SINTRAN III

Real Time Loader

A/S NORSK DATA-ELEKTRONIKK

SINTRAN III

Real Time Loader

REVISION RECORD

Revision	Notes
5/74	Original Printing
11/74	Version two, superceding the original printing
2/76	Version three, superceding all previous versions

Publ. No.
ND-60. 051.03
February 1976

A/S NORSK DATA-ELEKTRONIKK
Lørenvn. 57, Oslo 5, Tlf. 21 73 71

TABLE OF CONTENTS

+ + +

HOW TO USE THIS MANUAL

--ooOoo--

It is assumed that the reader has a good knowledge of the SINTRAN III system. The necessary background information is given in the manual SINTRAN III U sers Guide, especially chapters 1, 4, 5 and 7.

--ooOoo--

1 GENERAL REMARKS AND DEFINITIONS

The Real Time Loader (hereafter called the RT Loader) is a subsystem included in all versions of SINTRAN III mass storage. The RT Loader's main function is to load real time programs (hereafter called RT programs) in an active SINTRAN III system. An RT program is a program which has its own RT description and which has been loaded into the SINTRAN III system by the RT Loader. Some RT programs (the timer program, the error message program, the file system programs and one program for each batch process and each timesharing terminal, and a few others) are included in the SINTRAN III system when the system is generated.

A SEGMENT is a continuous limited area on a mass storage device containing executable code or data for RT programs or for the SINTRAN III system itself. When an RT program is started, its segment or segments or part of them are copied from their place on mass storage into memory, and when a segment or part of segment has to be removed, it will be transferred to its original location on the mass storage device. The logical pages of which a segment consists will in general be scattered about in main memory because of hardware paging. An RT program's virtual address space may be divided into two parts (segments) each consisting of a number of 1K word pages, and an RT program cannot use more than two segments simultaneously.

A segment is specified by its segment number, of which a limited number are available in a SINTRAN III system.

All segments in a SINTRAN III system are kept on continuous files, SEGMENT FILES, on a mass storage device. There may be from one to four SEGMENT FILES in a SINTRAN III system. The SEGMENT FILES are numbered from zero to three. A SEGMENT FILE may be defined in any file directory in a SINTRAN III system.

1.1 The RT Loader's Tables

There are two tables in the RT Loader with which users of the RT Loader should be familiar. These are the Linking table and the RTFIL table.

The linking table is a linked table containing all symbols available for the current load operation. Available symbols are symbols defined or referred to in the current load operation, symbols defined in the segment currently used as linking segment, all RT program names, all core common labels and all symbols defined in resident memory, segment 0. When a load operation is terminated by an END-LOAD command, all defined symbols in the linking table which do not exist in the RTFIL table are transferred to the RTFIL table, and only RT program names, core common labels and symbols defined in segment 0 (resident memory) will remain in the linking table (these symbols will also be present in the RTFIL table).

The RTFIL table contains all defined symbols (RT program names, names of entry points, etc.) in all the existing segments (including resident memory) built by the RT Loader. The RTFIL table is copied to a mass storage file named (SYSTEM) RTFIL: DATA after each load operation which changes the content of the RTFIL table. The SINTRAN III operator communication uses this file to find symbolic RT program names.

1.2 How to Start the RT Loader

Only the user SYSTEM and the user RT may use the RT Loader. The user RT must be defined as a friend of the user SYSTEM to be allowed to update (write on) the segment files and the RTFIL file. Only one user at a time may use the RT Loader. The RT Loader is started by the command @RT-LOADER.

If the RT Loader is free to use, it will print a version number and go into command input mode, otherwise the error message ALREADY IN USE will be given. The break characters "Escape" and "Break" will stop the execution of the RT Loader and give control to the operator communication, except in sequences where the RT Loader is updating the RTFIL table, the segment table or the RT description table. The command @CONTINUE cannot be used to restart the RT Loader.

2 COMMANDS

The RT Loader is ready to accept a command when an asterisk (*) is printed on the terminal. All RT Loader commands may be abbreviated in the same way as the SINTRAN III and file system commands. Missing parameters will be asked for by the RT Loader. Parameter delimiters are space, comma or carriage return. Parameter default values may be specified by giving two commas or carriage return. The character "control L" (octal 14) given in a command (or parameter) line, will terminate and cancel the command, and the RT Loader will be ready to accept a new command. The line editing characters "control Q" for deleting the current line and "control A" for deleting one character on the current line, are available in the RT Loader. Parameter types used by the RT Loader are:

- Octal numbers, the six last digits will count.
- File names.
- Octal logical device (file) numbers.
- Symbolic names, up to seven characters.

Decimal numbers cannot be used as parameter values, and all numbers written by the RT Loader are octal numbers.

All questions that the RT Loader may ask must be answered with Y for yes or N for no; other alphabetical characters will result in the question being repeated. All non-alphabetical characters will give an error message.

In the following sections a parameter surrounded by parantheses has default value, whereas parameters not surrounded by parantheses do not.

Example:

```
        DEFINE-SYMBOL <symbol>(<segment no.>)
```

The parameter <symbol> has no default value, but the parameter (<segment no.>) has a default value.

Note:

In the examples given in this manual user input is underlined to distinguish it clearly from the computer output. (On the terminal no underlining occurs.)

2.3 Binary Dump of a Segment on a File

*BINARY-DUMP <output file><segment no.> (<lower addr>)
      (<upper addr>)

This command will dump the segment <segment no.> in binary format on the file <output file>. The parameter <segment no> must refer to closed segment, i.e. a segment on the segment file, or it can have the value zero meaning core common. <output file> may be a file number or a file name. Default file type for <output file> is SYMB. The parameters (<lower addr>) and (<upper addr>) are respectively the lower and upper addresses of the area to be dumped. Default value for (<lower addr>) is the first address of the segment and default value for (<upper addr>) is the last address of the segment. There will be no bootstrap in front of the binary dump. The output from the BINARY-DUMP command may be read by the various MAC assemblers or by the RT Loader commands READ-BINARY and COMPARE.

Example of dumping segment no. 33 on the output file BIN-DUMP:SYMB:

```
*NREENTRANT-LOAD
INPUT FILE: 200-USER
LINKING-SEGMENT NO.:
NEW SEGMENT NO:  33
*END-LOAD
*WRITE-SEGMENT
SEGMENT NO: 33
OUTPUT FILE:

  33      0   13777   1300   0   0   1 RFW NON DEMAND
*BINARY-DUMP
OUTPUT FILE: BIN-DUMP
SEGMENT NO: 33
LOWER ADDRESS:
UPPER ADDRESS:
*
```

2.4 Change Content of One of the New Segments

*CHANGE-LOCATION <segment no.>

The CHANGE-LOCATION command is used to look at or to change locations on the segment <segment no>. The <segment no.> must be one of the segments currently being built.

The syntax is an address followed by a slash (/). The content of the location addressed will be printed out. If a new value is wanted in the location, the new value followed by a carriage return is typed. The content of the next location is then printed. If no change is wanted, just type carriage return and the content of the next location will be printed. This command must be terminated with the point (.) character.

The syntax of the CHANGE-LOCATION command is the same as the LOOK-AT command in SINTRAN III.

Example:

```
*NREENTRANT-LOAD 200-USER,,
NEW SEGMENT NO:  32
*WRITE-LOAD-ADDRESS 32

L.ADR:       0  U.ADR:  13776  C.LADR:  13776
*CHANGE-LOCATION
SEGMENT NO: 32
0/ 125005
 125005
 125005 10/   6614 1
   6427 2
 177777 3
 177777 10/      1 1
      2
      3 .
*
```

2.5 Change RT Description Table Element

*CHANGE-RT-DESCRIPTION <rt prog>(<prior>)(<segno 1>)
(<segno 2>)(<stadr>)

This command changes an already created RT description. Thus the parameter <rt prog> must be the name of a defined or declared RT program, and the RT program may not be active when this command is executed. The parameter (<prior>) is the new priority of the RT program, (<segno 1>) is the first segment (right byte in the SEGM word in the RT description table) and (<segno 2>) is the second segment (left byte) of the RT program. The parameter (<stadr>) is the start address of the RT program. The default values of the parameters are their old (current) values.

Example:

```
*WRITE-PROGRAMS,,

   CDC4  24517   35   0
   CDC3  24473   35   0
   CDC2  24447   35   0
   CDC1  24423   35   0
   CDC0  24377   35   0

*CHANGE-RT-DESCRIPTION
RT-PROGRAM: CDC0
PRIORITY: 100
SEGMENT ONE: 35
SEGMENT TWO:
START ADDRESS: 200
*END-LOAD
*
```

2.6 Clear an Existing Segment

*CLEAR-SEGMENT <segment no>

The segment <segment no> will be cleared, i.e. the space on the segment file occupied by the segment <segment no> will be released, and the segment number <segment no> will be free again. The segment cannot be one of the segments initially present in the SINTRAN III system. The segment will not be cleared if it is one of the segments of an existing RT program, or if the segment is currently being used by an RT program, or if it has been fixed using the FIX or FIXC command.

If the parameter <segment no> is given the value zero, which is equivalent to core common, and the question "CLEARING CORE COMMON?" will be printed. If the answer Y for yes is given, the core common pointers will be reset to their initial values, and all core common labels will be deleted from the linking table and the RTFIL.

When clearing a segment, all symbols defined on this segment will be deleted from RTFIL and the linking table.

Example:

```
*CLEAR-SEGMENT
SEGMENT NO: 35

ERROR - RT-PROGRAMS ON SEGMENT:

   CDC4
   CDC3
   CDC2
   CDC1
   CDC0
*DELETE-PROGRAM CDC4
*DELETE-PROGRAM CDC3
*DELETE-PROGRAM CDC2
*DELETE-PROGRAM CDC1
*DELETE-PROGRAM CDC0
*CLEAR-SEGMENT
SEGMENT NO: 35
*
```

2.7 Compare a Segment with a File

*COMPARE <segment no><file> (<output file>)(<lower addr>)
(<upper addr>)

The content of the segment <segment no> will be compared with the content of the file <file>. The segment <segment no> must be a closed segment and the content of the file <file> must be in binary format (produced by a)BPUN or a BINARY-DUMP command). The parameters (<lower addr>) and (<upper addr>) set the limits of the area to be compared. If there are any differences between the content of the <file> and the segment <segment no>, the addresses where the differences are, and the contens of those addresses for the segment and the file will be printed out on the (<output file>).

Default values for the parameters (<lower addr>) and (<upper addr>) are the first and the last address of the specified segment. The default value of the parameter (<output file>) is the communication device (the terminal). This command is useful for debugging RT programs. After loading the segments, they may be dumped with the BINARY-DUMP command. If anything goes wrong during execution of the RT programs using these segments, the COMPARE command may be used to see if anything in the original segments has been destroyed.

Example:

```
*NREENTRANT-LOAD 200-USER,,
NEW SEGMENT NO:   36
*END-LOAD
*BINARY-DUMP BIN-DUMP:SYMB 36,,,
*EXIT-LOADER

@LOOK-AT SEGMENT 36
READY:
0/ 125005 1
 125005 1000/        0 2
      0 5000/ 124010 3
 170401 10000/   50771 4
 142006 .
-END
@RT-L


REAL-TIME LOADER 76.02.06
```

```
*COMPARE
SEGMENT NO: 36
BINARY FILE: BIN-DUMP:SYMB
LOWER ADDRESS:
UPPER ADDRESS:
OUTPUT FILE:

  ADR      SEGMENT    FILE

     0           1   125005
  1000           2        0
  5000           3   124010
 10000           4    50771
*
```

2.8 Allocation of an RT Description

*DECLARE-PROGRAM <rt-program name>

The symbol <rt-program name> is the name of an RT program to be loaded at a later time, and an entry in the RT description table will be allocated. This command must be used when loading RT programs which have other as yet undefined or undeclared RT programs as "externals". All such "external RT programs" must be declared using the DECLARE-PROGRAM command before the loading process can be completed.

Example:

```
*DECLARE-PROGRAM
RT-PROGRAM: PROGR1
*DECLARE-PROGRAM PROGR2
*WRITE-PROGRAMS,,

 PROGR2  24567 ??????
 PROGR1  24543 ??????
   CDC4  24517    36     0
   CDC3  24473    36     0
   CDC2  24447    36     0
   CDC1  24423    36     0
   CDC0  24377    36     0

*
```

2.9 Name an Existing RT Description

*DEFINE-PROGRAM <rt-program name><rt-description address>

The DEFINE-PROGRAM command may be used to give a name to RT programs which are not loaded by the RT Loader. <rt-program name> is the name of the RT program and <rt-description address> is the address of the RT program's RT description.

Example:

```
*DEFINE-PROGRAM
RT-PROGRAM: TERM1
RT-DESCRIPTION ADDRESS: 23153
*DEFINE-PROGRAM TERM2 23177
*WRITE-PROGRAMS,,

  TERM2  23177    3   13
  TERM1  23153    3   11

*
```

2.10 Define Name of a Segment File

*DEFINE-SEGMENT-FILE <segment file name><segment file no>

Define the segment file number <segment file no>. The parameter <segment file name> will be the name of the segment file. If the segment file number <segment file no> is already defined, then this segment file's name and the question REDEFINE SEGMENT FILE? will be printed out, the answer Y for yes will result in the segment file's name being changed to <segment file name>.

Before using the DEFINE-SEGMENT-FILE command, the specified segment file must have been defined with the ALLOCATE-FILE command and the mass storage address of the segment file must have been inserted in the "BLST" array in the SINTRAN III system.

Example:

```
*DEFINE-SEGMENT-FILE
SEGMENT FILE NAME: (FIXED-PACK:SYSTEM)SEGFIL1:DATA
SEGMENT FILE NO.: 1
*
```

2.11 Define a Symbol

*DEFINE-SYMBOL <symbol><value>(<segment no>)

Define the symbol <symbol> on the segment (<segment no>) and give it the value <value>. The parameter (<segment no>) must be an existing segment or one of the segments currently being built. The default value of the parameter (<segment no>) is the current "load segment", the segment last loaded into the current load operation.

Example:

```
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  31
*DEFINE-SYMBOL
SYMBOL NAME: SYMB1
VALUE: 0
SEGMENT NO: 31
*DEFINE-SYMBOL SYMB2 1 31
*
```

2.12 Delete a Common Label

*DELETE-COMMON-LABEL <common label>

The common label <common label> will be deleted from the linking table.

Example:

```
*WRITE-COMMON-LABELS,,

COMLAB3          31     454
COMLAB2          31     454
COMLAB1          31     454

*DELETE-COMMON-LABEL
COMMON LABEL: COMLAB2
*DELETE-COMMON-LABEL COMLAB3
*WRITE-COMMON-LABELS,,

COMLAB1          31     454

*
```

2.13 Delete Names of Non-Reentrant Routines

*DELETE-NOT-REENTRANT

The names of the non-reentrant routines in the reentrant FORTRAN library will be deleted. This command is useful when building a reentrant system with more than one RT program on the same segment. After each RT program is loaded: define the end of the stack, delete the names of the non-reentrant routines, set the new load address (equals end of stack plus one), load the next RT program etc.

The names of the non-reentrant routines in the "reentrant" FORTRAN Library are:

> 8DXI, DEXP, DLOG, DLOG10, DSIN, DCOS, DSQRT, DATAN, DTAN2, DMOD, 8DIV, 8STAC, STPNT, STBEG, STEND, 8RTEN, 8ENTR, 8STKI

Example:

```
*WRITE-SYMBOLS,,

 STBEG   5741    31
 STPNT   5740    31
 OUTBT   5677    31
  INBT   5674    31
 8RLDN   5254    31
  ERR9   5250    31
  ERR8   5242    31
  8DMU   5521    31
  8DSB   5260    31
  8DAD   5256    31
 8ENTR    100    31
 8STAC   5704    31
 8STKI   5733    31
 8LEAV    255    31
  8FIO    306    31
 WAITF     50    31
 RESRV     52    31
 8RTEN     54    31
```

```
*DELETE-NOT-REENTRANT
*WRITE-SYMBOLS,,

  OUTBT   5677      31
   INBT   5674      31
  8RLDN   5254      31
   ERR9   5250      31
   ERR8   5242      31
   8DMU   5521      31
   8DSB   5260      31
   8DAD   5256      31
  8LEAV    255      31
   8FIO    306      31
  WAITF     50      31
  RESRV     52      31

*
```

2.14 Delete an RT Program

*DELETE-PROGRAM <rt-program name>

The RT program named <rt-program name> will be deleted from RTFIL and from the linking table, and the RT program's entry in the RT description table will be free again. When the RT program <rt-program name> is active, the DELETE-PROGRAM command is illegal.

Example:

```
*WRITE-PROGRAMS,,

   CDC4  23723    31     0
   CDC3  23677    31     0
   CDC2  23653    31     0
   CDC1  23627    31     0
   CDC0  23603    31     0

*DELETE-PROGRAM
RT-PROGRAM: CDC2
*DELETE-PROGRAM CDC0
*WRITE-PROGRAMS,,

   CDC4  23723    31     0
   CDC3  23677    31     0
   CDC1  23627    31     0

*
```

2.15 Delete a Symbol in the RTFIL

*DELETE-RTFIL-SYMBOL <symbol name> <segment no>

The symbol <symbol name> defined on the segment <segment no> will be deleted from the RTFIL.

Example:

```
*WRITE-RTFIL,,

   TW2  23603   32     0
 8RTEN     54   32
 RESRV     52   32
 WAITF     50   32
  8FIO    306   32
 8LEAV    255   32
 8STKI   5733   32
 8STAC   5704   32
 8ENTR    100   32
  8DAD   5256   32
  8DSB   5260   32
  8DMU   5521   32
  ERR8   5242   32
  ERR9   5250   32
 8RLDN   5254   32
  INBT   5674   32
 OUTBT   5677   32
 STPNT   5740   32
 STEND   5000   32
 STBEG   5741   32

*DELETE-RTFIL-SYMBOL
SYMBOL NAME: OUTBT
SEGMENT NO: 32
*DELETE-RTFIL-SYMBOL WAITF 32
*WRITE-RTFIL,,

   TW2  23603   32     0
 8RTEN     54   32
 RESRV     52   32
  8FIO    306   32
 8LEAV    255   32
 8STKI   5733   32
 8STAC   5704   32
 8ENTR    100   32
  8DAD   5256   32
  8DSB   5260   32
  8DMU   5521   32
  ERR8   5242   32
  ERR9   5250   32
 8RLDN   5254   32
  INBT   5674   32
 STPNT   5740   32
 STEND   5000   32
 STBEG   5741   32

*
```

2.16 Remove a Symbol from the Linking Table

*DELETE-SYMBOL <symbol>

The symbol named <symbol> will be deleted from the linking table. The symbol <symbol> must not be a common label or an RT program.

Example:

```
*WRITE-SYMBOLS,,

 TABP6    6575      33
 IOINI    6310      33
 8RLDN    5605      33
  ERR8    5475      33
  ERR9    5473      33
 OUTBT    6330      33
  INBT    6313      33
  8DMU    6134      33
  8DSB    5673      33
  8DAD    5671      33
 8CONV    6624      33
  8LIB     254      33
 8ENTR     124      33
 8LEAV     255      33
  8FIO     266      33
 WAITF     111      33
 RESRV     113      33
 8RTEN     115      33

*DELETE-SYMBOL
SYMBOL NAME: 8CONV
*DELETE-SYMBOL OUTBT
*DELETE-SYMBOL 8RTEN
*WRITE-SYMBOLS,,

 TABP6    6575      33
 IOINI    6310      33
 8RLDN    5605      33
  ERR8    5475      33
  ERR9    5473      33
  INBT    6313      33
  8DMU    6134      33
  8DSB    5673      33
  8DAD    5671      33
  8LIB     254      33
 8ENTR     124      33
 8LEAV     255      33
  8FIO     266      33
 WAITF     111      33
 RESRV     113      33

*
```

2.17 Dump Segment Files' Bit Map

*DUMP-SEGFILE-BITMAP (<segment file no>)(<output file>)

The bit map of the segment file (<segment no>) will be dumped on the (<output file>). There will be one bit for each page of the segment file. A bit with value one means that the corresponding page on the segment file is used, and a bit with value zero means that the corresponding page on the segment file is free. The default value for the parameter (<segment file no>) is all 4 segment files, and the default value for (<output file>) is the terminal.

Example:

```
*DUMP-SEGFIL-BITMAP
SEGMENT FILE NO.:
OUTPUT FILE:

SEGMENT FILE NO:  0
     0 177777 177777 177777 177777 177777 177777 177777 177777
   200 177777 177777 177777 177777 177777 177777 177777 177777
   400 177777 177777 177777 177777 177777 177777 177777 177777
   600 177777 177777 177777 177777 177777 177777 177777 177777
  1000 177777 177777 177777 177777 177777 177777 177777 177777
  1200 177777 177777 177777 177777 140000 000000 000000 000000
  1400 000000 000000 000000 000000
FREE PAGES ON SEGMENT FILE:   176
NUMBER OF CONTINUOUS FREE PAGES:  176
SEGMENT FILE NO:  1
     0 037777
FREE PAGES ON SEGMENT FILE:     2
NUMBER OF CONTINUOUS FREE PAGES:    2

SEGMENT FILE NO.   2 NOT DEFINED


SEGMENT FILE NO.   3 NOT DEFINED

*
```

2.18 End a Load Operation

*END-LOAD

The END-LOAD command must terminate all load operations. This command will close the segments currently being built. The segments will be moved from the scratch file to the segment file and the RTFIL. The linking table, the segment table and the RT description table will be updated. The RTFIL table will be written to the file RTFIL during the END-LOAD command. If there are undefined symbols in the linking table when an END-LOAD command is typed, the question NEGLECTING REFERENCES? will be printed out; if the answer is Y for yes then the END-LOAD command will continue, otherwise the END-LOAD command is terminated and the load operation may continue.

If the command NREENTRANT-LOAD was the last "load" command, then the file FTNLIBR will be automatically scanned in the END-LOAD command if there are undefined symbols in the linking table.

Example:

```
*NREENTRANT-LOAD 200-USER,,
NEW SEGMENT NO:  33
*END-LOAD
*
```

2.19 Exit from RT Loader

*EXIT-LOADER

This command will update the file RTFIL and then leave the RT Loader and give control to the SINTRAN III command processor.

Example:

```
@RT-LOADER


REAL-TIME LOADER 76.02.06

*EXIT-LOADER

@
```

2.20 List Available Commands

*HELP (<output file>)

This command will list all the RT Loader's commands on the (<output file>). The output will be in alphabetic order.

If the terminal is used as (<output file>), the output is divided into three parts, and for each part the RT Loader will give the question NEXT COMMANDS?. If the answer is Y for yes, then the next part is listed, otherwise the command is terminated. The terminal is the default value of the (<output file>) parameter.

Example:

```
*HELP
OUTPUT FILE:

ALLOCATE-AREA  <SEGMENT NO.> <AREA SIZE> (<LOW. ADR.>)
BACKUP-LOAD <RTFIL> <SEG.FI. 1> (<CUR.SEG.FI.>) (<SEG.FI.NO.>)
BINARY-DUMP  <OUTPUT FILE> <SEGMENT NO.> (<LOWER ADR.>) (<UPPER ADR.>)
CHANGE-LOCATION  <SEGMENT NO.>
CHANGE-RT-DESCRIPTION <RT PROG> (<PRIOR>) (<SEGNO1>) (<SEGNO2>) (<STAD
CLEAR-SEGMENT  <SEGMENT NO.>
COMPARE  <SEGMENT NO.> <FILE> (<LOW.ADR.>) (<UPPER ADR.>) (<OUTPUT FIL
DECLARE-PROGRAM  <RT-PROGRAM NAME>
DEFINE-PROGRAM  <RT-PROGRAM NAME> <RT-DESCRIPTION ADDRESS>
DEFINE-SEGMENT-FILE  <SEGMENT FILE NAME> <SEGMENT FILE NO.>
DEFINE-SYMBOL  <SYMBOL> <VALUE> (<SEGMENT NO.>)
DELETE-COMMON-LABEL  <COMMON LABEL>
DELETE-NOT-REENTRANT
DELETE-PROGRAM  <RT-PROGRAM NAME>
DELETE-RTFIL-SYMBOL  <SYMBOL NAME> <SEGMENT NO.>
DELETE-SYMBOL  <SYMBOL>
DUMP-SEGFILE-BITMAP  (<SEGMENT FILE NO.>) (<OUTPUT FILE>)
END-LOAD
EXIT-LOADER
HELP  (<OUTPUT FILE>)
```

```
NEXT COMMANDS? Y

IMAGE-LOAD  <IMAGE-FILE> <OUTPUT FILE> (<BOOTSTRAP START ADR.>)
LIST-FREE-SEGMENTS  (<OUTPUT FILE>)
LIST-REFERENCES-ADDRESS  (<SYMBOL>) (<OUTPUT FILE>)
LOAD  (<INPUT FILE>) (<LOAD-SEGMENT>) (<LINK-SEGMENT>)
NEW-SEGMENT (<SEGMENT NO.>) (<RING>) (<DEMAND/NON DEM.>) (<PROTECT BITS
NREENTRANT-LOAD  (<INPUT FILE>) (<LINK-SEGMENT>)
OCTAL-DUMP (SEGMENT NO.>) (<LOWER ADR.>) (<UPPER ADR.>) (<OUTPUT FILE>)
PARTIAL-CLEAR-RTFIL  <SYMBOL/SEGMENT NO.> (<SEGMENT NO.>)
PARTIAL-CLEAR-TABLE  <SYMBOL>
READ-BINARY  (<INPUT FILE>) (<SEGMENT NO.>)
REENTRANT-LOAD  (<INPUT FILE>) (<LINK-SEGMENT>) (<STACK LENGTH>)
REFER-SYMBOL  <SYMBOL>
RELEASE-SEGMENT  <SEGMENT NO.>
RENAME-SYMBOL  <OLD SYMBOL> <NEW SYMBOL>
REORGANIZE-SEGMENT-FILE  (<SEGMENT FILE NO.>)
RESET-LOADER
RESET-NEW-PAGE

NEXT COMMANDS? Y

SET-CORE-COMMON  <COMMON LABEL>
SET-LOAD-ADDRESS  <SEGMENT NO.> <LOAD ADDRESS>
SET-NEW-PAGE
SET-PAGE-TABLE  <PAGE INDEX TABLE NO.>
SET-SEGMENT-COMMON  <COMMON LABEL>
SET-SEGMENT-FILE  <SEGMENT FILE NO.>
WHAT-IS  <SYMBOL>
WRITE-COMMON-LABELS  (<OUTPUT FILE>)
WRITE-LOAD-ADDRESS  <SEGMENT NO.>
WRITE-NOT-REENTRANT  (<OUTPUT FILE>)
WRITE-PROGRAMS  (<OUTPUT FILE>)
WRITE-REFERENCES  (<OUTPUT FILE>)
WRITE-RTFIL  (<SEGMENT NO.>) (<OUTPUT FILE>)
WRITE-SEGMENTS  (<SEGMENT NO.>) (<OUTPUT FILE>)
WRITE-SYMBOLS  (<OUTPUT FILE>)
WRITE-TABLE  (<OUTPUT FILE>)
X-LOAD  (<INPUT FILE>) (<LOAD-SEGMENT>) (<LINK-SEGMENT>)
*
```

2.21 Load a SINTRAN III Core Only System

*IMAGE-LOAD <image file><output file> (< bootstrap start addr >)

This command will set the RT Loader in "image load" mode, i.e., loading will be to a file instead of to a segment.

The parameter < image file > is the name of the file where the SINTRAN III C system is resident in binary format. < output file> is the name of the file where the completed SINTRAN III C system will be dumped by the END-LOAD command. The parameter (<bootstrap start addr>) is the address of the bootstrap, i.e. the address where the boostrap will be placed in memory when the SINTRAN III C system is loaded and started. The default value of (<bootstrap start addr>) is the value of the load address when the load operation is terminated.

The "image load" mode is reset by the END-LOAD and the RESET-LOADER commands.

Example:

```
*IMAGE-LOAD
IMAGE FILE: CORE-SINTRAN:SYMB
OUTPUT FILE: TAPE-PUNCH
BOOTSTRAP START ADDRESS:
*SET-LOAD-ADDRESS 26000
*NREENTRANT-LOAD 200-USER
*END-LOAD
```

2.22 List Free Segment Numbers

*LIST-FREE-SEGMENTS (<output file>)

The unused segment numbers in the system will be listed on the (<output file>). Default value of (<output file>) is the terminal.

Example:

```
*LIST-FREE-SEGMENTS
OUTPUT FILE:

     34      35      36      37      40      41      42      43
     44      45      46      47      50      51      52      53
     54      55      56      57      60      61      62      63
     64      65      66      67
*
```

2.23 List Reference Addresses of Undefined Symbol

*LIST-REFERENCES-ADDRESS (< symbol>)(< output file>)

List all addresses where the undefined symbol named (< symbol>) is referred to. If no parameter (< symbol>) is specified, all undefined symbol references will be listed. Default value of the parameter (< output file>) is the terminal.

Example:

```
*WRITE-REFERENCES,,

  8LEAV
   8FIO
  WAITF
  RESRV
  8RTEN

*LIST-REFERENCES-ADDRESS
OUTPUT FILE:
SYMBOL NAME:

  8LEAV     43
   8FIO     32
  WAITF     12
  RESRV      4
  8RTEN      2
*
```

2.24 Load BRF Code onto a Segment

*LOAD (<input file>)(<load-segment>)(<link-segment>)

Load BRF code from the file (<input file>) into the segment (<load-segment>). The (<load-segment>) must have been specified in a NEW-SEGMENT command before it may be used in the LOAD command. The (<link-segment>) must be an existing segment, or one of the two segments currently being built. Link-segment means that all symbols defined on the link-segment will be available in the load operation. There must be no virtual address overlap between the load-segment and the link-segment. If no (<input file>) parameter is specified, the last input file specified will be used. If no (<load-segment>) is specified, the last segment used as load-segment or the last segment specified in a NEW-SEGMENT command will be used. Default value of the parameter (<link-segment>) is the second segment currently being built, or no link-segment if no "second" segment is specified. The parameter (<link-segment>) may be given the value zero to avoid linking to another segment in a load operation.

Example:

```
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  34
*LOAD
INPUT FILE: TW2
LOAD-SEGMENT NO.: 34
LINKING-SEGMENT NO.:
*LOAD WAITF,,
*LOAD FTNLIBR,,
*END-LOAD
*
```

2.25 Specify New Segment

*NEW-SEGMENT (<segment no>)(<ring>)(<demand/non demand>)(<protect bits>)

Allocate a segment number to be used in the current load operation. The (<segment no>) must be an available free segment number, and the default value is the first free segment number. The parameter (<ring>) specifies on which protection ring the segment will reside; legal values are 0, 1 and 2, with a default value of 0. The parameter (<demand/non demand>) specifies whether the segment will be a demand segment or non-demand segment, and the default type is non-demand. Legal values of the parameter (<demand/non-demant>) are the characters ND for non-demand and DM for demand. The parameter (<protect bits>) specifies whether the segment is to be fetch permitted, read permitted or write permitted. Legal values for this parameter are F for fetch, R for read and W for write permitted or a combination of these three characters. Default value is RFW.

A maximum of two segments may be specified by the NEW-SEGMENT command in the same load operation.

Example:

```
*NEW-SEGMENT
SEGMENT NO: 40
RING: 2
SEGMENT TYPE: DM
PROTECTION BITS: RF
*NEW-SEGMENT
SEGMENT NO:
RING:
SEGMENT TYPE:
PROTECTION BITS:
NEW SEGMENT NO:  35
*
```

In the first NEW-SEGMENT command in the example, the segment no. 40 is specified to be a demand segment on protect ring 2, and only read and fetch permitted. In the second NEW-SEGMENT command only default parameters are used and the result is that the first free segment, number 35, is allocated. This segment is non-demand, it resides on protection ring 0 and it is read, write and fetch permitted.

2.26 Load into Current Load-Segment

*NREENTRANT-LOAD (<input file>)(<link-segment>)

Load BRF code from the file (<input file>) into the current load segment, which is the last segment loaded into in the current load operation or the last segment specified in a NEW-SEGMENT command. If no current load segment exists, the first free segment number will be allocated and used as the current load segment. If a new segment is allocated, it will be a non-demand segment residing on protection ring 0 and it will be read, write and fetch permitted. The link segment (<link-segment>) must be one of the two segments currently being built or an already existing segment, or (<link-segment>) can equal zero meaning that no linking is wanted. The default value of the parameter (<link-segment>) is the last segment used as link segment or the "second" segment currently being built. The default value of the parameter (<input file>) is the last file used as (<input file>).

The file FTNLIBR, containing the FORTRAN runtime system, will be scanned (loaded from) in the END-LOAD command if there are undefined symbols, and the last load command is the NREENTRANT-LOAD command.

Example:

```
*NREENTRANT-LOAD
INPUT FILE: TW2
LINKING-SEGMENT NO.:
NEW SEGMENT NO:  35
*NREENTRANT-LOAD WAITF,,
*END-LOAD
*
```

2.27 Octal Dump of a Segment

*OCTAL-DUMP (<segment no>)(<lower addr>)(<upper addr>)
(<output file>)

Dump the specified area of the specified segment (<segment no>) in octal format on the file (<output file>). The parameter (<segment no>) must refer to an already existing segment or one of the segments currently being built or it may have the value zero meaning core common. The default value of the parameter (<segment no>) is the current load segment. (<lower addr>) is the first address and (<upper addr>) is the last address of the area to be dumped. The default value of the parameter (<lower addr>) is the first address of the segment and the default value of the parameter (<upper addr>) is the last address of the segment. The default value of the parameter (<output file>) is the terminal.

Example:

```
*OCTAL-DUMP
SEGMENT NO: 35
LOWER ADDRESS:
UPPER ADDRESS: 47
OUTPUT FILE:

     0: 135002    270    115 135603    113  11003   4037   4036
    10:   4036 135603    111   5002   4033   4031   4615  44615
    20: 172400 175040 146755 144156 140006 170401 131403 125001
    30:     42 135603    266  13000   4011      0 135604  11001
    40: 105615 135605 135001    255 124004      1      0    100
*
```

2.28 Delete Symbols in RTFIL

*PARTIAL-CLEAR-RTFIL <symbol/segment no> (<segment no>)

Delete all symbols in RTFIL defined after the specified symbol <symbol> on segment (<segment no>). If the parameter <symbol/segment no> is a segment number, then all symbols defined on this segment will be deleted from RTFIL.

Example:

```
*WRITE-RTFIL,,

   TW2   24113    35     0
 8RTEN     115    35
 RESRV     113    35
 WAITF     111    35
  8FIO     266    35
 8LEAV     255    35
 8ENTR     124    35
  8LIB     254    35
 8CONV    6624    35
  8DAD    5671    35
  8DSB    5673    35
  8DMU    6134    35
  INBT    6313    35
 OUTBT    6330    35
  ERR9    5473    35
  ERR8    5475    35
 8RLDN    5605    35
 IOINI    6310    35
 TABP6    6575    35

*PARTIAL-CLEAR-RTFIL
SYMBOL NAME/SEGMENT NO.: 8DMU
SEGMENT NO: 35
*WRITE-RTFIL,,

  8DMU    6134    35
  INBT    6313    35
 OUTBT    6330    35
  ERR9    5473    35
  ERR8    5475    35
 8RLDN    5605    35
 IOINI    6310    35
 TABP6    6575    35

*
```

2.29 Remove Symbols from the Linking Table

*PARTIAL-CLEAR-TABLE <symbol>

All symbols defined after the symbol <symbol> will be deleted from the linking table. RT program names will not be deleted by the PARTIAL-CLEAR-TABLE command.

Example:

```
*WRITE-SYMBOLS,,

  TABP6    6575       36
  IOINI    6310       36
  8RLDN    5605       36
   ERR8    5475       36
   ERR9    5473       36
  OUTBT    6330       36
   INBT    6313       36
   8DMU    6134       36
   8DSB    5673       36
   8DAD    5671       36
  8CONV    6624       36
   8LIB     254       36
  8ENTR     124       36
  8LEAV     255       36
   8FIO     266       36
  WAITF     111       36
  RESRV     113       36
  8RTEN     115       36

*PARTIAL-CLEAR-TABLE
SYMBOL NAME: 8DAD
*WRITE-SYMBOLS,,

   8DAD    5671       36
  8CONV    6624       36
   8LIB     254       36
  8ENTR     124       36
  8LEAV     255       36
   8FIO     266       36
  WAITF     111       36
  RESRV     113       36
  8RTEN     115       36

*
```

2.30 Load Binary Code onto a Specified Segment

*READ-BINARY (<input file>)(<segment no>)

Load binary code from the file (<input file>) onto the segment (<segment no>), which may be one of the segments currently being built, or one of the existing segments, or if the parameter (<segment no>) equals zero, core common. When loading into an existing segment the question CHANGING EXISTING SEGMENT? will be printed, and must be answered with Y for yes if the loading is to continue. When loading into core common, the question CHANGING CORE COMMON? will be printed, and this question must be answered with Y for yes before any loading into core common is done. If the specified segment is one of the segments currently being built, then the load address on this segment will be set equal the last address loaded into plus one. The current load address of core common will be affected in the same way when loading into core common. The default value of the parameter (<input file>) is the last file used as (<input file>), and the current load segment or the segment last specified in a NEW-SEGMENT command in the current load operation is the default value of the parameter (<segment no>).

Example:

```
*NREENTRANT-LOAD TW2,,
NEW SEGMENT NO:  36
*END-LOAD
*BINARY-DUMP BIN-DUMP:SYMB 36,,,
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  37
*READ-BINARY
INPUT FILE: BIN-DUMP:SYMB
SEGMENT NO: 37
*END-LOAD
*
```

2.31 Load Reentrant Programs onto a Segment

*REENTRANT-LOAD (<input file>)(<link-segment>)(<stack length>)

Load BRF code into the current load segment from the file (<input file>). The current load segment is the last segment loaded into in the current load operation, or the last segment specified in a NEW-SEGMENT command. If no current load segment exists, then the first free segment number will be allocated as the current load segment. This segment will be a non-demand segment, residing on protection ring 0 and it will be read, write and fetch permitted.

The (<link-segment>) may refer to one of the segments currently being built, an already existing segment or have the value zero if no linking is wanted. The default value of the parameter (<link-segment>) is the last segment used as link segment in the current load operation. The last file used as (<input file>) is default value of the parameter (<input file>).

After each REENTRANT-LOAD command the file FTNRTLIBR, containing the "reentrant" FORTRAN run-time system, is scanned if the symbol STEND (end of stack) is undefined. Then the symbol STEND is defined and the names of the non-reentrant routines are deleted from the linking table. The symbol STEND will receive a value equal to the load address after the file FTNRTLIBR is scanned plus the value of the parameter (<stack length>). The load address of the segment will be set equal to STEND plus one. 1K words is the default value of the parameter (<stack length>).

This command is useful when building a system consisting of reentrant FORTRAN programs on the same segment. The BRF code of the various RT programs should be placed on different files and one then uses a single REENTRANT-LOAD command for each RT program.

Example:

```
*REENTRANT-LOAD
INPUT FILE: REENT-TW2
LINKING-SEGMENT NO.:
STACK LENGTH: 400
NEW SEGMENT NO: 740
*END-LOAD
*
```

2.32 Release Segment Table Entry

*RELEASE-SEGMENT <segment no>

This command will release the segment <segment no> from the segment table. There will be no check as to whether any RT programs use this segment, but if the segment is currently in memory, an error message will be given, and the segment will not be released. No symbols in RTFIL or in the linking table will be deleted by this command, only the segment table entry and the segment's space on the segment file will be released.

Example:

```
*NREENTRANT-LOAD 200-USER,,
NEW SEGMENT NO:  41
*END-LOAD
*WRITE-SEGMENT 41,,

  41       0   13777   1334   0   0   1 RFW NON DEMAND
*RELEASE-SEGMENT
SEGMENT NO: 41
*WRITE-SEGMENT 41,,

SEGMENT NO. NOT USED
*
```

2.33 Make a Symbol Undefined in the Linking Table

*REFER-SYMBOL <symbol>

This command will make the symbol <symbol> undefined in the linking table. This command is useful when loading from a file containing library BRF units when the REFER-SYMBOL command may be used to select the BRF units one wishes to load.

Example:

```
*REFER-SYMBOL
SYMBOL NAME: REFSY1
*REFER-SYMBOL REFSY2
*WRITE-REFERENCES,,

 REFSY2
 REFSY1

*
```

2.34 Change Symbol Name in the Linking Table

*RENAME-SYMBOL <old symbol> <new symbol>

The symbol named <old symbol> will be renamed <new symbol>.

Example:

```
*DEFINE-SYMBOL DEFSY1 1 0
*DEFINE-SYMBOL DEFSY2 2 0
*WRITE-SYMBOLS,,

 DEFSY2        2         0
 DEFSY1        1         0

*RENAME-SYMBOL
OLD SYMBOL: DEFSY1
NEW SYMBOL: DEFSYM1
*RENAME-SYMBOL DEFSY2 DEFSYM2
*WRITE-SYMBOLS,,

DEFSYM2        2         0
DEFSYM1        1         0

*
```

2.35 Reorganize Segment File

*REORGANIZE-SEGMENT-FILE (<segment file no>)

This command will reorganize the segment file (<segment file no>) in order to avoid loss of space on the specified segment file. After this command, the segments will use a continuous area on the segment file. The segments not built by the RT Loader will not be moved, and no segments may be in use by RT programs when executing the REORGANIZE-SEGMENT-FILE command. If no parameter (<segment file no>) is specified, then all segment files used in the system are reorganized.

Example:

```
*
*DUMP-SEGFILE-BITMAP 0,,


     0 177777 177777 177777 177777 177777 177777 177777 177777
   200 177777 177777 177777 177777 177777 177777 177777 177777
   400 177777 177777 177777 177777 177777 177777 177777 177777
   600 177777 177777 177777 177777 177777 177777 177777 177777
  1000 177777 177777 170000 001777 140000 000000 037777 177777
  1200 177777 177777 140000 000000 000000 000000 000000 000000
  1400 000000 000000 000000 000000
FREE PAGES ON SEGMENT FILE:   320
NUMBER OF CONTINUOUS FREE PAGES:   236
*REORGANIZE-SEGMENT-FILE
SEGMENT FILE NO.: 0
*DUMP-SEGFILE-BITMAP 0,,


     0 177777 177777 177777 177777 177777 177777 177777 177777
   200 177777 177777 177777 177777 177777 177777 177777 177777
   400 177777 177777 177777 177777 177777 177777 177777 177777
   600 177777 177777 177777 177777 177777 177777 177777 177777
  1000 177777 177777 177777 177777 177777 177777 177777 000000
  1200 000000 000000 000000 000000 000000 000000 000000 000000
  1400 000000 000000 000000 000000
FREE PAGES ON SEGMENT FILE:   320
NUMBER OF CONTINUOUS FREE PAGES:   320
*
```

2.36 Reset RT Loader

*RESET-LOADER

This command will reset the RT Loader to its initial state, which is the state after the last EXIT-LOADER, END-LOAD or RESET-LOADER command.

Example:

```
*
*RESET-LOADER
*
```

2.37 Reset "New-Page" Mode

*RESET-NEW-PAGE

The "new page" mode is reset by this command. (See the SET-NEW-PAGE command.)

Example:

```
*
*RESET-NEW-PAGE
*
```

2.38 Allocate Common Area in Resident Core

*SET-CORE-COMMON<common label>

The common area labeled <common label> will be allocated in resident core. This command must be used before the common area<common label> is loaded.

Example:

```
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  16
*SET-CORE-COMMON
COMMON LABEL: COMLAB1
*SET-CORE-COMMON COMLAB2
*LOAD PROG1,,,
*LOAD PROG2,,,
*LOAD PROG3,,,
*WRITE-COMMON-LABELS,,

COMLAB3            16      454
COMLAB2             0      454
COMLAB1             0      454

*
```

2.39 Set Load Address of a Segment

*SET-LOAD-ADDRESS <segment no> <load address>

Set the current load address of the segment <segment no> to the value <load address>. The segment <segment no> must be one of the segments currently being loaded in, or <segment no> can have the value zero meaning core common. When core common is specified, the question CHANGING LOAD ADDRESS OF CORE COMMON? is printed, and this must be answered with Y for yes before any change of core common load address can occur.

Example:

```
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  20
*SET-LOAD-ADDRESS
SEGMENT NO: 20
ADDRESS: 40000
*WRITE-LOAD-ADDRESS
SEGMENT NO: 20

L.ADR:   40000   U.ADR:   40000   C.LADR:   40000
*
```

2.40 Start Each Load on a New Page

*SET-NEW-PAGE

This command will set the RT Loader in "new page" mode, i.e. the current load address will be set to the start of a new page for each new BRF unit loaded. The "new page" mode will be reset by the RESET-LOADER, END-LOAD and the RESET-NEW-PAGE commands.

The following example shows the difference (note the addresses of the symbols) between loading a program with and without using the SET-NEW-PAGE command.

Example:

```
*SET-NEW-PAGE
*NREENTRANT-LOAD TW2,,
NEW SEGMENT NO:  16
*LOAD FTNLIBR,,
*WRITE-SYMBOLS,,

  TABP6  32004      16
  IOINI  26000      16
  8RLDN  15320      16
   ERR8  15210      16
   ERR9  15206      16
  OUTBT  26020      16
   INBT  26003      16
   8DMU  22000      16
   8DSB  20002      16
   8DAD  20000      16
  8CONV  32033      16
   8LIB   6137      16
  8ENTR   6007      16
  8LEAV   6140      16
   8FIO  10001      16
  WAITF   2000      16
  RESRV   4000      16
  8RTEN   6000      16

*RESET-LOADER
*NREENTRANT-LOAD TW2,,
NEW SEGMENT NO:  16
*LOAD FTNLIBR,,
*WRITE-SYMBOLS
OUTPUT FILE:

  TABP6    6575      16
  IOINI    6310      16
  8RLDN    5605      16
   ERR8    5475      16
   ERR9    5473      16
  OUTBT    6330      16
   INBT    6313      16
   8DMU    6134      16
   8DSB    5673      16
   8DAD    5671      16
  8CONV    6624      16
   8LIB     254      16
  8ENTR     124      16
  8LEAV     255      16
   8FIO     266      16
  WAITF     111      16
  RESRV     113      16
  8RTEN     115      16

*
```

2.41 Specify Page Index Table

*SET-PAGE-TABLE <page index table no>

This command will specify which page index table the currently loaded segments and RT programs are to use. Usually page index table 1 is used for the RT programs and for core common, and page index table 1 is the default value used if other values are not set by the SET-PAGE-TABLE command. The commands RESET-LOADER and END-LOAD will always reset the page index table to 1.

Example:

```
*
*SET-PAGE-TABLE
PAGE INDEX TABLE NO.: 3
*
```

2.42 Allocate Common Area on Second New Segment

*SET-SEGMENT-COMMON <common label>

The common area labeled <common label> will be allocated on the segment specified in the second NEW-SEGMENT command in a load operation. This command must be used before the common label <common label> is defined.

Example:

```
*NEW-SEGMENT,,,,
NEW SEGMENT NO:   16
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  20
*SET-SEGMENT-COMMON
COMMON LABEL: COMLAB3
*SET-SEGMENT-COMMON COMLAB1
*LOAD PROG1,16,20
*LOAD PROG2,,
*LOAD PROG3,,
*LOAD SUBR,,
*WRITE-COMMON-LABELS,,

COMLAB2         16     454
COMLAB1         20     454
COMLAB3         20     454

*
```

2.43 Specify Segment File for New Segment(s)

*SET-SEGMENT-FILE <segment file no>

The segment file where the segments currently being built are to reside is specified by this command. This "current segment file" can only be changed by this command.

Example:

```
*
*SET-SEGMENT-FILE
SEGMENT FILE NO.: 0
*
```

2.44 Print Information about Specified Symbol

*WHAT-IS < symbol >

This command will print all information about all symbols in the linking table and the RTFIL, with the name < symbol>.

Example:

```
*
*WHAT-IS
SYMBOL NAME: CDC1

   CDC1   24447   16   20 DEFINED RT-PROGRAM
*
```

2.45 List Common Label Names in Linking Table

*WRITE-COMMON-LABELS (< output file >)

List the names, addresses and the segment numbers of all the common labels defined or declared in the linking table, on the file (< output file >). The terminal is the default value of the parameter (< output file >).

Example:

```
*WRITE-COMMON-LABELS
OUTPUT FILE:

COMLAB3          22     454
COMLAB2          22     454
COMLAB1          22     454

*
```

2.46 Write Address Limits and Load Address

*WRITE-LOAD-ADDRESS <segment no>

Write the lowest virtual address, the highest virtual address and the current load address of the specified segment <segment no>. This segment must be one of the segments currently being built. When the value zero is given for the parameter <segment no>, the addresses of core common are listed.

Example:

```
*NREENTRANT-LOAD 200-USER,,
NEW SEGMENT NO:  22
*WRITE-LOAD-ADDRESS
SEGMENT NO: 22

L.ADR:       0  U.ADR:  13776  C.LADR:  13776
*
```

2.47 List Non-Reentrant Runtime Routines in Linking Table

*WRITE-NOT-REENTRANT (<output file>)

List the names and the values of the non-reentrant FORTRAN runtime routines defined. This command is useful when loading a reentrant system and should be used before the command DELETE-NOT-REENTRANT in order to see the addresses of the routines which will be deleted by the DELETE-NCT-REENTRANT command.

The default value of the parameter (<output file>), is the terminal.

Example:

```
*
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  22
*LOAD REENT-TW2,,
*LOAD FTNRTLIBR,,
*WRITE-LOAD-ADDRESS 22
```

```
L.ADR:       0  U.ADR:   5741  C.LADR:   5742
*DEFINE-SYMBOL STEND 6344,,
*SET-LOAD-ADDRESS 22 6344
*WRITE-NOT-REENTRANT
OUTPUT FILE:

  8STAC   5704
  STPNT   5740
  STBEG   5741
  STEND   6344
  8RTEN     54
  8ENTR    100
  8STKI   5733
*
```

2.48 List RT Program Names

*WRITE-PROGRAMS (<output file>)

List the names of all the RT programs defined and declared on the file (<output file>).

Each of the RT program's two segment numbers, and the address of each RT program's RT description will also be listed. Declared RT programs will not have segment numbers, so question marks will be written instead of segment numbers.

The default value of the parameter (<output file>) is the terminal.

Example:

```
*WRITE-PROGRAMS
OUTPUT FILE:

 PROGR1  24567 ??????
 PROGR2  24377 ??????
    TW2  24353    22     0

*
```

2.49 List Undefined Symbols

*WRITE-REFERENCES (<output file>)

All undefined symbols in the linking table will be listed on the (<output file>). The terminal is the default value of the parameter (<output file>).

Example:

```
*WRITE-REFERENCES
OUTPUT FILE:

  8LEAV
   8FIO
  WAITF
  RESRV
  8RTEN

*
```

2.50 List Symbols in RTFIL

*WRITE-RTFIL (<segment no>)(<output file>)

List all the symbols with the segment number (<segment no>) on the file (<output file>). If no (<segment no>) is specified, all the symbols in RTFIL will be listed. The terminal is the default value of the parameter (<output file>).

Example:

```
*
*WRITE-RTFIL
SEGMENT NO: 22
OUTPUT FILE:

     TW2  24353    22      0
   8RTEN    115    22
   RESRV    113    22
   WAITF    111    22
    8FIO    266    22
   8LEAV    255    22
   8ENTR    124    22
    8LIB    254    22
   8CONV   6624    22
    8DAD   5671    22
    8DSB   5673    22
    8DMU   6134    22
    INBT   6313    22
   OUTBT   6330    22
    ERR9   5473    22
    ERR8   5475    22
   8RLDN   5605    22
   IOINI   6310    22
   TABP6   6575    22

*
```

2.51 List Information about Segment

*WRITE-SEGMENTS (<segment no>)(<output file>)

List all information about the specified segment (<segment no>). The information listed is the segment number, the segment's lower and higher virtual addresses, the mass storage address (in pages) relative to the start of the segment file, the segment file number, the page index table number, on which protect ring the segments reside, the memory protection type (demand/non-demand).

If no parameter (<segment no>) is specified, then all segments are listed. When the value zero is given for the parameter (<segment no>), the address limits of the core common area are listed. The terminal is the default value of the parameter (<output file>).

Example:

```
*WRITE-SEGMENTS
SEGMENT NO:
OUTPUT FILE:
S.NO. L.ADR  U.ADR  M.ADR   SF RI PT

   1       0 173777     0   0  1  0 RFW NON DEMAND
   2       0  43777     0   0  2  2 RFW NON DEMAND
   3   52000  75777    25   0  2  0 RFW DEMAND
   4   50000 147777    37   0  2  0 RFW DEMAND
   5   44000  47777    22   0  2  0 RFW NON DEMAND
   6   50000 133777   110   0  2  0 RFW DEMAND
   7   50000  73777   142   0  2  0 RFW NON DEMAND
  10       0  77777   154   0  2  2 RFW NON DEMAND
  11   40000  47777   214   0  2  0 RFW NON DEMAND
  12       0 177777   220   0  0  2 RFW DEMAND
  13   40000  47777   320   0  2  0 RFW NON DEMAND
  14       0 177777   324   0  0  2 RFW DEMAND
  15   40000  47777   424   0  2  0 RFW NON DEMAND
  16       0 177777   430   0  0  2 RFW DEMAND
  17   40000  47777   530   0  2  0 RFW NON DEMAND
  20       0 177777   534   0  0  2 RFW DEMAND
  21   40000  47777   634   0  2  0 RFW NON DEMAND
  22       0 177777   640   0  0  2 RFW DEMAND
  23   40000  47777   740   0  2  0 RFW NON DEMAND
  24       0 177777   744   0  0  2 RFW DEMAND
CORE COMMON AREA:  174000 177777
*
```

2.52 List Defined Symbols in the Linking Table

*WRITE-SYMBOLS (<output file>)

List the names, the segments and the values of all defined symbols in the linking table, on the file (<output file>). The terminal is the default value of the parameter (<output file>).

Example:

```
*WRITE-SYMBOLS
OUTPUT FILE:

  TABP6    6575     23
  IOINI    6310     23
  8RLDN    5605     23
   ERR8    5475     23
   ERR9    5473     23
  OUTBT    6330     23
   INBT    6313     23
   8DMU    6134     23
   8DSB    5673     23
   8DAD    5671     23
  8CONV    6624     23
   8LIB     254     23
  8ENTR     124     23
  8LEAV     255     23
   8FIO     266     23
  WAITF     111     23
  RESRV     113     23
  8RTEN     115     23

*
```

2.53 List Information about Symbols in Linking Table

*WRITE-TABLE (<output file>)

List on the file (<output file>) all information about all symbols of all types, in the linking table. The default value of the parameter (<output file>) is the terminal.

Example:

```
*WRITE-TABLE
OUTPUT FILE:

  8LEAV              30       REFERENCED SYMBOL
   8FIO              30       REFERENCED SYMBOL
  WAITF              30       REFERENCED SYMBOL
  RESRV              30       REFERENCED SYMBOL
  8RTEN              30       REFERENCED SYMBOL
    TW2   24707      30     0 DEFINED RT-PROGRAM
   CDC4   24663      23     0 DEFINED RT-PROGRAM
   CDC3   24637      23     0 DEFINED RT-PROGRAM
   CDC2   24613      23     0 DEFINED RT-PROGRAM
   CDC1   24567      23     0 DEFINED RT-PROGRAM
   CDC0   24377      23     0 DEFINED RT-PROGRAM

*
```

2.54 Load Library BRF Units

*X-LOAD (<input file>)(<load-segment>)(<linking-segment>)

When using this command in a load operation, then BRF units with library format will be loaded if the library symbol of the BRF unit is either undefined or does not exist in the linking table. If the library symbol of the BRF unit is defined in the linking table, then the BRF unit will be skipped. Otherwise this command has the same function as the LOAD command.

Example of loading the FORTRAN runtime system FTNLIBR (which is in library format) into segment number 26:

```
*NEW-SEGMENT
SEGMENT NO:
RING:
SEGMENT TYPE:
PROTECTION BITS:
NEW SEGMENT NO:  30
*X-LOAD
INPUT FILE: FTNLIBR
LOAD-SEGMENT NO.:
LINKING-SEGMENT NO.:
*END-LOAD
*WRITE-SEGMENT 30,,

  30      0  17777   467  0  0  1 RFW NON DEMAND
*
```

3 EXAMPLES

3.1 An example of compiling the FORTRAN program PROGA, loading it into a segment and starting the program with the RT command. The RT program PROGA is a program to write the message "THIS IS PROGRAM PROGA CALLING".

```
@

@FTN

NORD FTN
$COM PROGA,0,PROGA
7 STATEMENTS COMPILED
$EX
@RT-LOADER


REAL-TIME LOADER 76.02.06

*NREENTRANT-LOAD PROGA,,
NEW SEGMENT NO:  35
*END-LOAD
*EXIT-LOADER

@RT PROGA

@LOG
 16.21.04     20 FEBRUARY   1976
--EXIT--

THIS IS PROGRAM PROGA CALLING
```

3.2 An example of loading the reentrant FORTRAN input/output system FIO into a segment, and then loading 2 reentrant RT programs PROGA and PROGB into other segments and linking them to the segment containing FIO. The entry point 8FIO must be referenced to extract FIO from the file FTNRTLIBR.

```
@FTN

NORD FTN
$RT
$COM PROGA,0,REENT-PROGA
7 STATEMENTS COMPILED
$RT
$COM PROGB,0,REENT-PROGB
7 STATEMENTS COMPILED
$EX
@RT-LOADER


REAL-TIME LOADER 76.02.06

*NEW-SEGMENT,,,,
NEW SEGMENT NO:  33
*SET-LOAD-ADDRESS 33 150000
*REFER-SYMBOL 8FIO
*LOAD FTNRTLIBR,,
*END-LOAD
*REENTRANT-LOAD
INPUT FILE: REENT-PROGA
LINKING-SEGMENT NO.: 33
STACK LENGTH: 1000
NEW SEGMENT NO:  34
*END-LOAD
*REENTRANT-LOAD REENT-PROGB 33 1000
NEW SEGMENT NO:  35
*END-LOAD
*
```

3.3 An example of loading 3 reentrant RT programs PROG1, PROG2 and PROG3 to 3 different segments. All three RT programs call the reentrant subroutine SUBR, and this subroutine is loaded into a segment which will be common for the three RT programs.

```
@

@FTN

NORD FTN
$RT
$COM PROG1,0,REENT-PROG1
6 STATEMENTS COMPILED
$RT
$COM PROG2,0,REENT-PROG2
6 STATEMENTS COMPILED
$RT
$COM PROG3,0,REENT-PROG3
6 STATEMENTS COMPILED
$RT
$COM SUBR,0,REENT-SUBR
4 STATEMENTS COMPILED
$EX
@RT-LOADER


REAL-TIME LOADER 76.02.06

*NEW-SEGMENT,,,,
NEW SEGMENT NO:  36
*SET-LOAD-ADDRESS 36 100000
*LOAD REENT-SUBR,,
*LOAD FTNRTLIBR,,
*END-LOAD
*REENTRANT-LOAD REENT-PROG1,36,1000
NEW SEGMENT NO:  37
*END-LOAD
*REENTRANT-LOAD REENT-PROG2,36,1000
NEW SEGMENT NO:  40
*END-LOAD
*REENTRANT-LOAD REENT-PROG3,36,1000
NEW SEGMENT NO:  41
*END-LOAD
*
```

3.4 An example of loading the three reentrant RT programs PROG1, PROG2 and PROG3 into the same segment.

```
@

@FTN

NORD FTN
$RT
$COM PROG1,0,REENT-PROG1
6 STATEMENTS COMPILED
$RT
$COM PROG2,0,REENT-PROG2
6 STATEMENTS COMPILED
$RT
$COM PROG3,0,REENT-PROG3
6 STATEMENTS COMPILED
$RT
$COM SUBR,0,REENT-SUBR
4 STATEMENTS COMPILED
$EX
@RT-LOADER


REAL-TIME LOADER 76.02.06

*REENTRANT-LOAD REENT-SUBR,,,
NEW SEGMENT NO:  42
*REENTRANT-LOAD REENT-PROG1,,1000
*REENTRANT-LOAD REENT-PROG2,,1000
*REENTRANT-LOAD REENT-PROG3,,1000
*END-LOAD
*
```

3.5 An example of loading a common area named COMMLAB to one segment, and then load the two RT-programs COMPRO1 and COMPRO2 into other segments and link these segments to the "COMMON" segment, i.e. the two RT-programs both refer to the COMMON area COMMLAB.

```
@FTN

NORD FTN
$COM COMPRO1,0,COMPRO1
5 STATEMENTS COMPILED
$COM COMPRO2,0,COMPRO2
5 STATEMENTS COMPILED
$EX
@RT-L


REAL-TIME LOADER 76.01.07

*NEW-SEGMENT,,,,
NEW SEGMENT NO:  40
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  41
*SET-SEGMENT-COMMON COMMLAB
*LOAD COMPRO1,40,41
*LOAD FTNLIBR,,,
*END-LOAD
*NEW-SEGMENT,,,,
NEW SEGMENT NO:  42
*LOAD COMPRO2,42,41
*LOAD FTNLIBR,,,
*WRITE-TABLE,,,

   8LIB     217  42        DEFINED SYMBOL
  8ENTR      67  42        DEFINED SYMBOL
  8LEAV     220  42        DEFINED SYMBOL
  RESRV      56  42        DEFINED SYMBOL
  8RTEN      60  42        DEFINED SYMBOL
COMPRO2   21115  42   41   DEFINED RT-PROGRAM
COMMLAB 1000000  41        DEFINED COMMON LABEL    , SIZE:      454
COMPRO1   21071  40   41   DEFINED RT-PROGRAM

*END-LOAD
*
```

A/S NORSK DATA-ELEKTRONIKK
Lørenveien 57, Oslo 5 - Tlf. 21 73 71

COMMENT AND EVALUATION SHEET

ND-60.051.03

SINTRAN III
REAL TIME LOADER

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

FROM: ____________________

____

____