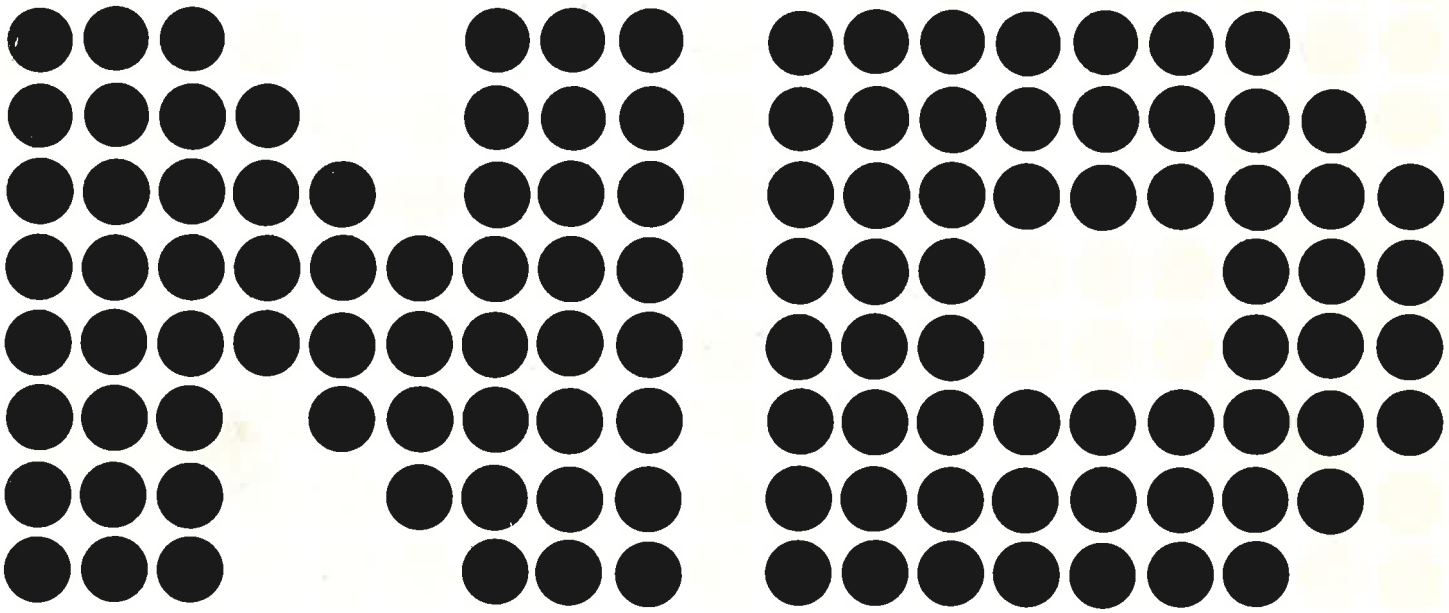
**NORDNET**

**System Documentation**

# NORSK DATA A.S



# NORDNET

## System Documentation

# REVISION RECORD

| Revision | Notes |
|---|---|
| | |
| 1/79 | New version replacing the original SINTRAN III/SINTRAN III Communication Manual |

ND–60.081.02
January 1979

NORSK DATA A.S
P.O. Box 4, Lindeberg gård
OSLO 10

# *TABLE OF CONTENTS*

# 1 INTRODUCTION

## 1.1 *THE PURPOSE OF THE SINTRAN III/SINTRAN III COMMUNICATION*

The SINTRAN III/SINTRAN III communication system is an optional part of the SINTRAN I/O system for communication between two or more SINTRAN III/10 or SINTRAN III/12 systems. The communication serves the following purposes:

a) Data transfer between two user programs, one in each SINTRAN III system. The two user programs may access the communication channels just like a local peripheral device or file.

b) Remote terminal communication. This means that a user sitting on a terminal connected to one of the SINTRAN III systems may run the operator communication system and background system on the remote SINTRAN III system.

c) Remote file access. A program on the local computer may open and access files sequentially on a remote computer just as if they where local files.

d) Remote load. The main memory of a remote computer may be loaded from a file on the local SINTRAN III system.

## 1.2 *THE COMMUNICATION LINE*

The communication is essentially the same, regardless of the communication line used. The line must fullfill the following requirements:

a) Logical full duplex connection. At least half duplex hardware is required.

b) Binary transparent transmission facilities on byte level.

c) Sufficient capacity for the actual load.

The communication line may be divided in up to 16 logical lines in each direction. The logical lines will hereafter be called channels, and the word line will be reserved for the physical communication line.

The channels may be used completely independent of each other, by different programs or one program may use several channels at the same time. To ensure that a channel is not used by two programs at the same time, a channel must be reserved by the program before it is used. Each channel is assigned a logical device number.

## 1.3 *COMMUNICATION PROCEDURE*

The SINTRAN III/SINTRAN III communication procedure is binary transparent on byte level. This means that any multiplum of 8 bits may be sent on a communication channel.

The vehicle for information exchange between the communicating SINTRAN III systems is the communication frame. The communication frame consists of a header (4 bytes), a data field (0 - 256 bytes) and a trailer (2 bytes). For some lines a sync/flag sequence is required in front of and after the frame.

The header contains sequence information for the frame and acknowledge (ACK) or not acknowledge (NAK) information for a frame received in the opposite direction. The header also contains a channel number and a frame type. A frame will contain data bytes for one channel only.

The data field may contain user data or system data (for example the name of a file to be opened) or it may be empty.

The trailer contains a 16 bit cyclic redundancy check for the whole frame.

Up to four frames may be sent in one direction before an ACK is required. In this way a full duplex line will be fully utilized, because the sender does not have to wait for acknowledge of one frame before the next one is sent.

# 2 GENERAL SYSTEM DESIGN

## 2.1 *HARDWARE*

Except for the line driver, all communication software is independent of communication hardware. For information on communication hardware equipment, the appropriate hardware manual and programming specification should be consulted.

## 2.1 *SOFTWARE*

### 2.2.1 *System Overview*

The system architecture is sketched on the figure on the next page. The part of the system below the dotted line is the "basic system" for exchange of data between programs in two SINTRAN III systems. Seen from the basic system, the remote terminal processors and the remote file processors act like ordinary user programs sending and receiving data on a communication channel.

The REMOTE-LOAD command is treated specially because it uses a simpler line procedure.

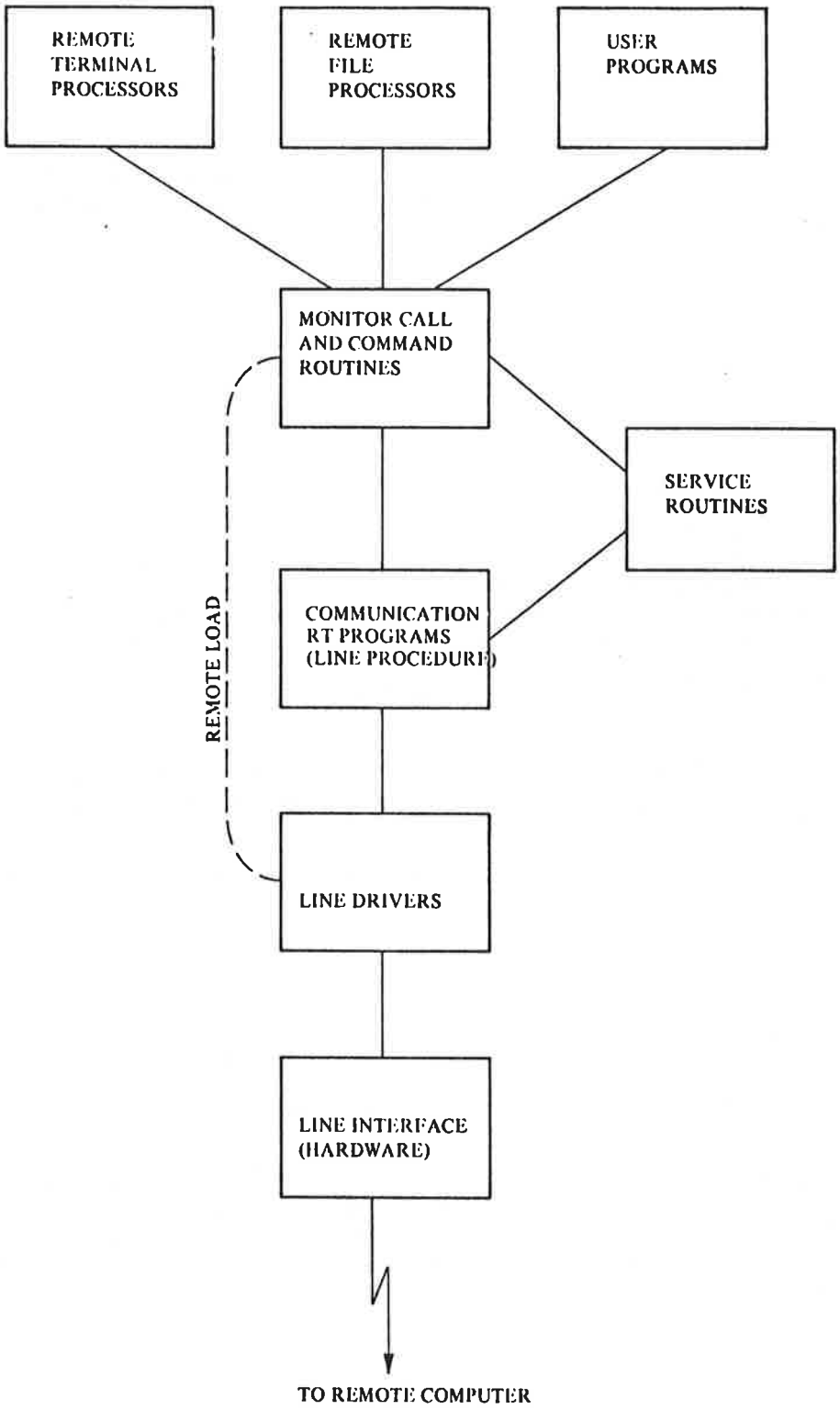
All buffer space needed by the communication system is allocated dynamically from a pool of free buffers and returned to the buffer pool when no longer needed.

The buffers are linked together to form queues. The buffer system is described in detail in Section 5.

### 2.2.2 *Line Drivers*

The task of the line output driver is to send a frame stored in one or more buffers to the line interface.

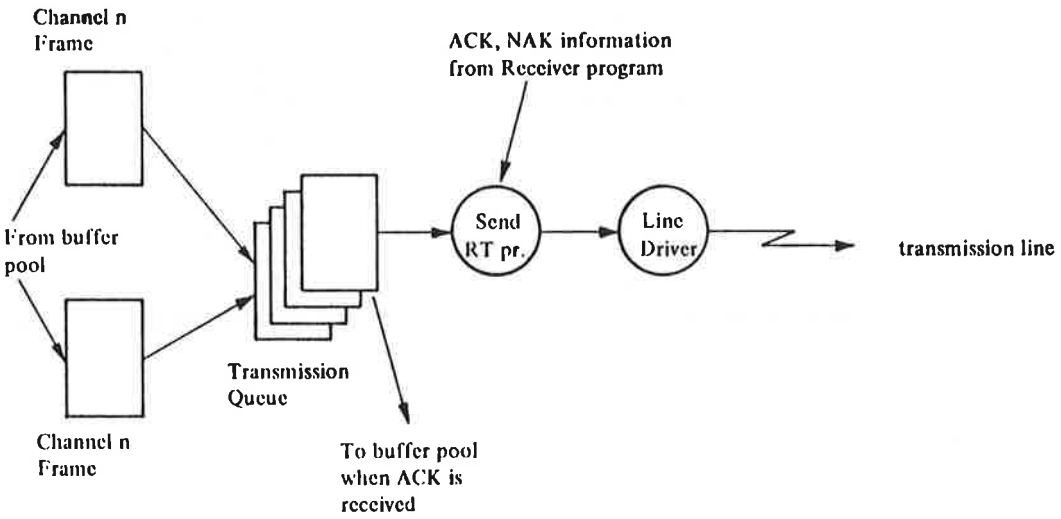
The task of the line input driver is to receive frames from the line interface and store them in buffers allocated by the Receive RT program (see Section 2.2.3). The input driver will always have buffer space for up to four frames. This is to ensure that no information is lost if the load on the receiving computer is so high that the Receive RT program cannot handle the input frames at the rate they arrive.


REMOTE TERMINAL PROCESSORS
REMOTE FILE PROCESSORS
USER PROGRAMS
MONITOR CALL AND COMMAND ROUTINES
SERVICE ROUTINES
COMMUNICATION RT PROGRAMS (LINE PROCEDURE)
REMOTE LOAD
LINE DRIVERS
LINE INTERFACE (HARDWARE)
TO REMOTE COMPUTER

## 2.2.3 *Communication RT Program*

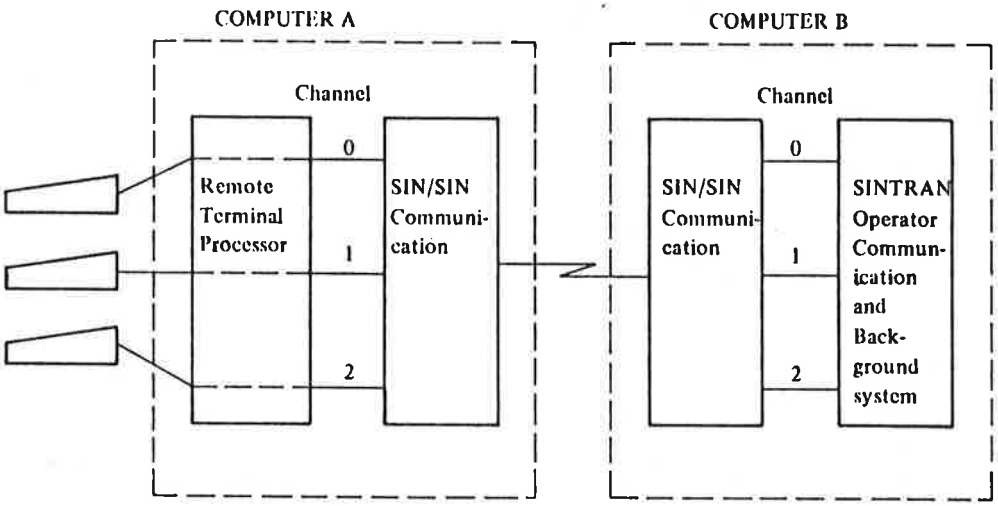
The heart of the communication software is two system real time programs, the Receive RT program, and the Send RT program. They are responsible for the communication procedure, such as sequence number generation and checking. ACK and NAK responses, retransmission of frames transferred in error, and routing of data to the destination program. The Send and Receive RT programs exchange information through a common data field.

The Send RT program fetches frames from the Send Queue, adds sequence information and ACK, NAK information for a frame received in the opposite direction, and initiates sending of the frame by starting the line output driver.



## 2.2.5 *Remote Terminal Processor*

The remote terminal processor is illustrated in the figure below.

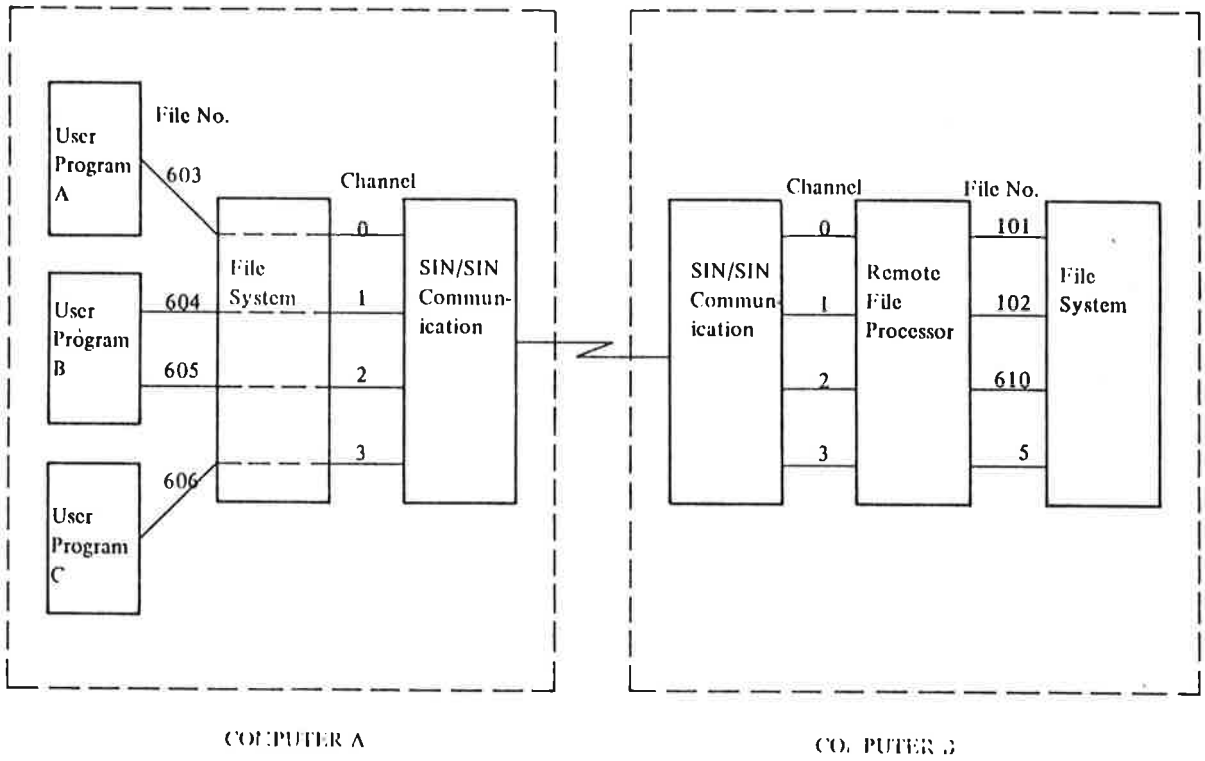



The main task of the Remote Terminal Processor in computer A is to copy the input given from the terminal to a communication channel, and to copy the output returned on the channel to the terminal.

Seen from the operator communication and background system in computer B, the remote terminals connected by the communication channels look like extra local terminals.

## 2.2.6 *Remote File processor*

The Remote File Processor is illustrated in the figure below.




The Remote File Processor in computer B is able to perform the following tasks:

- Open files on request from computer A
- Copy data from/to a file to/from a communication channel
- Close files on request from computer A
- Report errors in accesing a file to computer A

The Remote File Processor consists of one RT program for each channel.

The open file requests, close file requests, and error messages are sent as special frames on the communication line.

In computer A, the user programs may access the remote opened files by the ordinary INBT/OUTBT or MAGTP RFILE/WFILE monitor calls.

Error messages are returned to the user by no skip return or non-zero A register in the usual way.

# 3 FRAME FORMAT AND CONTROL PROCEDURE

## 3.1 *TRANSMISSION FRAME*

The vehicle for all information exchange on the communication line is the transmission frame. A transmission frame is a sequence of 7 to 264 contiguous bytes in the following format:

SS, H, I, CRC

where

| | | |
|---|---|---|
| SS | = | Synchronization Sequence |
| H | = | Header |
| I | = | Information Field |
| CRC | = | Cyclic Redundancy Check |
| CFL | = | Closing Flag (only HDLC) |

## 3.2 *SYNCHRONIZATION SEQUENCE (SS)*

This field will vary for various communication equipment.

For asynchronous modem and accumulator link (Teletype interface or terminal buffer) a single SYNC character ($28_8$) followed by a SOH character ($1_8$).

For HDLC the synchronization sequence consists of a "flag" ($176_8$).

The synchronization sequence is used by the receiver to synchronize on frame start.

Note that, except for HDLC, recognition of a synchronization sequence does not guarantee that the start of a new frame is received, because nothing is done to prevent that a synchronization sequence occurs in the H, I or CRC field. For HDLC the bit stuffing mechanism will prevent flags to be sent within the frame.

## 3.3 *HEADER (H)*

The header consists of four bytes with the following contents:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|
| | GS | | IS | | GR | IR | | STS | 1. byte |
| | Control | | | | Channel | | | | 2. byte |
| | Number of bytes in I–field | | | | | | | | 3. byte |
| | | | | | | | | | 4. byte |

where GS and IS identifies this frame,

GS = Group number for this frame

IS = Phase bit for this GS

GR, IR and STS gives status information for a frame transmitted in the opposite direction

GR = Group number

IR = Phase bit for GR

STS = Status

STS may have the following values:

1 — Not acknowledge

2 — Acknowledge

3 — Wait acknoledge

The control field determines the type oaf the frame; it tells the receiver how to interpret the I field.

The control field may have the following values:

0 — Data frame. Data in I field.

1 — Define break strategy. Break strategy in I field.

2 — Define echo strategy. Echo strategy in I field.

3 — Request for input. I field always empty.

4 — Turn off request for input. I field always empty.

5 — System configuration frame. Configuration table in I field (used for initialization).

6 — Load data. Load data in I field (used for REMOTE-LOAD).

7 — Remote file access frame. Further information in I field.

The channel field contains the channel number for this frame. It may be any integer between 0 and $37_8$.

The third and forth bytes of the header contains the number of bytes in the I field. It may be any integer between 0 and $400_8$. It is used by the receiver to determine the end of the I field.

The only frames that are not acknowledged in the ones where both the control field, the channel number, and the number of bytes in I field equals zero. These frames are sent only for the status information in the GR, IR and STS fields, and are not sequence numbered (the GS and IS fields).

## 3.4 *INFORMATION FIELD (I)*

The purpose of the SINTRAN/SINTRAN communication is to transfer the data contained in the I fields of transmission frames.

This field may contain any sequence of 0 — $400_8$ bytes in any code, or binary data.

## 3.5 *CYCLIC REDUNDANCY CHECK (CRC)*

The last two bytes of a frame check sequence (CRC). The CRC field serves to detect errors induced by the transmission link and validate transmission accuracy. The 16 bits result from a mathematical computation on the digital value of all binary bits in the H and I fields of the frame.

The process is known as cyclic redundancy checking.

The transmitter performs the computation and send the resulting CRC value. The receiver performs a similar computation and checks its results. The receiver discards a frame that is found to be in error and does not change the phase bit for this group (GS).

## 3.6 *CLOSING FLAG (CFL)*

The closing flag is only used on HDLC, and consists of the same 8 bit "flag" as the synchronization sequence ($176_8$).

## 3.7 *CONTROL PROCEDURE*

### 3.7.1 *General*

The communication procedure may be divided in four levels:

1. Hardware line procedure.
2. Communication link procedure.
3. Communication channel procedure.
4. User level procedure.

In a network of more than two nodes, the procedures 1, 2 and 3 are node to node procedures, while 4 may be an end to end procedure.

Note that in a SINTRAN/SINTRAN communication network, there is no difference between nodes and hosts. All nodes may be used as hosts and vice versa.

### 3.7.2 *Hardware Line Procedure*

The hardware line procedure depends on the communication hardware used and will not be discussed here. For information on this subject, consult the appropriate hardware manual.

### 3.7.3 *Communication Link Procedure*

The purpose of the communication link procedure is to detect and recover transmission errors, and to concentrate the data streams on the various channels to one data stream on the communication line.

Error detection and recovery is done by the first byte of the header field and the CRC field. The CRC check will detect frames transmitted in error, and the receiver will request a retransmission by sending a NAK status with a frame in the opposite direction. A frame will be retransmitted if a NAK status is received on that frame, or by timeout if an ACK status is not received within 6 seconds.

The GS and IS fields of the header is also used to detect frames totally missing and frames retransmitted by error (because an ACK is lost). The sequence numbering of the frames by the GS and IS fields is illustrated in the table below.

| *Frame No.* | *GS* | *IS* | *Response* |
|---|---|---|---|
| 1 | 0 | 0 | ACK |
| 2 | 1 | 0 | ACK |
| 3 | 2 | 0 | ACK |
| 4 | 3 | 0 | ACK |
| 5 | 0 | 1 | ACK |
| 6 | 1 | 1 | ACK |
| 7 | 2 | 1 | ACK |
| 8 | 3 | 1 | ACK |
| 9 | 0 | 0 | ACK |
| 10 | 1 | 0 | NAK |
| 11 | 2 | 0 | NAK |
| 12 | 3 | 0 | NAK |
| 13 | 0 | 1 | NAK |
| 14 ( = 10) | 1 | 0 | ACK |
| 15 | 2 | 0 | ACK |
| 16 | 3 | 0 | ACK |
| 17 | 0 | 1 | ACK |
| 18 | 1 | 1 | ACK |

In the table above, frames 10, 11, 12 and 13 should be received in that order by the user programs. Therefore, frames 11, 12 and 13 are not acknowledged to force them to be retransmitted. The sequence above is correct if the NAK response of frame 10 is received during or after sending of frame 13. If the NAK response on frame 10 is received at an earlier point of time, the sender will not bother to send the frames it knows will be NAKed, but goes directly on to retransmission of frame 10.

If both the second and the third byte in the header equals zero, an ACK is not expected for the frame and the sequence counts GS and IS is not increased. Such frames are sent when there is nothing waiting to be sent, but a frame requiring a status response is received. Thus, the only relevant information in such a frame is the GR, IR and STS fields. Because a Status frame is not acknowledged, a frame correctly received may be retransmitted. This will be detected by the receiver by the IS field and the frame will be acknowledged once more and rejected.

From the communication link procedure's point of view, a wait acknowledge status (WACK) is equivalent to an ACK status. The special actions taken on channel procedure level on receipt of a WACK is described in Section 3.6.4.3.

## 3.7.4 *Communication Channel Procedure*

The purpose of the communication channel procedure is to superwise the data transmitted on each channel. This is done by sending special frames on the communication channels. The format and function of special frames are described below.

### 3.7.4.1 Define Break Strategy

Format:

Control = 1
Channel = Channel number

I-field contains two bytes, the break strategy number and the maximum number of bytes between break. If the break strategy number is 7, the frame also contains a 16 bytes break table. The break strategy number may have the following values:

0 = Break on all bytes

1 = Break on control characters (bytes $\leqslant 37_8$ and $\neq 12_8$)

2 = MAC break strategy

3 = System defined break strategy

4 = System defined break strategy

5 = System defined break strategy

6 = System defined break strategy

7 = User defined break strategy

−1 = Never break

The current break strategy of a channel determines when a data frame is to be sent (put into the send queue). A data frame is scheduled for sending when one of the following conditions is fullfilled:

1. The I-field has reached its maximum size (256 bytes).
2. A break character is inserted in the I-field.
3. The maximum number of bytes between break is reached.
4. The input part of the channel is reserved by the same program as the output part and this program is waiting for input on the channel.

Condition 4 is necessary for interactive programs using a communication channel as command input/output device.

The break strategy is set by the receiving user program by the monitor call BRKM or IOSET.

The monitor call will cause the pointer to the appropriate break table to be inserted in the channel input datafield on the receiving side and the channel output datafield on the sending side.

The break strategy should be set to break as seldom as possible to minimize the communication overhead.

If the "never break" strategy is set, all data frames will be of maximum length and the last (partly filled) frame will be sent when the channel is closed by the CLOSE or IOSET monitor call.

When a break character is sent, a request for input frame (see Section 3.6.4.3) is required from the receiving side before the next frame is sent. A wait acknowledge frame (see Section 3.6.4.4) from the receiving side will have the same effect. This prevents the buffer pool on the receiving side from being filled up even if the receiver is reading data at a lower rate than it is sent by the sending program. If a frame is ready for sending but no request for input is present, the sending program will enter waiting state.

### 3.7.4.2 Define Echo Strategy

Format:

Control = 2
Channel = Channel number

I-field contains two bytes, the echo strategy number, and a dummy byte. If the echo strategy number is 7, the frame also contains a 16 bytes echo table. The echo strategy number may have the following values:

0 = Echo all characters except $177_8$

1 = Echo bytes $\geq 40_8$ and 7

2 = Echo bytes $\geq 40_8$ and 7, $12_8$ and $/5_8$

3 = System defined echo strategy

4 = System defined echo strategy

5 = System defined strategy

6 = System defined break strategy

7 = User defined break strategy

−1 = Never echo

The current echo strategy of a channel determines which characters to echo when the channel is used for interactive communication (by the @REMOTE command). The echo strategy is set by the user program by the monitor call ECHOM. The monitor call will cause the pointer to the appropriate echo table to be inserted in the channel data fields. The REMOTE command routine will, for each break character typed, insert the echo table pointer into the terminal input data field.

### 3.7.4.3 Request for Input (Poll)

Format:

Control = 3
Channel = Channel number
I-field = Empty

A request for input frame is sent from a receiver of data when the channel input queue is empty and there was a break character in the last frame and the user program requests more data. If a wait acknowledge is sent earlier, a request for input will be sent even if the last frame contains no break character.

The request for input frame serves the following purposes:

1. To allow type ahead and give correct echo in interactive communication (together with the break and echo frames).

2. To control data exchange between asynchronous programs in communicating computers (together with the wait acknowledge frame).

When a request for input frame is received, the request for input bit in the channel output data field is set.

### 3.7.4.4 Turn off Request for Input

Format:

Control = 4
Channel = Channel number
I-field = Empty

This frame is sent from the input side when the channel is closed to turn off the request for input status at the sending side. This is necessary if the last frame sent did not contain a break character.

When a turn off request for input frame is received, the request for input bit in the channel output data field is reset. A wait acknowledge (WACK) will have the same effect as "Turn off Request for Input".

### 3.7.4.5 System Configuration Frame

Format:

Control = 5
Channel = bit 0–3 remote line number,
bit 4 not set if poll for configuration frame (sent by START-COMMUNICATION), set if response on configuration poll frame.

I-field = 1 — 64 byte - configuration table.

The configuration table contains one 16 bit entry for each channel on the communication line. The entry for channel 0 occupies the 2nd and 3rd byte and 31 the 64th and 65th byte.

Each entry carries the following information:

bit 0 — 14:
logical device number of channel,
0 if channel not implemented.

bit 15:
1 if the channel has a background processor,
0 otherwise.

When the START-COMMUNICATION command is given, the system starts to send a configuration poll frame every 6 seconds. This continues until a configuration response frame is received, or the STOP-COMMUNICATION command is given.

When a configuration frame (poll or response) is received, the configuration table of the frame is stored in the remote part of the memory resident configuraion table for the line (see Section 4.3). If it was a configuration poll, a configuration response frame is returned together with the acknowledge of the received frame.

All internal system variables are initialized and all user programs eventually waiting for channel I/O are restarted and given an error return.

### 3.7.4.6 Load Data Frame

Format:

Control = 6
Channel = 0
I-field = Load data to be stored in main memory of receiving computer.

When a REMOTE-LOAD command is given, a bootstrap is first fetched from the system file REMOTE-BOOTSTRAP:BPUN, and sent on the line without frame header or CRC (to be read by hardware on satellite). Then the specified binary file is sent in load data frames. All load data frames except the last one, will contain 256 data bytes in the I-field.

The load data frames are read by the bootstrap on the satellite computer. There is for the moment no automatic recovery (retransmission) if a load data frame is received in error.

### 3.7.4.7 Remote File Access Frame

Format:

Control = 7
Channel = Channel number used for file access
I-field = Depends on frame type. The coding of the I-field is shown in the table below.

| *Function* | *1. byte* | *Rest of I-field* |
|---|---|---|
| Remote open file | 0 | 2. byte: access code<br>3. byte: background command channel (used by REMOTE command) if background open, else = $377_8$<br>4. byte: not used<br><br>If 3. byte - $377_8$,<br>Then<br>5. and 6. byte = RT password<br>file name from 7. byte<br>Else<br>file name from 5. byte<br>Fi |
| File input error message | 1 | 2. byte: error code, 0 if no error |
| File output error message | 2 | 2. byte: error code, 0 if no error |
| Close remote input file | 3 | not present |
| Close remote output file | 4 | not present |

A remote open file frame is sent if an open file monitor call or command is given with a file name containing a decimal point. All characters up to and including the first decimal point (from the left) in the file name are removed and the rest of the file name is sent in a remote open file frame.

When a remote open file frame is received, a system RT program is started. This program opens the file, returns a file error message (2. byte = 0 if OK) and starts the data transfer on the communication channel.

A file input error message frame is sent if an error condition occurs upon input from a remote file. The frame will cause the appropriate error return from the user programs INBT, MAGTP or RFILE monitor call.

A file output error message frame is sent if an error condition occurs upon output to a remote file. The frame will cause the appropriate error return from the user program OUTBT, MAGTP or WFILE monitor call.

A close remote input file frame is sent when a channel used for remote file read is closed. When the frame is received, the file is closed and the transfer RT program is aborted.

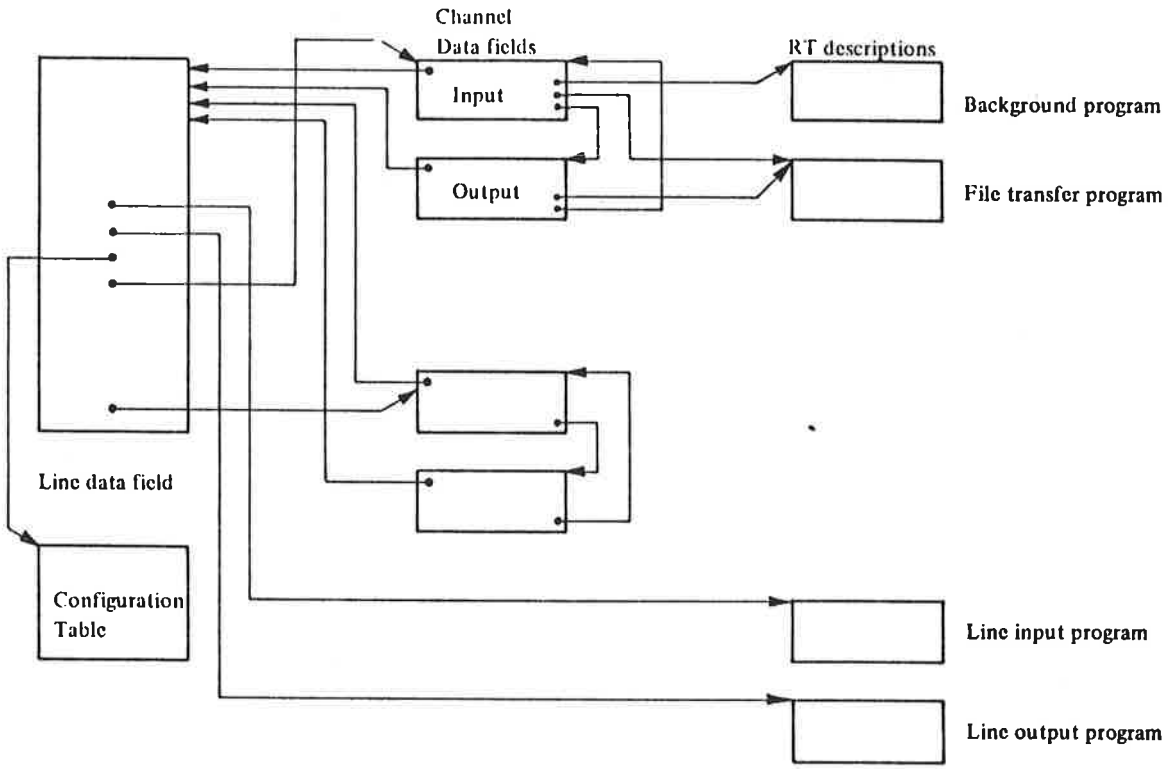
A close remote output file frame is sent when a channel used for remote file write is closed. When the frame is received, the last page is written to the file, the file is closed and the transfer RT program is aborted.

### 3.7.5 *User Level Communcation Procedure*

The link and channel procedures are completely invisible to the user. The user is seeing a communicaton channel as a byte oriented peripheral device. He may therefore define any communicaton procedure between user programs.

# 4 INTERNAL DATA STRUCTURE

The internal data fields for a communication line is illustrated below:




There is one line data field for each communication line. A line table contains pointers to the line data fields. The line table has the following format:

| *Label* | *Contents* |
|---|---|
| LINAR | Number of entries in table |
| | Pointer to line data field of line 1 |
| | Pointer to line data field of line 2 |
| | Pointer to line data field of line 3 |
| | Pointer to line data field of line 4 |
| | Pointer to line data field of line 5 |
| | Pointer to line data field of line 6 |

## 4.1 *REMOTE PASSWORD TABLE*

The remote password table contains the passwords given in the REMOTE-PASSWORD command and has the following format:

| *Label* | *Contents* |
|---|---|
| RPASS | Password for line 1<br>Password for line 2<br>Password for line 3<br>Password for line 4<br>Password for line 5<br>Password for line 6 |

When a remote open file is executed from an RT program, the password in the remote password table is compared with the password of user RT in the remote computer.

## 4.2 *LINE DATA FIELD*

There is one line data field for each communication line. It contains the following information (n denotes line number):

| *Label* | *Disp. from label* | *Entry Type and Name (NORD PL Notation)* | *Contents* |
|---|---|---|---|
| | –7 | INTEGER CCTRL | Hardware control word |
| | –6 | INTEGER POINTER TMSUB | Timer routine |
| | –5 | INTEGER TMR | Counter for timer |
| | –4 | INTEGER TTMR | Time out time (–seconds) |
| | –3 | INTEGER HDEV | IOX input instruction |
| | –2 | INTEGER POINTER STDRIV | Initial start of input driver |
| | –1 | INTEG POINT DRIVER | Input driver restart address |
| TTILn | 0 | INTEGER RESLINK | Reservation link |
| | 1 | INTEGER RTRES | Reserving RT program |
| | 2 | INTEGER BWLINK | Wait link |
| | 3 | INTEGER TYPRING | Flag word |
| | 4 | INTEGER ISTATE | Wait flag. ≠ 0 when reserving RT-prog is waiting for transfer on this device. |
| | 5 | INTEGER MLINK | Monitor link |
| | 6 | INTEGER MFUNC | Restart after I/O wait routine |
| | 7 | INTEGER CHKO | Character pointer in input buffer |
| | 10 | INTEGER STDEV | Start transfer routine |
| | 11 | INTEGER ANCHA | Character counter for input driver |
| | 12 | INTEGER DFOPP | Pointer to TTULn |
| | 13 | INTEGER POINTER IINI | I/O initializing routine |
| Not DMA HDLC | 14 | INTEGER NPFBUF | Current buffer chain used by input driver. Value in range [0,3] |
| | 15 | INTEGER SWITCH | Go switch for input driver |

| *Label* | *Disp. from label* | *Entry Type and Name (NORD PL Notation)* | *Contents* |
|---|---|---|---|
| Not DMA HDLC | 16 | INTEGER NPHBUF | Current buffer chain read by received RT program. Range [0,3] |
| | 17 | INTEGER CUDBU | Current buffer filled by driver |
| | 20 | INTEG ARRAY IDBST (4) | Buffer chain status of buffer chain X.<br>0 — empty (free to use)<br>6 — filled |
| | 24 | INTEG ARRAY IDBAD (4) | Address of first buffer in buffer chain X |
| Only for HDLC DMA | 14 | INTEGER LIBEG | Start of buffer descriptor list |
| | 15 | INTEGER SWITCH | |
| | 16 | INTEGER POINTER HPEK | Read pointer in buffer descriptor list |
| | 17 | INTEGER CUDBU | |
| | 20 | INTEGER ARRAY INIAD (7) | Initializing parameters for interface |
| | 27 | INTEGER ARRAY LIBOI (124) | Buffer descriptor list |
| | −4 | INTEGER OCTRL | Output control word |
| | −3 | INTEGER HDEV | IOX output instruction |
| | −2 | INTEG POINTER STDRIV | Inital start of output driver |
| | −1 | INTEG POINTER DRIVER | Output driver restart address |
| TTULn | 0 | INTEGER RESLINK | Reservation link |
| | 1 | INTEGER RTRES | Reserving RT program |
| | 2 | INTEGER BWLINK | Wait link |
| | 3 | INTEGER TYPRING | Flag word |
| | 4 | INTEGER ISTATE | Wait flag ≠ 0 when reserving RT program is waiting for transfer on this device. |
| | 5 | INTEGER MLINK | Monitor link |
| | 6 | INTEGER MFUNC | Restart after I/O wait routine |
| | 7 | INTEGER CHKO | Character pointer in output buffer |

| *Label* | *Disp. from label* | *Entry Type and Name (NORD PL Notation)* | *Contents* |
|---|---|---|---|
| | 10 | INTEGER STDEV | Start transfer routine |
| | 11 | INTEGER ANCHA | Character counter for output driver |
| | 12 | INTEGER DFOPP | Pointer to TTILn |
| | 13 | INTEGER POINTER RSICH | Reset device routine |
| | 14 | INTEGER BUFST | Pointer to first buffer of current output buffer chain |
| | 15 | INTEGER SWITCH | Go switch for output driver |
| | 16 | INTEGER LMDAT | Pointer to CMOn |
| | 17 | INTEGER CUDBU | Current output buffer |
| | 20 | INTEGER ARRAY SLIPO (25) | Buffer descriptor list for HDLC DMA interface |
| | –14 | INTEGER TSPEED | Line speed |
| | –13 | | |
| | –12 | DOUBLE SIAD | Save AD registers while computing CRC on input |
| | –10 | DOUBLE SOAD | Save AD regvisters while computing CRC on output |
| | –6 | INTEGER POINTER TMSUB | Timer routine |
| | –5 | INTEGER TMR | Counter for timer |
| | –4 | INTEGER TTMR | Time out time (–seconds) |
| | –3 | INTEGER XTEMI | Save X register while computing CRC on input |
| | –2 | INTEGER XTEMO | Save X register while computing CRC on output |
| | –1 | INTEGER ISIT | Frame phase bit on input |
| CMOn | 0 | INTEGER GSI | Group number on input |
| | 1 | INTEGER ANI | Status information in input frame |
| | 2 | INTEGER GRI | Group number for ANI |
| | 3 | INTEGER IRI | Phase bit for GRI |

| *Label* | *Disp. from label* | *Entry Type and Name (NORD PL Notation)* | *Contents* |
|---|---|---|---|
| | 4 | INTEGER ISTATE | Send status<br>1 when waiting for acknowledge<br>0 otherwise |
| | 5 | INTEGER ARRAY ISI (4) | Last received phase bit on group X |
| | 11 | INTEGER GSO | Current group number output |
| | 12 | INTEGER CDFILD | Pointer to TTILn |
| | 13 | INTEGER POINTER SINIT | Initializing routine |
| | 14 | INTEGER POINTER BSINIT | Initializing routine for remote load |
| | 15 | INTEGER PLMSG | Last frame of send queue } Send queue head |
| | 16 | INTEGER PFMSG | First frame of send queue } Send queue head |
| | 17 | INTEGER ARRAY ISO (4) | Last sent phase bit on group X |
| | 23 | INTEGER POINTER SEND | Send routine |
| | 24 | INTEGER POINTER BSEND | Send bootstrap routine for remote load |
| | 25 | INTEGER POINTER RECEIVE | Receive routine |
| | 26 | INTEGER POINTER VENTX | Wait routine (used for half duplex communication) |
| | 27 | INTEGER RMLNR | Remote line number |
| | 30 | INTEGER FRETR | Frames retransmitted |
| | 31 | INTEGER MISTRART | Start of current input frame |
| | 32 | INTEGER SBYTS | Number of I-field bytes in output frame |
| | 33 | INTEGER CUIBU | Current input buffer |
| | 34 | INTEGER CUUBU | Current output buffer |
| | 35 | INTEGER RGSI | Temporary variable to hold last received group number |

| *Label* | *Disp. from label* | *Entry Type and Name (NORD PL Notation)* | *Contents* |
|---|---|---|---|
| | 36 | INTEGER SQERR | Number of sequence errors since communication was started |
| | 37 | INTEGER RNACO | Number of NAK's received since communication was started |
| | 40 | INTEGER SNACO | Number of NAK's sent since communication was started |
| | 41 | INTEGER CTRCH | Control and channel of last received frame |
| | 42 | INTEGER BADANT | Retransmit flag. Set by reception of NAK and by timeout. |
| | 43 | INTEGER SMI | CRC on input |
| | 44 | INTEGER SMO | CRC on output |
| | 45 | INTEGER ACQFP | Write pointer in status information buffer. Range [0,3 ] |
| | 46 | INTEGER ACQHP | Read pointer in status information buffer. Range [0,3 ] |
| | 47 | INTEGER ACQBH | Number of items in status information buffer. Range [0,4 ] |
| | 50 | INTEGER IBYTS | Number of I-field bytes in input frame |
| | 51 | INTEGER CURID | Address of current input buffer |
| | 52 | INTEGER SPRS | Send RT program |
| | 53 | INTEGER RPRS | Receive RT program |
| | 54 | INTEGER ARRAY ACQU (4) | Status information buffer Used to pass status information from receive program to send program. Contains status information for the last four frames received. There is one word per item, containing GRI, IRI and ANI right justified. |
| | 60 | INTEGER POINTER ILSAV | Temporary L-save |
| | 61 | INTEGER PGR OLSAV | Temporary L-save |

| *Label* | *Disp. from label* | *Entry Type and Name (NORD PL Notation)* | *Contents* |
|---|---|---|---|
| | 62 | INTEGER LINR | Line number |
| | 63 | INTEGER DNACO | Number of NAK's sent since last ACK sent. |
| | 64 | INTEGER ARRAY | Output status for last sent frame in group X.<br>0 = EMPTY if acknowledged<br>2 = FULLN if rejected (NAK received)<br>3 = FULLS if sent, but no status information received yet. |
| | 70 | INTEGER ARRAY BUSTH (4) | Address of first buffer of output frame in group X. |
| | 74 | INTEGER INHBT | Address of first buffer of output frame while it is retransmitted. Used to avoid that the receive program releases the output buffer while the send program retransmits it. |
| | 75 | INTEGER LOST | Group number frame to retransmit. Set by reception of NAK. |
| | 76 | INTEGER ARRAY POINTER COTAB | Pointer to configuration table |
| | 77 | INTEGER COFLAG | Line status<br>0 when communication is running<br>2 when communication is dead |
| | 100 | INTEGER RETRN | Retransmission count. Incremented by one when a frame is retransmitted. Reset to zero when an ACK is received. If RETRN exceeds 4, COFLAG is set to 2. |
| | 101 | INTEGER AKMCH | Number of channels with lower number than first channel on this line. |
| | 102 | INTEGER MXCHN | Maximum channel number on this line (number of channels – 1). |
| | 103 | INTEG ARRAY IDADR (MXCHN) | Pointer array containing pointers to the channel input data fields of this line. |

## 4.3 *CONFIGURATION TABLE*

There is one configuration table for each communication line. The configuration table consists of 32 two-word entries, one for each possible channel. The first word of each entry describes the local end of the channel and the second word the remote end. Each word has the following contents:

Bits 0 — 14: logical device number of channel
Bit 15: set if the channel has background processor.

The local part of the configuration table is initated at system generation time. The remote part is received from the remote computer when the STAR-COMMUNICATION command is given. When two computers are communicating, their configuration tables will be equal, except that the two words of each entry are swapped.

## 4.4 *CHANNEL DATA FIELDS*

There is one input and one output data field for each communication channel on each line. The channel data fields serve the same purpose as the I/O data fields for other peripheral devices and have their pointers in the logical number table.

The channel data fields contains mostly the same entries as the Teletype data fields. The deviations from the Teletype data fields are described below.

*Channel input data field (cc = Channel number, n = Line number)*

| *Label* | *Disp. from label* | *Entry Type and Name* | *Contents* |
|---|---|---|---|
| | —17<br>—16<br>—15 | | Same as TTY |
| | —14 | INTEGER ANTORD | Number of words to read (for MAGTP) |
| | —13 | INTEGER XSAC | Save X register in IOTRANS |
| | —12<br>—<br>—7 | | Save as TTY |
| | —6 | INTEGER CMDAT | Pointer to line data field |
| | —5 | INTEGER IXSAC | Save DF datafield for MAGTP |
| | —4 | INTEGER ANTMEL | Number of frames in input queue |
| | —3 | INTEGER BYTS | Byte counter in frame |
| | —2 | INTEGER INCR | Byte counter in buffer |
| | —1 | INTEGER CHAN | Channel number [0, $37_8$] |
| SccnR | 0—13 | | Same as TTY |
| | 14 | INTEGER MSSTART | Pointer to first buffer of frame |
| | 15 | INTEGER PLMSG | Last frame of input queue } Input queue head |
| | 16 | INTEGER PFMSG | First frame of input queue } Input queue head |
| | 17 | INTEGER CURBU | Current input buffer |
| | 20 | INTEGER CHNST | Channel input status<br>ITOM = 1<br>if input queue contains a partly emptied frame,<br>EMPTY = 0<br>else. |
| | 21 | INTEG POINTER LRSA | Save L register in IOTRANS |
| | 22—42 | | Same as TTY |
| | 43 | INTEGER RTUT | RT description of file transfer program |

*Channel Output Data field (cc = Channel number, n = Line number)*

| *Label* | *Disp. from label* | *Entry Type and Name* | *Contents* |
|---|---|---|---|
| | –14 | INTEGER ANTORD | Number of words to write (for MAGTP) |
| | –13 | INTEGER XSAC | Save X register in IOTRANS |
| | –12 | INTEGER DFLAG | Flag word (described below) |
| | –11 | INTEGER ECHOTAB | Pointer to echo table |
| | –10 | INTEGER BRKTAB | Pointer to break table |
| | –7 | INTEGER LAST | Save last byte |
| | –6 | INTEGER CMDAT | Pointer to line data field |
| | –5 | INTEGER MSIZE | Maximum number of bytes in a frame (406) |
| | –4 | INTEGER SCPRI | Channel priority |
| | –3 | INTEGER BYTS | Byte counter in frame |
| | –2 | INTEGER INCR | Byte counter in buffer |
| | –1 | INTEGER CHAN | Channel number [0, $37_8$] |
| SccnW | 0–13 | | Same as TTY |
| | 14 | INTEGER MSSTART | Pointer to first buffer of frame |
| | 15 | INTEGER UANTMEL | Number of frames in send queue for this channel |
| | 16 | | Not used |
| | 17 | INTEGER CURBU | Current output buffer |
| | 20 | INTEGER CHNST | Channel output status<br>ITOM = 1<br>if partly filled<br>frame on output,<br>EMTY = 0<br>else |
| | 21 | INTEGER LRSA | Save L register in IOTRANS |
| | 22 | INTEGER RTIN | RT description of file transfer program |
| | 23–24 | | Same as TTY |

Bit 5COM = $13_8$ in TYPRING is set in the channel data field to indicate SINTRAN/SINTRAN communication channel data field.

The bits 5SPEC = 5 and 5WRQU = 6 in DFLAG is used in the following way:

*Channel input:*

5SPEC
: is set when a break character is read, and reset when request for input is sent.

*Channel output:*

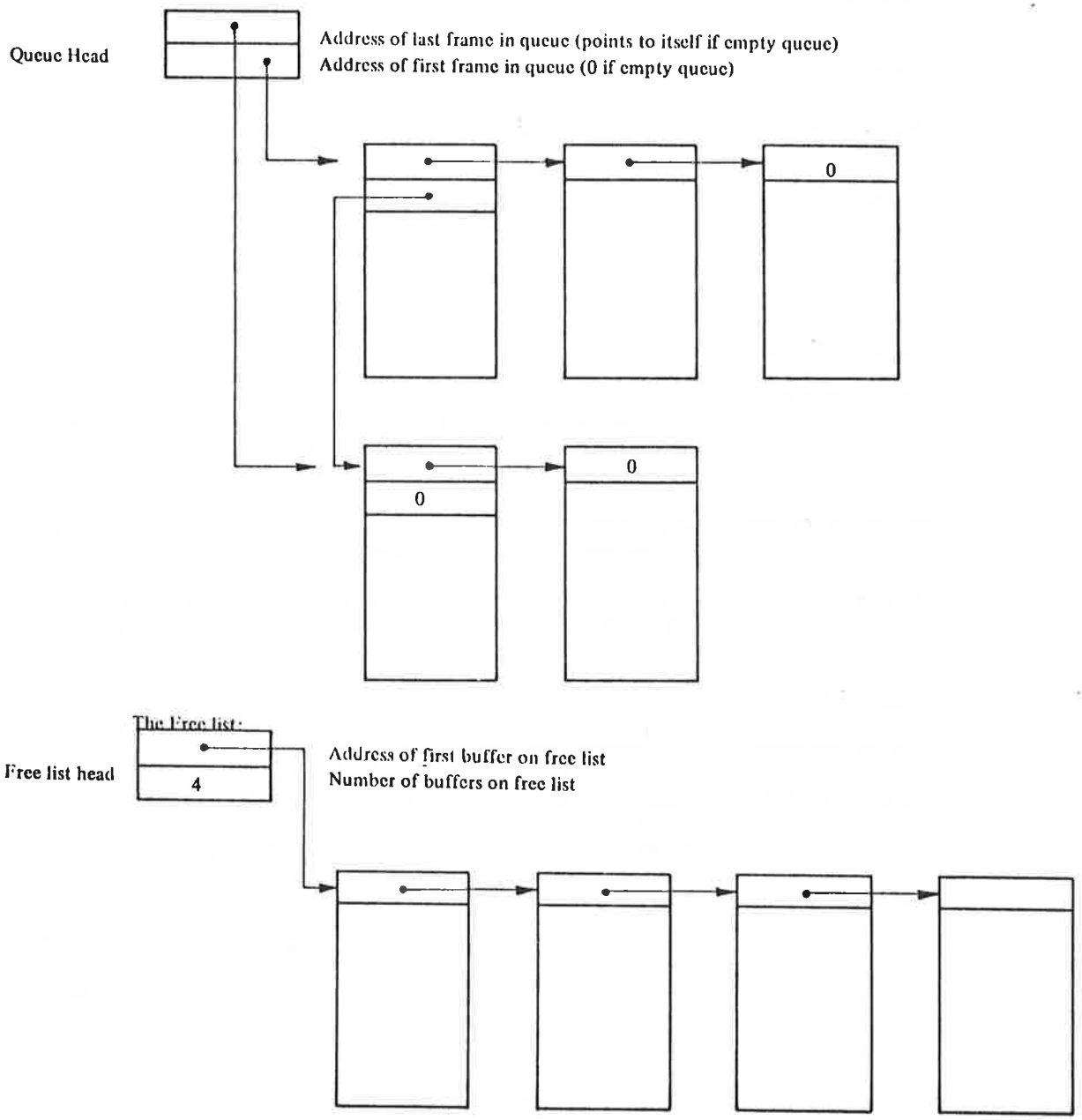
5SPEC
: is set when a break character is inserted in the output frame, and reset when the frame is appended to the send queue.

5RQI
: is set when a request for input frame is received, and reset when a frame containing a break character is appended to the send queue. This bit is also reset when a Turn off request for input or WACK frame is received.

5WRQI
: is set when an output user program enters waiting state because no request for input is received, and reset when the request for input arrives.

# 5 THE BUFFER POOL SYSTEM

To utilize the available memory resident buffer space, a dynamic buffer allocation and queueing system is used. The system is illustrated below.




The size of each buffer is $34_8$ words, giving a maximum of 5 buffers for one frame.

In addition to the free list, the system contains the following queues:

— The send queue
This queue contains frames wiating to be sent.
Queue head in line data field.

— The receive queue
Queue of received frames for a channel.
Queue head in channel input data field.

## 5.1 *BUFFER POOL AND QUEUEING ROUTINES*

All allocating of buffers and queuing of frames is done by the monitor level routines PUT, PRIPUT and GET. The free list in initiated on system start by the routine BUFIN.

*Routine: PUT*

| | |
|---|---|
| Input parameters: | X = address of queue head<br>A = address of first buffer in frame |
| Output parameters: | Skipreturn if OK<br>No skipreturn if error |
| Function: | Links a frame to a queue (or free list) |

*Routine PRIPUT*

| | |
|---|---|
| Input parameters: | X = address of queue head<br>A = address of first buffer in frame<br>Priority stored as first byte in the frame,<br>(left halft of 3. word of first buffer). |
| Output parameters: | Skipreturn if OK<br>No skipreturn if error or X = free list. |
| Function: | Links a frame to a queue in front of the first frame with lower priority than this frame. |

*Routine GET*

| | |
|---|---|
| Input parameters: | X = queue address |
| Output parameters: | Skipreturn if OK. A = address of first buffer of frame, or buffer address (if free list).<br>No skipreturn if error, or no frame or buffer available. |
| Function: | If X = free list |

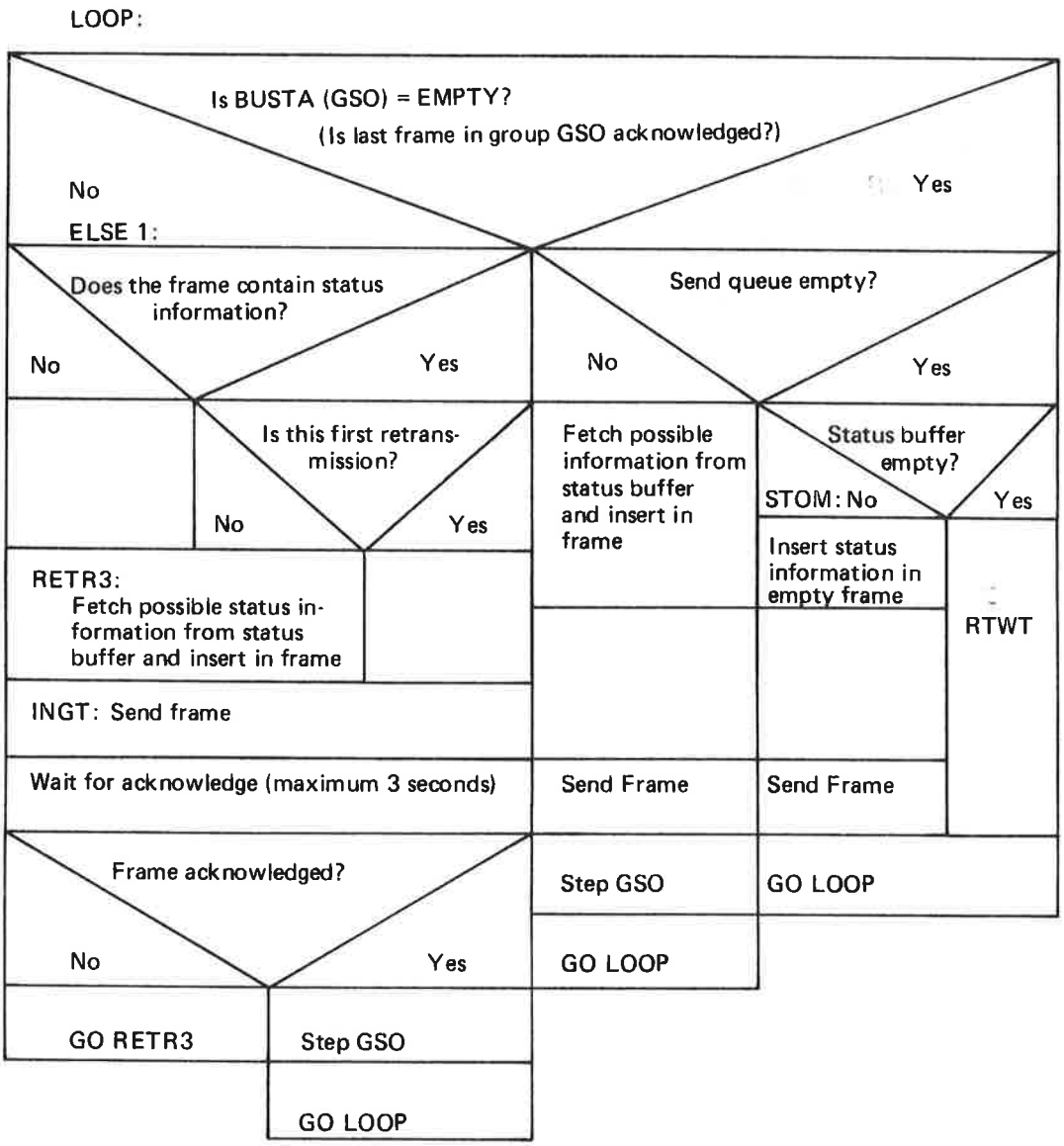
# 6 SYSTEM ROUTINES

## 6.1 *RSEND*

This is the send RT program fetching frames from the send queue and sending them on the communication line by activation of the line driver. There is one RSEND routine for each communication line, but except for the first two instructions setting the B register, they share the same code.

The RSEND routine enters RTWAIT when the send queue and status buffer (ACQU) is empty, and enters I/O wait while a frame is sent.

The functions of RSEND are illustrated below. The figure is simplified. Consult the listing to get all details.



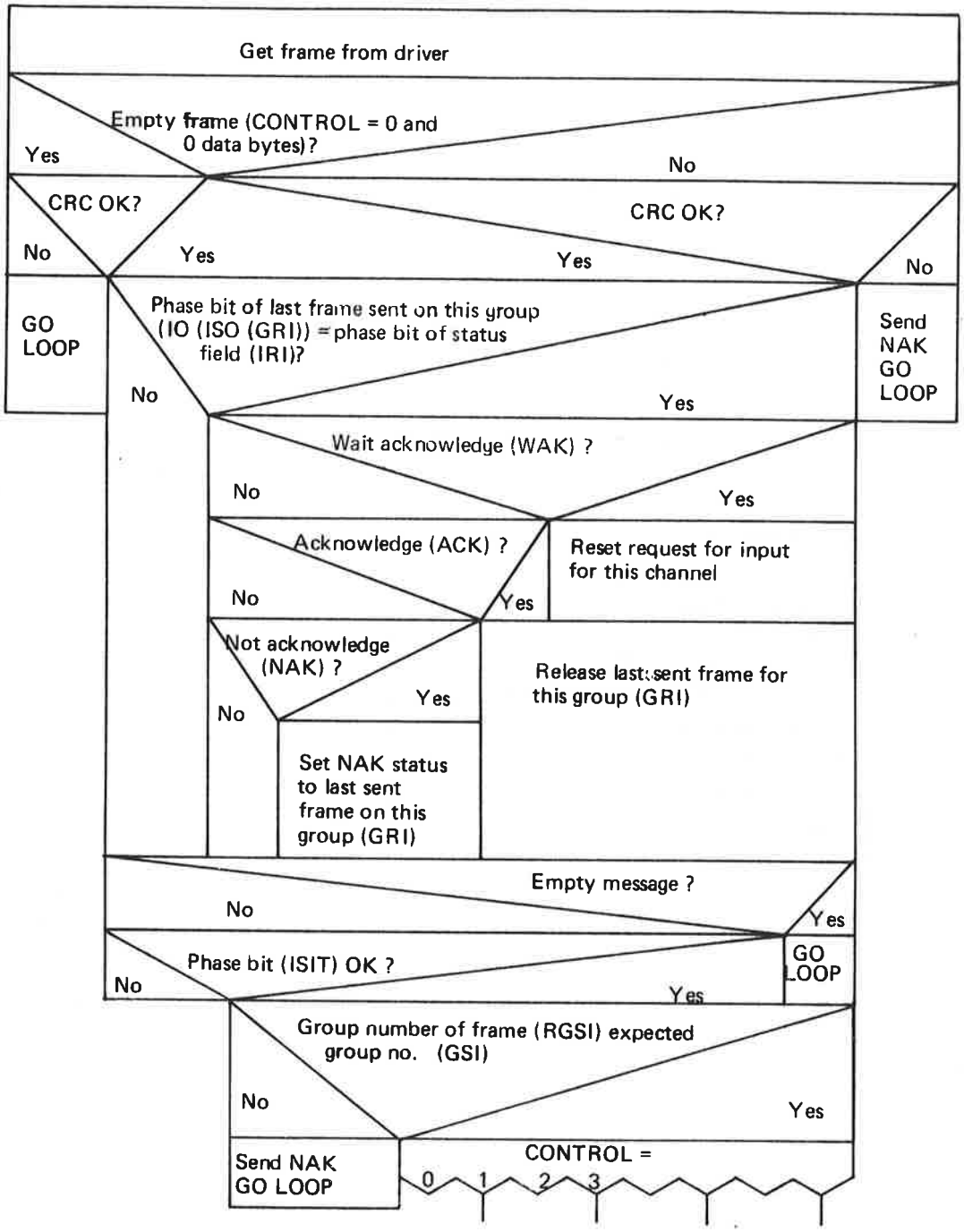
## 6.2 *RTREC*

This is the receive RT program. Its task is to fetch frames from the communication line handled over by the line driver, check for validity and rout them to the channel input queues. There is one RTREC routine for each communication line, but except for the first two instructions setting the B register, they share the same code.

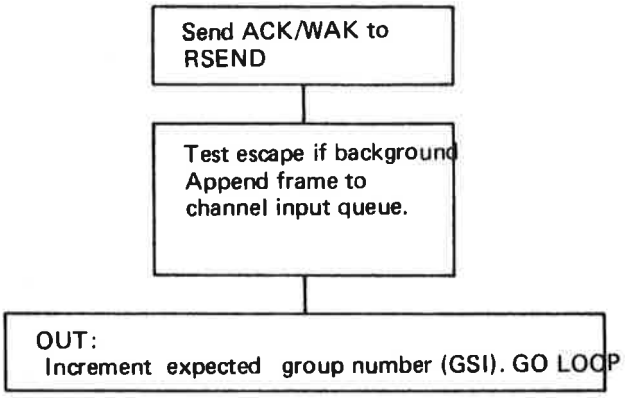
The RTREC routine enters I/O wait if the line input driver has no frame ready when RTREC asks for it.

The functions of RTREC is illustrated below. The figures is simplified. Consult the listing to get all details.

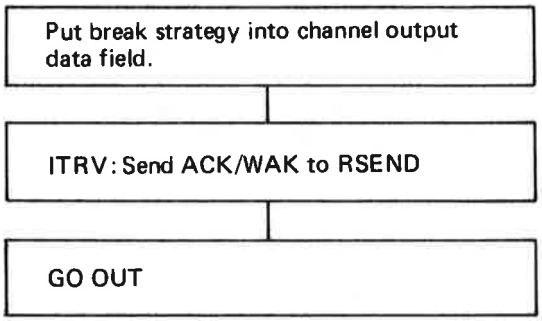
LOOP:


Get frame from driver
Empty frame (CONTROL = 0 and 0 data bytes)?
Yes
No
CRC OK?
No
Yes
CRC OK?
Yes
No
GO LOOP
Phase bit of last frame sent on this group (IO (ISO (GRI)) = phase bit of status field (IRI)?
No
Yes
Send NAK GO LOOP
Wait acknowledge (WAK) ?
No
Yes
Acknowledge (ACK) ?
No
Yes
Reset request for input for this channel
Not acknowledge (NAK) ?
No
Yes
Release last sent frame for this group (GRI)
Set NAK status to last sent frame on this group (GRI)
Empty message ?
No
Yes
Phase bit (ISIT) OK ?
No
Yes
GO LOOP
Group number of frame (RGSI) expected group no. (GSI)
No
Yes
Send NAK GO LOOP
CONTROL =
0
1
2
3

DATMESS:




DEFBREAK:



DEFECHO:

| |
|---|
| Put echo strategy into channel output data field. |
| GO ITRV |

REQIN:

| |
|---|
| **Set** request for input bit in channel output data field. |
| GO ITRV |

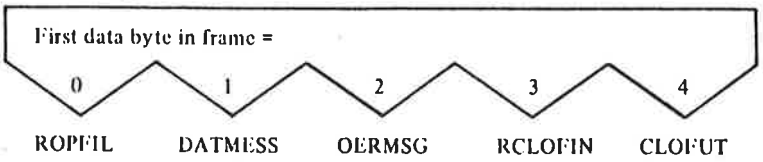
IRQIN:

| |
|---|
| Reset request for input bit in channel output data field |
| **GO** ITRV |

CONFIG:

| | |
|---|---|
| Initialize system variables | |
| Put data part of frame into remote part of configuration table. | |
| Activate all programs waiting for I/O on a channel. | |
| Answer on request frame? Yes | No |
| | Append configuration message to send queue. |
| GO ITRV | |

FILMESS:




ROPFIL:

| |
|---|
| Reserve channel for file transfer program |
| Initialize RT description for file transfer program |
| Start file transfer program |
| GO ITRV |

OERMSG:

| |
|---|
| Put error code into channel output data field and activate reserving program |
| GO ITRV |

RCLOFIN:

| |
|---|
| Restart the file transfer program at the entry point CLOSIN |
| GO ITRV |

CLOFVT:

| Channel reserved by file transfer program? | |
|---|---|
| No | Yes |
| Return file output operation OK to remote computer | Set second data byte in frame to 377 |
| | GO DATMESS |

## 6.3 *RTFIN*

This is an RT program used to transfer data from a file to a remote program. RTFIN is activated when a remote open file command or monitor call is executed on a remote connected computer. There is one RTFIN RT program for each communication channel. The program resides on a 2K segment together with RTFUT and a 1K buffer. If the calling program is a background program, a system segment is used for the open file table.

RTFIN has two entry points RTFIN and CLOSIN. It is activated at RTFIN when a remote open frame is received by RTREC. The file is first opened and a response is sent to the remote computer. Then RTFIN starts reading data from the file page by page and writing it to the communication channel.

Because more than one RTFIN/RTFUT program may be active at the same time using the same system segment, the system segment must be reserved when it is used. This is done by reserving the corresponding terminal input data field.

Input parameters to RTFIN are:

T register = frame address of open file frame
B register = line data field

RTFIN is activated at CLOSIN when a "close input file" frame is received by RTREC. Then the file is closed and a response to the remote close is appended to the send queue.

If the calling program is an RT program, RTFIN checks that the password in the frame is the correct password for user RT. In this case no system segment is used and the file is opened as an RT file.

The format of the remote file access frames may be found in Section 3.6.4.7.

## 6.4 *RTFUT*

This is an RT program used to transfer data from a remote program (via a communication channel) to a file. Its operation is similar to RTFIN (Section 6.3) except that a "close output file" command is given from RTREC to RTFUT by error return $377_8$ on channel INBT. Thus, RTFUT has only one entry point (RTFUT).

Input parameters to RTFUT are:

T register = frame address of open file frame
B register = line data field

The format of the remote file access frames may be found in Section 3.6.4.7.

## 6.5 *TSEND*

This routine is called from RSEND to send a frame on the communication line. TSEND activates the driver, usually on level 10 or 12, and puts the calling program into waiting state. The calling program is reactivated by the driver when the entire frame is sent. The driver is adding the Synchronization Sequence (SS).

The input parameters to TSEND are:

A register = frame address
B register = line data field

## 6.6 *TBOSND*

This routine is called from the REMOTE-LOAD command to send a byte string without frame header or trailer. This is necessary when sending the bootstrap to be read by hardware on the remote computer. The operation of TBOSND is similar to TSEND (Section 6.5).

The input parameters to TBOSND are:

A register = byte string address (stored in linked buffers)
B register - line data field
X register = number of bytes to send

## 6.7 *TRECEIVE*

This routine is called from RTREC to get a frame from the communication line. The function of TRECEIVE depends on the type of line interface used. For interfaces giving interrupt on every byte received, it works the following way:

There are four buffer chains filled by the input driver in a cyclic manner and emptied by TRECEIVE. This is done by maintaining a four word ring buffer containing pointers to the buffer chains. The writer pointer is stepped by one, by the driver when a frame is received. The reader pointer is stepped by one by TRECEIVE when a frame is fetched.

The buffer chains are allocated long enough to contain a frame of maximum length, but unused trailing buffers are returned to the buffer pool by TRECEIVE when the chain is fetched. When a buffer chain is fetched, TRECEIVE immediately allocates a new buffer chain and inserts its address in the four word ring buffer.

For interfaces not checking the CRC by hardware, it is calculated and checked by TRECEIVE.

For the DMA version of the HDLC interface TRECEIVE works in a slightly different way. The data field contains a 20 elements ring buffer (LIPOI) of buffer descriptors. Each buffer descriptor occupies 4 words of memory and describes a single 28 word buffer. The interface operates directly on the buffer descriptors. The writer pointer in the buffer descriptor ring buffer is maintained by the interface (hardware) while the reader pointer (HPEK) is maintained by TRECEIVE (software).

The driver is started (by interrupt from the interface) only when an entire frame is received or an error condition has occurred. The buffer descriptors have the following format:

| *Displacement* | *Name* | *Contents* |
|---|---|---|
| 0 | KEY | Buffer status |
| 1 | BYCO | Number of bytes in buffer |
| 2 | MOAD | Most significant 2 bits of buffer address |
| 3 | LEAD | Leas significant 16 bits of buffer address |

For further information about the KEY element, see the HDLC manual.

If there is no frames received when TRECEIVE is called, it starts the input driver and puts the calling program into waiting state. The calling program is reactivated from the driver when the first frame is received.

The input parameter to TRECEIVE is:

B register = line data field

The output parameter is:

T register = address of frame

## 6.8 *SSTI*

SSTI is the I/O transfer (IOTRANS) input routine for a communication channel. It is called on INBT/OUTBT level when an INBT monitor call is executed. Each call moves one byte from the channel input queue to the user's A register. If the channel input queue is empty, the user program is put into waiting state. If, in addition, the last received character was a break character, a poll (request for input) frame is sent to the remote computer.

If both the input and output part of a channel is used by the same program, and when SSTI is called, the input queue is empty and the last received character was a break character, then any partly filled frame on the *output* channel is appended to the send queue. This is required when running interactive programs where output and input is typed on the same line.

The DFLAG bit 5SPEC is used to flag if a break character is received.

The input parameter to SSTI is:

B register = channel input data field

Output parameters:

Skip return: A register = data byte
Return: Channel queue empty

## 6.9 *SSTO*

SSTO is the I/O transfer (IOTRANS) output routine for a communication channel. It is called on INBT/OUTBT level when an OUTBT monitor call is executed. Each call appends one byte to the current channel output frame. The channel output frame is appended to the send queue when it has reached its maximum length or a break character is inserted, if a poll (request for input) is received since the last break character was sent. If this is not the case, the frame is not appended to the send queue, and the calling program is put into waiting state. It is also checked by SSTO that the number of frames in the send queue from this channel does not exceed a certain maximum. If it does, the calling program is put into waiting state.

The DFLAG bit 5SPEC is used to flag that a break character is inserted in the current frame. The bit 5RQI (set by RTREC) indicates that a poll is received for this channel since the last frame containing a break character was sent.

The input parameters to SSTO are:

B register = channel output data field
A register = data byte

Output parameter:

Skip return: Byte put into current output frame.
Return: "Busy" (no poll received, send queue too long, or buffer pool full).

## 6.10 *MSSTI*

MSSTI is called from the monitor calls MAGTP and RFILE if the parameter "logical unit number" corresponds to a communication channel.

MSSTI transfers the number of words requested by the user from the channel input queue to the users buffer area.

If the requested amount of data is not available, the user program is put into waiting state until more data or an error message arrives. If a break character is received since the last poll was sent (5SPEC in DFLAG set), a new poll is sent.

The input parameters to MSSTI are:

B register = "DF" data field
X register = channel input data field

## 6.11 *MSSTO*

MSSTO is called from the monitor calls MAGTP and WFILE if the parameter "logical unit number" represents a communication channel.

MSSTO transfers the number of words specified by the user from his buffer to the send queue.

If a poll is not received on this channel since the last break character was sent (5RQI in DFLAG is not set), the user program is put into waiting state and the data is not appended to the send queue until a poll arrives.

The input parameters to MSSTO are:

B register = "DF" data field
X register = channel input data field

NORSK DATA A.S

P.O. Box 4, Lindeberg gård
Oslo 10

# COMMENT AND EVALUATION SHEET

ND–60.081.02

NORDNET System Documentation

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** _____

_____

_____