Norsk Data

SINTRAN III J-version Release Information

ND-60.230.01

SINTRAN III J-version Release Information

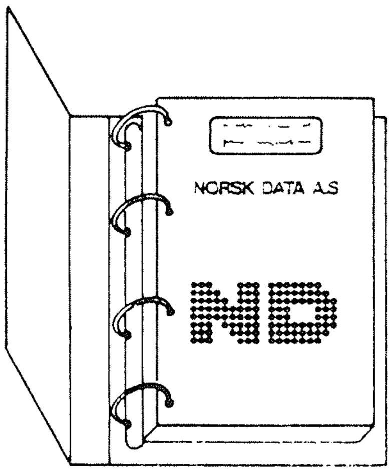
ND-60.230.01

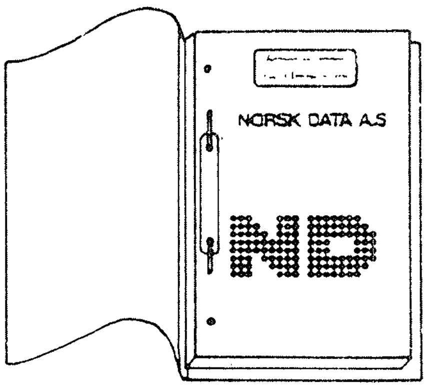
NOTICE

This manual is in loose-leaf form for ease of updating. Old pages may be removed and new pages easily inserted if the manual is revised.

The loose-leaf form also allows you to place the manual in a ring binder (A) for greater protection and convenience of use. Ring binders with 4 rings corresponding to the holes in the manual may be ordered in two widths, 30 mm and 40 mm. Use the order form below.

The manual may also be placed in a plastic cover (B). This cover is more suitable for manuals of less than 100 pages than for large manuals. Plastic covers may also be ordered below.





A: Ring Binder

B: Plastic Cover

Please send your order to the local ND office or (in Norway) to:

Norsk Data A.S
Graphic Center
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

ORDER FORM

I would like to order

...... Ring Binders, 30 mm, at nkr 20,- per binder

...... Ring Binders, 40 mm, at nkr 25,- per binder

...... Plastic Covers at nkr 10,- per cover

Name ....

Company ....

Address ....

....

City ....

PRINTING RECORD

Printing	Notes
01/85	Version 01

SINTRAN III J-version Release Information
Publ.No. : ND-60.230.01
January 1985

Norsk Data

Norsk Data A.S
Graphic Center
P.O.Box 25, Bogerud
0621 Oslo 6, Norway

Manuals can be updated in two ways, new versions and revisions. New Iversions consist of a complete new manual which replaces the old manual. New versions incorporate all revisions since the previous version. Revisions consist of one or more single pages to be merged into the manual by the user, each revised page being listed on the new printing record sent out with the revision. The old printing record should be replaced by the new one.

New versions and revisions are announced in the ND Bulletin and can be ordered as described below.

The reader's comments form at the back of this manual can be used both to report errors in the manual and to give an evaluation of the manual. Both detailed and general comments are welcome.

These forms and comments should be sent to:

Documentation Department
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

Requests for documentation should be sent to the local ND office or (in Norway) to:

Graphic Center
Norsk Data A.S
P.O. Box 25, Bogerud
0621 Oslo 6, Norway

TABLE OF CONTENTS

V A L I D   F O R :

S I N T R A N III / V S E	- ALL ND-100
S I N T R A N III / V S X	- ONLY ND-100/CX
S I N T R A N III / V S X - 5 0 0	- ONLY ND-100/CX

V E R S I O N   J

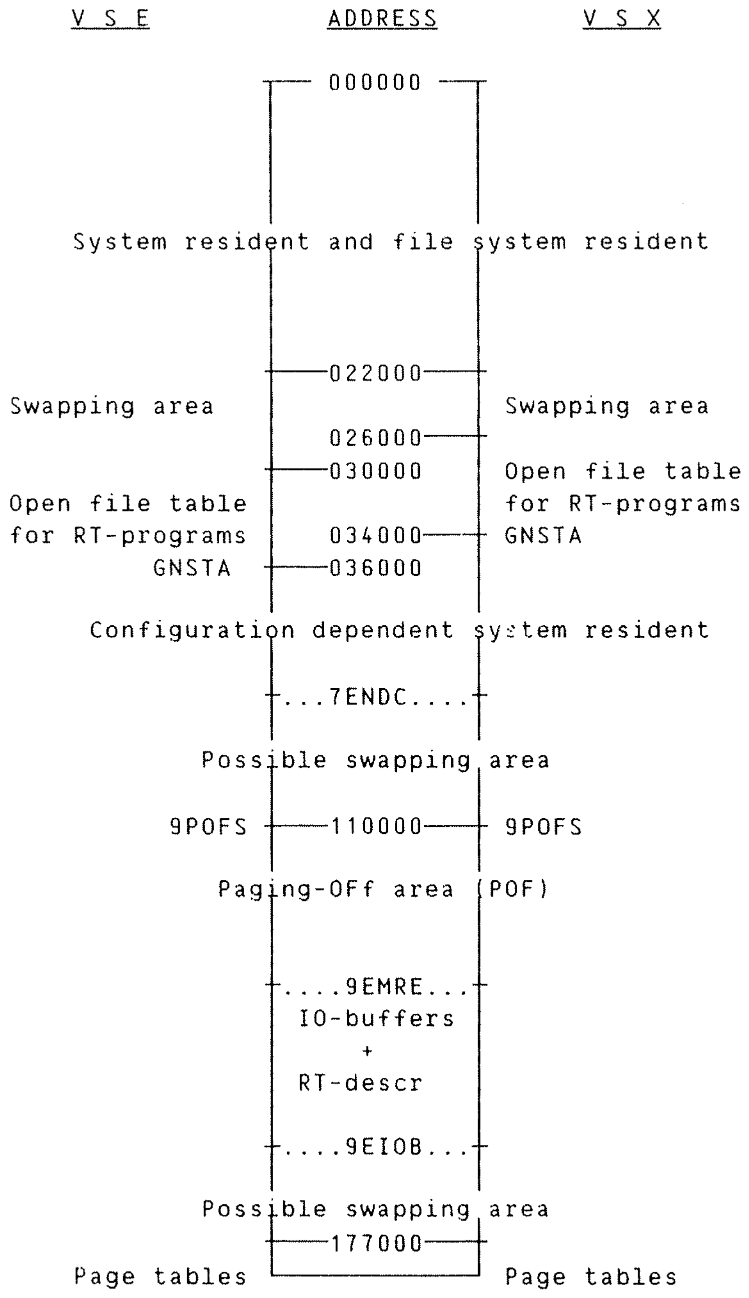
N O T E !

S I N T R A N   III   V E R S I O N **H**   I S   T H E   L A S T

V E R S I O N   A V A I L A B L E   O N   N O R D - 1 0

1. SINTRAN III J-VERSION SYSTEM LAYOUT

1.1 PHYSICAL MEMORY



1.2 PAGE INDEX TABLE 0

```
V S E                        ADDRESS      V S X

System resident          +---000000---+   System resident
                         +---002000---+
File system resident     |            |   File system resident
                         +---006000---+
System resident          |            |   System resident
(config. independent)    |            |   (config. independent)
                         |            |
Device buffer window     +---022000---+
                         +---024000   |   System segment
System segment           |            |
                         |   034000---+
                         +---036000   |
                         |            |   System resident
System resident          |            |   (config. dependent)
(config. dependent)      |            |
                         +...7ENDC....+
Possibly free            |            |   Possibly free
                         +---110000---+
                         |            |
Segment area             |            |   Segment area
(file sys. segms,        |            |   (file sys. segms,
command segm,            |            |   command segm,
RT-Loader etc.)          |            |   RT-Loader etc.)
                         |            |
                         |            |
                         +---174000---+
User window              |   176000---+   User window
                         +------------+   Window for backg.term. datafield
```

All resident pages are mapped physical page equal to logical page.

1.3 PAGE INDEX TABLES 1 AND 2

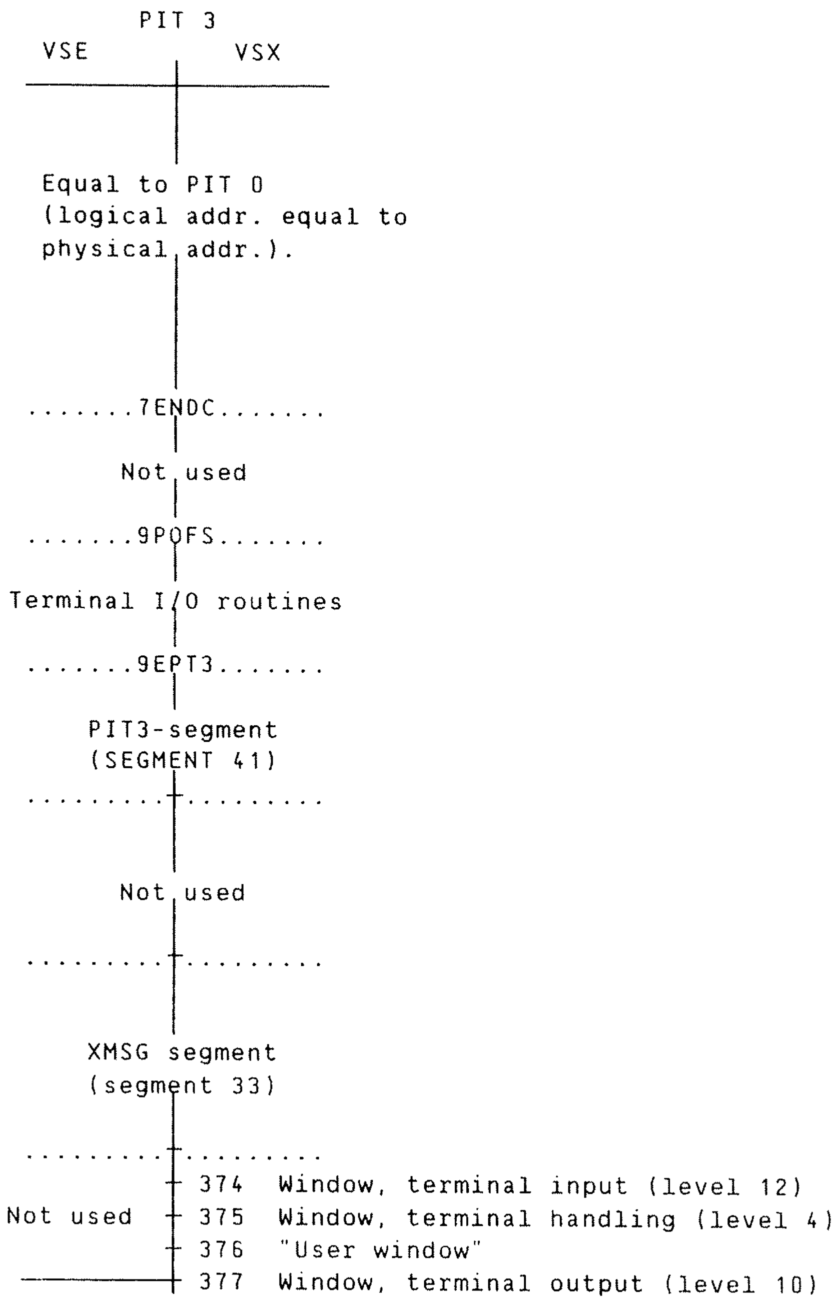
PIT 1 (VSE & VSX)

RT-PROGS.	BACKGROUND
Program and data bank.	Program bank when running 2-bank. Program and data when running 1-bank.
.... RTCOMMON (demand)	

PIT 2 (VSE & VSX)

RT-PROGS.	BACKGROUND
Normally not used (can be used for program and data).	Data bank when running 2-bank

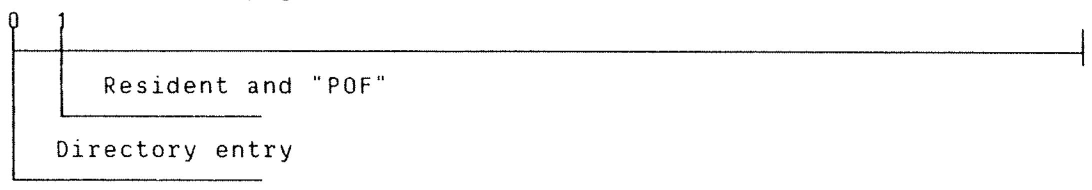
1.4 PAGE INDEX TABLE 3



1.5 SYSTEM LAYOUT ON DISK

1.5.1 SINTRAN:DATA

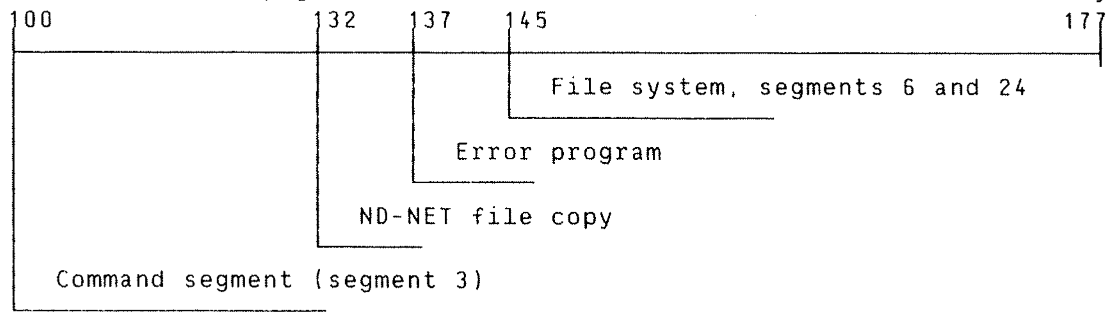
Disk address in pages



1.5.2 MACM-AREA:DATA

1.5.2.1 LAYOUT

Disk address in pages (octal), relative to the start of the directory

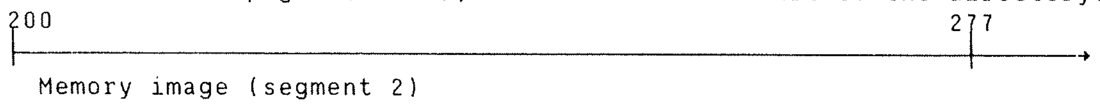


1.5.2.2 DISPLACEMENTS WHEN PATCHING

Command segment:	-110000
File system seg. 6 + 24:	2000

1.5.3 SEGFILO:DATA

Disk address in pages (octal), relative to the start of the directory.



Other segment files may reside in any directory at any disk address. The maximum size of a segment file is 16383 pages. Due to limitations in the RT-Loader, the sum of the used segment files cannot be greater than 32768 pages.

1.6 SINTRAN III SEGMENTS

1.6.1 SYSTEM INCLUDED SEGMENTS

Note: Segments 2-43 will be given standard segment names the first time the RT-Loader is entered.

SEG. NO.	SEG. NAME	ADDRESS RANGE	PT	DESCRIPTION
2	S3IMAGE	0 - 175777	1	Memory image & POF
3	S3COM	110000 - 173777	0	Command segment
4	S3RTL	110000 - 147777	0	RT-Loader program segment
5	S3ERRS	22000 - 25777	0	Error program "system segment"
6	S3FSCOM	110000 - 137777	0	File system common segment
7	S3DMAC	110000 - 153777	0	DMAC segment
10	S3RTFIL	0 - 177777	2	RTfil segment
11	S3ERRL	0 - 17777	1	Error log segment
12	S3FS2SV	140000 - 173777	0	Initial reentrant filesys seg. no.1
13	S3RTLSV	56000 - 147777	1	Initial RT-Loader (RT-l. save-area)
14	S3ERRP	110000 - 123777	0	Error program segment
15	S3SMSV	110000 - 173777	0	Initial SINTRAN-SERVICE-PROG & MAIL
16	S3NNSV	150000 - 167777	0	Initial NORD-net segment
17	S3PT3SV	0 - 11777	1	Initial PT3 segment
20	S3SDT5	0 - 177777	2	ND-500 standard domain table
21	S3NMS5	0 - 177777	2	ND-500 table segment (name segment)
22	S3RFUS1	110000 - 163777	0	Reentrant file user segment no. 1
23	S3SMSEG	110000 - 173777	0	SINTRAN-SERVICE-PROGRAM and MAIL
24	S3FSRS1	140000 - 173777	0	File system reentrant segment no. 1
25	S3FSRS2	140000 - 173777	0	File system reentrant segment no. 2
26	S3RFUS2	110000 - 163777	0	Reentrant file user segment no. 2
27	S3NDNET	150000 - 167777	0	NORD-net segment
30	S3SM5S1	122000 - 173777	0	ND-500 system monitor segment no.1
31	S3SM5S2	122000 - 173777	0	ND-500 system monitor segment no.2
32	S3RTACC	110000 - 127777	0	RT-accounting segment
33	S3XMSGP	146000 - 167777	2	XMSG POF segment
34	S3XMSGD	0 - 35777	2	XMSG demand segment (XROUT)
35	S3XMSGR	0 - 37777	2	Reserved for XMSG
36	S3TAD	110000 - 133777	0	Tadadm segment
37	S3RTD	0 - 177777	1	RT-Loader data segment
40	S3FUDRT	164000 - 173777	0	File user data segment for RT-prog.
41	S3PT3	116000 - 135777	1	PT3 segment
42	S3SPLSV	110000 - 137777	0	Initial spooling program segment
43	S3SPL	110000 - 137777	0	Spooling program segment

1.6.2 OTHER SEGMENTS USED, DEPENDENT UPON THE CONFIGURATION

Spooling	:	1 segment for each spooling process (queue)
Background	:	2 segments for each background process
ND-500	:	2 segments for each ND-500 process + 1 "when no process to communicate with"
ND-100 SYMBOLIC DEBUGGER	:	1 segment for the debugger program and 1 segment for each background process using it

Remote file access
data segment: max. 64 segments. The number of segments equals the number of users who can use remote file access simultaneously

Maximum number of segments: 255

1.6.3 SYSTEM INCLUDED RT-PROGRAMS

PROGRAM	PURPOSE
BPTMP	Timeout program for background allocation system
1SWAP	Queuing program requests for swapping
5SWAP	Performs ABSTR in ND-100 for the 500-swapper
ACCRT	RT accounting
BAKnn	Background process for terminal (BAK01-BAK99)
BKnnn	- " - (BK100-BK128)
BCHnn	Batch process
BCxy	ND-net. Background process for remote terminal (x=Channel no., y=Line number)
COSPO	COSMOS-spooling server
CRxy	Read/write via ND-net channels without background
CWsy	(x=Channel, y=Line). DFxy and DRxy datafields.
DUMM2	Dummy program used by the spooling system
DUMMY	Dummy program to prevent empty execution queue
FDRT1	Transfer data between interface buffer and memory. Floppy formatting. (FLOPPY-1)
FDRT2	Transfer data between interface buffer and memory. Floppy formatting. (FLOPPY-2)
FIXRT	Monitor call/command FIXC execution
FSART	Administration of the file servers (COSMOS - remote file access)
RCOMn	Receive program for line number n. (ND-net)
RTDIL	Buffer transfer program for DISC-ACCESS-LOG
RTERR	Output error messages
RTRFA	Does remote file access for RT-programs (COSMOS - remote file access)
RTSLI	Time slicer. Changes priority on background processes.
RUxy	RT-program to handle output for a remote-connected terminal. (x=Channel, y=Line). (ND-net)
RWRT1	Block data transfer. Activated from RFILE, WFILE, RPAGE, WPAGE
RWRT2	Open file from RT-programs
RWRT3	Block transfer on MAGTAPE-1 (MAGTP)
RWRT5	VERSATEC-1 DMA
RWRT6	CDC-DMA LINK
RWRT7	MAGTAPE-2
RWRT8	VERSATEC-2 DMA
RWRT9	FLOPPY-DISC 1
RWRT10	FLOPPY-DISC 2
RWRT11	LINE-PRINTER/VERSATEC -1 I/O
RWRT12	LINE-PRINTER/VERSATEC -2 I/O

RWRT13	Block oriented internal device 1 INPUT
RWRT20	Block oriented internal device 1 OUTPUT
RWRT14	Block oriented internal device 2 INPUT
RWRT21	Block oriented internal device 2 OUTPUT
RWRT15	Block oriented internal device 3 INPUT
RWRT22	Block oriented internal device 3 OUTPUT
RWRT16	Block oriented internal device 4 INPUT
RWRT23	Block oriented internal device 4 OUTPUT
RWRT17	Block oriented internal device 5 INPUT
RWRT24	Block oriented internal device 5 OUTPUT
RWRT25	HASP DMA 1 INPUT
RWRT26	HASP DMA 1 OUTPUT
RWRT27	HASP DMA 2 INPUT
RWRT28	HASP DMA 2 OUTPUT
RWRT29	HASP DMA 3 INPUT
RWRT30	HASP DMA 3 OUTPUT
RWRT31	HASP DMA 4 INPUT
RWRT32	HASP DMA 4 OUTPUT
RWRT33	HASP DMA 5 INPUT
RWRT34	HASP DMA 5 OUTPUT
RWRT35	HASP DMA 6 INPUT
RWRT36	HASP DMA 6 OUTPUT
SCOMn	Send program for line number n. (ND-net)
SPRTn	Spooling programs (1-9)
SPRnn	Spooling programs (10-30)
STSIN	Initialize SINTRAN III and start systems RT-programs
TADnn	Terminal Access Device
TADAD	Administrates connections to TADs from requesting users.
TERMP	Starts the user defined "clean-up" RT-program when RT-programs are aborted (if enabled)
TIMRT	Timer RT-program. Start timeout-routine for all devices in timer-table.
XROUT	XMSG routing program
XTRACE	XMSG trace program
XFTRA	COSMOS file transfer server

2. INSTAL TAD/TADADM

TAD/TADADM is now removed from the COSMOS Basic Module and included in SINTRAN III as a standard option. The desired number of TADs must be specified in the SINTRAN III order form. During installation of your system, the NEW-SYSTEM program will copy a new TADADM to your disk. User and file names are:

```
(SYSTEM)COS-TADADM:BPUN
```

Your HENT-MODE file should include the commands:

```
@RT-LOADER
READ-BINARY COS-TADADM 36
YES
END
EXIT
```

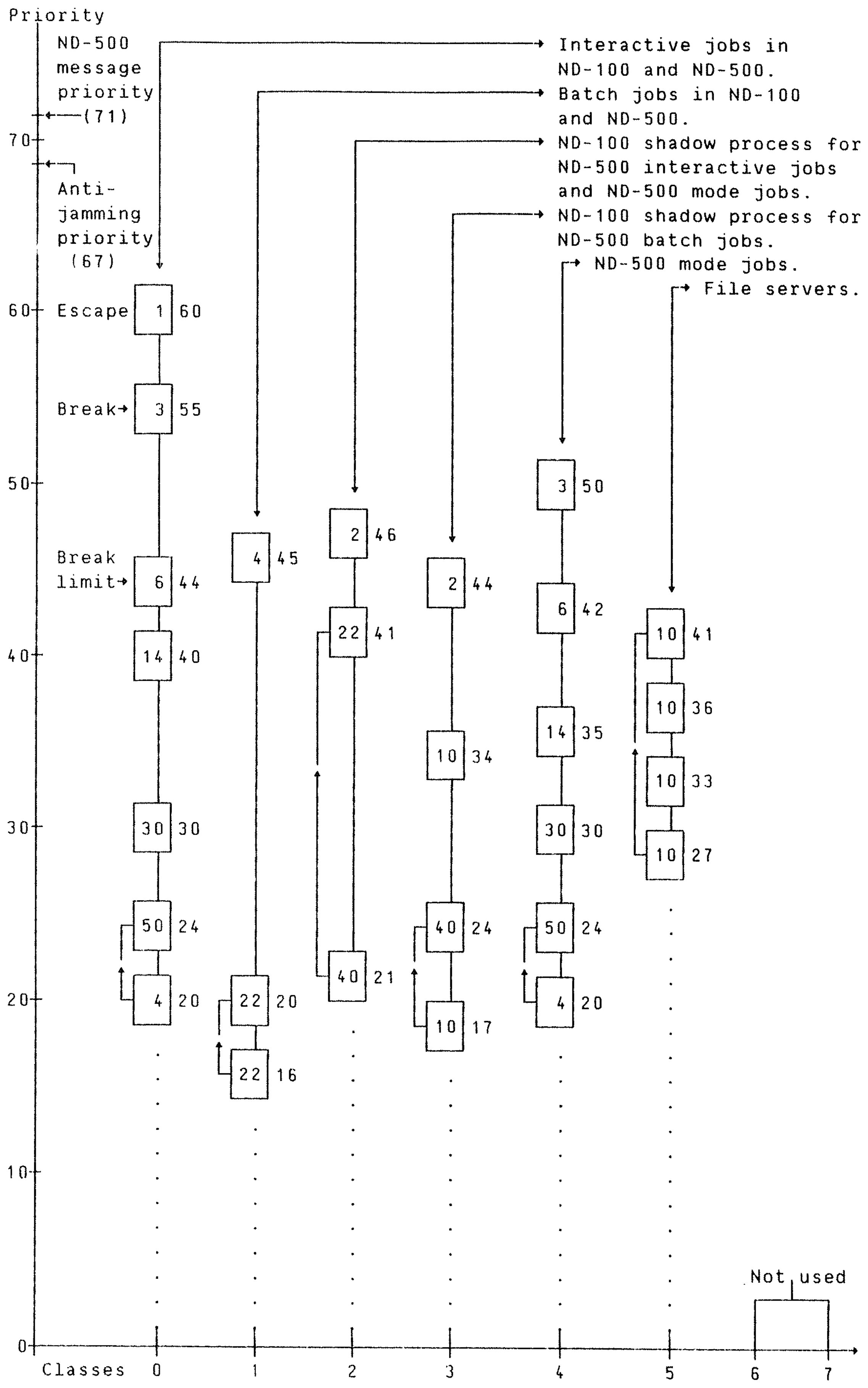
This ensures that TADADM is loaded at cold start. In addition the command: @START-TADADM should be included in your LOAD-MODE file.

3. TIME SLICING

A resident RT-program, called RTSLI, changes the priority of the processes according to the CPU-time they use, in order to share the CPU-resource between them. This program is called the time slicer program. By default all ND-100 background programs, including batch, and all ND-500 processes, are time sliced. RT-programs in ND-100 cannot be time sliced. The priority of a process and the CPU-time the process can consume on that priority, are defined in the time slice class the program belongs to. There are 8 (0-7), time slice classes in the system, of which classes 0-5 are used. Classes 6-7 are not used. However, these can be defined and used with the @SINTRAN-SERVICE-PROGRAM command *DEFINE-TIME-SLICE. The following figure shows the usage of time slice classes 0-5, the different priority levels, and the number of time slice units of CPU-time to consume on the various priority levels.

A process running in ND-500 has two priorities, one in ND-500 and another in ND-100. The same algorithm is used for time slicing in ND-500 as in ND-100, except that ND-500 CPU-time is used instead of ND-100 CPU-time. The ND-100 priority of an ND-500 process is used when the shadow process executes monitor calls, or gives other services, on behalf of the ND-500 process, in the ND-100.

The figure on the next page illustrates the time slice mechanism. Note that all numbers are in octal format.


Priority
ND-500 message priority
(71)
70
Anti-jamming priority
(67)
Interactive jobs in ND-100 and ND-500.
Batch jobs in ND-100 and ND-500.
ND-100 shadow process for ND-500 interactive jobs and ND-500 mode jobs.
ND-100 shadow process for ND-500 batch jobs.
ND-500 mode jobs.
File servers.
60
Escape
1 60
Break
3 55
50
3 50
Break limit
6 44
4 45
2 46
2 44
6 42
22 41
10 41
40
14 40
10 36
14 35
10 34
10 33
30
30 30
30 30
10 27
50 24
40 24
50 24
40 21
20
4 20
22 20
4 20
10 17
22 16
10
Not used
0
Classes
0
1
2
3
4
5
6
7

<u>More about time slicing.</u>

The time slicer program, RTSLI, is running at an interval of 0.5 seconds. One time slice unit is equal to 240ms CPU-time, (ND-500 or ND-100 CPU-time). On the figure on the previous page, the number of time slices on each priority level is given inside the squares, the priorities themselves to the right of it. In interactive mode, a process is reset to "break-priority" (55_8), if the actual priority is less than 44_8 when a break-condition is met. Also in interactive mode, a process is reset to "escape priority", (60_8) if an escape is typed when escape is enabled. Due to the fact that the time slicer program is running at an interval of 0.5 seconds, and that the priority is only changed by the time slicer program, a delay of max. 0.5 seconds can occur when typing a break-character or the escape-character, before the priority is raised to the break-priority or to the escape-priority.

Each time a time sliced program is started (logging-in, entering a batch job, etc.) the time sliced program will start on the highest priority level of the time slice class the program belongs to. Time sliced programs always move steadily downwards to the lowest priority of the time slice class, except from break-conditions and escape-conditions in interactive mode. When reaching the lowest priority of a class, the time sliced program will get the "next-to-lowest" priority, and then the lowest priority again. The time sliced program will change between the lowest and next-to-lowest priority (until the program is finished or an escape or break-condition occurs). The only exception to this rule is programs running in time slice class 5 (file servers), which will change from lowest to highest priority and then start looping through the four priority levels for this class.
This is illustrated by the back-arrows on the figure.

There are two special priorities used on time sliced programs; one called anti-jamming priority, (67_8), and another called ND-500 message priority (71_8). The anti-jamming priority is given to time sliced programs which have reserved a system resource, for which another program with higher priority than the program holding the system resource is waiting. A system resource is defined as a datafield, semaphore, with the protection ring value set equal to 2 or 3 in the TYPRING location in the datafield.

After the program with the anti-jamming priority has released the last system resource, for which other programs with higher priority are waiting, the program gets the same priority as it had when it was "anti-jammed". The anti-jamming priority is used only in ND-100. Setting anti-jamming priority will have immediate effect - it will not wait for the next execution of the time slicer program (RTSLI).

The ND-500 message priority is used on ND-500 processes when the shadow-processes send messages to the ND-500 Swapper or to the ND-500 microprogram. Example of such messages are: examine or deposit ND-500 registers, write or read monitor call parameters. The ND-500 message priority is used only in ND-500. Setting ND-500 message priority will have immediate effect - it will not wait for the next execution of the time slicer program (RTSLI).

In ND-500, the priority of a process is increased by one for each monitor call executed by the process, if the current priority is less than or equal to the next-to-lowest priority of the class. This is done to distribute resources evenly between heavy jobs.

A special algorithm is included to avoid situations where the time slicing becomes "too stable". This algorithm checks the number of time slices to be consumed at the current priority level. If this number is greater than or equal to 22_8, a number in the interval $0-17_8$ is added to the number of time slices that can be consumed at the actual priority level. The maximum number of time slices to be consumed before adding a value in the range $0-17_8$ (the limit 22_8 above), can be changed with the @SINTRAN-SERVICE-PROGRAM command *DEFINE-TIME-SLICE. This parameter is called <Lowest time count before getting hashed>. The range of the value to add can also be changed by the same command. The parameter for this is called <Bit mask used when hashing>.

This time slicing mechanism was included as standard in the I-version of SINTRAN III, and only small adjustments have been made for the J-version. However, time slice class number 4 for ND-500 mode jobs, and time slice class number 5 for file servers in ND-100, are new features in the J-version.

4. SECURITY PRIMITIVES

In the J-version of SINTRAN III the following security primitives are default, specified in the variable named EXSECURITY (the bits are set to one).

Bit #0: No listing of command lines in the @TERMINAL-STATUS command. If the command is performed by user SYSTEM, the command lines for all background programs logged in will be listed. The command lines will also be listed for the background programs running under the same user as the one executing the @TERMINAL-STATUS command.

Bit #1: The background segment, both program and data bank, will be set to zero when logging out. This feature will delay the log-out sequence considerably (seconds). If the background program was terminated abnormally, this zeroing will take place when you log in the first time after the abnormal termination.

Bit #2: The scratch file pages written to in the last session, will be set to zero when logging out. This will slow down the log-out sequence.

The following security primitives are optional, specified in the variable named EXSECURITY (the bits are zero).

Bit #3: Zeroing of pages released from a file, normally in the @DELETE-FILE command.

Bit #4: Not allowed to log in if the user has no password. It is legal to log in only once after @CREATE-USER without password.

The value of the variable EXSECURITY can be changed by the @SINTRAN-SERVICE-PROGRAM command *CHANGE-VARIABLE.

5. ND-100 SWAPPING

There are four different ways of queuing programs requesting the "ND-100" swapper. The variable named SWPFLAG is used to distinguish between the four possibilities.

SWPFLAG=0: No queuing of requests. If the swapping semaphore or the mass storage datafield which a page is to be transferred to/from is occupied when a program requests the swapper, this program is skipped in the execution queue, and the next is started. The program requesting the swapper will still be marked as ready in the execution queue. It may be started several times, without managing to execute any instructions due to page faults and no service from the swapper. This can cause much overhead in a system with heavy swapping. All programs requesting the swapper while the swapper or the mass storage is occupied, will be treated in the same way. SWPFLAG=0 is equal to the mechanism which was standard in SINTRAN III before the I-version.

SWPFLAG=1: This value is default in the I-version. If a program which has reserved the swapper must wait for the mass storage to tranfer a page, a special resident program named 1SWAP is given the swapper and is linked into the waiting queue for the mass storage. (The mass storage is occupied with other transfers, file system transfer, ND-500 swapping etc.) The priority of 1SWAP is 100_8. The program that originally reserved the swapper will be put in a "waiting-for-swapper" state to tell the monitor not to start this program before the swapper is released. A program is marked as "waiting-for-swapper" by setting bit 15_8 in the ACTPRI location in the RT-description. All other programs requesting the swapper while it is occupied, will be set in the same wait state. When the mass storage is reserved by 1SWAP, this program will release the mass storage datafield and the swapper. Then all programs in the execution queue in the "wait-for-swapper" state will be reset from this state, and the first program in the execution queue will be started.

SWPFLAG=2: This value of SWPFLAG gives almost the same function as SWPFLAG=1, except that with this value the program that reserved the swapper before the program 1SWAP was linked into the waiting queue for the mass storage, will be the first program to start when 1SWAP releases the swapper and the mass storage datafield.

SWPFLAG=3: This value of SWPFLAG is default in the J-version. As for SWPFLAG values 1 and 2, the program requesting the swapper will be marked in the execution queue as "waiting-for-swapper", and the program 1SWAP will be linked into the waiting queue for the mass storage. This time 1SWAP will have the priority equal to the priority of the program requesting the swapper. When 1SWAP is started it links the swapper and the mass storage datafield to the program which first requested the swapper while the mass storage was occupied. Then all programs in "wait-for-swapper" state will be reset from this state and marked as ready. The first program after 1SWAP in the execution queue, will now be started.

6. SINTRAN III MONITOR CALLS

6.1 MODIFIED MONITOR CALLS

6.1.1 BRKM (MON 4)

Two new functions have been added:

10B - Use "old" break 7 table, but update maxbreak (contained in D-register)

11B - Set new maxbreak, but do not change break-table

6.1.2 FIXC5 (MON 61)

Function= 7: Release all memory areas reserved by a specific program. Only memory areas reserved with Function=5 in MON FIXC5 will be released.

Function=10: Get a segment's fixed status.

<u>Monitor call format:</u>

```
LDA (PLIST
MON FIXC5
JMP ERROR
.........

PLIST, FUNC      % Pointers
       PAR1      %
       PAR2      % to
       PAR3      %
       PAR4      % parameters
       PAR5      %
```

<u>Function=7 : Release all memory areas reserved by a specific program</u>

<u>Input parameters:</u>

FUNC=	7
PAR1=	RT-program which has reserved the memory areas to be released. PAR1=0 means calling program.
PAR2=	Dummy (not used)
PAR3=	Dummy (not used)
PAR4=	Dummy (not used)
PAR5=	Dummy (not used)

Output parameters:

Error return:

A=3 : Illegal RT-program in PAR1.

OK return:

None.

Function=10: Get a segment's fixed status

Input parameters:

FUNC=	10
PAR1=	Segment number
PAR2=	Page number within the segment, the physical page of which should be found if the segment is fixed.
PAR3=	Dummy (not used)
PAR4=	Dummy (not used)
PAR5=	Dummy (not used)

Output parameters:

Error return:

A=174 : Illegal segment number in PAR1, or
illegal page number within segment in PAR2.
A=2 : Segment, in PAR1, not loaded.

OK return:

T= Fixed status
- T=0 : Segment not fixed.
- T=1 : Segment is fixed scattered.
- T=2 : Segment is fixed contiguously.

A= Physical page number where the page number, in PAR2, resides in memory.
D= Flag word in the segment table entry of segment specified in PAR1.

6.1.3 ABSTR (MON 131)

New functions for STC magtape:

Function 50:	Read multiple records, i.e. read a number of magtape records to a contiguous memory area in one ABSTR call.
Function 51:	Write multiple records, i.e. write a contiguous memory area as a number of magtape records to magtape in one ABSTR call.

Monitor call format:

```
LDT   LDN
LDA   (PLIST
MON   ABSTR

......

PLIST,FUNC          %  Pointers
      DMEM          %  to
      UNIT          %  parameters
      RECD          %
```

Input parameters:

```
T    = Logical device number
FUNC = Function code (6/50B/51B)
DMEM = Double word, physical memory address
UNIT = Unit number (0-3)
RECD = Double word:
RECD, RECNO = Number of records to read/write in one call
      RECSI = Record size in words
```

Parameters have to be in resident memory.

Output Parameters:

```
A ≥ 0 :  Transfer completed
         Bit 16 -0 contains hardware status
               8
         RECD   (RECNO) : No. of records read/ written
         RECD+1 (RECSI) : Record size of records read

A < 0 :  Error
         (A=-1 and FUNCTION=50 : Trying to read records of different
                                 sizes in one ABSTR call)
    Bit 16 -0 contains hardware status
          8
    RECD  (RECNO) : No. of records read/written before error occurred
    RECD+1 (RECSI) : Record size of records read
```

Rules:
As in ordinary ABSTR calls (see Ref. Manual)

Function 6 : Change retry counters to be used in functions 50/51

Monitor call format:

```
LDT    LDN
LDA    (PLIST
MON    ABSTR
     ......
PLIST,FUNC
      MEMA
      UNIT
      RECD
```

<u>Input parameters:</u>

T	= logical device number		
FUNC	= function code (6)		
UNIT	= unit number (0-3)		
MEMA	= not used by this function		
RECD	= double word:		
RECD, RETCO	New start value of retry counter	(Initially =-4)	
ERACO	New start value of erase counter	(Initially =-2)	

Involved locations in datafield:

	Symbol	displ.	Function
ERASE:	MACOU	-40	Erase gap counter
	MWCNT	-14	Start value of MACOU, number of times to write erase gaps
	CMWCNT	-54	Value of MWCNT in multiple record calls. Initially = -2; may be changed by this function
RETRY:	TACOU	-15	Retry counter
	TACNS	-16	Start value of TACOU, number of times to repeat call to driver
	CTACNS	-53	Value of TACNS in multiple record calls. Initially = -4, may be changed by this function.

TACNS indicates how many times a call to the mag.tape driver (SMAGT) should be tried in a read/write operation, i.e. number of retries. If a write operation did not succeed after the specified number of retries, an erase gap is written and a new series of write operations is attempted. The erase gap procedure is repeated as many times as MACNS indicates.

```
Do MACNS  times
      Do TACNS times
          Call driver
          If OK go out
      Enddo
      Write erase gap
Enddo
Go error
Out:....
```

For programming reasons, both types of counter contain negative values, i.e. -4 means repeat 4 times.

NOTE: As in most ABSTR parameters, there are no checks on the new values of these counters.

6.1.4 LAMU (MON 315)

Functions 7 and 8 are added, and the monitor call is available from background.

The LAMU System is intended to be an extension to the Segment Structure in SINTRAN III, making it possible for RT-programs and background programs to address more address space than covered by the available 3 segments, and to address memory shared by several CPUs.

A LAMU is a limited continuous logical and continuous physical address area. The size range of a LAMU is 1 page to 128 pages, and the logical address range covers page 100(8) to page 277(8) (page table 1 to page table 2).

Speed of LAMU setting/resetting: due to the fact that a LAMU must be continuous in logical and physical memory, the setting and clearing of the page index table will be very fast - faster than a segment with the same length.

A LAMU can be created and deleted by monitor call or by command in the SINTRAN-SERVICE-PROGRAM.

The logical page of a LAMU is defined at connect time.

Several RT-programs can use the same LAMU simultaneously.

A LAMU is identified by a number returned from the system when the LAMU is created, or specified in the "create LAMU" call.

The same physical page can exist in several LAMUs at the same time.

A physical page must be removed from the SINTRAN III swapping pages before it is used in a LAMU.

A LAMU requires no space on the segment files.

The following parameters will be used in the description below.

<func>	: Will specify the actual operation on the LAMU.
<LAMU id>	: Is the LAMU identifier
<Prog>	: The program a LAMU is connected to/disconnected from, 0 means own program.
<Size>	: Is the number of pages in the LAMU
<Prot>	: LAMU protection and ring
<Laminf>	: 3-word array used for LAMU information
<Log.addr.>	: Specifies the first logical address covered by the LAMU. Legal range: 100_8 - 277_8
<Phys.addr.>	: Specifies the start of the physical memory the LAMU will cover. Legal value: All existing physical pages currently defined as legal LAMU areas.

Correct execution of MLAMU is reported by skip return. Errors are reported by normal return, and the A-register will then contain the error code.

Create LAMU

```
<func>        = 1
<LAMU id>     = 0  : The system will return the selected <LAMU id>
              ≠ 0  : The LAMU will be identified by the given <LAMU id>
                     if the <LAMU id> is unused and inside legal range.
<size>             : Number of pages in the LAMU. Legal range 1-200₈
<phys.addr> = 0    : The system will select the first LAMU memory area
                     large enough for the LAMU, and not used by other
                     LAMUs. The first physical page number of this
                     memory area will be returned in <phys.addr>.
              ≠ 0  : Specifies the first physical page for the LAMU.
```

Legal value:
All existing physical pages currently used as LAMU areas.

Rules:
Legal from user SYSTEM and RT-programs.

Delete LAMU

```
<func>     = 2
<LAMU id>        : Specifies the LAMU to be deleted
<dummy>
<dummy>
```

Rules:
The LAMU will be deleted if it is not in use by other RT-programs.
The physical pages used in the LAMU will remain LAMU area.
Legal from user SYSTEM and RT-programs.

Connect LAMU

```
<func>  = 3
<LAMU id>    : The specified LAMU will be inserted in the address
               area of the calling RT-program.
<prog>       : Program connecting the LAMU to.
<prog> =0      Means calling program.
<log.addr.>  : First logical page used to access the LAMU.
               Legal range 100₈ - 277₈ .
```

Rules:
The maximum number of LAMUs an RT-program can have connected to it at the same time is decided before system startup time. A connected LAMU must not overlap, in logical address area, any other connected LAMU or RT common. The connected LAMU will be disconnected by the Disconnect LAMU function in MLAMU, or by the RT-Loader when the RT-program is deleted. There will be no disconnection of LAMUs when an RT-program is aborted.

Disconnect LAMU

```
<func>     =  4:
<LAMU id> = -1: Disconnect all connected LAMUs of the RT-program
          ≠ -1: Disconnect the specified LAMU
<prog>        : Program which LAMUs are to be disconnected from.
                <prog>=0  means calling programs.
<dummy>
```

LAMU pages to nothing

```
<func>          =  5
<first physical page>
<no. of pages>
<dummy>
```

Rules:
In a configuration with multiple CPUs connected to the same MPM, it is assumed that all the CPUs start up with the entire MPM as LAMU area. By this function a CPU excludes an area from both LAMU and swapping area, and another CPU can use this area as swapping area. The area must be within a LAMU area.
Legal from user SYSTEM and RT-programs.

LAMU pages to swapping

```
<func>          = 6
<first physical page>
<no. of page>
<dummy>
```

Rules:
After this function, the specified LAMU pages can be used as swapping pages. Legal from user SYSTEM and RT-programs.

Protect LAMU

```
<FUNC>     = 7
<LAMU id> : LAMU identifier
<PROT>    : Bits 9-10   = Ring
            Bit 13 set = Fetch Permitted
            Bit 14 set = Read  Permitted
            Bit 15 set = Write Permitted
<Dummy>
```

Rules: Legal from user SYSTEM and RT-programs.

LAMU information

```
<FUNC>     = 8
<LAMU id> : LAMU identifier
<LAMINF>  : First physical page
            Number of pages in LAMU
            LAMU protection
<Dummy>
```

Calling MLAMU

```
             LDA  (PLIST
             MON  MLAMU
             JMP  ERROR                % ERROR, A = error code
             JMP  OK                   % OK Return


PLIST, FUNC
       LAMID
       PROG
       LOGAD

FUNC,  3                               % connect LAMU
LAMID, 12                              % LAMU # 12
PROG,  0                               % calling program
LOGAD, 140                             % logical page no.
                                       % page 40 on page table one
```

or

```
PLIST, FUNC
       LAMID
       PROT
       DUMMY

FUNC, 7                                % Protect LAMU
LAMID, 12                              % LAMU # 12
PROT, 161000                           % RFW Ring 1
DUMMY, 0
```

6.2 NEW MONITOR CALLS

6.2.1 FSMTY (MON 327)

File system multifunction monitor call.

Input parameters:

```
      T-REG =  Function code

               T=1  WRBIX
               A-REG: Open file number

  Return: A-REG: Error code
  Skip-return:   OK
```

This forces the writing, to disk, of modifications to the datafield index blocks which may not yet have been written back (i.e. where changes have only been made in the index blocks in the open file entry). This leaves the file consistent in the event of an uncontrolled system stop with the file still open. It will be particulary useful for SIBAS and ISAM applications.

6.2.2 TERST (MON 330)

Monitor call to get information similar to that of the command @TERMINAL-STATUS, but for one terminal (device) only.

Input parameters:

```
    T-reg = Logical Device Number
    A-reg = Memory address to buffer where information is returned
            Words   0-7: User  name of logged in user, terminated
                         by  "'"  if less than 16 characters.
                    10:  Mode :  1 = Command
                                 2 = User
                                 3 = RTWT
                                 4 = Hold
                    11:  State : -1 = Passive
                                  0 = Idle (batch only)
                                  1 = Active
                    12:  CPU minutes used
                    13:  Login time (minutes)
                 14-25:  Last  command  (terminated by single quote)

  Return: A-REG = Error number.
  Skip-return:    OK
```

6.2.3 TREPP (MON 332)

Monitor call to get terminal-related information and to control termination of programs.

Input parameters:

```
    T = Logical device number (1: own terminal background)
    A = Function  = 0: disable terminal report
                  = 1: enable  terminal report
                  = 2: read    terminal status

  Return      :  A = Error number.
  Skip return :  Function 0: OK (no parameter)
                 Function 1: OK (no parameter)
                 Function 2: OK  A = Terminal status
                                 bit  0: Terminal line dead
                                 bit  1: Logout waiting for MON 0
                                 bit  2: Overrun in input buffer
                                 bit  3: Parity  error on input
                                 bit  4: Framing error on input
```

The bits 2-4 in status are cleared at read status. These bits indicate that some characters in the input stream may be lost.

When the enable function has been used, the program will not be terminated by line breaks or timeout.

6.2.4 UDMA (MON 333)

NOTE! Only included when ordered (library marks = 8UDMA 8UDnn).

Transfer data between memory and universal DMA interface (ND-852)

Monitor call format

```
LDT I   (LDN
LDA     (PAR
MON     333
STA     ISTAT
..........

PAR,    IFUNC
        MARRY
        DPARI
        DPARO
```

Input parameters:

T-reg:	Logical device number of DMA channel.
A-reg:	Address to parameter list.
IFUNC:	Function code, see below.
MARRY:	Single word, logical memory address of data.
DPARI :	Double word, function dependent, see below.

Output parameters:

DPARO :	Function dependent, see below. Double word (32-bit)
A-reg:	=0 if OK, else SINTRAN III error code.

IFUNC (oct)		MARRY	DPARI	DPARO
0	DMA input	x	Length	Actual length
1	DMA output	x	Length	-
2	Same as 0 but with no wait	x	Length	
3	Same as 1 but with no wait	x	Length	-
7	Test mode	-	-	Status
20	Read status	-	-	Status
21	Clear device	-	-	-
24	Read last status	-	-	Status
54	PIO input without interrupt	PIO D.	us.co.lin.	-
55	PIO output without interrupt	PIO D.	us.co.lin.	-
56	PIO input with interrupt	PIO D.	us.co.lin.	-
57	PIO output with interrupt	PIO D.	us.co.lin.	-
62	Wait on interrupt/DMA finish	-	1 / 0	Status
64	Enable attention interrupt	-	-	-
65	Disable attention interrupt	-	-	-
70	Write user control lines	-	Us.co.lin (Bit 8-15)	-

-	Parameter not used
x	Parameter used as usual

Rules:

1) Callable from background and RT
2) The output part of LDN must be reserved (for all functions), do not reserve the input part.
3) In status, bits 0-7 and bit 15 is copied from hardware status word, bits 8-14 are user status lines, bit 16 means attention interrupt has occured and bit 17 is a time out flag.
4) IFUNC=54,55,56,57: Bits 8-15 in DPARI will be written to user control lines together with the last command to the control word. These bits are not saved, and will be cleared in the next operation or interrupt to the interface.
5) IFUNC=62, IPARI=0 : Wait until DMA transfer is finished (if transfer is finished, there will be an immediate return). If attention interrupt is enabled, there will also be return on attention interrupt. The Hardware status should be examined.
   IF DPARI=1 : Wait on attention interrupt. If DMA transfer is initiated, there will also be return on DMA transfer complete interrupt. The Hardw. status should be examined.
6) If RT and the segment which holds MARRY is FIXCed, the DMA-data will be transferred directly into user address space.
7) Function code 2 and 3 only legal from ND-500, or if rule 6.

XX Datafields for 1 to 16 devices may be ordered with library marks (in addition to 8UDMA)

8UD01 - 8UD16	for Universal DMA or
8VI01 - 8VI16	for Vicom DMA.

It is possible to mix 8UDxx and 8VIzz (8UD01;8VI02;8VI03)

6.2.5 GETXM (MON 334)

Purpose: Get error-message text.

Monitor call format:

```
        LDA ECODE          % A=error code
        LDX (BUFFR         % X=address of buffer for text
        MON GETXM
        JMP ERROR          % Error return
        .........

ECODE, 10
BUFFR,0; *+100/
```

Input parameters:

A-reg = Frror code for which error-message text is wanted.
The error code is one of the normal "file system" error codes used in MON 64 and MON 65.

X-reg = Address of buffer to receive the error-message text.
This buffer must have a minimum size of 100_8 words.

Output parameters:

Return:	Error, A-reg= error code
Skip return:	OK, the error message text, terminated by a ' is returned into the specified buffer.

Rules:

Callable from background and from foreground (RT-programs).

6.2.6 EXABS (MON 335)

Purpose: Execute Mon ABSTR from programs on PT 1,2 or 3.

Parameters are as for ABSTR, but the last three parameters are double integers. If the ABSTR function to be used is expecting a single integer as parameter, that parameter value must be in the most significant part of the EXABS parameter.

The value of the last ABSTR parameter, after ABSTR is executed, is always written back into the last parameter in the EXABS parameter list, independent of the function code. The A-register will contain the same information after executing MON EXABS as after executing MON ABSTR.

Rules:

Rules are mainly as for ABSTR.
Programs calling EXABS may use any page table, and can use both demand and non-demand segments. Page faults in parameter fetching are legal.
Ring 2 is still necessary for both monitor call and parameters.
Memory areas used for DMA transfers must be specified by physical address, and must be fixed and contiguous.
Only 2 programs can execute MON EXABS simultaneously.

Monitor call format and parameter example:

```
     LDT   LDN
     LDA   (PAR
     MON   335
     JAN   ERROR

PAR, IFUNC
     DMEM
     IBLOCK
     NBLOCK

IFUNC,   0          %     READ
DMEM,    2;0        %     TO PAGE 200 (400000)
IBLOCK,  120000;0   %     FROM SECTOR 120000
NBLOCK,  1;0        %     ONE BLOCK
                    %
LDN,     1100       %     BIG DISK
```

6.2.7 IOMTY (MON 336)

The IO multifunction (IOMTY) monitor call is used to change some of the attributes of terminal input and output.

The L-register contains a function code from the following table.

Function code	Function
0	Set alternative unit 0. T= logical unit to become unit 0. A= 1 if translation to capital letters is wanted, 0 if not.
1	Reset unit 0 to own terminal.
2	Set break mode on alternative terminal. Parameters as MON BRKM.
3	Set echo mode on alternative terminal. Parameters as MON ECHOM.

Return: Error, A-reg = error code.
Skip return: OK.

Rules:

Callable from background only.
The monitor call can affect all terminals reserved by the calling background program.

6.2.8 SPCHG (MON 337)

The Segment and Page table CHaNge monitor call implements the same functions as the MCALL and MEXIT monitor calls, but with the extension that the normal and alternative page tables may be changed.

```
Parameters:
       D-reg bit 15 is set if MEXIT (mon 132) is wanted,
                it is reset if MCALL (mon 133) is wanted.

    if MCALL: D bits 2-3 contain new normal page table.
              D bits 0-1 contain new alternative page table.
              T like MCALL
                (points to address and segment numbers).

    return:   T and L like MCALL.
              D contains old page tables.

    if MEXIT: D contains D-reg from MCALL return
                (but bit 15 set).
              T and L from MCALL return
                (like MEXIT).
```

Rules:

This monitor call is only callable from RT.

Example:

```
        % ON INITIAL SEGMENT

        SAA 3               % PT 1 AND 2
        COPY SA DD          % IN D REGISTER
        LDT (SEGAD
        MON 337
CONTI,  ...
        ...

SEGAD,  SUBRO               % ADDRESS OF SUBROUTINE ON NEW SEGMENT
        100201              % SEGMENTS 200 AND 201 (OCTAL)

        % ON NEW SEGMENT

SUBRO,  STT SAVET           % SAVE T, L, AND D REGISTERS
        COPY SL DT
        STT SAVEL
        COPY SD DT
        STT SAVED
        ...
        ...
        LDT SAVED
        COPY ST DD
        BSET ONE DD 170     % SET BiT FOR MEXIT
        LDT SAVEL
        COPY ST DL
        LDT SAVET
        MON 337             % BACK TO ORIGINAL SEGMENTS (CONTI)
```

7. GPIB DRIVER

The driver for General Purpose Interface Bus can now be ordered together with SINTRAN III, as an option. The GPIB driver communicates with user programs (or GPIB-Monitor) through XMSG. The GPIB driver must be started/stopped by @SINTRAN-SERVICE-PROGRAM

Driver and datafields for 1 to 8 interfaces can be ordered with library marks 8GPI0 - 8GPI7.

8. OCTOBUS

This implementation of OCTOBUS supports a computer network in tight coupling, i.e. with shared memory. The OCTOBUS can be used for signalizing (kick) between the computers, but the data must be transferred through a common memory.

8.1 OCTOBUS PROGRAMMING DESCRIPTION

The OCTOBUS can be accessed both from RT-programs, SINTRAN III tasks and direct tasks. RT-programs uses the monitor call 324, and the SINTRAN III and direct tasks use a special system routine to access the OCTOBUS. The destination station (CPU) number is the physical station number for the specific destination OCTOBUS unit. This will probably be changed to a logical destination station number in the future. RT-programs have to reserve the OCTOBUS input part before using it. No reservation is necessary from SINTRAN III and from direct tasks.

The logical device numbers are: 2400_8 - 2477_8. It is possible to have up to four OCTOBUS units (interfaces) in each CPU system.

40_8 slots can be accessed in each OCTOBUS unit. Slot numbers 0 - 17_8 are reserved for direct tasks. Slot numbers 20_8 - 37_8 are accessible from both direct tasks and RT programs. The first logical device number in each unit corresponds to slot number 20_8 in that OCTOBUS unit.

OCTOBUS unit 0 - log.dev.nos	:	2400_8 - 2417_8	(slots no 20_8 - 37_8)
OCTOBUS unit 1 ----- " ----	:	2420_8 - 2437_8	(slots no 20_8 - 37_8)
OCTOBUS unit 2 ----- " ----	:	2440_8 - 2457_8	(slots no 20_8 - 37_8)
OCTOBUS unit 3 ----- " ----	:	2460_8 - 2477_8	(slots no 20_8 - 37_8)

The 6 following functions are implemented:

Function - 0	- kick
Function - 1	- hang in I/O wait to be kicked (RT-programs only)
Function - 2	- activate when kicked (RT-programs only)
Function - 3	- prepare for kick (SINTRAN III / direct tasks only)
Function - 4	- send status/control information (SINTRAN III / direct tasks only)
Function - 5	- read OCTOBUS status (check last kick)
Function - 6	- who am I (get own station / CPU number)

8.1.1 Examples of use

Kick:

```
      A-register     -  0 (kick)
      D-register     -  destination station (CPU) number
      T-register     -  RT-programs    : logical device numbers
                     -  SINTRAN III tasks : slot number
      X-register     -  SINTRAN III tasks only : OCTOBUS unit number.

      Return information:

      Skip return    - OK (A-register = 0)
      Error return   - A-register contains the error number (see table)
```

Hang in I/O wait to be kicked/activated when kicked:

```
      These functions are only legal from RT-programs
      A-register     - 1 (hang in I/O wait to be kicked)
                     - 2 (activate when kicked)
      T-register     - logical device number

      Return information:
      Skip return    - OK (A-register = 0)
      Error return   - A-register contains the error number (see table)
```

Prepare for kick:

```
      This function is only legal from SINTRAN III / direct task
      A-register     - 3 (prepare for kick)
      D-register     - address of code to be executed on OCTOBUS driver
                       level when a kick arrives at this slot
      T-register     - slot number
      X-register     - OCTOBUS unit number (0,1,2,3)

      Return information:
      Skip return    - OK (A-register = 0)
      Error return   - A-register contains the error number (see table)

      "Kicked code" on driver level will get the source station number
      returned in the A-register.
```

Send status / control information:

```
      This function is only legal from SINTRAN III / direct task
      A-register     - 4 (send status/ control information)
      D-register     - destination station (CPU) number
      T-register     - status/ control information
      X-register     - OCTOBUS unit number (0,1,2,3)

      Return information:
      Skip return    - OK (A-register = 0)
      Error return   - A-register contains the error number (see table)
```

Read OCTOBUS status (check last kick)

```
A-register    -  5 (read OCTOBUS status)
T-register    -  RT-programs   : logical device numbers
              -  SINTRAN III tasks : slot number
X-register    -  SINTRAN III tasks only : OCTOBUS unit number.

Return information:
Skip return   - OK (A-register = 0)
Error return  - last kick (if any!) is probably not successful,
                yet A-register contains the error number.

In both cases the following information is returned:

A-register    - error number (could be 0)
D-register    - transmit status last used for this slot
                or logical device number
X-register    - actual hardware transmit status
T-register    - actual hardware receive  status
```

Who am I (get own station / CPU number)

```
A-register    -  6 (who am I)
T-register    -  RT-programs   : logical device numbers
              -  SINTRAN III tasks : slot number
X-register    -  SINTRAN III tasks only : OCTOBUS unit number.

Return information:
Skip return   - OK (A-register = 0)
Error return  - A-register contains the error number (see table)

In both cases the following information is returned:

                A-register     -   error number (could be 0)
                T-register     -   actual CPU / station number
```

8.1.2 Possible error numbers

Table of possible error numbers returned from the OCTOBUS driver:
(Those marked with an asterisk are not related to any tasks or RT-programs. They are only written on the error device as a system error message)

OKRET	- 0	normal return with skip
	- 0 - 31	hardware control/status numbers written on the error device when transmitted/received
ILCPUNO	- 41	illegal destination station (CPU) number
OCTOTIMOUT	- 42	OCTOBUS timeout
ILFUNC	- 43*	function in receive data is wrong
NORES	- 44*	no RT-program has reserved the mailbox
ILHFUNC	- 45*	illegal basic control function
MULERR	- 46*	multibyte message with no header
ILSLOT	- 47	wrong slot (task) number
ILTSS	- 48	wrong transmit status
TRERR	- 49	transmit error (return error from "kick")
PTRERR	- 50	error in previous transmit (return from next "kick" or read OCTOBUS status)
NOTTR	- 51	current transmit not finished (return from "kick")
PRTRANS	- 52	previous transmit not finished (return from next "kick" or read OCTOBUS status)
NOKICK	- 53*	task is not prepared for "kick"
SCPC	- 54	send-control-package not implemented
WFU	- 55	wrong OCTOBUS function in transmit
NOINIT	- 56*	no "P-reg" is initiated for direct task
HWST	- 57*	hardware status
MULP	- 58*	multibyte package
MULPD	- 59*	multibyte package data byte
NORT	- 60	not legal for RT-programs
NOTAS	- 61	not legal for tasks

9. SINTRAN III COMMANDS

9.1 GENERAL

All commands requiring a segment number as parameter accept symbolic segment name or segment number in octal format. Segment names can consist of 1 to 7 characters. Legal characters are all alphanumeric and _ (underline). Segment names cannot start with a numeric character.

9.2 REENTRANT SUBSYSTEMS

The table of reentrant subsystems and ND-500 standard domains is increased to a maximum of 75 entries, when the average subsystem name length is less than 14 characters.

9.3 RENAMED COMMANDS

```
The commands:  LIST-OPENED-FILES,
               LIST-RTOPENED-FILES
               SET-PERMANENT-OPENED

are renamed to:  LIST-OPEN-FILES,
                 LIST-RTOPEN-FILES
                 SET-PERMANENT-OPEN
```

9.4 MODIFIED COMMANDS

9.4.1 @CHANGE-BACKGROUND-SEGMENT-SIZE

The first parameter is changed from terminal number to segment number/name.

9.4.2 @LOOK-AT

New area to look at: PHYSICAL
Purpose: Inspect/change contents of all physical memory in the system.
Addresses may be double integers when using @LOOK-AT PHYSICAL

9.4.3 @COLD-START

This command will have [<terminal no.>] as optional parameter. After the @COLD-START is performed the specified terminal will be the "console" to give further commands from. The "normal console" cannot be used until the command @INITIAL-BACKGROUND-PROGRAMS is executed and the user logged in on the "temporary console" has logged out. Then the "normal console" will take over as console again.

9.5 NEW COMMANDS

9.5.1 @UE-AUTOMATIC-LOGIN

```
Parameters:     ALL TERMINALS ?  <yes>/<no>
                ENABLE=1/DISABLE=0:
                TERMINAL NUMBER:
```

This command enables or disables the user environment automatic login facilities.

If automatic login is enabled on a terminal, an attempt to log in (by pressing "escape") on the terminal, will lead to automatic login on the user USER-ENVIRONMENT, and the UE-LOGIN program is started.

The command is restricted to user SYSTEM.

9.5.2 @CLEAR-BATCH-QUEUE

```
Parameter:     <BATCH NUMBER>
```

This command deletes all entries in the batch queue of the specified batch process.

The command is restricted to user SYSTEM.

9.5.3 @DEFINE-SPOOLING-FILE-MESSAGE

```
Parameters: <USER TEXT>
            <PRINTING MESSAGE INDEPENDENT OF SPOOLING CONDITION?>
```

This command was removed from the I-version and has been reinstalled. The command is available to every user and defines a text to be written to the error device whenever one of the terminal user's spooling queue files is to be emptied on a peripheral. The text given by the parameter will be used until another text is defined. If no text is wanted, a single quote (') must be used as parameter. The text is suppressed if the second or third parameters of @DEFINE-SPOOLING-CONDITIONS are NO and the value of the parameter <printing message independent of spooling conditions?> is also NO.

10. SINTRAN-SERVICE-PROGRAM

10.1 GENERAL

All @SINTRAN-SERVICE-PROGRAM commands are now legal from batch. All input-lines in batch and mode jobs in SINTRAN-SERVICE-PROGRAM commands must be started with the character @.

Example:

```
@SINTRAN-SERVICE-PROGRAM
@SET-CLOSED-SCRATCH-FILE-SIZE
@37D
@64D
@YES
@NO
@EXIT
```

The first parameter is logical unit number, 37_{10}, the next is size of closed scratchfile, 64_{10}, the third is an answer to question IMAGE?, YES, and the last is a NO as answer to the question SAVE-AREA?. The last line is the command EXIT to leave the SINTRAN-SERVICE-PROGRAM.

Symbolic segment names can be used in the same way in @SINTRAN-SERVICE-PROGRAM commands as described in chapter 9, SINTRAN III commands.

10.2 COMMANDS REMOVED

10.2.1 *DEFINE-BASIC-TIME-UNIT

10.3 MODIFIED COMMANDS

10.3.1 *SET-CLOSED-SCRATCH-FILE-SIZE

The default closed scratchfile size is increased to 64K. The closed scratchfile size may be set to a negative value, which means that the scratchfile should not be shrunk unless the number of unused pages (of user SCRATCH) is less than the given negative value. The scratchfile is then shrunk by the necessary number of pages to obtain the given number of unused pages. If SINTRAN III is generated with Background Allocation System, this command cannot be used on terminals with dynamic connection to a background program.

10.3.2 *SET-SPOOLING-DEVICE-NUMBER

This command can only affect MEMORY and SAVE-AREA. IMAGE and MEMORY are the "same area" for this function.

10.3.3 *INSERT-SPOOLING-HEADER

This command can only affect MEMORY and SAVE-AREA. IMAGE and MEMORY are the "same area" for this function.

10.3.4 *REMOVE-SPOOLING-HEADER

This command can only affect MEMORY and SAVE-AREA. IMAGE and MEMORY are the "same area" for this function.

10.3.5 *INITIALIZE-SYSTEM-SEGMENT

The first parameter is changed from terminal number to segment number/name.

10.3.6 *CHANGE-BUFFER-SIZE

This chapter is valid for SINTRAN III/VSX and VSX-500 only. This command can be used on terminals, with one restriction. The restriction is that the size of the datafields (input+output), plus the buffers (input+ouput), must fit inside one page. This gives the maximum size of the sum of input and output buffers to 3424_8 bytes. The default buffer size for terminals are 134_8 bytes in input and 270_8 bytes in output.

10.3.7 *INSERT-IN-TIME-SLICE

This command has got an additional parameter for specifying the time slice class the program shall use. The command format: *INSERT-IN-TIME-SLICE <log.unit>,<time slice class>

10.3.8 *DEFINE-TIME-SLICE

This command is rewritten to operate upon the new time slicer program. The command has the following format:

```
*DEFINE-TIME-SLICE
IMAGE? YES
SAVE-AREA? NO
CHANGE TIMESLICE PARAMETERS (YES/NO) (DEFAULT IS NO): YES
                                                      IMAGE
PRIORITY FOR OWNER OF SYSTEM RESOURCES WHICH
         ARE WAITED FOR BY OTHER PROGRAMS      (1B -  77B) /  67/:
NO. OF BASIC TIME UNITS IN ONE TIMESLICE UNIT  (1B - 400B) /  14/:
```

```
LOWEST PRIORITY BEFORE GETTING RAISED ON BREAK (1B -  70B) /  40/:
LOWEST TIME COUNT BEFORE GETTING HASHED        (1B - 400B) /  22/:
           BIT MASK USED WHEN HASHING          (1B - 177B) /  17/:

CHANGE TIMESLICE ELEMENTS (YES/NO) (DEFAULT IS NO)? YES
TIMESLICE CLASS                (0B -   7B) /      0/: 6
                                              IMAGE
ESCAPE ELEMENT FOR THIS CLASS (0B -  37B) /  30/: 30
BREAK ELEMENT FOR THIS CLASS  (0B -  37B) /  30/: 30

TIMESLICE ELEMENT TO CHANGE   (0B -  37B) /  30/: _

PRIORITY FOR THIS ELEMENT     (1B -  77B) /  10/: 10
TIME COUNT FOR THIS ELEMENT   (1B - 400B) /   2/: 2
POINTER TO NEXT ELEMENT       (0B -  37B) /  31/: 31
CHANGE NEXT ELEMENT (YES/NO) (DEFAULT IS NO): YES
TIMESLICE ELEMENT NO. 31
PRIORITY FOR THIS ELEMENT     (1B -  77B) /   6/: 6
TIME COUNT FOR THIS ELEMENT   (1B - 400B) /   4/: 4
POINTER TO NEXT ELEMENT       (0B -  37B) /  32/: 32
CHANGE NEXT ELEMENT (YES/NO) (DEFAULT IS NO): YES
TIMESLICE ELEMENT NO. 32
PRIORITY FOR THIS ELEMENT     (1B -  77B) /   1/: 1
TIME COUNT FOR THIS ELEMENT   (1B - 400B) /  30/: 30
POINTER TO NEXT ELEMENT       (0B -  37B) /  31/: 31
CHANGE NEXT ELEMENT (YES/NO) (DEFAULT IS NO):

MORE CLASSES (YES/NO) (DEFAULT IS NO): NO
```

The values listed in parentheses () are the legal value range of the parameter, and the value listed between slashes // is the current value, which is also equal to the default value of the parameter. The time slice classes 0-5 are used by the system, and two classes are then free to use. The free time slice elements to use are the elements number 30_8-37_8.

For more information about the time slicer program, see chapter 3, time slicing.

10.4 NEW COMMANDS

10.4.1 *LIST-TIME-SLICE-CLASS

The format of the command:

```
*LIST-TIME-SLICE-CLASS
TIMESLICE CLASS: 6
IMAGE OR SAVE-AREA: IMAGE
```

```
TIMESLICE    PRIORITY OF    NO. OF TIMESLICE
ELEMENT NO.   ELEMENT       UNITS ON ELEMENT
     30          10                2  (ESCAPE ELEMENT)
     30          10                2  (BREAK ELEMENT)
     31           6                4
     32           1               30
     31
```

All listed values are in octal format.

10.4.2 *LIST-TIME-SLICE-PARAMETERS

The format of the command:

```
*LIST-TIME-SLICE-PARAMETERS
IMAGE? YES
SAVE-AREA? NO
                                                     IMAGE
PRIORITY FOR OWNER OF SYSTEM RESOURCES
         WHICH ARE WAITED FOR BY OTHER PROGRAMS:       67
NO. OF BASIC TIME UNITS IN ONE TIMESLICE UNIT:         14
LOWEST PRIORITY BEFORE GETTING RAISED ON BREAK:        40
LOWEST TIME COUNT BEFORE GETTING HASHED:               22
BIT MASK USED WHEN HASHING:                            17
```

All listed values are in octal format.

10.4.3 *LIST-TIME-SLICED-PROGRAMS

The format of the command:
*@LIST-TIME-SLICE-PROGRAMS

```
*LIST-TIME-SLICE-PROGRAMS


LOG. DEV.  BACKG. PROGR.  TIMESLICE CLASS

      1        BAK01              0
     36        BAK02              0
     37        BAK03              0
     38        BAK04              0
     39        BAK05        .     0
     48     NO FIXED PROGR.       0
     49     NO FIXED PROGR.       0
     50     NO FIXED PROGR.       0
     51     NO FIXED PROGR.       0
     52     NO FIXED PROGR.       0
    768     NO FIXED PROGR.       0
    769     NO FIXED PROGR.       0
    770     NO FIXED PROGR.       0
    670        BCH01              1
    672        BCH02              1
```

10.4.4 *DEFINE-HDLC-BUFFER

Parameters: <LOGICAL DEVICE NUMBER>
<BUFFER SIZE>
Purpose: Allocate buffer for the specified HDLC interface.

10.4.5 *START-GPIB

Parameter: <CONTROLLER NO.>
The GPIB driver will clear the GPIB controller and set up XMSG communication (name a port and reserve message and buffer space)

10.4.6 *STOP-GPIB

Parameter: <CONTROLLER NO.>

10.4.7 *CHANGE-GPIB-BUFFERSIZE

Parameters: <CONTROLLER NO.>
<USER BUFFER SIZE (OCT)>
<DMA BUFFER SIZE (OCT)>

10.4.8 *CHANGE-TABLE

This is a command to operate upon the four tables

USER-RESERVED-DEVICE-NUMBERS
This table is used to define the hardware device number intervals that should not be accessed when the system is initialized at system start-up, or at system restart after powerfail. The hardware device number is the number used in the IOX and/or IOXT instructions. An element in this table is defined by the first and last hardware device number that should not be accessed when the system is initialized.

USER-RESERVED-MENORY-AREA
This table specifies the memory pages that should not be used by the system for system-tables, i.e memory-map, device-buffers, page-owner-tables etc. A memory area is specified by the first and last physical page to reserve. The memory area set up in this table is normal swapping area. Segment can be fixed in this area.

MEMORY-AREA-UNAVAILABLE-FOR-SWAPPING
This table specifies memory area that should not be used for swapping. It may be used for LAMUs or some other special use. A memory area is defined by the first and last physical page in the area.

MEMORY-AREA-INVISIBLE-FOR-THIS-SYSTEM
This table specifies memory area that the system should not access at all. This is implemented to take care of several CPUs accessing shared multiport memory. When one system is started, another system may have initialized the shared memory. The system starting should therefore not access this memory area, because the contents of the shared area will then be destroyed. A memory area is defined by the first and last physical page in the area.

The command has the following format:

```
*CHANGE-TABLE

USER-RESERVED-DEVICE-NUMBERS,
USER-RESERVED-MEMORY-AREA,
MEMORY-AREA-UNAVAILABLE-FOR-SWAPPING,
MEMORY-AREA-INVISIBLE-FOR-THIS-SYSTEM
TABLE: USER-RESERVED-DEVICE-NUMBERS

FUNCTION:
```

The CHANGE-TABLE command has the following sub-commands:

```
LIST-TABLE
CHANGE-TABLE
CHANGE-ELEMENT
DELETE-ELEMENT
INSERT-ELEMENT
CLEAR-TABLE
EXIT
HELP
```

Some examples of the use of this command:

```
*CHANGE-TABLE

USER-RESERVED-DEVICE-NUMBERS,
USER-RESERVED-MEMORY-AREA,
MEMORY-AREA-UNAVAILABLE-FOR-SWAPPING,
MEMORY-AREA-INVISIBLE-FOR-THIS-SYSTEM
TABLE: USER-RESERVED-DEVICE-NUMBERS

FUNCTION: INSERT-ELEMENT
IMAGE OR SAVE-AREA (DEFAULT IS IMAGE):
FIRST DEVNO. (OCT): 0
LAST DEVNO. (OCT): 2

FUNCTION: CHANGE-ELEMENT
IMAGE OR SAVE-AREA (DEFAULT IS IMAGE):
ELEMENT NUMBER: 1
FIRST DEVNO. (OCT): 4
LAST DEVNO. (OCT): 10
```

```
FUNCTION: LIST-TABLE
IMAGE OR SAVE-AREA (DEFAULT IS IMAGE):
OUTPUT FILE:

ELEMENT NO.      FIRST DEVNO      LAST DEVNO
         0                0               2
         1                4              10
         2                0               0
         3                0               0
         4                0               0
         5                0               0
         6                0               0
         7                0               0

FUNCTION: DELETE-ELEMENT
IMAGE OR SAVE-AREA (DEFAULT IS IMAGE):
ELEMENT NUMBER: 1

FUNCTION: CLEAR-TABLE
IMAGE OR SAVE-AREA (DEFAULT IS IMAGE):

FUNCTION: EXIT
```

10.4.9 *MONCALL-LOG

This command may be used to log how many times each monitor call is performed by one specific program or by all programs in the system. The log is kept in an internal buffer, and may be dumped to an output file when desired.

The moncall-log facility is an option that must be ordered at the time of system generation, to be implemented in the system.

Subcommands:

START-MONCALL-LOG
    Starts the log procedure for one or all programs
    Example:

```
    FUNCTION: START-MONCALL-LOG
    LOG MONCALLS FOR ONLY ONE PROGRAM (DEFAULT IS YES)? YES
    RT NAME: BAK08
```

STOP-MONCALL-LOG
    Stops the log procedure. The log may be restarted by

RESTART-MONCALL-LOG

PRINT-MONCALL-LOG <output file>
    Prints the current contents of the log. This command may be given both during the log (before STOP), and after.

HELP

EXIT

10.4.10 *SWAPPING-LOG

This command is used to keep a log of swapping and pagefaults.

The swapping-log facility is an option and must be ordered at time of system generation, to be implemented in the system.

Subcommands:

START-SWAPPING-LOG
    Starts swapping-log for one specific program, or for all programs.

    Example:

```
FUNCTION: START-SWAPPING-LOG
LOG SWAPPING FOR A SPECIFIC PROGRAM (DEFAULT IS YES)? YES
RT NAME: BAK17
```

STOP-SWAPPING-LOG
    Stops the current logging. The log may be restarted by:

RESTART-SWAPPING-LOG

READ-SWAPPING-LOG
    Example:

```
FUNCTION: READ-SWAPPING-LOG

TOTAL NUMBER OF PAGEFAULTS WITHOUT DISC ACCESS            189
TOTAL NUMBER OF PAGEFAULTS IN RT-COMMON                     0
TOTAL NUMBER OF PAGEFAULTS ON LEVEL 4                       1
TOTAL NUMBER OF PAGEFAULTS ON LEVEL 1                      46
TOTAL NUMBER OF PAGES SWAPPED OUT (WRITTEN TO DISC)         9
```

SWAPPING-LOG
    Prints swapping log information at specified intervals, default 60 seconds. This command must be terminated by an <escape>

    Example:

```
FUNCTION: SWAPPING-LOG
LOG SWAPPING FOR A SPECIFIC PROGRAM (DEFAULT IS YES)? NO
INTERVAL IN SECONDS (DEFAULT IS 60 SECS): 2

           PF WITHOUT   PF IN        PF ON       PF ON      PAGES
           DISC ACCESS  RT-COMMON    LEVEL 4     LEVEL 1    SWAPPED OUT

TOT.ACCUM:     197           0           0           4           3
IN INTV.:        5           0           0           0           0
```

    If either of the commands START-SWAPPING-LOG or SWAPPING-LOG has been used, they cannot be used again before the command STOP-SWAPPING-LOG is given.

EXIT

HELP

10.4.11 *CPU-LOG

The command CPU-LOG will print the CPU activity as a percentage at specified intervals (default 60 seconds). The percentage printed is based on the contents of a SINTRAN III global variable CPULOOPTIME. (location 342 in resident). The contents of this variable indicates the number of passes pr. second through the idle loop on level 0 when no other activity takes place in the computer. This variable has an initial value, but may be more accurately calibrated through the command:

10.4.12 *FIND-CPULOOPTIME

This command should only be issued when no activity takes place on the machine. After a delay of 30 seconds the number of "CPU loops" per second (decimal) will be printed. The variable CPULOOPTIME may be changed to this value by the @SINTRAN-SERVICE-PROGRAM command *CHANGE-VARIABLE, to get a more accurate percentage from CPU-LOG.

10.5 DISK ACCESS LOG

10.5.1 GENERAL

The disk access log may be used to log all or a selection of the disk accesses performed by the system. Communication with the disk access log system may be done through SINTRAN-SERVICE-PROGRAM commands or through MON ABSTR.

The disk access log facility is an option and must be ordered at the time of system generation, to be implemented in the system.

10.5.2 MAINTAINING THE DISK LOG

The SINTRAN-SERVICE-PROGRAM command *DISC-ACCESS-LOG has several subcommands:

```
DEFINE-DISC-ACCESS-LOG
START-DISC-ACCESS-LOG
STOP-DISC-ACCESS-LOG
START-DISC-ACCESS-COUNTER
STOP-DISC-ACCESS-COUNTER
CLEAR-DISC-ACCESS-COUNTER
DISC-ACCESS-COUNTER
DISC-DRIVER-ERROR-INFORMATION
DISC-ERROR-STATUS
LOG-DISC-ACCESS-COUNTER
EXIT
HELP
```

The function DEFINE-DISC-ACCESS-LOG is used to:

- Define a log file.
  The log file must be contiguous. The area occupied by the log file will be used to log disk accesses but will not be accessed through the file system, i.e the byte pointer of this file will remain zero.

- Select log record size.
  Small (4 words) or big (8 words). Layout of record is described below.

- Specify which disk accesses should be logged.
  All disk accesses may be logged, or only the accesses to a specified controller or drive. Logging of only read or write accesses may be specified. Accesses to a limited part of a disk may be specified.

Simple example, specifying log of all disk accesses:

```
FUNCTION: DEFINE-DISC-ACCESS-LOG
DISC ACCESS LOG FILE: LOGGING:LOG
SMALL OR BIG RECORD SIZE ON DISC LOG FILE (DEFAULT IS BIG)? SMALL
LOG ALL DISC ACCESSES (DEFAULT IS YES)? YES
```

Example using all the possible specifications:

```
FUNCTION: DEFINE-DISC-ACCESS-LOG
DISC ACCESS LOG FILE: LOGGING:LOG
SMALL OR BIG RECORD SIZE ON DISC LOG FILE (DEFAULT IS BIG)? BIG
LOG ALL DISC ACCESSES (DEFAULT IS YES)? NO
LOG DISC ACCESSES TO ONE CONTROLLER ONLY (DEFAULT IS NO)? YES
LOGICAL DEVICE NUMBER OF DISC TO LOG (OCTAL): 1100
LOG DISC ACCESSES TO ONE DRIVE ONLY (DEFAULT IS NO)? YES
DRIVE NUMBER OF DRIVE TO LOG: 0
LOG ONLY WRITE ACCESSES (DEFAULT IS NO)? YES
LOG ONLY READ ACCESSES (DEFAULT IS NO)? NO
LOG ONLY ACCESSES TO A LIMITED PART OF THE DISC (DEFAULT IS NO)? YES
FIRST DISC ADDRESS IN THE DISC PART TO LOG (OCTAL ONLY): 3400
LAST DISC ADDRESS IN THE DISC PART TO LOG (OCTAL ONLY): 3417
```

The function START-DISC-ACCESS-LOG will start the log procedure if the log has been defined.

The function STOP-DISC-ACCESS-LOG will stop the log procedure.

The function START-DISC-ACCESS-COUNTER starts and defines the disk access counter, which is a simpler feature that just counts the number of read and write accesses.

Example:

```
FUNCTION: START-DISC-ACCESS-COUNTER
COUNT ALL DISC ACCESSES (DEFAULT IS YES)? NO
COUNT DISC ACCESSES TO ONE CONTROLLER ONLY (DEFAULT IS NO)? YES
LOGICAL DEVICE NUMBER OF DISC TO LOG (OCTAL): 1100
COUNT DISC ACCESSES TO ONE DISC UNIT NUMBER ONLY (DEFAULT IS NO)? YES
DRIVE NUMBER OF DRIVE TO LOG: 0
```

The function DISC-ACCESS-COUNTER will give the present value of the disk access counters.

The function LOG-DISC-ACCESS-COUNTER will give values of the disk access counters at specified intervals. Default interval is 60 seconds. This log must be terminated by an <escape>.

The function CLEAR-DISC-ACCESS-COUNTER will reset the counters to 0.

The function DISC-DRIVER-ERROR-INFORMATION will list certain error variables from the disk driver. These are explained in the driver listing.

The function DISC-ERROR-STATUS will list error information from the disk datafield.

Layout of disk log file record:

SMALL: (4 words)

0	Contents of CTREG	-	Function
1	Contents of CAREG	-	disk address
2	Contents of CDREG	-	disk address
3	CLUNIT	-	logical unit no.

BIG:(8 words)

0	Contents of CTREG	-	Function
1	Contents of CAREG	-	disk address
2	Contents of CDREG	-	disk address
3	Contents of CXREG	-	No. of sectors to transfer
4	Contents of MEMA1	-	memory address
5	Contents of MEMA2	-	memory address
6	CLUNIT	-	logical unit no
7	Contents of RTREF	-	Current RT-program

10.5.3 HOW THE DISK ACCESS LOG WORKS

Information about the disk log is found in the datafield DFDIL.If the disk log is defined and started, CTRDISK will check the disk log datafield for each transfer , to find out whether this transfer should be logged or not.

When the disk log is started, a buffer page is allocated. This page is divided into two buffers. When one of the buffers is full, the RT-program RTDIL is activated. This RT-program will dump the contents of the full buffer to the log file. Subsequent disk transfers will be logged to the other buffer.

Layout of disk access log datafield - DFDIL:

DISP	SYMBOL	COMMENT
-2		
....		Standard mass storage locations
11		
12	DILBPNT	Buffer pointer for disk log
13	DILBANK	Memory bank for disk log buffer
14	2DIBADDR	16 lower phys. memory add. bits of disk log buffer
	DOUBLE DDIBADDR=DILBANK	
15	DILDADDR	Start address of disk log file (in disk addr)
	15 DIL1DADDR	
	16 DIL2DADDR	
17	DILNSEC	Number of sectors per page on "disk log file" disk
20	DILLADDR	Last legal disk address on disk log
	20 DIL1LDADD	
	21 DIL2LDADDR	
22	DILGFLAG	Disk log flags
23	DILFLOG	Logical device number of disk log file disk
24	DILFUNIT	Drive number for disk log file disk drive
	DOUBLE DDILFLOG=DILFLOG	
25	DLLOGDV	Logical unit number of disk to log
26	DLDRIVE	Drive number of disk to log
27	DLALOGDV	Disk controller to count disk accesses on
30	DLAUNIT	Disk unit number to count disk accesses on
31	DILFADDR	First disk address to log
	31 DIL1FADDR	
	32 DIL2FADDR	
33	DILGLADDR	Last disk address to log
	33 DI1LADDR	
	34 DI2LADDR	
35	DXNDACCESS	Disk access counter
	35 1XNDACCESS	
	36 2XNDACCESS	
37	DXNWDACCESS	Write disk access counter
	37 1XNWDACCESS	
	40 2XNWDACCESS	
41	DALFUNC	Function code in MON ABSTR for disk access log RT-program
42	DALCMADDR	Memory address in MON ABSTR
43	DALCDADDR	Disk address in MON ABSTR

BITS IN DILGFLAG :

0	DIL1CONTROLLER	Log for one controller only
1	DIL1UNIT	Log for one unit (drive) number only
2	DILLIMIT	Log a part of the disk only
3	DILRACCESS	Log read accesses only
4	DILWACCESS	Log write accesses only
5	DILSMALL	Write small record (4 words) on disk log file
6	DAC1CONTROLLER	Disk access counter for one controller only
7	DAC1UNIT	Disk access counter for one unit number only

10	DILSTART	Disk log started
11	DILDEFINED	Disk log file defined
12	1DILBFULL	Disk log file buffer #1 is full
13	2DILBFULL	Disk log file buffer #2 is full
14	DILCOUNT	Count disk accesses
15	DILBOK	Disk log buffer is fixed

10.5.4 MON ABSTR AGAINST DISK LOG

The contents of the disk log datafield may be changed by using MON ABSTR (MON 131), and thereby defining the log, start, stop, etc.

Example:

```
LDT  LDN
LDA (PARLI

PARLI, FUNC
       DESAD

FUNC,  3        % Function (define log in this example)
DESAD, DESA1    % Physical address of
       DESA2    % working area
....
LDN,2200        % Logical device number for MON ABSTR against disk log
```

The function determines which operation is to be performed. The working area will be used to give or receive information.

Working area layout for function 3:

0	ZFSTART	Start address of disk log file
1	xx	(sector address)
2	ZFNBLCK	Sectors per page on log file disk
3	ZFEND	Last legal address on disk log file
4	xx	
5	ZFLOGU	Logical device number of log file disk
6	ZFUNIT	Drive number of log file disk
7	ZDILGFLAG	Flag (see below)
10	ZLLOGU	Logical device number of disk to be logged
11	ZLUNIT	Drive number of disk to be logged
12	ZFLGADDR	First sector address of specified area
13	xx	
14	ZLLGADDR	Last sector address of specified area
15	xx	

Possible functions:

Function = 1:
Write disk log record. Contents of working area is written as a record to the disk log buffer. The 4 first words are written if the current record size is small, otherwise the 8 first words will be written.

Function = 2:
As for function 1, the rest of the disk log buffer is filled with zeros and the buffer is written to the log file.

Function = 3:
Define disk log record.
Contents of working area is written to the datafield DFDIL.

Function = 4:
Start disk log. Working area is not used.

Function = 5:
Stop disk log. Working area is not used.

Function = 6:
Start disk access counter.
First word in working area contains flag word. See DILGFLAG bits.

Function = 7
Stop disk access counter.

Function= 10B
Clear disk access counter.

Function =11B
Read disk access counter.
The 4 first words of the working area filled with:

```
      0,1 :  number of read accesses
      2,3 :  number of write accesses
```

Function = 12B
Read last disk error information
Variables from the disk driver transferred to the working area

Function = 13B
Variables from the disk datafield transferred to the working area.

MON ABSTR activates the "driver" DVDIL on level 11 which will perform the necessary updating of the disk access log datafield DFDIL, according to the specified function.

10.6 COMMANDS FOR THE LAMU SYSTEM

See also description of MON MLAMU.

10.6.1 *PAGES-TO-LAMU

Parameters:	<Number of pages>.
	<first physical page>.
	<memory>
	<image>

Create a LAMU area.

10.6.2 *PAGES-FROM-LAMU

Parameters:	<First physical page>.
	<memory>
	<image>

10.6.3 *LAMU-AREAS

Parameter:	<output file>

List the memory parts reserved for the LAMU system.

10.6.4 *CREATE-LAMU

Parameters:	<LAMU ID>
	<SIZE>
	<PHYS.ADDR.>

10.6.5 *PROTECT-LAMU

Parameters:	<LAMU ID>
	<RING>
	<PROTECTION BITS (RFW)>

10.6.6 *DELETE-LAMU

Parameter:	<LAMU ID>

10.6.7 *LAMU-INFORMATION

Parameters:	<LAMU ID>
	<OUTPUT FILE>

List all relevant information on a LAMU or on all LAMUs.

10.6.8 *SET-LAMU-CONSTANTS

Parameters:	<NO. OF LAMUS PER RT PROGRAM>
	<TOTAL NUMBER OF LAMUS>

The system needs to be restarted to make any changes effective.

10.6.9 *LIST-LAMU-CONSTANTS

11. RT-LOADER

11.1 CORRECTED ERRORS

CHANGE-RT-DESCRIPTION

This command did not work properly with declared programs. When this command is used, a declared program will now receive the correct parameters, will change from declared to defined, and will be entered into the RTFIL.

11.2 DICTIONARY TABLE

The RT-Loader is now able to make use of a dictionary table when loading library files. A dictionary table can be made with the BRF-editor command MAKE-LIBRARY-FILE or with the BRF-Linker command PREPARE-BRF-LIBRARY-FILE. It contains information about the :BRF units on the file.

The loading of such files is faster, in particular if only a few units are referenced. Another advantage is, that units on the file may refer backwards to earlier units on the file.

Older versions of the RT-Loader will just skip the dictionary table and load the file in the usual way.

11.3 NEW COMMANDS

11.3.1 READ-PROGFILE

As a first step towards a two-bank RT-Loader, a new RT-Loader command has been implemented:

READ-PROGFILE <input file>,<segment>,<initial page table>

The command reads a :PROG file onto a segment and terminates the load of that segment (END-LOAD is not necessary).

- <Input file> is of default type :PROG and may be one-bank or two-bank.

- <Segment> Must be the segment number or name given in a previous NEW-SEGMENT or NEW-BACKGROUND-SEGMENT command.
Default value is that segment.
Only one NEW-SEGMENT or NEW-BACKGROUND-SEGMENT command is legal before this command.

- <Initial page table> Initial page table for program bank.
Default value is the one given by a previous SET-PAGE-TABLE command.
Legal values are 1 or 2.

If a two-bank program file is loaded, the databank will be placed on the "opposite" bank. (2 or 1)
The address space of the segment will be determined by the program file.

11.3.2 SET-FORTRAN-100-DEFAULT

This command is implemented to simplify the use load procedures developed for FTN-compiled programs with FORTRAN-100 compiled programs.

Originally, the commands REENTRANT-LOAD and NREENTRANT-LOAD were tailor-made to systems compiled with the FTN compiler. These commands may now be used to load programs compiled with the FORTRAN-100 compiler, if SET-FORTRAN-100-DEFAULT is given first.

After each REENTRANT-LOAD the library FORTRAN-1BANK will be loaded. Stack space will be allocated and non-reentrant symbols deleted.

If the command NREENTRANT LOAD is used, the library FORTRAN-1BANK will be loaded at END-LOAD.

The RT-Loader will remain in "FORTRAN-100-mode" until the command RESET-FORTRAN-100-DEFAULT is given, or until the RT-Loader is left and entered again.

12. BACKGROUND ALLOCATION SYSTEM

12.1 GENERAL

The background allocation system makes it possible to generate SINTRAN III with a larger number of Terminals/TADs than the number of background processes. When a Terminal/TAD requests a background process (escape), one will be allocated if there is a free one.

Since each background process uses two segments, system segment and user segment, this makes it possible to increase the number of terminals without increasing the number of segments.

A timeout function is provided. If no activity has taken place for a set period, the terminal will be logged out. Warnings about this will be given before logout.

The background allocation system is an option, it is generated by including the library mark 8BACS.

12.2 SINTRAN-SERVICE-COMMANDS

The command *BACKGROUND-ALLOCATION-UTILITIES has these subcommands:

SET-PERMANENT-CONNECTION <termno> <memory> <image> <save>
A terminal will be permanently connected to a background process, if one is free. The connection is reset by:

RESET-PERMANENT-CONNECTION

DISABLE-TIMEOUT <termno> <memory> <image> <save>
The timeout function will not affect this terminal. The timeout function is restored by:

ENABLE-TIMEOUT

CHANGE-LOGOUT-TIME <time> <memory> <image> <save>
Changes the amount of time a terminal may be inactive before it is logged out (if the timeout function is on). Original value = 30 minutes.

CHANGE-WARNING-TIME <time> <memory> <image> <save>
Changes the time a terminal may be inactive before the first logout warning is printed . Original value = 25 minutes.

TIMEOUT-OFF
Disables the timeout function for the entire system.

TIMEOUT-ON
Enables the timeout function for the entire system.

DISPLAY
    Gives an overview of all the terminals and background processes.

LIST-PARAMETERS
    Lists the current timeout parameters.

FREE-BACKGROUND-PROGRAMS
    Lists those background programs presently not connected to a terminal/TAD.

13. SINTRAN III/VSX-500 VERSION J AND ND-500 MONITOR VERSION F

13.1 OUTLINE

The following ND-500 topics are covered:

- General news
- Modified commands
- New commands
- Modified monitor calls
- New monitor calls

Some of the monitor calls described here, may also be further described elsewhere in this document.

13.2 GENERAL NEWS

13.2.1 MESSAGE BUFFERS

The buffers for messages to the microprogram will now be dynamically allocated (when first user enters monitor after system start). They can now be in any of the ND-100 memory banks 0-3. This makes it possible to generate SINTRAN III with many more ND-500 processes.

13.2.2 NEW PACKAGING OF THE ND-500 MONITOR

The ND-500 MONITOR SUBSYSTEM PART is now delivered as a :PROG-file. For the user this means that more macros and debugging info (break points etc.) can now be used. For the system supervisor it implies a slight change in installation procedures. The ND-500 MONITOR now comes on 3 diskettes. When installing the subsystem part (ND-500-MONITOR-F:PROG) the command DUMP-PROGRAM-REENTRANT should be used (on previous versions DUMP-REENTRANT was used). It is now also possible to define an extra command to be used for starting the symbolic debugger without first entering the ND-500 MONITOR.

13.2.3 DISASSEMBLY IN LOOK-AT

Longer instructions can now be disassembled. Up to four lines of output can be displayed.

13.2.4 EXTENDED ADDRESS RANGE IN LOOK-AT-CONTROL-STORE

This is only significant for machines with additional control store.

13.2.5 NEW TIME QUEUE FOR THE ND-500

A new time queue for ND-500 processes is introduced in this version. The time queue is different from the standard time queue, in that the monitor calls STARTPR, SWITCHP, SIBSURV, A5XMSG and B5XMSG work on this queue. Those monitor calls that operate upon the standard SINTRAN III time queue (RT, INTV etc.) will have no effect on processes in the ND-500 time queue. A process enters the ND-500 time queue when the new 5TMOUT monitor call is used.

13.2.6 TIME SLICING

A new time slice class (4) has been implemented for mode-jobs using ND-500.

TIME SLICE ELEMENT NO.	PRIORITY OF ELEMENT	NO. OF TIME SLICE UNITS ON ELEMENT	
21	50	3	(ESCAPE ELEMENT)
21	50	3	(BREAK ELEMENT)
22	42	6	
23	35	14	
4	30	30	
5	24	50	
6	20	4	
5			

Processes running on the low priorities will now get their priority incremented each time they do a monitor call. This results in a more fair distribution of CPU-time between I/O-bound and CPU-bound processes.

13.2.7 MESSAGES APPEARING WHEN USING AUTOMATIC-ERROR-MESSAGE

These messages will now contain the monitor call number instead of the monitor call name.

13.3 MODIFIED COMMANDS

13.3.1 PROCESS-LOG-ALL

Computation of the OTHERS column has been revised.

13.3.2 BREAK AND TEMPORARY-BREAK

These commands have now been extended by an extra parameter. This extra parameter is executed as a command when the break-point is reached. This option, combined with the built-in macro facility of the ND-500 MONITOR, can be very useful during debugging.

13.3.3 DEBUG-STATUS

This command will show break-point interval and command, as specified in the BREAK and TEMPORARY-BREAK commands.

13.3.4 SET-SYSTEM-PARAMETERS and LIST-SYSTEM-PARAMETERS

The format and output of these commands have been changed. Some parameters no longer used by the time slicer have been removed.

13.3.5 SET-PRIORITY

The format of this command has changed.

13.3.6 LIST-EXECUTION-QUEUE

If no interval is given, the queue will be listed once only.

13.3.7 DEFINE-STANDARD-DOMAIN

More types of domains than before can now be defined as standard domains. It is still not possible to define, as standard domains, the domains with RT-COMMON access, common segment with ND-100 or shared segment.

13.3.8 CHANGING REGISTER CONTENTS WHEN IN LOOK-AT

In certain cases the modification of a register could also result in the modification of data or instructions. This has been corrected on the F-version.

13.4 NEW COMMANDS

13.4.1 EXIT FROM LOOK-AT

A period (.) may now be used as a synonym for the EXIT command in LOOK-AT mode.

13.4.2 INSERT-IN-TIME-SLICE and REMOVE-FROM-TIME-SLICE

Two new commands to enable and disable time slicing of a process.

13.4.3 LIST-TIME-QUEUE

A new command to list the ND-500 time queue. A process enters the ND-500 time queue when the new 5TMOUT monitor call is used.

13.5 MODIFIED MONITOR CALLS

13.5.1 FOPEN

The last (5th) parameter previously returned the SINTRAN III open file number. It will now return the ND-500 connect number.

13.5.2 DVINST

The possibility of using device 0 (edited input) has been added.

13.5.3 DVOUTS

This monitor call may again be used for output to terminal 1 and error device.

13.5.4 NOWT

Processes terminating with terminal in nowait state would either,
1) Log the user out (if the process was started from SINTRAN III)
   or
2) Write END OF FILE endlessly.

This has been corrected.

13.5.5 PASET and PAGET

These monitor calls now have the same meaning as PASET and PAGET on ND-100. The old functions of PASET and PAGET on ND-500 are now supplied by the monitor calls 5PAGET and 5PASET.

13.5.6 TMOUT

Parameters:
   <number of time units>
   <time unit>
   <return status>

The format has been changed. It is now the same as for HOLD and 5TMOUT.

13.5.7 WFILE/RFILE/MAGTP USING DIRECT TRANSFER

The previous restriction that bytecount/byteaddress must be modulo block size has been removed.

13.6 NEW MONITOR CALLS

13.6.1 201 - MHDLC

```
Parameters:
   <function>
   <ldn>
   <DCB-adr>
   <DCB-usize>
   <DCB-msize/wflag>;
```

The DCB has the same format in ND-500 as in ND-100, i.e. the COMMAND, STATUS, and HARDWARE STATUS fields of the DCB still occupy 2 bytes. If programs written in high level languages should run on both ND-100 and ND-500, you should take that into account, for example declaring the DCB with INTEGER*2 in FORTRAN and INTEGER2 in PLANC. An INTEGER in PLANC and FORTRAN will occupy 2 bytes in ND-100 and 4 bytes in ND-500. But INTEGER*2 and INTEGER2 will always occupy 2 bytes regardless of the machine.

Normally, the SUPER-DEVICE-CLEAR should be used during program initialization. This command will delete all DCBs currently being processed by the driver. "Garbage DCB reading" should then be unnecessary. The SUPER DEVICE CLEAR command looks like the DEVCL command, but the command is 0 (not 3).

Maximum size of DCB-MSIZE and DCB-USIZE is 176000_8.

DCB-ADDR must be even.

Function code 2 for read DCB and function code 0 for send DCB are legal from ND-500. Max length of user buffer must be specified in parameter 4 for function code 2.

13.6.2 256 - DEABF

```
Parameters:
   <abbr. file name>
   <full file name>
   <default file type>
```

This monitor call now has an extra (optional) parameter: default file type.

13.6.3 325 - OCTO

Parameters:
<function>
<Logical device no.>
<Dest. stn./Ret.value>

Legal function codes are:

- 0 - Kick
- 1 - Wait for kick
- 5 - Read OCTOBUS status
- 6 - Who am I.

<Dest. stn./Ret.value> is destination station in function 0 (kick), return value in functions 5 (read status) and 6 (who am I) and dummy in function 1 (wait). If function is 5, then the value returned will be the last used transmit status in the lower 16 bits, and actual hardware transmit status in the upper 16 bits.

13.6.4 327 - FSMUL

Parameters:
<function>
<fileno>

Multifunction file system monitor call. Only function code 1 is legal. This code means write back index buffers. <fileno> is the connect number of the file which indexes should be written back.

13.6.5 333 - UDMA

Parameters:
<func>
<buffer>
<unit>
<ipar1>
<ipar2>

Monitor call for Universal DMA. There is a fast version of this monitor call as an option in SINTRAN.

13.6.6 435 - PRT

Parameters:
<process number>
<reason code>

Cause programmed trap in another process. <process number> composed of cyclic number and process index (as in for instance STARTPR) identifies the process to be trapped. This process will get the lower half of <reason code> (in PRT) as the <error code> (in GERRCOD) when using GERRCOD inside the trap handler. The upper half of <error code> (in GERRCOD) will contain the process number of the trapping process.

13.6.7 330 - TRMST

Parameters:
<ldn>
<buffer>

Terminal status.

13.6.8 332 - TLREP

Parameters:
<function>
<ldn>

Terminal line report.

13.6.9 514 - TMOUT

Parameters:
<number of time units>
<time unit>
<return status>

Put program in ND-500 time queue.

13.6.10 436 - 5PASET

Parameter:
<buffer>

Same as PASET on previous versions.

13.6.11 437 - 5PAGET

Parameter:
<buffer>

Same as PAGET on previous versions.

14. IS-XMSG

Only version H or later of IS-XMSG can run under SINTRAN III version J.

15. FLOPPY DRIVER, BOTH NEW AND OLD

It is now possible to order SINTRAN III with both the old and the new floppy driver. The floppy status register is examined and the appropriate driver is selected when the system starts up.

15.1 HOW TO GENERATE

Select new (big) floppy in SINGEN, and add 8FDI1 (for controller 1) to free library marks (8FDI2 for controller 2).

15.2 REQUIRED SPACE

In resident, this facility require space for an extra datafield (172 locations per controller). In POF, an extra driver is required (715 locations).

16. DEFAULT FILE ACCESS

The default file accesses are changed: When a new user is created, the default file access is set to:

Public access: NONE
Friend access: RWACD
Own access: RWACD

Files with public access equal to NONE, may be appended to a batch process, and to a spooling queue. Scratch files may have public access equal to NONE.

When a friend is created, the default access is R.

17. FILE ACCESS ON SCRATCH FILES

The scratch files may now have no public nor friend access.

18. SPOOLING SYSTEM

The spooling system has been reorganized in order to save space in resident memory and to reduce the number of segments used by each spooling process. Thus, the maximum allowable number of spooling processes in a system is increased by 15 to 30 (plus COSMOS spooling) processes.

18.1 Spooling Datafield

The memory resident spooling datafield is reduced to 10 words:

```
SPPRx,  JPL I *+1          % START ADDRESS TO SPOOLING PROCESS
        SPORT              % LABEL ON SPOOLING PROGRAM SEGMENT
        SPROG              % SPOOLING PROG RT-DESC. ADDRESS
        SPERI              % PERIPHERAL DEVICE NUMBER
        SQUEU              % SPOOLING QUEUE SEGMENT NUMBER
        SQSEM              % SPOOLING QUEUE SEMAPHORE NO
        SQIOS              % SPOOLING QUEUE I/O SEMAPHORE NO
        SSTOP              % STOP COMMAND FLAG
        SABOR              % ABORT COMMAND FLAG
        SPINX              % SPOOLING INDEX (1 - 30)
```

Spooling datafield on Spooling Program Segment:

```
DISP -40
INTEGER ARRAY ACCBUFF(40)     % GENERAL PURPOSE BUFFER USED BY
PSID                          % SPOOLING PROGRAM
DISP 1
INTEGER SPROG             % ADDRESS OF RT-DESCRIPTION
INTEGER SPERI             % PERIPHERAL DEVICE NUMBER
INTEGER SQUEU             % SPOOLING QUEUE NUMBER
INTEGER SQSEM             % SPOOLING QUEUE SEMAPHORE
INTEGER SQIOS             % SPOOLING QUEUE I/O SEMAPHORE
INTEGER SSTOP             % STOP COMMAND FLAG
INTEGER SABOR             % ABORT COMMAND FLAG
                                        % 1 - ABORT CURRENT PRINT
                                        % 2 - RESTART CURRENT PRINT
                                        % 3 - STOP PRINT
   SYMBOL BBASP=10                      %     BACKSPACE PRINT
   SYMBOL BFSSP=11                      %     FORWARD SPACE PRINT
   SYMBOL BNFIQ=12                      %     NEW FILE IS SET FIRST IN QUEUE
INTEGER SPINX   % SPOOLING INDEX (1 - 30) (end of resident part)
INTEGER POINTER HEADER,TRAILER          % DEVICE DEPENDENT ROUTINES
INTEGER POINTER PRINTBUFFER             %         "
INTEGER FILENUMBER                      % FILENUMBER
INTEGER LASTPAGE,PAGENUMBER             % LAST PAGENO IF ANY AND CURRENT PAGE
INTEGER REMAINING,BYTENUMBER            % BYTENO ON LAST PAGE AND CURRENT BYTEN
INTEGER BUFHAD                          % ADDRESS OF DEVICE BUFFER RESERVED FOR
INTEGER PCBNK                           % DEVICE BUFFER ADDRESS, BANK NO
INTEGER PCADR                           % DEVICE BUFFER ADDRESS WITHIN BANK
DOUBLE PCORAD=PCBNK                     % PHYSICAL ADDRESS OF DEVICE BUFFER
INTEGER SNPAGES                         % NUMBER OF PAGES TO BACKSPACE/ADVANCE
INTEGER SNLINES                         % NUMBER OF LINES TO BACKSPACE/ADVANCE
INTEGER SCONDITIONS          %  = 0 - DO NOT PRINT FILE NAME
                             %  = 1 - PRINT FILE NAME BUT DO NOT STOP.
                             %  = 2 - PRINT FILE NAME AND WAIT FOR START-PRINT.
INTEGER NLPAGE               % NUMBER OF LINES PER PAGE
INTEGER SPMODE               % SPOOLING MODE; 0: PRINT; 1: PLOT
INTEGER SPAGENO              % SAVED PAGENUMBER
INTEGER WSNLINE              % SAVED SNLINE
INTEGER NOSINQ=WSNLINE       % NO OF ELEMENTS IN SPOOLING QUEUE
INTEGER WSNPAGE              % SAVED SNPAGE
INTEGER NUMINQ=WSNPAGE       % NO OF ELEMENTS IN SPOOLING QUEUE
INTEGER LFCOUNT              % LINE-FEED COUNTER
INTEGER ARRAY SPFNAM(16)     % PERIPHERAL FILE NAME FOR SPOOLING DEVICE
INTEGER STORX                % GENERAL SAVE LOCATION
INTEGER BUFFAD               % ADDRESS OF GENERAL DATA BUFFER
INTEGER FPAR1LIST,FPAR2LIST  % GENERAL PARAMETER LIST
INTEGER FPAR3LIST            %
INTEGER NBYTS                % NUMBER OF BYTES TO PRINT (FOR OUTSTRING)
INTEGER POINTER LNKSP        % RETURN ADDRESS
INTEGER SAVFN                % SAVED FILE NUMBER
INTEGER STORT                % GENERAL SAVE LOCATION
INTEGER DBHCACHE             % ADDRESS OF DEVICE BUFFER HEADER IN DISC CACHE
INTEGER SOUR1,SOUR2          % ADDRESS OF DEVICE BUFFER IN CACHE
DOUBLE  SOURX=SOUR1          % ADDRESS OF DEVICE BUFFER IN CACHE
INTEGER RFIEL                % ADDRESS OF SPOOLING DATAFIELD IN RESIDENT
INTEGER ARRAY SBUFR(0)       % QUEUE ELEMENT BUFFER
INTEGER NOCOPYS                        % FIRST WORD=NO OF COPIES
INTEGER FSPMESS                        % FLAG FOR SPOOLING-FILE-MESSAGE (RIGHT BYTE)
                                       % AND BACKGROUND/RT (LEFT BYTE)
INTEGER ARRAY FNAME(57)                % POSITION OF SPOOLING FILENAME
INTEGER NAME1=FNAME                    %
INTEGER ARRAY SPJNAME(0)               % PROJECT NAME BUFFER
INTEGER SPJN1                             % 0 IF NO ACC., RT-DESCR IF RT-ACC
INTEGER SPJN2                             % ACCTAB INDEX IF RT-ACC.
INTEGER ARRAY SPJN3(6)
INTEGER ARRAY SPMESSBU(117)            % USER MESSAGE BUFFER
                             % END OF QUEUE ELEMENT BUFFER
PSID

SYMBOL SPLEN=12              % LENGTH OF TABLE ENTRY IN RESIDENT
SYMBOL SSPLEN=340            % LENGTH OF TABLE ENTRY ON SEGMENT
```

18.2 The Spooling Program Segment

The spooling program is removed from segment 25 (File System Reentrant segment no. 2) to its own segment, named Spooling Program Segment. This segment has its save area on segment 42 (Initial Spooling Program Segment).

Layout of Spooling Program Segment:

```
110000  ┬
        │   31 spooling datafields
        │      (30 spooling processes + COSMOS spooling)
        │
125443  ┼
        │   31 Form datafields
        │
132677  ┼   SPORT: Start of spooling program
        │
        │
137632  ┴           End of spooling program
```

18.3 The Spooling Queue Segments

Each spooling process has one spooling queue segment containing the spooling queue together with routines operating on the queue.

Layout of the spooling queue segment:

```
140000  ┬
        │   Subroutines operating on the queue
        │          (copied from segment no 25)
        │
140345  ┼   Number of elements in the queue
        │
140353  ┼   Maximum number of elements in the queue
        │
140356  ┼   Start of queue elements
        │
        │
147777  ┴   End of segment
```

The first time the system is started after loading (or in a "cold start"), the spooling queue segments are initialized. The length of the queue segment is used to compute the maximum number of elements in the queue. The standard length of the queue segment is 4K, which gives space for 28 queue elements. The segment length can be increased to 14K, giving a maximum queue length of 103 elements.

18.4 New Functions

If the machine stops while the spooling program is printing, the printing of the file which was interrupted, is restarted.

The command DEFINE-SPOOLING-FILE-MESSAGE is reinstalled.

Files with no public access may now be appended to a spooling queue.

Systems that put people first

NORSK DATA A.S OLAF HELSETS VEI 5 P.O. BOX 25 BOGERUD 0621 OSLO 6 NORWAY
TEL.: 02 - 29 54 00 - TELEX: 18284 NDN