RCSL No: 31-D607

Edition: April 1983

Author: Finn G. Strøbech

Title:

System 3 Utility Programs, Part Two
User's Guide.

RCSL No 42-i 2164

A/S REGNECENTRALEN
af 1979

Keywords:

RC8000, RC4000, Basic Software, File Processor, Utility Programs, Users Guide.

Abstract:

This second part of the utility program manual contains detailed descriptions of the individual programs performing catalog handling, data handling and job control.

(210 printed pages)

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL No 42-I 2164

FOREWORD

First Edition: RCSL No 31-D233.
The file processor and the utility programs in system 3 are based on system 2 versions. The necessary changes in the programs and coding of new programs were done by Tove Ann Aris, Bo Tveden-Jørgensen, Jørgen Zachariassen, and the author.

The advices and corrections from Christian Gram and Tove Ann Aris have been of great help during the preparation of this manual.

Hans Rischel
A/S Regnecentralen, April 1973

Third edition: RCSL No 31-D345.
This edition is similar to second edition with below exceptions.
Following descriptions have been added: assign, changeentry, char, correct, edit, headpunch, and setmt, clearmt.

Following descriptions have been changed: account, backfile, bossjob, catsort, claim, convert, copy, entry, head, i, load, move, newjob, o, print, save, and scope.

Tove Ann Aris
A/S Regnecentralen, September 1974

Fourth edition: RCSL No 31-D494.
Following descriptions have been added: procsurvey, rubout.
Following have been totally rewritten: save, load.

Following have been changed: binout, catsort, changeentry, char, clearmt, copy, convert, correct, edit, entry, head, headpunch, mode, move, replace, set, and setmt.

Tove Ann Aris
A/S Regnecentralen, March 1977

Fifth edition: RCSL No 31-D590.
Following description has been added: label.
Following have been changed: binin, catsort, copy, finis, load, print, and save.

Tove Ann Aris
A/S Regnecentralen, April 1978

Sixth edition: RCSL No 31-D607.
Following decriptions have been added: claimtest, job, permanent, translated.
Following decriptions have been changed: assign, catsort, claim, claimtest, clear, copy, finis, job, load, print, and save.
For the sake of clarity in TABLE OF CONTENTS, the subsections in Chapter 3 are not mentioned.

Finn G. Strøbech
A/S Regnecentralen af 1979, April 1983

TABLE OF CONTENTS PAGE

TABLE OF CONTENTS (continued) PAGE

1. INTRODUCTION 1.

Part I of the Utility Program Manual gives a general introduction to the file processor and utility program system and a detailed description of certain important features of the system.

The file processor and the utility programs in system 3 are based on the system 2 versions. The necessary changes in the programs and coding of new programs were done by Tove Ann Aris, Bo Tveden-Jørgensen, Jørgen Zachariassen, and Hans Rischel.

This second part of the Utility Program Manual contains detailed descriptions of the individual programs.

Chapter 2 contains a list of programs together with a short abstract of each program.
The list is divided in three sections, one for each category of programs: catalog handling programs, data handling programs, and job control programs.

Chapter 3 contains the detailed descriptions of the programs, one section for each, and all in alphabetic order.

2. ABSTRACTS

2.1 Catalog Handling Programs

assign	Creates or changes a temporary entry so that the tail of the two specified entries become identical.
backfile	Subtracts one from the file number (unless it is 0) in the tails of the entries specified and signals reach of file 0.
catsort	Lists on current output selected parts of the main catalog (or any subcatalog) sorted according to the parameters. At last also total number of entries and segments output are listed.
changeentry	Changes an existing catalog entry according to the parameters in the call. The program is a supplement to the programs "set" and "entry" and is used when one wants to change some of the elements in the entry tail by copying from the tails of other catalog entries.
clear	Removes catalog entries with name and scope as specified.
clearmt	Removes catalog entries according to the parameters.
entry	Creates or changes a temporary catalog entry according to the parameters in the call. The program is a supplement to the program "set" and is used when one wants to set some of the elements in the entry tail by copying from the tails of other entries.
lookup	Finds and lists catalog entries with specified name.

nextfile	Adds one to the file number in the tail of the catalog entries specified.
permanent	Changes the permanent key of the specified entry to the specified integer.
procsurvey	Lists types of procedures and their parameters, as well as the procedure date.
rename	Changes the names of catalog entries as specified.
scope	Changes the scope of catalog entries as specified in the call of the program.
search	Finds and lists all catalog entries with a given scope.
set	Creates a new catalog entry with scope temp or changes an already existing entry (with scope temp) according to the parameters.
setmt	Creates catalog entries of scope temp describing files one magnetic tape according to the parameters.

2.2 Data Handling Programs 2.2

binin	The program can input files generated by the program "binout". The program "binin" and "binout" are primarily used when binary files are stored on paper tape.
binout	The program can output catalog entries and contents of files in a format (a binout file) which may be input by the program "binin" or the program "initialize catalog". The program can furthermore output autoload tapes.

char	Outputs the specified character the specified number of times.
copy	Copies one or several files into another file and calculates the number of characters copied and the sum of their ISO values. Blind characters are not copied. The program can be used instead of "edit" if only a simple copying is wanted. Furthermore the program may be used for check reading of text files (e.g. texts punched on paper tape).
correct	The program corrects specified words on the backing storage according to the parameters. The program may also be used to print specified bits as integers.
edit	Edit is a line oriented program for editing of text files.
head	Prints a number of form feeds and a page head containing the name of the job and the date.
headpunch	The program punches a readable text pattern according to the parameters. The same information is also written on current output.
label	Outputs a BOSS label on file 0 of the specified magnetic tape.
load	The program can input catalog entries and bs-files from a magnetic tape file generated by the program "save".
message	May be used (together with "head") to make headings on the output. The parameter list in the call of message is simply output when the program is called.

move — Performs blockwise copying of files on backing storage or magnetic tape.

print — Prints from a backing storage area or directly from the core store with specified formats. The program is primarily intended for printing of dumped core areas.

rubout — Rubs out the contents of the specified backing storage files. If demanded the catalog entry is removed after the cleaning.

save — The program can output catalog entries and bs-files to a magnetic tape file for later reestablishment by the program "load".

translated — Prints the data of translation which is found in all ALGOL/FORTRAN programs.

2.3 Job Control Programs 2.3

account — Sends an account message to the parent (operating system) who is then expected to produce a record in the account file. Only used when jobs running under BOSS wants to produce special account information.

bossjob — Sends a newjob message to BOSS (the internal process named BOSS) demanding the specified file enrolled as job file in an off line job. In this way a job running under another operating system may create a BOSS job. The actual job continues with the next FP command. Further details are found in sec. 1.3, newjob and replacejob, in ref. [10].

change — Sends a change paper message to the parent (operating system). The program is only used when a job executed under BOSS uses job controlled printer. (cf. ref. [10], section 6.2).

claim — Lists selected parts of the bs area claims of the process together with area, buf, size and first core.

claimtest — Checks the claims of the calling process according to the call parameters and leaves the ok bit true if the claims specified are present, false otherwise.

convert — Sends a convert message to the parent (the operating system) who is then expected to print the specified backing storage area. A file with scope login is not accepted and the file must not be in use (for instance the file must not be current output). A temporary file converted will immediately disappear from the reach of the job. Each convert operation performed by BOSS requires a cbuffer which must be reserved in the job specification (cf. ref. [10], ch. 3 and sec. 6.2).

corelock — Sends a corelock message to the parent (the operating system) demanding that the job should stay in the core the specified number of seconds. This feature is only used in connection with process control devices producing data with a high rate, cf. ref. [10], secs. 3.4 and 6.7.

coreopen — Sends a coreopen message to the parent (the operating system) signalling the end of a corelock period (cf. the program "corelock"). The program is only used on process control installations.

end — Returns current input to previous current input at the positions where it was left.

finis — Finis terminates the job.

i — Selects a new file as current input. The former file may later be resumed at the position where it was left (for instance by a call of "end").

if	Makes the execution of the next FP command conditioned by the values of one (or several) mode bits. The condition may reflect the success of the latest program end (or it may correspond to the mode bits as set by a call of the program "mode").
kit	Sends a mount disc message to the parent (the operating system) demanding a disc kit with a specified name to be mounted on a specified disc unit (cf. ref. [10], ch. 3 and sec. 5.3).
job	Makes it possible to use tapes containing a BOSS job request in runs directly under "s".
mode	Changes the FP mode bits specified in the call and may thereby change the working cycle of FP.
mount	Sends a mount message to the parent (the operating system) who is then expected to ask the operator to mount the tape reel (cf. ref. [10], sec. 6.1). The program does not await the mounting, unless there is asked for mounting of an unspecified worktape.
mountspec	Sends a mount special message to the parent (the operating system) limiting a later mounting of the specified magnetic tape reel to the station with the specified device number (cf. ref. [10], sec. 6.1).
newjob	Sends a newjob message to the parent (operating system) demanding the specified file enrolled as job file in a new off line job i.e. in this way a new job is created. The actual job continues with the next FP command. Further details are found in sec. 1.3, newjob and replacejob, in ref. [10].
o	Selects a new file as current output.

online — Turns the job into the conversational mode where the current input to the job is typed on the terminal at run time. A conversational job is very resource demanding and the user must have a special option in the user catalog (cf. ref. [10], sec. 3.2).

opcomm — Sends the parameter list in the call as a print message to the parent (the operating system) with request for an answer from the operator and types the answer (when received) on current output.

opmess — Sends the parameter list in the call as a print message to the parent (the operating system). If the operating system is BOSS the message is typed on the main console.

release — Sends a release message to the parent (the operating system) releasing the specified magnetic tape reel (cf. ref. [10], sec. 6.1).

repeat — The program makes it possible to repeat (a specified number of times) a series of FP commands placed in brackets.

replace — Sends a replace message to the parent (the operating system) defining a file as replacement for the current job file. After termination of the job BOSS will create a new job with the same name and the specified file as job file. A replace message from an on line is not accepted by BOSS.

ring — Sends a mount ring message to the parent (the operating system). The program is normally not used as the software sends the mount ring message automatically when needed.

skip — Bypasses parts of current input as specified in the parameter list.

suspend — Sends a suspend message to the parent (the operating system) asking for suspension of the specified magnetic tape reel. This is relevant for worktapes only. The station is now available for mounting of another tape reel, but the suspended worktape is still reserved for the job until it terminates or releases the tape reel.

Each suspend operation uses a suspend buffer. (cf. ref. [10], sec. 6.1).

timer — Sends a timer message to the parent (the operating system) demanding a provoked interupt after a certain time.

3. THE PROGRAMS 3.

3.1 account 3.1

Sends an account message to the parent (the operating system) who is then expected to produce a record in the account file. Only used when jobs running under BOSS wants to produce special account information.

3.1.1 Call 3.1.1

account <s> <account kind> $\left\{ \text{<s> <integer>} \right\}_{1}^{3}$

where the parameters <account kind> and <integer> are integers.

3.1.2 Function 3.1.2

The program sends an account message containing the intergers.

3.1.3 Storage Requirements 3.1.3

1536 bytes plus space for FP.

3.1.4 Error Messages 3.1.4

***account call
The program was called with a left hand side.

***account <parameter list> parameter error
Parameter error in the call of the program.

***account <parameter list> kind illegal
The account kind was not accepted by the operating system.

In case of any no account record is produced.

3.2 assign 3.2

Creates or changes a tempoary entry so that the new entry becomes a sub entry to the old one, if the old one is an area entry, and the two entries become identical if the old one is a non-area entry. The program is used together with the programs entry and nextfile.

3.2.1 Example 3.2.1

The programmer wants to set an entry in the file longname and instead of

```
new=entry longname longname longname longname longname,
          2.6 longname
```

the following commands are used

```
t=assign longname
new=entry  t t t t t 2.6
```

The program calls:

```
nextfile tape
...
...
nextfile tape
programfile=assign tape
```

will set progfile equal to the current value of tape.

3.2.2 Call 3.2.2

<resultname> = assign <oldname>

3.2.3 Function 3.2.3

If <oldname> is an area entry, <resultname> will become a bs-entry, i.e. modekind = 1<23+4 and document name = oldname>. If <oldname> is a non-area entry, <resultname> will become identical to <oldname>.

3.2.4 Storage Requiments

1536 bytes plus space for FP.

3.2.5 Error Messages

***assign call
No left hand side in the call of the program.

***assign param <parameter>
Parameter error in the call of the program.

***assign <oldname> unknown
The file <oldname> was not found in the catalog.

***assign <reultname> change kind impossible
A change of an area entry to a non-area entry or vice versa was attempted.

***assign <result name> change bs device impossible
A change of kit/doc name of an area entry was attempted.

***assign <result name> bs device unknown
The bs device specified was not found.

***assign <result name> no resources
The resources of the job did not allow the wanted creation or change of an entry.

***assign <result name> no room
The wanted creation of an entry was not possible because the name overflow for the new name in the main catalog exceeded the limit.

***assign <result name> entry in use
The entry could not be changed because another job was using it.

If any message appears no entry is created or changed.

3.3 backfile 3.3

Substracts one from the file number (unless it is 0) in the tails of the entries specified and signals reach of file 0.

Examples

If the catalog entries old and describe file 4 of magtape mt310514 and file 2 of mt310515 respectively, then the command

```
backfile old new
```

will change old to describe file 3 of mt310514 and new to describe file 1 of mt310515. A repeated call will change old to decribe file 2 and new to describe file 0 and set the warning bit to yes. A following call will change old to descibe file 1 and leave new unchanged - the ok bit is set to no.

3.3.1 Call 3.3.1

backfile $\left\{ \text{<s><name>} \right\}_{1}^{\infty}$

3.3.2 Function 3.3.2

For each name in the list a catalog lookup is made and the file number in the tail of the entry is decreased by one unless it is zero.

If any file number becomes zero then the warning bit is set to yes.

If any file number already was zero then the ok bit is set to no.

3.3.3 Storage Requirements 3.3.3

Space for FP

3.3.4 Error Messages

***backfile call

Left hand side in the call. Program terminates without further actions.

***backfile <name> param

Parameter error. The faulty parameter starting with the name specified is skipped and the program continues with the next parameter.

***backfile param

Same as above except that the faulty parameter does not start with a name.

***backfile <name> unknown

No entry with the specified name was found. The program continues with the next parameter.

***backfile <name> protected

The job was not allowed to change the tail in the entry found. The program continues with the next parameter.

If any error message occurs then the ok bit is set to no.

3.4 binin 3.4

The program can input files generated by the program "binout". The programs "binin" and "binout" are primarily used when binary files are stored on paper tape.

3.4.1 Example 3.4.1

A paper tape was produced by the FP command:

```
tpo=binout fup
```

It may be loaded by the FP command

```
binin tro
```

and thereby the catalog entry 'fup', the area and its contents are reestablished. When using BOSS one should load the tape by a load command in the job specification like this

```
load trn pip
```

and then get the file 'fup' by the following call of binin in the job file:

```
binin pip
```

Note, that the no parity mode is used in the load command.

3.4.2 Call 3.4.2

$$\left\{\text{<other output> =}\right\}_{0}^{1} \quad \text{binin} \quad \left[\text{list.} \left\{\begin{matrix}\text{yes}\\ \text{no}\end{matrix}\right\}_{0}^{1} \left\{\text{<binout file>} \left\{\text{.<modifier>}\right\}_{0}^{1}\right\}^{4}\right]$$

<other output>::= <name of output file>

<binout file>::= <name of input file>

$$\text{<modifier>::=} \quad \left\{\begin{matrix}\text{<binout segments>}\\ \text{s.<binout segments>}\\ \text{c.<binout segments>}\end{matrix}\right\}$$

$$\text{<binout segments>::=} \quad \left\{\begin{matrix}\text{<no of binout segments>}\\ \text{<no of binout segments> . <first binout segment>}\end{matrix}\right\}$$

If the parameter list.yes is specified, all entry names found are listed on current out.

The input to "binin" is a number of binout files, each consisting of a number of binout segments. A binout segment is a stream of 8-bit characters with odd parity, the second bit of each character being 0. A binout segment is terminated by a sum character, a character with the second bit being 1. A binout segment input by "binin" is transformed to a number of words, each composed of the rightmost 6 bits of 4 characters. the rightmost 6 bits of the sum character form the sum modulo 64 of all other characters in the binout segment; this sum is checked by "binin". "binin" scans the parameter list from left to right, and loads the sequence of binout segments defined by the binout files. When a file is exhausted, the input is continued from the file decribed by the next element in the parameter list, and when it is exhausted, the execution of "binin" is terminated.

The left side in the call of "binin" determines how the binout segments are interpreted:

1) <other output> is not present.
   The very first binout segment is input and interpreted as a command segment. The commands in the command segment are executed one by one, and when the command segment is exhausted, the next binout segment includes a load command, a number of binout segments following the present command segment is input and moved to backing store or magnetic tape as defined by the load command. The following segment is interpreted as a command segment and so on. A tape produced by "binout" may be read in this way.

2) <other output> is present.
   All binout segments of the binout files are interpreted as load segments and loaded to the file by <other output>.

A command segment must not exceed 256 words; a load segment can be of any length.

3.4.4 Modifier

<binout segment> This modifier has only effect if other output is specified (left side in call occurs). The first <first segment> binout segments of the actual in-file are skipped, and only <no of binout segments> binout segments are loaded to the output file. If <first segment> does not occur, no segments are skipped.

.s The modifier causes each load segment to be preceded by one word in the output. This word is an integer which is the length of the entire segment (no of bytes). The last segment is terminated by a word being 0.

.c This modifier causes the binout file to be checked only; i.e. the commands in the command segments are checked for syntax errors, and only the <:end:> command is executed. The sums of all binout segments are checked, but no load segments are output to the files specified.

3.4.5 Commands

"binin" uses the same command language as the program "initialize catalog" (cf. ref. 2).

A command in a command segment is identified by a textstring consisting of at most 6 ISO characters (including NULL characters). This textstring may be followed by a fixed number of parameters. Parameters can be catalog entry names and words. A name is a textstring of 12 ISO characters beginning with a small letter followed by a maximum of 10 small letters or digits terminated by NULL characters. The possible commands are:

<:newcat:> has no effect

<:oldcat:> has no effect

<:end:> terminates binin.

<:create:>,<name>,<entry tail of 10 words>

Creates a temporary catalog entry with the name and contents as specified. If the first word of <entry tail> is positive, an area of that size is reserved on the backing store. If the entry already exists, it is first removed.

<:change:>,<name>,<entry tail of 10 words>

Changes an existing catalog entry with a given name as specified. If the entry decribes an area on the backing store, the number of segments is reduced to the value specified by the first word of entry tail.

<:rename:>,<name>,<newname>

The catalog entry, <name> is renamed to <newname>.

<:remove:>,<name>

Removes the catalog entry specified; if the entry decribes a backing store area, this is removed too.

<:perman:>,<name>,<catalog key>

Makes the catalog entry specified permanent with the catalog key <catalog key>. If <catalog key> equals 3, then the entry base is changed to the user base i.e. the entry becomes user scope.

<:load:>,<name>,<no of binout segments>

Loads a number of binout segments following the present command segment to the file decribed by <name>. On magnetic tape each binout segment is output as one block. On backing store the boundaries of backing store segments are ignored. The sum characters are not transferred to the output file.

3.4.6 Storage Requirements 3.4.6

The core storage requirement for "binin" is approx. 4096 bytes plus the space needed by FP.

***binin param <erroneous parameters>
Parameter error in call of "binin". The program proceeds, ignoring the erroneous parameters.

***binin <binout file> exhausted
The last character of <binout file> is not a sum character, when <binout file> is the last input file.

***binin input name missing
The parameter list does not include a <binout file> or <end of parameter list> is found before a normal termination of "binin".

***binin <binout file> input impossible
<binout file> is unknown or the input process can not be initialized.

***binin <output file> output impossible
<output file> cannot be reserved or is unknown.

***binin <binout file> core size
No core space for buffers etc.

***binin <binout file> sizeerror
A command segment from <binout file> occupies more than 256 words in core store.

***binin <binout file> sumerror in command segment

***binin <binout file> sumerror in load <output file>

***binin <text string> syntaxerror
The <textstring> is not recognized as a command.

***binin <binout file> create <name> result <result>
Create entry, result <> 0 (monitor function).

***binin <binout file> remove <name> result <result>
Remove entry, result <> 0 (monitor function).

***binin <binout file, change <name> result <result>
Change entry, result <> 0 (monitor function).

***binin <binout file> rename <name> reslut <result>
Rename entry, result <> 0 (monitor function).

***binin <binout file> perman <name> result <result>
Permanent entry, result <> 0 (monitor function).

If an error is detected "binin" continues with the next parameter in the list.

Further examples:

binin tro tro
inputs two paper tapes; command segments are required in the input.
The tapes may e.g. be proceduced by the FP commands:
tpo=binout fpnames.p move.b
tpo=binout algolprog

binin tro.c
The binout paper tape is checked, but no catalog functions are called, and no output is produced.

copyarea=binin tro.s
tpo=binout copyarea.ne.a
In this way it is possible to copy binout tapes. Another copy is made by new call of "binout", without reading the tape again.

code3=binin bincode1.2 bincode2.1.2 bincode1.4.3
loads segments 1,2 from bincode1, segment 3 from bincode2 and segments 4,5,6,7 from bincode1 thus merging two binouts of slang programs into code 3.
Possible command segments are regarded as load segments, because <other output> is specified.
The areas bincode1 and bincode2 may e.g. be produced by the FP commands:
bincode1=binout code1.s.ne
bincode2=binout code2.s.ne

3.5 binout

The program can output catalog entries and contents of files in a format (a binout file) which may be input by the program "binin" or the program "initialize catalog" (cf. ref. 3). The program can furthermore output autoload tapes.

3.5.1 Example:

The program file named 'fup' is output on paper tape by the FP command

```
tpo=binout fup
```

(compare with the example under the program "binin").

3.5.2 Call

$$\text{<out file>} = \text{binout} \left\{ \text{<s> <input description>} \right\}_{1}^{\infty}$$

$$\text{<input description>} ::= \text{<name>} \left\{ \begin{array}{l} \\ \text{.p} \\ \text{.b} \left\{ \text{.<bytes>} \right\}_{1}^{0} \\ \text{.s .<field>} \\ \text{.a .<field>} \\ \text{.np} \\ \text{.ne} \\ \\ \end{array} \right\}_{0}^{\infty}$$

$$\text{<bytes>} ::= \text{<no of bytes>}$$

$$\text{<field>} ::= \left\{ \begin{array}{l} \text{<no of blocks>} \\ \text{<no of blocks> . <first block>} \end{array} \right\}$$

The elements <no of bytes>, <no of blocks>, and <first block> are integers. The elements .p, .b.<bytes>, .s.<field>, .a.<field>, .np, and .ne are in the following called modifiers.

3.5.3 Function

The output from "binout" is a binout file consisting of binout segments.

The binout file is a stream of 8-bit characters on magnetic tape, in a backing store area, or on paper tape. Each binout segment is terminated by a special character, called the sum character.

Normally each <input description> causes the output of a number of binout segments. The first of these consists of the catalog entry defined by <name>, and determines the number of the remaining binout segments. This first binout segment is called a command segment. If the <input description> defines a program file, the command segment is followed by a number of binout segments, being the contents of this file. The latter segments are called load segments.

Depending on the modifiers of the input description, either the command segment or the load segments may be omitted, and it is also possible to output text files as load segments. The output from "binout" is normally used as follows:

1) As input to "binin".

2) As input to "initialize catalog", as described in ref. 2, chapter 14-15. In this case, the output from "binout" must be a paper tape or a magnetic tape file, including the command segments.

3) As an "autoload program". The output must then be a paper tape without command segments.

3.5.4 binout

The output file is defined by:

<out file>, which must be the name of a catalog entry describing a paper tape punch, a backing store area, or a file on magnetic tape. If the output is paper tape, "binout" will select the output mode to odd parity, independent of the mode defined by the file descriptor.

3.5.5 Input Description

The <input description> is a name, which may be followed by a set of modifiers; it defines the binout segments to be output:

<name> is a name of an arbitrary catalog entry. If the <input description> consists of the name only, the corresponding catalog entry determines the format of the output: The command segment is output but load segments are only output, if <name> describes a file containing a program. A file on magnetic tape, and a backing store area, which is organized as logical blocks, is output as a number of load segments, each load segment being a block of the file. Other program files are output as a single load segment.

The format of the output may also be chosen explicitly, by means of the modifiers. The effects of these modifiers are as follows:

P Intended for output of text files. The <name> must describe a file on magnetic tape or a backing store area. The contents of this file is output as a single load segment.

b.<bytes> Intended for output of slang programs. Has the same effect as p, except that only the first <bytes> bytes of the actual file are output. If <bytes> is not present, the last word of the filedescriptor associated with <name> determines the number of bytes. This number

may be set by "slang", just after translating a program.

s.<field> Intended for output of "slang" programs below requirements. The <name> must describe a file on magnetic tape or a backing store area, which is assumed to be organized as logical blocks (i.e. the first word of each block defines the length of the entire block; a block with a non-positive length terminates the area). The contents of the file are output as <no of blocks> load segments, and if <first block> is present, the first <first block> blocks of the file are skipped. In this case the modifier .ne is normally used too. If the <field> specification is empty, all blocks of the file are output.

a.<field> Intended for output of autoload programs.
Has the same effect as s.<field>, except that the first word of each block is not output.

np No program, i.e., no load segments are output. Normally not used.

ne No entry, i.e., the command segment is not output. Used e.g. for output of files which may later be loaded to defined areas (fuss=binin tro).

Note, that in a sequence of modifiers, only the latest of the modifiers:

p, b.<bytes>, s.<field>, a.<field>, and np

has effect; e.g. the <input description>:

jza.s.ne.a.1.3.p

has the same effect as the <input description>:

jza.ne.p

3.5.6 Binout Segment 3.5.6

A binout segment is a stream of 8-bit characters with odd parity, the left-most bit of each character being the parity bit. The

last character in the segment is a sumcharacter, which is characterized by the second bit being one. The right-most 6 bits of this character form the sum modulo 64 of all other characters in the segment.

Each byte of the input is output as two characters. The second bit of these is always 0, whereas the right-most 6 bits are a copy of the corresponding 6-bit group of the byte.

3.5.7 Command Segment

The contents of a command segment are a number of commands, sufficient to create a catalog entry and load the load segments in a later call of "binin" or "initialize catalog". The command segment, as output by "binout", consists of at most 3 commands, which are the output of the following words:

```
<:create:>                ; 2 words, text string
<name of entry>           ; 4 words, text string
<entry tail>              ; 10 words
<:perman:>                ; 2 words, text string
<name of entry>           ; 4 words, text string
<catalog key>             ; 1 word, integer
<:load:>                  ; 2 words, text string
<name of entry>           ; 4 words, text string
<no of load segments>     ; 1 word, integer
```

The <:perman:> command is omitted if the catalog entry has catalog key 0; and the <:load:> command is only included if load segments are output.

3.5.8 Storage Requirements

The core storage requirement for "binout" is approx. 3072 bytes plus the space needed for FP.

***binout <name> output impossible
No left side in the call, or the output device defined by <name> is reserved or does not exist, or <name> does not describe a binary file.

***binout <name> <list of erroneous parameters>
Parameter error in call of "binout". If the parameters are part of an input description, this is ignored.

***binout input name missing
End of parameter list is found before an expected input description.

***binout <name> unknown
<name> is not name of catalog entry.

***binout <name> input impossible
<name> describes an input device from which input is not possible, or <name> is unknown.

***binout core size
The core store space needed for buffers etc. is too small.

***binout <name> prog or entry
The input description demands output of load segments in spite of that <name> does not describe a file, or the input description causes no output.

***binout <name> segments <integer>
The input description demands more output than possible; only <integer> load segments from the file described by <name> are output.

If an error is detected "binout" continues with the next parameter in the list.

3.5.10 Examples on the Use of the Modifiers

The contents of the areas textarea and codearea containing a text and a program file respectively (e.g. produced by "edit" and "slang") are output on a paper tape by the FP command

```
tpo=binout textarea.p codearea.b
```

Only the part of codearea which contains code is output.
The tape may be input later by the FP command:

```
binin tro
```

A binary paper tape may be copied by the FP commands

```
copyarea=binin tro.s
tpo=binout copyarea.ne.a
```

(cf. the description of "binin").

The ALGOL compiler may be moved to magnetic tape - say mt471100, file 1 (this may be useful if the backing storage is very small). If ALGOL is present on the backing storage, this is done by the FP commands:

```
tapealgol=entry mto mt471100 0 1 0 algol algol
auxarea=binout algol.ne.s.12      ; as algol has 12 logical segments
tapealgol=binin auxarea
```

Now the areas algol and auxarea may be cleared and tapealgol renamed to algol and permanented in the catalog. (The tape reel may now be dismounted and will be requested whenever ALGOL is called.) The ALGOL STANDARD PROCEDURES are of course not moved.

3.6 bossjob 3.6

Bossjob sends a newjob message to BOSS (the internal process named BOSS) demanding the specified file enrolled as job file in an off line job. In this way a job running under another operating system may create a BOSS job. The actual job continues with the next FP command. Further details are found in sec. 1.3, newjob and replacejob in ref. [10].

3.6.1 Call 3.6.1

bossjob <s> <file name> $\left\{ \begin{matrix} \text{<name of remote batch printer>} \\ \end{matrix} \right\}_{0}^{1}$

where <file name> is a name of a permanent job file.

<name of remote batch printer>::=<name of max 6 char>

3.6.2 Function 3.6.2

A newjob message containing the specified name(s) is sent to BOSS.

3.6.3 Storage Requirements 3.6.3

1536 bytes plus space for FP.

3.6.4 Error Messages 3.6.4

***bossjob call

Left hand side in the call of the program.

***bossjob <parameter list> parameter error

Parameter error in the call of the program.

***bossjob <filename> <error cause>

Error during creation of the new job. The cause may be any of the following:

job queue full
job file not permanent
job file unknown
job file unreadable
user index too large
illegal identification
user index conflict
job file too long
temp claim exceeded
option unknown
param error at job
syntax error at job
line too long
attention status at remote batch terminal
device unknown
device not printer
parent device disconnected
remote batch malfunction

In case of any error no new job is created.

3.7 catsort se endringer 3.7

Lists on current output selected parts of the main catalog (or any auxiliary catalog) sorted according to parameters. At last also total number of entries and segments output are listed.

3.7.1 Example 3.7.1

The FP command:

catsort base.project.min

will output all files with a base contained in the project base, e.g. belonging to the actual project. The parameter min causes that only name, segments docname, date, and scope is output.

The FP command:

catsort

will output all non-system entries in the main catalog sorted according to base and entry name.

See also Further Examples.

3.7.2 Call 3.7.2

$$\left\{\text{<outfile>=}\right\}_{0}^{1} \quad \text{catsort} \quad \left\{\begin{array}{l}\text{<catalog spec>}\\ \text{<limit spec>}\\ \text{<sorting spec>}\end{array}\right\}_{0}^{\infty}$$

$$\text{<catalog spec> ::=} \quad \begin{array}{l}\text{maincat.}\left\{\begin{array}{l}\text{yes}\\ \text{no}\end{array}\right\}\\ \text{subcat}\left\{\begin{array}{l}\text{yes}\\ \text{no}\\ \text{<integer>}\\ \text{<document name>}\end{array}\right\}\end{array}$$

$$\text{<limit spec> ::=} \left\{\begin{array}{l}\text{system.}\left\{\begin{array}{l}\text{only}\\ \text{yes}\\ \text{no}\end{array}\right\}\\ \text{name.<entry name>}\\ \text{docname.<document name>}\\ \text{base.}\left\{\begin{array}{l}\text{<scope>}\left\{\text{.min}\right\}\\ \text{<baselow>.<baseup>}\end{array}\right\}\end{array}\right\}$$

<sorting spec> ::= [basesort | docsort | slicesort | nosort] . [yes | no]

<scope> ::= [project | user | login | temp]

[<baselow> | <baseup>] ::= <integer>

3.7.3 Format of Output

Each entry is output on one line in the form:

<entryname> <first slice> <name key> <catalog key> <lower entry base> <upper entry base> <mode.kind or segments> <kit/doc name> <remaining entry tail>

If the parameter min is specified the output is:

<entryname> <segments> <kit/docname>

3.7.4 Function

If an outfile is specified, this file is used for output, otherwise current output file is used.

The catalogs are one by one copied into a working bs file, which is sorted according to the parameters.

The sorting parameters are:

basesort.yes	meaning sorted after the entry base (which means grouped after project and users).
docsort.yes	meaning that each area entry is followed by all subentries (which have a kit/document name equal to the entry name of the main entry).
slicesort.yes	meaning sorted according to first slice. This parameter will cancel the parameter docsort.yes and has the same priority.

The priority of the sorting parameters are basesort, docsort. The last sorting criterion will always be alfabetic sorting on entry name.

nosort.yes	meaning that no sorting at all is performed. The total catalog will be output, neglecting all other parameters but maincat and subcat.

Other parameters:

maincat	defining whether the maincatalog is output.
subcat	defining whether the subcatalogs are output (if any), an integer or a documentname specifies a subcatalog to be output (0 corresponds to maincat). Will prevent the maincat from being output, unless explicitly specified by maincat.yes or subcat.0.
system	defining whether the system files are output.
name.<entry name>:	only entries with the same <entry name> are output. Only 1 name parameter is allowed.
docname.<document name>:	only entries containing the kit/doc name <document name> are output. Only 1 docname parameter is allowed.

base.<scope>	only entries with the specified scope are output.
base.<baselow>. <baseup>:	only entries contained in the specified base are output. A negative value of <baselow> or <baseup> must be given as the positive complement e.g. the integer -1000 is specified as 16777216-1000=16776216.

The parameters are initialized as follows:

maincat.yes
subcat.no
system.no
basesort.yes
docsort.no
slicesort.no

3.7.5 Error Messages 3.7.5

***catsort error param <erroneous and following parameters>
Parameter error in the call.

***catsort, create sortarea impossible
It was impossible to create an area for sorting.

In case of any error message, the program terminates.

3.7.6 Further Examples 3.7.6

catsort nosort.yes

will output the total main catalog in unsorted form.

catsort subcat.yes system.yes

will output all entries in the subcatalogs sorted according to base and entry name.

catsort name.pip docname.pap basesort.no

will output all non-system entries in the main catalog with entry name pip or document name pap, sorted according to entry name.

catsort docname.disc system.yes

will output all entries in the main catalog with document name disc, sorted according to base and entry name.

catsort subcat.disc1 system.yes

will output all entries in the subcatalog with document name disc1 sorted according to base and entry name.

3.8 change

Sends a change paper message to the parent (the operating system). The program is only used when a job executed under BOSS uses job controlled printer (cf. ref. [10], section 6.2).

Output on printer from a job running under BOSS is normally made either by printing on current output or as off line printing initiated by the FP command "convert".

3.8.1 Call

change <s> <device name> <s> <paper type>

where the parameter <paper type> is an integer.

3.8.2 Function

A change message containing the specified device name and paper type is sent to the parent who is then expected to perform the necessary actions (message to the operator etc.).

3.8.3 Storage Requirements

1536 bytes plus space for FP.

3.8.4 Error Messages

***change call

The program was called with a left hand side.

***change <parameter list> parameter error

Parameter error in the call of the program.

***change <parameter list> <error cause>

The change message was not accepted by BOSS for one of the following causes:

1. no buffers
2. job printer not allowed (cf. the BOSS User>s Manual).

In case of any error the change action is not performed by BOSS.

3.9 changeentry 3.9

Changes an existing catalog entry according to the parameters in the call. The program is a supplement to the program "set" and "entry" and is used when one wants to change some of the elements in the entry tail by copying from the tails of other catalog entries.

3.9.1 Example 3.9.1

Suppose that the catalog entry named 'source' contains the name of a magnetic tape reel in the document name field. By the FP commands

filex=changeentry filex source filex filex filex filex filex

the entry filex is changed to contain the name of the tape reel.

A catalog entry named 'source' containing the name - say mt471100 - may be created by a call of "set":

source = set mto mt471100

3.9.2 Call 3.9.2

```
<result name> = changeentry { <s> <kind>  { <s> <kit/doc name>
                            { <s> <free param> { <s> <file> { <s> <block>
                            { <s> <contry> { <s> <length> } 1 } 1 } 1 } 1 } 1 } 1
                                           } 0 } 0 } 0 } 0 } 0 } 0
```

```
{ <kind>          }  ::=  { <integer>
{ <kit/doc name>  }       { <integer1> . <integer2>
                          { <name>
```

```
<free param>  ::=  { <word>
                   { <byte1>.<byte2>
                   { d.<isodate>.<clock>
```

```
<isodate>  ::=  { <yymmdd>
                { 0          0 is interpreted as now
```

```
<clock>  ::=  <hhmm>   may be omitted in case
                       no entry named d exists
```

<file> | <block> | <contry> | <length> ::= <word> | <byte1> . <byte2>

<word> | <byte1> | <byte2> ::= <integer> | <name>

3.9.3 Function 3.9.3

The left hand side is looked up. If it does not exist, the program terminates. Otherwise the parameters are interpreted as described below yielding the wanted entry tail. From this point the program continues exactly as program "set".

3.9.4 Parameters 3.9.4

Kind:

<integer>:	The value is placed in the tail.
<integer1> . <integer2>:	The value <integer1> shift 12 + <integer2> is placed in the tail.
<name>:	First the name is searched for in the table of modekind abbreviations and if found here the value found is used. If not found the modekind table (see Utility Programs, part 1, Appendix) it is searched for in the catalog and the kind of the entry found is used.

Kit/doc name:

<integer>:	The value is placed in the tail.
<integer1> . <integer2>:	The value <integer1> shift 12 + <integer2> is placed in the tail.
<name>:	If the kind just found is the modekind bs (2048 shift 12 + 4) the name itself is used in the tail. For all other kinds the name is looked up in the catalog and the kit/doc name in the tail of the entry found is used.

The other parameters:

A parameter of the form <byte1> gives separate specifications of the two 12-bit bytes in the word.

<integer>:	The value is placed in the tail as the word or byte in question.
<name>:	The name is looked up in the catalog and the value of the word or byte in question in the entry tail is used.

If the parameter list does not specify all of the tail, the rest of the tail is set to zero.

3.9.5 Storage Requirements 3.9.5

1536 bytes plus space for FP.

3.9.6 Error Messages 3.9.6

***changeentry call

No left side in call of the program.

***changeentry param <parameter>

Parameter error in call of the program.

***changeentry <name> unknown

Lefthand side or a parameter was searched in the catalog but not found.

***changeentry <result name> change kind impossible

A change of an area to a non-area entry or vica versa was attempted.

***changeentry <result name> change bs device impossible

A change of kit/doc name of an area was attempted.

***changeentry <result name> bs device unknown

The bs device specified was not found.

***changeentry <result name> no resources

The resources of the job did not allow the wanted creation or change of an entry.

***changeentry <result name> entry in use

The entry could not be changed because another job was using it.

If any message appears no entry is changed.

3.10 char

Outputs the specified character the specified number of times.

3.10.1 Example

The current output is divided in groups by the call

```
char nl.8
```

which produces 8 newlines on current output.

```
char ff nl
```

produces a top of form and a newline on current output.

3.10.2 Call

$$\left\{\text{<outfile> =}\right\}_{0}^{1} \quad \text{char} \quad \left\{\begin{array}{l}\text{<iso value>} \\ \text{<iso value>.<repeat factor>}\end{array}\right\}_{0}^{\infty}$$

<iso-value> ::= <integer>|nl|ff|em|sp

<repeat factor> ::= <integer>

3.10.3 Function

If no repeat factor is specified the character will be output one time else the character will be output as many times as specified by repeat factor.

The repeat factor may be changed by the program, e.g. ff.19 will be changed to nl.64. Other characters will be repeated max. 133 times.

If an outfile is specified this is used for output else current output is used.

3.10.4 Storage Requirements 3.10.4

1024 bytes plus space for FP.

3.10.5 Error Message 3.10.5

***char param <parameter>

Parameter error in the call. The program continues in the parameter list.

3.11 claim 3.11

Lists some claims of the process.

3.11.1 Examples 3.11.1

In an installation with the 5 discs:
disc, disc1, disc2, disc3 and dk546123
the call:

```
claim
```

may print:

```
area 6   buf 4   size 30000   first core 290632

disc: 42 segm/slice
  temp      966 segm   23 slices   225 entr
  login     546 segm   13 slices   185 entr
  perm        0 segm    0 slices   182 entr

disc3: 21 segm/slice
  temp       21 segm    1 slices
  login      21 segm    1 slices     8 entr
  perm       21 segm    1 slices     8 entr

disc1: 42 segm/slice
  temp      252 segm    6 slices
  login     252 segm    6 slices    16 entr
  perm      252 segm    6 slices    16 entr

disc2: 105 segm/slice
  temp      525 segm    5 slices
  login     525 segm    5 slices     4 entr
  perm      525 segm    5 slices     4 entr

dk546123: 2 segm/slice  no resources
```

The call:

```
  claim key.disc
```

would print:

```
area 6   buf 4   size 30000   first core 290632

disc: 42 segm/slice
  key 0    882 segm   21 slices   222 entr
  key 1    882 segm   21 slices   224 entr
  key 2    462 segm   11 slices   184 entr
  key 3      0 segm    0 slices   182 entr
```

and the call:

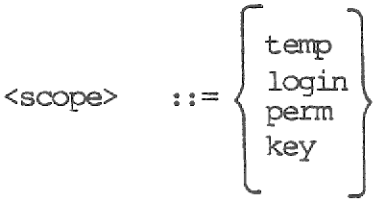
```
     claim perm.disc temp
```

would print:

```
area 6  buf 4  size 30000  first core 290632

disc: 42 segm/slice
  perm       0 segm    0 slices    182 entr

disc: 42 segm/slice
  temp     882 segm   21 slices    222 entr

disc3: 21 segm/slice
  temp      21 segm   21 slices

disc1: 42 segm/slice
  temp     252 segm    6 slices

disc2: 105 segm/slice
  temp     525 segm    5 slices

dk546123: 2 segm/slice
  temp       0 segm    0 slices

dk546104: 2 segm/slice
  temp       0 segm    0 slices
```

3.11.2 Call

<docname> ::= <name of drum or disc kit>



3.11.3 Function

The program scans the parameter list. For each parameter group, the internal tables in the monitor are scanned. If a document name is specified in the parameter group, the resources of catalog entries and segments for each permanent key on that device are listed, else the resources on all bs devices are listed. If <scope> is specified the listing of resources is restricted to the specified scopes.

perm is equal to scope user + project.

If <scope> is specified to key, the permanent keys will be output instead of the scope names.

Note that temp entries are only output for the main catalog, since all temporary entries are counted only here, cf. ref. 2 and 3.

If claim is called in the beginning of a job, the value of area is already reduced by 1, which is the one used by FP.

An empty parameter list means: all bs devices, all scopes.
If there is a left side in the call of claim, the output will appear on <output file> otherwise on current output.

3.11.4 Storage Requirements 3.11.4

900 bytes plus space for FP.

3.11.5 Error Messages 3.11.5

***claim connect <output file>

The specified output file could not be connected. Current output is chosen as output.

***claim param <list of erroneous parameters>

Parameter error in call of claim.

***claim <docname> unknown

A bs device named <docname> does not exist.

3.12 claimtest 3.12

Checks the claims of the calling process according to the call parameters and leaves the ok bit true if the claims specified are present, false otherwise.

3.12.1 Example 3.12.1

The job

```
claimtest perm.disc1.1000.10
if ok.no
finis
```

is terminated if the permanent resources on disc1 are less than 1000 segments and 10 entries.

3.12.2 Call 3.12.2

```
                  ⎧ <key>.<bs claims>   ⎫ ∞
                  ⎪ buf.<buffer claim>  ⎪
claimtest<s>      ⎨ area.<area claim>   ⎬
                  ⎪ size.<size>         ⎪
                  ⎩ int.<internal>      ⎭ 0
```

<key>::= perm/login/tempspec/temp

<bs claims>::= <document name>.<segments>.<entries>

<document name>::= <name of drum or disc>

```
⎧ <segments>   ⎫
⎪ <entries>    ⎪
⎪ <bs claims>  ⎪
⎨ <buffer claim> ⎬ ::= integer
⎪ <area claim> ⎪
⎪ <size>       ⎪
⎩ <internals>  ⎭
```

3.12.3 Function 3.12.3

The parameters in the call are examined one by one.

The <key> parameter names mean:

temp:	key = 0
temp spec:	key = 1
login:	key = 2
perm:	key = 3

If an error in the parameter occurs or the claims specified exeed the claims available according to the process description of the calling process, the program terminates setting the ok bit false. If the program reaches the end of the parameter list, the program terminates setting the ok bit true.

Temporary and temporary special entry claims are checked on the disc containing the main catalog no matter the document name specified.

3.12.4 Storage Requirements 3.12.4

7144 halfwords (4096 halfwords + space for FP).

3.12.5 Error Messages 3.12.5

***claimtest: parametererror, unknown fpparameter <parameter>

The parameter is neither of the seven names: perm, login, temp, buf, area, size, int.

***claimtest: parametererror, parameter must be {name / integer} read <param>

The parameter is a name or an integer when it should be an integer or a name.

***claimtest: syntaxerror, separator must be <point> read <sep>

Separator not a point.

***claimtest: unknown bs device <name>

The bs device with the name <name> is not included in the bs system.

3.13 clear 3.13

Removes catalog entries with name and scope as specified.

3.13.1 Example 3.13.1

By the FP command

clear user text4

the catalog entry (if any) with scope user and name text4 is removed from the catalog. A catalog entry with the same name but another scope is not affected.

3.13.2 Call 3.13.2

clear <s> <scope spec> $\left\{ \text{<s> <name>} \right\}_{1}^{\infty}$

<scope spec> ::= <scope> $\left\{ \text{.<device name>} \right\}_{0}^{1}$

<scope> ::= { temp | login | user | project }

<device name> ::= <name of drum or disc kit>

3.13.3 Function 3.13.3

The scope specification is interpreted and then the name list is scanned. For each name in the list the name is searched in the catalog. If an entry with the specified name and scope is found, it is removed from the catalog.

3.13.4 Scope Specifications 3.13.4

The concept of scope of a catalog is explained in ref. [10], sec. 4.1. A device name means a further restriction to entries which are either

(a) area entries, where the data area is placed on the specified bs device, or

(b) non-area entries, which are present in the auxiliary catalog on the device cf. ref. 3.

3.13.5 Storage Requirements 3.13.5

2048 bytes plus space for FP.

3.13.6 Error Messages 3.13.6

***clear call
The program was called with a left hand side. No entries removed.

***clear <scope spec> illegal scope
The scope specification was illegal. No entries removed.

***clear <scope spec> bs device unknown
The specified device was not on the computer. No entries removed.

***clear <scope spec> bs device not ready
The bs device specified was not ready or catalog i/o error. No entries removed.

***clear param <parameter>
Illegal parameter. The rest of the parameter list is skipped.

***clear <scope spec> <name> unknown

The entry to be removed was not found. The program continues with the next name in the parameter list.

***clear <scope spec> <name> entry in use

The entry could not be removed because another job was using it. The program continues with the next name in the parameter list.

3.14 clearmt 3.14

Removes catalog entries according to the parameters.

3.14.1 Example 3.14.1

The FP command:

```
pap=clearmt mt004711.3
```

will remove the entries pap1 pap2 pap3.

The FP command:

```
f=clearmt f.3.5
```

will remove the entries f3 f4 f5.

3.14.2 Call 3.14.2

<result name> = clearmt <mtname>.{ <upper integer> | <lower integer>.<upper integer> }

The <mtname> is not used during interpretation of the parameters. If no <lower integer> is specified, it is set to 1.

3.14.3 Function 3.14.3

Entry names <resultname> followed by <lower integer> to <upper integer> are removed.

3.14.4 Storage Requirements 3.14.4

512 bytes plus space for FP.

3.14.5 Error Messages

***clearmt call

No left hand side of more than 9 characters

***clearmt param

Parameter error in the call, e.g. <integer> greater than 99.

***clearmt <resultname> catalog error

Error in catalog, monitor, or hardware

In case of above error messages the program terminates.

***clearmt <resultname> unknown

The specified entry was not found. The program continues.

3.15 convert 3.15

Sends a convert message to the parent (the operating system) who is then expected to print the specified backing storage area. A file with scope login is not accepted and the file must not be in use (for instance the file must not be current output). A temporary file converted will immediately disappear from the reach of the job. Each convert operation performed by BOSS requires a cbuffer which must be reserved in the job specification (cf. ref. [10], ch. 3 and sec. 6.2).

3.15.1 Example 3.15.1

A program has produced a text file in the area out1. It is printed by the FP command

```
convert out1
```

3.15.2 Call 3.15.2

convert <s> <name> $\left\{ \text{<name of remote batch printer>} \right\}_{0}^{1}$ $\left\{ \text{<s> <integer>} \right\}_{0}^{1}$

<name of remote batch printer>::= <name of max. 6 char>

3.15.3 Function 3.15.3

The convert message with the specified name(s) and integer (or zero if no integer is specified) is sent to the parent).

3.15.4 Paper Types

0	Standard paper, i.e. monitor format, one copy. A page is 64 lines of 133 positions.
1	A4 upright, one copy. A page is 64 lines of 72 positions.
2	A4 across, one copy. A page is 42 lines of 112 positions.
3	Monitor, two copies.
4	A4 upright, two copies.
5	A4 across, two copies.
6	Monitor, three copies.
7	A4 upright, three copies.
8	A4 across, three copies.
9-99	For extensions.
100-999	Special forms. Requires agreement with the operator.

3.15.5 Storage Requirements

1536 bytes plus space for FP.

3.15.6 Error Messages

***convert call

Left hand side in call of the program.

***convert <parameter list> parameter error

Parameter error in call of the program.

***convert <parameter list> <error cause>

The convert message was not accepted by BOSS for one of the following causes:

1. no cbuffers
2. file does not exist
3. file has login scope
4. no resources
5. file in use
6. file is not area

7. attention status at remote batch terminal 3.15.4
8. device unknown
9. device not printer
10. parent device disconnected
11. remote batch malfunction
12. not textfile

In case of any error the convert operation is not performed by BOSS.

3.16 copy

Copies one of several text files into another file and calculates the number of characters copied and the sum of their ISO values. Blind characters are not copied. The program can be used instead of "edit" if only a simple copying is wanted. Furthermore the program may be used for check reading of text files (e.g. texts punched on paper tape).

3.16.1 Example

The text files 'text1' 'text2' are output as one paper tape file by the FP command

```
tpe=copy text1 text2
```

and the number and the sum of the characters are printed on current output. One may then check the tape by reading it in a later job by the FP command

```
copy tre
```

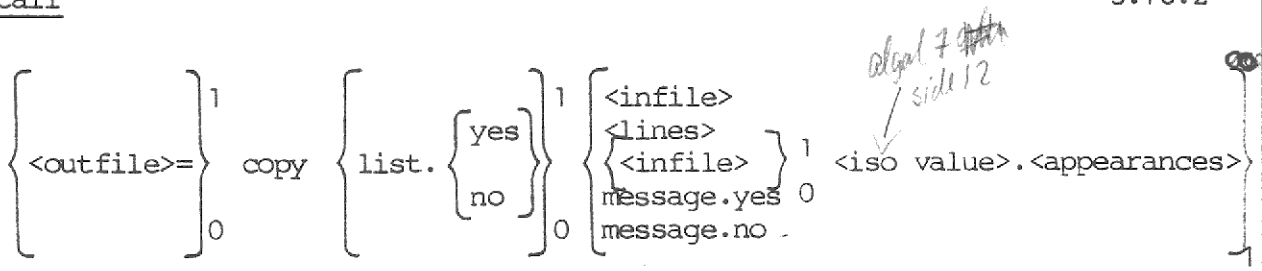
Under BOSS the tape should be input by a load command

```
load tre pip
```

in the job specification. The check reading is then performed in the job file by the FP command

```
copy pip
```

3.16.2 Call



<infile> ::= <name>

<lines> ::= <integer>

<iso value> ::= { <small letter> | <integer> }

<appearances> ::= <integer>

3.16.3 Function

If the parameter list.yes is specified, the input is listed on current out. The program interpretes one parameter at a time as follows:

<infile>
The file is copied on <outfile> if any. If no <outfile> is specified only the calculation of number and sum of characters is performed.

<lines>
This number of visible lines are copied from current input on <outfile> if any.

<iso value>.<appearances> and <infile>.<iso value>.<appearances>
The program copies from <infile> if specified, else from current input on <outfile> if any.
Copying stops when the specified number of appearances of the iso character are met. The last character is not output.

message.yes or message.no
Determines whether the following should be output on current output (standard: message.yes)

1. after each param:
   <infile> segm. <number of segments>
   number of characters < 128
   sum of characters
   number of characters $\geq$ 128 (if any)
   number of blind characters (0, 127) - (if any)
   number of sub characters (26) - (if any)

2. at program end (only if the call contains an <outfile> and more than one param):
   <outfile> segm. <number of segments>
   total number of characters < 128
   total sum of characters
   total number of characters $\geq$ 128 (if any)
   total number of sub characters (26) - (if any)

3.16.4 Storage Requirements 3.16.4

1536 bytes plus space for FP.

3.16.5 Error Messages 3.16.5

All errors cause the warning bit to be set.

***copy connect <outfile> <cause>
The output file cannot be connected for output. The ok bit is set to no and the program is terminated.
<cause> may be:

1. no resources
2. not found
3. in use - maybe file is the job file
4. convention error - output attempted on input device or vice versa
5. error - catalog, monitor, or hardware error.

***copy connect <infile> <cause>
An input file cannot be connected for input. The parameter is ignored.

***copy param <illegal parameter>
Illegal parameter syntax. The parameter is ignored.

***copy end medium
Current input is exhausted because the parameter <lines> or <iso value>.<appearances> demands reading past EM. The program continues with the next parameter.

***copy no core
The call is not executed because the process is too small.

3.17 corelock 3.17

Sends a corelock message to the parent (the operating system) demanding that the job should stay in core the specified number of seconds. This feature is only used in connection with process control devices producing data with a high rate, cf. ref. [10], secs. 3.4 and 6.7.

3.17.1 Example 3.17.1

The FP command:

```
corelock 5
```

demands corelock for a period of 5 seconds.

3.17.2 Call 3.17.2

```
corelock <s> <seconds>
```

where <seconds> is an integer.

3.17.3 Storage Requirements 3.17.3

1536 bytes plus room for FP.

3.17.4 Error Messages 3.17.4

***corelock call
Left hand side in the call of the program.

***corelock <parameter list> parameter error
Parameter error in the call of the program.

In case of any error no corelock message is sent.

3.18 coreopen 3.18

Sends a coreopen message to the parent (the operating system) signalling the end of a corelock period (cf. the program "corelock"). The program is only used on process control installations.

3.18.1 Example 3.18.1

The program is called without parameters:

```
coreopen
```

3.18.2 Call 3.18.2

```
coreopen
```

3.18.3 Storage Requirements 3.18.3

1536 bytes plus room for FP.

3.19 correct

The program corrects specified words on the backing storage according to the parameters. The program may also be used to print specified bits as integers.

3.19.1 Example

The FP call:

```
correct bsfile.4 addr.0 bits.0.11 if 700 then neg.456,
   bit.12.23 if neg.1234 then 4000,
   adr.8 if 0 then 1
```

will make the following corrections on segment 4 of bsfile:

```
byte 0   is changed to -456 (in case it is   700)
-    1   -  -       -  4000 (-  -    -  -  -1234)
-    8:9 -  -       -     1 (-  -    -  -      0)
```

No corrections are made if <oldvalue> is not correct in all cases.

3.19.2 Call

correct <bsfile>.<segmno>

{address.<addr> {<bitspec> if <oldvalue> then <newvalue>}$_{1}^{\infty}$}$_{0}^{\infty}$

<bitspec> ::= { empty (identical to bits.0.23) | bits.<firstbit>.<lastbit> }

{ <oldvalue> | <newvalue> } ::= { <integer> | negative.<integer> }

{ <segmno> | <addr> | <firstbit> | <lastbit> } ::= <integer>

3.19.3 Function

Segment number <segmno> is input and for each address it is tested whether the specified <oldvalue> is found, in which case it is replaced by <newvalue>. If no errors are found the segment is output.

Note that the file will be connected in the standard way for utilities, i.e. segmno is calculated as segmno + block count cf. Utility Programs, Part I, section 5.5.

During syntax check only the first 3 letters in the words:

address, bits, then, negative

are tested. adr is accepted for address.

Odd addresses are reduced by 1.

<segmno> and <addr> are counted from 0.

Shortblock in catalog entry is updated. In case <bsfile> describes an external procedure, the internal date is updated.

3.19.4 Storage Requirements

726 bytes plus space for FP.

3.19.5 Error Messages

***correct call

Left hand side in the call.

***correct param <faulty parameter>

Syntax error in the call.

***correct param missing

End of parameter list when more parameters are expected.

***correct <bsfile> not connected

<bsfile> could not be connected, maybe not present or not kind bs.

***correct segm.<segmno>

<segmno> ≥ size of <bsfile>.

***correct addr.<addr>

<addr> > 511.

***correct addr.<addr> bits.<firstbit>.<lastbit>

<firstbit> > <lastbit> or <lastbit> > 23.

***correct addr.<addr> bits.<firstbit>.<lastbit> oldvalue=<oldvalue>

<oldvalue> is greater than the specified bits allow.

***correct addr.<addr> bits.<firstbit>.<lastbit newvalue=<newvalue>

<newvalue> is greater than the specified bits allow.

***correct addr.<addr> bits.<firstbit>.<lastbit>
oldvalue=<oldvalue> ,found=<oldvalue found>

The specified <oldvalue> is not equal to the found value.

In the last case the program continues in the parameter list (but no corrections will be made), in all other cases the program terminates immediately.

In case of any of above error messages no corrections are made.

***correct entry inconsistent

***correct code inconsistent

The date of an external procedure is incorrectly described either in the catalog entry or in the code. The correction has been performed.

3.20 edit 3.20

"edit" is a line oriented program for editing of text files.

3.20.1 Example 3.20.1

The FP call and edit commands:

```
                                        COMMENTS
    betterfinal=edit finaltext          fp call
    1./bad/,r/bad/good/,f               edit command
```

will produce in betterfinal a corrected version of the text finaltext.

The FP call:

```
    (i corrfile
    newtext=edit oldtext
    end)
```

will correct the text in oldtext with the edit commands in corrfile. The FP command end ensures that FP will not read from corrfile in case edit exists before the finis command.

BOSS User>s Manual shows several very relevant eaxmples of the use of "edit".

3.20.2 Call 3.20.2

$$\left\{\text{<outfile>=}\right\}_{0}^{1} \quad \text{edit} \quad \left\{\text{<source>}\right\}_{0}^{\infty}$$

3.20.3 Function 3.20.3

The program will edit the text in <source> by the commands in current input and store the resulting text in outfile.

<outfile> can be any kind of document. If no outfile is specified, no text is stored.

<source> if no source is specified this is interpreted as an empty source.

When "edit" is loaded and prepared for input of commands the message

edit begin

is printed, and before "edit" exists, it prints the message:

edit end.

3.20.4 Edit Commands 3.20.4

The editing is performed by means of the following commands. Only the first letter in the word is tested by "edit":

line

delete

insert

replace

global

finis

and less important

print

source

mark

verify

where

; <comments> (this line is skipped by "edit")

The commands are separated by NL or COMMA. Superfluous NLs are blind. SPs between commands are blind. Commands separated by COMMA form a sequence. At end of each single command or sequence, the line on which the line pointer points is printed (unless the command: v n is given), see "verify".

3.20.5 Delimiters 3.20.5

A special feature of edit is that the delimiter is chosen each time as the first symbol following the command letter(s), e.g.

```
1
/
*
,
:
2
p
```

In the last case p was the first letter in the following word. Illegal symbols, SP, NL, and EM cannot be used as delimiters.

The delimiter must not be a part of the string to be searched, the string to be removed, or the replacing or inserted string.

In all following examples only the delimiter / is shown.

3.20.6 Warning æ ø å 3.20.6

Those letters have a special meaning and cannot be used in the strings unless the following command is given:

	COMMENTS
m e	mark empty

see "mark".

3.20.7 Line 3.20.7

All corrections are made in the aktuelle current line, so first of all this must be found. At start the line pointer points at the first line.

	COMMENTS
17	Move line pointer 7 lines forwards.
1-4	Move line pointer 4 lines backwards.
1 t	Line top, move line pointer to line 1.
1 b	Move line pointer to line bottom, i.e. the line containing EM.
1./find/	The line pointer is moved forward to point at the first line containing the string find.

Empty lines are not counted. They have the same number as the following line. This is the case for all commands. The line pointer points at the first of the empty lines.

In case the search string consists of several lines, the line pointer will point at the last line. A NL must not be specified by æ10æ as NL has a special representation. This is also the case for the commands delete and print.

3.20.8 Delete 3.20.8

	COMMENTS
d	Delete current line.
d 4	Delete current and 4 following lines.
d-2	Delete current and 2 preceding lines.
d t	Delete current and all in front.
d b	Delete current and all following.
d./find/	Delete current and including the line containing the textstring find.

After deletion the line pointer points at the line following the last deleted line. Re. NL see Line.

3.20.9 Insert

	COMMENTS
i/ elefants monkeys /	The two lines are inserted in front of the current line. After insert the line pointer points at the line in front of the terminating delimiter. (here in the line monkey).

Note that it is a syntax error if the first delimiter is not followed by NL. SPs between the first delimiter and the NL are blind.

3.20.10 Replace

	COMMENTS
r/bad/good/	In the current line the first string bad is replaced by the string good.
r/something//	Remove the first string something from the line.
r//something/	Before anything else on the line place the string something.

The string which is to be replaced, must be within one line, i.e. a NL character can only be used in connection with empty lines. A NL character must not be specified by æl0æ. The replacing string can be of any number of lines. The line pointer points at the last line in the replacing string.

If position not found, the line pointer points at the next line.

COMMENTS

g/bad/good/

In the current line any bad is replaced by good. The line pointer is unchanged.
If, however, the replacing string is two or more lines, the line pointer is changed. It will point at the last line before the terminating delimiter, as for 'insert'. In this case only the first occurrence of the string is replaced, since later occurrences are no longer in current line.

g2/bad/good/

In the current and 2 following lines any bad is replaced by good The line pointer is moved 2 lines forward.
If, however, the replacing string is two or more lines it must be noticed that only occurrences in the original lines are replaced, and only as long as the line pointer does not exceed the initial value of current line + the number of lines specified, which is where the line pointer will end.

g-6/bad/good/

In current and 6 preceding lines any bad is replaced by good. The line pointer is not moved.

The effect is the same as 1-6, g 6/bad/good/

If the replacing string is two or more lines, this is important, because it means that the interval for the line pointer is frozen to initial value of current line up to initial value + number of lines specified, which is where the line pointer will end.
The contents of that line interval is changed for each replacement, moving some of the original lines beyond the interval limit.

g t/bad/good/	In current and all preceding lines any bad is replaced by good. The line pointer is not moved. If, however, the replacing string is two or more lines it must be remembered that the effect is the same as: 1 t, g<current line>/bad/good/
g b/bad/good/	In current and all following lines any bad is replaced by good. The line pointer points at the line following the last line. The effect is the same for replacement strings or more lines.
g b/unwanted//	Remove the string unwanted from current line and until bottom.

Re. NL see Replace.

3.20.12 Finis

3.20.12

f	COMMENTS "edit" copies to EM and exits.

3.20.11 Global

se endring

	COMMENTS
g/bad/good/	In the current line any bad is replaced by good. The line pointer is unchanged.
g 2/bad/good/	In the current and 2 following lines any bad is replaced by good. The line pointer is moved 2 lines forward.
g-6/bad/good/	In current and 6 preceding lines any bad is replaced by good. The line pointer is not moved.
g t/bad/good/	In current and all preceding lines any bad is replaced by good. The line pointer is not moved.
g b/bad/good/	In current and all following lines any bad is replaced by good. The line pointer points at the line following the last line.
g b/unwanted//	Remove the string unwanted from current line and until bottom.

Re. NL see Replace.

3.20.12 Finis

	COMMENTS
f	"edit" copies to EM and exits.

/

end medium

3.20.13 Print

	COMMENT
p	Prints current line.
p2	Current and 2 following lines are printed.
p-2	Current and 2 preceding lines are printed with normal direction.
p t	All lines in front of and including current line are printed with normal direction.
p b	Current and all following lines until EM are printed.
p./find/	The current and all lines inclusive the line with the string find are printed.

The line pointer points at the last printed line.
Re. NL see Line.

3.20.14 Source

går uden om filen

The sources are the parameters to the call of edit and are numbered from 1.

	COMMENTS
s 2	Edit with input from source number 2.

eks s, f (du har lavet fejl i edit kal f. s, f edit en f)

g t/bad/good/	In current and all preceding lines any bad is replaced by good. The line pointer is not moved. If, however, the replacing string is two or more lines it must be remembered that the effect is the same as: 1 t, g<current line>/bad/good/
g b/bad/good/	In current and all following lines any bad is replaced by good. The line pointer points at the line following the last line. The effect is the same for replacement strings or more lines.
g b/unwanted//	Remove the string unwanted from current line and until bottom.

Re. NL see Replace.

3.20.12 Finis 3.20.12

f	COMMENTS "edit" copies to EM and exits.

3.20.13 Print

	COMMENT
p	Prints current line.
p2	Current and 2 following lines are printed.
p-2	Current and 2 preceding lines are printed with normal direction.
p t	All lines in front of and including current line are printed with normal direction.
p b	Current and all following lines until EM are printed.
p./find/	The current and all lines inclusive the line with the string find are printed.

The line pointer points at the last printed line.
Re. NL see Line.

3.20.14 Source

The sources are the parameters to the call of edit and are numbered from 1.

	COMMENTS
s 2	Edit with input from source number 2.

Example: the programmer wants to produce a textfile new which is text1 with the procedure error from text2 placed between procedure testoutput and procedure calculate and to link text3 to text1:

```
text1:                                  text2:
begin real a, b,c,d;                    begin integer i,j,k;
procedure testoutput;                   boolean ok;
begin                                   procedure error(i);
write(out,<:<10>:>,a,b,c);              integer i;
end testoutput;                         begin
procedure calculate(x);                 write(out,<:<10>alarm :>,i;
real x;                                 end error;
begin                                   procedure merge(a,b,x);
...                                     ...
```

```
                                        COMMENTS
new=edit text1 text2 text3              Edit call with 3 sources.
1./ure calculate/,                      Copy until this line from source 1.
s 2                                     Continue from source 2.
d./boolean ok/,                         Delete inclusive this line.
1./end error/,11,                       Copy until this line.
s1                                      Continue from source 1.
d./end testoutput/,                     Delete inclusive this line.
1 b                                     Copy to last line.
s 3                                     Continue from source 3.
f                                       Copy and exit.
```

"edit" is initialized to:	COMMENTS
m s	Mark standard, which is equivalent to the 3 following commands.
m n æ	Mark numeric æ The character æ is here chosen to be used to specify a character by its numeric code, i.e an integer between 0 and 127, e.g. æ12æ.
m c ø	Mark character ø The character ø is here chosen to be used as character replace mark, see example.
m l å	Mark line å The character å is here chosen to be used as line erase mark, i.e. the total line containing the letter å is erased.

If those 3 characters should be treated like other letters, use the following command

m e	Mark empty

Any other characters may be chosen as mark numeric, mark character or mark line, e.g.

m n z	Mark numeric z

The selected characters should not be used in any other context in the edit commands.

3.20.15.1 Examples of Use of Mark Characters

	COMMENTS
r/formfeed/æl2æ/	Replace the text formfeed by the character formfeed.
r/a**b/a//ba	Erase faulty line (on a console or terminal % should be used as this. Causes the monitor to erase the line. Under BOSS % only works until timeout, later BEL can be used).
r/abgde<BS><BS><BS>ø<BS>c<SP><SP>fg/alphabet/	Result: r/abcdefg/alphabet/ only to be used when typed on devices which has a backspace BS character. Used for correction of one character without changing the rest of the line.

3.20.16 Verify

Normally the line is listed at the end of each command sequence. This may be omitted by the "edit" command.

	COMMENTS
v n	Verify no

and reset by

v y	Verify yes

3.20.17 Where

The "edit" command

```
w
```

prints a number of the current source text line, e.g.

```
3 line.
```

3.20.18 Matching Strings

The characters SP, NL, and non-graphic characters are blind for identification, i.e. they are skipped by the matching procedure when met in the source string.

In case those characters are part of the search string, they will take part in the matching.

Two strings are considered identical, if the aource text has as a minimum the same number of SP and NL (and other blind characters) as the search string, e.g.

```
r/a b/ak/
```

will accept

```
a b
a        b
```

but not

```
ab
```

3.20.19 Parity Errors

When a parity error is met in the source text, the message

```
parity error on <source>
```

is typed on current out, and "edit" continues and copies the character 26. During verification and printing of a line, the character will be printed as the character 38 (ampersand).

The character may be changed as any other symbol, by using the numerical value of the character, e.g.

	COMMENTS
1./æ26æ/,r/æ26æ/g/,f	The faulty character is replaced by g.

3.20.20 Error Messages

3.20.20.1 Initial Alarm

***edit end. no core
The current process is too small.

***edit end: param
The parameters to the call of edit are not syntactically correct.

***edit end: connect object
The object document cannot be connected by the file processor. If an output area should be created the alarm may indicate that there is no room on backing store.

***edit end: work area
There is no room on the backing store for the work area needed for intermediate storage of commands.

3.20.20.2 Alarms Concerning Communication with Peripheral Devices

***edit <command no.> connect source
The source document cannot be connected by the file processor. Note: when the source command is used to select a source outside the source given in the parameter list, the source document is defined as empty.

***edit <command no.> source unknown
The source is not found.

***edit <command no.> work area
Not enough backing storage for output.

***edit <command no.> character
A character with a code greater than 127 has been input either from the source document or the command document.

***edit <command no.> correction area
Not enough backing storage for a long correction.

<command no> is reset to 1 at start of each sequence.

Other errors in connection with the transfer of characters and blocks are handled by the file processor and treated as hard errors.

3.20.20.3 Alarms Caused by Erroneous Commands

***edit <command no.> syntax
A syntax error in the command format is found.

***edit <command no.> position not found
A line position cannot be found, or no match with the string in a replace command can be obtained. The string looked for is printed.

***edit <command no.> backspace error
If random access to the text is not allowed, i.e. when the outfile is not backing storage, backspacing is only allowed a limited number of lines. The alarm is given when backspacing is attempted beyond this number of lines, which among other things is dependent on process size.

<command no.> is reset to 1 at start of each sequence.

3.21 end

Returns current input to the previous current input at the position where it was left.

3.21.1 Call

end

3.21.2 Function

The function is the same as when an EM character is read by FP from current input. The actual input is unstacked, and FP continues reading from the previous current input.

3.21.3 Storage Requirements

1024 bytes plus space for FP.

3.21.4 Error Messages

***end call
Left hand side in the call. The end is still performed.

***end param <parameter>
Wrong parameter in the call. The end action is still performed.

3.22 entry 3.22

Creates or changes a temporary catalog entry according to the parameter in the call. The program is a supplement to the program "set" and is used when one wants to set some of the elements in the tail by copying from the tails of other catalog entries.

3.22.1 Example 3.22.1

Suppose that the catalog entry named 'source' contains the name of a magnetic tape reel in the document name field. By the FP commands

```
file1=entry mt1h source 0 1
file2=entry mt1h source 0 2
file3=entry mt1h source 0 3
```

one gets catalog entries 'file1', 'file2', 'file3' which serve as file descriptors for file 1, 2 or 3 on the tape reel.

A catalog entry named 'source' containing the name - say mt471100 - may be created by a call of "set":

```
source = set mt1h mt471100
```

3.22.2 Call 3.22.2

```
                          {<s><kind> {<s><kit/doc name>{<s><free>
<result name>=entry       {<s><file> {<s><block>{<s><contry>
                          {<s><length>}          1}1}1}1}1}1}1
                                                 0}0}0}0}0}0}0
```

```
<kind>                 ::=  {<integer>
<kit/doc name>              {<integer1> . <integer2>}
                            {<name>
```

```
<free>                 ::=  {word
                            {byte1.byte2
                            {d.<isodate>.<clock>
```

<isodate> ::= { <yymmdd> | 0 } 0 is interpreted as now

<clock> ::= <hhmm> may be omitted in case no entry named d exists

{ <file> | <block | <contry> | <length> } ::= { <word> | <byte1> . <byte2> }

{ <word> | <byte1> | <byte2> } ::= { <integer> | <name> }

3.22.3 Function

3.22.3

The parameters are interpreted as described below yielding the wanted entry tail. From this point the program continues exactly as the program "set".

3.22.4 Parameters

3.22.4

Kind:

<integer>: The value is placed in the tail.

<integer1> . <integer2>: The value <integer1> shift 12 + <integer2> is placed in the tail.

<name>: First the name is searched for in the table of modekind abbreviations and if found here the value found is used. If not found in the modekind table (cf. ref. [8], Appendix) it is searched for in the catalog and the kind of the entry found is used.

Kit/doc name:

<integer>: The value is placed in the tail.

<integer1> . <integer2>: The value <integer1> shift 12 + <integer2> is placed in the tail.

<name>: If the kind just found is the modekind bs (2048 shift 12 + 4) the name itself is used in the tail.
For all other kinds the name is looked up in the catalog and the kit/doc name in the tail of the entry found is used.

The other parameters:
A parameter of the form <byte1> . <byte2> gives separate specifications of the two 12-bit bytes in the word.

<integer>: The value in the tail as the word or byte in question.

<name>: The name is looked up in the catalog and the value of the word or byte in question in the entry tail found is used.

If the parameter list does not specify all of the tail, the rest of the tail is set to zero.

Note: If <free> = d.<isodate> and an entry named d exists, the left half of <free> will be taken from d and the right half will become <isodate>.

3.22.5 Storage Requirements 3.22.5

1536 bytes plus space for FP.

3.22.6 Error Messages 3.22.6

***entry call
No left side in call of the program.

3.21 end se endring 3.21

Returns current input to the previous current input at the position where it was left.

3.21.1 Call 3.21.1

end

3.21.2 Function 3.21.2

The function is the same as when an EM character is read by FP from current input. The actual input is unstacked, and FP continues reading from the previous current input.

3.21.3 Storage Requirements 3.21.3

1024 bytes plus space for FP.

3.21.4 Error Messages 3.21.4

***end call
Left hand side in the call. The end is still performed.

***end param <parameter>
Wrong parameter in the call. The end action is still performed.

3.22 entry 3.22

Creates or changes a temporary catalog entry according to the parameters in the call. The program is a supplement to the program "set" and is used when one wants to set some of the elements in the tail by copying from the tails of other catalog entries.

3.22.1 Example 3.22.1

Suppose that the catalog entry named 'source' contains the name of a magnetic tape reel in the document name field. By the FP commands

```
file1=entry mto source 0 1
file2=entry mto source 0 2
file3=entry mto source 0 3
```

one gets catalog entries 'file1', 'file2', 'file3' which serve as file descriptors for file 1, 2, or 3 on the tape reel.

A catalog entry named 'source' containing the name - say mt471100 - may be created by a call of "set":

```
source = set mto mt471100
```

3.22.2 Call 3.22.2

```
                                  { <s> <kind> { <s> <kit/doc name> { <s> <free>
<result name> = entry             { <s> <file> { <s> <block>        { <s> <contry>
                                  { <s> <length> } 1 } 1 } 1 } 1 } 1 } 1 } 1
                                                 0   0   0   0   0   0   0

<kind>
<kit/doc name>  ::= { <integer>
                    { <integer1> . <integer2>
                    { <name>
```

<isodate> ::= { <yymmdd> | 0 } 0 is interpreted as now

<clock> ::= <hhmm> may be omitted in case no entry named d exists

{ <file> | <block | <contry> | <length> } ::= { <word> | <byte1> . <byte2> }

{ <word> | <byte1> | <byte2> } ::= { <integer> | <name> }

3.22.3 Function 3.22.3

The parameters are interpreted as described below yielding the wanted entry tail. From this point the program continues exactly as the program "set".

3.22.4 Parameters 3.22.4

Kind:

<integer>: The value is placed in the tail.

<integer1> . <integer2>: The value <integer1> shift 12 + <integer2> is placed in the tail.

<name>: First the name is searched for in the table of modekind abbreviations and if found here the value found is used. If not found in the modekind table (cf. ref. [8], Appendix) it is searched for in the catalog and the kind of the entry found is used.

Kit/doc name:

<integer>: The value is placed in the tail.

<integer1> . <integer2>: The value <integer1> shift 12 + <integer2> is placed in the tail.

<name>: If the kind just found is the modekind bs (2048 shift 12 + 4) the name itself is used in the tail. For all other kinds the name is looked up in the catalog and the kit/doc name in the tail of the entry found is used.

The other parameters:
A parameter of the form <byte1> . <byte2> gives separate specifications of the two 12-bit bytes in the word.

<integer>: The value in the tail as the word or byte in question.

<name>: The name is looked up in the catalog and the value of the word or byte in question in the entry tail found is used.

If the parameter list does not specify all of the tail, the rest of the tail is set to zero.

3.22.5 Storage Requirements 3.22.5

1536 bytes plus space for FP.

3.22.6 Error Messages 3.22.6

***entry call
No left side in call of the program.

***entry param <parameter>

Parameter error in call of the program.

***entry <name> unknown

A parameter was searched in the catalog but not found.

***entry <result name> change kind impossible

A change of an area to a non-area or vice versa was attempted.

***entry <result name> change bs device impossible

A change of kit/doc name of an area was attempted.

***entry <result name> bs device unknown

The bs device specified was not found.

***entry <result name> no resources

The resources of the job did not allow the wanted creation or change of an entry.

***entry <result name> no room

Name overflow in the main catalog exceeded the limit.

***entry <result name> entry in use

The entry could not be changed because another job was using it.

If any message appears no entry is created or changed.

3.23 finis

"finis" terminates the job.

3.23.1 Call

finis output.$\left\{\begin{matrix}\text{yes} \\ \text{no}\end{matrix}\right\}_{0}^{1}$

3.23.2 Function

The current output file is terminated (emptying of buffers etc.) and a finis job message is sent to the parent (the operating system), who is then expected to remove the job.

If parameter output.no is specified, and the parent is BOSS, the finis message will specify that output is not wanted.

3.23.3 Storage Requirements

1024 bytes plus space for FP.

3.23.4 Error Messages

***finis call
The program was called with a left hand side - the finis action is still performed.

***finis param <parameter>
Erroneous parameter in the call - the finis action is still performed.

3.24 head 3.24

Prints a number of form feeds and a page head containing the name of the job and the date and clock.

3.24.1 Example 3.24.1

The output from two programs is separated in a nice way by calling "head" in between:

```
head 1
```

This command prints one form and a page head on current output.

```
head iso cpu
```

This command prints a page head with the date in iso form (i.e. year month day), followed by the cpu time used by the job.

3.24.2 Call 3.24.2

```
{ <out file> = } 1
{             } 0

head { <s> <integer> } 3
     { <s> cpu       }
     { <s> iso old   } 0
```

3.24.3 Function 3.24.3

In an integer is given as parameter that many form feeds are printed. Next, one line consisting of job name, date, and clock is printed. In an outfile is specified this is used for the output, else the current output file is used. Date in iso form is standard. The parameter old will cause the date to be printed as day month year.

3.24.4 Storage Requirements 3.24.4

1024 bytes plus space for FP.

***head param <parameter>

Parameter error in the call. A page head is still output.

3.25 headpunch 3.25

The program punches a readable text pattern to the parameters.
The same function is also written on current output.

3.25.1 Examples 3.25.1

The FP call:

```
headpunch
```

punches a textpattern consisting of: jobname, date, and clock

```
headpunch data 2
```

punches the text pattern: data 2

```
headpunch in.textarea
```

punches a text pattern corresponding to the contents of textarea.

The FP calls:

```
o top
head
message tre
copy textprogram
o c
headpunch in.top
tpe=copy taxprogram
```

will cause the output to start with an optically readable text,
e.g. ta0 1977.03.22 11.12 tre taxprogram 7 segm. 12345/678901.

3.25.2 Call 3.25.2

headpunch $\left\{ \text{<parameter list>} \right\}_{0}^{1}$

<parameter list> ::= $\left\{ \begin{array}{l} \text{in.<bs-area>} \\ \text{<any sequence obeying the fp syntax>} \end{array} \right\}$

3.25.3 Function 3.25.3

The textpattern is output on punch in tpn mode, and the text is written on current output. Current output should not be punch.

If no parameters are specified, the output will be: jobname, date, and clock.

If the parameter is in.<bs-area> and the program succeeds to connect to this area, the contents of this area are output until a character=25 (EM) or >127 is found or until 120 characters have been output. NL is punched as space.

In all other cases the parameter list is copied.

3.25.4 Storage Requirements 3.25.4

1536 bytes plus space for FP.

3.25.5 Error Messages 3.25.5

***headpunch call
Left hand side in the call of the program.

***headpunch connect tpn
tpn cannot be connected. The program terminates.

3.26 i

Selects a new file as current input. The former file may later be resumed at the position where it was left (e.g. by a call of "end").

3.26.1 Example

If we have the following FP commands in a job file

```
i commds1
i commds2
```

the first will cause FP to start reading from the file 'commds1'. When this file is exhausted FP will return to the job and read the text call of "i", which in turn causes FP to read commands from the file 'commds2'.

"edit" reads the editorial commands from current input. The commands to "Edit" may be kept on a separate file 'editcomds' if the editing is done by the following composite FP commands:

```
(i editcomds              ; the file 'editcomds' is connected
                          ; as current input file.
newtext=edit oldtext      ; call of "edit"
end)                      ; reselects the previous current
                          ; input file
```

The parenthesis are essential here. If they were omitted FP would immediately start reading from the file 'editcomds' instead of calling "edit" (the "end" command is not necessary if "edit" reads and accepts all of the file 'editcomds'. It ensures however that FP does not start reading from 'editcomds').

3.26.2 Call

i <s> <file name>

3.26.3 Function

The current input file is stacked so that reading may be resumed later (when the new file is exhausted or by a call of "end"). Next the specified file is connected as current input.

3.26.4 Storage Requirements

1024 bytes plus space for FP.

3.26.5 Error Messages

***i call

Left hand side in the call.

***1 param

Parameter error in the call.

***i <document name> <cause>

The specified file could not be connected for some reason which is explained by <cause> as follows:

no resources	forbidden by the parent (the operating system)
disconnected	device disconnected
name unknown	the file did not exist
kind illegal	the file could not be used for input
reserved	the file was used by another job.

In case of any error FP forgets about all previous current input files and returns to the primary input file (the job file).

3.27 if

Makes the execution of the next FP command conditioned by the values of one (or several) mode bits. The condition may reflect the success of the latest program executed as the ok and warning bits are set at program end (or it may correspond to the mode bits as set by a call of the program mode).

3.27.1 Example

If the translation of an ALGOL source program goes wrong, you want to do the translation once more with listing of the program. If the translation error is serious you want to terminate the run. Proceed as follows:

```
prog1=algol text           ; translate
if warning.yes             ; if syntactical errors
prog1=algol text list.yes  ; then translate and list
if ok.no                   ; if serious errors
finis                      ; then terminate job
prog1                      ; else execute the program
```

3.27.2 Call


```
if { <s> <mode bit> . { yes } }∞
                      { no  }1

                  { <integer }
                  { ok       }
                  { listing  }
<mode bit>::=     { warning  }
                  { error    }
                  { pause    }
                  { all      }
```

3.27.3 Function 3.27.3

The next (possibly composite) FP command is executed if each of the mode bits mentioned in the parameter list has the specified value 'yes' or 'no'. If not, the next FP command is skipped. The program "if" does not change any mode bit (even not the ok and warning bits) hence repeated questions may be asked on the same mode bits by several successive calls of "if".

3.27.4 Storage Requirements 3.27.4

1024 bytes plus space for FP.

3.27.4 Error Messages 3.27.4

***if call

Left hand side in the call - does not affect the function of the program.

***if param <parameter>

Wrong parameter in the call. The erroneous parameter is skipped and the program continues with the next parameter.

3.28 job 3.28

Makes it possible to use files containing a BOSS job specification in the first line in runs directly under s.

Under BOSS, it becomes possible to enroll the current edit file as a terminal job by the command "go", regardless of a possible job specification in the first line.

3.28.1 Example 3.28.1

Assume you have a file, 'jobfile', containing:

```
job fgs 1 274001
p=algol tp
p pip
finis
```

If you type 'i jobfile', FP will simply skip the line >job ...> and read the next line.

If you run under BOSS and enroll the same file as a terminal job by the command 'run', BOSS will read the job specification and FP will start reading the next line.

If you enroll the same file as a terminal job by the command "go", FP will read the job specification, skip it, and read the next line.

3.28.2 Call 3.28.2

```
job <s> <parameter list>
```

<parameter list> may consist of any sequence of parameters obeying the FP syntax.

3.28.3 Function 3.28.3

"job" returns to 'FP end program' with ok.yes and warning.no.

3.28.4 Storage Requirements 3.28.4

512 halfwords plus space for FP.

3.28.5 Error Messages 3.28.5

None.

3.29 kit 3.29

Sends a mount disc message to the parent (the operating system) demanding a disc kit with a specified name to be mounted on a specific disc unit (cf. ref. 9, ch. 3 and 5).

3.29.1 Example 3.29.1

The FP command

```
kit 12 disc5
```

asks for mounting of the disc kit 'disc5' on the disc unit with device number 12.

3.29.2 Call 3.29.2

kit <s> <device no> <s> <kit name>
where <device no> is an integer and <kit name> is a name.

3.29.3 Function 3.29.3

A mount kit message containing the device number and name specified is sent to the parent.

3.29.4 Storage Requirements 3.29.4

1536 bytes plus space for FP.

3.29.5 Error Messages 3.29.5

***kit call
Left hand side in the call of kit.

***kit <parameter list> parameter error

Parameter error in the call of kit.

***kit <parameter list> not available

The kit specified by the kit name is not available for the job.

In case of any error no mount kit message is sent.

3.30 label 3.30

Outputs a BOSS label on file 0 of the specified tape.

3.30.1 Example 3.30.1

The FP call:

```
label mt1h mt123456 p 789012
```

outputs a BOSS label on mt123456.

If you have a filedescriptor, e.g.:

```
f=set mt1h mt123456 0 1
```

the call:

```
label f f p 789012
```

will have the same effect.

3.30.2 Call 3.30.2

label<modekind><mtname>{<access><project number>}$_{0}^{1}$

<modekind> ::= mt1h / mto, mte | mt11 / nrz, nrze | mthh | mthl

<mtname> ::= <name of magnetic tape> | <name of filedescriptor>

<access> ::= p | r | w

<project number>::=<integer, max. 999999>

Name of magnetic tape must start with mt followed by exactly 6 characters, the first 2 may be letters or digits, the 4 last must be digits.

3.30.3 Function 3.30.3

A label in the format accepted by BCSS (cf. ref. [10], sec. 5.3) will be written in file 0 of the tape. Next, two tapemarks are written (i.e. an empty file 1), and approx. 5 inches of tape is erased.

Notice: this means that the first part of the previous contents of the tape will unconditionally be destroyed.

3.30.4 Error Messages 3.30.4

***label, call
Left hand parameter in the call.

***label, <parameter> param
Illegal parameter in the call.

***label, <parameter> modekind error
Filedescriptor does not describe a magnetic tape.

***label, <modekind param> unknown
Modekind does not describe mto, mte, nrz, nrze, or a filedescriptor.

***label, <parameter> illegal tapename
Tapename is illegal.

***label, <parameter> illegal access kind
Access must be p, r, or w.

***label, project number missing
If access specified, a project number is demanded.

***label, <parameter> illegal project number
Project number must be max. 999999.

3.30 label

Se ændring

3.30

Outputs a BOSS label on file 0 of the specified tape.

3.30.1 Example

3.30.1

The FP call:

```
label mto mt123456 p 789012
```

outputs a BOSS label on mt123456.

If you have a filedescriptor, e.g.:

```
f=set mto mt123456 0 1
```

the call:

```
label f f p 789012
```

will have the same effect.

3.30.2 Call

3.30.2

kun boss

label <modekind> <mtname> { <access> <project number> }$^{1}_{0}$

<modekind> ::= { mto | mte | nrz | nrze | <name of filedescriptor> }

<mtname> ::= { <name of magnetic tape> | <name of filedescriptor> }

<access> ::= { p | r | w }

<project number> ::= <integer, max. 999999>

Name of magnetic tape must start with mt followed by exactly 6 characters, the first 2 may be letters or digits, the 4 last must be digits.

3.30.3 Function 3.30.3

A label in the format accepted by BOSS (cf. ref. [10], sec. 5.3) will be written in file 0 of the tape. Next, two tapemarks are written (i.e. an empty file 1), and approx. 5 inches of tape is erased.

Notice: this means that the first part of the previous contents of the tape will unconditionally be destroyed.

3.30.4 Error Messages 3.30.4

***label, call
Left hand parameter in the call.

***label, <parameter> param
Illegal parameter in the call.

***label, <parameter> modekind error
Filedescriptor does not describe a magnetic tape.

***label, <modekind param> unknown
Modekind does not describe mto, mte, nrz, nrze, or a filedescriptor.

***label, <parameter> illegal tapename
Tapename is illegal.

***label, <parameter> illegal access kind
Access must be p, r, or w.

***label, project number missing
If access specified, a project number is demanded.

***label, <parameter> illegal project number
Project number must be max. 999999.

***label, too many parameters

The program accepts max. 4 parameters.

***label, parameter missing

The program demands at least 2 parameters.

***label, connect tape unsuccessful

Hard error.

3.31 load 3.31

The program can input catalog entries and bs files from magnetic tape files generated by the program "save".

3.31.1 Example 3.31.1

All catalog entries and bs files saved on mt471100 file 1 are reestablished by the FP command:

```
load mt471100.1
```

In case: t=set mto mt471100 0 1
the same is obtained by the command:

```
load t.0
```

All catalog entries and bs files of scope temp plus the entry by name pap are loaded by the FP command:

```
load mt471100.1 scope.temp pap
```

See also: Further examples.

3.31.2 Call 3.31.2

$$\left\{\text{<outfile>=}\right\}_{0}^{1}$$

load $\left\{\text{<mountparam>.}\right\}_{0}^{1}$ <tape parameter> $\left\{\text{<special param>}\right\}_{0}^{1}$ $\left\{\text{<load spec>}\right\}_{0}^{1}$

<mountparam>::= $\left\{\text{mountspec.<deviceno>.}\right\}_{0}^{1}$ $\left\{\text{<modekind>.}\right\}_{0}^{1}$ $\left\{\text{release.}\begin{matrix}\text{yes}\\\text{no}\end{matrix}\right\}_{0}^{1}$

<tape parameter>::= <tapename>.<fileno> $\left\{\text{.<tapename for next volume>}\right\}_{0}^{9}$

<tapename>::= { <filedescriptor describing a magnetic tape file> | <name of magnetic tape> }

<fileno>::= { last | <integer> }

<special param>::= { check { .yes | .no } | survey { .yes | .no } | load { .yes | .no } | list { .yes | .no | .name } }

<load spec>::= { <modifiers> | <kit spec> | <entry spec> }

<modifiers::= { changekit.<bs device spec>.<bs device spec> | changekit.all.<bs device spec> | newscope.<newscope spec> } ∞ / 0

<kit spec>::= kit.<bs device spec>

<bs device spec>::= { <bs device name> | main | 0 | 1 }

<entry spec>::= { <name> | <name>.scope.<scope spec> | docname.<docname> | docname.<docname>.scope.<scope spec> | scope.<scope spec> } ∞ / 0

3.31.3 Function

3.31.3

The contents of the dump label (see "save") are checked and listed on current out.

Next the program loads from the magnetic tape all the entries and bs files specified by <load spec>. If <entry spec> is empty all the entries are loaded.

Each entry is created with scope and <bs device spec> as defined in the entry record. For area entries <bs device spec> defines the kit name for non-area entries, the kit into which the entry is permanented.

3.31.3.1 Function, mountparam 3.31.3.1

If no mountparam is specified, the program will use a standard magtape station, modekind=mto (or in case mtname is a filedescriptor, then the modekind of this filedescriptor) and the tape will be released at end of program.

e.g. mountspec.10.nrz.release.no.
mountspec.10.
nrz.
release.no.

3.31.3.2 Function, tapeparameter 3.31.3.2

In case mtname is a filedescriptor, filenumber will be understood relative to the filenumber in the filedescriptor. The modekind of the filedescriptor will be used.

If <fileno>=last, the file in front of the first file which does not contain a version label is loaded. This parameter gives a longer run time than an integer parameter.

Tapenames of following volumes are only necessary in case the following volume has a name different from what is stated in the continuation block. This may be the case if saving was performed with 2 parallel tapes.

3.31.3.3 Function, special param

check.{yes / no}

Std. is check.yes
If check.no, then the program continues, if the mtname or the fileno in the dumplabel is wrong. Also when the dumplabel is a continuation label.

survey.{yes / no}

Std. is survey.no
If survey.yes, then all entries from file 1 to <fileno> are listed but not loaded.

load.{yes / no}

Std. is load.yes, the specified entries are loaded.
If load.no, then all specified entries in the file are listed but not loaded.

list.{yes / no / name}

Std. is list.yes, all loaded entries are listed.
If list.name, then only the names of the entries are listed.
If an outfile is specified, this file is used for output, otherwise current output file is used.

3.31.3.4 Function, modifiers

If <bs device spec> = 0, the area is - if possible - created on a drum, otherwise on a disc.

If <bs device spec> = 1, the area is - if possible - created on a disc, otherwise on a drum.

If <bs device spec> = main, the area is created on the bs-device containing the main catalog.

If <bs device spec> = a name <> main, the area is created on the bs device with this name.

changekit.<saved kit>.<loaded kit>

This parameter is valid for the total call.
Each entry, which on the tape is described as an entry on <saved kit> will be loaded on <loaded kit>.
E.g. changekit.disc1.disc2

changekit.all.<loaded kit>

As above, except all entries specified, no matter their document name, are loaded on <loaded kit>.

newscope.<newscope spec>

Std. is newscope.std, meaning no change in scope.
This parameter is valid for the following entry specifications. They will all be created with <newscope spec>.
<newscope spec>::= temp login user project std
e.g. newscope.temp

3.31.3.5 Function, kit spec

3.31.3.5

General about <bs device spec>, see modifiers.

kit.<saved kit> Std. is main, meaning all devices.
Only entries which in the tape is described as addressing this kit will be loaded (or entries which after a changekit parameter is addressing this kit). The parameter is valid for all following <entry spec> until a new kit parameter is specified. If a kit parameter specifies a not connected kit, a changekit, changing this kit to a connected kit name, must be specified earlier in the parameter list.
E.g. kit.disc2

3.31.3.6 Function, entry spec 3.31.3.6

<scope spec>::= temp | login | user | project | own | system | perm | all

<name>
All entries of the name are loaded.

scope.<scope spec>
All entries with this scope are loaded.

<name>.scope.<scope spec>
An entry of specified name and scope is loaded.

docname.<docname>
All entries with specified docname (maybe kitname) are loaded.

docname.<docname>.scope.<scope spec>
All entries with specified docname (maybe kitname) and specified scope are loaded.

3.31.4 Tape Format 3.31.4

See "save".

3.31.5 Load of systemdump 3.31.5

Entries which are saved at scope.perm (or scope.all) may be loaded in a BOSS job. Entries with bases corresponding to scopes temp, login, and user will be loaded if their name is specified. For sake of security it is decided that entries of scope project must be specified by <name>.scope.project.

3.31.6 Storage Requirements 3.31.6

12000 bytes.

***load: error in tapeparam <erroneous and following parameters>
Parameter error in the call. The program terminates.

***load: error in modekind spec.
<tapename> describes an entry of kind 18, but mode is neither 0 nor 4.

***load: error in param <erroneous and following parameters>
Parameter error in the call. The program terminates.

***load param, kitnames exceeded
The program does not accept more than 10 bs device names different from the bs devices connected. The program terminates.
Remedy: two calls of load.

***load: no dumplabel on file <fileno>
The file contains no dumplabel. The program terminates.

***load: dumplabel <specification>
Error in the specified part of the dumplabel. The program terminates.

<name> entry inconsistent
<name> code inconsistent
The date of an external procedure is incorrectly described, either in the catalog entry or in the code. The entry is loaded.

<name> bad tape: <pattern>
Hard error during run. The pattern shows the status word.

<name> bad tape: <pattern> blocklength = <blocklength>
Blocklength error on the tape.

<name> bad tape, blocks skipped <skipped blocks>
The blocks could not be interpreted by the program.

<name> bad tape, segm. loaded <segments>
The segments loaded do not correspond to the number of segments specified in the entry record.

bad tape, entry no. <d> missing
<name> bad tape, segm. no. <d> missing

<name> monitor <xx> result <y> <explanation>
The call of the ALGOL procedure monitor with the parameter <xx> gave the unwanted result <y>.
<explanation>:
device not mounted
process base error
no work resources
no perm resources
entry in use
impossible (catalog error)

***not found <entry spec>
<entry spec> was not found on the tape.

***load not ok <d>
This message occurs at program exit in case of any error.
<d> is the number of errors.

3.31.8 Further Examples

3.31.8

1) The programmer wants to load the entries and bs files pr1 and pr2 from a saved file, which contains several other entries, furthermore he wants to change the scope of pr2 to user:

```
load mt471100.2 pr1 newscope.user pr2
```

2) The programmer wants to load the entry named pip and all his entries which belong to the catalog on kit5, from a file which contains other entries as well:

```
load mt471100.3 pip kit.kit5 scope.own
```

3) The programmer wants to check the contents of mt471100

```
load mt471100.last survey.yes
```

4) The programmer wants to load file 8 but gets the output

```
dump mt471100  006  vers. 130473.12  s=1  unhappydays
***load dumplabel fileno
```

at repeated calls. This suggests that the start of the magnetic tape has been overwritten and probably the wanted file will be found on file 10.
Try the FP command:

```
load mt471100.10 check.no load.no
```

3.32 lookup 3.32

Finds and lists catalog entries with specified name.

3.32.1 Example 3.32.1

The FP command

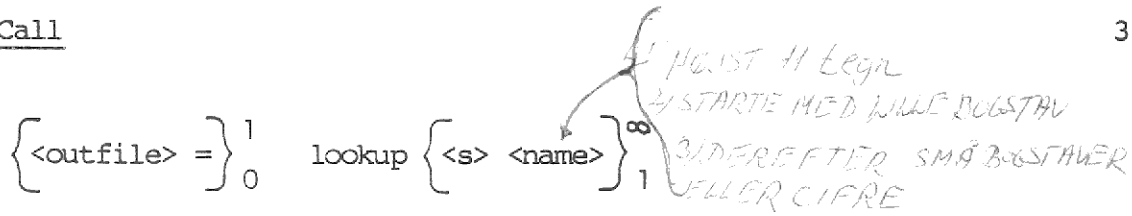
```
lookup pip
```

finds and lists the entries with name 'pip' and prints something like:

```
pip         =set 16 disc d.770523.1021 0 0 0 0    ; temp
            ;  92 17 0 -56 -56
```

The first line gives the tail and the scope of the entry, the second line gives the entry head.

3.32.2 Call 3.32.2



3.32.3 Function 3.32.3

Each name in the list is searched in the catalog and all entries with this name which may be accessed by the job are listed. If an <outfile> is present this file is used for the output - otherwise the current output file is used.

3.32.4 Format of the Output 3.32.4

Each catalog entry is listed as two lines:

```
<name> =set <entry tail> ; <scope spec>
        ;  <entry head>
```

The name and entry tail appear exactly as in a call of the program "set" for creating the entry.

The scope specification has the form

<scope> { .<device name> }[1 0]

where <scope> is one of temp, login, user, project, system, or *** (the last one means scope undefined) and where a <device name> tells that the entry is permanented into the auxiliary catalog on this device.

The entry head is output as the five integers

<first slice> <name key> <catalog key> <interval lower> <interval upper>

as described in the manuals for the monitor (ref. [1] and [2]).

3.32.5 Storage Requirements 3.32.5

2560 bytes plus space for FP.

3.32.6 Error Messages 3.32.6

***lookup connect <outfile>
The specified output file could not be connected - current output is used instead.

***lookup param <parameter>
Parameter error. The remainder of the parameter list is skipped.

***lookup <name> unknown
No entries with the given name was found. The program continues with the next name in the list.

***lookup <name> no resources

The program has terminated because the job has too few area processes.

Lookup link <s> 10 er et algolprogram som fortæller hvorvidt eks 10 (båndstationen) er knyttet til processen eller ej.

3.33 message 3.33

May be used (together with "head") to make nice headings on the output. The parameter list in the call of message is simply output when the program is called.

3.33.1 Example 3.33.1

The FP command

message program run no.1

outputs the text , program run no.1 , on current output.

3.33.2 Call 3.33.2

$$\left\{\text{<outfile> =}\right\}_{0}^{1} \quad \text{message <s> <parameter list>}$$

<parameter list> may consist of any sequence of parameters obeying the FP syntax.

3.33.3 Function 3.33.3

The parameter list is copied on <outfile> or current output (if no outfile is specified). The output is terminated by an NL character.

3.33.4 Storage Requirements 3.33.4

512 bytes plus space for FP.

***message connect <outfile>

The specified output file could not be connected. Current output is used instead.

3.34 mode 3.34

Changes the FP mode bits specified in the call and may thereby change the working cycle of FP.

3.34.1 Example 3.34.1

The FP command:

```
mode list.yes
```

causes FP to change to list mode i.e. each FP command is listed on current output just before execution.

The FP command:

```
mode what
```

causes all modebits to be listed.

3.34.2 Call 3.34.2

```
mode { <s> what                    }
     { <s> <mode bit>. { yes } }   } ∞
                       { no  }     } 1
```

```
<mode bit> ::= { <integer> }
               { listing   }
               { warning   }
               { ok        }
               { error     }
               { pause     }
               { list      }
               { all       }
```

The integer values of the mode bit names are as follows: listing=15, warning=17, ok=18, error=19, pause=20, list=23. Bit 16 is used internally by FP.

The mode bits are explained in ref. [8], section 4.2.

3.34.3 Function

The FP mode bits are changed as specified in the call.

3.34.4 Storage Requirements

1024 bytes plus space for FP.

3.34.5 Error Messages

***mode call
Left hand side in the call - does not affect the function of the program.

***mode param <parameter>
Wrong parameter in the call. The parameter is skipped and the program continues with the next parameter.

3.35 mount

Sends a mount message to the parent (the operating system) who is then expected to ask the operator to mount the tape reel (cf. ref. [10], section 6.1). The Program does not await the mounting, unless there is asked for mounting of an unspecified worktape.

3.35.1 Example

When a program needs a magnetic tape reel which is not mounted, the mounting is automatically requested and the job waits for it. The scheduling of a job which uses several tape reels is however improved if the tape reels are requested right at the beginning of the job, i.e. if the tape reels named 'mt280007', 'mt280008', and 'mt280009' are needed during the job one may start the job file with the FP commands:

```
mount mt280007
mount mt280008
mount mt280009
```

If p7, p8, p9 are names for files on these magtapes, e.g.

```
p7=set mto mt280007 0 3
p8=set mto mt280008 0 1
p9=set mto mt280009 0 2
```

the same result is obtained by the FP commands:

```
mount p7
mount p8
mount p9
```

An unspecified worktape is requested as follows:

```
workfile=set mto 0 0 1
mount workfile
```

This call of "mount" asks for mounting of a worktape and places the name of the magtape reel in the entry. File number 1 on the tape is now available under the name 'workfile'.

The workfile is released and made available to other users when the job terminates or if the tape is released during the job. One may suspend the use of the worktape by a "suspend" command (cf. the description of "suspend").

3.35.2 Call 3.35.2

mount <s> <name>

3.35.3 Function 3.35.3

A mount message is sent to the parent.
The name in the message is found as follows: the name is looked up in the catalog. If an entry describing a magnetic tape file (kind=18) is found and if this entry is not protected (e.g. not of scope system) the document name in the entry is used. Otherwise the name specified is used. The document name in the entry may be empty (zero). In this case a worktape is mounted and the name of the worktape placed as document name in the entry.

3.35.4 Storage Requirements 3.35.4

1536 bytes plus space for FP.

3.35.5 Error Messages 3.35.5

***mount call
Left hand side in call of mount.

***mount <parameter list> parameter error
Parameter error in call of the program.

In case of any error no mount message is sent.

3.36 mountspec 3.36

Sends a mount special message to the parent (the operating system) limiting a later mounting of the specified magnetic tape reel to the station with the specified device no. (cf. ref. [10], section 6.1).

3.36.1 Example 3.36.1

If the installation has some standard magnetic tape stations (say 9-track) and a non standard (say 7-track) with device number 6 and one has a 7-track tape reel named mt123456 the FP command

```
mountspec 6 mt123456
```

ensures that BOSS will accept the reel only when mounted on station No 6. If 'pip' is the name of a file on a magtape e.g.

```
pip=set mto mt123456 0 7
```

the same result is obtained by the FP command

```
mountspec 6 pip
```

3.36.2 Call 3.36.2

mountspec <s> <device no> <s> <name>
where <device no> is an integer.

3.36.3 Function 3.36.3

The name is looked up in the catalog. If an entry describing a magnetic tape file (kind=18) is found and if this entry is not protected (e.g. not of scope system) the document name in the entry is used. Otherwise the name specified is used. Next a mount special message containing the specified device number and the name is sent to the parent.

3.36.4 Storage Requirements

1536 bytes plus space for FP.

3.36.5 Error Messages

***mountspec call
Left hand side in the call of the program.

***mountspec <parameter list> parameter error
Parameter error in the call of the program.

***mountspec <parameter list> tape name missing
The entry specified has a zero document name.

In case of any error no mountspec message is sent.

3.37 move

Performs blockwise copying of files on backing storage or magnetic tape.

3.37.1 Example

The contents of the backing storage area with name , text4 , is moved to file 5 on the magnetic tape reel named , mt314711 , by the FP commands:

```
file5=set mto mt314711 0 5
file5=move text4
```

Files number 3, 4, 5, 6 on the magtape , mt312223 , will be copied to the magnetic tape , mt312224 , starting at file number 7 by the FP commands:

```
fromfile=set mto mt312223 0 3
  tofile=set mto mt312224 0 7
  tofile=move fromfile.4
```

3.37.2 Call

The program may be called in two ways depending on the kind of the left hand side:

$$\text{<bs file>} = \text{move} \left\{ \text{message.yes} \right\}_{0}^{1} \left\{ \text{<bs-file>} \mid \text{<mt-file>} \right\}$$

or

$$\text{<mt-file>} = \text{move} \left\{ \text{message.yes} \right\}_{0}^{1} \left\{ \text{<bs-file>} \mid \text{<mt-file-set>} \right\}_{0}^{\infty}$$

<bs-file> must be a catalog entry describing a file on the backing storage.

<mt-file> must be a catalog entry describing a file on a magnetic tape.

<mt-file-set>::= <mt-file> [.<no-of-files> [.<skip>]$_{0}^{1}$]$_{0}^{1}$

<no-of-files> is an integer defining how many files to copy.
<skip> is an integer defining how many files to skip before the copying.

3.37.3 Function 3.37.3

Move performs blockwise copying of files on backing storage or magnetic tape.

The parameter message.yes will cause output of bytes and checksum.

If the output file specifies magnetic tape, as many files as specified in the input parameters will be written, separated by tape marks.

If the files on both sides of the move-call specifies magnetic tape, the block lengths of the input file(s) are kept on the output file(s).

If the output file describes a bs area, the length of the area will be decreased corresponding to its new contents. If the input file describes a bs area too, the last 5 words of the catalog entry tail will be inserted in the output tail.

3.37.4 Storage Requirements 3.37.4

The core storage required for move is 2850 bytes plus space for FP.

When copying from a magnetic tape with block lengths greater than 512 bytes, more core storage will be needed. In a process with 10000 bytes of core storage, the maximum block length is about 1600 bytes.

3.37.5 Error Messages

3.37.5

***move: no core

The process area is too small to contain the input and output buffers.

***move call

No left hand side is specified in the call.

***move param: <parameter list>

An input specification has an erroneous format. The specification is shown as <parameter list>. *)

***move: input kind

***move: output kind

The specified file is neither a bs file nor an mt file. *)

***move: connect input

***move: connect output

It has not been possible to connect an input or an output file.

***move: too many parameters

It is attempted to copy more than one file to a bs file.

***move: change error

It is not possible to change the catalog entry describing the output bs file.

*) The parameters will be checked and handled one by one. Therefore one or more files may have been copied even if the program is terminated by an alarm.

3.37.6 Further Examples of Use

3.37.6

In the following the catalog entries mt1 and mt2 describe file number one on two magnetic tapes, and bs1, bs2 —— describe areas on the backing storage.

3.38 newjob

Sends a newjob message to the parent (the operating system) demanding the specified file enrolled as job file in a new off line job i.e. in this way a new job is created. The actual job continues with the next FP command. Further details are found in section 1.3, newjob and replacejob, in ref. [10].

3.38.1 Call

newjob <s> <file name> $\left\{ \begin{matrix} \text{<name of remote batch printer>} \\ 0 \end{matrix} \right\}^{1}$

where <file name> is a name of a permanent job file.

<name of remote batch printer>::= <name of max 6 char>

3.38.2 Function

A newjob message containing the specified name(s) is sent to the parent.

3.38.3 Storage Requirements

1536 bytes plus space for FP.

3.38.4 Error Messages

***newjob call
Left hand side in the call of the program.

***newjob <parameter list> parameter error
Parameter error in the call of the program.

***newjob <filename> <error cause>

Error during creation of the new job. The cause may be any of the following:

job queue full
job file not permanent
job file unknown
job file unreadable
user index too large
illegal identification
user index conflict
job file too long
temp claim exceeded
option unknown
param error at job
syntax error at job
line too long
attention status at remote batch terminal
device unknown
device not printer
parent device disconnected
remote batch malfunction

In case of any error no new job is created.

3.39 nextfile 3.39

Adds one to the file number in the tails of the catalog entries specified.

3.39.1 Example 3.39.1

If the catalog entries 'to' and 'from' describe file 3 of the magtape 'mt312223' and file 6 of the magtape 'mt312224', respectively, the FP command

nextfile to from

will change them to describe file 4 and 7 of the tapes in question.

3.39.2 Call 3.39.2

nextfile $\left\{ \text{<s> <name>} \right\}_{1}^{\infty}$

3.39.3 Function 3.39.3

For each name in the list a catalog lookup is made and the file number in the tail of the entry is increased by one.

3.39.4 Storage Requirements 3.39.4

1536 bytes plus space for FP.

3.39.5 Error Messages 3.39.5

***nextfile call

Left hand side in the call. The program terminates without further actions.

***nextfile param <parameter>

Parameter error. The faulty parameter is skipped and the program continues with the next parameter.

***nextfile <name> unknown

No entry with the specified name was found. The program continues with the next parameter.

***nextfile <name> protected

The job was not allowed to change the tail in the entry found. The program continues with the next parameter.

3.40 o

Selects a new file as current output.

3.40.1 Example

The text output from an ALGOL translation may be put on a special file in the following way:

```
o list                          ; the file 'list' is chosen as current output
program=algol text list.yes     ; translation of the algol program
o c                             ; current output is shifted back to the
                                  primary output file
```

Note that in case of larger programs, the area 'list' may be too small and unable to be extended. Remedy: set a sufficiently larger area before the call of the program '.' e.g. list=set 150.

3.40.2 Call

o <s> <file>

3.40.3 Function

The actual use of the current output file is terminated (emptying of buffers) and the file given as parameter is connected as current output.

There is no stacking and unstacking of previously used output files as for current input files.

If <file> is not found in the catalog an area with this name on the backing storage (preferably on a disc) is created and connected as current output. The name 'c', however, is used for the primary output file and is treated in the following way. Whenever the program "o" connects current output to 'o' (either

because of the command 'o c' or because of some error) the following is done: if a catalog entry named 'c' is present, the file described by this entry is connected. If the catalog entry is not present it is created as describing the primary output file and current output is connected to the file.

3.40.4 Storage Requirements 3.40.4

1024 bytes plus space for FP.

3.40.5 Error Messages 3.40.5

***o call
Left hand side in the call.

***o param <param>
Parameter error in the call.

***o <document name> <cause>
The file could not be connected. The reason is explained by <cause>:

no resources	the job resources are exceeded
disconnected	the device is disconnected
kind illegal	the file could not be used for output
reserved	the file was used by another job.

In case of any error the primary output file is connected as current output file.

3.41 online 3.41

Turns the job into the conversational mode where the current input to the job is typed on the terminal at run time. A conversational job is very resource demanding and the user must have a special option in the user catalog (cf. ref. [10], section 3.2).

3.41.1 Call 3.41.1

online.

3.41.2 Function 3.41.2

The process 'terminal' is connected as current input and selected as a new primary input.

Contrary to the FP command

```
i term
```

the FP command

```
online
```

has the advantage that an FP syntax error will not return current input to the job file.

3.41.3 Storage Requirements 3.41.3

512 bytes plus space for FP.

3.41.4 Error Messages 3.41.5

***online connect terminal

The job does not have the option 'online yes'.

3.42 opcomm

Sends the parameter list in the call as a print message to the parent (the operating system) with request for an answer from the operator and types the answer (when received) on current output.

3.42.1 Example

A user with initials hsr and project number 47 is placed at a terminal and needs a new project tape reel. The labeling of the reel and an answer back telling the reel name may be requested by the FP commands:

```
opmess label new p 47 reel
opcomm return name of reel
```

This causes the following lines to appear (among the other messages from BOSS) on the main console

```
message hsr0 label new p 47 reel
pause hsr0 return name of reel
```

When the operator has labeled the reel - say with the name 'mt271536' - he returns the name to hsr by typing

```
answer hsr0 mt271536
```

on the main console.

In the meantime "opcomm" has been waiting for the answer. The answer is now output as the text

```
*operator answer: mt271536 0
```

on current output (for hsr0).

3.42.2 Call

opcomm <s><parameter list>

The parameter list may consist of any sequence obeying the FP syntax.

3.42.3 Function

3.42.3

The first 21 characters (if that many are present) in the parameter list are packed as a print message and sent to the parent.
The answer is then awaited and when it arrives printed on current output in the form

*operator answer: <name> <integer>

where <name> and <integer> are the answer as typed by the operator.

3.42.4 Storage Requirements

3.42.4

1536 bytes plus space for FP.

3.42.5 Error Messages

3.42.5

***opcomm call

Left hans side in call of the program. No message is sent and no waiting is performed.

3.43 opmess 3.43

Sends the parameter list in the call as a print message to the parent (the operating system). If the operating system is BOSS the message is typed on the main console.

3.43.1 Example 3.43.1

An example of the use is given in the description of the program "opcomm".

3.43.2 Call 3.43.2

opmess <s> <parameter list>

The parameter list may consist of any sequence of parameters obeying the FP syntax.

3.43.3 Function 3.43.3

The first 21 characters (if that many are present) in the parameter list are packed as a print message and sent to the parent.

3.43.4 Storage Requirements 3.43.4

1536 bytes plus space for FP.

3.43.5 Error Messages 3.43.5

***opmess call
Left hand side in call of the program. No message is sent.

3.44 permanent 3.44

The program changes the catalog key of the specified entry to the specified integer.

3.44.1 Example 3.44.1

The fp call:

```
permanent pip.3
```

will change the catalog key of the entry pip to 3. Normally the program scope should be used.

3.44.2 Call 3.44.2

permanent $\left\{ \text{<s><name>.<integer>} \right\}_{0}^{\infty}$

3.44.3 Function 3.44.3

For each name a catalog lookup is made and the catalog key of the entry found is changed to the specified value. This may cause an illegal scope.

3.44.4 Storage Requirements 3.44.4

2048 bytes plus space for FP.

3.44.5 Error Messages 3.44.5

***permanent call

Left hand side in the call. The program terminates.

***permanent param <parameter>

Parameter error in the call. The faulty parameter is not treated.

***permanent <name> unknown

No entry with the specified name was found.

***permanent <name> protected

The job was not allowed to change the entry key of the specified entry.

***permanent <name> no resources

The job has no permanent resources left on the relevant kit.

***permanent <name> error

Catalog or hard error.

3.45 print 3.45

Prints from a backing storage area or directly from the core store with specified formats. The program is primarily intended for printing of dumped areas.

3.45.1 Example 3.45.1

The core of the job process has been dumped into a backing storage area named 'image' (under BOSS this is for instance provoked by the FP command 'mode pause.yes' just before the call of the program we are going to debug).

By the FP command

```
print image 0.14 1536.1600
```

the words number 0 to 14 and 1536 to 1600 of the area are printed on current output as integers, halfwords, and code. (The words 0 to 14 contain the start address of the core area and the registers at the time of the dump).

If the area is described with contents 7 (dumped core areas should always have this contents. When BOSS makes a core dump the contents are set to 7) the output is numbered with absolute addresses (as the program was placed in core when the dump was made). One can select the part to be printed by specifying such absolute addressed: The command

```
print image 45236.45344.a
```

prints the part of the dump originating from the core addresses 45236 to 45344.

3.45.2 Call 3.45.2

$$\left\{\text{<out file> =}\right\}_{0}^{1} \quad \text{print <source>} \quad \left\{\text{<format list> <field>}\right\}_{0}^{\infty}$$

<source>::= { <bs area name> | <internal process name> | <integer> }

<format list>::= { integer | word | half | abshalf | char | octal | code | text | bits <pattern> | all | words . <words per line> } ∞ 0

<pattern>::= { . <first bit> . <last bit> } ∞ 0

<field>::= { <from - to> { .i | .c. | .r | .a } 1 0 | <from addr>.<to addr>.<from block> | <from addr>.<to addr>.<from block>.<to block> }

<from - to>::= { <from addr> | <from addr>.<to addr> }

<words per line>, <first bit>, <last bit>, , <from addr>, <to addr>, <from block>, and <to block> are integers.

3.45.3 Function

The format list is initialized to all (see below).

The parameter list is scanned and the printing source is determined. If <source> is the name of an area on the backing storage "print" prints from this area. If <source> is the name of an internal process "print" prints from core with the start address of the process as base address. If <source> is an integer the printing takes place from core with this integer as base address.

The program enters the following cycle until the end of the parameter list:

1) When a <format list> is recognized the printing format is changed accordingly.

2) When a <field> is recognized the printing is activated. The printing is done with the current format.

The output occurs on <outfile> if specified - otherwise on current output.

3.45.4 Format List

The elements of a <format list> defines how the current word of the actual field appears in the output:

integer	current word is printed as a signed integer.
word	current word is printed as a signed integer.
half	current word is printed as two signed integers, being the two halfwords of the word.
abshalf	current word is printed as two positive integers, being the two halfwords of the word.
char	current word is printed as three unsigned integers i.e. the iso values of a text is printed.

octal	current word and address is printed as octal. If code is also specified, the final address is printed as octal.
code	current word is printed as an instruction in symbolic form. If the instruction includes relative addressing, the output is supplied with the corresponding final address according to the numbering of words: final address = displacement + number of current word This final address is printed immediately after the displacement.
text	current word is printed as 3 ISO characters, non-graphic characters replaced by SP.
bits.<pattern>	current word is printed as a number of unsigned integers according to <pattern>. Denoting the bits from 0 to 23, each integer is the value of the bit group defined by <first bit> and <last bit>. The value of <pattern> is initialized to: 0.0.1.1, ... , 22.22.23.23 which causes the current word to be printed as 24 integers, being the value of each bit of the word.
words.<words per line>	determines the number of words to be printed in each line. The line is headed by an integer corresponding to the numbering of words, as explained above. The value of <words per line> is initialized to 1.
all	is equivalent to the <format list> integer bits.0.11 code.

If a <format list> consists of more elements, the current word is printed in all forms, as defined by the elements of this list. The different forms occur in a certain order in the output, according to the following sequence:

<text> <integers, halfwords, and bit patterns> <instruction>;

<integers, halfwords, and bit patterns> are printed in the same order as the corresponding elements in <format list>.

The <format list> is initialized to:

integer bits.0.11 code

which causes current word to be printed in the 3 forms:

<integer> <left-most halfword> <instruction>.

3.45.5 Field Specification 3.45.5

The limits for the printing are determined from the integers <from addr> and <to addr> by rules depending on the other part of the field specification. (A <from addr> alone means <from addr>.<from addr>.) If only <from addr> and <to addr> are in the field specification they give the limits relative to the start of the backing storage area or to the base address for the core area.

Specification of <from block> and <to block> is significant only for bs areas. The area is considered divided into segments (each on 512 bytes) and from each segment with segment number between <from block> and <to block> (<from block> alone means just this single block) the part of the segment determines by <from addr> and <to addr> is printed.

The modifier .i (indirect addressing) causes the contents of the words specified by <from addr> and <to addr> to be interpreted as absolute addresses and used as limits. The values <from addr> and <to addr> are interpreted relative to the base address. (If the source is a bs area it should have contents = 7).

The modifier .c. (indirect addressing around a center): The contents of the word with relative address (relative to the area start or the base address) is interpreted as an absolute address and taken as center for printing and the printing limits become <from addr> below the center and <to addr> above the center. (If the source is a bs area it should have contents = 7).

In case of the modifier .a (absolute addressing) the printing limits are the integers <from addr> and <to addr> taken as absolute addresses. (A bs area source should have contents = 7).

The modifier .r (relative addresses in output) belongs in a way to the format specification. It causes the absolute addresses used as numbering in the output to be replaced by relative addresses.

3.45.6 Storage Requirements 3.45.6

The core store space needed by "print" is approx. 2048 bytes plus the space needed by FP.

3.45.7 Error Messages 3.45.7

***print param <erroneous parameters>
Parameter error in call of "print". If the parameters are part of a syntax element, this has no effect.

***print numbering
The field specification attempts to define words outside area.

***print <name> area
Area process cannot be created or trouble during input data transfer.

***print connect out
Output file cannot be connected.

***<name> unknown
<name> is neither name of a catalog entry or an internal process.

***print core size
No core space for segment buffers; at most 512 halfwords more are needed.

In the first two cases "print" continues with the next parameter in the list. In the other cases "print" terminates.

3.45.8 Further Examples 3.45.8

print sin

prints the total area sin as integer bits.0.11 code

print datas integer 0.510.1

prints the second segment of datas as integers

print algol text all 10.20.0.4

prints halfwords 10 to 20 on the first 5 segments of ALGOL as text integer bits.0.11 code

print image 0.14 16.10.c.12 1594.1604 1614.1616 1598.1582.i

prints relevant parts after break of ALGOL program (cf. ref. [12]). Corrections in Running System may change these numbers in which case a correction for above manual will be issued:

0	first address
2	w0
4	w1
6	w2
8	w3
10	exeption register
12	instruction counter
14	interrupt cause

16.10.c.12 8 words before and 5 words after breakpoint

1594	UV
1596	UV
1598	lastused
1600	last of program
1602	first of program
1604	segment table base

1614 saved stack ref

1616 saved w3

1598.1582.i total stack

3.46 procsurvey 3.46

Lists types of procedures and their parameters, as well as the procedure date.

3.46.1 Example 3.46.1

The FP command:

```
     procsurvey invar in
```

will produce the output:

```
     integer procedure invar: d.760830.1405
       param 1: zone
     zone in, rs entry no.: 26
```

3.46.2 Call 3.46.2

$$\text{procsurvey}\left\{\text{<s> <name>}\right\}_{0}^{\infty}$$

3.46.3 Function 3.46.3

Each name is looked up in the catalog, if several entries with the same name exist, only the one with the smallest scope will be listed. Procedures and standard variables will be listed, as above, other entry types will cause an error message.

3.46.4 Storage Requirements 3.46.4

2500 bytes plus space for FP.

3.46.5 Error Messages 3.46.5

***procsurvey call
   Left hand side in the call.

***procsurvey <integer> param
   Integer parameter.

***procsurvey <name> unknown
The name was not found in the catalog.

***procsurvey <name> connect error
The area could not be connected.

***procsurvey <name> not procedure
The name does not describe a procedure or an ALGOL standard identifier.

***procsurvey <name> entry inconsistent
The start external list in the entry description, contains a byte > 500, i.e. the entry does not describe a legal procedure.

***procsurvey <name> code inconsistent
Illegal contents of the internal list in the code body.

In case of error, procsurvey continues after the error message.

3.47 release

Sends a release message to the parent (the operating system) releasing the specified magnetic tape reel (cf. ref. [10], section 6.1).

3.47.1 Example

If the total number of tape reels used during a job exceeds the number of stations available one has to release one of the tapes during the job in order to tell BOSS that the reel could be dismounted. The FP command

```
release mt123456
```

tells BOSS that mt123456 can be dismounted.

If 'pip' is a name of a file on the magtape e.g.

```
pip=set mto mt123456 0 7
```

the same result is obtained by the FP command

```
release pip
```

In general it is good manners to release a tape reel as soon as it is not longer required.

3.47.2 Call

release <s> <name>

3.47.3 Function

A release message is sent to the parent. The name in the message is found as follows: the name is looked up in the catalog. If an entry describing a magnetic tape file (kind=18) is found and if this entry is not protected (e.g. not of scope system) the document name in the entry is used. Otherwise the name specified is used.

3.47.4 Storage Requirements 3.47.4

1536 bytes plus space for FP.

3.47.5 Error Messages 3.47.5

***release call

Left hand side in the call of the program.

***release <parameter list> parameter error

Parameter error in the call of the program.

***release <name> tape name missing

The entry specified has a zero document name.

3.48 rename 3.48

Changes the names of catalog entries as specified.

3.48.1 Example 3.48.1

By the FP command

```
rename pip.fup
```

the name of the catalog entry named 'pip' is changed to 'fup'. The scope, entry tail, and the contents of an associated data area remain unchanged.

3.48.2 Call 3.48.2

rename $\left\{ \text{<s> <oldname> . <newname>} \right\}_{1}^{\infty}$

3.48.3 Function 3.48.3

Each <oldname> in the list is looked up in the catalog and the name of the entry found is changed to the corresponding <newname>.

Remark: if several entries with the same name are present, the catalog lookup will find the entry with the 'smallest' scope (corresponding to the order: temp, login, user, project).

3.48.4 Storage Requirements 3.48.4

1536 bytes plus space for FP.

***rename call

Left hand side in the call. The program terminates without further actions.

***rename param <parameter>

Parameter error. The remainder of the parameter list is skipped.

***rename <oldname>.<newname> name conflict

The entry could not get the name changed because an entry named <newname> already exists.

***rename <oldname>.<newname> unknown

No entry named <oldname> was found.

***rename <oldname>.<newname> protected

The job was not allowed to change the name of the entry.

***rename <newname> no room

The name overflow in the main catalog exceeded the limit.

***rename <oldname>.<newname> entry in use

The entry could not be renamed because another job was using it.

In the last four cases the program continues with the next parameter.

3.49 repeat 3.49

The program makes it possible to repeat (a specified number of times) a series of FP commands placed in brackets.

3.49.1 Example 3.49.1

By the following FP commands files number 1 to 20 on mt471100 and mt471200 are checked by the program "copy" (which outputs the number and sum of characters for each of the 40 files):

```
t1=set mto mt471100
t2=set mto mt471200
(repeat 20
nextfile t1 t2
copy t1 t2)
```

3.49.2 Call 3.49.2

$$\left\{\text{<outfile>=}\right\}_{0}^{1}\ \text{repeat <s> <total number of times> <parameter list> <NL>}\ \left(\left\{\text{<FP command>}\right\}_{0}^{\infty}\ \left\{\text{<FP command>}\right\}_{1}^{\infty}\ \right)$$

<total number of times>::= an integer greater than 0
<parameter list>::= any sequence obeying the FP syntax

3.49.3 Function 3.49.3

The program augments the command stack so that the rest of the compound command containing the call of repeat, will be executed the specified number of times.

<outfile> and <parameter list> have no effect at all, but in mode list.yes they may be used to identify the repeat call to be executed.

3.49.4 Storage Requirements 3.49.4

512 bytes plus space for FP.

3.49.5 Error Messages 3.49.5

***repeat no core

There is no room in the process area for the augmentations of the command stack made by repeat (the command to be repeated must be exceptionally long).

***repeat no factor

Either there are no right hand parameters to the call or the first right hand parameter is not an integer.

***repeat factor 0

The integer <total number of times> is equal to 0.

***repeat nothing to repeat

The call of repeat is the last command in the compound command containing repeat.

In case of error messages, the commands following the repeat call will be executed once.

3.50 replace

Sends a replace message to the parent (the operating system) defining a file as replacement for the current job file. After termination of the job BOSS will create a new job with the same name and specified file as job file. BOSS only accepts replace messages from off line jobs, not from on line jobs.

3.50.1 Example

The FP command

```
replace pip
```

defines the file 'pip' as replacement for the job file.

The FP command

```
replace pip newid
```

defines the file 'pip' as replacement for the job and the identification to be changed according to the job head in the file 'pip'.

3.50.2 Call

replace <s> <job file> <s> $\left\{ \begin{matrix} \text{oldid} \\ \text{newid} \end{matrix} \right\}_{0}^{1}$

where <job file> is a name of a permanent bs file. oldid is standard.

3.50.3 Function

A replace message containing the specified name is sent to the parent.

3.50.4 Storage Requirements 3.50.4

1536 bytes plus space for FP.

3.50.5 Error Messages 3.50.5

***replace call

Left hand side in the call of the program.

***replace <parameter list> parameter error

Parameter error in the call of the program.

***replace <parameter list> not allowed from on line job

The replace message was not accepted as the job is an on line job.

In case of any error no replace message is sent.

3.51 ring

Sends a 'mount ring' message to the parent (the operating system). The program is normally not used as the software sends the mount ring message automatically when needed.

3.51.1 Call

ring <s> <name>

3.51.2 Function

A 'mount ring' message is sent to the parent. The name in the message is found as follows: the name is looked up in the catalog. If an entry describing a magnetic tape file (kind=18) is found and if this entry is not protected (e.g. not of scope system) the document name in the entry is used. Otherwise the name specified is used.

3.51.3 Storage Requirements

1536 bytes plus space for FP.

3.51.4 Error Messages

***ring call
Left hand side in the call of the program.

***ring <parameter list> parameter error
Parameter error in the call of the program.

***ring <name> tape name missing.
The entry specified has a zero document name.

In case of any error no mount ring message is sent.

3.52 rubout 3.52

Rubs out the contents of the specified backing storage files. If demanded the catalog entry is removed after the cleaning.

3.52.1 Examples 3.52.1

By the FP command.

```
runout user clear.yes text4
```

the file text4 is filled with a fillpattern after which the entry is cleared.

The following two FP commands

```
runout user text4
runout user clear.no text4
```

are identical, since the clear parameter is initialized to no. The entry is not removed.

3.52.2 Call 3.52.2

```
rubout <s> <scope> [ <s> [ <name>          ] ]∞
                   [       clear. [ yes ]  ]
                                  [ no  ]  ]0
```

```
<scope>::= [ temp    ]
           [ login   ]
           [ user    ]
           [ project ]
           [ own     ]
```

3.52.3 Function 3.52.3

The files are filled with a fillpattern after which it is cleared in case the value of the parameter is yes. The fillpattern is a

consisting of 3 NUL characters and 3 EM characters. Scope own means all of temp, login, user, and project.

3.52.4 Storage Requirements 3.52.4

1536 bytes plus space for FP.

3.52.5 Error Messages 3.52.5

***rubout call
The program was called with a left hand side.
No file rubout.

***rubout param <parameter>
Illegal parameter.
The rest of the parameter list is skipped.

***rubout <scope> illegal scope
The scope was illegal.
No file rubout.

***rubout <scope> <name> unknown
The entry was not found.
The program continues with the next parameter in the list.

***rubout <scope> <name> entry in use
The entry was not changed or removed because another job was using it.
The program continues with the next parameter in the list.

***rubout <scope> <name> not bs area
The entry did not describe a backing storage area.
The program continues with the next parameter in the list.

***rubout <scope> <name> catalog error
Catalog, monitor, or hard error.
The rest of the parameter list is skipped.

3.53 save 3.53

The program transfers catalog entries and backing storage areas to magnetic tape for backup purpose.

The catalog entries and backing storage areas are transferred back to disc by the program load.

3.53.1 Examples 3.53.1

3.53.1.1 Example 1 3.53.1.1

All catalog entries and backing storage areas of scope temp are transferred to the magnetic tape mtdp0001 file 2 by the call:

```
save mtdp0001.2
```

In case

```
t = set mto mtdp0001 0 2
```

the same is obtained by the call:

```
save t.0
```

3.53.1.2 Example 2 3.53.1.2

All catalog entries and corresponding backing storage areas specified in the file 'savefiles' are transferred to file number 2, overwriting the ones saved in example 1, by the call:

```
save mtdp0001.2 in.savefiles
```

3.53.1.3 Example 3 3.53.1.3

All catalog entries with corresponding backing storage areas of scope project, those of scope user on the disc named 'disc3' and finally the best entry with the name 'pap' are saved after the ones in example 2 by the call:

```
save mtdp0001.last scope.project,
     disc.disc3 scope.user,
     pap
```

3.53.2 Call

3.53.2

```
{<outfile> =}(1,0) ,
save
{{<mount param>}(1,0) <tape param>}(2,1) ,
{<special param>}(1,0) ,
{<save specifier>}(1,0)
```

3.53.2.1 outfile

3.53.2.1

```
<outfile>      ::=   name of any filedescriptor
```

3.53.2.2 mountparam

3.53.2.2

```
<mount param> ::= [ mountspec. <device no>
                    release. {yes/no}
                    <modekind>              ]*1

<device no> ::=     <integer>

<modekind> ::=  [ mto
                  mte
                  nrz
                  nrze ]
```

3.53.2.3 Tape param

<tape param> ::= <tape name>.<file no> [.<next volume>]$_{0}^{31}$,
[.label.<fpname>]$_{0}^{1}$

<tape name> ::=

<next volume> ::= [<name>/<filedescriptor>]

<name> ::= name of magnetic tape

<file descriptor> ::= name of magnetic tape file descritor

<file no> ::= [<integer>/last]

<fpname> ::= name obeying fp syntax

3.53.2.4 Special param

<special param> ::= [segm. <integer>
list.[<yes/no/names>]
reserve.[<yes/no>]]$_{1}^{*}$

3.53.2.5 Save specifier

<save specifier> ::= [<modifier>
<disc specifier>
<entry specifier>]$_{1}^{*}$

<modifier> ::= [changedisc [. <from disc>.<to disc>]$_{1}^{*}$
newscope. <new scope>]$_{1}^{*}$

```
<from disc>       ::= {all/maincatdisc/<disc name>}
<to disc>         ::= { no/maincatdisc/<disc name>}
<new scope>       ::= { no/temp/login/user/project}
<disc specifier>  ::= disc [.<from disc>]*1
<entry specifier> ::= <entry factor> [.<entry factor>]*0
<entry factor>    ::= {<entry name>/scope.<scope>/docname.<docname>}
<entry name>      ::= <name>
<scope>           ::= {temp/login/user/project/own/system/perm/all}
```

3.53.2.6 Infile parameter 3.53.2.6

Everywhere the delimiter <s> is allowed in the parameter list according to syntax for FP commands (ref. [8]), the parameter pair <s> in.<file> is allowed and will be syntactically equivalent to <s>.

<file> ::= name of any filedescriptor.

3.53.3 Function 3.53.3

The program will save the catalog entries and possible backing storage areas specified by <disc specifier> and <entry specifier> with the modifications in <modifier> on the magnetic tapes specified in <tape param>, i.e. maybe in two copies and maybe extensions over more volumes.

The program interprets the parameter groups, one by one.

The tape is mounted according to possible <mount param> and positioned to the file number(s) specified.

A version dumplabel record (cf. below) is output as the first block and displayed on current output.

Each <entry specifier> starts a catalog scan, picking out the entries specified, and the entries satisfying the current disc specifications are saved after a possible change according to the actual state of modifiers.

During the save, succeeding magnetic tape volumes, as far as specified, are mounted whenever actual volume is filled up.

The last block in each volume will be a continue record, specifying the number of entries and segments saved so far and the name of the next volume. The first block of the next volume will be a continuation dump label record, which is displayed on current output as well.

When the parameter list is emptied, the magnetic tape file is closed with an end record as the last block, specifying the amount of entries and segments saved.

The tape is positioned to the next file, an empty dump label record is output in the file and displayed on current output, the file is closed and the tape released if so specified in <mount param>.

If two sets of tapes are specified they are treated in parallel, except for different <mount param> and except for volume change, which allows for tapes of different lengths in the two sets.

3.53.3.1 Function, outfile parameter 3.53.3.1

<outfile> ::= name of any file descriptor

Current output zone is stacked and connected to the file specified at program start.

If the program terminates through its final end, the current output zone is unstacked again.

If the program terminates with a runtime alarm, the current output zone remains connected to <outfile>.

3.53.3.2 Function, mount param 3.53.3.2

<mount param> ::= { mountspec .<device no> / <modekind> / release. { yes/no } } *1

<mountspec. <device no>
A mount special parent message with the device number and proper tape name will be sent each time a new volume in the set of tapes specified in <tape param> is mounted.
Default: mountspec.0 meaning no parent message.

<modekind>
The modekind specified will be used for all volumes in the set specified in <tape param>.
Default: mto

release.{yes/no}
If release.yes the tape in the set actually used at program termination will be released
Default: release.yes

The mount parameters may be repeated, but the last one will stand.

3.53.3.3 Function, tape param 3.53.3.3

<tape param> ::= <tape name>.<file no> {.<next volume>}$_{0}^{31}$,
{.label.<fpname>}$_{0}^{1}$

One tape parameter specifies a set of magnetic tapes, consisting of one to thirtytwo tapes.

<tape name> ::= <next volume> ::= <name>/<file descriptor>
The name of the tape. If <file descriptor> is used, the name and modekind are taken from the descriptor.

<file no> ::= {<integer>/last}
The file number of the first tape in the set where the save shall start.

The save will start in file number one in all succeeding volumes.

If the first tape is specified by <file descriptor>, its file count will be added to <file no>.

If <file no> = last, the first file, which does not start with a version or a continue dump label, is searched along the tapes in the set, even if they are specified by <file descriptor>'s.

label.<fpname>
If a label is specified, the name will be written as the last field in the version dumplabel record, and it will appear on current output.

3.53.3.4 Function, special param

<special param> ::= { segm.<integer> | list.{yes/name/no} | reserve.{yes/no} }* 1

The special parameters may be repeated, but the last one will stand.

segm. <integer>
Backing storage areas will be saved in magnetic tape blocks of <integer> segments, 1<=<integer><=9.
Default: the value found in the file count of the programs own catalog entry tail. If the value is outside the legal value interval, the value one is used.

list.{yes/name/no}
If list.yes, the entries saved are listed on current output in the form:

<name><modekind> <scope>/<permkey><docname> {<base>}0 1 {<shortclock>}0 1

If list.name, only the entry name is listed.
If list.no, the entries are not listed.
Default: list.yes

reserve.{yes/no}
If reserve.yes is specified, the program will try to reserve the area process for an area entry to be saved, unless the monitor offers write protection, cf. 3.53.4.
If reserve.no is specified, no reservation is tried.
If the monitor offers write protection, the parameter has no function.
Default: reserve.yes

3.53.3.5 Function, save specifier 3.53.3.5

$$\text{<save specifier>} ::= \left\{ \begin{array}{l} \text{<modifier>} \\ \text{<disc specifier>} \\ \text{<entry specifier>} \end{array} \right\}^{*}_{1}$$

The elements of the save specifier are treated one by one, until the parameter list is emptied.

A modifier will modify the state of current modifiers.

A disc specifier will cancel current disc specifiers and define a new set of disc specifiers.

An entry specifier will start a catalog scan, picking out entries specified.

Entries picked out which belongs to a disc specified in current disc specifiers will be saved after having been modified according to current modifiers.

If no <entry specifier> is found in the parameter list after a modifier or a disc specifier or no save specifier is found, a default entry specifier will be used.

<modifier> ::= [changedisc { .<from disc>. <to disc> }*1 | newscope. <newscope>]*1

A modifier is valid until changed by another modifier.

changedisc { .<from disc>.<to disc> }*1

An entry belonging to <from disc> will be changed as if it belongs to <to disc>.

<from disc> ::= {all/maincatdisc/<disc name>}

all	means all discs
maincatdisc	means the disc with the main catalog
<disc name>	means the disc with that name

<to disc>::= {no/maincatdisc/<disc name>}

no	means changedisc. <from disc>.<from disc>
others	as for <from disc>

Default, changedisc.all.no

newscope. <new scope>
All entries will be changed to have the scope <newscope>
<newscope> ::= {temp/login/user/project/no}

If the entries are saved using a scope key (cf. specifier 'scope') they will be saved with the one denoting <newscope>.

If they are saved using bases they will be saved with permkey and bases corresponding to <newscope>.
<newscope> = no means no change of scope.
Default: newscope.no

<disc specifier> ::= disc { .<from disc> }*1

A disc specifier is valid until the next disc specifier, which will cancel it.

<from disc> ::= {all/maincatdisc/<disc name>}

The parameters have the same meaning as for 'changedisc'.
Default: disc.all

<entry specifier> ::= <entry factor> {.<entry factor>}* 0

An entry specifier is composed of one or more entry factors of which three kinds exist, each of which specify an entry attribute.

If one kind of entry factor is repeated, the last one stands, except for the factor <name>, where a warning will be given and the first name stands.

The entry specifier specifies a set of entries, each of which have all the attributes specified

<entry factor> ::= {<name>/scope.<scope>/docname.<docname>}

<name>
The attribute is the entry name.

All entries visible with this name have the attribute. The entry names 'c' 'v' and primout cannot be specified.

Entries of name 'c' or 'v' with permkey = 0 and entries with name 'primout' and permkey = 2 are considered to have no name attribute.

scope. <scope>
<scope> ::= {temp/login/user/project/own/system/perm/all}

The attribute is the scope.
All entries with the scope specified have the attribute.
temp, login, user, project have the usual meaning
own means one of above.

system means permkey = 3 and entry base = system base
perm means permkey = 3 and entry base inside or equal to the standard base of the process
all means any permkey and entry base inside or equal to the standard base of the process.

Entries specified by the attribute scope.perm or scope.all will be saved using permanent key and entry base. Entries specified by any other scope will be saved using a key denoting the scope, making them loadable in any process (scope.system in a process with maxbase = system base, though).

cf. the modifier 'newscope'.
Default: name specified: the best name
no name specified: temp

docname. <docname>
The attribute is the document name of the entry. All entries visible with a document name equal to <docname> have the attribute.
Changes the default for scope to <any visible scope>.

The best name means the name with the best scope among the scopes temp, login, user and project or any scope visible changed into one of above by the modifier 'newscope'.

Default entry specifier: scope. <temp>

3.53.3.6 Function, infile parameter 3.53.3.6

Every where the delimiter <s> is syntactically correct in the parameter list, the parameter pair <s>in.<filename> is allowed and syntacitally equivalent to <s>.

<filename> ::= name of any file descriptor

When <s> in.<filename> is met in the parameter list, current input zone is stacked and connected to the file specified by the

file descriptor, and the parameter reading is continued in current input zone.

The parameter reading takes place using the special fp input alphabet and according to normal fp parameter syntax (cf. Ref [9]), except the character 'nl' is equivalent to the character 'sp'.

When the separator 'em' is met, current input zone is unstacked and parameter reading continues from current input zone, except when unstacked to the initial level, in which case parameter reading continues in the fp command stack.

In case of illegal character or fp syntax error a syntax alarm is written on current output zone, current input zone stack chain is emptied, listing the chain on current output zone, and parameter reading continues in fp command stack.

The fp command listing governed by the mode bit 'list' will list the parameters in the fp command stack, not the parameters in any file.

3.53.3.7 Alternative parameter names

3.53.3.7

For compatibility reasons, some alternative parameter names are allowed.

The modifier 'changekit' is allowed and is equivalent to 'changedisc'.

The changedisc parameter <from disc> = main is allowed and equivalent to <from disc> = all.

The changedisc parameter <to disc> = main is allowed and is equivalent to <to disc> = maincatdisc.

The disc specifier 'kit' is allowed and is equivalent to the disc specifier 'disc'.

The disc specifier parameter <from disc> = main is allowed and is equivalent to <from disc> = all.

3.53.4 Entries and backing storage areas

Catalog entries are picked out of the main catalog according to <entry specifiers>, checked with <disc specifier>, changed according to <modifier> and transferred to a record, which is output on the tape as one block (cf. 3.53.5, Tape Format).

If, however, the entry is an area entry, steps are taken to protect the area during the save.

Before the entry is changed, the catalog base of the process is changed to equal the entry base. If, however, the entry base is outside the max base of the process, to equal the max base.

An area process with the name of the entry is created. If the creation fails, the entry is skipped with a warning on current output, telling the reason of the failure.

Next, the area process is write protected, unless its base is outside the max base of the process.

If the current monitor does not offer write protection, the process is reserved instead, unless reserve.no is specified.

If the protection fails because the area process is already reserved by another process, the entry is skipped with a warning on current output.

If the base of the area process created does not equal the entry base, i.e. the entry base is outside max base and a better entry exists, the entry is skipped with a warning on current output, telling that the entry is covered by a better entry.

At last, the write access counter and the name table address of the area process are read and the catalog base of the executing process is reset.

Now the entry is changed and saved, followed by the segments of the backing storage area.

When the area has been saved, the write access counter of the area process is compared with its value before the area was saved. If the value has changed, i.e. the base of the area process is outside the max base of the existing process (it was not protected) and another process has had write access to it or the executing process itself has had write access to it (area connected to current output zone), the entry is listed on current output followed by a warning, that the area has been changed during the save.

Finally, the number of segments saved is compared to the size of the area in the entry tail, and if not equal, the entry is listed on current output followed by a warning, that the area and the entry are inconsistent.

The area process is removed, unless the area is the program itself.

Foreign read accesses to the area will not influence the save.

If the current monitor does not offer write protection, foreign write accesses will only hold up the save if reserve.no is specified or the entry base is outside max base of executing process (scope.system).

If the current monitor offers write protection, foreign read accesses will not be affected by the save.

If the current monitor does not offer write protection, foreign read accesses will be held up by the save, unless reserve.no is specified or the entry base is outside max base.

Foreign write access during the save will be rejected unless reserve.no is specified or the entry base is outside max base of executing process.

Using one of the entry specifiers scope.perm or scope.all, which are intended for system back-up, entries with a base equal to or inside the standard base of the executing process are specified.

Areas saved this way (as well as areas saved using temp, login, user, project or own) will never be changed by other processes during the save. Either they are protected during the save or they are skipped because they are reserved by another process at the time of reservation.

3.53.5 Tape Format 3.53.5

The magnetic tape file created by save starts with a block containing a
- version dump label record

followed by a number of blocks each containing
- an entry record or
- a segment record

ending with a block containing
- an end record.

The file may extend over more tapes, in which case each tape ends with a block containing
- a continue record

and each new volume tape starts with a block containing
- a continuation dump label record.

After the file, save creates another file with only one block containing
- an empty dump label record.

The segment records are 8+segm*512 halfwords long, all other records are 100 halfwords long.

The format and contents of the records are as follows.

Dump label records:

version-, continuation- and empty label records contain a number of characters, terminated by an em-character (the records may be read by edit or copy):

hw addr : contents

```
 0 -  3 : <:dump  :>
 4 - 11 : tapename followed by a number of spaces
12 - 15 : file number followed by a number of spaces
16 - 19 : <:vers. :>,<:cont. :> or <:empty :>
20 - 23 : <<ddmmyy>, date,
24 - 27 : <<.hh   >, hour,
28 - 31 : <:s=<segments per block>:>
32 - 51 : <: save sw8010/1 release :>, <<dd.d>, release,<:   :>
52 - 55 : <:<10><0><0><0><0><25>:>
56 - 99 : null characters
```

Entry record:

An entry record contains parts of a catalog entry head and all of the tail:

hwd addr: contens

```
 0 -  3 : 1, if saved using bases then 52 else 48
 4 -  7 : entry count, no of segments
 8 - 15 : entry name
16 - 35 : entry tail
36 - 39 : if saved using bases then permanent key else scope key
          (temp=8, login=7, user=6, project=5, system=3)
40 - 47 : disc name
48 - 51 : if saved using bases then
          (entry base lower, entry base upper)
          else
          1,48
52 - 99 : if saved using bases then (1,52) else (1,48),...
```

Segment record:

A segment record contains some saved segments:

```
hwd addr : contents
  0 -  3 : 2, 8+512* number of segments in record
  4 -  7 : entry count, segment count
  8 -519 : contents of one segment
520-1031 : contents of one segment
    .
    .
    .
```

End record:

An end record contains the entry and segment counts:

```
hwd addr: contents
 0 - 3  : 3,8
 4 - 7  : entry count, segment count
 8 -99  : 3,8,...
```

Continue record:

A continue record contains the name of the continuation tape:

```
hwd addr: contents
 0 - 3  : 4, 16
 4 - 7  : entry count, segment count
 8 -23  : name of the continuation tape
 24-99  : 4, 16, ...
```

3.53.6 Requirements

3.53.6

The program will allocate memory space to internal data buffers of at least 2*segm*512 halfwords.

If that does not leave space for two program segments, the program will terminate with a stack alarm, in which case the size of the process must be increased.

If free memory space at time for buffer allocation leaves space for more than enough program segments to avoid program paging in inner loops, the memory surplus will be allocated to databuffers, giving maybe double buffered tape zone and all the rest to extend a single buffered disc zone.

The program needs
4 area processes (fp, save, catalog, area to be saved) +
1 area process if current output zone is connected to a backing storage area +
1 area process as long as current input zone is connected to a backing storage area for parameter reading.

If the program fails to have the necessary number of area processes, it will
- write a parameter alarm and continue writing or reading in the present file, if the resource was missing at time for zone connection
- skip area entries with a warning (cf. 3.53.7).

The program may need temporary entries and segments to perform zone stacking in case of parameter input from backing storage area. In case of shortage of these resources, the program will terminate with a 'break 10' alarm.

3.53.7 Error messages

Error messages from the program are written in current output zone.

The following kinds of error messages exist:
- parameter alarm
- parameter warning
- save specifier warning
- area entry warning
- parameter input syntax error message

3.53.7.1 Parameter alarm

At parameter alarm, the parameter list is emptied, listing the parameters from current parameter and on in the alarm message.

The modebits are set: 'warning yes, ok.no'.
Since the parameter list is emptied, the program terminates.

*** save alarm <text> <parameter list>

Text:	Explanation:
mountspec param syntax	mount param not followed by .<integer>
tape param too many volumes	tape param specifies more than 32 volumes
label param syntax	label not followed by .<name>
tape param missing	no tape parameter found
segm param syntax	segm not followed by .<integer>

3.53.7.2 Parameter warning

The parameter warning skips current parameter, displaying it in the error message and continues, setting the modebits:
'warning yes, ok.yes'

*** save warning <text><current parameter>

Text:	Explanation:
outfile param connect impossible <cause>	The current output zone could not be connected to the file specified for the reason explained in <cause>. The stacked output zone is unstacked again and output continues.
infile param connect impossible <cause>	The current input zone could not be connected to the file specified for the reason explained in <cause>.

Text:	Explanation:
	The stacked input zone is unstacked again and input continues, maybe from fp command stack.
mountspec param syntax	The parameter 'mountspec' was not followed by .<integer>. The value remains the latest read or default.
release param syntax	The parameter 'release' was not followed by .yes or .no. The value remains the latest read or default.
list param unknown	The parameter 'list' was not followed by .yes, .no or .name. The value remains the latest read or default.
reserve param unknown	The parameter 'reserve' was not followed by .yes or .no. The value remains the latest read or default.
changedisc param syntax	The parameter 'changedisc .<from disc>' was not followed by .<name> The value becomes <to disc> = no.
newscope param syntax	The parameter 'newscope' was not followed by .<name> The value is not changed.
newscope param unknown	The parameter to newscope was neither of temp/login/user/project/no The value is not changed.

Text:	Explanation:
disc spec param unknown	The disc specified in disc specifier was unknown. Previous disc specifier is cancelled, all other discs specified in this disc specifier become specified.
scope param syntax	The scope parameter was not followed by .<name> No entries will be saved according to this entry specifier.
scope param unknown	The scope specified was neither of temp/login/user/project/own/system/perm/all. No entries will be saved according to this entry specifier.
docname param syntax	The 'docname' parameter was not followed by .<name> No entries will be saved according to this entry specifier.
name illegal	The name specified in entry specifier was 'c', 'v' or 'primout'. The entry is not saved.
name double defined	A name was already specified. The new name is ignored and the program continues with the current entry specifier.
save spec param unknown	Syntactically the parameter would start a save specifier, but the parameter is not <s><name>. The rest of the parameter list is read, each parameter with this warning as a result.

3.53.7.3 Save specifier warning

The save specifier is listed in the error message, the mode bits are set 'warning.yes, ok.yes' and the program continues.

*** save no entries found/saved according to following specifier
disc: <disc specifier>
entry: <entry specifier>

Explanation: no entries are found in the main catalog according to the save specifier or the entries specified have been skipped, cf. below.

3.53.7.4 Area entry warning

The warning appears in current output following the entry concerned.

The modebits are set 'warning.yes, ok.yes' and the program continues.

<entry>
*** warning: area changed during save

Explanation: Some other process has had write access to the backing storage area during the save (or the area is connected to current output zone of the program itself and own process has had write access during the save).

The entry and the area have been saved.

The warning appears no matter what the list parameter specifies.

<entry>
*** warning: area and entry inconsistent, area length <integer> segments

Explanation: the number of segments actually saved does not equal the size in the catalog entry tail.

The entry and the area have been saved.

The warning appears no matter what the list parameter specifies.

<entry>
*** warning: entry skipped <cause>

Explanation: the area process could not be created, not be protected/reserved or the area was inaccessible for the reason stated in <cause>.

The entry and the area have not been saved.

The warning appears only if list.yes or list.name is specified.

Cause:	Reason:
area claims exceeded	create area process failed
catalog i/o error, state of document does not permit call	create area process failed
entry not found	create area process failed
entry not an area entry	create area process failed
name format illegal	create area process failed
reserved by another process	set write protect/reserve failed
process does not exist, process is not user of area	set write protect/reserve failed process
covered by a better entry	area is inaccessible from executing process

3.53.7.5 Parameter input syntax error message

3.53.7.5

This message is caused by a syntax error in the parameter list read from a file connected to current input zone.

The parameter list must follow the syntax of an fp parameter list and must be coded in the special fp input alphabet, cf. Ref. [9],

with the one exception that the character 'NL' is equivalent to the character 'SP'.

The current parameter is listed in the error message and the zone stack chain is emptied, listing the chain on current output the same way fp does in an fp syntax error message.

The parameter reading is continued in the fp command stack.

The modebits are left unchanged.

```
*** save syntax <parameter>
  * read from <file>
  * selected from <file>
  .
  .
  .
*** save reinitialized
```

3.53.8 Futher examples 3.53.8

3.53.8.1 Example 1 3.53.8.1

The file pip of scope user and all files with documentname pip and scope user are saved on mtdp0001 file 1 by the call:
save mtdp0001.1 pip.scope.user docname.pip.scope.user

3.53.8.2 Example 2 3.53.8.2

All files of scope temp, login, user or project on the discs: disc, disc1 and disc2 are saved changing their disc names:
disc becomes disc3
disc1 becomes disc2
disc2 becomes disc1
by the call:
save mtdp0001.1,

changedisc.disc.disc3.disc1.disc2.disc2.disc1,
disc.disc.disc1.disc2.scope.own

3.53.8.3 Example 3

All files in the main catalog, except
- the main catalog itself
- the auxiliary catalogs
- files of name c or v with permkey = 0
- files of name primout with permkey = 2
- files belonging to other discs than disc, disc1 and disc2

are saved, disc by disc by the call:

```
save in. magtapes,
disc.disc  scope.all,
disc.disc1 scope.all,
disc.disc2 scope.all
```

provided the executing process has standard base = system base and the file 'magtapes' contain a tape parameter, e.g.

```
mtdp0001.1.mtdp0002.mtdp0003.mtdp0004.mtdp0005.
mtdp0006.1.mtdp0007.mtdp0008.mtdp0009.mtdp0010.
```

The files are saved in two copies, each copy containing at most 5 volumes.

3.54 scope

Changes the scope of catalog entries as specified in the call of the program.

3.54.1 Example

By the FP command

```
scope user pip
```

the scope of the catalog entry named 'pip' is changed to 'user'. The catalog entry is now a permanent entry and is not removed when the job terminates.

3.54.2 Call

scope <s> <scope spec> $\left\{ \text{<s> <name>} \right\}_{1}^{\infty}$

<scope spec>:= <scope> $\left\{ \text{. <device name>} \right\}_{0}^{1}$

<scope>::= $\left\{ \begin{array}{l} \text{temp} \\ \text{login} \\ \text{user} \\ \text{project} \end{array} \right\}$

<device name::= <name of drum or disc kit>

3.54.3 Function

The scope specification is interpreted and then the name list is scanned. For each name a catalog lookup is made and the scope of the entry found is changed to the specified scope. The entry may hereby replace a catalog entry with the same name (this 'old' entry is removed from the catalog).

Remark: if several entries with the same name are present, the catalog lookup will find the entry with the 'smallest' scope (corresponding to the order: temp, login, user, project).

3.54.4 Scope Specification 3.54.4

The concept of scope of a catalog entry is explained in the ref. [10], section 4.1.

A device name in the scope specifications means that the catalog entry should be permanented into the auxiliary catalog on the device mentioned and thereby coupy permanent claims on the device mentioned, but not in the main catalog.

This is meaningful for the scopes user and project only and the entry should be either a non-area entry or an area where the data area is situated on the specified bs device.

3.54.5 Storage Requirements 3.54.5

2048 bytes plus space for FP.

3.54.6 Error Messages 3.54.6

***scope call
Left hand side in the call. The program is terminated without further actions.

***scope <scope spec> illegal scope
The scope specification is illegal.

***scope <scope spec> bs device unknown
The bs device specified in the scope specification is not on the computer.

In all cases above the program terminates without changing the scope of any catalog entry.

***scope param <parameter>
Parameter error in the call. The rest of the name list is skipped.

***scope <scope spec> <name> unknown
No entry with the given name was found. *)

***scope <scope spec> <name> protected
The job was not allowed to change the scope of the entry found. *)

***scope <scope spec> <name> entry in use
Another job was using the entry and hence the scope could not be changed. *)

***scope <scope spec> <name> no resources
The resources of the job did not allow the change of the entry scope. *)

***scope <scope spec> <name> change bs device impossible
The entry could not be permanented into the specified auxiliary catalog. *)

***scope <scope spec> <name> catalog error
Catalog error, monitor error, or hardware error.

*) The program continues with the next name in the name list.

3.55 search 3.55

Finds and lists all catalog entries with a given scope.

3.55.1 Example 3.55.1

By the FP command

search user

one gets a list on current output of all catalog entries which have scope user under the actual job.

By the FP command

search own

one gets all entries with scope temp, login, user, or project listed on current output.

3.55.2 Call 3.55.2

$$\left\{\text{<out file> =}\right\}_{0}^{1} \text{search <s> <scope spec>}$$

$$\text{<scope spec>::= <scope> } \left\{\text{. <device name>}\right\}_{0}^{1}$$

$$\text{<scope>::=} \left\{\begin{array}{l}\text{temp}\\ \text{login}\\ \text{user}\\ \text{project}\\ \text{system}\\ \text{own}\end{array}\right\}$$

<device name>::= <name of drum or disc kit>

3.55.3 Function

The catalog is scanned and all catalog entries with the specified scope are listed. If an <out file> is present it is used for the output. Otherwise current output is used.

3.55.4 Scope Specification

The scope concept is explained in ref. [10], section 4.1. The scope own means belonging to the project and available for the job i.e. all of temp, login, user, or project (cf. the example above). If a device is specified, only area entries where the data area is on this device and non-area entries which are in the auxiliary catalog on the device are listed.

3.55.5 Output Format

Each entry found is listed exactly as described under the program "lookup".

3.55.6 Storage Requirements

2560 bytes plus space for FP.

3.55.7 Error Messages

***search connect <outfile>

The output file could not be connected; current output is used instead.

***search param <parameter>

Parameter error in the call. No entries listed.

***search <scope spec> no entries found

No entries with the specified scope (and specified device) was found.

***search <scope spec> illegal scope

Incorrect scope specification. No entries listed.

3.56 set

Creates a new catalog entry with scope temp or changes an already existing entry (with scope temp) according to the parameters.

3.56.1 Example

An area named 'pip' with an area size of 20 segments on the bs device 'disc3' and with data now, is created by the FP command:

```
pip=set 20 disc3
```

(Actually the area may get a slightly larger size because the size is always a multiple of the slice length on the device, cf. ref. 3).

A non-area entry 'file7' which may serve as file descriptor for file 7 on the magtape with name 'mt314711' is created by the FP command

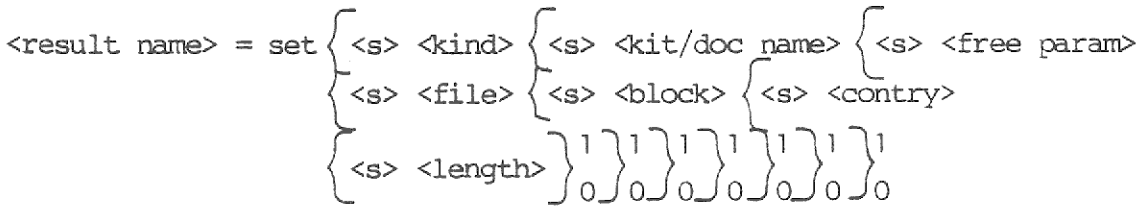
```
file7=set mto mt314711 0 7
```

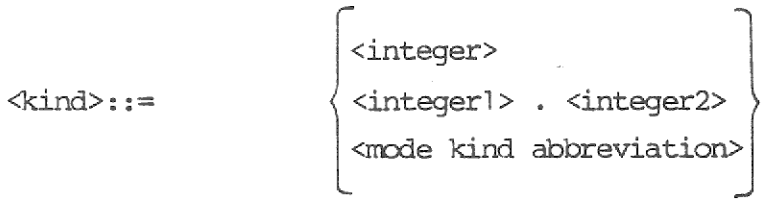
An area named 'image' on disc (intended for core store dump) is created by the FP command

```
image=set 40 1 0 0 0 7.0
```

(The parameters 0 0 0 7.0 may be omitted as BOSS will automatically set contents 7 when the dump is made).

3.56.2 Call





<kit/doc>::= [<name> | 0 | 1]

<free param>::= [<integer> | <integer1> . <integer2> | d. <isodate> | d. <isodate> . <clock>]

<isodate>::= [yymmdd | 0] 0 is interpreted as now

<clock>::= <hhmm>

[<file> | <block> | <contry> | <length>] ::= [<integer | <integer1> . <integer2>]

3.56.3 Function 3.56.3

The parameters are interpreted as described below yielding the wanted entry tail. Next, creation of the catalog entry <result name> with this tail is attempted. If the result is 'entry already exists' (cf. ref. 2 and 3) the existing entry is changed to get the entry tail wanted.

Each element in the entry tail except <kit/doc name> is a 24 bits word.

1. <integer> : the integer is placed in the tail
2. <integer1> . <integer2> : is interpreted as two bytes i.e. as the binary number <integer1> shift 12 + <integer2>
3. <mode kind abbreviation> : only relevant for <kind>. The table of mode kind abbreviations is scanned and the value is used.

4. <name> : only relevant for <kit/doc name>. The name is placed in the tail.

If the parameter list does not specify all of the tail, the rest is set to zero.

When an area entry is created, the bs device is determined by <kit/doc name>:

If <kit/doc name> is 0, the area is, if possible, created on a drum, otherwise on a disc.

If <kit/doc name> is 1, the area is, if possible, created on a disc, otherwise on a drum.

If <kit/doc name> is a name, the area is created on the bs device with this name.

3.56.4 Storage Requirements 3.56.4

1536 bytes plus space for FP.

3.56.5 Error Messages 3.56.5

***set call

No left hand side in the call.

***set param <parameter>

Parameter error in the call.

***set <result name> change kind impossible

Change of an area entry to a non-area entry or vice versa was attempted.

***set <result name> change bs device impossible

A change of <kit/doc name> on an area entry was attempted.

***set <result name> bs device unknown

The bs device specified was not on the computer.

***set <result name> no resources

The resources of the job did not allow the wanted creation of a catalog entry.

***set <result name> no room

The name overflow in the main catalog exceeded the limit.

***set <result name> entry in use

The entry could not be changed because another job was using it.

If any error message appears, no entry is created or changed.

3.57 setmt 3.57

Creates catalog entries of scope temp describing files on magnetic tape according to the parameters.

3.57.1 Examples 3.57.1

The FP command

```
pap=setmt mt004711.3
```

creates the same catalog entries as the FP commands

```
pap1=set mto mt004711 d.0 1
pap2=set mto mt004711 d.0 2
pap3=set mto mt004711 d.0 3
```

If t is a name of a file on this magtape, e.g.

```
t=set mto mt004711
```

the same result is obtained by

```
pap=setmt t.3
```

The FP command

```
f=set nrz mt004711 0 2
f=setmt f.3.5
```

creates the same catalog entries as the FP commands

```
f3=set nrz mt004711 d.0 5
f4=set nrz mt004711 d.0 6
f5=set nrz mt004711 d.0 7
```

3.57.2 Call 3.57.2

<resulting name> = setmt <mtname>. { <upper integer | <lower integer>.<upper integer> }

If no <lower integer> is specified, it is set to 1.

3.57.3 Function

Entries describing files on the magnetic tape <mtname> are created with names <resultname> followed by <lower integer> to <upper integer>. If a temporary entry already exists, it is first moved.

The mtname is looked up in the catalog. If an entry describing a magnetic tape file (kind=18) is found, the document name and the modekind of this entry is used and the file will be addressed relative to the file number. Otherwise the name specified is used.

3.57.4 Storage Requirements

512 bytes plus space for FP.

3.57.5 Error Messages

***setmt call
No left hand side or left hand side of more than 9 characters.

***setmt param
Parameter error in the call, e.g. an integer greater than 99.

***setmt <resultname> no resources
The resources of the job did not allow creation of the catalog entry.

***setmt <resultname> no room
The name overflow in the main catalog exceeded the limit.

***setmt <resultname> catalog error
Error in catalog, monitor, or hardware.

3.58 skip 3.58

Bypasses parts of current input as specified in the parameter list.

3.58.1 Example 3.58.1

```
(test2=edit text1
skip æ)
1./error/,r/er/err/
12,r/sory/sorry
f
æ
```

In case of an error during editing the remainder edit commands will be skipped, i.e. when skip is called the input position in current input is forwarded to just after the second æ character.

3.58.2 Call 3.58.2

skip { <s> <lines> | <s> <iso value>.<appearances> | <s> <small letter> }^∞_1

<lines>::= <integer>

<iso value::= { <integer> | <small letter> }

<appearances::= <integer>

3.58.3 Function 3.58.3

The program interpretes one parameter at a time and skips current input as follows:

<lines>
This number of graphical lines are skipped.

<iso value>.<appearances>
Skips until the specified number of appearances of the iso character is bypassed.

<small letter>
Skips up to and inclusive this letter.

3.58.4 Storage Requirements 3.58.4

1024 bytes plus space for FP.

3.58.5 Error Messages 3.58.5

***skip call
An output file has been specified in the call. This is ignored.

***skip param <illegal parameter>
Illegal parameter syntax. The parameter is ignored.

***skip end medium
Current input is exhausted. The program is terminated. Notice: current input is not unstacked.

3.59 suspend 3.59

Sends a suspend message to the parent (the operating system) asking for suspension of the specified magnetic tape reel. This is relevant for worktapes only. The station is now available for mounting of another tape reel but the suspended worktape is still reserved for the job until it terminates or releases the tape reel. Each suspend operation uses a suspend buffer (cf. ref. [10], section 6.1).

3.59.1 Example 3.59.1

A worktape has been mounted by the FP commands

```
workfile=set mto 0 0 1
mount workfile
```

The job has produced some output on 'workfile' but now needs the station for another purpose. The worktape is therefore suspended by the FP command

```
suspend workfile
```

When the name 'workfile' is referred to later in the job, the worktape is demanded. In the meantime no other job is allowed to use the tape.

3.59.2 Call 3.59.2

suspend <s> <name>

3.59.3 Function 3.59.3

A suspend message is sent to the parent. The name in the message is found as follows: the name is looked up in the catalog. If an entry describing a magnetic tape file (kind=18) is found and if this entry is not protected (e.g. not of scope system) the document name in the entry is used. Otherwise the name specified is used.

3.59.4 Storage Requirements 3.59.4

1536 bytes plus space for FP.

3.59.5 Error Messages 3.59.5

***suspend call

Left hand side in the call of the program.

***suspend <parameter list> parameter error

Parameter error in the call of the program.

***suspend <name> tape name missing

The entry specified has a zero document name.

In case of any error no suspend message is sent.

3.60 timer 3.60

Sends a timer message to the parent (the operating system) demanding a provoked interrupt after a certain time.

3.60.1 Example 3.60.1

The FP call

```
timer 30 2
```

will provoke an interrupt after 30 seconds.

3.60.2 Call 3.60

timer <s> <run time> <s> <break time>
where <run time> and <break time> are integers, denoting time in seconds.

3.60.3 Function 3.60.3

A timer message containing the two integers is sent to the parent. If BOSS is the parent <run time> will be the number of seconds to the interrupt, <break time> the number of seconds allowed the job to respond to the interrupt.

3.60.4 Storage Requirements 3.60.4

1536 bytes plus space for FP.

3.60.5 Error Messages 3.60.5

***timer call
Left hand side in the call of timer.

***timer <parameter list> parameter error

Parameter error in the call of timer.

In case of any error no timer message is sent.

3.61 translated 3.61

The program prints the date of translation which is found in all ALGOL/FORTRAN programs.

3.61.1 Example 3.61.1

The call

```
translated onefile twofile
```

may produce the following output:

```
onefile translated by algol 83.04.07 12.40
twofile translated by fortran 83.04.07 11.09
```

3.61.2 Call 3.61.2

translated $\left\{\text{<name>}\right\}_{0}^{\infty}$

3.61.3 Function 3.61.3

If <name> describes a program, the file is connected and the program searches for the date, which will be output.

3.61.4 Storage Requirements 3.61.4

2048 bytes plus space for FP.

3.61.5 Error Messages 3.61.5

***translated call

Left hand side in the program call.

***translated param <parameter>

Parameter error in the program call.

***translated <name> not found

The parameter was not found in the catalog.

***translated <name> not program

The catalog entry <name> does not describe a program (i.e. contents are not 2) or it does not describe an area or the first word is not 4 which is the case for all ALGOL and FORTRAN programs.

***translated <name> error

The file <name> could not be connected.

***translated <name> date not found

The program did not succeed to find a date.

A. REFERENCES A.

[1] RCSL No 31-D476:
RC8000 Monitor, Part 1, System Design,
Henrik Sierslev, Pierce Hazelton

[2] RCSL No 31-D697:
RC8000 Monitor, Part 2, Reference Manual,
Torben Steen Hansen

[3] RCSL No 31-D478:
RC8000 Monitor, Part 3, Definition of External Processes,
Palle Andersson

[4] RCSL No 31-D643:
Operating System s, Reference Manual,
Henrik Sierslev

[5] RCSL No 42-i1278:
ALGOL8, User's Guide, Part 2,
Edith Rosenberg

[6] RCSL No 31-D392:
RC FORTRAN User's Manual,
Jens Hald and Alan Wessel

[7] RCSL No 42-i785:
Slang Assembler, Programming Guide,
Palle Andersson

[8] RCSL No 31-D676:
Utility Programs, Part 1,
Finn Strøbech

[9] RCSL No 31-D680:
BOSS Operating Guide,
Carl Henrik Dreyer

[10] RCSL No 42-i1265:
BOSS User's Manual,
Bent Bæk Jensen

[11] RCSL No 31-D379:
Utility Programs, Part 3,
Tove Ann Aris and Hans Rischel

[12] RCSL No 31-D199:
Code Procedures and Run Time Organization of ALGOL Programs,
Tom Sandvang

RETURN LETTER

Title: System 3 Utility Programs, Part Two, User's Guide.

RCSL No.: 31-D607

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

Do you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

4211288

. . . . . Fold here . . . . . .

. . . . . Do not tear - Fold here and staple . . . . . .

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark

information	repl.		ident FGS840613		page 1/16
			X RC8000		class EXT

subj. CATSORT AND CAT, SW8010/2, SYSTEM UTILITY PACKAGE, RELEASE 1.0.

In System Utility Package, version 2, Release 1.0, some changes and extensions have been made in the program catsort, and another program, cat has been added.

The following description replaces the description of catsort in RCSL No 31-D607, System 3 Utility Program, Part Two, Users Guide.

RCSL NO: 42-i 1121

TABLE OF CONTENTS **PAGE**

TABLE OF CONTENTS **PAGE**

1. catsort

Lists on current output selected parts of the main catalog (or any auxiliary catalog) sorted according to parameters. At last also total number of entries and segments output are listed.

1.1 Examples

1.1.1 Example 1

The FP command:

```
catsort base.project.min
```

will list all entries with a base contained in the project base, e.g. belonging to the actual project. The parameter min causes that only name, segments, docname, date, and scope is listed.

1.1.2 Example 2

The FP command:

```
catsort scope.project min.yes
```

will do the same but only permanent entries with an entry base equal to the project base are output.

1.1.3 Example 3

The FP command:

```
catsort
```

will list all entries in the main catalog sorted according to base and entry name.

See also 1.6, Further Examples.

1.2 Call

```
[<outfile>=]1/0   catsort [<catalog spec>
                           <entry spec>
                           <sorting spec>]*0

<catalog spec> ::=  main.[yes
                          no]
                    cat .[yes
                          no
                          <integer>
                          <document name>
                          main]

<entry spec> ::= [system.[only
                          yes
                          no]
                  name.<entry name>
                  docname.<document name>
                  base.  [<scope>   .min
                          <lower>.<upper>]
                  scope. [<scope>   .min
                          <power>.<upper>]
                  before. <clock>
                  after.  <clock>
                  size.   <lower>.<upper>
                  cont.   <lower>.<upper>
                  min.    yes/no]
```

<sorting spec> ::= { basesort | docsort | slicesort | sort } . { yes | no }

<scope> ::= { system | project | user | login | temp }

{ <lower> | <upper> } ::= <integer>

<clock> ::= <yymmdd>.<hhmmss>

{ <yymmdd> | <hhmmss> } ::= <integer>

1.3 Format of Output

In case of basesort.yes each entry is listed in one line in the form:
<entryname> <modekind> <document name> <remaining entry tail>

If the parameter min is specified the line is:

<entryname> <modekind> <document name>

Before each group of entries with the same base, one line stating the base and permkey is listed.
In case of basesort.no, this line is replaced by a supplement to each entry line. The supplement comes after entry name and consists of the information from the entry head:

<first slice> <name key> <catalog key> <lower base> <upper base>

RCSL NO: 42-i 1321

If, however, an auxiliary catalog is listed, the document name of area entries is replaced by:
-> <write access counter><read access counter> <-
and another line follows:
-> d.<latest changed> <- d.<latest read>

1.4 Function

If an outfile is specified, this file is used for output, otherwise current output file is used.

The catalogs are copied, one by one, into a temporary work file, which is sorted according to the parameters.

The parameters are processed, one by one, before any catalog is accessed.
Any parameter being repeated is a modification to the previous one.

1.4.1 Sorting specifiers:

basesort.yes — meaning sorted after the entry base (which means grouped after project and users).

docsort.yes — meaning that each area entry is followed by all subentries which have a document name equal to the entry name of the main entry.

slicesort.yes meaning sorted according to first slice. This parameter will cancel the parameter docsort.yes and has the same priority.

The priority of the sorting parameters are basesort, docsort. The last sorting criterion will always be alfabetic sorting on entry name.

sort.no meaning that no sorting at all is performed. The total catalog will be listed, neglecting all other parameters but main and cat.

1.4.2 Catalog specifiers:

main.<yes/no> defines whether the main catalog is listed or not.
Default: main.yes

cat.<integer> .<name> .main .yes .no	defines whether the auxiliary catalog is listed or not. An integer or a document name specifies an auxiliary catalog to be listed. The name 'main' specifies the disc holding the main catalog. The name 'yes' specifies all auxiliary catalogs. The name 'no' specifies no auxiliary catalogs. The option prevents the main catalog from being listed, unless explicitly specified by main.yes. Default: cat.no.

1.4.3 Entry specifiers:

system.yes .no .only	defining whether the system files are listed or not. Default: system.yes
name.<name>:	only entries with the entry name specified are listed. Default: any name.
docname.<name>:	only entries containing the document name specified are listed. Default: any document name.

base.<scope>:	only entries with an entry base contained in the base specified by the scope are listed. Default: base.system.
base.<lower>.<upper>:	only entries contained in the specified base are listed. A negative value of <lower> or <upper> must be given as the positive complement e.g. the integer -1000 is specified as 16777216-1000= 16776216.
scope.<scope>	only entries with the specified scope are listed.
scope.<lower>.<upper>	only entries with entry base as specified are listed (cf. above).
before.<clock> after.<clock>	only entries updated before (or after) the clock specified are listed. In the main catalog, short-clock is used as latest up-date, i.e. only non procedure entries are candidates. In auxiliary catalogs, latest changed in the tail is used for area entries, shortclock for non procedure file descriptors. Default: after.zero before.infinity.

size.<lower>.<upper> only area entries with a size in the interval specified are output, (lower and upper size included).
Default: size.0.infinity.

cont.<lower>.<upper> only entries with a contents key in the interval specified are output (lower and upper included).
Default: cont.0.infinity.

1.5 Error Messages

*** catsort error param <erroneous and following parameters>.
Parameter error in the call.

*** catsort, create sortarea impossible.
It was impossible to create an area for sorting.

In case of any error message, the program terminates.

1.6 Further Examples

1.6.1 Example 1

```
catsort sort.no
```

will list the total main catalog segment by segment, in unsorted form, empty entries represented by a line marked: -
For each segment is stated the segment number and the number of non empty entries on the segment.

1.6.2 Example 2

```
catsort cat.yes
```

will list all entries in the auxiliary catalogs sorted according to base and entry name.

1.6.3 Example 3

```
catsort name.pip docname.pip basesort.no
```

will list all entries in the main catalog with entry name pip or document name pip, sorted according to entry name.

1.6.4 Example 4

```
catsort docname.disc
```

will list all entries in the main catalog with document name disc, sorted according to base and entry name.

1.6.5 Example 5

```
catsort cat.discl
```

will list all entries in the auxiliary catalog with document name discl sorted according to base and entry name.

1.6.6 Example 6

catsort main.yes cat.yes size.5000.8000000
will list all area entries from the main catalog and all auxiliary catalogs which have a size greater than 5000 segments.

1.6.7 Example 7

catsort after.840506.174500 before.840506.240000
will list all entries in the main catalog opdated since the date 1984.0506 at 174500 until the date 840506 at 240000, i.e. entries with a shortclock (all entries which are not procedure entries) set in the interval given.

1.6.8 Example 8

catsort cat.disc3 after.840601.000000
will list all entries in the auxiliary catalog on the disc 'disc3' which have been updated since the date 840601 at 000000 hours, i.e. area entries (last changed in the tail is used) or file descriptors which are not procedure entries (the shortclock of the tail is used).

2. Cat

The program works as catsort, except no sorting of entries prior to output takes place, i.e. the entries specified in the call are listed in the order in which they are found in the catalog.

2.1 Examples

2.1.1 Example 1

The call

```
cat scope.system
```

will list all entries in main catalog of scope system in the order they are found in the catalog.

2.1.2 Example 2

The call

```
cat
```

will list all entries in the main catalog in the order they are found in the catalog.

2.1.3 Example 3

The call

```
cat cat.disc3 name.pip docname.pip
```

will list all entries in the catalog with document name disc3 which have either the name 'pip' or the documentname 'pip'.

2.2 Call

Exactly as for catsort, except in spite of sorting parameters, no sorting takes place.

2.3 Format of output

Exactly as for catsort.
Note that the parameter basesort.no will influence the format of the output.
Since the entries appear unsorted, entries of different bases cause an additional line stating the entry base and permkey in case of basesort.yes.

2.4 Function

As for catsort, except the temporary work file is not sorted.

2.5 Error messages

As for catsort.

RCSL No: 42-i 1321

2.6 Further examples.

2.6.1 Example 1

```
cat sort.no
```

will list all entries in the main catalog, discregarding the parameter sort.no

2.6.2 Example 2-8

The examples in 1.6.2-1.6.8 go for cat as well, except the entries are listed in the order they are found in the catalogs.