK8HI. Eine NETROM-Protokollbeschreibung suche ich noch selber, im NETROM-Handbuch steht leider nur eine Header-Beschreibung.

73 de Andreas, DL8XAS @ DB0HHT

ANHANG K:

AX.25 Link-Layer Protokoll Spezifikation

AX.25 AMATEUR PACKET-RADIO LINK-LAYER PROTOKOLL VERSION 2.0

Übersetzt und bearbeitet von DK8HI als Beilage zum PRIMUS-Handbuch. (c) für die Übersetzung HPRG 1986.

Vorwort

Diese Beschreibung bezieht sich auf das AX.25 Level 2 Version 2 Protokoll des Amateurfunkdienstes /1/. Ein Protokoll ist ein Satz von Regeln, die den zuverlässigen Transport von Daten zwischen zwei Punkten sicherstellen sollen.

Die Bezeichnung AX.25 für dieses Protokoll bedeutet Amateur-X.25, d.h. es basiert auf dem CCITT X.25 Protokoll /2/ und wurde für die spezielle Umgebung des Amateurfunkdienstes angepasst und erweitert.

Level 2 (Ebene 2) kennzeichnet die Schicht des ISO 7-Schichten Modells, die das Verfahren zum Austausch von Nachrichten zwischen zwei Stationen beschreibt (Link-Layer). Der Vollständigkeit halber wurde hier auch die Beschreibung der Ebene 1 (physical Layer) mit aufgenommen. Die Beschreibung der Ebene 1 legt die physikalischen, elektrischen und verfahrensmäßigen Eigenschaften fest, um die physikalische Verbindung zwischen zwei Stationen herzustellen, zu erhalten und wieder aufzuheben. Dazu gehören im Fall des Amateurfunks die Sendefrequenzen, Modulationsarten und AFSK-Töne.

Die Version 2 dieses Protokolls enthält gegenüber der älteren Version eine genauere Definition der Kommando-/Antwortstruktur und die Möglichkeit, die Rufzeichen von bis zu 8 Digipeatern (von Digital- Repeater) im Adressfeld anzugeben. (Die Originalfirmware des TAPR-TNC-1 verwendet die Version 1, kann aber trotzdem schon 8 Digipeaterrufzeichen verarbeiten.)

1. Bedingungen für die Verbindung der Ebene 1

Als Modulationsverfahren wird, unabhängig von der Modulationsart, NRZI verwendet (zumindest bei Übertragungsgeschwindigkeiten bis 1200 Bd). Dabei wird ein zu übertagenes 0-Bit als ein Wechsel zwischen den Ton- oder Sendefrequenzen, ein 1-Bit als kein Wechsel übertragen. Demnach ist es nicht sinnvoll von Mark- oder Spacetönen zu sprechen. Die Aussendung kann ebensogut mit dem höheren, als auch mit dem niedrigeren Ton beginnen.

1.1 Frequenzen oberhalb von 144 MHz

In diesem Frequenzbereich werden zurzeit fast ausschließlich FM-Funkgeräte mit einem Kanalabstand von 25 kHz verwendet. In der Modulationsart F2B werden die Daten mit 1200 Baud übertragen. Die AFSK-Töne entsprechen denen der Bell-202-Norm mit 1200 und 2200 Hz. Auf Frequenzen oberhalb des 2-Meter-Bandes ist es geplant, die Daten auch mit höheren Geschwindigkeiten (9600 Bd, 56 oder 64 kBd) in der Modulationsart F1B zu übertragen.

1.2 Frequenzen unterhalb von 30 MHz

Für Verbindungen auf Kurzwelle wird die Modulationsart F1B mit einem Frequenzhub von 200 Hz und einer Geschwindigkeit von 300 Bd verwendet. Um den Einfluss von Störungen klein zu halten, wird zudem oft die Länge des Informationsfeldes auf 64 Bytes oder weniger begrenzt.

2. AX.25 Link-Layer Protokoll Spezifikation

2.1 Rahmen und Anwendungbereich

Das AX.25 Link-Layer Protokoll wurde entworfen, um einen zuverlässigen Transport von Daten zwischen zwei sendenden Endpunkten zu erreichen, unabhängig von anderen Protokollebenen, die evtl. implementiert sein könnten, insbesondere unabhängig von den verschiedenen Typen von Kommunikationsverbindungen der Ebene 1. So wie es definiert ist, kann dieses Protokoll ebenso gut in Halb- wie Vollduplexverbindungen des Amateurfunks arbeiten und ebenso gut zwischen zwei einzelnen Amateur Packet-Radio Stationen oder einer einzelnen Station und einem Multiport-Controller.

Dieses Protokoll entspricht den ISO Empfehlungen 3309, 4335 (einschliesslich DAD 1 und 2) und 6256 High-level Data Link Control (HDLC) und benutzt einiges aus der in diesen Dokumenten benutzten Terminologie. Es entspricht ebenso ANSI X3.66, welches ADCCP, balanced mode, beschreibt.

Dieses Protokoll hält sich an die CCITT X.25 Empfehlung, mit der Ausnahme eines erweiterten Adressfeldes und des zusätzlich aufgenommenen unnumerierten Informations- (UI) Blocks. Es folgt ebenso den Prinzipien der CCITT Empfehlung Q.921 (LAPD) im Gebrauch von mehreren Verbindungen, die sich, unterschieden durch das Adressfeld, einen einzigen, mehrfach verwendeten Übertragungskanal teilen.

Dieses Protokoll ermöglicht mehr als eine Link-Layer Verbindung pro Gerät, falls das Gerät dafür ausgelegt ist.

Dieses Protokoll verhindert keine Selbst-Verbindungen. Eine Selbst-Verbindung erhält man, wenn ein Gerät eine Verbindung aufbaut, mit der eigenen Adresse sowohl für den Absender, als auch für den Empfänger des Datenübertragungsblocks.

Die meisten Link-Layer Protokolle nehmen an, dass ein primäres (oder Meister-) Gerät (normalerweise DÜE oder Datenübertragungseinrichtung genannt, im Englischen DCE, data circuit-terminating equipment) mit einem oder mehreren sekundären Geräten (DEE, Datenendeinrichtungen, DTE data terminating equipment) verbunden ist. Diese Art des Betriebes mit nicht gleichberechtigten Stationen ist im Amateurfunk unvorteilhaft. Demgegenüber wird im AX.25-Protokoll angenommen, dass beide Enden der Verbindung gleichberechtigt sind. Dadurch entfallen die zwei verschiedenen Klassen der Geräte. In dieser Protokollbeschreibung wird der Ausdruck DXE (Datenschalteinrichtung, data switching equipment) gebraucht, um die gleichberechtigten Arten von Geräten zu beschreiben, die man im Amateur-Packet-Radio findet.

2.2 Aufbau des Datenübertragungsblocks

Link-Layer Packet-Radio Übertragungen werden in Datenübertragungsblocks (DÜ-Blocks, englisch frames) ausgesendet. Jeder Block besteht aus mehreren kleineren Gruppen, Felder genannt. Abb. 1 zeigt den Aufbau der drei grundlegenden Typen von Blocks. Man beachte, dass hier das zuerst übertragene Bit links steht.

```
zuerst übertragenes Bit
  |
-------------------------------------------------------------
|   Flag   |    Adresse     | Control |   FCS   |   Flag   |
|-----------------------------------------------------------|
| 01111110 | 112...560 Bits | 8  Bits | 16 Bits | 01111110 |
-------------------------------------------------------------
Abb. 1A - Aufbau von U und S Blocks

  zuerst übertragenes Bit
  |
-------------------------------------------------------------------------------
|   Flag   |    Adresse     | Control|  PID   | Info.    |   FCS   |   Flag   |
|-----------------------------------------------------------------------------|
| 01111110 | 112...560 Bits | 8 Bits | 8 Bits | N*8 Bits | 16 Bits | 01111110 |
-------------------------------------------------------------------------------
Abb. 1B - Aufbau von Informationsblocks
```

Sowohl in /1/ als auch in /2/ werden die Gruppen von 8 Bits als Oktett (engl.: octet) bezeichnet, um anzudeuten, dass HDLC als bitorientiertes Übertragungsverfahren unabhängig von einer Bytestruktur ist. In dieser Übersetzung wurde trotzdem das Wort „Byte“ verwendet, weil im AX.25 zunächst nur die Übertragung von Datenbytes vorgesehen ist. Im Prinzip ist jedoch auch im AX.25 die Übertragung von Daten möglich, die nicht als 8-bit-Bytes vorliegen. In diesem Fall werden die Bits ohne Beachtung der Bytegrenzen in die I-Felder eingetragen und auf eine durch 8 teilbare Anzahl aufgefüllt (siehe die Forderung im nächsten Absatz).

Jedes Feld besteht aus einer ganzen Zahl von Bytes und erfüllt die im Folgenden beschriebenen Funtionen.

2.2.1 Blockbegrenzung (Flag)

Das Blockbegrenzungszeichen (Flag) ist ein Byte lang und erscheint am Anfang und am Ende eines Blocks. Ein einzelnes Flag kann gleichzeitig das Ende des vorhergehenden und den Anfang des folgenden Blocks markieren. Ein Flag besteht aus einer Null, gefolgt von sechs Einsen, gefolgt von einer weiteren Null (01111110 = 7E hex). Um zu verhindern, dass diese Folge irgendwo innerhalb eines Blocks auftritt, verwendet man das so genannte Bit-Stuffing (siehe 2.2.6).

2.2.2 Adress-Feld

Das Adress-Feld wird verwendet, um sowohl den Absender als auch den Empfänger des Blocks zu kennzeichnen. Zusätzlich enthält das Adress-Feld die Kommando/Antwort-Informationen sowie die Angaben für den Betrieb über Level-2-Digipeater.

Die Kodierung des Adressfeldes wird in 2.2.13 genauer beschrieben.

2.2.3 Kontrollfeld

Das Kontrollfeld (Steuerfeld) wird benutzt, um den Typ des gerade übertragenen Blocks anzuzeigen und um den Status der Level-2-Verbindung zu kontrollieren. Es ist ein Byte lang, und seine Kodierung wird in 2.3.2.1 besprochen.

2.2.4 PID-Feld

Das Protokoll-Identifier-(PID-)Feld soll nur in Informationsblocks (I und UI) erscheinen. Es bestimmt, welche Art von Layer-3-Protokoll, wenn überhaupt, benutzt wird.

Das PID-Feld zählt bei der Länge des Informations-Feldes nicht mit. Es ist wie folgt kodiert:

```
M      L (Das LSB ist das zuerst übertragene Bit)
S      S
B      B
yy01yyyy  AX.25 Layer 3 implementiert.
yy10yyyy  AX.25 Layer 3 implementiert.
11001100  Internet Protokoll Datagramm Layer 3 implementiert.
11001101  Adress Resolution Protokoll Layer 3 implementiert.
11001111  NET/ROM Protokoll Layer 3/4 implementiert.
11110000  Kein Layer 3 implementiert.
11111111  Fluchtsymbol, das nächste Byte enthält weitere  Layer  3
          Protokoll Informationen.

Wobei y entweder 0 oder 1 sein kann.
```

Anmerkung:
Alle nicht oben aufgelisteten Formen yy00yyyy und yy11yyyy sind für zukünftige Layer 3 Protokolle reserviert. Die Zuordnung dieser Kodes ist Sache einer Übereinkunft der Funkamateure. Es wird empfohlen, dass die Entwickler von Layer-3-Protokollen sich mit dem ARRL Ad Hoc Committee on Digital Communications wegen vorgeschlagener Kodierungen in Verbindung setzen.

2.2.5 Informations-Feld

Das Informations-(I-)Feld wird benutzt, um Benutzerdaten von einem Ende der Verbindung zum anderen zu übertragen. I-Felder sind nur in drei Typen von Frames zulässig: dem I-Frame, dem UI-Frame und dem FRMR-Frame. Das I-Feld darf maximal 256 Bytes lang sein und muss immer eine ganze Zahl von Bytes enthalten. Diese Einschränkungen treffen zu, bevor die in 2.2.6 beschriebenen Null-Bits eingefügt werden. Die I-Felder sollen ohne jede Veränderung entlang der Übertragunsstrecke transportiert werden, mit der Ausnahme, dass die in 2.2.6 beschriebenen Null-Bits eingefügt werden, die nötig sind, damit keine Flag-Bytes innerhalb des Frames auftreten.

Das Informationsfeld der I- oder der UI-Blocks enthält die zu übertragenden Pakete. Ohne eine höhere Protokollschicht sind dieses die Textpakete, die von einer Station zur anderen gesendet werden sollen. Anderenfalls sind es die verschiedenen Typen von Paketen der höheren Protokollschichten.

2.2.6 Bit-Stuffing

Um sicherzustellen, dass die Bitfolge eines Flag-Bytes nirgends innerhalb eines Blocks auftritt, soll die sendende Station innerhalb eines Blocks nach jeweils 5 aufeinander folgenden Eins-Bits ein Null-Bit einfügen. Die empfangende Station soll jedes Null-Bit, dass nach einer Folge von genau 5 aufeinander folgenden Eins-Bits auftritt, wieder aus dem Datenstrom entfernen.

2.2.7 Blockprüfzeichenfolge

Die Blockprüfzeichenfolge (Frame-Check Sequence, FCS) ist eine sechzehn bit lange Zahl, die vom Sender und Empfänger des Blocks berechnet wird. Sie soll sicherstellen, dass der Block auf dem Weg vom Sender zum Empfänger nicht durch das Übertragungsmedium verfälscht wurde. Sie soll in Übereinstimmung mit der ISO Empfehlung 3309 (HDLC) berechnet werden.

2.2.8 Übertragunsreihenfolge der Bits

Mit Ausnahme des FCS-Feldes sollen alle Felder eines AX.25-Blocks mit dem niedrigwertigsten Bit der Bytes zuerst ausgestrahlt werden. Die FCS soll mit dem höchstwertigsten Bit zuerst ausgestrahlt werden.

2.2.9 Ungültige Blocks

Jeder aus weniger als 136 Bits (exklusive der Flags am Anfang und am Ende) bestehende, oder nicht aus einer ganzen Anzahl von Bytes bestehende Block soll von dem Link-Layer als ungültiger Block angesehen werden. Siehe dazu auch 2.4.4.4.

2.2.10 Block-Abbruch

Wenn die Aussendung eines Blocks vorzeitig abgebrochen werden muss, sollen mindestens 15 aufeinander folgende Eins-Bits, ohne Bit-Stuffing, gesendet werden.

2.2.11 Füllzeichen zwischen Blocks

Wann immer es für eine DXE nötig ist, den Sender eingeschaltet zu lassen, ohne einen Block auszusenden, soll die Zeit zwischen den Blocks mit aufeinander folgenden Flags gefüllt werden. (Auch in der Zeit nach dem Hochtasten des Senders bis zur Aussendung des ersten Blocks sollen aufeinander folgende Flags gesendet werden.)

2.2.12 Zustände am Übermittlungsabschnitt

Im Amateur Packet-Radio findet man vor allem Halbduplexverbindungen mit mehrfacher Ausnutzung des Übertragungskanals. Die Zustände am Übermittlungsabschnitt lassen sich deshalb nicht so klar definieren wie im X.25 Protokoll. Im Originaltext der AX.25-Definition steht an dieser Stelle deshalb nur ein kurzes „nicht anwendbar“.

2.2.13 Kodierung des Adressfeldes

Im Adressfeld eines Blocks sollen sowohl das Rufzeichen des Empfängers als auch das des Absenders dieses Blocks enthalten sein. Mit Ausnahme des Zweitstations-ID (Secondary Station Identifier, SSID), soll das Adressfeld nur aus Grossbuchstaben und Ziffern des ASCII-Codes bestehen. Falls Level-2-Digipeater benutzt werden, sollen auch ihre Rufzeichen mit im Adressfeld enthalten sein.

AX.25 verwendet das erweiterte Adressfeld nach DIN 66221. Dabei kann das HDLC Adressfeld eine beliebige Anzahl Bytes lang sein. Das niedrig-wertigste Bit eines Adress-Bytes wird als Erweiterungsbit verwendet. Ist dieses Bit Null, so folgt ein weiteres Adressbyte, ist es Eins, so kennzeichnet es das letzte Byte im Adressfeld. Um für dieses Erweiterungsbit Platz zu schaffen, wird das Rufzeichen um ein Bit nach links verschoben. (Im AX.25 ist die Anzahl der Bytes im Adressfeld immer durch 7 teilbar. Das kürzeste Adressfeld besteht aus 14 Bytes (Sender- und Empfängerrufzeichen), das längste aus 70 Bytes (Sender-, Empfänger- und 8 Digipeaterrufzeichen).)

2.2.13.1 Kodierung des Adressfeldes ohne Digipeater

Falls man keine Level-2-Digipeater verwendet, wird das Adressfeld wie in Abb. 2 gezeigt kodiert. Die Empfänger-Adresse enthält das Rufzeichen der Station, an die das Frame gesendet wird, die Absender-Adresse das Rufzeichen der Station, die das Frame ausgesandt hat. Diese beiden Rufzeichen definieren nur die beiden Endpunkte einer Level-2-Verbindung.

```
Erstes ausgestrahltes Byte
 |
-------------------------------------------------
|            Adressfeld eines Blocks             |
|-----------------------------------------------|
| Empfänger-Adresse   |     Absender-Adresse     |
|-----------------------------------------------|
|A1 A2 A3 A4 A5 A6 A7 | A8 A9 A10 A11 A12 A13 A14|
-------------------------------------------------
Abb. 2 -- Kodierung des Adressfeldes ohne Digipeater
```

A1 bis A14 sind die vierzehn Bytes, aus denen die beiden Adressubfelder des Adressfeldes bestehen. Die Empfänger-Subadresse ist 7 Bytes lang und wird zuerst ausgestrahlt. Diese Adressfolge gibt dem Empfänger die Möglichkeit, das Empfänger-Adresssubfeld zu prüfen, während er den Rest des Frames empfängt. Das Absender-Adresssubfeld wird dann anschliessend in den Bytes A8 bis A14 übertragen. Diese Reihenfolge entspricht auch der im Funkdienst üblichen Praxis, zuerst das Rufzeichen des Empfängers, dann das des Absenders zu nennen. Beide Subfelder sind in der gleichen Weise kodiert, mit der Ausnahme, das im letzten Byte der Absender-Adresse das HDLC-Erweiterungsbit gesetzt ist.

2.2.13.1.1 Kodierung des Empfänger-Subfeldes

Abb. 3 zeigt den Aufbau des Emfänger-Subfeldes, das in den Bytes A1 bis A7 des Adressfeldes übertragen wird. In der binären Darstellung befindet sich das zuerst übertragene Bit in der rechten oberen Ecke.

```
---------------------------------
| Byte | ASCII |  Binär   | Hex. |
|-------------------------------|
|  A1  |   D   | 10001000 |  88  |
|  A2  |   L   | 10011000 |  98  |
|  A3  |   3   | 01100110 |  66  |
|  A4  |   H   | 10010000 |  90  |
|  A5  |   C   | 10000110 |  86  |
|  A6  |   M   | 10011010 |  9A  |
|  A7  | SSID  | CRRSSID0 |      |
---------------------------------
Bit Position --> 76543210

Abb. 3 -- Kodierung des Empfänger-Subfeldes
```

Als weitere Erklärung dazu:

1. Das erste Byte (A1) ist das zuerst ausgestrahlte.
2. Das Bit 0 jedes Bytes ist das Adresserweiterungsbit des HDLC. Es ist Null für alle Bytes, bis auf das letzte Adressbyte. Im letzten Adressbyte wird es auf Eins gesetzt.
3. Das Rufzeichen soll nur aus den Grossbuchstaben und Ziffern des ASCII-Codes bestehen, jedoch um ein Bit zum höherwertigen verschoben werden, um das niedrigwertigste Bit für das Erweiterungsbit freizuhalten. Ist das Rufzeichen weniger als 6 Zeichen lang, so soll es mit Leerzeichen auf 6 Zeichen aufgefüllt werden.
4. Byte A7 enthält den SSID (Secondary Station IDentifier). Der SSID ermöglicht es, mehrere Packet-Radio Stationen unter dem gleichen Rufzeichen zu betreiben, sie aber unabhängig voneinander zu adressieren. Der SSID 0000 ist der Standard-SSID, der von einer „normalen“ Packet-Radio Station benutzt werden soll.
5. Die mit „R“ markierten Bits sind für zukünftige Erweiterungen reserviert. Man kann sie in individuellen Netzen auf Grund einer Übereinkunft benutzen. Falls sie nicht verwendet werden, sollen sie auf Eins gesetzt sein.
6. Das mit „C“ markierte Bit wird, wie in 2.4.1.2 beschrieben, als Kommando-/Antwortbit eines AX.25-Blocks verwendet.

2.2.13.1.2 Kodierung des Absender-Subfeldes

Das Absender-Subfeld wird in der gleichen Weise kodiert, wie das Empfänger-Subfeld und in den Adressbytes A8 bis A14 ausgestrahlt. Falls keine Level-2-Digipeater verwendet werden, ist das Bit 0 im SSID-Byte (A14) auf Eins gesetzt.

Byte	ASCII	Binär	Hex.
A8	D	10001000	88
A9	K	10010110	96
A10	8	01110000	70
A11	H	10010000	90
A12	I	10010010	92
A13		01000000	40
A14	SSID	CRRSSID1	

Bit Position --> 76543210

Abb. 3A -- Kodierung des Absender-Subfeldes

2.2.13.2 Kodierung der Level-2-Digipeateradresse

Wenn ein Block über einen Level-2-Digipeater (Digital-Repeater) übertragen werden soll, wird ein zusätzliches Adress-Subfeld als Adressbytes A15 bis A21 an die Adresse angehängt. Dann ist das Bit 0 von A14 auf Null gesetzt, um anzuzeigen, dass ein weiters Adress-Subfeld folgt und das Bit 0 von A21 auf Eins um das Ende des Adressfeldes anzuzeigen.

Das Digipeater-Subfeld ist in der gleichen Weise kodiert, wie das Empfänger-Subfeld, mit der Ausnahme, dass das höchstwertigste Bit im letzten Byte (A21) als „H“-Bit gekennzeichnet wird und anzeigen soll, ob der Block von einem Digipeater bereits übertragen wurde. Auf dem Weg zum Digipeater soll das Bit auf Null gesetzt sein. Der Digipeater setzt dann das Bit, bevor der Block wieder ausgestrahlt wird, auf Eins.

Byte	ASCII	Binär	Hex.
A15	D	10001000	88
A16	L	10011000	98
A17	0	01100000	60
A18	C	10000110	86
A19	C	10000110	86
A20	C	10000110	86
A21	SSID	HRRSSID1	

Bit Position --> 76543210

Abb. 4 -- Kodierung des Digipeater-Subfeldes

2.2.13.3 Betrieb über mehrere Digipeater

Das Link-Layer AX.25 Protokoll gestattet den Betrieb über mehr als einen Digipeater und schafft damit die Möglichkeit eines einfachen Mechanismus zur Weiterleitung von Blocks. Insgesamt dürfen bis zu acht Digipeater-Subfelder im Adressfeld stehen. Dabei ist das erste Digipeater-Subfeld, das auf die Absenderadresse folgt, auch die Adresse des ersten Digipeaters, über den der Block übertragen werden soll.

Die Digipeater setzten jeweils das „H“-Bit des zu ihnen gehörenden Adress-Subfeldes um anzuzeigen, dass sie den Block übertragen haben. Ausser der notwendigen Neuberechnung der FCS sollen am Block keine weiteren Veränderungen vorgenommen werden. Die empfangende Station kann anhand des Adressfeldes sehen, über welchen Weg der Block zu ihr gelangt ist.

Die Anzahl der Digipeater-Subfelder ist variabel. Nur das Bit 0 des SSID-Bytes des letzten Digipeaters wird auf Eins gesetzt, um anzuzeigen, dass das Adressfeld hier aufhört. In allen anderen SSID-Bytes ist das Bit 0 auf Null gesetzt.

Die Werte der verschiedenen Zeitgeber müssen zum Teil an die zusätzlichen Verzögerungszeiten angepasst werden, wenn ein Block über viele Digipeater weitergereicht wird, und die Bestätigung den gleichen Weg wieder zurücklaufen muss.

Man erwartet, dass der Betrieb über mehrere Digipeater ein vorübergehendes Verfahren sein wird, um grosse Entfernungen zu überbrücken, bis ein Level-3-Protokoll eingeführt ist. Danach sollte die Übertragung über mehrere Level-2-Digipeater nicht mehr verwendet werden.

2.3 Kenngrössen des Steuerungsverfahrens/der Übermittlungsvorschrift

2.3.1 Elemente des Verfahrens

Die Elemente des Verfahrens sind in Form von Aktionen beschrieben, die sich nach dem Empfang von Blocks ergeben. (Klar?)

Die Elemente des Verfahrens sind die verschiedenen Befehle und Meldungen, die in Form von U-, S- und I-Blocks übermittelt werden. Die Art des Befehls oder der Meldung ergibt sich aus dem Kontrollfeld des Blocks, sie ist im Blocktyp kodiert. Die Übermittlungsvorschrift muss ferner noch den Zustand der Folgezähler (state vabiables) beachten, um nach dem Empfang eines Blocks die zutreffende Aktion auszulösen.

Abschnitt 2.2 handelte vom grundsätzlichen Aufbau der Übermittlungsblocks. Hier, in 2.3, werden die verschiedenen Typen von Kontrollfeldern und Folgezählern definiert. Abschnitt 2.4 enthält die Beschreibung der Übermittlungsvorschrift selbst.

2.3.2 Formate des Kontrollfeldes und der Folgezähler

2.3.2.1 Formate des Kontrollfeldes

Das Kontrollfeld gibt den Typ des übertragenen Blocks an und wird verwendet, um Befehle und Meldungen zwischen den Endpunkten der Verbindung auszutauschen und damit die Verbindung zu überwachen.

Das AX.25 verwendet die Kontrollfelder des CCITT X.25 für den Betrieb mit gleichberechtigten Stationen (balanced operation, LABP), mit einem zusätzlichen Kontrollfeld, das dem ADCCP entnommen ist, um auch den Betrieb ohne aufgebaute Verbindung und Aussendungen an mehrere Stationen gleichzeitig zu ermöglichen.

Im Allgemeinen gibt es drei verschiedene Typen von AX.25-Blocks: Informationsblocks (I-Blocks), Steuerblocks mit Folgenummern (Supervisor-Blocks, S-Blocks) und Steuerblocks ohne Folgenummern (Unnumbered-Blocks, U-Blocks). Abb. 5 zeigt die grundlegenden Formate des Kontrollfeldes dieser drei Typen von Blocks.

```
-----------------------------------------
| Kontrollfeld |    Kontrollfeld-Bits     |
|     Typ      | 7 6 5 |  4  | 3 2  1   0 |
|---------------------------------------|
|    I-Block   |  N(R) |  P  |  N(S)   | 0 |
|---------------------------------------|
|    S-Block   |  N(R) | P/F | S S | 0   1 |
|---------------------------------------|
|    U-Block   | M M M | P/F | M M | 1   1 |
-----------------------------------------
Abb. 5 -- Formate des Kontrollfeldes
```

Dazu:

1. Bit 0 ist das zuerst, Bit 7 das zuletzt ausgesandte Bit des Kontrollfeldes.
2. N(S) ist die Sendefolgenummer (Bit 1 ist das Bit mit der niedrigsten Wertigkeit).
3. N(R) ist die Empfangsfolgenummer (Bit 5 ist das Bit mit der niedrigsten Wertigkeit).
4. Die „S“-Bits spezifizieren eine bestimmte Steuerungsfunktion in den Supervisor-Blocks. Ihre Kodierung wird in 2.3.4.2 beschrieben.
5. Die „M“-Bits spezifizieren eine bestimmte Steuerungsfunktion in den unnummerierten Blocks. Ihre Kodierung wird in 2.3.4.3 beschrieben.
6. Das Poll/Final-Bit (P/F-Bit) ist das Bit zur Sendeaufruf/Ende-Anzeige. Seine Funktion wird in 2.3.3 behandelt, das Verfahren zu seiner Benutzung in 2.4.2. Die Unterscheidung zwischen Sendeaufruf und Ende, d.h. zwischen Kommando und Antwort ist durch die Adressierungsregeln in 2.4.1.2 festgelegt.

2.3.2.1.1 Format der Informationsblocks

I-Blocks werden zur Übertragung von Datenpaketen (Text oder Pakete höherer Protokollschichten) verwendet. In I-Blocks ist das Bit 0 des Kontrollfeldes auf Null gesetzt. N(S) ist die Sendefolgenummer des Absenders (die Sendefolgenummer dieses Blocks). N(R) ist die Empfangsfolgenummer des Absenders (die Folgenummer des nächsten erwarteten I-Blocks in der anderen Richtung) und kann zur evtl. Bestätigung noch nicht bestätigter I-Blocks dienen. Die Folgenummern sind in 2.3.2.4 beschrieben.

2.3.2.1.2 Format der Supervisor-Blocks

S-Blocks werden für überwachende Steuerfunktionen benutzt, wie die Bestätigung von Datenblocks, die Aufforderung zur Wiederholung von Blocks oder die zeitweilige Unterbrechung der Übertragung von Blocks. In S-Blocks ist das Bit 0 des Kontrollfeldes auf Eins, das Bit 1 auf Null gesetzt. S-Blocks besitzen kein Informationsfeld.

2.3.2.1.3 Format der unnumerierten Blocks

U-Blocks werden zur Darstellung weiterer Steuerfunktionen benutzt. Sie sind unter anderem für den Aufbau und die Trennung einer Verbindung verantwortlich. Ausserdem erlauben die U-Blocks die Übertragung von Informationen ausserhalb der normalen Flusskontrolle. In U-Blocks sind die Bits 0 und 1 des Kontrollfeldes beide auf Eins gesetzt. Einige Typen von U-Blocks können Informations- und PID-Felder enthalten.

2.3.2.2 Parameter des Kontrollfeldes

Die verschieden Teile des Kontrollfeldes werden in den folgenden Abschnitten beschrieben. (Die Numerierung scheint hier schon im Original der X.25 Beschreibung nicht ganz konsistent zu sein...)

2.3.2.3 Folgenummern

Jeder Block ist fortlaufend numeriert (DIN-Deutsch: benummert), wobei diese Nummer modulo 8 von 0 bis 7 laufen soll. Dadurch sind maximal 7 unbestätigte Blocks in einer Level 2 Verbindung möglich.

2.3.2.4 Folgezähler und Folgenummern

2.3.2.4.1 Sendefolgezähler V(S) (send state variable)

Der Sendefolgezähler ist eine interne Variable der DXE. Er enthält die Folgenummer des nächsten auszustrahlenden I-Blocks und wird nach der Aussendung jedes neuen I-Blocks um eins erhöht.

2.3.2.4.2 Sendefolgenummer N(S) (send sequence number)

Die Sendefolgenummer ist im Kontrollfeld aller I-Blocks enthalten und stellt die laufende Nummer des Blocks dar. Vor der Aussendung eines I-Blocks wird die Sendefolgenummer gleich dem Inhalt des Sendefolgezählers gesetzt.

2.3.2.4.3 Empfangsfolgezähler V(R) (receive state variable)

Der Empfangsfolgezähler ist eine interne Variable der DXE. Er enthält die Folgenummer des nächsten erwarteten I-Blocks und wird erhöht, wenn ein fehlerfreier I-Block empfangen wurde, dessen Folgenummer gleich dem Wert des Empfangsfolgezählers ist.

2.3.2.4.4 Empfangsfolgenummer N(R) (receive sequence number)

Die Empfangsfolgenummer ist in den Kontrollfeldern der I- und S-Blocks enthalten. Vor der Aussendung eines I- oder S-Blocks wird die Empfangsfolgenummer auf den Wert des Empfangsfolgezählers gesetzt. Dadurch wird angezeigt, dass die DXE, die N(R) aussendet, alle I-Blocks der Gegenstation mit Folgenummern bis zum Wert N(R)-1 richtig empfangen hat.

2.3.3 Aufgabe des Poll/Final-Bits

Das Poll/Final-Bit wird in allen Blocktypen verwendet. Als Kommando (Poll) fordert es eine sofortige Antwort auf einen Block an. Die Antwort auf dieses Kommando ist durch ein ebenfalls gesetztes Bit (Final) angezeigt. In jeder Richtung der Verbindung ist nur ein solcher Poll-Zustand zurzeit gestattet. Das Verfahren zur Verwendung des P/F-Bits ist in 2.4.2 beschrieben.

2.3.4 Kommandos und Meldungen

Folgende Kommandos und Meldungen, angezeigt durch das Kontrollfeld, können von einer DXE verwendet werden:

2.3.4.1 Befehl zur Datenübermittlung mit Flusskontrolle

I-Blocks übertragen Daten- oder Textpakete unter Flusskontrolle. Das Kontrollfeld von I-Blocks ist in Abb. 6 gezeigt. Die Übertragung von Daten in I-Blocks wird stets als Kommando angesehen (gekennzeichnet durch ein P im Feld für das P/F-Bit). I-Blocks sind im Kontrollfeld durch N(S) fortlaufend (modulo 8) numeriert, um ihren Weg entlang der Link-Layer Verbindung verfolgen zu können.

2.3.4.2 Supervisor-Blocks

Die Kontrollfelder der drei verschiedenen Typen von Supervisor-Blocks sind in Abb. 6 dargestellt.

2.3.4.2.1 Meldung oder Kommando „empfangsbereit“ (RR, receive ready)

Die Meldung oder das Kommando Empfangsbereit wird in einem RR-Block übertragen. Als Meldung wird der RR-Block zu folgendem verwendet:

1. um anzuzeigen, dass der Absender des RR-Blocks in der Lage ist, weitere Informationsblocks zu empfangen,
2. um den Empfang der einwandfrei empfangenen I-Blocks bis zur Nummer N(R)-1 zu bestätigen und
3. um einen evtl. vorher bestehenden Zustand „nicht Empfangsbereit“ aufzuheben.

Man kann den Zustand des anderen Endes der Verbindung durch Senden eines RR-Blocks als Kommando mit gesetztem P-Bit erfragen.

2.3.4.2.2 Meldung oder Kommando „nicht empfangsbereit“ (RNR, receive not ready)

Ein RNR-Block wird von einer DXE ausgestrahlt, um anzuzeigen, dass sie zurzeit beschäftigt ist und keine weiteren I-Blocks empfangen kann. Empfangene I-Blocks bis zur Nummer N(R)-1 gelten als bestätigt. Alle I-Blocks mit einer Nummer N(R) und höher, die evtl. während des Zustandswechsels übertragen wurden, sind nicht bestätigt.

Der Zustand „nicht empfangsbereit“ kann aufgehoben werden durch das Aussenden eines UA-, RR-, REJ- oder SABM-Blocks.

Man kann den Zustand des anderen Endes der Verbindung durch Senden eines RNR-Blocks als Kommando mit gesetztem P-Bit erfragen.

2.3.4.2.3 Meldung oder Kommando „Wiederholungsaufforderung“ (REJ, reject)

Ein REJ-Block wird von einer DXE ausgestrahlt, um eine Wiederholung von I-Blocks ab der Nummer N(R) anzufordern. Alle I-Blocks mit einer Folgenummer bis N(R)-1 sind bestätigt. Falls neue I-Blocks zur Übertragung anstehen, dürfen diese anschliessend an die Wiederholung ausgestrahlt werden.

In jeder Richtung einer Verbindung ist nur ein Zustand „Wiederholungsaufforderung“ möglich. Aufgehoben wird dieser Zustand durch den einwandfreien Empfang des I-Blocks, der diesen Zustand ausgelöst hat.

Man kann den Zustand des anderen Endes der Verbindung durch Senden eines REJ-Blocks als Kommando mit gesetztem P-Bit erfragen.

Kommando oder Meldung		Bits im Kontrollfeld					
		7 6 5	4	3 2	1	0	
Informationsübertragung	I	N(R)	P	N(S)			0
Empfangsbereit	RR	N(R)	P/F	0 0	0	1	
Nicht empfangsbereit	RNR	N(R)	P/F	0 1	0	1	
Wiederholungsaufforderung	REJ	N(R)	P/F	1 0	0	1	
Verbindungsanforderung	SABM	0 0 1	P	1 1	1	1	
Verbindungsabbruch	DISC	0 1 0	P	0 0	1	1	
Verbindungsrückweisung	DM	0 0 0	F	1 1	1	1	
Unnummerierte Bestätigung	UA	0 1 1	F	0 0	1	1	
Rückweisung eines Blocks	FRMR	1 0 0	F	0 1	1	1	
Unnummerierte Information	UI	0 0 0	P/F	0 0	1	1	

Abb. 6 -- Kodierung der Kontrollfelder

2.3.4.3 Nicht numerierte Blocks

Die Kontrollfelder der nicht numerierten Blocks sind in Abb. 6 dargestellt.

2.3.4.3.1 Kommando „Verbindungsanforderung“ (SABM, set asynchronous balanced mode)

Das Kommando SABM wird dazu benutzt, zwei DXE's in den Zustand „asynchronous balanced mode“ - „gleichberechtigter Spontanbetrieb“ zu versetzen. Der Betrieb mit diesem Zustand ist bekannt unter der Bezeichnung LAPB (link acces procedure balanced), bei dem beide Stationen gleichberechtigt sind.

In einem SABM-Block darf kein Informationsfeld enthalten sein. Evtl. vorher ausgesandte und noch nicht bestätigte I-Blocks bleiben unbestätigt.

Eine DXE bestätigt den Empfang und die Annahme eines SABM-Kommandos durch die Aussendung einer UA-Meldung zum frühest möglichen Zeitpunkt. Falls die DXE das SABM-Kommando nicht akzeptieren kann, so soll sie, falls möglich, einen DM-Block aussenden.

2.3.4.3.2 Kommando „Verbindungsabbruch“ (DISC, disconnect)

Das Kommando DISC wird dazu benutzt, die Verbindung zwischen zwei Stationen auf der Link-Ebene zu trennen. Es darf in einem DISC-Block kein Informationsfeld enthalten sein.

Bevor die empfangende DXE den DISC-Block bearbeitet, bestätigt sie dessen Empfang durch Aussendung einer UA-Meldung zum frühest möglichen Zeitpunkt. Die DXE, die den DISC-Block ausgesandt hat, trennt die Verbindung, wenn sie die UA-Meldung empfängt. Evtl. vorher ausgesandte und noch nicht bestätigte I-Blocks bleiben unbestätigt.

2.3.4.3.3 Meldung „Rückweisung eines Blocks“ (FRMR, frame reject)

Der FRMR-Block wird ausgesandt, um anzuzeigen, dass ein empfangener Block nicht ausgewertet und der Fehler nicht durch die nochmalige Übertragung dieses Blocks behoben werden kann. Normalerweise tritt dieser Fall ein, wenn ein Block mit korrektem FCS-Feld empfangen wurde und eine der folgenden Bedingungen zutrifft:

1. Empfang eines ungültigen oder nicht implementierten Kommando- oder Meldungsblocks, d.h. der Block hat ein unbekanntes Kontrollfeld.
2. Empfang eines I-Blocks, dessen I-Feld länger ist als die vereinbarte maximale Länge.
3. Empfang eines Blocks mit einer ungültigen N(R), d.h., die N(R) bestätigt einen I-Block, der noch nicht übermittelt worden ist.
4. Empfang eines Blocks mit einem I-Feld, wenn in diesem Blocktyp kein I-Feld zugelassen ist. Um dieses anzuzeigen sollen die Bits W und Y im I-Feld des FRMR-Blocks gesetzt werden.
5. Empfang eines S-Blocks mit gesetztem F-Bit, wenn dieses nicht als Antwort auf einen Block mit gesetztem P-Bit folgt. Bit W im I-Feld des FRMR-Blocks soll gesetzt werden.
6. Empfang eines unerwarteten UA- oder DM-Blocks. Bit W soll gesetzt werden.
7. Empfang eines I-Blocks mit einer ungültigen N(S), d.h., die N(S) ist grösser als die N(S) des zuletzt bestätigten I-Blocks plus die maximal zulässige Anzahl unbestätigter I-Blocks.

FRMR-Blocks enthalten ein drei Byte langes I-Feld, das zusätzliche Informationen darüber enthält, wo der Fehler aufgetreten ist. Das

Format des I-Feldes ist in Abb. 7 dargestellt.

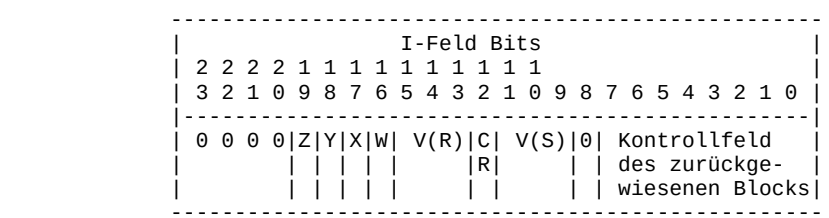


Abb. 7 -- I-Feld des FRMR-Blocks

Dazu:

1. Bits 0-7 des I-Feldes enthalten das Kontrollfeld des Blocks, der die Zurückweisung ausgelöst hat.
2. V(S) ist der Sendefolgezähler der DXE, die den FRMR-Block aussendet.
3. Das CR-Bit ist Null, wenn der zurückgewiesene Block ein Kommando, Eins, wenn er eine Meldung war.
4. V(R) ist der Empfangsfolgezähler der DXE, die den FRMR-Block aussendet.
5. Bit W wird gesetzt, wenn das empfangene Kontrollfeld ungültig oder nicht implementiert ist.
6. Bit X wird gesetzt, wenn der empfangene Block ein I-Feld enthält obwohl in diesem Typ von Block kein I-Feld zugelassen ist. Zusätzlich zum X-Bit soll das W-Bit gesetzt werden.
7. Bit Y wird gesetzt, wenn der empfangene I-Block ein I-Feld enthält, das länger ist als die vereinbarte maximale Länge.
8. Bit Z wird gesetzt, wenn ein Block mit einer ungültigen N(R) empfangen wurde.
9. Bits 8 und 20 bis 23 sind Null.

2.3.4.3.4 Meldung „Unnummerierte Bestätigung“ (UA, unnumbered acknowledge)

Die UA-Meldung wird verwendet, um dem Empfang und die Annahme eines SABM- oder DISC-Kommandos zu bestätigen. Das empfangene Kommando wird erst bearbeitet, nachdem die UA-Meldung ausgesandt ist. UA-Blocks dürfen kein I-Feld enthalten.

2.3.4.3.5 Meldung „Verbindungsrückweisung“ (DM, disconnected mode)

Die DM-Meldung wird von einer DXE ausgesandt, falls sie Blocks von einer Station empfängt, mit der sie nicht verbunden ist, die keine SABM- oder UI-Blocks sind. In diesem Fall soll im Kontrollfeld des DM-Blocks das F-Bit auf Eins gesetzt sein.

Die DM-Meldung wird auch als Antwort auf einen SABM-Block ausgesandt, um anzuzeigen, dass die DXE z.Zt. keine Verbindung annehmen kann. (Dieses ist der Fall, wenn die DXE bereits die maximal mögliche Anzahl von Verbindungen hergestellt hat, also besetzt ist, oder wenn die DXE keine Verbindungen annehmen soll.)

DM-Blocks enthalten kein I-Feld.

2.3.4.3.6 Kommando oder Meldung „Unnummerierte Information“ (UI, unnumbered information)

UI-Blocks können sowohl als Kommando als auch als Meldung ausgesandt werden.

UI-Blocks enthalten ein PID- und ein Informationsfeld und werden verwendet, um Informationen entlang der Verbindung ohne Flusskontrolle zu transportieren. Da diese Blocks nicht der Flusskontrolle unterliegen, kann man den Verlust eines UI-Blocks nicht erkennen oder korrigieren.

Ein empfangener UI-Block mit gesetztem P-Bit sollte bestätigt werden. Die Antwort sollte ein DM-Block sein, wenn die DXE nicht mit der anderen verbunden ist, sonst ein RR- oder RNR-Block.

2.3.5 Meldung von Fehlern und deren Berichtigung

Es gibt einige Fehler, die auf dem Link-Level entstehen können und die man berichtigen kann, ohne die Verbindung abzubrechen. Dieses sind z.B. Übertragungsfehler oder Ablaufstörungen innerhalb der DXE.

2.3.5.1 Nicht-Empfangsbereitschaft

Falls eine DXE zeitweilig nicht in der Lage ist, weitere Blocks zu empfangen, weil z.B. die Empfangspuffer voll sind, so wird sie dieses der Gegenstation durch Aussenden eines Blocks mit der Meldung RNR (nicht empfangsbereit) mitteilen. Das Ende der Nicht-Empfangsbereitschaft wird der Gegenstation durch Aussenden eines RR-, REJ-, UA- oder SABM-Blocks angezeigt.

2.3.5.2 Fehler in der Sendefolgenummer

Falls die Sendefolgenummer N(S) eines ohne CRC-Fehler empfangenen I-Blocks nicht mit dem Empfangsfolgezähler V(R) übereinstimmt, dann ist ein Fehler in der Sendefolgenummer aufgetreten, und das I-Feld dieses Blocks wird verworfen. Der Empfänger wird diesen Block, wie auch folgende Blocks nicht bestätigen, bis er einen Block empfängt, dessen Sendefolgenummer mit dem Empfangsfolgezähler übereinstimmt.

Das Kontrollfeld solcher I-Blocks wird jedoch ausgewertet und zur Steuerung der Verbindung verwendet. Z.B. kann die Empfangsfolgenummer die in der Gegenrichtung ausgestrahlten I-Blocks bestätigen.

(Fehler in der Sendefolgenummer treten auf, wenn zwischenzeitlich ein I-Block gestört wurde und auf Grund eines CRC-Fehlers nicht ausgewertet werden konnte.)

2.3.5.3 Wiederholung nach REJ

Eine DXE verwendet die Meldung REJ dazu, die Gegenstation zur Wiederholung von I-Blocks aufzufordern, nachdem sie einen Fehler in der Sendefolgenummer erkannt hat. Es ist in jeder Richtung nur eine Wiederholungsaufforderung zurzeit zulässig.

Die Gegenstation berichtigt den Fehler, indem sie alle unbestätigten I-Blocks, beginnend mit dem im REJ-Block angeforderten, wiederholt (maximal soviel, wie die erlaubte Zahl unbestätigter I-Blocks).

2.3.5.4 Wiederherstellung nach Ablauf der Zeitüberwachung

2.3.5.4.1 Wiederherstellung nach Ablauf von Zeitgeber T1

Falls eine DXE wegen eines Übertragungsfehlers einen einzelnen I-Block oder den letzten I-Block einer Folge nicht empfängt, so wird sie keinen Fehler in der Sendefolgenummer feststellen und somit auch keinen REJ-Block senden. Die andere DXE, die die unbestätigten I-Blocks ausgesandt hat, soll nach Ablauf des Zeitgebers T1 geeignete Massnahmen einleiten, um zu bestimmen, mit welchem I-Block eine Wiederholung begonnen werden muss. Die Massnahmen sind in 2.4.4.9 beschrieben. Dieser Zustand wird aufgehoben durch den Empfang einer Bestätigung der ausgesandten I-Blocks oder durch das Rücksetzen der Verbindung (siehe 2.4.6).

2.3.5.4.2 Wiederherstellung nach Ablauf von Zeitgeber T3

Der Zeitgeber T3 soll sicherstellen, dass die Verbindung noch besteht, auch wenn längere Zeit keine Information übertragen wurde. Immer wenn T1 nicht läuft (keine unbestätigten I-Blocks), wird T3 benutzt, um in regelmässigen Abständen die andere DXE der Verbindung abzufragen. Wenn T3 abläuft, wird ein RR- oder RNR-Block als Kommando mit gesetztem P-Bit ausgesandt. Anschliessend wird wie in 2.4.4.9 beschrieben auf die Bestätigung gewartet.

(Ausserdem könnte der RR-Block einer DXE, die nach einem Zustand „nicht empfangsbereit“ wieder empfangsbereit wird, verloren gehen. Mit Hilfe des Zeitgebers T3 kann dann festgestellt werden, dass diese Station empfangsbereit ist, da sie mit einem RR-Block antworten wird, nicht mit einem RNR-Block.)

2.3.5.5 Fehler in der Blockprüfzeichenfolge

Ein Block mit einem Fehler in der Blockprüfzeichenfolge wird verworfen, ohne dass weitere Aktionen veranlasst werden.

2.3.5.6 Zustand der Blockrückweisung

Der Zustand der Blockrückweisung tritt ein, sobald ein sonst fehlerfreier Block empfangen wurde, der mindestens eine der Bedingungen aus 2.3.4.3.3 erfüllt.

Ist einmal der Zustand der Zurückweisung eingetreten, so werden keine weiteren I-Blocks akzeptiert (mit Ausnahme der Beachtung des P/F-Bits), bis der Fehler beseitigt ist. Der Fehlerzustand wird der anderen DXE durch Aussenden eines FRMR-Blocks angezeigt (siehe 2.4.5).

2.4 Beschreibung der Übermittlungsvorschrift

In den folgenden Absätzen werden die Vorschriften (engl. procedures) zum Aufbau, Gebrauch und Abbau einer Verbindung zwischen zwei gleichberechtigten DXE's beschrieben.

2.4.1 Vorschriften für die Adressierung

2.4.1.1 Adressinformation

Alle ausgesandten Blocks sollen ein Adressfeld enthalten, wie es in 2.2.13 beschrieben ist. Alle Blocks sollen sowohl die Adresse (Rufzeichen) des Empfängers als auch die Adresse des Absenders enthalten, wobei die Empängeradresse an erster Stelle steht. Dadurch ist es möglich, dass viele Verbindungen gleichzeitig auf demselben HF-Kanal bestehen.

Die Empfängeradresse kann auch der Name einer Gruppe, oder ein Club-Rufzeichen sein, wenn eine Aussendung an mehrere Stationen erlaubt ist (z.B. Rundsprüche oder Bakenmeldungen mit UI-Blocks). Das Arbeiten mit Empfängeradressen, die kein gültiges Amateurfunk-Rufzeichen sind, wir weiter untersucht.

2.4.1.2 Unterscheidung zwischen Kommando und Meldung

In der Version 2.0 des AX.25 Protokolls ist die Unterscheidung zwischen Kommando und Meldung im Adressfeld enthalten. Um mit früheren Versionen kompatibel zu bleiben, wird diese Information in zwei Bits übermittelt. Sie ist im Bit 7 des SSID-Bytes der Empfänger- und der Senderadresse enthalten.

Eine abwärtskompatible AX.25 DXE kann anhand der Information in den Kommando/Meldungs-Bits erkennen, ob sie mit einer Gegenstation verbunden ist, die eine ältere Version des Protokolls verwendet. Falls beide C-Bits auf Null gesetzt sind, verwendet die Gegenstation das ältere Protokoll. In der neueren Version des Protokolls ist jeweils ein Bit auf Null, das andere auf Eins gesetzt, abhängig davon, ob es sich um ein Kommando oder eine Meldung handelt.

Die Kommando/Meldungs-Bits sind kodiert wie in Abb. 8 gezeigt.

Blocktyp	Empf. SSID C-Bit	Abs. SSID C-Bit
frühere Versionen	0	0
Kommando (V. 2.0)	1	0
Meldung (V. 2.0)	0	1
frühere Versionen	1	1

Abb. 8 -- Kommando/Antwort-Kodierung

so dass in der Version 2.0 dieses Protokolls immer eines der C-Bits auf Null, das andere auf Eins gesetzt sein soll.

Durch die Zusätzliche Kommando/Meldung-Information können im AX.25 S-Blocks sowohl Kommando-, als auch Meldungsblocks sein. Dieses hilft bei der Aufrechterhaltung der Kontrolle über die Verbindung während der Phase der Datenübermittlung. (Im Original steht nichts darüber, wie diese Hilfe aussehen soll: Wenn längere Zeit keine I-Blocks ausgestrahlt worden sind, soll nach Ablauf des Zeitgebers T3 nachgefragt werden, ob die Verbindung noch besteht. Dieses geschieht mit einem RR- oder RNR-Block mit gesetztem P-Bit, der als Kommando ausgesandt wird. Die Antwort soll ein RR- oder RNR-Block mit gesetztem F-Bit sein, der als Meldung übertragen wird. Dieses kann nur dann richtig funktionieren, wenn beide Stationen zwischen Kommando und Meldung bei RR- oder RNR-Blocks unterscheiden können. Siehe dazu auch 2.3.5.4.2 und 2.4.4.9.)

2.4.2 Vorschriften für das P/F-Bit

Eine DXE soll auf einen SABM- oder DISC-Kommandoblock, in dem das P-Bit auf Eins gesetzt ist, mit einem UA- oder DM-Meldungsblock antworten, in dem das F-Bit auf Eins gesetzt ist. (Anmerkung: Das P- und das F-Bit stehen im Kontrollfeld an der gleichen Stelle. In einem Kommandoblock wird dieses Bit als P-Bit interpretiert, in einem Meldungsblock als F-Bit.)

Eine DXE soll auf einen I- oder S-Kommandoblock, in dem das P-Bit auf Eins gesetzt ist, mit einem RR-, RNR- oder REJ-Meldungsblock antworten, in dem das F-Bit auf Eins gesetzt ist, wenn der I- oder S-Block von der Gegenstation stammt, mit der die DXE verbunden ist.

Eine DXE soll auf einen I- oder S-Kommandoblock, in dem das P-Bit auf Eins gesetzt ist, mit einem DM-Meldungsblock antworten, in dem das F-Bit auf Eins gesetzt ist, wenn der I- oder S-Block von einer Station stammt, mit der die DXE nicht verbunden ist.

Das P-Bit wird im Zusammenhang mit der Wiederherstellung nach Ablauf des Zeitgebers T3 verwendet (siehe 2.3.5.4).

Wird das P/F-Bit nicht benötigt, so soll es auf Null gesetzt sein.

2.4.3 Vorschriften für den Auf- und Abbau einer Verbindung

2.4.3.1 Aufbau einer LAPB-Verbindung

Wenn eine DXE eine Verbindung mit einer anderen DXE wünscht, so sendet sie einen SABM-Kommandoblock zu dieser Station und startet den Zeitgeber T1. Falls die andere DXE diesen Block empfangen und eine Verbindung aufnehmen kann, wird sie mit einem UA-Meldungsblock antworten und ihre beiden Folgezähler (V(S) und V(R)) auf Null setzen. Der Empfang des UA-Blocks veranlasst die DXE, welche die Verbindung aufbauen wollte, den Zeitgeber T1 zu stoppen und auch ihre Folgezähler auf Null zu setzen.

Falls die andere DXE nicht antwortet bevor der Zeitgeber T1 abläuft, wird die anfordernde DXE den SABM-Block noch mal aussenden und T1 wieder starten. Die DXE soll N2 Versuche machen, durch Aussenden des SABM-Blocks die Verbindung herzustellen. N2 ist in 2.4.7.2 definiert.

Falls eine DXE nach Empfang eines SABM-Blocks entscheidet, dass sie nicht bereit ist eine Verbindung aufzunehmen, soll sie einen DM-Block aussenden (siehe auch 2.3.4.3.5).

Die DXE, die den SABM-Block ausgesandt hat, soll, nachdem sie einen DM-Block empfängt, den Zeitgeber T1 stoppen und nicht weiter versuchen, die Verbindung aufzubauen.

Die die Verbindung anfordernde DXE wird alle Blocks ignorieren und verwerfen, ausser SABM-, DISC-, UA- und DM-Blocks der gerufenen DXE. (Im Original wird hier nicht darauf eingegangen, wie sich eine DXE verhalten soll, die mehrere Verbindungen gleichzeitig aufnehmen kann. Sinnvollerweise sollte das in diesem Absatz Gesagte dann nur für den einen logischen Kanal gelten, über den die andere DXE gerufen wird.)

Andere als UA- oder DM-Blocks als Antwort auf einen SABM-Block werden nur dann ausgesandt, wenn die Verbindung hergestellt ist und keine unbestätigten SABM-Blocks existieren. (Vollkommen unverständlich! Die einzigen im AX.25 Protokoll als Antwort auf einen SABM-Block vorgesehenen Blocks sind UA- oder DM-Blocks. Siehe auch 2.3.4.3.1.)

2.4.3.2 Phase der Datenübermittlung

Nachdem die Verbindung hergestellt ist, wird die DXE den Zustand „Phase der Datenübermittlung“ einnehmen. In diesem Zustand wird die DXE nach der in 2.4.4 beschriebenen Vorschrift I- und S-Blocks annehmen und aussenden.

Falls die DXE während der Phase der Datenübermittlung ein SABM-Kommando von der Gegenstation empfängt, so wird sie die Verbindung nach der in 2.4.6 beschriebenen Vorschrift rücksetzen.

2.4.3.3 Abbau der Verbindung

In der Phase der Datenübermittlung kann jede der DXE's einen Abbruch der Verbindung anfordern, indem sie einen DISC-Block aussendet und den Zeitgeber T1 startet.

Eine DXE soll, nachdem sie ein gültiges DISC-Kommando erhalten hat, einen UA-Block aussenden und in den Zustand „Frei“ übergehen. Sobald die andere DXE als Antwort auf ihr DISC-Kommando einen UA- oder DM-Block empfängt, soll sie den Zeitgeber T1 stoppen und ebenfalls in den Zustand „Frei“ übergehen.

Falls kein UA- oder DM-Block empfangen wird, bevor der Zeitgeber T1 abläuft, soll der DISC-Block wiederholt und der Zeitgeber erneut gestartet werden. Nachdem dieses N2-mal geschehen ist, soll die DXE in den Zustand „Frei“ übergehen.

2.4.3.4 Der Zustand „Frei“

Im Zustand „Frei“ soll eine DXE alle empfangenen Kommandos beobachten und auf ein SABM-Kommando, wie in 2.4.3.1 beschrieben, reagieren. Als Antwort auf ein DISC-Kommando soll sie einen DM-Block aussenden.

Im Zustand „Frei“ kann die DXE den Aufbau einer Verbindung beginnen, wie es in 2.4.3.1 beschrieben ist.

Auf Kommandoblocks mit gesetztem P-Bit ausser SAMB- oder UI-Blocks soll eine DXE mit einem DM-Block antworten, in dem das F-Bit gesetzt ist. Der empfangene Block soll nicht weiter beachtet werden.

Wenn eine DXE nach einem Fehlerzustand oder aufgrund eines internen Fehlers in den Zustand „Frei“ übergeht, so soll sie dieses anzeigen, indem sie einen DM-Meldungsblock aussendet anstatt einen DISC-Block auszusenden und die Abbruchprozedur in 2.4.3.3 durchzuführen. Danach kann die DXE versuchen, die Verbindung, wie in 2.4.3.1 beschrieben, neu aufzubauen.

2.4.3.5 Wiederherstellung nach Kollisionen

2.4.3.5.1 Kollisionen in einer Halbduplex-Umgebung

Kollisionen von Blocks in einer Halbduplex-Umgebung werden durch das Verhalten des Zeitgebers T1 und des Wiederholungszählers aufgefangen. Es sind keine weiteren Aktionen nötig.

2.4.3.5.2 Kollisionen von nicht nummerierten Kommandos

Falls sich zwei SABM- oder DISC-Kommandos überschneiden, sollen beide DXE's zum frühest möglichen Zeitpunkt einen UA-Block aussenden und den angegebenen Zustand einnehmen.

Falls sich ein SABM- mit einem DISC-Kommando überschneidet, sollen beide DXE's den Zustand „Frei“ einnehmen und zum frühest möglichenZeitpunkt einen DM-Block aussenden.

2.4.3.5.3 Kollisionen eines DM-Blocks mit einem SABM- oder DISC-Block

Falls eine nicht angeforderte DM-Meldung ausgesandt wird, kann es zu Überschneidungen dieses Blocks mit einem SABM- oder DISC-Kommando kommen. Um zu vermeiden, dass der DM-Block fehlinterpretiert wird, sollen alle nicht angeforderten DM-Blocks mit auf Null gesetztem F-Bit ausgesandt werden und alle SABM- und DISC-Blocks mit auf Eins gesetztem P-Bit. Die Antwort auf einen solchen SABM- oder DISC-Block kann dann nur ein DM- (oder UA-)Block mit auf Eins gesetztem F-Bit sein, so dass eine Verwechslung ausgeschlossen ist.

2.4.3.6 Betrieb ohne Flusskontrolle

Im Amateurfunk gibt es eine zusätzliche Betriebsart, die mit Level-2-Verbindungen nicht zu realisieren ist, die Gesprächsrunde, bei der mehrere Funkamateure an einer Unterhaltung teilhaben können.

Die Implementation dieser Betriebsart liegt technisch ausserhalb der AX.25 Verbindungen, benutzt aber trotzdem die Blockstruktur des AX.25.

Im AX.25 benutzt man einen speziellen Blocktyp für diese Betriebsart, genannt unnummerierter Informationsblock (UI-Block). In dieser Betriebsart sollte im Subfeld der Empfängeradresse ein Kodewort eingetragen werden, mit dessen Hilfe die an der Runde Beteiligten die Blocks aussortieren können, damit sie nicht alle Blocks mitlesen müssen, die über den gemeinsamen HF-Kanal übertragen werden. Z.B. sollte eine Gesprächsrunde über Packet-Radio als Empfängeradresse das Wort PACKET verwenden. Die meisten DXE's haben eine Möglichkeit, nur Blocks mitzuschreiben, die an eine bestimmte Adresse gerichtet sind (Monitor-only-Kommando). Das Rufzeichen des Absenders ist mit in dem Block enthalten, so dass man erkennen kann, von wem einzelne Beiträge am Gespräch stammen.

Da diese Betriebsart ohne Flusskontrolle arbeitet (die UI-Blocks sind nicht nummeriert), gibt es keine Anforderungen zur Wiederholung von verloren gegangenen oder fehlerhaften Blocks. Kollisionen können ebenso auftreten, mit der Gefahr, dass die sich überschneidenden Blocks verloren gehen.

2.4.4 Vorschriften für die Datenübermittlung

Wenn eine Verbindung, wie oben beschrieben, aufgebaut ist, können beide Stationen I-, S- und U-Blocks akzeptieren.

2.4.4.1 Senden von I-Blocks

Jedes Mal, wenn eine DXE einen I-Block auszusenden hat, wird sie die Sendefolgenummer N(S) des I-Blocks auf den Wert ihres Sendefolgezählers V(S) setzen. Nachdem der I-Blocks ausgesandt ist, erhöht sie den Sendefolgezähler um Eins. Falls der Zeitgeber T1 nicht läuft, soll er gestartet werden. Wenn er läuft, soll er erneut gestartet werden.

Die DXE soll keine weiteren I-Blocks aussenden, wenn der Wert ihres Sendefolgezählers gleich der letzten von der Gegenstation empfangenen N(R) plus Sieben ist. Der nächste auszusendende Block hätte die gleiche Sendefolgenummer, wie der letzte bestätigte Block und könnte mit diesem verwechselt werden.

Falls eine DXE im Zustand „nicht empfangsbereit“ ist, kann sie trotzdem weiter I-Blocks aussenden, solange die Gegenstation empfangsbereit ist.

Im Zustand „Blockrückweisung“ sendet die DXE keine weiteren I-Blocks aus.

2.4.4.2 Empfang von I-Blocks

2.4.4.2.1 DXE empfangsbereit

Wenn eine DXE einen gültigen I-Block empfängt (mit richtiger FCS und einer Sendefolgenummer, die dem Empfangsfolgezähler entspricht) und empfangsbereit ist, wird sie den I-Block akzeptieren, den Empfangsfolgezähler um Eins erhöhen und eine der folgenden Aktionen ausführen:

1. Wenn die DXE selbst einen I-Block auszusenden hat, wird sie die Empfangsfolgenummer N(R) dieses Blocks auf den Wert ihres Empfangsfolgezählers V(R) setzen und den Block aussenden. Damit bestätigt sie den empfangenen I-Block. Alternativ dazu kann sie zuerst einen RR-Block mit N(R) gleich V(R) aussenden und anschliessend den I-Block.
2. Hat die DXE keine I-Blocks auszusenden, so bestätigt sie den Empfang mit einem RR-Block, dessen N(R) auf den Wert von V(R) gesetzt ist. Die DXE kann eine kurze Zeit warten, um sicher zu sein, dass keine zusätzlichen I-Blocks zu übertragen sind.

Der Empfang eines I-Blocks, dessen I-Feld die Länge Null hat, soll dem höheren Layer mitgeteilt werden ohne das I-Feld zu übergeben.

2.4.4.2.2 DXE nicht empfangsbereit

Befindet sich die DXE im Zustand „nicht empfangsbereit“, so kann sie alle empfangenen I-Blocks ignorieren, ausser dass sie mit einem RNR-Block erneut auf ihren Zustand hinweist.

Falls eine DXE von der Gegenstation die Meldung „nicht empfangsbereit“ empfängt, soll sie periodisch bei dieser Gegenstation nachfragen, bis die Gegenstation wieder empfangsbereit ist. Dieses kann mit RR- oder RNR-Blocks geschehen, in denen das P-Bit auf Eins gesetzt ist.

2.4.4.3 Empfang von Blocks mit falscher Sendefolgenummer

Wenn ein I-Block empfangen wird, dessen FCS richtig ist, aber dessen Sendefolgenummer N(S) nicht mit dem Inhalt des Empfangsfolgezählers N(R) des Empfängers übereinstimmt, soll der Block verworfen werden. Ein REJ-Block soll ausgesandt werden, mit einer Empfangsfolgenummer, die um Eins höher ist, als die N(S) des letzten, richtig empfangenen I-Blocks. Bevor der I-Block verworfen wird, soll die DXE die Empfangsfolgenummer und das P-Bit ausgewerten.

2.4.4.4 Empfang von fehlerhaften Blocks

Alle Blocks mit falscher FCS, ungültige Blocks oder Blocks mit falscher Adresse werden ignoriert.

2.4.4.5 Empfangsbestätigung

Immer, wenn ein I- oder S-Block richtig empfangen wurde, selbst wenn die DXE nicht empfangsbereit ist, soll die N(R) des Blocks ausgewertet werden, um zu sehen, ob sie ausgesandte, vorher unbestätigte I-Blocks bestätigt. Ist dies der Fall, so soll der Zeitgeber T1 gestoppt werden. Wurde T1 gestoppt, und sind noch weitere I-Blocks unbestätigt, so soll T1 erneut gestartet werden. Läuft der Zeitgeber T1 ab, bevor eine Bestätigung empfangen wurde, soll die Station die I-Blocks erneut aussenden (siehe 2.4.4.9).

2.4.4.6 Empfang einer Wiederholungsaufforderung

Wenn eine DXE einen REJ-Block empfängt, so setzt sie ihren Sendefolgezähler auf den Wert der Empfangsfolgenummer des REJ-Blocks. Die DXE wird dann alle unbestätigten I-Blocks zum frühest möglichen Zeitpunkt erneut aussenden:

1. Sendet die DXE z.Zt. nicht, und der Kanal ist frei, so kann sie sofort mit der Wiederholung beginnen.
2. Arbeitet die DXE im Vollduplex-Betrieb, und sendet sie gerade einen UI- oder S-Block aus, so kann sie, ohne diesen Block abzubrechen, die zu wiederholenden Blocks anschliessen.
3. Arbeitet die DXE im Vollduplex-Betrieb und empfängt den REJ-Block, während sie einen anderen I-Block an die Gegenstation aussendet, so kann sie den I-Block abbrechen und sofort mit der Wiederholung beginnen.
4. Anschliessend an die Wiederholung kann die DXE, falls weitere I-Blocks zur Übertragung anstehen, diese sofort aussenden, solange die Anzahl insgesamt nicht die maximal mögliche Zahl unbestätigter Blocks übersteigt (siehe 2.4.7.4).

Empfängt die DXE einen REJ-Block mit gesetztem P-Bit, so soll sie, bevor sie die I-Blocks wiederholt, einen RR- oder RNR-Block mit gesetztem F-Bit aussenden.

2.4.4.7 Empfang der Meldung „nicht empfangsbereit“

Empfängt eine DXE einen RNR-Block, so soll sie keine weiteren I-Blocks an die Gegenstation aussenden, bis diese wieder empfangsbereit ist. Falls der Zeitgeber T1 abläuft, nachdem ein RNR-Block empfangen wurde, soll die DXE die in 2.4.4.9 beschriebene Warteprozedur beginnen. Die Gegenstation soll periodisch nach ihrem Status abgefragt werden, bis sie wieder Empfangsbereitschaft meldet (siehe 2.4.4.2.2).

2.4.4.8 Aussendung der Meldung „nicht empfangsbereit“

Wenn eine DXE vorübergehend nicht in der Lage ist, weitere I-Blocks zu empfangen, teilt sie dieses der Gegenstation durch die Aussendung eines RNR-Blocks mit. Eine nicht empfangsbereite DXE kann, falls möglich, S-Blocks empfangen und verarbeiten. Falls sie einen S-Block mit gesetztem P-Bit empfängt, soll sie als Antwort einen RNR-Block mit gesetztem F-Bit aussenden.

Ist die DXE wieder empfangsbereit, so sendet sie der Gegenstation einen RR- oder REJ-Block, je nachdem, ob im letzten empfangenen I-Block die Sendefolgenummer richtig war oder falsch (siehe 2.4.4.2 und 2.4.4.3).

2.4.4.9 Warten auf die Bestätigung

Wenn der Zeitgeber T1 abläuft, während die DXE auf die Bestätigung eines I-Blocks durch die Gegenstation wartet, so startet sie T1 erneut und sendet einen entsprechenden S-Block (RR oder RNR) mit gesetztem

P-Bit aus. Empfängt die DXE einen S-Block mit gesetztem F-Bit, so startet sie den Zeitgeber erneut und setzt ihren V(S) auf den Wert der in diesem S-Block empfangenen N(R). Anschliessend fährt sie fort I-Blocks zu senden oder, falls noch I-Blocks unbestätigt sind, diese zu wiederholen. Empfängt die DXE jedoch einen S-Block ohne gesetztes F-Bit, einen S-Kommandoblock oder einen I-Block, so wird sie T1 weiterlaufen lassen, aber die empfangene N(R) als Bestätigung für I-Blocks bis zur Nummer N(R)-1 benutzen.

Falls T1 abläuft, ohne dass ein S-Block mit gesetztem F-Bit empfangen wurde, so wird die DXE erneut einen S-Block mit gesetztem P-Bit aussenden. Nachdem die DXE N2-mal versucht hat, einen S-Block mit gesetztem F-Bit von der Gegenstation zu erhalten, wird sie die Verbindung nach der in 2.4.6 beschriebenen Vorschrift rücksetzen.

2.4.5 Zustände nach Rückweisung eines Blocks

Eine DXE soll die Prozedur der Blockrückweisung beginnen, wenn sie während der Phase der Datenübermittlung einen Block mit richtiger FCS empfängt, der einer der Bedingungen aus 2.3.4.3.3 genügt.

Die DXE fordert in diesem Fall die andere DXE auf, die Verbindung rückzusetzen, indem sie einen FRMR-Block aussendet (siehe 2.4.6.3).

Nach Aussendung des FRMR-Blocks geht die DXE in den Zustand der Blockrückweisung über. Dieser Zustand wird gelöscht, wenn die DXE ein SABM- oder DISC-Kommando oder eine DM-Meldung empfängt. Alle anderen Kommandos die die DXE während des Zustandes der Blockrückweisung empfängt, bewirken die Aussendung eines weiteren FRMR-Blocks, mit dem gleichen I-Feld wie im ersten FRMR-Block.

Im Zustand der Blockrückweisung sendet die DXE keine weiteren I-Blocks, und empfangene I- und S-Blocks werden nicht weiter ausgewertet.

Die DXE, die den FRMR-Block aussendet, startet den Zeitgeber T1. Falls sie keinen SABM-, DISC- oder DM-Block von der Gegenstation empfängt, bevor T1 abläuft, sendet sie den FRMR-Block erneut aus und startet T1 wieder. Nachdem der FRMR-Block N2-mal ohne Erfolg ausgesandt wurde, führt die DXE die in 2.4.6 beschriebene Rücksetzung der Verbindung aus.

2.4.6 Rücksetzen der Verbindung

Das Rücksetzen der Verbindung dient der Neuinitialisierung der Datenübertragung in beiden Richtungen, nachdem ein nicht behebbarer Fehler aufgetreten ist. Diese Prozedur des Rücksetzens der Verbindung wird nur während der Phase der Datenübermittlung bei einer AX.25 Verbindung durchgeführt.

Eine DXE soll ein Rücksetzen der Verbindung einleiten, wenn sie einen nicht angeforderten Meldungsblock mit gesetztem F-Bit oder einen unerwarteten UA-Block empfängt. Sie kann die Verbindung zurücksetzen, wenn sie einen FRMR-Block empfängt. Alternativ dazu kann sie nach Empfang eines FRMR-Blocks die Verbindung durch Aussenden eines DISC-Blocks abbrechen.

Die DXE setzt die Verbindung zurück, indem sie einen SABM-Block aussendet und den Zeitgeber T1 startet. Die Gegenstation soll, nachdem sie den SABM-Block empfangen hat, zum frühest möglichen Zeitpunkt mit einem UA-Block antworten, ihre Folgezähler V(S) und V(R) auf Null setzen und ihren Zeitgeber T1 stoppen, sofern sie nicht selbst auch einen SABM- oder DISC-Block ausgesandt hat. Wenn der UA-Block bei der ersten DXE richtig empfangen wird, setzt sie ebenfalls ihre Folgezähler V(S) und V(R) auf Null, stoppt T1 und beendet den Zustand „Rücksetzen“. Jeder Zustand „nicht empfangsbereit“ wird ebenfalls gelöscht.

Empfängt die DXE einen DM-Block als Antwort, geht sie in den Zustand „Frei“ über und stoppt T1. Falls T1 abläuft, bevor sie einen UA- oder DM-Block empfängt, sendet sie erneut einen SABM-Block aus und startet T1 wieder. Nachdem T1 N2-mal abgelaufen ist, geht die DXE in den Zustand „Frei“ über.

Andere Kommandos oder Meldungen werden von der DXE ignoriert, bis die Verbindung zurückgesetzt ist.

Eine DXE kann ein Rücksetzen der Verbindung durch die Gegenstation anfordern, indem sie eine DM-Meldung aussendet und anschliessend in den Zustand „Frei“ übergeht.

2.4.7 Liste der Systemparameter

2.4.7.1 Zeitgeber

Um die Integrität einer AX.25 Level 2 Verbindung zu wahren, ist die Verwendung dieser Zeitgeber zu empfehlen.

2.4.7.1.1 Primär-Zeitüberwachung (Acknowledgement Timer) T1

Der Zeitgeber T1 stellt sicher, dass eine DXE nicht endlos auf die Antwort auf Blocks wartet, die sie ausgesandt hat. Die Ablaufzeit dieses Zeitgebers sollte mindestens zweimal so gross sein, wie die Übermittlungsdauer eines Blocks von maximaler Länge und die Rückübermittlung der Bestätigung. Dadurch ergäbe sich auch genügend Zeit für die Auswertung des Blocks bei der Gegenstation.

Falls Level 2 Digipeater in der Verbindung benutzt werden, soll man T1 entsprechend der Anzahl der Digipeater vergrössern.

2.4.7.1.2 Zeitgeber T2 (Response Delay Timer)

Der Zeitgeber T2 kann in einer DXE verwendet werden, um eine Wartezeit anzugeben, zwischen dem Empfang eines I-Blocks und dem Aussenden der Bestätigung. Diese Verzögerung kann die DXE warten, ob noch weitere I-Blocks an sie gerichtet sind, um diese dann gemeinsam (maximal 7) zu bestätigen, anstatt jeden einzelnen I-Block. Die Effektivität des Übertragungskanals kann dadurch gesteigert werden. Im Vollduplex-Betrieb sollten die Bestätigungen nicht um mehr als eine Anzahl k/2 I-Blocks verzögert werden, um einen maximalen Durchsatz zu erhalten. Der Parameter k ist in 2.4.7.4 definiert.

2.4.7.1.3 Zeitgeber T3 (Inactive Link Timer)

Der dritte Zeitgeber T3 wird verwendet, um die Integrität der Verbindung aufrecht zu erhalten, wenn T1 nicht läuft. Es ist zu empfehlen, dass immer, wenn keine I-Blocks oder Blocks mit gesetztem P-Bit unbestätigt sind (während der Phase der Datenübermittlung), der Zustand der Gegenstation alle T3 Zeiteinheiten mit einem RR- oder RNR-Block mit gesetztem P-Bit abgefragt wird. T3 ist stark abhängig von der Qualität der Level-1-Verbindung. T3 sollte grösser sein alsT1 und kann sehr gross sein auf Kanälen mit hoher Integrität.

2.4.7.2 Maximale Anzahl der Wiederholungen (N2)

Die maximale Anzahl der Wiederholungen wird in Verbindung mit dem Zeitgeber T1 verwendet. Falls die Übertragungsqualität auf einem Kanal zu schlecht ist, kann man mit Hilfe von N2 die Verbindung abbrechen.

2.4.7.3 Maximale Anzahl der Bytes in einem I-Feld (N1)

Die maximale Anzahl der Bytes in einem I-Feld ist in AX.25 Version 2 auf 256 festgelegt. Version 1 hatte eine Begrenzung auf 128 Bytes, die Original-Firmware des TAPR-TNC-1 erlaubte 200 Bytes. Im Betrieb auf stark gestörten Kanälen (z.B. Kurzwelle) kann es nötig sein, N1 kleiner zu wählen als vom Protokoll her definiert, um die Wahrscheinlichkeit zu erhöhen, dass ein Block ungestört übertragen wird. Auf jeden Fall ist nur eine ganze Zahl von Bytes (inklusive Null) als Anzahl zulässig.

2.4.7.4 Maximale Anzahl unbestätigter I-Blocks (k)

Die maximale Anzahl unbestätigter I-Blocks zu einem Zeitpunkt ist sieben.

Literaturnachweis:

1. AX.25 Amateur Packet-Radio Link-Layer Protocol Version 2.0 October 1984, Veröffentlichung Nr. 56 der Radio Amateur's Library, veröffentlicht durch die American Radio Relay League, 1984.
2. X.25 Schnittstelle zwischen Datenendeinrichtung (DEE) und Datenübertragungseinrichtung (DÜE) für Endeinrichtungen, die im Paket-Modus in öffentlichen Netzen arbeiten, Genf 1976.

In 1.) angegebene Literatur:

American Telephone and Telegraph Company, „Operations Systems Network Communications Protocol Specification BX.25 - Issue 2.“

ANSI X3.66, „Advanced Data Communication Control Procedure,“ (ADCCP).

CCITT Recommendation X.25, „Interface between Data Terminal Equipment (DTE) and Data-Circuit Terminating Equipment (DCE) for Terminals Operating in the Packet Mode on Public Data Networks.“

ISO 3309, „Data communication -- High-level data link control procedures -- Frame structure.“

ISO 7205, „Reference Model of Open Systems Architecture.“

ISO 7776, „Information Processing Systems - Data Communications - 2nd DP 7776 REVISED - Description of the 1984 x.25 LAPB - Compatible DTE Data Link Procedures.“

Zustandstabellen

Block / Zustand	I mit Poll	I ohne Poll	RR mit Poll	RR ohne Poll	REJ mit Poll	REJ ohne Poll	RNR mit Poll	RNR ohne Poll	SABM	DISC
S1 Frei	DM		DM		DM		DM		UA,S5 3)	DM 4)
S2 Verb.-Aufbau									UA,S5 3)	DM,S1
S3 Blockrückw.	FRMR		FRMR		FRMR		FRMR		UA,S5 3)	UA,S1
S4 Verb.-Abbau	DM,S1		DM,S1		DM,S1		DM,S1		DM,S1	UA,S1
S5 Info.-Übertrag.	RR	I/RR 1)	RR	I/- 2)	RR	I	RR,S9	S9	UA	UA,S1
S6 REJ ausgesandt	RR,S5	I/RR,S5	RR	I/-	RR	I	RR,S15	S15	UA,S5	UA,S1
S7 Warten auf Final	RR	I/RR	RR	I/-	RR	I	RR,S12	S12	UA,S5	UA,S1
S8 nicht bereit	RNR	RNR	RNR	I/-	RNR	I	RNR,S10	S10	UA	UA,S1
S9 Gegenstelle nicht bereit	RR	RR	RR,S5	I/-,S5	RR,S5	I,S5	RR		UA,S5	UA,S1
S10 beide Seiten nicht bereit	RNR	RNR	RNR,S8	I/-,S8	RNR,S8	I,S8	RNR		UA,S8	UA,S1
S11 nicht bereit u. Warten auf Final	RNR	RNR	RNR	I/-	RNR	I	RNR,S13	S13	UA,S8	UA,S1
S12 Gegenstelle nicht bereit u. Warten auf Final	RR	RR	RR,S7	I/-,S7	RR,S7	I,S7	RR		UA,S5	UA,S1
S13 beide Seiten nicht bereit u. Warten auf Final	RNR	RNR	RNR, S11	I/-, S11	RNR, S11	I,S11	RNR		UA,S8	UA,S1
S14 REJ ausgesandt und nicht bereit	RNR	RNR	RNR	I/-	RNR	I	RNR,S16	S16	UA,S8	UA,S1
S15 REJ ausgesandt u. Gegenstelle nicht bereit	RR,S9	RR,S9	RR,S6	I/-,S6	RR,S6	I,S6	RR		UA,S5	UA,S1
S16 REJ ausgesandt u. beide Seiten nicht bereit	RNR	RNR	RNR, S14	I/-, S14	RNR, S14	I,S14	RNR		UA,S5	UA,S1

Falls keine weiteren Informationen vorliegen, soll, evtl. nach Ablauf des Zeitgebers T2, ein RR ausgesandt werden.

Falls keine weiteren Informationen vorliegen, wird nichts ausgesandt.

DM,S1, wenn die Verbindung nicht akzeptiert wird.

UA, wenn der DISC-Block ohne Poll empfangen wurde.

Abb. A1 -- Level 2 Zustandstabelle für empfangene Kommandoblocks

Block / Zustand	RR mit Final	RR ohne Final	REJ mit Final	REJ ohne Final	RNR mit Final	RNR ohne Final	UA	DM	FRMR
S1 Frei									
S2 Verb.-Aufbau							S5	S1	
S3 Blockrückw.									SABM,S2
S4 Verb.-Abbau							S1	S1	
S5 Info.-Übertrag.	I/- 2)	I/- 2)	I	I	S9	S9		SABM,S2	SABM,S2
S6 REJ ausgesandt	I/-	I/-	I	I	S15	S15	SABM,S2	SABM,S2	SABM,S2
S7 Warten auf Final	I/-,S5	I/-	I,S5	I	S9	S12	SABM,S2	SABM,S2	SABM,S2
S8 nicht bereit	I/-	I/-	I	I	S10	S10	SABM,S2	SABM,S2	SABM,S2
S9 Gegenstelle nicht bereit	I/-,S5	I/-,S5	I,S5	I,S5			SABM,S2	SABM,S2	SABM,S2
S10 beide Seiten nicht bereit	I/-,S8	I/-,S8	I,S8	I,S8			SABM,S2	SABM,S2	SABM,S2
S11 nicht bereit u. Warten auf Final	I/-,S8	I/-	I,S8	I	S10	S13	SABM,S2	SABM,S2	SABM,S2
S12 Gegenstelle nicht bereit u. Warten auf Final	I/-,S5	I/-,S7	I,S5	I,S7			SABM,S2	SABM,S2	SABM,S2
S13 beide Seiten nicht bereit u. Warten auf Final	I/-,S8	I/-,S11	I,S8	I,S11			SABM,S2	SABM,S2	SABM,S2
S14 REJ ausgesandt und nicht bereit	I/-	I/-	I	I	S16	S16	SABM,S2	SABM,S2	SABM,S2
S15 REJ ausgesandt u. Gegenstelle nicht bereit	I/-,S6	I/-,S6	I,S6	I,S6			SABM,S2	SABM,S2	SABM,S2
S16 REJ ausgesandt u. beide Seiten nicht bereit	I/-,S14	I/-,S14	I,S14	I,S14			SABM,S2	SABM,S2	SABM,S2

2) Falls keine weiteren Informationen vorliegen wird nichts ausgesandt.

Abb. A2 -- Level 2 Zustandstabelle für empfangene Antwortblocks

Block Zustand	lokales Start Kommando	lokales Stop Kommando	Station nicht bereit	Station wieder bereit	T1 läuft ab	T3 läuft ab	N2 wird über-schritten	falsches N(S) empfangen	falsches N(R) empfangen	unbe-kannter Blocktyp
S1 Frei	SABM,S2									
S2 Verb.-Aufbau		DISC,S4			SABM	SABM	S1			
S3 Blockrückw.	SABM,S2	DISC,S4			FRMR	FRMR	SABM,S2			
S4 Verb.-Abbau	SABM,S2				DISC	DISC	S1			
S5 Info.-Uebertrag.	SABM,S2	DISC,S4	RNR,S8		RRc,S7	RRc,S7		REJ,S6	FRMR,S3	FRMR,S3
S6 REJ ausgesandt	SABM,S2	DISC,S4	RNR,S14		RRc,S7	RRc,S7	SABM,S2		FRMR,S3	FRMR,S3
S7 Warten auf Final	SABM,S2	DISC,S4	RNR,S11		RRc		SABM,S2		FRMR,S3	FRMR,S3
S8 nicht bereit	SABM,S2	DISC,S4		RR,S5	RNRc,S11	RNRc,S11		RNR	FRMR,S3	FRMR,S3
S9 Gegenstelle nicht bereit	SABM,S2	DISC,S4	RNR,S10		RRc,S12	RRc,S12		REJ,S15	FRMR,S3	FRMR,S3
S10 beide Seiten nicht bereit	SABM,S2	DISC,S4		RR,S9	RNRc,S13	RNRc,S13		RNR	FRMR,S3	FRMR,S3
S11 nicht bereit u. Warten auf Final	SABM,S2	DISC,S4		RR,S7	RNRc		SABM,S2		FRMR,S3	FRMR,S3
S12 Gegenstelle nicht bereit u. Warten auf Final	SABM,S2	DISC,S4	RNR,S13		RRc		SABM,S2		FRMR,S3	FRMR,S3
S13 beide Seiten nicht bereit u. Warten auf Final	SABM,S2	DISC,S4		RR,S12	RNRc		SABM,S2		FRMR,S3	FRMR,S3
S14 REJ ausgesandt und nicht bereit	SABM,S2	DISC,S4		RR,S6	RNRc,S11	RNRc,S11		RNR	FRMR,S3	FRMR,S3
S15 REJ ausgesandt u. Gegenstelle nicht bereit	SABM,S2	DISC,S4	RNR,S16		RRc,S12	RRc,S12			FRMR,S3	FRMR,S3
S16 REJ ausgesandt u. beide Seiten nicht bereit	SABM,S2	DISC,S4		RR,S15	RNRc,S11	RNRc,S11			FRMR,S3	FRMR,S3

S-Blocks die XXXc gekennzeichnet sind, werden als Kommando, alle anderen als Antwort ausgesandt.

VERSIONSHISTORY:

1.78nk01

- L3 neu sortiert - es entfallen l3a.c, l3b.c l3c.c; iproute.c heisst nun l3ip.c. Neu: l3inp.c, l3misc.c, l3nbr.c, l3netrom.c, l3rtt.c, l3tab.c, l3var.c, l3vc.c, l7showl3.c. l3.h entfällt, stattdessen gibt es nun l3local.h (nur für L3) und l3global.h (für alle Layer). Verbesserungsvorschläge zur Aufteilung bzw. zu Filenamen sind willkommen!
- L3: Bei der automatischen Nachbarerkennung werden beim umschalten von NETROM auf INP alle bisher als UI gesendeten Ziele erneut gemeldet.
- Node-Call in Kleinschrift anzeigen, wenn INP-Options vorhanden.
- GO32-Include-Files include/{hardware.h,kiss.h,pc.h,vanessa.h,api.h,api32.h} nach os/go32 verschoben.
- Linux-Version: bei STAT wird "Free RAM (Fmem)" nicht mehr angezeigt.
- neues Kernel-Interface fuer Linux-Version (siehe history/178mh01.his und history/kernelif.his). (DG9OBU)
- neuer Befehl AXIPR (siehe history/axipr.his) für Einstellungen, die bisher nur über ax25ip.cfg (incl. Neustart) vorgenommen werden konnten. (DG9OBU)
- Auto-Update der Konfigurationsdateien TNNxxx.PAS und TNNxxx.TNB.

1.78nk02

- DG9OBU: Kernel-Interface nun für 2.2er und 2.4er Kernel
- DG9OBU: IPR-Automatik
- Bei DAMA zusätzliche Sendepause alle ca. 30s
- L4:
  - Zielknoten des Circuits wird als Call gespeichert, nicht mehr als Eintrag der Nodesliste
  - itol3 nun ohne Priorität (wurde sowieso nicht tatsächlich verwendet)
- Hostmode: Anzeige bei "ESC Y" korrigiert - der eigentliche Fehler, dass numhsts manchmal <0 wird, ist damit nicht beseitigt!

1.78mh01

- Neues Feature: direktes IP-Interface zu Linux-Kernel, in include/all.h per #define aktivierbar, defaultmässig deaktiviert
- Befehl "KERNELIF", kurz "KERN"
- KERN INIT: Initialisiert das Interface, es wird geprüft ob der Kernel die benötigten Funktionen bereitstellt. Unterstützt werden die Kernel 2.0.x, 2.2.x und 2.4.x. Bei 2.0.x und 2.2.x muss die benötigte Funktionalität über ein Zusatzpaket hinzugefügt werden, bei 2.4.x ist nur eine Option beim Kernelcompilieren entsprechend zu setzen.
  - Kernel 2.0.x/2.2.x: es sind zusätzliche Programmpakete erforderlich.
  - Kernel 2.4.x : bei "Network device support" der Unterpunkt "Universal TUN/TAP device driver support", möglichst fest in den Kernel bauen, als Modul bisher nicht getestet.
    Das Interface initialisiert sich NICHT von selbst, dies muss explizit mit INIT erfolgen!!! Es ist sonst nicht nutzbar und zeigt sich ziemlich bockig ;) Es erfolgt ein Warnhinweis, falls zu diesem Zeitpunkt mit IPA noch keine Node-IP festgelegt wurde. Die Interfacestatistik wird gelöscht.
- KERN STATUS: Zeigt den momentanen Status und die Statistik des Interfaces an.
- KERN CLEAR: Löscht die Statistik, dies passiert auch bei "CLEAR" für die Gesamtstatistik.
- KERN SETKIP: Setzt die IP des Linuxkernels. Hier kann entweder eine IP-Adresse oder ein Hostname angegeben werden, letzterer wird entsprechend aufgelöst *wenn* ein Nameserver verfügbar ist. Die Angabe einer IP-Nummer ist deshalb zu bevorzugen!
  Bsp: "KERN SETKIP 44.130.13.110" oder
  "KERN SETKIP db0uhi.ampr.org"
  Dies muss passieren, bevor das Interface geUPt wird, ist dies nicht geschehen kann das Interface nicht benutzt werden!!! Die IP von tnn wird wie üblich mit dem IPA-Kommando gesetzt, dies muss ebenfalls vor dem UPpen passiert sein.
- KERN UP: Aktiviert das konfigurierte Interface. Es wird im Kernel ein Interface mit dem Namen "tnn" erzeugt (mit ifconfig prüfbar) und eine Route für tnn eingetragen (route). Dies passiert alles automatisch auf der Basis der gesetzten IPs. Im tnn-IP-Router wird ebenfalls ein passender Routeneintrag erzeugt.
  Das Interface lässt sich nur UPpen wenn folgendes vorher passiert ist: IPA, KERN INIT, KERN SETKIP <ip/hostname>.
- KERN DOWN: Deaktiviert das Interface zum Kernel. Das eingetragene Interface und die automatisch im Kernel und tnn eingetragenen IP-Routen werden gelöscht, sonstige Routen im tnn-Router die auf das Interface zeigen, bleiben eingetragen, sie funktionieren aber natürlich nicht mehr. Daten, die bei deDOWNtem Interface durch noch

bestehende Routeneinträge auf ein solches geroutet werden, werden in den Biteimer befördert.

- allgemeines: etliche Kommandos sind nur nach vorheriger, erfolgreicher Ausführung anderer Kommandos möglich oder hängen vom derzeitigen Zustand des Interfaces ab. Wenn man etwas machen will was derzeit nicht geht, erhält man (mehr oder weniger) ausführliche Meckermeldungen. Die Konfiguration eines Interfaces kann nur geändert werden wenn es DOWN ist. Ausnahme:
  IPA-Änderungen, aber die werden bei aktivem Interface (noch) nicht an den Kernel durchgereicht. Das Resultat ist denkbar einfach: das Interface geht dann nicht mehr.
  Ablaufbeispiel:
  IPA 44.130.13.100
  KERN INIT
  KERN SETKIP 44.130.13.102 bzw. KERN SETKIP db0uhi.ampr.org
  (KERN STATUS zum gucken)
  KERN UP
  ...
  (eventuell zusätzliche Routeneinträge mit IPR)
  ...
  KERN DOWN
- STAT zeigt auch die Statistik des Kernelinterfaces mit an, diese kann auch direkt mit "STAT K" abgefragt werden wenn eincompiliert.
- Versionsbefehl zeigt Feature und Kernel als Interface an wenn eincompiliert

1.78mh03

- IPA-Befehl modifiziert, Angabe der Subnetz-Bits jetzt möglich. Es findet KEINE UEBERPRUEFUNG statt ob das Subnetz zur IP passt, also selber ausrechnen! (soll später mal anders werden) Bsp: IPA 44.130.13.100/32
- Meldung der eigenen IP und Subnetz-Bits per INP an andere Nodes
- bei "SP" (SaveParam) Speicherung der eigene IP jetzt auch mit den Subnetz-Bits

1.78mh04

- Änderung bei den Subnetz-Bits bei IPA: sind keine angegeben so melden wir uns mit 32 weiter, das sorgt dafür, dass wir nur die eigene IP an uns ziehen und nicht alles.
- per INP empfangene IP-Adressen und Subnetz-Bits anderer Digis werden in entsprechende IPR- und ARP-Einträge umgesetzt.
  anlegen/ändern über src/l3inp.c::update_options(..)
  löschen über src/l3tab.c::add_route(..., qual=0) ->del_route(...))
- haben wir keine eigene IP (IPA 0.0.0.0) melden wir auch keine Subnetz-Bits
- ARP- und IPR-Eintraege werden nur gemacht wenn wir eine eigene IP gesetzt haben und Parameter 12 gesetzt (!= 0) ist (siehe unten)

= neuer Par 12 ("AutoIPR")
  0 : ausgeschaltet, KEINE automatischen IPR- und ARP-Ein/Austragungen
  (damit kann man die Tabellen einfrieren wenn sie gefüllt sind!)
  1 : automatische Ein/Austragung OHNE jegliche Prüfung der IP-Adressen
  2 : automatische Ein/Austragung, unmögliche IP-Adressen werden ignoriert
  (x.x.x.0 x.x.x.255)
  3 : automatische Ein/Austragung, es werden nur IP-Adressen berueckichtigt,
  die im GLEICHEN Netz wie man selbst und gültig sind. (default)
  (Bsp: hat man die IP 44.1.2.3 werden nur andere 44.x.x.x-IPs eingetragen)

Die Prüfung mit Stufe 3 ist *dringend* zu empfehlen, wer sich mal genau anguckt was da z.T. für schrottige Adressen von einigen Knoten verbreitet werden, weiss warum. Möchte man alles offen haben ist Stufe 1 zu wählen, man sollte sich dann aber nicht wundern, wenn z.B. das lokale Ethernet plötzlich in die Botanik geroutet wird!

Auf die Befehle IPR und ARP haben diese Einstellungen keine Wirkung, von Hand sind jederzeit weitere (auch unsinnige) Eintragungen möglich.

= IPA: IP-Adressen die auf .0 oder .255 enden werden nicht akzeptiert.
(.0: Netzwerkbasisadresse, .255: Broadcastadresse, beide sind lt. IP-Spec nicht als Hostadressen zugelassen)

1.78mh05

- maximale Anzahl der möglichen Connects von Usern zum Digi auf *einem Port* jeweils für jeden Port separat einstellbar. Aktivierung: in include/all.h das "#define USERMAXCON" entkommentieren
  Anzahl der maximalen Connects einstellbar mit dem PORT-Befehl:
  "PORT <Portnummer> MAXCON <Anzahl>", die Anzeige des eingestellten Wertes erfolgt mit dem Kommando "PORT *" !!!
  Eine Anzahl von 0 deaktiviert die Funktion. (Voreinstellung)
  Diese Einstellung gilt für *alle* User auf dem angegebenen Port, einzelne User müssen weiterhin mit SUSPEND beschränkt werden. Es werden nur neue Connects abgelehnt wenn die maximale Anzahl erreicht ist, wird die Anzahl auf einen Wert niedriger als die aktuelle Anzahl an Connects eines Users eingestellt, so bleiben diese Verbindungen bestehen. Neue Verbindungen die über die Maximalzahl hinausgehen werden von Knoten mit BUSY abgelehnt.

  !!! ACHTUNG, WICHTIG !!! !!! ACHTUNG, WICHTIG !!! !!! ACHTUNG, WICHTIG !!!
  "SP" schreibt auch MAXCON in parms.tnb wenn die Funktion eincompiliert ist, eine TNN-Version OHNE diese Funktionalität kann dann die generierten PORT-Zeilen NICHT (!!!) verstehen und IGNORIERT SIE KOMPLETT!!! (d.h. alle Porteinstellungen für diesen Port bleiben WIRKUNGSLOS und der Port deshalb ggf. auch falsch konfiguriert und funktioniert nicht!!!

  Abhilfe: jede Porteinstellung (oder nur die absolut wichtigen) in separate Zeilen schreiben. (oder mal den PORT-Befehl überarbeiten...)
  Nachtrag: PORT wurde irgendwann verändert, beschriebenes Verhalten tritt bei unbekannten Befehlen nicht mehr auf.
- AutoIPR setzte zwar die Subnetze, aber nicht die Gateways. Korrigiert.
- das Kernelinterface funktionierte wegen Änderungen am Kernel mit Kernel 2.4.6 nicht mehr. Jetzt geht es wieder, bei Problemen mit Kernel 2.2.x und 2.4.x-Kerneln VOR 2.4.6 die Kommentare in os/linux/kernelif.h beachten!!!
- bei "SP" wurden beim Dump der IPR- und ARP-Eintraege Eintraege, die auf den Kernel-Port zeigten, nicht korrekt geschrieben. Jetzt ok.
- bei "SP" wird auch die Konfiguration des Kernel-Interfaces geschrieben WENN das Interface konfiguriert UND zum diesem Zeitpunkt aktiv ist. Damit soll sichergestellt werden, dass nur gültige und funktionierende Konfigurationen geschrieben werden.
- bei "SP" wird auch die Konfiguration der AX25IP-Routen geschrieben
- neuer Sysop-Befehl "ALIAS" zum Einrichten von Kommando-Aliassen.
  Aktivierung: in include/all.h das "#define ALIASCMD" entkommentieren, die maximale Länge von Aliassen und den zugeordneten Kommandostrings kann in include/typedef.h (ganz am Ende) geändert werden, in der Standardeinstellung können Aliasse acht Zeichen und die Kommandos 16 Zeichen lang sein.
  !!! Das Alias und das Kommando werden in Grossbuchstaben konvertiert!!!
  Liste der definierten Aliasse abrufen: "alias".
  Alias hinzufügen: "alias funkruf c db0xyz-12" legt das Kommandoalias "FUNKRUF" an und verknüpft es mit dem Kommando "C DB0XYZ-12".
  Alias löschen: "alias funkruf" löscht das zuvor definierte Alias "FUNKRUF", falls solch ein Alias nicht existiert gibts ne Meckermeldung.
- INP: halbstündlich einmal die ganze Nodesliste an den Nachbarn melden (l3misc.c::brosrv10(...))
- INP: versteckte Nodes werden ignoriert wenn sie empfangen werden (dafür dass die erst gar nicht gemeldet werden ist eigentlich der Ursprungsnode verantwortlich, aber einige andere Nodesofts kriegen das anscheinend nicht gebacken)
- LINK-Befehl überarbeitet, durch Zufall eine nicht erkannte Abbruchbedingung bei der Parameterauswertung gefunden (er suchte wild in der Gegend rum wenn die Zeile nicht ganz gefüllt war, sprich nicht alle erwarteten Parameter vorhanden waren)

1.78mh06

- Linux: serielle Ports (KISS, RKISS, TOKENRING usw.) bis 460300 Baud
- Linux: Interfaces des Kernel-AX.25 können als Ports verwendet werden
  Aktivierung: #define KERNELIF (wie für das IP-Interface) muss aktiviert sein
  Einstellen: in tnn.ini: als "device" den Interfacenamen (ax0, ax1 usw.) angeben, "kisstype" auf den Wert 10 stellen
  Beispiel:
  device ax0
  kisstype 10
  port 1
  Das Interface muss natürlich laufen wenn man sich daran binden will, also das Kernel-AX.25 vor TNN starten, die Portparameter von TNN werden an das Interface durchgereicht und falls von Kernel-Seite (z.b. durch kissparms) Portparameter verändert werden, so werden diese von TNN erkannt, durch AutoParameter *alle* Parameter des Ports

neu berechnet und an das Interface zurückübertragen. Hierdurch kann es natürlich vorkommen, dass die vorher gesetzten Parameter wieder verändert werden.
Treten während des Betriebs Probleme mit einem Kernelinterface auf, so wird der betreffende Port ausgetragen ("PORT x OFF"). Ein Wiedereinschalten (mit den alten Einstellungen) ist ggf. mit "PORT x ON" möglich.
DAMA-Master ist ungetestet, er sollte *NICHT* richtig funktionieren weil wir nicht wissen wann das Interface das Frame wirklich fertig gesendet hat!!! Aktivieren lässt er sich aber trotzdem, was dabei rauskommt ist Glückssache, der Kernel *kann* nicht melden wenn er fertig mit der Aussendung ist. (ist nicht vorgesehen im Kernel-AX.25)
DAMA-Slave lässt sich wie gewohnt mit "DAMA S" über den Port-Befehl aktivieren und sollte funktionieren.

- Linux: Watchdog beendet sich auch bei Fehlern auf der Pipe, der Hauptprozess wird schneller gekillt.
- MHeard korrigiert, Fehler wenn User auf mehr als einem Port gehört wurde (DG8BR)
- in der LINK-Struktur Variable "brotim" (Broadcast-Timer) von BOOLEAN (???) auf UWORD geändert
- kosmetische Korrekturen beim VER-Befehl. Inzwischen waren dort so viele anzeigbare Optionen möglich dass die Ausgabe ziemlich chaotisch aussah. Hinweis auf nordlink.org hinzugefügt.

1.78mh07

- INP: voller Nodes-Broadcast nur noch einmal pro Stunde, Änderungen wegen Broadcast an l3netrom.c::inform_peer() wieder rückgängig gemacht, bessere Methode für INP nun direkt in l3misc.c::brosrv10() implementiert.
  Es werden jetzt *alle* Nodes *unabhängig* vom Horizont und den Filtern bei der normalen Meldung gesendet!!! Dies soll sicherstellen, dass auch Nodes, die bisher immer durch den Filter gefallen sind, geupdated werden.
- INP: nach erster RTT-Messung sofort per zweiter Bake dem Nachbarn die RTT mitteilen. Dies macht den Link sofort benutzbar und nicht wie bisher erst nach der zweiten Bake. (nur bei Nachbarn die L3RTT unterstützen, nicht bei Nachbarn die per BROAD arbeiten, funktioniert also auch bei N/N+ Links)
- INP: wenn wir einen N/N+ Link aufbauen wollen senden wir die INP-Kennung nicht mehr, empfangen wir INP-Faehigkeit des Nachbarn und wollen aber N/N+, so ignorieren wir sie. Wollen wir I, der Nachbar kann(/will) das aber nicht, so bauen wir N+ auf.
  ACHTUNG: INP-Links müssen jetzt also mit I auf *beiden* Seiten eingestellt werden sonst gibts nur N+ !!!!!!!!!!
- INP: Filter geaendert, Knoten die mit "#temp" als Alias gemeldet werden, werden übernommen und der Alias gelöscht. (XNet-Fehler) Andere geheime Aliasse werden weiterhin ignoriert.
- 7conn.c::isheard() durchsucht die die MHeard-Liste nun umgekehrt. (DG8BR)
- Änderungen am Flexnet-Modul und am GO32-Watchdog von DG8BR (siehe flex178mh06.his)
- Linux: pty-Schnittstellen bei einer Baudrate von 0 auf nichtblockierenden IO-Modus stellen (NICHT auf echte ser. Schnittstellen anwenden!)
- bei wenigen freien Buffern (100) kontrollierter Ausstieg, so wenig Buffer kommen nicht vor, nur bei "Bufferfressern". Damit der Digi dann nicht stehen bleibt sondern wiederkommt wird hier kontrolliert abgebrochen (Exitcode -2). Ggf. werden die Parameter (save_parms()) nicht mehr gesichert wenn zu dem Zeitpunkt nicht wieder genug Buffer zur Verfügung stehen.

1.78mh08

- INP: Soll ein INP-Link aufgebaut werden, so wird die NODES-Bake nicht mehr zusammen mit dem SABM während des Verbindungsaufbaus gesendet.
- INP: Check der empfangenen Subnetzbits eines Nodes, ist deren Anzahl kleiner oder gleich null oder grösser als 32, so werden die IP-Daten des Nodes gelöscht, der Node aber trotzdem übernommen.
- INP: wird eine Abmeldung fuer einen Node empfangen, dann wird die Lifetime dieses Knotens auf 0 gesetzt. Dies hilft anscheinend gegen Nodes die mit einer Laufzeit von 0 in der Liste stehen und nicht ausgetragen werden
- Linux: Kernelinterfaceupdate wegen Fehlern in den Kerneln nach 2.4.6.
  Es werden bei Fehlern jetzt ausführlichere Meldungen geschrieben, für den Kernelfehler gibt es ein Workaround: Bei fehlerhaften 2.4.x-Kerneln heisst das Interface jetzt nicht mehr 'tnn' sondern auch 'tunX'. (siehe Beschreibung für 2.2.x-Style).
  Es erfolgt eine Info über den Namenswechsel beim UPpen des Kernelinterfaces.
- pacsat.c und pacserv.c überarbeitet, bessere Beschreibung und Beseitigung des Fehlers der Pacsat unter Linux gelegentlich abstürzen liess (DH6BB)
- Fehler behoben, dass Pacsat eine nicht vorhandene Datei löschen wollte (DH6BB)
- Connect im FlexNet-Stil (c -3 etc.) möglich
- in der Statistik werden die auf Links gesendeten UI-Frames erfasst, da bisher hierzu die Monitorfunktion benutzt wurde, konnte nicht festgestellt werden an welchen Nachbarn das Frame eigentlich gesendet worden war da das Ziel NODES ist.
- Linux: externe Texte deren Filenamen exakt acht Zeichen lang waren konnten nicht verwendet werden (DH6BB)
- bei AutoIPR wurden in Mode 2 und 3 keine Einträge durch del_node() geloescht. Korrigiert.

1.78mh09

- INP: die stündliche Nodes-Meldung an den Nachbarn konnte Nodes aus der Tabelle entfernen die noch abgemeldet werden sollten. Jetzt werden nur noch Nodes angefasst die eine Laufzeit haben.
- ALIAS-Kommando komplett neu implementiert, die alte Variante schien Memory-Leaks zu produzieren, neue Variante arbeitet jetzt mit der Listenverwaltung ähnlich wie MHeard.
- Kommando-Aliasse werden bei SP in die Parameterdatei geschrieben
- NETROM: Filter ergänzt der verhindert, dass Nodes mit geheimen Alias beim RX des alten NETROM Nodes-Broadcasts eingetragen werden (nur altes NETROM, UI-Broadcasts) (DG8BR)
- FlexNet: Maxtime-Meldung wieder ausgebaut, lässt manchmal den Link stehen bleiben (DG8BR)
- L2: pollt der Digi einen User und erreicht die maximale Anzahl an Polls (N2-Zaehler) weil keine Antwort erhalten wurde, dann bekommt der User ein DISC. Bisher erfolgte keine Reaktion ausser dem stillen "vergessen" der Verbindung.
- Callcheck beim Connect-Befehl nicht machen wenn der User Sysop ist
- Linux: Kernel-AX.25 Interfaces konnten nicht immer geöffnet werden wegen einem Fehler in der Fehlerbehandlung. Es werden jetzt detaillierte Fehlermeldungen geschrieben wenn Operationen fehlschlagen.
- Linux: Funktion xtempnam() neu implementiert wegen Meldung des Compilers dass die bisher verwendete Systemfunktion tempnam() nicht sicher sei. Jetzt wird mkstemp() verwendet.
- Problem mit nicht ausgetragenen Nodes war wieder aufgetreten, Erkennung ob ein Peer-Eintrag benutzt ist oder nicht in drop_unreachable_nodes() geaendert (pp->used anstelle von pp->routes)

1.78mh10

- Linux: AX25IP und AXIPX aufgeräumt und von eigenen select()-Funktionen befreit bei {ax25ip,axipx}_recv(). Die Prüfung auf lesbaren Descriptor geschieht nun mit in linux.c::update_timer(), dort wird nun für alle Interfacetypen die einen Filedescriptor benutzen (AX25IP, AXIPX, Kernel-AX.25, Kernel-IP) auf die jeweiligen RX-Funktionen verzweigt. Diese Massnahme sollte bei Verwendung der beiden Modi eine geringere Auslastung bei höherer Reaktionsschnelligkeit erbringen. ACHTUNG: AXIPX nicht getestet!!!
- viele Änderungen und Fixes von Odo, DL1XAO (siehe 178or*.his)
- Soll ein INP-Link aufgebaut werden UND die RTT-Messung UND der Fähigkeitsabgleich läuft noch, dann senden wir keine Node-Infos im NETROM-Format an den Nachbarn falls die RTT-Bake nicht innerhalb von ein paar Sekunden wieder zurückgekommen ist (l3misc.c::brosrv10())
- L2: Änderungen an der Statemachine wieder teilweise rückgängig gemacht da mit DAMA nicht 100%ig verträglich, bei nicht-DAMA funktionsfähig.
- Linux: Kernel-AX.25 mit dem Kernel-Stack von DG1KJD kompatibel gemacht.
  Es erfolgt eine automatische Erkennung, welcher Stack vorliegt, allerdings erfolgt eine Umschaltung in den KJD-Mode erst mit dem ERSTEN EMPFANGENEN Frame!!! Frames, die auf einem KJD-Stack VORHER gesendet werden, werden vom Stack als fehlerhaft erkannt und NICHT GESENDET!!! Ein Setzen der L2-Parameter wie TX-Delay etc. ist mit dem KJD-Stack zwar möglich, wird aber noch nicht unterstützt (mangels Testmöglichkeit). Parameteränderungen (TX-Delay, TX-Tail etc.) müssen mit den für das jeweilige Interface zur Verfügung stehenden Tools erledigt werden!!!
- L4: Erweiterung des L4, es können Pakete jeglicher PID übertragen werden. Schaltbar per #define NEW_L4 in all.h. Funktioniert nur, wenn Quell- und Zielknoten diese Modifikation unterstützen, d.h. nur zwischen zwei TNN's!!! Defaultmässig NICHT mit eincompiliert, sollte nur zu Testzwecken eingeschaltet werden!!!
- ARP- und IPR-Einträge werden (intern) markiert, ob sie von einer Automatik (INP-Routenlerner) oder von Hand gemacht wurden. Über Automatik gelernte Routen können von der Automatik UND von Hand (ARP/IPR-Befehl) ausgetragen werden, von Hand gemachte Einträge NUR von Hand. Dies verhindert, dass die Automatik Einträge austrägt, die schon vorher existent waren und unabhängig von den INP-Infos immer vorhanden sein sollen. So sind statische Routen möglich die nicht verloren gehen, auch wenn die INP-Infos zu diesem Ziel entfernt werden.
  !!! aus .tnb-Files geladene Einträge gelten als VON HAND gemacht!!!
- l3ip.c: Cache für IP-Routinginformationen implementiert, war zwar in den Kommentaren zum Code erwähnt, aber nirgends implementiert. Steht nun in rt_find(). Zu aktivieren in all.h, #define IPRTCACHE.
- L2 nun mit Extended-AX.25. Eincompilieren mit #define EAX25 in all.h.
  Ob eine Gegenstation EAX.25 kann wird in MHeard vermerkt, aber nicht gespeichert! Über die zusätzlichen Parameter EAXMAXF und EAXMODE beim Port-Befehl lässt sich das Maxframe und das EAX.25-Verhalten auf dem Port steuern. Hierbei gilt für EAXMODE folgendes:
  0 nur AX.25 zugelassen, EAX.25-Verbindungen werden abgelehnt.
  1 Mode nach Fähigkeiten der Gegenstation bzw. nach MHeard. (AX.25 und EAX.25)
  2 nur EAX.25 zugelassen, AX.25-Verbindungen werden abgelehnt.
  Bei Mode 1 erfolgt ein Fallback auf AX.25 wenn ein EAX.25-Connect zur Gegenstation mit DM beantwortet wurde.

Mode 1 ist standardmässig eingestellt.
EAXMAXF kann max. 127 sein, sollte jedoch 32 nicht überschreiten, default ist 16. Bisher ist ein Maxframe grösser 7 noch nicht mit allen Hardwareinterfaces getestet!!! Hier muss experimentiert werden. (auf VANESSA funktionierts einwandfrei).
Die zusaetzlichen Parameter bei Port sind mit "P *" abrufbar. Parameter 1 (NoAckBuf) sollte bei Verwendung von EAX.25 entsprechend vergrössert werden, hier sollte mindestens 1,5 * grösstes verwendetes EAX.25-Maxframe eingestellt werden, damit immer genug Daten da sind.
Die Maxframe-Automatik funktioniert analog wie bei AX.25 aber nur dann, wenn sie bei AX.25 aktiviert worden ist! Also dort beim Maxframe ggf. das "a" machen (PO x MAXF=?a). Änderung in Zukunft möglich.

- Tokenring: mit neueren Rechnern, die mehrere tausend Runden pro Sekunde schaffen, konnte der Tokenring auf ein frühzeitiges "Timeout" laufen und einen Tokenverlust annehmen weil zu viele Runden vergangen, das Token aber noch nicht wieder da war, da ein von den Runden abhängiger Zähler benutzt wurde. Jetzt wird ein echter Timer und ein Token-Timeout von zwei Sekunden verwendet, Timeout einstellbar ueber TOKENTIMEOUT in all.h.
  Dieser Fehler betraf die DOS und die Linux-Version!!!
- Auf Ports, die als Interlinks arbeiten, Persistence anders einstellen (l2misc.c::autopar() und l3misc::islinkport())

1.79pre1

- MH zeigt bei gehörten Stationen die EAX.25 verwenden dies hinter dem Rufzeichen an
- kleine Änderungen von Bernd DG8BR an l3vc.c.
- AX25IP-Routen konnten doppelt eingetragen werden, jetzt werden Routen für ein Call das schon in der Liste steht nicht mehr angenommen, der Eintrag muss zuerst gelöscht werden. Es erfolgt eine Meckermeldung.

1.78or01

- fdfl und co sind weg, ersetzt durch etwas flexibleres, kein fdefblk mehr, sondern fname, was man auch noch ausbauen könnte, um die Begrenzung bei edit oder uploads auf ein File zu umgehen.
- Der Convers ist ausschaltbar, er wird zwar nur durch Leeroutinen ersetzt, aber zum Platz sparen reicht das.
- NOPORTINMON baut die Monitorausgabe etwas um, damit bestimmte Hostmodeprg etwas glücklicher sind, leider beeinflusst dies auch die Ausgabe beim Monitortrace.
- SETTAILTIME schaltet das Setzen der Tailtime ein, ist für TNC3 recht nützlich. Langfristig würde ich es begrüßen, wenn in der Expert-version alle Pars des L1 setzbar sind, sie können ja gerne automatisch vorberechnet werden, aber wenn man was dran drehen will, kann man es dann wenigstens.
- Einige Funktionen, die ich in Assembler progr. hab, sind ausgedef't

1.78or02

Änderungen in or02, ausgehend von mh09:

- Beenden der TNN bei < 100 Buffer in buffers.c nicht übernommen.
- Variablen in l2rx.c für MaxCon lokalisiert und abgemagert.
- Erweiterungen beim BC in eigne Funktion ausgelagert.
- In l3nbr.c eine Variable eingespart.
- ALIASCMD überarbeitet, Ersatz findet zentraler in ccpcmd() statt, wird mit Listen realisiert und erlaubt es Parameter hinter den Aliassen zu übernehmen, nur der Alias selbst wird in Großschreibung konvertiert.
- bezugnehmend auf mh05 History, ccpport überarbeitet, so dass auch nichtaktivierte Sachen "gesetzt" werden können, sie werden halt einfach überlesen.
- Da mir die po + Zeile zu breit wurde, hab ich die Überschrift 2-zeilig gemacht
- In l7ip.c ist mal wieder unser Problem mit der Bitmaske zu sehen, wir sollten wirklich mal über eine sinnvolle Prüfung diskutieren. (siehe vorherige History, unten) und wer bits == 1 liefert kann doch auch nicht die Wahrheit sagen, hier könnte man durchaus ein minimum von 2..4 bits fordern, max 32 (keine 31!) und alles was außerhalb ist wird ignoriert und nicht umgeändert (gilt als Fehler) (bzw. 8 minimum, da ampr.net 44.x.x.x/8 ist)
- In l7showl3.c; dump-raw-node-info hab ich nicht übernommen (war als Test deklariert).
- dump_port etwas angepasst für meinen Kram.
- In search_file soll laut mh08-History ein Fehler sein, den muss mir aber erst einer beweisen, da ich ihn bei Nachprüfung nicht finden kann, bzw. der Fehler muss wenn, an anderer Stelle sein.
- In mh_port_lookup war ein if zuviel.

1.78or03

- Im Convers die Callprüfung korrigiert und die Benutzung von clilin in cvs_cmds.c ausgebaut.
- valcal() Rückgabewert korrigiert.

- Bei ccp_call() das if vorsorglich geändert.

Änderungen in or03, ausgehend von mh10:

- nur Übernahme in l3misc.c

1.79pre2

- EAX.25: Keine Auswertung des EAX-Flags im Frame bei eingehenden Connects mehr, die Umschaltung erfolgt nun nur noch bei Empfang von SABME.
- EAX.25: Neues Monitorformat bei I-Frames
- Fixes von DL1XAO an MHeard, Port und L7 eingearbeitet
- Externe Programme dem Paket hinzugefügt, Copyrights aktualisiert
- Änderungen am externen Programm pfhadd wegen Compilermeldung über unklare Schleife

1.79pre3

- EAX.25: Interne Änderungen, Einsparung vieler lokaler Variablen. TNN verhält sich an einigen Stellen nun geringfügig anders als das EAX25 des Linuxkernels, eine Zusammenarbeit ist trotzdem problemlos möglich. Die jetzige Implementation hält sich strikt an die Spezifikation. Das maximal mögliche Maxframe wurde auf 32 begrenzt, standardmässig wird nun mit einem Maxframe von 16 gearbeitet.
- Linux: Behandlung von DG1KJD's AX.25-Kernelvariante geändert, soll dieser Kernel verwendet werden, dann ist dies mit "kisstype 11" in der tnn.ini bei dem entsprechenden Port anzugeben. Die automatische Erkennung ENTFAELLT hiermit ersatzlos da sie eine nicht behebbare Schwäche bei der Erkennung des Kerneltyps hatte, sie war darauf angewiesen dass zuerst ein Frame empfangen werden musste bevor auf dem KJD-Stack korrekt gesendet werden konnte. Die Portparameter für KJD-Ports müssen weiterhin mit den entsprechenden Zusatztools gesetzt werden, TNN kann dies mangels Testmöglichkeit noch nicht.

1.79pre4

- Statistik des IP-Kernelinterfaces wurde beim CLEAR-Kommando nicht mit gelöscht. Korrigiert.
- Weitere Anpassungen wegen KJD-Kernel
- Im Linuxteil testweise alle Zählvariablen in Schleifen von i++ auf ++i umgestellt da Präfixoperation keine temporäre Variable benötigen. Weiterhin alle Zählvariablen ohne weitere Bedeutung auf "register" umgestellt.
- + Aenderungen von Bernd DG8BR am FlexNet-Modul:
  l3vc.c
  - Digisystem (X)NET hinzugefügt.

  l7showl3.c
  - Erkennung von XNET in Funktion "putrou" erweitert.
  - Unbekannte Digiprogramme wurden nicht erkannt. Wurde nicht richtig beendet.

1.79pre5

- Zusätzliche Makros für Prompt:
  '%l' Anzahl aktiver L2-Links
  '%p' Portnummer
  '%P' Pseudo-Name des Ports
  '%u' Anzahl der User auf dem aktuellen Port
  '%%' %
  Zur Erinnerung: diese und die anderen Prompt-Makros funktionieren auch in CTEXT.TXT und CTEXT.<portnummer> !!!
- Linux: AX25IP umgekrempelt, es ist nun ein paralleler Betrieb von UDP und IP möglich. Ebenfalls kann die UDP-Portnummer, auf der TNN hört, nun während des laufenden Betriebs geändert werden.
  ACHTUNG !!! Die entsprechenden Ports sind netzwerkseitig nur offen, wenn es auch AXIPR-Routen gibt!!! So lange keine Route oder eine Defaultroute eingetragen wurde, empfängt TNN deshalb auch nix!!! Dies gilt für IP und UDP jeweils getrennt, ohne IP-Routen kein IP-Empfang, ohne UDP-Routen kein UDP-Empfang!!! Die Ports öffnen und schliessen automatisch je nach Vorhandensein von Routen. Ohne eingetragene Routen sind sie sowieso wertlos, da TNN Sendeframes wegschmeisst, wenn es keine IP-Adresse ermitteln kann. Für Empfangsversuche ist also z.B. eine Defaultroute zu 127.0.0.1 im zu beobachtenden Mode notwendig.
  Die standardmäßig geschlossenen Ports sollten auch gegen Angriffe aus dem Internet schützen (Flooding etc.).
- Linux: Fehler im Kernel-AX.25 behoben, es war kein RX/TX mehr möglich, weitere Anpassungen wegen KJD-Kernel
- Der "Routes"-Befehl zeigt nun an, wie viele Routen wirklich auf einen jeweiligen Nachbarn zeigen.

Die Zahl unter "Dst" gibt wie bisher an, wie viele Ziele von diesem Nachbarn bekannt sind, unter "Rou" steht nun zusätzlich, für wie viele Ziele dieser Nachbar derzeit der beste Weg ist.

1.79pre6

- Diese Version wurde nie veröffentlicht, sie enthielt nur Veränderungen, um unter AX25IP mehrere Ports zu ermöglichen. Dies wird benötigt, um auf Knoten mit Internetlinks mehrere Verbindungen in Mode N oder N- zu ermöglichen. Um der Verlinkung via Internet keinen weiteren Vorschub zu leisten, wurden die Ergebnisse nicht in den Code übernommen.
  Funktionieren tut das oben beschriebene dennoch in der nicht veröffentlichten pre6. Teilweise Anpassungen sind noch notwendig, werden aber nicht weiter verfolgt.

1.79pre7

- einige History-Files nachträglich geupdated
- Linux: Race-Condition bei Verwendung des Hostmode-Sockets behoben. Es konnte vorkommen, dass TNN sich nicht starten liess, wenn der Socket verwendet werden sollte. Starts mit Konsole gingen in diesem Fall jedoch problemlos.
- Linux: kernelif.c: einige Newlines in Ausgaben in Returns geändert

1.79pre8

- Probleme bei den neuen Makros behoben
- In l3tab.c::l3_find_route() wurde beim Aufbau des VIA-Feldes bei L2-Connects eine nicht initialisierte Variable zum Vergleich herangezogen. Dies konnte dazu führen, dass das VIA-Feld nicht korrekt aufgebaut wurde, weil eventuell ein falscher Linkpartnertyp festgestellt wurde.
- Funktion l7utils.c::getdig() bereinigt, Rückgabetyp war nicht mit erwartetem Typ an anderen Stellen im Code identisch und konnte zu falschen Ergebnissen führen.
- Linux: Problem mit dem Kernelinterface fuer KJD-Kernel behoben, es konnten keine Interfaces geöffnet werden.
- In all.h die schaltbaren Optionen neu gruppiert, selten benutzte Optionen weiter unten eingetragen, ausserdem sind die L4-Erweiterungen nun standardmässig eingeschaltet.

1.79

- neuen L4 vorerst wieder abgeschaltet um volle Kompatibilität zu anderen Systemen zu wahren
- sonst keine Änderungen seit pre8, nur Debugmodus ausgeschaltet und Hinweise auf geänderte Stellen aus dem Source entfernt, Adressen in der ALAS aktualisiert, ReadMe auf Release angepasst

1.79mh01

- An FlexNet-Nachbarn melden wir statt wie bisher nun jeden unserer Locals nicht mehr einzeln, sondern nur noch wie bei FlexNet üblich einen SSID-Bereich.
  Der gemeldete Bereich errechnet sich aus dem Knotencall sowie der minimalen und maximalen SSID aller per Lokaleintrag (L/L+) angeschlossenen Ziele, die das gleiche Call wie der Digi haben. Versteckte Locals (Alias mit '#' am Anfang) werden ignoriert.
  Die Berechnung des Bereiches erfolgt bei der Sendung des 0er-Frames beim Linkaufbau zu einem FlexNet-Nachbarn. Es kann nun aber vorkommen, dass nicht hinter allen SSID des gemeldeten Bereiches auch ein Local ist. Des Weiteren ist für andere Knoten nicht ersichtlich, dass DB0XYZ-4 gar nicht direkt am Netz hängt, sondern sich eigentlich hinter DB0XYZ befindet. Aus diesen Gründen müssen nun auch folgende Linkwünsche akzeptiert werden:
  - Linkwünsche, die an einen Local gerichtet sind. Diese Linkwünsche werden mit der normalen Gateway-Funktion umgesetzt, der VIA-Schwanz des eingehenden Links wird entfernt!
  - Linkwünsche, die an eine SSID unseres gemeldeten Bereiches gerichtet sind, hinter denen aber kein Local steckt. Diese Links landen nun ganz normal im Knoten.

  aber:
  - Linkwuensche an Locals die es zwar gibt, die aber grad nicht erreichbar sind (L+), bleiben unbeantwortet. In diesem Fall landet man auch nicht im Digi!

  Ändert sich der verfügbare SSID-Bereich waehrend der Interlink zum FlexNet-Nachbarn schon besteht, so wird kein Update des Bereichs an den Nachbarn durchgeführt, da hierzu der Link gekappt werden müsste. (FlexNet sieht hier nur die Möglichkeit vor, nach einer Änderung des SSID-Bereichs diesen per Link-Reset neu zu melden)
  Es ist nun auch möglich, direkt auf den Einstiegen die Locals unter deren Calls zu connecten. Dies wird später vielleicht noch so eingeschränkt, dass dies nur auf Ports mit einem FlexNet-Interlink möglich ist, und auf allen

anderen Ports wieder normal mit via connected werden muss.
- Linux: Beim Ändern des UDP-Ports beim AXIPR-Befehl blieb TNN komplett stehen. Es wurde keine Meldung über die positive Änderung angezeigt. (Problem gemeldet von Oliver Kern)
- Linux: Nicht existierende Kernel-AX.25-Interfaces wurden nicht korrekt als nicht funktionierend gekennzeichnet, dadurch war beim Start ein Hänger möglich. Korrigiert.
- Linux: Die PCISCC4-Einsteckkarte ist nun mit Hilfe des 2.4.x-Treibers von F6FBB mit normalem Kernel-AX.25 nutzbar. TNN verwaltet die wichtigsten Portparameter selbst (TXD, TXT, PERS, Duplex) und überträgt sie bei Änderungen an die Karte. Für die korrekte Konfiguration des angeschlossenen Modemtyps MUSS das zusätzliche Programm "setpciscc" verwendet werden!!! Mit diesem Programm gemachte Änderungen werden allerdings von TNN (noch) nicht wieder übernommen, da keine Benachrichtigung erfolgt.
  Zu aktivieren in include/all.h -> #define PCISCC4_KAX25 einschalten, aber NUR wenn ein 2.4-Kernel mit dem Treiber vorhanden ist!!! Mit einem normalen 2.4-Kernel kann dann NICHT compiliert werden!!!
- L4: "PID-Race" beim neuen L4 behoben. PID-Uebermittlung sollte nun fehlerfrei funktionieren.
- Der IP-Router baut nun auch ausgehend Extended-AX.25 auf, wenn die zu connectende Station im MHeard mit EAX.25 gekennzeichnet ist.
- EAX25: Werte von EAXMODE beim PORT-Befehl geändert:
  Mode 0: nur AX.25 (unverändert)
  Mode 1: AX.25 und EAX.25, Connects nach MHeard (unverändert, default)
  Mode 2: AX.25 und EAX.25, ausgehende Connects zuerst immer in EAX.25, erfolgt nach zwei SABME keine Antwort, dann Rückfall auf AX.25
  Mode 3: nur EAX.25 erlaubt (wie der alte Mode 2)
- Verbesserungsvorschläge von DL1XAO eingepflegt
- In allen Files im Copyright die Jahreszahl angepasst

1.79mh02

- Linux: "Test"-Befehl konnte AX25IP und AX25IPX zum Absturz bringen. Korrigiert.
- + Linux: AX25IP hat nun einen automatischen Routenlerner, der bei eingehenden Connects den Absender in die AX25IP-Routentabelle mit aufnimmt und mit einem inaktivitäts-Timeout versieht. Läuft dieser ab (1 Stunde), so wird der gelernte Knoten wieder ausgetragen. Gelernte Knoten werden bei "SP" nicht mit ausgegeben! Das neue Feld "Timeout" der AX25IP-Routenausgabe (Befehl "axipr") gibt die restliche Lebenszeit des Knotens bis zur Austragung an.
  Einzuschalten in all.h, #define AX25IP_DYNLEARN.
  Wird mit AXIPR (nur neue Syntax!!!) ein Timeout von 0 eingestellt, so werden gelernte Knoten NICHT vergessen und verbleiben in der Liste! Ihre IP wird aber weiterhin überprüft und geupdated wenn das Call plötzlich mit einer anderen IP gehört wird (dyndns). Knoten mit einem Timeout von 0 werden bei "SP" mit in die parms.tnb ausgegeben!
- Linux: LOOPBACK-Interface schaltbar gemacht, defaultmässig ist es nun deaktiviert. (hat wohl sowieso keiner benutzt, oder???) Wer es doch braucht, Einzuschalten in all.h, #define LOOPBACK
- Linux: Code an vielen Stellen aufgeräumt, mit zusätzlichen Kommentaren versehen und genauere Fehlermeldungen im L1 eingebaut.
- Linux: Neuer Kommandoparser und Syntax für AXIPR, der Befehl orientiert sich nun grob am Linux "route"-Kommando.
  ACHTUNG, es wird immer entweder NUR die neue ODER die alte Syntax verstanden!!! => tnb's in der alten Syntax versteht der neue Parser nicht und die AXIPRs fehlen dann! Soll die alte Syntax (= der alte Parser) wieder verwendet werden, dann #define AXIPROLDSYNTAX in all.h setzen! Bei "SP" wird ebenfalls in der neuen Syntax geschrieben falls nicht auf alte Syntax zurückdefine'd wurde.
  Noch mal: neue TNNs verstehen die AXIPR-Kommandos in der tnn179.tnb von alten Versionen nicht mehr (und umgekehrt)!!! Einen Konverter gibt es (noch) nicht.
  Nur Einträge in der ax25ip.cfg werden von beiden Versionen verstanden!
  Die neue Syntax lautet wie folgt:
  axipr {add, +} {call, "default"} <IP / Hostname> [<UDP> [<UDP-Port>]]
  axipr {delete, del, -} {call, "default"}
  axipr myudp [UDP-Port]
  axipr {loglevel, log} [Loglevel] (NEU!!!, siehe ax25ip.cfg)
  axipr timeout [seconds] (NEU!!!, siehe dyn. Routenlerner)
  Ein paar Beispiele:
  IP-Route hinzufügen: axipr add dg9obu-1 1.2.3.4
  UDP-Route hinzufügen: axipr add dg9obu-1 1.2.3.4 udp 12345
  Defaultroute (IP): axipr + default 1.2.3.4
  Route löschen: axipr - dg9obu-1
  Defaultroute löschen: axipr del default

Loglevel ändern: axipr loglevel 3
UDP-Port ändern: axipr myudp 12345
Timeout ändern: axipr timeout 7200
Bei erfolgreicher Abarbeitung erfolgt bei den Kommandos keine erneute Ausgabe der Liste und man erhält gleich wieder das Prompt. Nur im Fehlerfall erfolgen Fehlerhinweise. Bei Verwendung der neuen Syntax kann (eigentlich) auf die ax25ip.cfg verzichtet werden, alle Einstellungen können nun dynamisch verändert werden und sind über die tnn179.tnb einlesbar.
Weiterhin wurde bei der alten und neuen Syntax ein kleiner Fehler bei der MyUDP-Port-Änderung behoben. Der UDP-Port konnte nicht geändert werden, falls es nicht mindestens eine aktive UDP-Route gab.

- Anpassungen an GCC 3.4.0
- Linux: Die CPU-Auslastung kann nicht aus /proc/loadavg ermittelt werden, da dort etwas ganz anderes ausgesagt wird, nämlich die IO-Last. Die echte CPU-Last wird nun intern selbst aus dem Verhältnis von Uptime zu Idletime der letzten zehn Sekunden berechnet. Die Werte aus /proc/loadavg werden zur Information weiterhin ausgegeben.
- Neues L1-Interface: 6PACK (vorerst nur unter Linux)
  6PACK ist ein zum KISS-Tokenring ähnliches, jedoch von Kanalzugriff weiter ausgereiftes Protokoll. Die gesamte Steuerung des TNC erfolgt durch die Software, der TNC ist nur ein dummer Befehlsempfänger. Genau wie beim Tokenring können mehrere TNC in Reihe geschaltet werden. Im TNC ist ein spezielles EPROM mit einer 6PACK-Firmware notwendig!

  Konfiguration: in der tnn.ini:
  Als kisstype ist bei dem entsprechenden Device "12" anzugeben, danach für jeden vorhandenen TNC eine "port"-Zeile (wie beim Tokenring). In der tnn179.tnb muss bei diesen Ports nur "ON" eingetragen werden. Es sollten immer genau so viele Ports wie TNC vorhanden sind zugeordnet werden!
  Eine ausführliche Beschreibung findet sich in der tnnini.all in os/linux/ini.

  Mit dem neuen Befehl "6pack" kann eine Statistik und die aktuelle Zuordnung von TNC zu den Ports abgerufen werden, eine Veränderung von Parametern ist nicht möglich. Die Behandlung von möglichen Fehlern des Rings wird komplett ohne Eingriffsmöglichkeit durchgeführt. Falls ein TNC fehlerhaft sein sollte, so ist dies an den Checksum- und Reset-Zählern zu erkennen.

  Achtung, dieser L1-Treiber befindet sich noch in der Erprobung, er sollte nur nach ausgiebigen Tests produktiv eingesetzt werden! Bitte etwaige Fehler melden! Außerdem sind das Copyright und die Bestimmungen der aus dem FlexNet-Paket stammenden 6PACK-Firmware für TNC2 zu beachten, insbesondere was den Einsatz im CB-Funk betrifft!
- Linux: Die Ausführung von Shellkommandos mit "sh" aus der tnn179.tnb und anderen von dort gestarteten tnb-Files war nicht möglich. Korrigiert.
- AX25IP: Probleme beim Empfang behoben, TCP hat nun Vorrang vor UDP wenn beides aktiv ist. Bei eingeschaltetem Logging wird das Log nicht mehr mit jedem empfangenen Paket unnötig voll geschrieben.
- L3VC: Bei der Meldung an FlexNet-Nachbarn wurden Locals, deren Call nicht gleich dem des Knotens ist, nicht gemeldet. (DAC922 Stefan)
- Zielsystem "DOS16" entfernt und kleinere kosmetische Änderungen (DG8BR Bernd)

1.79mh03

- "Mailbox" und "DX"-Kommando konnten fehlerhaft arbeiten, wenn keine Mailbox oder kein DX-Cluster eingetragen wurde. Initialisierungen verschoben, wurden offensichtlich ignoriert. (Compilerfehler ?)
- Diverse uninitialisierte Variablen gefixt und zusätzliche Sicherheits- Initialisierungen eingebaut. Aufräumen des Speichers bei Programmende.
- Der L2-RX ignorierte die "QST" ARP-UI-Frames, korrigiert.
- Linux: 6PACK hatte RX-Probleme auf TNC2 mit original 6PACK-Firmware, auf dem TNC3 tritt das Problem nicht auf (fehlerhafte/unvollständige 6PACK-Implementation im TNC3).
  Weiterhin gibt es fuer den Sysop nun ein Kommando "6pack loglevel x", wobei 0 <= x <= 4 gilt. Je grösser x, desto mehr Debugausgabe erfolgt in eine Datei namens "6pack.log". Achtung, x = 4 ist sehr ausführlich und sollte nur benutzt werden, wenn ein Anlass besteht und auch genug Plattenplatz vorhanden ist ! Es erfolgt keine Ausgabe der Meldungen auf der Console bzw. im verbundenen Kanal, es wird nur in besagte Datei geloggt.
- Linux: AX25IP vereinheitlicht, so dass Aufgaben des L1-Treibers nicht wie bisher an Stellen ausserhalb des Treibers erledigt werden.
- Linux: AX25IP: Korrektur eines Fehlers, der den Empfang bei IP-Links verhinderte.
- Linux: AX25IP: Mehr Frametypen führen beim dyn. Routenlerner nun zu einem Eintrag.
- Linux: "ax25ip.cfg"-Konfigurationsdatei wurde bis zum Programmende offen gehalten. Wird jetzt nach dem Einlesen wieder geschlossen.
- Linux: Vanessa nur deinitialisieren wenn auch eine Karte gefunden wurde, Speicher nur unmappen wenn er auch

gemapt war.
- Linux: Compilieren war ohne gesetzten "#define KERNELIF" nicht mehr moeglich.
- Linux: Lock-Files ordentlich abräumen bei Fehlern während des L1-Setups.
- Folgende schaltbare Funktionen wurden aus all.h entfernt und befinden sich nun fest im Code:
  + IPRTCACHE IP-Routencache
  + REALROUTECOUNT Zählung der Routen zum Nachbar
- INP-Modul aufgeräumt, hier war sehr viel versionsspezifischer Code bzgl. erster INP-Implementationen. Dieser Code wird nun NICHT mehr mit compiliert, er ist nur noch auf Knoten notwendig die mit TNN-Nachbarn mit Versionen kleiner als 1.76 linken! Wer diese Codeteile benötigt, muss das "#define OLD_INP" in all.h aktivieren, sonst werden so alte Nachbarn nicht mehr verstanden! Besagter Code wird asap dauerhaft entfernt, dies ist nur eine Übergangslösung!!!
- Linux: "sm02"-Patch integriert, fuer alle HDLC-Devices ("bc*"-Interfaces) und SoundModem ("sm*") wird nun DCD und PTT vom Kernel abgefragt.
  Einzuschalten mit "#define HDLC_DCDPTTSTAT" in all.h.
  Für den SCC-Treiber kann nun ebenfalls die PTT-Information vom Kernel erhalten werden, einzuschalten mit "#define SCC_DCDPTTSTAT".
- Weiterhin kann nach einem kleinen Patch am Kernel auch der DCD-Zustand der SCC-Devices abgefragt werden, hierzu ist zusätzlich der "#define OBU_SCC_DCD" zu aktivieren. Achtung, dies funktioniert NUR mit verändertem Kerneltreiber, mit "vanilla"-Kerneln bzw. deren SCC-Treiber ist ein compilieren nicht mehr möglich! Nähere Infos finden sich in "sccpatch.txt".
- Kommandozeile für externe Programme konnte überlaufen. Behoben. (sm04)
- Linux: Der Routenlerner muss die direkt gehörte Station, also ggf. das letzte Via, lernen und nicht unbedingt den eigentlichen Absender. (sm04)
- Linux: AXIPR wertete in der alten Syntax bei "R +" die angegeben IP-Adresse nicht korrekt aus.
- Linux: Interaktive Shell. Wird bei "sh" kein direkt auszuführender Befehl angegeben, so erhält man eine interaktive Shell. Befehle wie z.B. "sh ls -l" werden wie bisher direkt ausgeführt. Der Timeout für nicht-interaktive Befehle beträgt weiterhin eine Minute, die interaktive Shell hat hier fünf Minuten mit Timeout-Warnung. (Nur als Sysop, nicht für Benutzer verfügbar)
- Bei UI-Frames wird der via-Pfad ausgewertet und der Weg zum nächsten Hop bestimmt. Es gelten die folgenden Einschränkungen:
  Falls das eigene Knotencall ...
  … das nächste Via ist: Falls es noch weitere Via nach uns gibt UND der nächste Hop per L2 erreichbar (also ein Local oder Flexnet-Nachbar) ist, wird das Frame auf dem entsprechenden Port ausgegeben. Sind wir das letzte Via in der Liste, dann wird geprüft, ob das Ziel ein Local oder ein lokaler User ist und falls dem so ist, das Frame auf dem entsprechenden Port ausgegeben.
  ... nicht das nächste Via ist: Es wird geprüft, ob das gewünschte Via bekannt UND per L2 (Flexnet oder Local) erreichbar ist, und das Frame dann auf dem entsprechenden Port ausgegeben. Calls auf Userports werden hier nicht als mögliche nächste Via-Ziele berücksichtigt. Ebenfalls werden auf Ports empfangene Frames, die nicht an uns gerichtet sind und wenn der Port keine Interlinks hat, ignoriert.

  Grundsätzlich findet keine Ausgabe von UI-Frames statt, wenn das naechste notwendige Ziel, also das nächste Via oder der Zielknoten, nur per NETROM erreichbar ist. Der via-Pfad wird bis auf das Ändern des H-Bit beim eigenen Call nicht verändert!

  Eventuell soll später mal ein Flexnet-aehnliches UI-Forwarding implementiert werden, da ich aber komplett im NETROM-Land sitze und keine Möglichkeit habe Flexnet zu beobachten, bin ich auf Traces unter genauer Angabe der Situation angewiesen. Wer so etwas oder sogar Dokumentation beisteuern kann, bitte zuschicken. (dg9obu@nordlink.org oder dg9obu@db0uhi.#nds.deu.eu)

  Die schon im Code vorhandene Mailbaken-Funktion, die mit Hilfe des Knoten-Aliasses eine gezielte Aussendung von UI-Frames auf bestimmten Ports ermöglichte, ist weiterhin unverändert vorhanden.

1.79mh04

- INP meldet bei Änderung der Knoten-IP oder Subnetzmaske dies allen INP-faehigen Nachbarn
- Output-Programm modifiziert: Patch von DK2CRN eingebaut (zusätzliche Wartezeit für STROBE-Leitung) und zusätzliche Prüfungen (DG9OBU).
- Ausgabe des SSID-Bereiches der Linkpartner beim "routes"-Befehl
- AX.25 ARP-"QST"-Frames die noch erreichbare Ziele im VIA-Pfad haben, reichen wir nun weiter, ist das nächste Ziel nicht erreichbar, schmeissen wir sie weg.
  Frames die uns als letztes VIA haben nehmen wir an, weil der Vorgänger es auf Grund des Pfades zu uns geschickt hat und wir kein weiteres Ziel dafür haben.

- Linux: make: Unterstützung von externen Linkerflags, make-Aufruf berücksichtigt nun die Variable "LDFLAGS_LIN" beim Linken.
  Um z.B. statische, gestrippte Binaries zu erhalten nimmt man:
  "make LDFLAGS_LIN=-static LDFLAGS_LIN+=-s"
  Variable wird auch an die makefiles in "contrib" durchgereicht.
- Linux: Erkennung für MIPS-Systeme beim Compilieren (WRT54G, MeshCube):
  (aut. Erkennung oder "make MIPS=YES [...]" bei Crosscompilierung)
  Für die interaktive Shell wird in anderen Verzeichnissen und mit anderem Namensschemata nach freien pty/tty-Paaren gesucht. (/dev/pty/{m,s}[0-255])
  "contrib/output" wird für MIPS-Systeme NICHT mehr übersetzt. Dies ist für den WRT54G oder MeshCube gedacht, diese beiden haben keinen parallelen Printerport wie die i386-kompatiblen Systeme. "make" gibt hier einen kurzen Hinweis aus dass nichts gebaut wurde.
  "Vanessa" wird ebenfalls für MIPS-Systeme NICHT mehr genutzt, aber noch ein restl. Rumpf mit eincompiliert. Diese Hardware gibt es für MIPS einfach nicht.
  Diese Variable wird ebenfalls an makefiles in "contrib" durchgereicht.
- Linux: kleine Anpassung im Convers damit keine Meckermeldungen mehr beim Compilieren mit dem GCC unter Linux ausgegeben werden
- Linux: mit dem Befehl "setshell" kann eine abweichende Shell gesetzt werden.
  Beim Start wird die Umgebungsvariable SHELL gelesen und eine dort eingestellte Shell uebernommen. Ist diese Variable leer so erfolgt eine Warnung und der Shell-Mechanismus ist nicht nutzbar.
  Wichtig fuer "busybox": die Shell sollte ein symbolischer Link sein damit busybox weiss, was zu tun ist. Also nicht "setshell /bin/busybox" sondern "setshell /bin/bash", wobei "/bin/bash" ein symbolischer Link auf "/bin/busybox" sein muss.
  Beispiel:
  "setshell" liefert eine kurze Hilfe
  "setshell ?" fragt die aktuelle Einstellung ab
  "setshell /bin/sh" setzt /bin/sh als zu verwendende Shell
- L2-Connects mit Via-Pfad berücksichtigen bei der Suche des nächsten Ziels nun auch die MHeard-Liste wenn kein Ziel im L3 gefunden wurde. Der Via-Pfad wird durchgereicht.
- Bei der verzögerten Bestätigung von Hop-to-Hop-Verbindungen konnte das UA-Frame in seltenen Fällen mit falschen Flags gesendet werden. Die Flags des in diesem Fall gesendeten UA-Frames sind nun fest verdrahtet (Response und Final -> "UA-").

1.79mh05

Diese Version besteht hauptsächlich aus Bugfixes, sowie einigen Funktionen, die aus der CB-Variante übernommen wurden. Im Einzelnen sind geändert:

- Fix für Absturz bei Connect mit Via-Angabe.
- Alle Makros, die in CTEXT.TXT funktionieren, können nun auch in QUIT.TXT verwendet werden.
- Linux: bei der nicht-interaktiven Shell konnte bei Kommandos, die keine Rückgabe erzeugten, die "ERROR while read()ing"-Meldung noch erscheinen, obwohl kein Fehler vorlag. Korrigiert.
  Weiterhin konnte die Ausgabe großer Datenmengen (sh cat /var/log/messages) dazu führen, dass eine Sicherheitsfunktion ansprach und den überfüllten Link abwarf. Jetzt wird die Verbindung nur noch mit maximal 100 ausstehenden Frames mit Daten aus der Shell gefüllt.
- MHeard/L3MHeard zeigt nun bei Änderung der Listengrösse nicht mehr die Liste an, sondern gibt nur noch eine Meldung über die neue Grösse aus.
- Nickname-Unterstützung für den Convers. Es ist nun die Angabe eines Namens möglich, der zusätzlich vor dem Rufzeichen angezeigt wird. Die Eingabe wird an andere Convers-Hosts weitergeleitet, sofern sie die Nickname-Fähigkeit in ihren Feature-Flags angezeigt haben. Die Umsetzung erfolgte in Anlehnung an die Implementierung im tpp-convers 1.14, jedoch erfolgt die Anzeige des Nicknames nicht so häufig wie im Original.
  Hinzugekommen sind die beiden Kommandos:
  /NICKname <Name> oder "@" setzt den Namen, der @ löscht ihn
  /NONickname löscht den Namen (wie "/nick @")
  Angepasste Hilfedateien finden sich im Verzeichnis "doc"!
  Einzuschalten mit dem #define CONVNICK in all.h. Ist es aktiviert, werden andere Feature-Flags gesendet und die Version "3.14c" bzw. "pp-3.14t" gemeldet.
- Onlinehilfe angepasst und geupdated
- 6PACK: Ein weiterer Wachhund sorgt nun dafür, dass der intern gespeicherte PTT-Status des TNC rückgesetzt wird, falls Infopakete des TNC verloren gehen.
- Datum im Copyright angepasst
- Linux: Das Interface zum Kernel heißt nun "tnn" statt "tun0" und wurde für die 2.6er-Kernelreihe angepasst.

Weiterhin ist nun die Angabe der Subnetzbits für die Kernel-Seite des Interfaces möglich. (IP-Adresse/Subnetz-Bits). Achtung, bei einigen Kerneln klappt die Aktivierung des Interfaces nicht, obwohl kein Fehler gemeldet wird. In diesem Fall muss in der Datei "kif_up.tnb" der Befehl "sh ifconfig <Interfacename> up" eingetragen werden. Der Interfacename ist "tnn".

- Empfangene und gesendete FRMR-Pakete werden im Trace dekodiert.
- Connectbewertung komplett überarbeitet, dem User sind nun mehr Übersteuerungsmöglichkeiten vorhanden. Die Angabe eines abweichenden Ports ist nun nur noch als letzter Parameter möglich! (c db0abc ... {Portnummer, Portname})
- Linux: bei AX25IP ist der dynamische Routenlerner nun fest eingebaut, der Schalter AX25IP_DYNLEARN in all.h ist somit entfallen.
- Linux: AX25IP arbeitet nun primär mit Hostnamen statt mit IP-Adressen. Wird ein Host angegeben der momentan nicht aufgelöst werden kann, so wird er mit der IP-Adresse 0.0.0.0 in die Routentabelle aufgenommen und bei jedem gesendeten Frame zu diesem Host ein Adressauflösungsversuch unternommen. Somit können über dynamisches IP erreichbare Hosts ohne Kenntnis der aktuellen IP eingetragen werden.
  Wird statt eines Hostnamens eine IP-Adresse eingetragen, so wird diese direkt übernommen, der Hostname wird auf die IP-Adresse gesetzt.
  Dynamische Routen werden nun bei "sp" ebenfalls in die parms.tnb geschrieben, jedoch werden sie auskommentiert und sind somit nicht aktiv. Durch einen manuellen Eingriff können diese Einträge auf einfache Weise permanent gemacht werden.
  Die Routentabelle wird in einem konfigurierbaren Intervall durchsucht und alle Hostnamen in ihre aktuellen IP-Adressen aufgelöst. Sofern geänderte IP-Adressen nicht schon durch den Routenlerner aktualisiert werden, passiert dies nun.
  Hinzugekommen ist der Befehl "LOOKUP <Sekunden>" beim AXIPR-Kommando (nur neue Syntax!!!), eine Einstellung von 0 deaktiviert die Ausführung, voreingestellt sind 180 Sekunden. Achtung, steht kein erreichbarer Nameserver zur Verfügung, kann der Knoten eventuell kurz einfrieren, da die Systemfunktionen hier blockieren.
- Linux: das übermittelte Datum beim AutoBIN-Transfer war um einen Monat daneben.
- Linux: bei gleichzeitiger Benutzung von AX25IP IP- und UDP-Verbindungen, konnte eine Sendung mit einer falschen Portnummer erfolgen (DH6BB)
- Die automatische Berechnung der Port-Parameter (Persistenz, L2-Retry, L2-Timer, ...) kann übersteuert werden. Werte, die weiterhin automatisch berechnet werden, werden bei "po *" mit einem kleinen "a"-Suffix versehen, von Hand übersteuerte Werte erscheinen ohne Suffix. Soll ein nicht mehr automatisch berechneter Port-Parameter wieder in die automatische Berechnung aufgenommen werden, so ist als Wert "a" anzugeben ("po 1 pers=a").
  Standardmäßig werden alle Port-Parameter automatisch berechnet.
  Die EXPERTPARAMETER müssen in all.h aktiviert sein! (ist nun Standard)
- Neues Makro %f zum Einlesen einer Textdatei. Der Dateiname muss direkt hinter dem %f stehen, nach dem Dateinamen ist ein Leerzeichen einzufügen, welches später nicht mit ausgegeben wird. (DH6BB)
  Beispiel im Ctext: "Aktuelle Temperatur %f/usr/local/wx/wx.txt Grad Celsius"

LIZENZ:

Allgemeine Lizenz für Amateurfunk Software (ALAS)

Karsten Heddenhausen
Mozartstraße 5
30823 Garbsen

Dieses Dokument darf beliebig vervielfältigt oder verteilt werden, solange es nicht verändert wird. Für Anregungen, wie es zu verbessern ist, bin ich dankbar.

1. Vorwort:

   Diese Lizenz entstand aus der General Public Licence der Free Software Foundation (GPL). Ich habe versucht, den Sinn zu erhalten und mehr Klarheit hineinzubringen. Einige Passagen sind vollständig entfallen. Es ist aber jedem Nutzer freigestellt, an Stelle dieser Lizenz die GPL Bedingungen anzuwenden. Der Sinn dieser Lizenz ist es, den Autor und den Anwender der Software zu schützen. Es sind daher einige Einschränkungen erforderlich, und es entstehen auch einige Pflichten für den, der die mit dieser Lizenz verbundene Software weitergibt oder verändert. Dies wird dadurch erreicht, dass die Software durch Copyright geschützt wird und der Nutzer durch diese Lizenz die Möglichkeit einer nahezu unbeschränkten Nutzung erhält.

2. Geltungsbereich

   Diese Lizenz gilt für jedes Programm oder Teil eines Programmpaketes, das eine Copyright-Notiz ausgibt, die sich auf diese Lizenz bezieht. Im Folgenden bedeutet „Programm" entweder das Programm oder einen Teil davon. „Du" bist der Lizenznehmer.

3. Deine Rechte:

   Du darfst das Programm nutzen oder kopieren oder verteilen oder verändern, solange Du damit keine kommerziellen Absichten verbindest.

4. Deine Pflichten:

4.1. Du darfst den Copyrightvermerk und den Hinweis auf diese Lizenz nicht verändern, und er muss bei jedem Start des Programms eindeutig für den Benutzer sichtbar sein.

4.2. Du musst jedem Dritten, dem Du eine Kopie des Programms gibst, die gleichen Rechte einräumen, die auch Dir gegeben wurden. Du musst ihm auch die gleichen Pflichten auferlegen.

4.3. Du darfst für die Weitergabe kein Geld verlangen außer den Kosten für das Medium und Porto.

4.4. Du darfst das Programm nur komplett weitergeben, so wie Du es bekommen hast.

4.5. Wenn Du das Programm veränderst oder Teile davon für eigene Arbeiten verwendest - wörtlich oder verändert - so gelten die folgenden Punkte für das daraus entstehende neue Programm:

4.5.1. Du musst deutlich sichtbar angeben:
- Deinen Namen und Deine Adresse und
- die Tatsache, dass Du das Programm geändert hast oder Teile des Programms verwendet hast.

4.5.2. Du musst das Programm entweder mit dem kompletten Quelltext weitergeben oder jedem auf Verlangen eine Kopie des Quelltextes gegen eine Gebühr von maximal der Kosten des Mediums und der Portokosten aushändigen.

4.5.3. Du darfst keine Beschränkungen aussprechen, die über diese Lizenz hinausgehen.

5. Sonstiges

Du erhältst das Programm ohne jede Garantie für Funktion, Fehlerfreiheit oder Anwendbarkeit für eine bestimmte Sache. Du verzichtest auf jede Schadenersatzforderung, gleich aus welchem Grunde. Mit der Nutzung des Programms erkennst Du diese Lizenzbedingungen vorbehaltlos an.

Vers. 1.0, 13-OCT-92

Ende ALAS

ABSCHLUSS:

Was noch genauer erklärt werden muss, werden die (hoffentlichen) Fragen zeigen. Dieses File, so weiss ich schon jetzt, wird noch öfters geändert und ergänzt werden müssen.

Mit freundlichen Grüssen

Nord><Link e.V.
DC7OS
Karsten Heddenhausen
Mozartstraße 5
30823 Garbsen

Ausdrucke (vom Laserdrucker) gibt es gegen Unkostenerstattung von € 0,05/Seite + Porto bei:

Nord><Link e.V.
DG9BHD
Claus Michaelis
Wohnpark 14
26419 Heidmühle

Geführt von	Jens	DH6BB.
Beiträge von	Peter	DB2OS
	Andreas	DB7KG
	Georg	DF2AU
	Nils	DF6LN
	Klaus	DF7BZ
	Andreas	DG1KWA
	Frank	DG3AAH
	Bernd	DG8BR
	Marc-Andre	DG9OBU
	Odo	DL1XAO
	Kurt	DL2LAY
	Rainer	DL5HCD
	Matthias	DL9HCJ
	Peter	HB9PAE
	Daniel	ON5ZS.

Diese Ausgabe wurde Korrektur gelesen von: