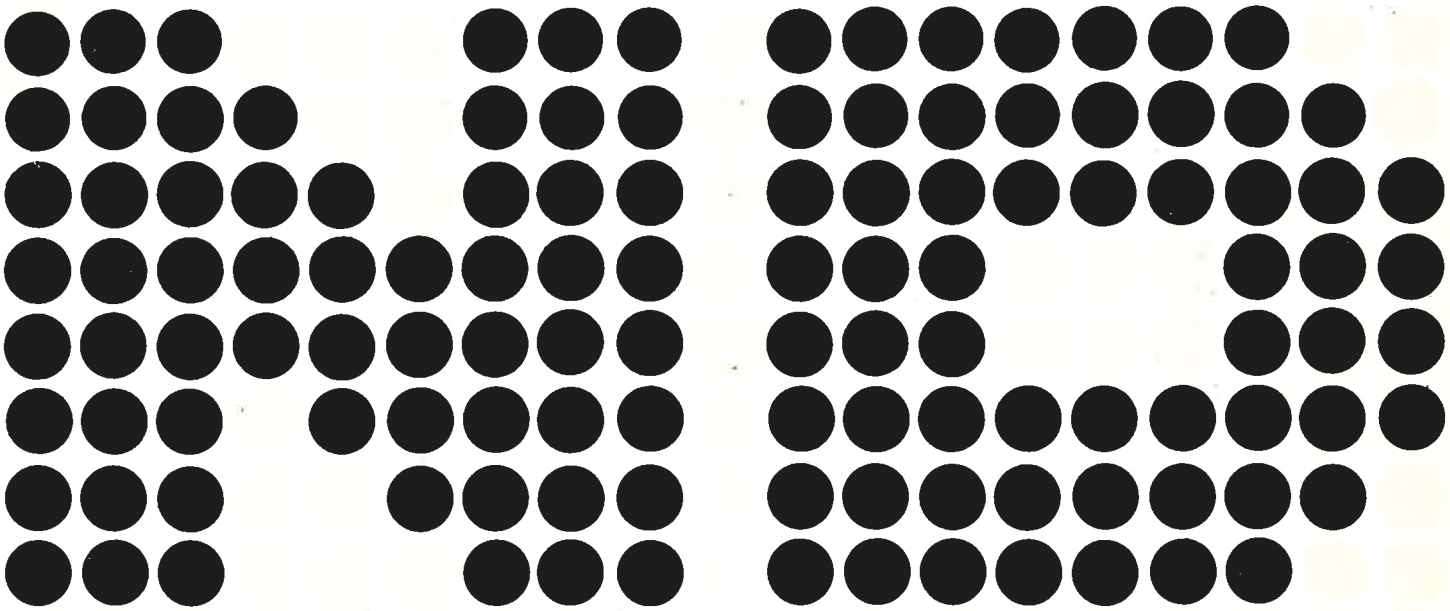
SCANNET

USERS GUIDE

NORSK DATA A.S



SCANNET

USERS GUIDE

K. E. M. Dijkman

May, 1976

REVISION RECORD

Revision	Notes
05/77	Revision 1

Scannet Users Guide
Publication No. ND-60.087.01

NORSK DATA A.S.

Lørenveien 57, Postboks 163 Økern, Oslo 5, Norway

TABLE OF CONTENTS

1.1 How to use this manual

A short introduction to packet-switching and the Scannet-operating principle is given in chapter 1.

Chapters 2 and 3 are intended for the terminal users of the network and describe the Terminal System and the available Network Service Terminals. To learn to use the network, chapters 1 to 3 should suffice.

To gain a better understanding of the system structure, chapters 4 through 6 give a detailed description of its most important parts. Especially the transmission system is treated in detail in chapter 6.

Chapters 7 through 9 are intended for the people responsible for the network management. They describe the possible management actions, error messages and software interfacing standards.

Throughout this manual the following symbols are used:

↵	carriage return
↓	line feed
␣	space
DLE	DLE character (20_8)
↑	control-key

In the examples all user input is underlined.

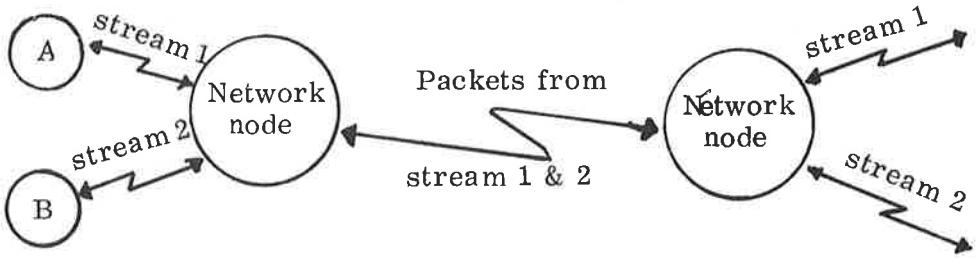
1.2 Introduction to packet-switching

Packet-switching is a comparatively new technique in data communications. It takes advantage of the statistical nature of information flow to share communication facilities between a number of users. Shared communication facilities are more efficiently used than private ones that lie idle when its owner temporarily does not use them. A network will also provide alternative routes from one point to another so that effects of communication line breakdown are minimised.

Packet-switching works as follows:

The users' information stream is cut into packets when entering the network. These packets are supplied with a packet header and a trailing error check part. The packets are then transmitted through the network to their destination. At the destination the packets are put together again to reproduce the original data stream and presented to the user.

I.e.



In a commercially operated packet-switching network the user is charged for his actual use of the facilities. He is not charged if he does not use them, as he would have been if he had permanently leased communication facilities.

Some examples of packet-switching applications:

- telex network
- computer - computer communications
- remote batch/interactive processing
- data entry/retrieval
- complex process control

etc.

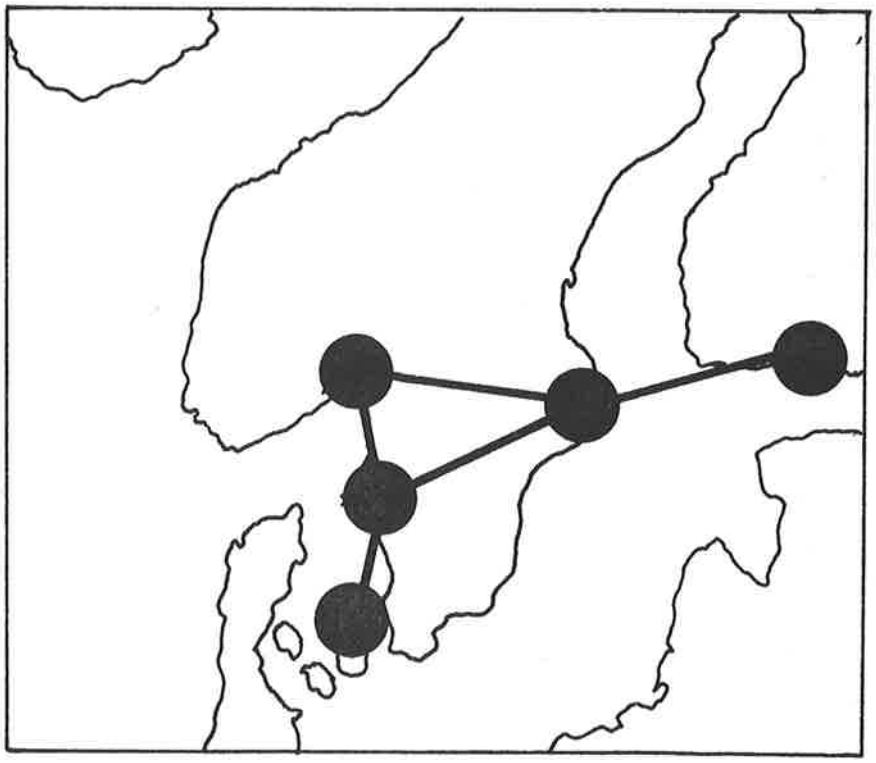
1.3 Scannet configuration

The initial network will consist of 5 nodes, one each in Oslo, Copenhagen, Stockholm, Gothenburg, and Helsinki. The nodes are NORD-12 CPU's with 32 K 16 bit Mos memory. Input/output devices consist of modem interfaces, tape reader and terminal interfaces for permanent or dial-up terminal connections. The communication lines are public leased lines, initially operated at 2400 baud, using synchronous transmission technique.

The network's main objective is to provide easier access to on-line bibliographic data-bases for numbers of the participating organisations.

A number of host-computers in the data-centra of the participants will be connected to the network.

The network is depicted below:



2. Terminal System User Guide

2.1 Introduction

2.1.1. General

Information that is transported through a packet switching network is bundled in packets. These packets contain a header, a data-part and a check-part. The packet header describes,among other things, the packet type, the size of the data part, the destination and the source of the packet, and the transmission priority. The check part is used when packets are transmitted over transmission lines to detect transmission errors.

Characters from a character oriented terminal device connected to the network enter the network one at a time. These must then be assembled into a packet before they can be sent to their destination. The reverse operation must be applied to packets arriving at a terminal. These packets must be disassembled and presented to the terminal one character at a time

These Packet Assembly / Disassembly functions (PAD) are performed by the Terminal System (hereafter called TS).

2.1.2. Some network considerations and terms:

A network overload condition can arise if a terminal keeps entering data into a network without checking that the data also leave the network at some other place. No terminal may therefore send data into the network if it has not received a Ready For Next Message (RFNM) packet from the adressee of its previous packet. Packet assembly may continue while waiting for a RFNM, but the resulting packet may not be sent until a RFNM is received.

The packet source must be sure that the packet destination will accept the packet when it arrives. This implies that the source and the destination must have some agreement for the duration of the data exchange. This agreement is called a connection. Connections are established and terminated by means of special packets exchanged between source and destination.

The terminal user has special commands at his disposal to establish and terminate connections. It would be better to drop the terms "source" and "destination" in this context since full duplex data exchange can take place once a connection is established between two parties. (Be they terminals, host-computers or network service functions) The parties presently "connected" by a connection will only accept packets from each other for the duration of the connection. One can visualize a connection as a two-way data-pipeline between the two parties to a connection. All packets arriving from other sources are returned with an error indication or discarded (depending on packet-type).

2.3.1. Break characters and packet assembly

TS performs the packet assembly function on characters coming from a terminal. However, assembly is not enough; packets must also be sent. So TS must have some means to recognise the end of a packet on input.

Break characters are defined as those characters that terminate packet assembly by signaling to TS that the packet is complete and can be sent. The break character is placed in the packet and is the last data-character of the packet.

All characters with an ASCII value less than 40 (8) are break characters.

NOTE! The maximum packet size is 255 characters. The 255th character is treated as a break character if the user enters a packet that exceeds the maximum packet size.

2.1.4. Full-duplex terminal devices

Most terminal devices can operate in full duplex mode. It may be irritating to the user to observe a mixture of output characters from packets arriving at his terminal and echo of his input. TS will therefore keep output packets waiting once the user has typed the first character of a new input packet. Output proceeds again when the user types the break character that terminates the input packet. If output already was on its way when the user starts his input, then TS will delay echo-output until the output from the output packet is completed.

2.2 Commands

2.2.1. General

All commands under TS have the same form.

They consist of:

- one predefined command mode character that tells TS that the user wishes to execute a command.
- followed by the command selection character that tells TS which command the user whishes to execute.
- followed by arguments, if the command needs arguments.

The command mode character is usually DLE (20_8), but may be changed by the user. TS outputs the appropriate command string after the user has typed the command selection character.

f.i.

DLE C

CALL: ADDRESS ↵

DLE = command mode character
C = command selection character
CALL-command
CALL = command string output by TS
ADDRESS↵ = argument supplied by user

The following commands are available:

Command selection character	command name	argument
L	LOGIN	-
Q	QUIT	-
C	CALL	ADDRESS
B	BREAK	-
D	DEVICE	DEVICE TYPE
E	EDIT	ON/OFF
X	EXCHANGE	NEW CONTROL CHAR
K	KILL OUTPUT	-
F	FEED CONTR.	ON/OFF

Commands are not always legal. Commands are rejected if they conflict with the state the terminal is in.

F.i. LOGIN is rejected if the terminal is already logged in.
QUIT is rejected if the terminal is not logged in.

Command selection characters other than L, Q, C, B, D, E, X, K are always rejected.

TS outputs "BELL" to the terminal if a command is rejected. ("BELL" is a audio-signal that leaves no trace on the terminals output medium.)

NOTE

Commands should not be given if the user has started entering a data-packet but has not yet terminated it with a break-character.

If packet input is not terminated when the user types any command (except KILL), the unfinished input packet is thrown away. TS indicates this by writing ←↙ (Backarrow, carriage return), following the last character of the input packet.

NOTE

Command mode character as data input.

Input of two command mode characters in succession has a special effect. TS will drop the first one and treat the second one as any other data input character.

Command arguments

No edit functions are available during input of command arguments. However, the user can abort the command by typing the command mode character before terminating the argument input with carriage return.

The number of significant characters in a command argument depends on the command in question. However, TS will generate echo on all input during argument input. Command arguments must always be delimited by carriage return (except in the X-command).

2.2.2. Command form and action

2.2.2.1. LOGIN (DLE L)

The user must give the LOGIN-command before he can avail himself of the network services. All commands other than LOGIN are rejected if the terminal is not logged in.

The following takes place when the LOGIN-command is executed:

- The logout-bit in the terminal status block is cleared.
- The terminal device type is set to TTY.
- The default command and edit characters are set and EDIT-mode is set to ON.
- The command string associated with login is written on the terminal.

  < Date.time > HELLO FROM SCANNET

  In which <date. time > reads as follows:

  year/month/day hour. minutes

  f.i. 76/12/31 12.45 HELLO FROM SCANNET

LOGIN is rejected if the terminal was already logged in.

2.2.2.2. QUIT (DLE Q)

The QUIT-command signifies that the user is leaving the network. The following takes place:

- The logout-bit in the terminal status block is set.
- The terminal device type is set to TTY.
- The default command selection character DLE is set.
- The default edit characters are set.
- Command string output:

  <date.time> BYE BYE

Note!! **The terminal is automatically logged off if the connection between the terminal and the node is broken for more than six seconds.**

2.2.2.3. CALL (DLE C)

The user must, as explained earlier, set up a connection before he may send data through the network. Connections are set up by means of the CALL-command. After detecting the command, TS outputs:

CALL : ␣

The name of the party to which the user wants the connection established, must now be spesified as the command argument. Type the name immediately following the command string output and terminate it with a carriage return.

f.i.

DLE C

CALL : HOST01↵

Only eight characters are significant in the CALL-argument. If less than eight characters are given, the name is padded with spaces before TS starts searching for it in the internal name table.

f.i.

CALL : JOHN↵ ≡ CALL : JOHN␣␣␣␣↵ ≡ CALL : JOHN␣␣␣␣DOE↵

TS will output one of the following answers to a CALL-command.

1. If the connection is successfully established:

   < date.time > CONNECTION ESTABLISHED TO <argument >

2. Wrong name supplied as argument.

   ADDRESS ?

3. Connections between terminals are restricted to maintenance purposes only. If the CALL-argument specifies a terminal name, TS will answer:

   SORRY, RESTRICTED ACCESS ↵

4. The connection cannot be made. The reason is explained in the answer, which can be:

- NODE UNREACHABLE	Communication break-down or node unoperative.
- PARTY BUSY, ALT.	The wanted party is fully occupied with already established connections. If the party has an alternative it is specified after "ALT."
- PARTY CLOSED, ALT.	The party is not logged into the network.
- PARTY OFF.LINE,ALT.	The party is logged in but not on-line. Applies to HOST-interfaces.
- POOL LOW	The number of buffers in the free buffer pool in your local node is too low to allow a new connection set up. Try again later.

5. No answer

If TS does not anser at all, one of the following error conditions has occured:

- Communication break-down after TS has sent a call-request to a remote node, but before an answer was received.
- The adressed party has received the call-request packet, but does not answer it with an ok or a denial.

The user can terminate this hangup by typing the Break-command (DLE B). TS will then output:

CALL ABORTED ↵

NOTE

All input, except the BREAK-command, is discarded if given between the terminating carriage return of the call argument and the start of the answer output.

A list of legal CALL-arguments is given in 2.6.

The CALL-command is rejected if:

- the terminal is not logged in.
- the terminal already has a connection.

Some examples

Normal sequence:

DLE C
CALL : DECH2↵

76/04/12 11.25 CONNECTION ESTABLISHED TO DECH2
111112223344↵
111112223344

DLE B
76/04/12 11.25 CONNECTION TERMINATED

Error situations:

DLE C
CALL : DECH4↵

76/04/12 11.27 NODE UNREACHABLE

DLE C
CALL : HOST00↵
DLE B
CALL ABORTED

DLE C
CALL : TER0.0↵
SORRY, RESTRICTED ACCESS

DLE C
CALL : DMP0↵

76/04/12 11.26 PARTY BUSY, ALT.

2.2.2.4. BREAK (DLE B)

The BREAK-command is used to terminate a connection. After DLE B input TS will normally respond with:

< date.time > CONNECTION TERMINATED

However, if the last packet entered by the user is not yet sent, owing to a delayed RFNM on the previous packet, TS will ask the user whether it may be deleted or not:

DELETE INPUT LINE ?:

The answer should be a single character; Y for yes, N for no. If Y is typed, TS will delete the waiting packet and proceed with the BREAK-command. If N is typed, the BREAK-command is aborted.

f.i.

CALL : MAKE0↵

76/04/12 13.13 CONNECTION ESTABLISHED TO MAKE0
1234567890↵
ABCDEFGHIJ↵
DLE B
DELETE INPUT LINE ?: N

DLE B
DELETE INPUT LINE ?:Y
76/04/12 13.13 CONNECTION TERMINATED

A connection can also be terminated by the other party, or abnormally terminated for various reasons.

In these cases TS outputs:

< date.time > CONNECTION TERMINATED

or:

< date.time > ABNORMAL DISCONNECT

If the terminal has an unsent but terminated packet, TS will preceed the above mentioned output with: LAST LINE DELETED.

An abnormal disconnect can be caused by:

- communication line failure

  Communication facilities with the other parties' node have been down longer than approx. 2 minutes. The disconnect occurs only if a packet from the users' terminal is waiting to be transmitted by the communication system. The abnormal disconnect is generated by the communication garbage collection module when it finds a packet that is older than 2 minutes and has not yet succeeded in crossing a communication line. The garbage-collector will then change the packet type to "abnormal disconnect" and bounce it back to the source of the packet.

- node reload or restart

  If a node is reloaded or restarted, all information about existing connections is destroyed in that node. If a user, prior to the reload, had a connection to a party in that node, he will receive an abnormal disconnect next time he sends a data packet.

The BREAK-command is illegal if:

- the terminal is not logged in
- the terminal has no connection

2.2.2.5. DEVICE (DLE D)

The DEVICE-command is used to select device dependant values of the following parameters:

- default command mode character
- default edit characters
- bell code
- line size
- number of filler characters after carriage return and after line-feed

After detection of the command,TS will output:

DEVICE:

The user should supply one item of the following list as argument:

TTY ↵	for	Teletype
ASR ↵	for	ASR 33/35
SIL ↵	for	Silent 700
VDU ↵	for	Display type devices

If anything else is specified, TS will answer with:

ARGUMENT?

The following values are set according to the device type selected:

	TTY	ASR	SIL	VDU
Command mode character	DLE	DLE	DLE	DLE
Cancel character	↑X	↑X	↑X	↑X
Backspace character	↑H	↑H	↑H	↑H
Bell code	$7_{(8)}$	$7_{(8)}$	$7_{(8)}$	$7_{(8)}$
Line size	$74_{(10)}$	$74_{(10)}$	∞	∞
CR-fillers	0	0	5	0
LF-fillers	0	0	0	0

NOTE

During the LOGIN and the QUIT commands the device type is always reset to SIL !

2.2.2.6. EDIT (DLE E)

The EDIT-command is used to switch the EDIT-mode ON or OFF.

The command string output for the EDIT-command is:
EDIT:

The user should supply ON or OFF as argument.

f.i.

DLE E	DLE E
EDIT: ON↵	EDIT: OFF↵

Only two characters in the EDIT-argument are significant.

If EDIT-mode is ON, the user has two edit functions available during packet input. These are:

- delete current packet input so far (cancel)
- delete last character (backspace)

Two special characters are assigned to trigger these functions if EDIT-mode is ON.

Default for these characters are CNTR X (↑X) for cancel and CTRL H (↑H) for backspace. TS will output ← ↵ as echo for cancel and \ for backspace.

If EDIT-mode is OFF, the edit characters are treated as ordinary data input.

2.2.2.7. EXCHANGE (DLE X)

The EXCHANGE command is used to select new characters for the command mode character and for the edit characters.

TS outputs as command string:
DLE/CAN/BSP:

The user should then supply three characters that will be taken as the new command/edit characters. After the third character TS will generate CR/LF.

If the character specified for the command mode character (the first) is equal to one of the edit characters, the command is rejected and TS outputs:
ARGUMENT?

In this case the old set of special characters is retained.

The EXCHANGE-command is illegal if the terminal is not logged in.

Examples:

Set @ as command mode character, ↑Q as cancel, ↑A as backspace.

DLE X
DLE/CAN/BSP: @ ↑Q ↑A ↵

2.2.2.8. KILL OUTPUT (DLE K)

Current packet output may be aborted by using the KILL OUTPUT-command (DLE K).

The current output is stopped and the output packet is removed as if output was indeed complete.

The command is illegal if output is not currently underway.

2.2.2.9 FEED CONTROL (DLE F)

Normally Line Feed control is determined by the default value of the called party during connection setup.

If Feed Control is turned off after connection setup the terminal system will no longer generate a line-feed after carriage return output.

Examples:

Line feed control on

DLE F

LF: ON ↵

Line feed control off:

DLE F

LF: OFF ↵

2.3 Data input-output

Data input is accepted when the terminal is logged in and a connection is properly set up.

Input data is buffered until a break-character is typed. The packet is then supplied with a header and sent to its destination. However, the packet is not sent if no RFNM is received on the previous packet.

All characters with an ASCII value less than $40_{(8)}$ are break characters. I.e. all nonprintable characters plus carriage return, line-feed and form-feed.

Line-feed echo / output

TS will generate a line feed after carriage return (both on input and on output) if Feed Control is ON. To prevent double line-feed where only one is intended, TS will supress line feed echo if the first character input after carriage return is a linefeed. The line feed is sent in the data packet. The same applies to output. If the first character after carriage return in the output packet is a line feed, it is skipped.
When a break character terminates a packet, input may continue until the next break character. Continuation of input then depends on whether a RFNM was received on the previous packet. If so, input may continue. If not, all input is discarded, except commands, as indicated by the fact that "BELL" is echoed for each character input.

NOTE

Input and output will normally interleave. However, all output is kept waiting once packet input is started. Output is started again when a break character is typed in.

The user should therefore not leave input unterminated while waiting for output.

Output will also continue if an input-packet is removed by the edit functions CANCEL (or backspace on the first character of the packet).

Illegal input

If the terminal is not logged in, all input will be echoed, but TS will write PLEASE LOGIN when a break character is typed. The packet is not sent.

If no connection is established, TS will write:

CONNECTION?

when a break character is encountered. The packet is not sent.

2.4 TS output summary

command strings	command / argument
<date.time>HELLO FROM SCANNET	LOGIN
<date.time>BYE BYE	QUIT
CALL:	CALL, supply address as argument
EDIT:	EDIT, supply ON/OFF as argument
DEVICE:	DEVICE, supply as argument TTY, ASR, SIL, VDU
DLE/CAN/BSP	EXCHANGE, supply new special characters
LF:	FEED CONTROL, answer ON/OFF
answers/error strings	**cause/meaning**
LOGIN PLEASE	break character input and terminal not logged in
CONNECTION?	break character input and no connection
"BELL"	- illegal command - unknown command - input temporarily closed
ADDRESS?	wrong argument in CALL
ARGUMENT?	wrong argument in command
SORRY, RESTRICTED ACCESS	terminal - terminal connections restricted to maintenance only
LAST LINE DELETED	last input line deleted when the connection is terminated by the other party, or by an abnormal disconnect
DELETE INPUT LINE?	querry when BREAK-command entered and a packet waits for RFNM
<date.time>CONNECTION ESTABLISHED TO <arg.>	connection suceessfully established

<date.time> PARTY BUSY, ALT. " NODE UNREACHABLE " PARTY CLOSED " PARTY OFF-LINE	connection cannot be made
POOL LOW	too few buffers in free buffer pool to warrant a new connection. Try again later.
CALL ABORTED	CALL aborted by BREAK-command

2.5 TS command summary

command	action	argument(s)
DLE L	Login	-
DLE Q	Quit	-
DLE C	Call	address
DLE B	Break	-
DLE E	EDIT mode	ON/OFF
DLE X	Exchange	New DLE/CAN/BSP
DLE K	Kill output	-
DLE D	Device	TTY, ASR, SIL, VDU
DLE F	Feed Control	ON/OFF

Edit functions (standard)

↑ X	Cancel input
↑ H	Backspace character

2.6 CALL arguments

In the names below n stands for the network node number, ranging from 0 to 4.

F.i. DECHn = DATA ECHO in node - n
DECH0 = " " in node - 0

Network service terminals

Call arguments:

DECHn	Data echo node n
DSNKn	Data Sink node n
DSCn	Data source node n
DMPn	Debug maintenance node n

Physical terminal names

Physical terminal names range from TER0.0 to TER0.5 in node ø.

TERn.0 to TERn.5 in node n $0 \leq n \leq 4$

HOST interface names:

Node 0	Node 1	Node 2	Node 3	Node 4
CHEM	MEDLI	HOST20	HOST30	HOST40
HELP	CAC	⋮	⋮	⋮
NEWS	BYGGD	⋮	⋮	⋮
HOST03	IRRD	⋮	⋮	⋮
⋮	EPOS	⋮	⋮	⋮
HOST07	HOST15	⋮	⋮	⋮
	⋮	⋮	⋮	⋮
	HOST17	HOST27	HOST37	HOST47

3. Network Service Terminals User's Guide

3.1. General

Each node supports 4 Network Service Terminals. These can perform the following functions:

1) Data Echo
2) Data Sink
3) Data Source
4) Debug and Maintenance.

The user can establish a connection to one of the Network Service Terminals by means of the CALL-command. Their names are:

DECHn

DSNKn

DSCn

DMPn

where $0 \leq n \leq 4$ stands for the node number in which they reside.

3.2. Data Echo (DECHn)

DECHn will accept packets from all parties that are connected to it. It will send a RFNM for each incoming packet and return the packet to its source.

Only data-type packets are accepted, all others are removed from the system.

DECHn may be used to test logical full-duplex connections.

Examples:

```
DLE C
CALL: DECH0↵

76/05/21 11.07 CONNECTION ESTABLISHED TO DECH0
1234567890↵
1234567890
ABCDEFGH↵
ABCDEFGH
XXX↵
XXX
YYY↵
YYY
DLE B
76/05/21 11.07 CONNECTION TERMINATED
DLE C
CALL: DECH1↵

76/05/21 11.08 CONNECTION ESTABLISHED TO DECH1
PQRSTUVW↵
PQRSTUVW
ZZZZZZZ↵
ZZZZZZZ
010101↵
010101
DLE B
76/05/21 11.08 CONNECTION TERMINATED
```

Any number of users can be connected to DECHn simultaneously.

3.3. Data Sink (DSNKn)

Data Sink will accept data packets from all parties connected to it. It will send a RFNM for each incoming packet and then remove the packet from the system.

DSNKn may be used to test logical simplex connections.

Examples:

```
DLE C
CALL: DSNK0↵

76/05/21 11.08 CONNECTION ESTABLISHED TO DSNK0
1234567890↵
ABC↵
DEF↵
1↵
02020202020202020↵
DLE B
76/05/21 11.08 CONNECTION TERMINATED
DLE C
CALL: DSNK1↵

76/05/21 11.09 CONNECTION ESTABLISHED TO DSNK1
BLA BLA BLA↵
HA HA HA↵
PI PI PI↵
0123456↵
DLE B
76/05/21 11.09 CONNECTION TERMINATED
```

DSNKn can sustain any number of connections at a time.

3.4. Data Source (DSCn)

Data Source is used to generate traffic to a terminal or Host computer. The first packet the user sends to DSCn, after connection setup, specifies the number of lines DSCn is to generate. All output lines are identical except for the first character that cycles from "0" through "9". If the number of lines specified is zero DSCn will generate output until the connection is terminated or a new specification is given. All user input is taken to be a new specification of the number of output lines, and overrides any existing specification.

DSCn can sustain 5 connections at a time, and will terminate a connection if it has been inactive for about 5 minutes.

Examples:

```
DLE C
CALL: DSC0↵

76/05/21 11.09 CONNECTION ESTABLISHED TO DSC0
3↵
1ABCDEFGHIJKLMNOP
2ABCDEFGHIJKLMNOP
3ABCDEFGHIJKLMNOP
8↵
1ABCDEFGHIJKLMNOP
2ABCDEFGHIJKLMNOP
3ABCDEFGHIJKLMNOP
4ABCDEFGHIJKLMNOP
5ABCDEFGHIJKLMNOP
6ABCDEFGHIJKLMNOP
7ABCDEFGHIJKLMNOP
8ABCDEFGHIJKLMNOP
100↵
1ABCDEFGHIJKLMNOP
2ABCDEFGHIJKLMNOP
3ABCDEFGHIJKLMNOP
4ABCDEFGHIJKLMNOP
5ABCDEFGHIJKLMNOP
6ABCDEFGHIJKLMNOP
2↵                    <-- new specification overrides old one
7ABCDEFGHIJKLMNOP
2ABCDEFGHIJKLMNOP
DLE B
76/05/21 11.10 CONNECTION TERMINATED
```

3.5. Debug and Maintenance (DMPn)

DMPn is used to inspect and/or modify memory locations in a node. Its use is restricted to authorized personnel only. The first input line after connection setup must specify the password. If the password is not correct DMPn will terminate the connection. After the password is accepted the user can ava il himself of the following commands:

Note: n, m, xx stand for unsigned octal integer numbers. The last location whose contents are listed is called the current location. DMPn operates with octal numbers only.

Ln↵	List the contents of memory location n
Ln, m↵	List the contents of memory location n through m. Note: n m
Pxx↵	Put xx in the current location and list the next one.
	List the contents of the next location.
•+n↵ •-n↵	Close the current location and open the one whose address is equal to current location ± n.
•↵	List contents of current location again.
/↵	Use the contents of the current location as address of the next one to list.

DMPn will only accept carriage return (↵) as legal break character. If the user tries to open a location that is outside the memory range, or give a command that is unknown, DMPn will answer with "?".

DMPn can sustain one connection at a time and will terminate a connection if it has been inactive for about 5 minutes.

All output from DMPn has the form:

Octal address/contents

Examples:

```
DLE C
CALL: DMP0↵

76/05/21 13.32 CONNECTION ESTABLISHED TO DMP0
                ←——— password
L100↵
000100/045063 L105,107↵
000105/146173
000106/054405
000107/046005 .↵
000107/046005 .+10↵
000117/045046 .-10↵
000107/046005 L25200↵
025200/060064 /↵
060064/060104 L70000↵
070000/177777 P0↵
070000/000000
070001/177777 .-1↵
070000/000000 P177777↵
070000/177777
070001/177777 DLE B

76/05/21 13.33 CONNECTION TERMINATED
```

4. System Structure

4.1. As indicated in 1.3. the system consists of a set of independent software modules that communicate with eachother by means of a queuing structure. Special monitor-calls have been added to the Sintran-III-C monitor to put elements into queues and remove elements from queues. All queue-heads are located in a common data area that all modules have access to. All modules are independent real-time programs which run asynchronously, each at its assigned priority. Modules can activate eachother by the RT-monitor call to the Sintran-III-C monitor. The Sintran-III-C monitor takes care of CPU-allocation according to task priority, the saving and restoring of work registers, interrupt handling and periodic execution of periodic tasks. For a description of the Sintran-III-C monitor we refer to the:

Sintran-III-C CODE SYSTEM USER'S GUIDE.

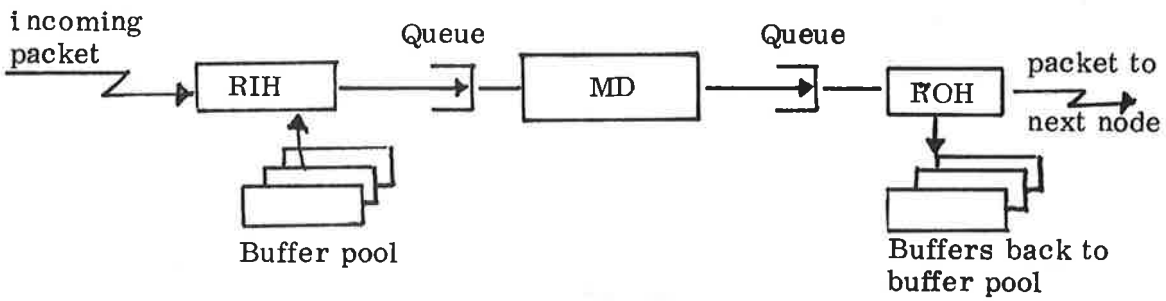
Principle of operation

The system consists of a set of independant software modules. These modules cooperate with each other by passing packets to each other by means of a queueing system. Each module performs the processing it is meant to do and then passes the packet to the next module. When all is done, the packet is removed from the system.

F.i. a data packet on transit through a node.

- The packet is received by the Remote Input Handler (RIH) that buffers the packet in buffers obtained from the free buffer pool.
- If the packet is received without errors, the packet is passed on to the Main Dispatcher (MD).
- The Main Dispatcher decides that the packet is on transit and determines where it should be sent next.
- The packet is passed on to the Remote Output Handler (ROH), with an indication on which line it is to be sent.
- The Remote Output Handler transmits the packet over the indicated line, and returns the buffers to the buffer pool.

In picture:



4.1.1. Queues

The queues used in this system are one-way linked lists which link fixed sized memory blocks into queues and/or into packets if a packet cannot be contained in one block. One fixed memory block is hereafter called a buffer.

Queue Structure

All queues consist of a queue head and queue elements if not empty. A queuehead consists of two consecutive memory locations, where the lower one points to the last element in the queue, while the second one points to the first element in the queue.

F.i.

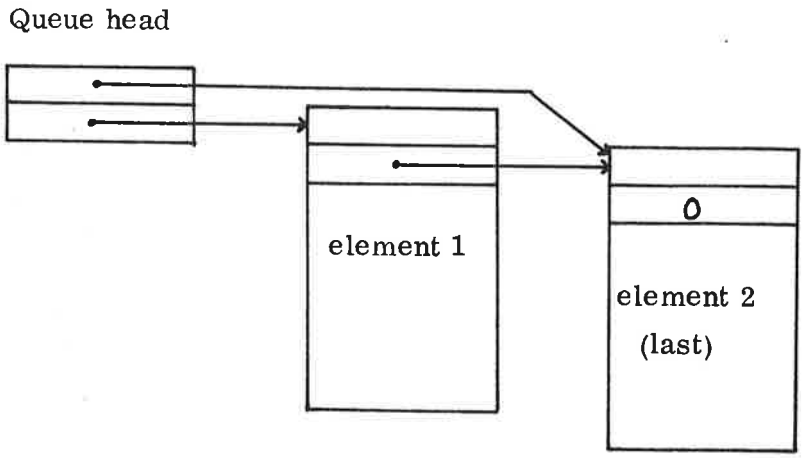


Fig. 4.1.1. Queue structure.

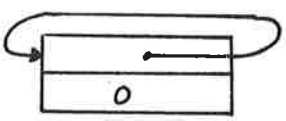
The address of the queuehead is the lower address of its two memory locations.

The elements in the queue are linked together by means of their second location, which points to the first location of the next element in the queue. This queue-link location should be equal to zero if the element is the last one in the queue.

Empty Queue Head

The first location of an empty queuehead points itself, the contents of the second location is zero.

Empty queuehead:



Queue Elements

Queue elements consist of one or more fixed sized buffers linked together by the first location of each buffer.

The contents of the first location of the last buffer is zero.
The first location of a buffer is called the extension buffer link.

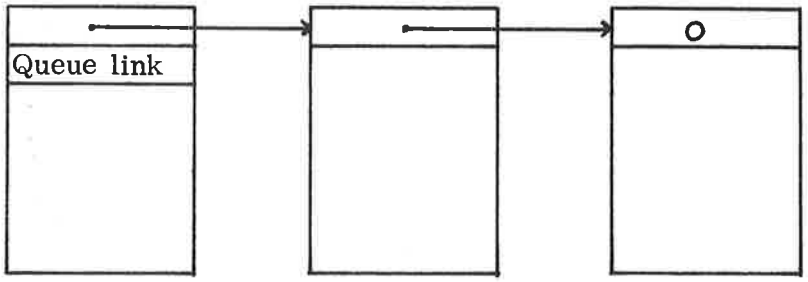


Fig. 4.1.2. Extended buffer

Free Buffer Pool

In the free buffer pool all unused buffers are linked together. Programs obtain buffers from the free buffer pool when they need them. After use the buffers are returned to the free buffer pool so that other programs may use them.

The free buffer pool has a structure that is different from ordinary queues. It is depicted below:

Pool head

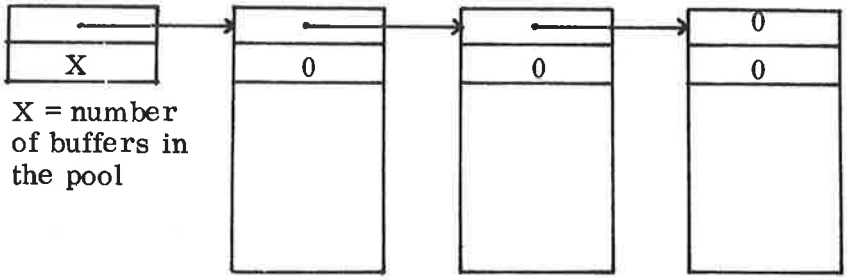


Fig. 4.1.3. Free buffer pool

The free buffer pool head consists of two consecutive memory locations. The lower one points to the first location of the first free buffer, that points to the second buffer, and so on.
The second word of the free buffer pool head contains the number of buffers in the pool.

A program may request buffers from the pool one at a time. Several linked buffers may be returned to the pool in one go.

The present buffer size is $16_{(10)}$ words, but can easily be changed if necessary.

4.1.2. Queue Handling Monitor Calls

The following monitor calls have been added to implement the queue structure:

1) Mon 170 = GET element from queue or buffer from the free buffer pool

Before monitor call: X-reg = queue head address or pool head address.

Non skip return: Queue or pool empty
Skip return : X-reg unchanged
A-reg = address of element taken from the specified queue or address of buffer taken from the pool.

NOTE: GET takes from the front of the queue (nearest the queue head.)

2) Mon 171 = PUT element in queue or put buffer(s) back in the free buffer pool.

Before monitor call: X-reg = queue head address or pool head address
A-reg = element address or first buffer address.

Non skip return : Error in queue structure or wrong A-reg or X-reg.
Skip return : O.K.

NOTE: PUT appends at the end of the queue (farthest from the queue head).

3) Mon 172 = PRIPUT, put element into a queue according to its priority.

The element is put into the specified queue, after the last element that has the same priority as the element to be added. For the prioroty specification see 4.2.1.

Before monitor call: X-reg = address of queue head
A-reg = address of element

Return conditions same as for PUT.

4.2. Packet description

4.2.1. Packet Header

All packets, except some internal communication system packets, have a packet header.

The packet header consists of 4 words that may each contain several fields. Their placement in the header and meaning is explained below:

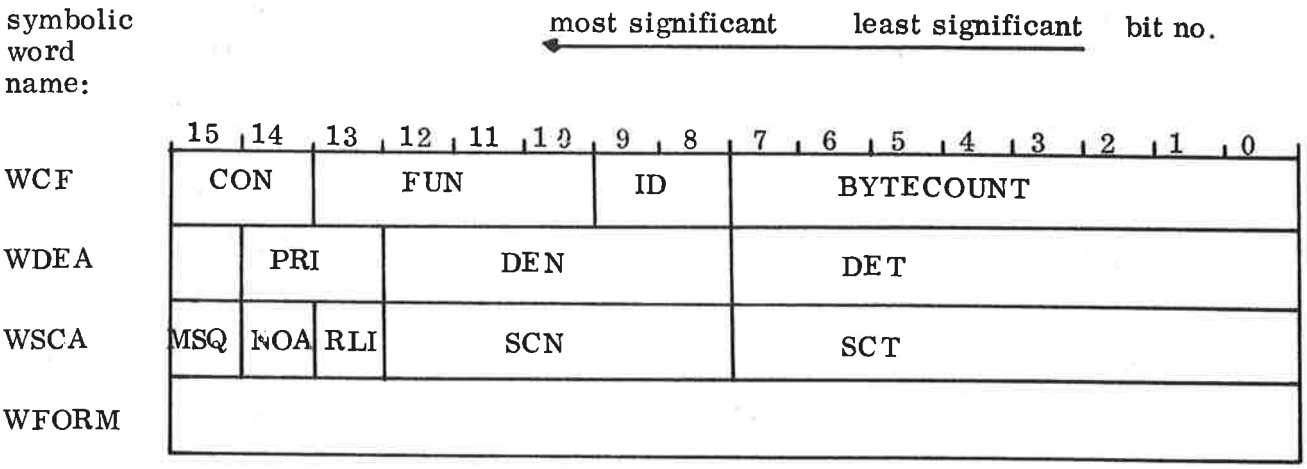


Fig. 4.2.1. Packet header

CON FUN	Control level Function } denote packet type, see
BYTECOUNT	Number of data bytes in the packet (header not included. i.e. max 255 data bytes in a packet).
ID	Sequence count for end - end duplicate control.
DEN DET	Destination node no. Destination terminal no.
SCN SCT	Source node no. Source terminal no.
PRI	Packet priority on transmission lines (3 = Highest. 0 = Lowest)
RLI	Reserved for Remote load information.
NOA	Reserved for NO - acknowledge (end to end)
MSQ	Reserved for message sequence
WFORM	General purpose word, its use depends on the packet destination.

4.2.2. Packet Placement in Pool Buffers

Internally in a node packets are placed, built or modified in buffers obtained from the free buffer pool.

The first three words of the first buffer are reserved for internal administration. The packet header is placed in the fourth through the seventh word of the first buffer. Packet data immediately follows the fourth word of the header. The first word of each extension buffer is reserved for the extension link.

I.e.

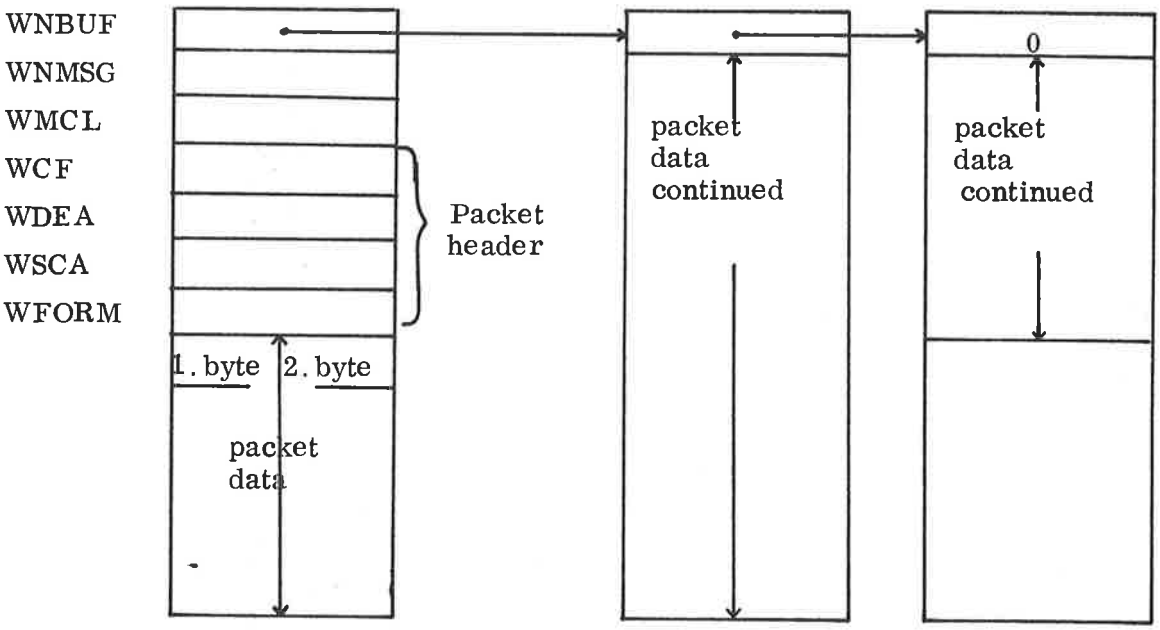


Fig. 4.2.2. Packet placement in buffers

The first seven words of the first buffer have symbolic names and are used as follows:

WNBUF	Pointer to first extension buffer, zero if no extension buffer
WNMSG	Pointer to next element in the queue, zero if last in queue or not in queue at all
WMCL	Contents denote packet age in the node
WCF-WFORM	The four words of the packet header.

4.2.3. Packet Type Description

The packet type is given by the CON and FUN fields in the packet header.

The CON-field is used to denote packet control-level.

CON = 0 Low level conversational control terminal to terminal packets

CON = 1 High level conversational control terminal to terminal packets specifying remote echo/break control. Not yet implemented but reserved for future use.

CON = 2 Terminal to Connection Monitor (or vice versa) control

CON = 3 Connection Monitor - Connection Monitor control

The FUN-field is used to specify the function of the packet.

FUN	CON = 0	CON = 1	Bytecount
0	DATA PACKET		$0 \leqslant \text{Bytec} \leqslant 377_8$
1	RFNM-PACKET		0
2			
3	NOT CONNECTED ABNORMAL DISCONNECT		0

FUN	CON = 2	CON = 3	Bytecount
0	Connect request	Connect request	4
1	Disconnect	Disconnect	4
2	Terminal closed	Terminal closed	6
3	Node unreachable	Terminal unavailable	4
4	Terminal unavailable	Terminal unavailable	6
5	Terminal busy	Terminal busy	6
6			
7	Command rejected		4
10	Command illegal		4
11	Connect acknowledge	Connect acknowledge	4
12			
13	Overload (no buffers)	Overload (no buffers)	4
14			
15			
16			
17			

4.3. Main System Modules

The system can logically be divided into 5 main sections:

- utilities
- transmission system
- terminal system
- network service terminals
- network management system.

A short description of these sections will be given below. The transmission system is described in detail in chapter 6.

4.3.1. Utilities

The following modules are considered utilities:

Main-Dispat cher	(MD)
Local-Dispatcher	(LD)
Clock-program	(CLCK)
Initialisation	(INIT)
Restart program	(RESTA)
Network Connection Monitor	(NCM)

4.3.1.1. Main-Dispatcher

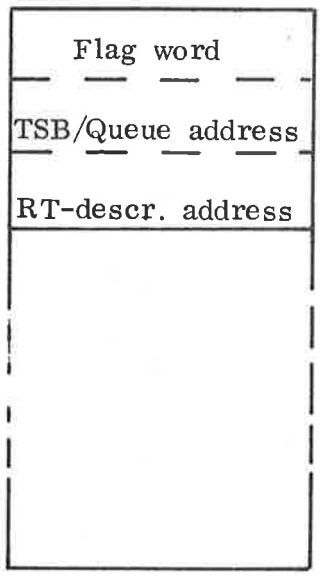
All traffic in a node passes through the Main-Dispatcher. The primary object of the Main-Dispatcher is to determine whether a packet is addressed to a terminal (logical, physical or program module) in this node or in a remote node. MD does this by comparing the DEN-field of each packet it takes from its queue (Main-Dispatcher Queue, MDQ) to a location in common core that holds the node number of this node (called GTHIS). If the DEN-field is equal to GTHIS the Main-Dispatcher puts the packet in the local Dispatcher Queue (LDQ) and activates the local Dispatcher. If the packet should be sent to another node the Main Dispatcher checks in a routing table called Minimum Line Table (see) to find the transmission line the packet should be sent on. The packet is then put, according to priority, into the transmission queue for that line (using PRIPUT). A time indication is put into the WMCL-word of the packet. If there is no minimum line (destination node unreachable), the packet is put into the Garbage Collection Queue (GARBQ) to be dealt with by the Garbage Collector ( ).

If the Main-Dispatcher encounters a packet with an illegal DEN-field (destination node no. greater than number of nodes in the network), it will send an error message packet to the System Command Message Decoder in node 0. The faulty packet is destroyed.

4.3.1.2. Local-Dispatcher

Only packets with a destination in this node pass the Local-Dispatcher. For each packet taken from its queue, the Local-Dispatcher uses the DET-field of the packet header as a table index to find the receiver's queue head address and the address of the RT-program description of the receiver. This RT-program description address is used as the argument of the RT-monitor-call (to activate the receiver). The table used is called ATERD (terminal description table) and has the following structure:

ATERD



One entry of three memory words per terminal (logical or physical).

1st word = flag word
2nd word = address of Terminal Status Block or address of Queue-Head (depends on flag word).
3rd word = address of RT-description.

For a full description see 4.4.

Fig. 4.3.1. Terminal entry table

In case of an illegal DEN-field the local Dispatcher will send an error packet to the System Command Message Decoder (SCMD) in node 0. The faulty packet is destroyed.

4.3.1.3. Clock-Program

The Clock-program is a program that runs every second and updates all time counters in a node. If a counter reaches zero the clock-program will activate the appropriate module. A counter whose contents is zero will not be modified until it is set to a non-zero value by its owner.

The Clock-program checks counters in the following order:

1. All Line Status Block counters (transmission system)
2. All Terminal Status Blocks (physical/logical terminals)
3. The Clock list.

4.3.1.4. Initialisation

The initialisation program (INIT) is run once for initial start and once for each restart. It performs the following actions:

- The free buffer pool is generated.
- Initiate the routing tables.
- Initiate the communication line drivers in Sintran-III-C.
- Initiate periodic execution of periodic tasks.
- Start the Terminal System (TS).
- Call the Host-initialisation routine.
- Start the Supervisor Command program (in node 0 only).

4.3.1.5. Restart

The restart program (RESTA) is run after the re start sequence of Sintran-III-C. It:

- clears all queue -heads (also pool-head)
- clears the routing tables
- clears all clock counters
- clears the work-fields of those modules that are sensitive to a restart
- clears the Terminal Status Blocks
- clears the Line Status Blocks
- calls the Host-clear routine
- activates INIT

4.3.1.6. Network Connection Monitor (NCM)

The Network Connection Monitor, one in each node, establishes and terminates connections. When a terminal user whishes to establish a connection, he executes the CALL-command. The CALL-command results in a Connection-request packet which is sent from the terminal to the NCM in the same node. The NCM then checks whether the wanted party is in the same node or elsewhere. If in the same node, the NCM investigates whether the wanted party is free and available or not. If free it is set occupied and the NCM sends an OK message back to the calling terminal and a "connected"-message to the called terminal. If not free the NCM sends a negative answer back to the calling terminal.

If the called party is not in the same node as the caller, the NCM will dispatch the connection request to the NCM in the node of the called party (if reachable). This remote NCM then investigates whether a connection can be set up, and sends the answer to the NCM in the caller's node. This last NCM then relays the answer to the calling terminal.

For some special logical terminals the NCM cannot decide whether the terminal is free or not. This is the case for logical terminals that can sustain multiple connections under the same name (net-address). These terminals will have a private built-in NCM that accepts and sends the same packets as the standard NCM. Examples of these terminals are Data-Source that can sustain 5 connections at a time, and Host-interfaces.

Other terminals are so simple, like Data-Echo and Data-Sink, that they can sustain any number of connections at a time. For these terminals the NCM always sends back OK-message to the caller, and nothing at all to the called terminal.

These differences, in respect to connection setup (and termination), are indicated by bits in the flag-word of the terminal description table (ATERD, see 4.4.).

These bits can indicate that a terminal, with respect to connection setup is:

- Standard, i.e. one connection at a time.
- Don't care, i.e. any number of connections at a time.
- Autonomic, i.e. has its own built-in "NCM".

4.3.2. The Transmission System

Because of its importance in a packet-switching network the transmission system is described in detail in chapter 6. In this section a functional description of its main modules is given.

The transmission system consists of 4 modules working closely together. They are:

Remote Input Handler	- RIH
Remote Output Handler	- ROH
Network Routing and Status	- NRS
Garbage Collection	- CGARB

4.3.2.1. Remote Input Handler

All input from the transmission lines into a node is received by the input driver routine that resides under Sintran-III-C. The input driver assembles incoming packets in buffers it obtains from the free buffer pool. The input driver also computes the Cyclic Redundancy Check (CRC). When a packet is completely assembled the driver passes it on to RIH with an indication whether it was received correctly or whether any errors were detected.

If the packet is not received errorfree RIH destroys the packet and all its buffers are returned to the free buffer pool.

If the packet is received without errors RIM accepts the packet.

Further action depends on the type of packet received.

Several packet types are possible:

- data packet
- link management packets
- network routing and status packets

Data packets are put into the Main-Dispatcher queue for treatment by the various other modules in the node, if the packet is not a duplicate. Duplicates can be generated in the following way:

Suppose node-A transmits a data packet to node-B and it is received by B without errors. Node-B will then send an acknowledge back to node-A. If the acknowledge gets lost node-A will retransmit the packet. Thus the same packet can be received more than once. However, RIH will detect whether a packet is a duplicate of a packet that has already been received correctly. Duplicates are not forwarded to the Main-Dispatcher but are destroyed. All incoming packets can carry "signals" acknowledgeing correct reception of packets sent in the opposite direction

RIH stores acknowledge information in the Line Status Block of the line concerned. This information is used by the Remote Output Handler to remove packets that are successfully transmitted to an other node.

Link Management Packets

Link management packets carry reception acknowledge information from the receiver to the sender. These packets are sent if there are no ordinary data packets to be sent. Otherwise the acknowledge information is carried by the data packets.

One special link management packet is of interest. It is called a Wait-Acknowledge packet and is used in the following way:

Suppose RIH in node-A cannot obtain enough buffers from the free buffer pool to accomodate a data packet coming from node-B. RIH in node-A will then cause transmission of a Wait-Acknowledge packet to node-B. The Wait-Ack packet instructs node-B to stop transmission of data-packets to node-A. B may continue transmission to A after a certain time or when it receives an ordinary link management packet from node-A.

Network Routing and Status Packets

There are two different Network Routing and Status packets called Hello-message and Hello-Acknowledge. They will be treated further in section 4.3.2.3.

When RIH detects a Hello-message or Hello-Acknowledge it will put it straight into a special queue and activate the Network Routing and Status program (NRS). These packets bypass the Main-Dispatcher because they do not have a standard packet header.

4.3.2.2. Remote Output Handler - ROH

The Remote Output Handler, ROH, regulates the transmission of all packets from a node.

But first we need a functional description of the Line Status Block. Each transmission line in a node has its own Line Status Block (LSB). In the description that follows the term LSB is understood to mean "the Line Status Block of a specific line."

The LSB contains information about the status of a line. I.e. whether it is down, OK, busy, idle etc. In addition to this it supplies storage cells for temporary information for use by RIH, ROH or NRS. It also contains the queue heads of the various queues of the line, i.e. one for data-packet, another queue for Routing and Status packets, etc. Statistical data about the use and performance of the line is also stored in the LSB. Now back to ROH:

Packets to be transmitted from a node to another are placed in the queues of the appropriate line. The Main-Dispatcher puts data packets in the Remote Output Queue (ROQ) of a line, while the Network Routing and Status program (NRS) puts Hello-message and Hello-Ack packets in the Hello-message queue of a line (HMOQ).

If a line is not busy, i.e. actually transmitting, ROH will inspect the queues of that line, and remove a packet if it finds one. ROH determines the packet type indication for the packet and passes the latter on to the output line driver under Sintran-III-C. ROH then marks the line as "busy" and continues with other work or lines. When the output-driver has transmitted the packet it signals output completion to ROH. ROH will remove the "busy" state of the line. The Cyclic Redundancy check for the packet is computed by the output driver.

After transmission data-packets are placed in the "retransmit"- queue, where they will stay until they are acknowledged by the receiver, or until a certain interval has elapsed (time-out). When a transmitted packet is acknowledged it is removed to the buffer pool. If an acknowledge is not received the time-out will occur and the data-packet is retransmitted. After a number of unsuccessfull retransmissions of the same packet the line is declared "dead", and recovery action is initiated by the Network Routing and Status program.

In the LSB ROH finds information about packets received by RIH that should be acknowledged. Outgoing acknowledgement information is carried by data-packets. If no data-packets can be sent the acknowledgement information is sent by ROH in Link-management packets. Also Routing and Status packets carry acknowledgement information.

4.3.2.3. Network Routing and Status Program - NRS

The Network Routing and Status program, NRS, is responsible for spreading routing and status information through the entire network.

NRS has three main functions:

- transmission line initiation
- recovery after line failure
- relaying status and routing information to neighbour nodes.

Transmission Line Initiation

Initially, after node program load or restart, all lines connected to that node are in the "dead" state. After a certain interval (silent period, see 6.6). NRS will cause periodic transmission of Hello-messages on all lines. The neighbour nodes will answer these Hello-messages with Hello-Ack packets, while they themselves will be transmitting Hello-messages. When the sequence described below is completed for a line, that line is declared OK and opened for traffic.

Each Hello-message and Hello-Ack carry a number, HM-number, ranging from zero to 13.

Every time a node receives a Hello-message it will answer with a Hello-Ack, carrying the same HM-number as the received Hello-message. During the initiation sequence two neighbour nodes start an internal counter at zero and emit Hello-messages with the HM-number equal to the contents of that counter. The counter is incremented for each Hello-Ack received with correct HM-number.

When the counters in both nodes reach 13 the connecting transmission line is declared OK. If during the initiation sequence a Hello-Ack is received with an incorrect number, or if no Hello-Ack is received when the next Hello-message is due for transmission the counter is reset to zero.

After the line is declared OK the HM-number is kept at 13 and Hello-messages are sent at a lower rate. The line will be declared dead again of no Hello-Ack is received for a certain number of Hello-messages in a row.

Line Recovery Action

When a line dead condition is detected (too many retransmissions or no Hello-Ack's) the node that detects the condition enters the Silent-period for the failing line. The Silent-period guarantees that the node on the other side of the line also detects the line-down condition. After the Silent-period has elapsed the messages in the retransmission queue for the line are moved back to the transmission queue. All acknowledgement information in the Line Status Block is set to its initial state and the initiation sequence is started.

Relaying Status and Routing Information

Hello-messages carry a "picture" of the network as the sending node sees it. The picture consists of the distance, in number of lines to cross, from the sending node to all other nodes in the network. The picture is built from the information in the routing tables (see). When a node receives a Hello-message NRS updates the routing tables with the information in the Hello-message. In this way information about status changes in the network is distributed through the entire network by means of the Hello-messages. For a full description see

4.3.2.4. Garbage Collection - CGARB

When a transmission line goes down the transmission queue of that line may contain packets for transmission over that line. All new traffic will be routed away from the failing line by the Main-Dispatcher, if an alternative route exists. If not the packets are placed in a garbage -queue.

The Garbage Collector inspects the transmission queue of lines that are not OK and moves packages found there to the garbage queue. It will then periodically inspect the garbage queue to see whether any of the packets there can be rerouted. If they can be rerouted the packets are placed in the transmission queue of the appropriate line. Packets that grow too old (ca. 2 minutes) while waiting for rerouting are destroyed by the Garbage Collector. If the over-aged packet is a data-packet the Garbage Collector sends an "abnormal disconnect" packet to the source of the over-aged packet. All other over-aged packets are destroyed without error indication.

4.3.3 The Terminal System - TS

The Terminal System, TS, consists of a set of reentrant subroutines used by all terminal programs. Each physical terminal connected to a node has its own Terminal Input Program (TIP), Terminal Output Program (TOP) and Terminal Status Block (TSB). The Terminal Status Block contains status information, the queue-head of the terminal output queue and provides storage space for the reentrant subroutines. The different TIP's and TOP's have a short initiation code sequence unique per terminal. After initiation all programs share the aforementioned reentrant subroutines. The TIP and TOP of each terminal share a semaphore for ordering critical sequences.

4.3.3.1. Terminal Input Program - TIP

The control flow of a TIP is regulated mainly by its state-variable that contains the current and the previous state. Four states are distinguished:

state 0 - IDLE
" 1 - ECHO
" 2 - TRANSPARENT
" 3 - COMMAND

All input from a terminal is stored in a unique input ringbuffer per terminal. This is done by the terminal driver under the Sintran-III-C operating system.

A terminal's TIP reads characters from the input ringbuffer. During the ECHO-state the characters are stored in buffers obtained from the free buffer pool, in such a way that TOP only needs to add the header when sending the packet later on. The buffers are linked to the Terminal Status Block and echo is generated by TOP. During the ECHO-state TIP will contintue to read from the ringbuffer until:

- the ringbuffer is empty
- a break character is found, TIP sets TRANSPARENT-state
- a command is found, TIP sets COMMAND-state.

When a break character is encountered TIP will set the TRANSPARENT-state, signalling to TOP that the packet is ready for sending. TOP will set the input state back to IDLE when the packet is sent.

During the TRANSPARENT and COMMAND states TIP will read one character at a time and pass it on to TOP for processing. TIP will then wait for a signal from TOP before reading the next character from the ringbuffer. (TOP may have changed the input state inbetween).

Transitions between input states take place as depicted in fig. 4.3.1. An explanation of the state graphe is given below.

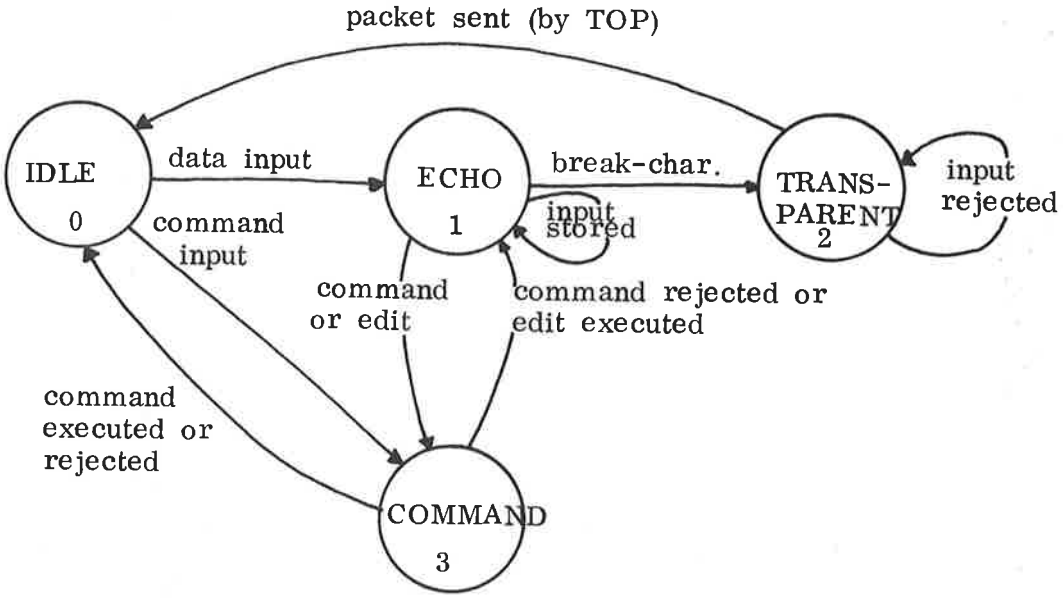


Fig. 4.3.1 Terminal System input state graphe

State 0 - IDLE

The IDLE state can be entered from all other states, except ECHO. Only when TIP is in the IDLE state is TOP allowed to start output of a output packet. The input state will change from IDLE to ECHO at the first character input of a packet, if that input is not "DLE+character". The input state will change to COMMAND after input of "DLE+character".

State 1 - ECHO

The ECHO state can be entered from the IDLE or from the COMMAND state.
During the ECHO state TOP is not allowed to start output of new packets arriving at the terminal. TOP will instead output echo of the characters stored by TIP. The transition from state ECHO to TRANSPARENT is made when TIP detects a break-character.

State 2 - TRANSPARENT

During the TRANSPARENT state all input is passed on to TOP, until TOP has sent the completed input packet. When the input packet is sent TOP will set the input state to IDLE.

State 3 - COMMAND

The COMMAND state is entered from the IDLE or the ECHO states on input of "DLE+charecter". All input is passed on to TOP. When TOP has executed or rejected the command it will set the input state back to its previous state, or to IDLE, depending on the command. The COMMAND state is also used for the edit functions Cancel and Back-space.

Some Additional Comments

The parity of incoming characters is ignored and stripped off before they are stored in the packet buffers. All output is generated with even parity.

The currently used character set is standard ASCII (CCITT nr.5) and no code conversion takes place on input or on output. Only capital letters are recognized in commands and command arguments.

4.3.3.2. Terminal Output Program - TOP

The TOP's perform a number of functions. The main functions are:

- generate echo output of input-characters
- output packets to the terminal, send RFNM
- establish and break connections
- send input packets and accept RFNM
- execute commands and edit functions
- regulate input/output on the terminal
- accept and decode system management packets.

Because of its many functions the flow control of a TOP is more complicated than of a TIP. Each TOP has an output-state variable that can indicate three distinct states. These are:

- IDLE
- MWAIT, waiting for RFNM
- COMMAND, waiting for answer to connection request.

The control flow of a TOP does not only depend on the output-state, but takes in account the input-state of the companion TIP as well. A TOP may change its own state and the state of its companion TIP.

A simplified state graphe is shown in fig. 4.3.2.

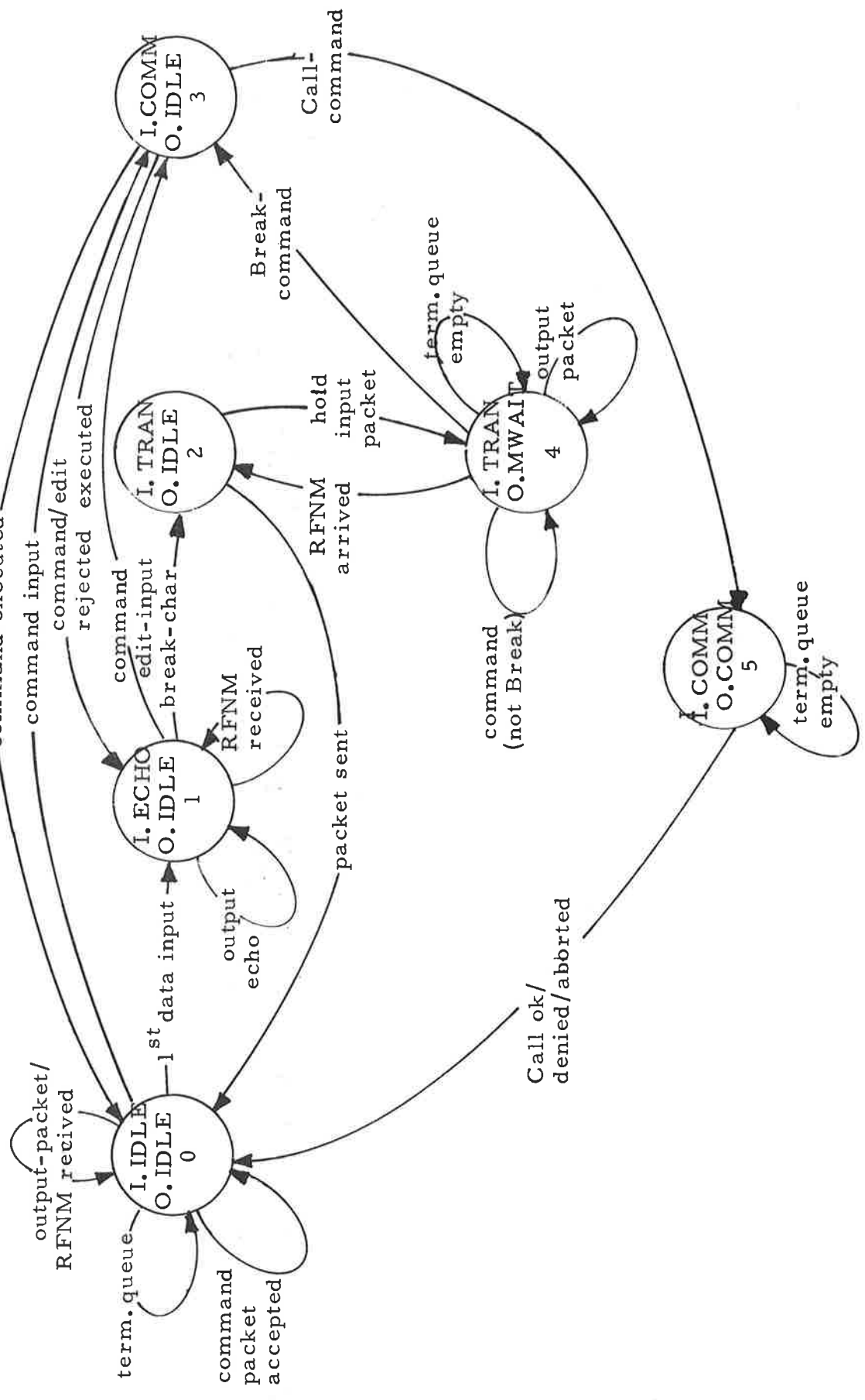


FIG. 4. 3. 2. TOP state graphe (simplified)

Explanation of the TOP State Graphe

The initial TOP state is 0, i.e. INPUT.IDLE + OUTPUT.IDLE. Data input will change the state to 1 (I.ECHO + O.IDLE) while during state 0 TOP will check the terminal output queue for output packets, RFNM or command packets. Any of these are accepted (if source OK) and after appropriate action TOP will revert to state 0 again.

If input of a COMMAND carries the state to 3, (I.COMM + O.IDLE) the command is executed or rejected and the state reverts to 0. If the command was a "CALL" command (DLE C) TOP state changes to 5 (I.COMM + O.COMM). This state (5) is solely used to await an answer to the connection request sent as a result of the CALL-command. During state 5 TOP will ch ck terminal input for a break-command (DLE B) and the terminal queue for an answer to the connection request. When the answer arrives the appropriate message is written on the terminal (OK or denial) and the state reverts to 0.

If TOP was in state 1 (I.ECHO + O:IDLE) and a break-character changes the input-state to TRANSPARENT, the TOP state will change to 2. Action there depends on whether a RFNM was received on the previous packet sent. If it was, the new packet is sent and the TOP state reverts to 0. If the RFNM was missing TOP state is set to 4.

During state 4 command input is accepted and the terminal queue is scanned for RFNM or data packets.

When the missing RFNM arrives the state reverts to 2, the waiting input packet is sent and the state drops to 0. If the missing RFNM is not forthcoming TOP will remain in state 4 until the connection is broken.

State 2 (I.TRANSPARENT + O.IDLE) is not stable. When a TOP enters this state it immediately checks whether the completed input packet can be sent. If so, the packet header is added and the packet is sent. The TOP then reverts to state 0. If the input packet must be held back, owing to a missing RFNM, the header is added and the packet is removed from the TIP, to await sending later on.

Some Additional Comments

To prevent exclusion of output by continuous input, or exclusion of input by continuous output a certain ordering is necessary. After sending a packet during state 2 TOP will first check whether a packet is present in the output queue before checking new input. After packet output (does not change TOP-state) TOP will recognize input-state changes before checking for more packet output.

During packet output all characters (except CR) are treated alike. TOP outputs the exact number of characters given in the Bytecount field in the packet header. All output characters have even parity.

Acceptance of arriving packets

Not all packets arriving at a terminal are accepted. Depending on the tests described below some may be rejected, with or without error action. The algorithm is given in pseudo-code and uses the following terms:

SCA = packet source address
CON = packet CON-field (0=data/RFNM, 2=control packet)
FUN = packet FUN-field
ID_M = packet ID-field (end to end duplicate control)
ID_S = sent packet sequence count
ID_R = received packet sequence count
CONN = address of connected party

```
IF terminal not logged in OR not connected THEN
  send abnormal disconnect
  RETURN
END IF

IF CON=0 OR =1 THEN
  IF SCA=CONN THEN
    IFpacket=abnormal disconnect THEN
      take down connection
      RETURN
    END IF
    IF packet=RFNM THEN
      IF ID_M ≠ ID_S THEN
        throw packet away
        ELSE
        accept packet and index ID_S (modulo 4)
      END IF
      RETURN
    END IF

    IF packet=data packet THEN
      IF ID_M ≠ ID_R THEN
        throw packet away
        ELSE
        accept packet, send RFNM with ID_M=ID_R
        index ID_R (modulo 4)
      END IF
      RETURN
    END IF

    ELSE
    packet source is not ok, send abnormal disconnect
    and throw packet away.
    RETURN
  END IF

  ELSE

  IF packet=control packet from NCM THEN
    accept packet if compatible with the terminal status,
    ELSE  throw it away
    RETURN
  END IF
END IF
```

4.3.3.3. Terminal Status Block - TSB

The Terminal Status Block provides storage cells for the reentrant routines of the Terminal System and contains the terminal status. Each terminal has its own status block. A description is given below:

word no. (octal)	name	function
0	WDFLG	status flag word bit 15 = 1 terminal connected " 14 = 1 terminal in command mode " 12 = 1 terminal logged off
1	WTFLG	input flag word bit 15 = 1 RFNM received " 14 = 1 input packet completed
2	WCONN	Connection address if WDFLG bit 15 = 1
3	WSCT	Internal TS-terminal number
4	WTIOW	Clock word for TOP
5	WITIM	Clock word for TIP
6	WIMOD	Input-status
7	WOMOD	Output-status
10 11	WETPQ WSTPQ	Output queue head
12-17	 WCHI WCHO WMLI WMLO	Terminal statistics no. character input no. character output no. packets in no. packets out + 2 words not yet used
20	WALT	Alternative terminal address (if ≠ 0)
21	WTIP	RT-description address of TIP
22	WTOP	RT-description address of TOP
23	WDEV	Logical Sintran-III-C device number for this terminal
24	WMESS	Pointer to current output packet
25	WIMSG	Pointer to current input packet
26	WIMCA	Pointer to completed input packet if WTFLG bit 14 = 1.

word no.	name	function
27	WILAST	Last input character
30	WIDR	Received packet sequence count
31	WIDS	Sent packet sequence count
32	WCURS	Cursor position on terminal
33	WICHAR	Character from TIP to TOP if input status is TRANSPARENT or COMMAND
34	WIYOUR	"YOUR-TURN" switch between TIP and TOP
35	WISW	Critical section switch. If $\neq 0$ it forces TIP to reserve the shared semaphore before testing the input status.
36	WEDIT	EDIT mode ON/OFF
37	WKSW	Output packet Kill-switch
40	WLFSW	Linefeed switch
41-46	WIAR	Cells for monitor call arguments for TIP
47-60	WOAR	Cells for monitor call arguments for TOP and temporary storage
61	WDLE	Current command character
62	WCAN	Current "cancel" character
63	WBSP	Current Backspace character
64	WBELL	Bell character
65	WLSIZ	Terminal line size
66	WFILR	Filler word, no. of fillers after CR and LF
67-72	WSCR1 WSCR4	4 scratch words used by TOP
73	WCARF	Carrier fail count
74-104	Stack	Stack used by TOP

4.3.4. The Network Service Terminals

4.3.4.1. Data Echo - DECH

Each node in the network has an echo process that generates RFNM on incoming packets and returns the packets to the sender. Only data-packets (CON, FUN = 0, 0) are accepted, all others are destroyed.

4.3.4.1. Data Sink - DSNK

Each node in the network has a data-sink process that generates RFNM on incoming packets. The incoming packets are destroyed.

4.3.4.2. Data Source - DSC

Each node in the network has a data-source process. A DSC can sustain five connections at a time, and will generate packets to the connected parties.

At connection setup time DSC obtains a buffer from the free buffer pool to store status information about the connection. Depending on the specification DSC will send a specified number of packets over a connection, or will send until the connection is taken down. For each RFNM received DSC will send a new packet to the connected terminal.

DSC also accepts commands from TEST, a module in node 0, used for generating test traffic. These commands are sent to DSC on a special queue, DSCC-queue (Data Source Control queue).

DSC accepts the following command from TEST:

CON, FUN = 0, 0; DEA = DSCCn; SCA = TEST

D2 = target terminal address (TT)

D1 = 0 : setup connection to TT
1 : generate traffic to TT, D3 specifies number of packets (if ≤ 0 send until stop-command).
2 : disconnect all connections initiated by TEST

(D1 = packet data word 1, D2 = packet data word 2, etc.)

DSC will send the following responses:

CON, FUN = 0, 0; DEA = TEST; SCA = DSCCn

D2 = TT
D1 = 0, D3 = 0 : connection OK
= 1 : disconnected/connection denied
= 3 : abnormal disconnect
= 5 : DSC busy
= 6 : Time-out disconnect
= 7 : Already connected

D1 = 1, D3 = 0 : start command accepted
= 1 : not connected
= 2 : specified traffic is generated

DSC does not have a standard Terminal Status Block, but has a built in Network Connection Monitor function.

DSC will terminate connections if they have been inactive for about 5 minutes.

4.3.4.3. Debug and Maintenance Program - DMP

The DMP in each node can sustain one connection at a time. Connections are set up and terminated in the normal way, via NCM. DMP has a short Terminal Status Block where the 21_8 words have the same structure as the physical terminal status block described in 4.3.3.3. Those cells that are only used internally have another meaning. The commands implemented in DMP are sufficiently described in section 3.5.

4.3.5. Network Management Modules

Network management consists of four modules of which the first three exist only in node zero, while the fourth one is present in each node.

They are:

- System Command Processor, SCP
- System Message Decoder , SCMD
- TEST
- Statistics and Status Collector, CLCT (in each node)

Only a short description is given below, they are described further in chapter 7.

4.3.5.1. System Command Processor - SCP

SCP resides in node zero only and uses the console terminal (logical Sintran device number 1) as input/output terminal. Depending on the command entered SCP either executes the command, releases the console to another module, or relays the command to CLCT. The answer from CLCT is passed on to SCMD that writes it in "longhand" on the console terminal.

4.3.5.2. System Command Message Decoder - SCMD

The SCMD accepts packets from any source and outputs these in longhand on the console terminal in node zero. These packets carry a format number in the fourth header word, that determines the output text. Variable arguments to the output message are carried in the data part of the packet.

4.3.5.3. TEST

The TEST module in node zero is used to establish connections from a Data-Source module to any other terminal in the network (except another Data-Source). After the connection is established the console operator can specify the amount of test traffic that DSC has to generate on a specified connection. In this way test traffic can be generated between a Data-Source and the Data-Echo or Data-Sink modules.

4.3.5.4. Statistics and Status Collector - CLCT

The CLCT modules in each node provide the following services:

- Send transmission line- or terminal status to SCP in answer to a command packet from SCP.
- Send and/or clear statistics information in the Line Status Blocks and/or Terminal Status Blocks in a node. This is done in response to command packets from a statistics module that may reside in a Host.
- Update the local calendar (date & time) with information received from SCP.

4.4. Common Data Area

The common data area resides in memory between the Sintran-III-C operating system (lowest) and the first module of the Scannet system. The following tables and/or variables are if interest:

- Free Buffer Pool Head

  The first two cells of the common data area form the free buffer pool head.

- Global Terminal Name Table - ANAMT

  This table contains all the names usable in the "CALL" command. The position in the table relates to the node and terminal address (DEN, DET) of the addressed terminal (logical, physical or Host).

  Each entry in the table consists of 5 words, of which the first four words contain the terminal name in ASCII. The fifth word is used as a flag word; bit 0 = 1 means physical terminal.
  bit 1 = 1 means restricted call access,
  bit 2 = 1 means default line-feed off

- Queue Heads

  In the following part of the common data area reside all the queue heads and queue head pointers. These are grouped together to provide easy cleaning on system restart.

- Routing and Line Status Tables

  For a description of the routing and line status tables see section

After the Routing tables follow the Terminal Status Block for DMP and the work field for DSC. The part of the common data area described so far is practically identical for each node. The following part can be generated differently for each node:

- Node Configuration

  In this section of the common data area information is stored about the:

  actual node number
  number of terminals (logical + physical)
  number of physical terminals
  number of transmission lines.

This part also contains the Line Status Blocks and the physical Terminal Status Blocks.

- Terminal Description Table (ATERD)

  This table contains a description of each terminal (logical or physical) in the node.

  Each entry consists of 3 cells:

  word 1 : flag word
  word 2 : TSB or queue head address
  word 3 : RT-description address of owner program.

The flag word holds the following information:

bit 17: 0 = logical terminal, 1 = physical terminal
bit 16: 0 = don't need connect/disconnect packets, 1 = needs connect/disconnect packets
bit 15: 0 = NCM connects and breaks, 1 = has own "NCM"
bit 14: 0 = second word gives queue head, 1 = second word gives TSB address.

- Clock List

The Clock list contains pointers to all the clock words to be updated by the clock module. It also contains the RT-description addresses of the modules to be activated when a clock "runs down".

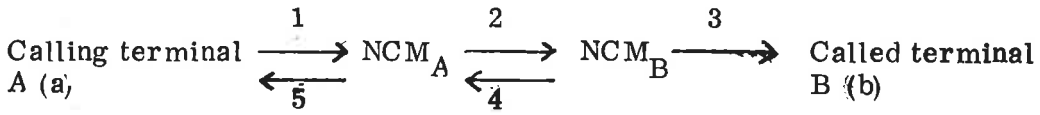
5. Connection establishment

5.1 Normal packet flow during connection setup

The packet flow between two terminals during connection setup is as described below. Assume that terminal a, in node A calls terminal b in node B, and that both terminals are "standard" with regard to connection setup

Before sending the call-request packet (1) terminal A (a) sets its busy flag in the Terminal Status Block to prevent a third terminal from interfering during the setup phase.

The packets exchanged during call-setup are:



where : 1 = Call Request, (CON, FUN) = 2, 0; DEA=A(0) ; SCA=A(a)
WD1=B(b), WD2= A(a)

(DEA = destination address
SCA = source address
WD1= data word 1
WD2 = data word 2
note that a NCM has terminal address zero in a node!)

2 = Call Request, (CON, FUN) = 3, 0; DEA = B(0) ;SCA=A(0)
high level WD1 = B(b) ; WD2 = A(a)

When NCM$_B$ receives packet 2, it checks the Terminal Status Block of B(b) to see if the called terminal is free. If it is free, NCM$_B$ sets it to busy and stores the callers address (WD2) in the third word of the TSB. NCM$_B$ then sends the two packets 3 and 4 that are:

3 = Call Request, (CON, FUN)= 2,0; DEA=B(b) ; SCA=B(0)
WD1=B(b) ; WD2 = A(a)

4 = Call Acknowledge (CON, FUN) =3,11 ; DEA=A(0) ;SCA=B(0)
high level WD1=A(a) ; WD2=B(b)

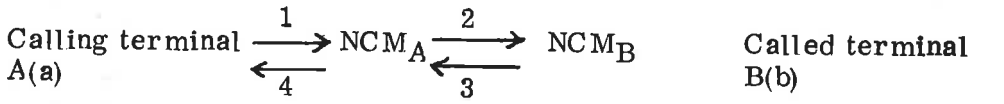
NCM$_A$ then turns packet 4 into packet 5 before sending it to the calling terminal A(a).

5 = Call Acknowledge (CON, FUN) = 2,11 ; DEA=A(a) ;SCA=A(0).
WD1=A(a) ; WD2= B(b)

When terminal A(a) receives packet 5, the connection setup is complete and data exchange may begin.

5.2 Called terminal is busy/unavailable

If the called terminal B(b) is busy or off-line, the packet exchange is as follows:



where 1 = Call Request (as in 5.1, packet 1)

2 = Call Request high level (as in 5.1, packet 2)

3 = Connection denied high level (CON,FUN)=3,x; SCA=B(0); DEA=A(0) WD1=A(a); WD2=B(b), [WD3=alternative]

4 = Connection denied (CON,FUN)=2,x; SCA=A(0); DEA=A(a) WD1=A(a); WD2=B(b); [WD3 = alternative]

The reason for the denial is indicated by the value x of the FUN field as described in 4.2.3.

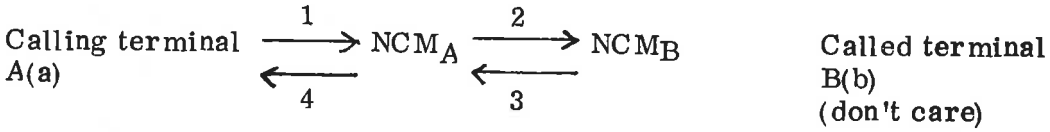
When terinal A(a) received a "connection denied" it will clear its own busy flag.

A terminal may have an alternative address. This is indicated by data word 3 (WD3) of a "connection denied" packet. - If WD3 is not equal to zero it contains the alternative address of the called terminal.

5.3 Called terminal is always available (don't care)

As described in sections 4.3.1.6 and 4.4 a terminal can be flagged as "don't care" in regard to connection setup (indicated by bit 16 of ATERD flag word).

If the called terminal B(b) is a "don't care" terminal, the packet exchange is as follows:



1 = Connection Request (as 1 in 5.1)
2 = Connection Request (as 2 in 5.1)

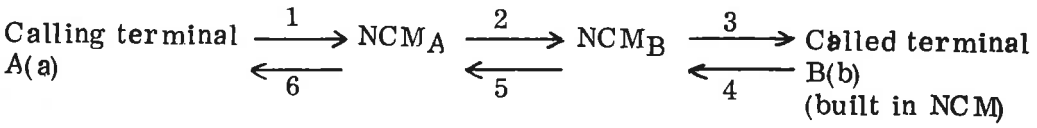
When NCM_B sees that B(b) is a "don't care", it immediately returns packet 3 to NCM_A. B(b) does not receive a connection request.

3 = Connection Acknowledge high level (as 4 in 5.1)

4 = Connection Acknowledge (as 5 in 5.1)

5.4 Called terminal has "built in" NCM

If a terminal has a "built-in" NCM, that means that the called terminal decides whether it will accept a connection-request or not. The package exchange will be:



1 = Connection Request (1 in 5.1)

2 = Connection Request (2 in 5.1)

3 = Connection Request CON,FUN=3,0; DEA=B(b) ; SCA=B(0)
WD1=B(b) ; WD2=A(a)

4 = Answer from called terminal
CON,FUN=3,X; DEA=B(0);SCA=B(b)
WD1=A(a); WD2=B(b)

X must be one of the legal values from the table in 4.2.3, but may not be zero (i.e. may not be "connect-request")

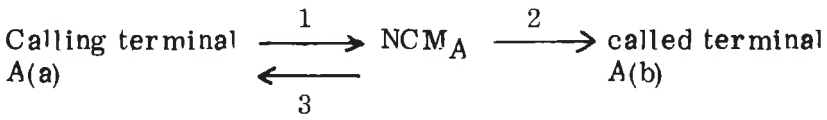
5 and 6 follow the normal pattern as described in 5.1 and 5.2.

5.5 Connections between terminal in the same node

If the calling and the called terminal resides in the same node, the examples given in 5.1 through 5.3 still hold if one drops the packet exchange between NCM_A and NCM_B and notes that NCM_B is equal to NCM_A in this special case.

f.i.

normal setup for terminals connected to the same node:



1 = Call Request (as 1 in 5.1)

2 = Call Request (as 3 in 5.1)

3 = Call Acknowledge (as 5 in 5.1)

5.6 Two special cases

If NCM$_A$ is asked to setup a connection to a terminal in node B and there is no transmission path available to node B, NCM$_A$ will turn the request down.

If node B is reachable but the number of buffers in the free buffer pool in node A is too low, NCM$_A$ will deny the connection.

If the number of buffers in the free buffer pool in node B is too low, NCM$_B$ will turn down the request in a similar manner.

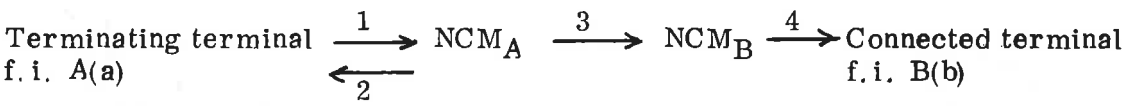
i.e.

1 = Connection Request (as 1 in 5.1)

2 = Connection denied (CON,FUN)=2,X ; DFA=A(a) ; SCA=A(0)
WD1=A(a) ; WD2=B(b)
where x = 3 if B unreachable
x =13 if buffer pool too low

5.7 Connection termination

Connections may be terminated by either party involved in a connection. Assume that terminals A(a) and B(b) have a connection and that A(a) wants to terminate the connection. The packet exchange during termination is as follows:



1 = Disconnect (CON,FUN)=2,1;DEA=A(0);SCA=A(a)
WD1=B(b);WD2=A(a)

2 = Disconnect (CON,FUN)=2,1;DEA=A(a);SCA=A(0)
WD1=A(a);WD2=B(b)

3= Disconnect
high level (CON,FUN)=3,1;DEA=B(0);SCA=A(0)
WD1=B(b); WD2=A(a)

4 = Disconnect (CON,FUN)=2,1; DEA=B(b);SCA=B(0)
WD1=B(b); WD2=A(a)

If node B is unreachable, packet 3 and 4 are dropped.

If the disconnecting terminal has a "build in" NCM, it should send a high level disconnect request to the NCM in its own node,

f i.

Terminating terminal $\xrightarrow{1}$ NCM_A $\xrightarrow{2}$ NCM_B $\xrightarrow{3}$ connected terminal
A(a) B(b)

1 = Disconnect
high level (CON,FUN)=3,1;DEA=A(0);SCA=A(a)
WD1=B(b);WD2=A(a)

2 = Disconnect
high level same as 1 but with DEA=B(0);SCA=A(0)

3= Disconnect (CON,FUN)=2,1;DEA=B(b);SCA=B(0)
WD1=B(b); WD2=A(a)

6. Communication system

In this section the various aspects of the communication system are described. The description is restricted to principles and functions rather than detailed implementation description.

6.1 Frame format

Data to be sent over a communication link is organized in packets. Packets themselves are contained within an envelope, suppled by the sending node, and stripped off by the receiving node. A frame is equal to the envelope with the contained packet.

The frame format is as depicted below, called Binary Syncronous control format:

SYN,SYN,DLE,STX,TLCL, .. .data bytes..,DLE,ETX,CRE1,CRE2,PAD

where SYN,DLE,STX,ETX are standard,8 bit, ASCII characters without parity.

Each frame starts with two SYN characters to obtain character synchronisation between the sender and the receiver. The start of the frame is defined by DLE, STX and the end of the frame is denoted by DLE,ETX, followed by two characters that contain the Cyclic Redundancy check, computed by the sender.

Depending on the frame type, denoted by the TLCL, data may be absent.

The frame contents inbetween DLE,STX and DLE,ETX are made binary transparent by duplicating every DLE character. The receiver strips off these extra DLE characters. In this way DLE can be sent as data, without the hasard that the combination DLE,ETX accidently occurs in the data part of the frame. I.e. an even number of DLE characters are accepted by the receiver as half so many data characters. An odd number of DLE characters must be followed by STX or ETX. If an odd number of DLE characters is followed by any other character than ETX,STX, the frame input is aborted.

The PAD character is necessary to ensure output of the last part of the CRC.

CRC1 is the most significant part, CRC2 the least significant part of the CRC.

The maximum frame size is 255 data bytes (inserted DLE's not included).

All bytes are transmitted with the least significant bit first; of a 16-bit memory word the most significant byte (bits 15-8) is transmitted first.

6.2 Transmission link control label

The first character following DLE, STX in a frame is called the Transmission Link Control Label (TLCL) and is part of the envelope. It contains the frame type and other information explained below. The TLCL is used exclusively for link-management and is invisible to the user.

The format of the TCLC is as depicted below.

most sign. 7	6	5	4	3	2	1	least sign. 0	
0	OETX	CHANNEL		OERX				DATA FRAME
1	0	0	0	OERX				HELLO-MESSAGE
1	0	0	0	OERX				HELLO ACK
1	0	1	0	OERX				GENERAL ACK
1	0	1	1	OERX				WAIT ACK
1	1	0	0	x	x	x	x	REMOTE LOAD

Fig. 6.2.1 TLCL structure

OETX = Tranmission channel odd/even bit
CHANNEL = Transmission channel number (0-3)
OERX = Receiver channel odd/even bits
x = Ignored bits

The different frame types and their function are explained below.

6.2.1 Data frame

The data-frame carries data-packets between two nodes in the network. The data part of the frame consists of the 8-byte packet header and the packet data as described in 4.2.1 and 4.2.2.

The most significant byte of a memory word is transmitted first.

The TLCL of a data frame also carries acknowledgement information about frames transmitted in the opposite direction.

The function of OETX, CHANNEL and OERX is explained in section 6.3.

6.2.2 Hello Message (HM)

Hello Messages carry routing and status information.

The HM-frame is depicted below.

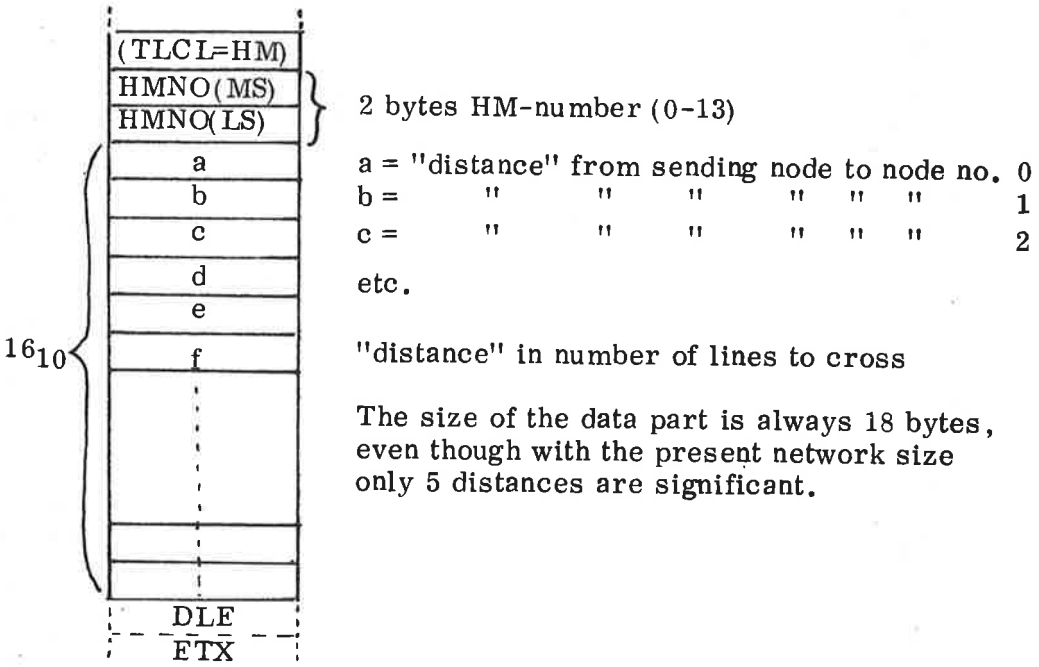


fig. 6.2.2 HM data part

6.2.3 Hello Acknowledge (HACK)

The HACK-frame acknowledges receiption of an Hello Message.

One HACK is sent for each HM received (if not silent-period).

The data part of the HACK-frame, depicted below, consists of two bytes that contain the same number as the HM that is being acknowledged.

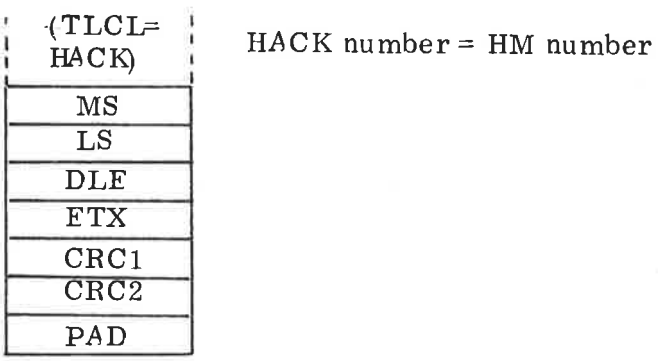


fig. 6.2.3 Hello Ack

6.2.4 General Ack (GACK)

The GACK frame serves two functions:

1. It acknowledges traffic in the opposite direction if no data frames can be sent from this node.

2. It signals the end of a busy condition (receiver pool low).

The GACK frame consists of the evelope only, and has no data bytes:

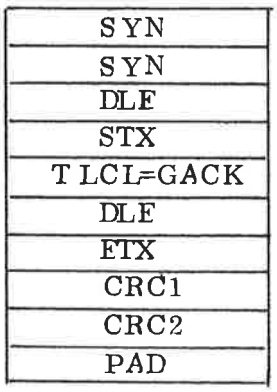


fig. 6.2.4 General Ack

6.2.5 Wait Ack (WACK)

The WACK-frame is used to signal a busy condition in the receiver of the node that emits the WACK-frame (not enough free buffers in the free buffer pool).

The transmitter of the node that receives the WACK-frame is instructed to stop transmitting data frames over the line on which the WACK was received. The busy condition is cleared by:

1. Reception of a GACK, or
2. by time out.

The node that received the WACK assumes that the busy condition is cleared after a certain time interval. If it is not cleared, the node will receive a new WACK after it has transmitted a data-frame.

6.2.6 Remote load frames (RL)

Remote load is not implemented in Scannet. The TLCL code, however, is presently reserved for future implementation.

6.3 Data-frame exchange mechanisms

A communication network must provide an efficient and reliable way of sending data through the network.

Ideally no data packets may get lost and data packets may not be duplicated. But the procedure used to gain these aims should not cause inefficient use of the communication facilities.

The below described procedures strike a good compromise between security and efficiency.

6.3.1 Channel selection

A condition for efficient use of a communication facility is that a number of data-frames may be sent in one direction, without waiting for the receiption acknowledgement of each preceding frame. This poses the problem of determining which data-frame(s) is (are) acknowledged by an incoming ACK.

To solve this problem, a communication line is divided into four logical channels. Each data frame is transmitted over a distinct logical channel and no new data frame may be transmitted over a channel if the previous one is not yet acknowledged (channel busy). This means that four data

frames may be transmitted before one has to wait for an acknowledge.

The logical channel number (0-3) on which a data frame is transmitted, is designated by the CHANNEL-field in the TLCL of the frame.

The transmission system selects the first free channel available when a data-frame has to be transmitted. If all channels are busy (unacknowledged), data frame transmission stops until a channel is acknowledged.

If a data-frame is not acknowledged within a certain time period, the frame is retransmitted, using the same channel as the original transmission.

6.3.2 Duplicate detection

Retransmission of data-frames after the retransmission timeout may lead to duplicate frames if the original frame was correctly received but the acknowledge was delayed or got lost.

To detect duplicate data-frames the following mechanism is used:

Each logical channel has a phase-bit that is odd or even. The phase-bit of the logical channel on which the data frame is sent is carried in the TLCL of that frame, and is called the OETX bit (see fig. 6.2.1).

If the receiver receives a number of consecutive frames on the same logical channel (of the same line) with the same setting of the phase bit (odd/even), then all frames are duplicates of the first one.

A data-frame retransmission after timeout always occurs on the same channel with the same phase value as the original transmission.

To keep track of the expected phase value of a channel the receiver maintains a "receiver odd/even" bit for each channel. Every time a frame is accepted, the "receiver odd-even" bit on the receiver node is flipped, thus ajusting the expected phase value of the next frame.

6.3.3 Packet acknowledgement / piggy backing

After the foregoing, packet acknowledgement is easy to explain. The receiver reports the expected value of the next phase bit for each channel back to the transmitter. From the value of these bits the transmitter decides whether the channel is acknowledged.

The phase values expected by the receiver are carried in the TLCL of all frames, except the Remote load frame, and are called OERX bits (fig. 6.2.1). Bit 0 belongs to channel 0, bit 3 to channel 3 etc.

If the transmitter decides that a channel has been acknowledged, the transmitter will flip the phase bit of that channel and mark the channel as being free. Thus the next frame to travel on the channel will have a different phase value.

F.i. data frame over channel 0:

Ncde A, Node B,

Assume before transmission:

$OETX_0 = 0$ $OERX_0 = 1$

Transmit frame on channel 0:

TLCL contains OETX=0, CHANNEL=0 → Receiver compares the OETX in the TLCL with the OERX(CHANNEL)bit. If they are not equal, the frame is accepted and OERX(CHANNEL)is flipped : i.e. OERX(CHANNEL)becomes 0.

If the bits were equal, the frame was a duplicate and is discarded.

Node B will now acknowledge the packet by reporting its OERX bits for all channels.

The transmitter in node A compares the $OERX_0$ of the TLCL to the $OETX_0$ of channel 0. If they are not equal, the channel is ack'ed and $OETX_0$ is flipped. ($OETX_0$ becomes 1). The ack'ed channel is released.

← TLCL with $OERX_0 = 1$

If $OERX_0$ of the TLCL is equal to the $OETX_0$, the channel is not ack'ed (or the channel is free).

Note that four channels (i.e. four data frames) may be acknowledged by the same TLCL. Note also that all frame types (except Remote Load) carry acknowledge information back to the transmitter.
The technique of carrying acknowledge information as part of a data frame in the opposite direction, is called piggy backing.

From the foregoing - it should be clear that the synchronisation of the OETX bits in one node and the OERX bits in the neighbour node is vital. If these bits get out of step, the receiver will only accept duplicates and throw away all originals. Therefore a node may never reset these bits to their initial state without making sure that its neighbour does the same. A node that resets a transmission line, enters the silent period for that line. This silent period is just long enough to guarantee that the neighbour node on the other side of the line detects a "line down" condition and also resets the line.

During line initiation the OERX bits in Hello Messages and Hello Ack's are ignored.

6.4 Routing and status

Network routing and status is maintainted in the routing tables in each node. The routing tables are updated every time a node receives a Hello-Message and also when a node detects a line-down condition.

6.4.1 Routing Tables

The routing tables consists of:

1. Net Status Table (ANST)

   The Net Status Table is a two dimensional table where ANST(i,j) gives the "distance" (in number of lines to cross) from this node to node j, when starting on line i in this node.

   All entries for j = "this node" are zero.

   If ANST (p, q) is greater than the total number of lines in the network, node q is not reachable from this node when starting out over line p.

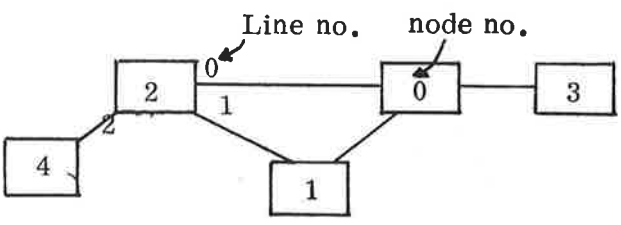
   If all entries ANST (i, p) $_{i=0,max}$ are greater than the max. number of lines, node p is totally unreachable from this node.

   Example:

   If all lines are up, the Net status table of node 2 of the below depicted network would be like:

ANST in node 2:

starting over line i → / to node j	0	1	2	3	4
0	1	2	0	2	3
1	2	1	0	3	3
2	3	3	0	4	1



From the Net Status Table the three other routing tables are derived.

2. Minimum Node Table (AMNT)

The Minimum Node Table gives the minimum "distance" from this node to all other nodes.

f.i. in node 2 of example above AMNT could be:

	to node i					
	0	1	2	3	4	
ANMT:	1	1	0	2	1	

if the distance is set greater than the max. number of lines in the network, the node is not reachable.

3. Minimum Line Table (AMLT)

The Minumum Line Table gives the starting line number of the shortest path from this node to all other nodes. I.e. the starting line number corresponding to the minimum distance given by the Minumum Node Table.

f.i. same example as above:

	minimum distance to node j occurs				
	j= 0	1	2	3	4
AMLT:	0	1	X	0	2

4. Node Dead Table (ADTB)

The Node Dead Table denotes which nodes are reachable and which are not. It is derived from the minimum node table and has the same form.

The node dead table is not used by the communication system, but is used by other modules in the system. F.i. NCM when processing a call-request.

	→ node j				
ADTB:					

ADTB (j) = 0:node j not reachable
ADTB (j) ≠ 0: node reachable

6.4.2 Use of the routing tables

The Node Dead Table is consulted by the Network Connection Monitor during processing of a call request.

The Minumum Line Table is used by the Main Dispatcher to rout outgoing packets directly to the line that gives the minimum distance to the destination node of a packet.

The Minumum Node Table is used by the Network Routing and Status modul NRS and is sent in the data part of Hello Messages to neighbours nodes.

The Net Status Table is updated, by NRS, from incoming Hello-Messages and new values for the other three tables are derived from it after each update.

6.4.3 Updating the Net Status Table

When node p receives a Hello Message on line i, the distances stated in the Hello Message are copied to $ANST(i,j)_{j=1,\ max}$ while adding one to each distance to account for the line between node j and its neighbour. $ANST(i,p)$ is kept at zero. After the copy the other three tables are updated.

If node p detects that its line i goes down, it will set

$ANST(i,j)_{j=i,\ max.} >$ max number of lines. ($ANST(i,p)$ is kept at zero)

During line down or line recovery the information in ANST for that line is not changed.

6.5 Line initiation / recovery

The procedure for initiating a line is described in section 4.3.2.3 and will not be repeated here. However, some points in connection to line failure detection and recovery may be clarified.

6.5.1 Line failure

Line failure is detected in three ways.

1. Too many retransmissions of a data packet without receiving an acknowledgement.
   The current max number of retransmission is set at five
   (i.e. one original + 5 retransmissions in all)
   The retransmission timeout period is given in section 6.6.

2. Too many Hello-Messages without Hello-Ack.

When a line is open for data traffic, a counter is incremented every time a Hello Message should be sendt. The counter is reset to zero for every Hello-Ack received.

If the counter exceeds a max. value, the line is declared down. Presently this value is set to 5. I.e. when the 6th Hello Message after the last received HACK is due, the line is taken down.

3. Wrong Hello-Message number

If a line is OK and NRS receives a Hello Message over that line with a Hello Message number that is not equal to 13 (max HM-number), the line is taken down immediately.

This situation can occur if a node has a "black-out" that last longer than the Silent Period ("black-out may be long power fail, or manual stop-continue). In such a case the neighbours of the node will have finished their Silent Period and may be trying to recover the line when the node comes out of its black-out.

6.5.2 Silent Period

During the Silent Period on a line all input is discarded and no frames are output to the line. The length of the Silent period depends on the line transmission-rate, max data-frame size and the max number of retransmissions allowed.

6.6 Timeouts

The retransmission timeout, Hello message frequency and Silent period all depend on the transmission rate of the lines and the max frame size. The timeout values are computed as follows:

6.6.1 Retransmission Timeout

When node A completes sending a data-frame to node B, B may just have started a max size frame transmission to A. This frame will not carry the acknowledge needed by A. If B puts the acknowledge to A piggy back on the next (max size) frame, A will not receive the acknowledge until after:

$T = 2 \times T_{max}$, T_{max} = max frame transmission time

The max frame size is 256 data bytes + 10 bytes evelope (if one forgets the double DLE's) = 266 bytes.

Tm ≈ 0.9 sec (2400 baud)
Tm ≈ 0.45 sec (4800 baud)

So the retransmission timeout should at least be:

$T_R > 1.8$ sec (2400 baud)
$T_R > 0.9$ sec (4800 baud)

The Timer Routine that checks timeouts runs once a second. This means that each timeout has a max uncertainty of one second, and average of 0.5 sec.

So practival values for T_R are

T_R = 3 sec. (2400 b)
T_R = 2 sec. (4800 b)

The present value of T_R in Scannet is 4 sec.

6.6.2 Hello Message Rate

The Hello-Message rate should be high enough to distribute routing information through the network, but not so high as to use a significant part of the transmission capacities.

The present rate in Scannet is once every four seconds for 2400 baud lines, but might be reduced to once every 2 seconds.

6.6.3 Silent Period

The time needed to detect a line down condition by means of retransmission or by means of missing Hello-Ack's, should be about equal. This time determines the Silent Period.

Data frames : One transmission + 5 retransm. gives 6 x T_R = 24 sec.

Hello Messages : 5 missing HACK's gives 6 x T_{HM} = 24 sec.
(5 + interval between last missing HACK and HM-due)

The present Silent Period in Scannet is set to 25 seconds.

6.6.4 Recovery time

After the Silent Period it takes at least 13 Hello Message exchanges before the line is declared OK again.

Recovery time : 13 x 2 = 26 sec.

NB. There is a fast and a slow HM-rate.
The slow rate is used when the line is OK.
The fast rate is used during line recovery.

6.7 Line Status Block

The structure of the Line Status Block is given below. Every line has its unique Line Status Block, used by RIH, ROH and NRS.

Word no:	name		function:
0	WFLG1		Dynamic status word
	- LOK,	bit 16	1 = line ok 0 = line not ok
	- LSD,	bit 15	Line "seamingly" dead Used when LOK = 0 and TYST = 0 LSD = 1 means HACK received with HM no $15_{(8)}$ but no HM received with no. 15
	- TYST,	bit 14	TYST = 1 denotes silent period for the line TYST = 0 not in silent period
	- LCLO,	bit 13	1 = line closed 0 = line not closed
	- LBUSY,	bit 12	1 = line busy, i.e. actually transmitting a frame 0 = line not busy
	- KONT,	bit 11	Used when LOK = 0, TYST = 0 KONT = 1 means HACK received on last HM
	- MMX,	bit 10	Used when LOK = 0, TYST = 0 MMX = 1 means HM received with max. number (i.e. 15_8)
	- TCA,	bit 9	1 = TYST Clock Activated. ROH has declared the line dead and initiated the silen period clock
	- LCHU,	bits 2-0	Last channel Used: bit 0-1 denote logical channel number bit 2 is cleared when LBUSY is set before transmitting a data-frame, and it is set when LBUSY is cleared after transmission done. Bit 2 is used to prevent ack'ing a message while it is being retransmitted

LSB - continued

Word no:	name		function :
1	WFLG2		Communication word between RIH and ROH
	- WCA,	bit 1	Wait Clock Activated, by RIH when received WACK. Cleared by timeout or by incoming GACK
	- TACK,	bit 2	Transmit Ack. Set by RIH to make ROH ack a received frame, either by piggy-back or by GACK
2	WOERX		ODD/EVEN Received bits + last received TLCL
	- ITLCL,	bit 16-8	Last received TLCL
	- OERX,	bit 3-0	OERX bits for logical channels (see 6.3.3)
3	WDIM		Temporary storage
	- OEACK,	bit 10-8	OEACK bits from TLCL in WOERX
4	WOETX		ODD/EVEN transmit bits + HM-number
	- HNO,	bit 8-4	Current HM-number when LOK=0 and TYST=0
	- OETX,	bit 3-0	ODD/EVEN bits for transmit channels
5	WBUSY		Logical channel state + HM error count
	- LAST,	bit 9-8	Last retransmitted channel (round robin)
	- HTEL,	bit 7-4	Hello Count, counts missing HACK's. If count exceeds max. number, the line is declared dead
	- BUSY,	bit 3-0	Channel Busy bits, 1= channel busy (i.e. not yet ack'ed)
6	WLIRET		Line dispatch address, when LBUSY = 1 WLIRET holds the return address of the sub-routine XMIT
7	WRTC		Value of retransmit time out period (constant)
8	WSTHM		Value of slow HM period (constant). Used when LOK = 1
9	WFTHM		Value of fast HM period (constant). Used when LOK = 0
10	WSIPC		Silent period duration (constant)

Word 7-10 hold time intervals in number of seconds, in 2's complements negative form.

11	WDIB	Logical Device Number for modem input device for this line
12	WDOB	Same for output device number
13	WRMSG	Address of received frame (RIH)
14-18	WFDS1	1st input descriptor used by RIH (see 6.9)
19-23	WDFS2	2nd input descriptor used by RIH
24	WTMSG	Address of frame to be transmitted, used by ROH
25-29	WDEST	Output descriptor used by ROH
30	WWCT	Clock for WACK timeout. Running if bit WCA in WFLG2 is set
31	WHCT	Clock for HM timeout (also used for silent-period)
32	WNORT	Number of retransmissions per channel. 4 bits per channel. The count for a channel is reset when the channel is allocated to a data-frame
33	WCHOP	Pointer to last message transmitted on channel 0, if channel 0 is busy
34	WCHOC	WCHOC is retransmission clock for channel 0, if channel 0 is busy
35-40		Same for channels 1-3
40-41	WHMQ	Queue head for HM/HACK queue
42-43	WRLQ	Queue head for remote load messages (not used in Scannet)
44-45	WRQ	Queue head for data-messages
46 to 66	WLSTAT to WTM1	Line statistics counters (see 6.8)
67	WNRSM	Message pointer used by NRS
68-69		Temporary pointers used by NRS
70	WWACK	Number of WACK's sent
71	WTTEMP	Temporary storage used by ROH

6.8 Line Statistics

The following statistical data is gathered in each Line Status block:

word no.	name	function
46-47		Double integer, accumulated byte-count from all sent packet heads Does not include retransmissions
48		Number of packets sent with priority 0
49-51		Same for priorities 1-3
52		Accumulated waiting time in WROQ of packets with priority 0 Seconds in bits 15-4 tick × 4 in bits 3-0 (80 ms)
53-55		Same for priorities 1-3
56		Number of retransmitted data packets
57		Number of data packets with CON, FUN not equal to 0,0 or 0,1 (also included in words 46-47)
58	WDIIE	Number of detected input driver errors Caused by input time-out, too long packet, lost synchronisation
59	WDIOE	Number of output driver errors Caused by output driver timeout
60	WCRCE	Number of input CRC-errors
61	WDUPL	Number of discarded duplicate data packets
62	WDISC	Not counted
63-64	WTBYT, WTBT1	Total number of bytes transmitted, includes envelopes but not inserted double DLE's Double integer (63 most sign., 64 least sign. part)
65-66	WTMO, WTM1	Total number of transmitted frames (packets), All included Double integer

6.9 Line driver interface

A special synchron-modem driver was developped for the Scannet software. The driver assembles incoming frames straight into buffers that it obtains from the free buffer pool. The input driver computes and checks the CRC and strips off the frame evelope. It also handles the "double" DLE's.

6.9.1 Input Driver

The following monitor calls are available for the input driver:

1. Total reset input

```
A : = "parblk"                  Integer parblk : = ("DEVno", "0",
mon  IOSET                                         "RTname", "-1")

return : A = 0  ok
         A ≠ 0  error
```

where DEVno holds the device number and RTNAME is the RT-program that has reserved the device

2. Enable frame input

Frame input is enables by passing a "descriptor" block to the driver. The driver can hold two input descriptor blocks at a time to prevent data overrun.

If the driver was not active at the time of the enable input monitor call, the driver is activated and starts searching for a SYN-SYN sequence.

If the driver was already activated, the descriptor address is merely stored.

The input descriptor consists of a block of 5 words that have the following functions:

word 0 : frame pointer, set by driver when frame completed (must be zero before mon-call)

word 1 : not used by driver

word 2 : event variable, denotes frame status (must be zero before mon-call)

word 3 : TLCL in right byte (set by driver)

word 4 : byte count of frame (count of data bytes between TLCL and DLE-ETX ; TLCL and DLE-ETX not included)

The following event variable values are returned:

EV = 0 frame not yet complete
1 frame input complete, CRC ok
EV 0 error during frame input

-1 = CRC-error

-2 = data overrun

-3 = no buffers available in the pool

-4 = synchronisation error or frame aborted by DLE- X , where X = not DLE and not ETX

-5 = too long frame, probably synchr. error or lost DLE-ETX

-6 = aborted because of reset mon call (IOSET)

-7 = input device time out, modem/interface/line malfunction

The descriptor block is given to the input driver in the following way:

Clear descriptor word 0 and 2

A : = Descriptor address

T : = Device number

mon 1

return : A = 0 already two descriptors in driver, this one is rejected

A ≠ 0 standard Sintran error code

skip return : ok, descriptor accepted, driver started if not already running

6.9.2 Output Driver

The following monitor calls are available

1. Initiation and reset

A : = "parblk"
MON IOSET

Integer parblk : = ("DEVnd", "1", "RTname", "function")

Return A = 0 ok
A ≠ 0 error

The value of the parameter function specify:

function =	-1	total reset output driver
=	0	no action
=	1	send 377_8 bytes inbetween data-frames
=	2	send ASCII SYNC inbetween data-frames
=	3	send EBCDIC-SYNC " " "
=	4	synchronize on ASCII-SYNC (26_8)
=	5	synchronize on EBCDIC-SYNC(62_8)

In Scannet the hardware is initiated to synchronize on ASCII-SYNC and to send EBCDIC-SYNC inbetween frames.

2. Frame output

Frame output is started by passing on output descriptor block to the output driver.

The descriptor block has the same structure as the input descriptor block, and the following words must be set before passing the descriptor to the driver:

word 0 : frame address

word 2 : event variable must be zero

word 3 : output TLCL in right byte

word 4 : frame byte count (number of bytes inbetween DLE-STX and DLE-ETX, i.e. TLCL included in byte count)

The following values for the event-variable are returned by the driver:

EV = 0 frame output not done yet

EV = 1 frame output complete, set when output driver starts on the DLE-ETX-CRC-CRC sequence. A new output descriptor may now be entered without disturbing the completion of the previous frame.

EV < 0 some error

- 1 = error in buffer link in frame, output aborted with DLE-blank

-2 = error in byte count, frame rejected

-6 = aborted by IOSET monitor call

-7 = output device timeout

The output driver can handle only one output-descriptor at a time. However, output completion is already signalled when the driver starts on the DLE-ETX-CRC-CRC sequence. The application program must therefore enter a new output-descriptor within the time it takes to output six bytes, if the output line is to be continually busy.

The descriptor block is passed to the output driver in the following way:

Set descriptor words 0,2,3,4

A : = descriptor address

T : = device number

mon 2

return : : A = 0, already descriptor in output driver, this one is rejected

skip return : Ok, descriptor accepted, output started

Note: The output driver will output SYN-SYN-DLE-STX before outputting the TLCL from the descriptor block. After the TLCL the data from the frame are output, starting with the packet header word zero, left byte. The last data byte from the frame (bytecount reacted zero) is followed by the DLE-ETX-CRC-CRC-PAD sequence.

f.i. Output GACK frame (envelope + TLCL only, no data bytes)

Descriptor:

word 0 : pointer need not be set if bytecount = 1

word 2 : EV = 0

word 3 : TLCL for GACK in right byte

word 4 : bytecount = 1

f.i. Output frame with 3 data bytes:

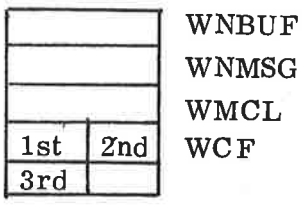
Descriptor:

word 0 : pointer

word 2 : EV = 0

word 3 : TLCL for data frame

word 4 : byte count = 4



6.10 Cyclic Redundancy Check

The CRC is a 16-bit remainder, computed by dividing the frame bits by the generator polynom $x^{16} + x^{12} + x^5 + 1$.

The computation is done by table lookup and takes about 40 µs per byte.

Of the generated CRC value the most significant 8 bits are sent first, immediately following the DLE-ETX.

The CRC is computed over all transmitted bytes, except the leading SYN-SYN and the CRC itself, i.e. DLE-STX, TLCL, "double"-DLE's + data ..... DLE-ETX.

7. System Management

7.1. System Management Modules

Four separate modules are available to perform system management functions. Three of these reside in node 0 only, while the fourth one is implemented in each node. The modules are:

- System Command Processor - SCP
- System Command Message Decoder - SCMD
- TEST
- Status and Satistics Collector - CLCT

SCP and TEST use the console terminal on node 0 (logical Sintran device 1) as input/output device.

7.1.1. System Command Processor - SCP

SCP accepts a number of commands from the console terminal in node 0. SCP indicates that it is ready to accept a command by writing "S" on the terminal. The following commands are available:

NST	report line/terminal status
TID	send data time to a node
TEST	activate test mode
TERM	switch to terminal module
EXIT	release console and exit.

NST - report line/terminal status

This command can be given in two forms; with one argument (Terminal name) for terminal status, and with two arguments (node and line number) for line status. If the first character of the first argument is a digit SCP assumes that line status is intended.

For execution of this command SCP despatches a packet to the CLCT module in the relevant node. The answer from CLCT is passed on to SCMD for output.

F.i.:

$ NST 0 3↵ - report line status of line no. 3 in node 0.

The answer is given as:

date/time ≪octal 1≫ LINE STATUS ⟨octal 2⟩

octal 1 is the Scannet address of the addressed CLCT module.
octal 2 is the contents of the first word of the Line Status Block of the line concerned. For the interpretation of this word see section

$NST TER1.0↵ report status of terminal "TER1.0". NB. The argument must be the name of a physical terminal.

The answer is given as:

date/time ≪octal 1≫ TERMINAL STATUS ⟨oct 2⟩ ⟨oct 3⟩ ⟨oct 4⟩

<oct 2>, <oct 3> and <oct 4> are the contents of the first three words of the Terminal Status Block of the indicated terminal.

Error Cases

If wrong arguments are supplied SCP answers with

?TERMINAL?	for illegal terminal name
?NODE ?	for illegal node number
?LINE ?	for illegal line number
?PARAMETER?	if wrong or missing parameter, or if CLCT has rejected the command packet.
NO ANSWER	if SCP has not received an answer from the addressed CLCT within 30 seconds.

TID n - send date and time to a node

This command takes as octal argument the number of the node whose date and time is to be updated. SCP sends a packet to CLCT in the addressed node containing the calendar and time setting in node 0. Seconds are ignored. This command should only be executed if the addressed node is reachable.

F.i.:

$ TID 2↵ - update calendar and clock in node 2.

TEST - activate test module

When receiving the TEST command SCP releases the console terminal and activates TEST. The commands for TEST are described in 7.1.3. Return to SCP is made from TEST with the "F" command.

F.i.:

$ TEST↵

ready sign from TEST module

TERM - switch to terminal mode

The console terminal can also be operated as ordinary Scannet terminal, with the functions described in chapter 2. To do this, SCP releases the console terminal and activates the TIP of logical device 1.

The console is returned to management mode by typing ↑F (CNTR F). The terminal must at that moment be logged off, and F must be the first input after ↑Q (logout).

Note

After start or restart both SCP and TIP "1" are activated. SCP reserves the console terminal and TIP "1" is waiting for it.

Thus the very first time after start or restart a slightly different procedure must be followed.

I.e.

$ EXIT↵ - Exit SCP, releases console

console now in terminal mode

↑F bring back SCP, release console from TIP "1".

EXIT - release console and exit

This command is used to make SCP release the console and exit. The console can then be used as Sintran-III terminal.

F.i.:

$ EXIT↵ - exit from SCP

(ESC) - enter Sintran-III-C

ENTER ↵

@ - Sintran-III-C command processor

The following sequence brings the console back to management mode:

@RT xxxxx where xxxxx = RT-description address of SCP

@LOG

date.time -EXIT- console released by Sintran-III-C

$ and back to SCP.

7.1.2. System Command Message Decoder - SCMD

The SCMD accepts packets from any source and outputs these in longhand on the console terminal (if it is not reserved). The packet format is as follows:

CON, FUN	= 0, 0
BYTECOUNT	= 2x (no. of arguments)
WFORM	= Format number (defines longhand message)
WD1-WDn	= argument of error message.

Those modules that can produce error packets but do not themselves have a node and terminal number use terminal number $377_{(8)}$ in the SCT field of the packet header. (All packets addressed to terminal DET = $377_{(8)}$ are put into the free buffer pool by the Local-Dispatcher).

The general output format of longhand messages is:

date/time ≪oct 1≫ "output message with arguments in octal."

<oct 1> is the octal representation of the source (SCN+SCT field) of the packet.

F.i.:

76/06/01 12.45<377>LINE 000000 OK

A complete list of all possible messages is given in chapter 8.

Note

SCMD can only output messages if it succedes in reserving the console terminal. SCP and TEST will temporarily release the console after output (or echo) of the following characters:

CR, LF, S,> .

The operator should therefore not give a "half" command to SCP or TEST, since SCMD is excluded until the operator types CR.

SCMD is excluded from using the console terminal while it is reserved by TIP "1". I.e. when the console terminal is in "terminal" mode all error messages queue up until the console is returned to "management"-mode. The same is true while the console terminal acts as Sintran-III-C terminal.

7.1.3. TEST

TEST is used to make DSC generate test traffic through the network. TEST sends command packets to a DSC to make it:

- establish a connection to a given terminal
- generate test traffic over the connection
- report completion to TEST
- take the connection down.

TEST can maintain 20_8 connections at a time. After connection setup, the connection is given a number ranging from zero to 17_8. The below described commands S and D refer to a connection by the assigned number.

Test is ready to accept a command when it has written " > " on the console terminal.

The following commands are available:

>C n, name↵

Connect DSCn to terminal "name". "name" may be the name of any terminal (logical, physical or Host) except DSC.

The answers to this command may be:

CON xxx OK - Connection established and number xxx is assigned to it

or one of the following messages:

?NODE - illegal node number

?TERM - illegal terminal name, or already connected through TEST

TABLE FULL - attempt to set up too many connections

DSC BUSY - the addressed DSC cannot take more connections.

>S n,m↵

Start test traffic on connection number n. DSC sends m packet if

if m > 0. If m ≤ 0 DSC will send until the connection is broken. (n and m octal).

As answers to this command one may expect:

N OK	where n = connection number
** ?	if command illegal or inconsistent with connection status.

>D n↵

Disconnect all connections from DSCn that are established on commands from TEST.

>L↵

List the connection table. For each connection one line of output is generated. The output is:

CON x1 FROM DSCn TO "name" STATUS = x2

x1 = connection number
x2 = connection status

0 = illegal
1 = doing connection setup
2 = connected OK
3 = starting test traffic
4 = started test traffic

Other output messages from test can be:

xxx	DONE	The specified test traffic is generated on connection xxx. Specify new test traffic.
xxx	DISCONN.	The connection xxx is disconnected by the terminal.
xxx	ABNORM. DISCONN.	The connection xxx is abnormally disconnected.
xxx	TIMEOUT	Connection xxx is taken down by DSC because of time out.
** ?		Unknown command

7.1.4. Status and Statistics Collector - CLCT

CLCT accepts command packets from SCP and from a statistics module that may reside in a host.

Command packets can specify that CLCT must:

- send line status
- send terminal status
- clear and/or send line statistics information
- clear and/or send terminal statistics information
- update the local calendar and clock (if node not 0)

All commands and responses are sent with CON, FUN = 0,0'!
In the responses the source and destination address of the command packet are swapped.

1. Send line status:

command	response
SCA = "SCP"	
BYTECOUNT = 6	BYTECOUNT = 14
D1 = -1	D1 = 0
D2 = line no.	D2 = unchanged
D3 = SCP sequence no.	D3 = unchanged
	D4 = line status

2. Send terminal status:

command	response
SCA = "SCP"	
BYTECOUNT = 6	BYTECOUNT = 14
D1 = -2	D1 = 0
D2 = terminal address (DEN,DET)	D2 = unchanged
D3 = SCP sequence no.	D3 = unchanged
	D4-D6 = term. status.

If a wrong line number or terminal address is specified the packet is returned with:

BYTECOUNT = 14
D1-D6 = unchanged.

3. Clear and/or collect statistical data:

Command packet:

SCA ≠ "SCP"
BYTECOUNT = 4
D1 = sequence no. ($0 \leq D1 \leq 377_8$)
D2 = command bits

The bits in D2 specify:

bit 0 = 1	- clear terminal data
bit 1 = 1	= send terminal data
bit 2 = 1	- clear line data
bit 3 = 1	- send line data

If bit 1= 1 then bit 3 is cleared before the command is executed.

If both "send" and "clear" bit are on,the data is stored in the response packet before the statistical areas are cleared.

Response packet if send line data:

BYTECOUNT = m, m = 2x (no. of lines $x21_{10} + 3$)

D1 = old D1 +400
D2 = interval start time (time of last "clear") See note next page:
D3 = interval stop time (time of collect)
D4 = Dx = statistical data, 21_{10} words for each line, starting with line 0, then line 1, etc.

The 21_{10} words of statistical data per line consist of:

word 1,2	: Number of data bytes sent (double integer)
words 3,4,5,6	: Number of packets sent for priorities 0,1,2,3.
words 7,8,9,10	: Accumulated waiting time for packets of priorities 0,1,2,3.
word 11	: Number of retransmissions
word 12	: Number of packets sent with CON, FUN ≠ 0,0 or 0,1
word 13	: Number of input driver errors timeout/too long packet/synchronisation error
word 14:	: Number of output driver errors Timeout.
word 15	: Number of CRC-errors on input
word 16	: Number of discarded duplicate packets on input
word 17	: Number of discarded packets
words 18,19	: Total number of bytes out,of all packets, all included (i.e. envelope, header, data, CRC)
words 20,21	: Total number of packets out, all included (data, Hello-msg, Hello-ack, GACK, WACK, etc.)

Response packet if send terminal data:

BYTECOUNT = m, m = 2x(number of terminals x6 + 6)
D1 = old D1 + 1000
D2 = interval start time
D3 = interval stop time
D4, D5 = total number of data bytes sent to logical terminals
D6 = total number of packets sent to logical terminals
D7 = Dx = terminal data, 6 words per terminal

The 6 words per terminal consist of:

word 1 : Number of characters in
word 2 : Number of characters out
word 3 : Number of data packets in
word 4 : Number of data packets out
words 5-6: Not yet used.

4. Command packet to update the local calendar and clock:

SCA = "SCP"
BYTECOUNT = 16_8
D1 = 0 or sequence count. The packet is ignored if D1 $\neq$ 0 and equal to the D1 of the previous clock setting packet.
D2 = seconds (ignored by receiver)
D3 = minutes
D4 = hours
D5 = day
D6 = month
D7 = year

Note:

The interval time coding in responses on command packet 3 are coded in the following way:

bit 15-11: day
bit 10-6 : hour
bit 5-0 : minutes.

8. Error messages

8.1 Scannet error messages on console terminal in node 0

All these error messages have the same basic format:

Date/time <source> message + arguments

<source> is the octal representation of the message source i.e. node number in bits 13-8, terminal number in bits 7-8.

(terminal no. 377 is a dummy number used by those modules that do not have a terminal number).

1. NODE ERROR n1 n2 n3 n4

Illegal DEN-field in a packet header detected by a Main-Dispatcher. n1 to n4 are the header words of the faulty packet. The faulty packet is destroyed.

2. TERM. ERROR n1 n2 n3 n4

Illegal DET-field in a packet header detected by a Local-Dispatcher. n1 to n4 are the header words of the faulty packet. The faulty packet i destroyed.

3. TERMINAL STATUS n1 n2 n3

Response to NST command in SCP. See 7.1.1.

4. LINE STATUS n1

Response to NST command in SCP. See 7.1.1.

5. LINE NR n1 DOWN (L-REG= n2)

Transmission time number n1 on the node that sent the message went down. n2 indicates the reason why .(too many retransmissions or no Hello-Ack compare n2 to the listing of the line- system to find out where the line closing routine was called from).

6. LINE NR n1 OK

The transmission line number n1 on the node that sent the message is opened for traffic.

NOTE

When a transmission line changes status (OK to DOWN or V.V.), the two nodes connected by that line will both send a message to node 0. However, if one of the nodes is isolated from node 0, its messages do not reach node 0 until the line is Ok again. If the node is isolated for a longer period than the garbage-collection time (ca. 2 minutes) the messages are destroyed.

7. UNKNOWN ERROR

SCMD received an error message with an illegal format specification in WFORM in the packet header.

8.2 Local node errors

Local node errors are reported on the console terminal of each node.

They consist of standard Sintran III-C messages.

Error 99 is generated by the Scannet software if it discovers a fatal error. The module that discovers the error is inhibited from further activation (RTOFF). It is recommended that one lists out the contents of the RT-description block of the inhibited module before restarting the system with

stop!
master clear
16!

The RT description address is given in the error message. F.i.

ERROR 99 IN xxxxx at ppppp

where xxxxx = RT-description address.

RUN TIME ERRORS

At run time, errors may be detected by the system. Most of the errors will cause the current RT program to be aborted and the error message:

aa.bb.cc. ERROR nn IN rr AT ll xx yy

will be printed on terminal 1.

The parameters have the following meaning:

aa.bb.cc - Time when the error message was printed.

aa - hours
bb - minutes
cc - seconds

nn - Error number. For further explanation, see list below.

rr - Octal address corresponding to an RT program name.

ll - Octal address where the error occurred.

xx, yy - Numbers carrying additional information about the error. One or both numbers can be omitted. For further explanation, see list below.

Example: @
01.43.32 ERROR 14 IN 16105 AT 114721;

Error number	Meaning	xx	yy	Program aborted
00	Illegal monitor call	RT prog.		yes
01	Bad RT program address	"		"
02	Wrong priority in PRIOR	"		"
09	Illegal parameter in CLOCK	"		"
10	Illegal parameter in ABSET	"		"
11	Illegal parameter in UPDAT	"		"
12	Illegal time parameters	"		"
22	False interrupt		level no.	no
23	Device error	hardware status	hardware device no.	"
24	Internal interrupt		bit no.	yes
26	Mass storage time-out		program	no
27	Error in CONCT	RT prog.		yes
28	FTN I/O	Error no.	(See NORD File system	
35	Stack error			"
38	Memory parity error	PES	PEA	no
40	Power failure			no
99	FATAL error in Scannet-software			program in RTOFF

NORSK DATA A.S.
Lørenveien 57 - Postboks 163, Økern
OSLO 1

COMMENT AND EVALUATION SHEET

SCANNET USERS GUIDE
May 1977

Publ. No. ND-60.087.01

In order for this manual to develop to the point where it best suits your needs, we must have your comments, corrections, suggestions for additions, etc. Please write down your comments on this pre-addressed form and post it. Please be specific wherever possible.

FROM _____

_____

_____