RCSL No: 31-D643

Edition: May 1981

Author: Henrik Sierslev

Title:

OPERATING SYSTEM s
Reference Manual

RCSL 42-i1592

A/S REGNECENTRALEN
af 1979

Keywords:

RC8000, RC6000, basic operating system, initializing commands.

Abstract:

This manual describes the functions of the basic operating system s, and the functions of the commands used for initializing the backing storage.

(100 printed pages)

Printed by A/S Regnecentralen af 1979, Copenhagen

RCSL 42-11592

FOREWORD

First edition: RCSL No 31-D455.
The major changes, compared to the version described in "RCSL No 31-D300: MONITOR3", are the introduction of a usercatalog (processes may be created by the JOB-command with prespecified claims) the initializing commands and the lock/unlock facility to prevent/reestablish the use of ALL and NEW commands, from other consoles than the mainconsole or privileged consoles.

Palle Andersson
A/S Regnecentralen, January 1979

Second edition: RCSL No 31-D595.
The major changes in this manual compared to RCSL No 31-D 455 are the inclusion of the corrections described in RC Information Note NCJ 790715 "Corrections to s Reference Manual", and some corrections in the examples in use of usercat.

Further the prio command has been changed:
The priority is now a parameter in the console buffer, and the prio command merely sets this parameter. The create command will then set the priority when the process is created. It is also possible to set the priority in the jobspecification in Susercat. The remarks about the priority are valid from monitor release 6.0. In previous releases the prio command works as described in the RC Information Note NJC 790715, "Corrections to s Reference Manual".

Henrik Sierslev
A/S REGNECENTRALEN af 1979, May 1980

Third edition: RCSL No 31-D643.
The major changes in this manual are the introduction of five new commands to s and the two parent messages finis and replace.
These changes are valid from monitor release 7.0.

It is now possible to let s read commands from a bs area instead of the console, and it is possible to set primary input in an internal process causing FP to start immediately reading commands.

Henrik Sierslev
A/S REGNECENTRALEN af 1979, May 1981

TABLE OF CONTENTS PAGE

1. BASIC OPERATING SYSTEM FUNCTIONS

The functions of the basic operating system, s, which can initiate and control the execution of parallel programs on request from terminals/consoles (in the following called consoles), are described.

1.1 Introduction

After initial system loading, the internal store contains the monitor and the basic operating system s, enabling operators to initiate and control internal processes from consoles.
In addition, s can name peripherals as for instance magnetic tape.

s may be the "pater familias" of a family tree of internal processes. Initially s owns all system resources, such as core storage, backing storage, peripherals, message buffers, process description tables etc. Apart from being the initial process in the system, s has no special status, and it is treated as any other internal process. It is possible to replace s with another operating system.

1.2 Handling of Consoles

Commands to s from a console are served sequentially in the order of their arrival. The processing of a line of commands is terminated by a short reply printed on the console.

The monitor permits simultaneous input of messages from all consoles. The operating system, however, can only respond to a limited number of messages. The maximum number of simultaneous console actions within s is an option defined in the assembly of the monitor with s.

A line of commands to s is processed from left to right, one command at a time. If syntax errors or other errors are detected du-

ring the processing of a line, an alarm is issued and a possible unprocessed rest of the line is dropped.
For every command-line you perform:

```
@att -> s
->command-line
```

1.3 Console Description

For each console communicating with the operating system, s maintains a description defining the parameters to be used in the creation of an internal process with the console as primary input and primary output (see ref. 1). This is done by means of a common consolebuffer pool. The size of this pool is an option to the monitor.

A console description is initiated by means of commands to s.

The parameters referred in this paper are listed below together with their type. The type describes what may be stored in each field. The meaning of type are: string ~ max 11 characters; integer ~ -8388608 <= number < 8388607; integer pair ~ 2 integers; short integer ~ 0 <= number <= 4095.

parameter name:	type:
process name	string
first address	integer
top address	integer
buffer claim	short integer
area claim	short integer
internal claim	short integer
function mask	short integer
max interval	integer pair
standard interval	integer pair
user interval	integer pair
process size	integer
program name	string
priority	short integer

1.3.1 Explanation of Command Mask

The command mask is a bit mask in the console buffer. It consists of two parts. A part defining the allowed s command from this console type (i.e. predefined or consolepool), and a part, used for signalling between commands, the command bits.

Format of command mask:

```
1   1 shift 10 ; command bit : all bs resources.
2   1 shift 9  ; mode, modify, print, date.
3   1 shift 8  ; job, start, stop, break, dump, list, max,
                 remove, proc, prog, load, I, O, get, unstack.
4   1 shift 7  ; include, exclude.
5   1 shift 6  ; size, login, user, project, prio, base.
6   1 shift 5  ; addr, function, buf, area, internal, bs, temp,
                 perm, all, call.
7   1 shift 4  ; new, create, run, init.
8   1 shift 3  ; privileged console.
9   1 shift 2  ; command bit : mode = 0
10  1 shift 1  ; command bit : absolute address.
```

bit 2 - 8 are generated as options to the system and cannot be altered.

bit 1, 9 and 10 can be used in connection with a job in susercat.

For each possible bs-device, we have:

```
temp  segments <bs device> integer
temp  entries  <bs device> integer
perm  segments <bs device> integer
perm  entries  <bs device> integer
```

1.4 Some Important Commands

Three commands to s, initializing the whole set of console parameters, exist:
The new command using some standard values, the all command, searching and using the maximum available resources, and the job command, searching a file named "susercat" in order to find suitable values to insert into. The job command will also create, load and start an internal process according to the parameters found. These three commands are followed by a string to be used as process name.

The run command will create, load and start an internal process according to the console description.
The stop command will stop a running process without removing it. The process may be restarted by means of the start command.
The start command will restart a stopped process. The process may either have been stopped by means of the stop command or by "itself", for instance after having requested a magnetic tape.

The remove command will stop and remove an internal process.

All commands are described in section 1.6 below.

1.5 The Command Language

A command line to s is one line which can be empty or contain a number of commands. A line is either terminated by a new line character or by semicolon followed by a comment consisting of any string of characters terminated by a new line character. The commands in a message may consist of one or more of the following syntactical elements:

<separator>::= <SP> |.|,|/|=|<separator><SP>
<unsigned number>::= <digit>|<unsigned number><digit>
<radix>::= <unsigned number>
<radix number>::= <radix>:<unsigned number>
<number>::= <unsigned number>|- <unsigned number>|<radixnumber>

<name>::= <letter> { <letter> | <digit> } 10 0

<parameter>::= <number>|<name>

<command word>::= addr|all|area|break|bs|buf|call|
create|date|dump|exclude|function|get|i|
include|init|internal|job|list|load|lock|
login|max|mode|modify|new|o|perm|prio|
print|proc|prog|project|read|remove|
replace|run|size|start|stop|temp|unlock|user

A command consists of a command word possibly followed by one or more parameters.
Command words and parameters are separated from each other by a separator. In the syntactical definition of the commands in section 1.6 below, separators are not shown explicitly. It must be understood that a definition like the following

temp <devicename><entries>

strictly speaking means

temp <separator><devicename><separator><segments><separator><entries>

Addr Command

Syntax: addr <first storage address>

<first storage address>::= <unsigned number>

The console parameter first address is defined and used in a possible subsequent creation of an internal process. An odd number is decreased by one.

Example: addr 16320

All Command

Syntax: all <process name>

<process name>::= <name>

First the console parameter process name is defined by the parameter to the all command. Next all the other console parameters are initiated by setting the name "fp" into the console parameter program name, and the maximum of s's rest claims into the remaining parameters. The core size used is the greatest free consecutive core area found. Notice that the values may be redefined (diminished) by means of commands defining only a few parameters. The bases (max, standard, user) are set to the standard base of s.

Example: all us

Area Command

Syntax: area <area claim>

<area claim>::= <unsigned number>

The console parameter area claim is defined. This defines the number of available area processes.

Example: area 20

If the job wants to communicate with 20 backing storage files at the same time, it will need 20 area processes.

Base Command

Syntax: base <number><number>

Assigns the number pair to the console parameters max interval, standard interval and user interval.

Example: base -100 200

Break Command

Syntax: break

Stops the execution of the internal process, defined by the console parameter process name, stores its registers and the interrupt cause 8 at the head of its internal interrupt procedure ("break sequence"), and restarts the process in its interrupt procedure (see ref. [1]).

Example: break

Bs Command

Syntax: bs <bs device name> <segments> <entries>
<bs device name>::= <name>
<segments>::= <number>
<entries>::= <number>

The claims of the (already created and possibly running) process defined by the console parameter process name are increased by the (possibly negative) amount given by entries and segments for all keys on the backing storage given by bs device name.

Example: bs disc2 1500 20

The claims on disc2 will be increased with 20 entries and 1500 segments for all perm keys.

Buf Command

Syntax: buf <buffer claim>

<bufferclaim>::= <unsigned number>

The console parameter buffer claim is defined. This defines the number of available message buffers.

Example: buf 14

If the job has 14 data transfers (or other I/O operations) going on simultaneously, it will need 14 message buffers.

Call Command

Syntax: call $\left\{ \text{<device number> <device name>} \right\}_{1}^{*}$

<device number>::= <unsigned number>

<device name>::= <name>

Assigns names to a sequence of peripheral devices. The devices must not be reserved by internal processes.

Example: call 0=reader 10=magtape 5 printer

Create Command

Syntax: create

Creates an internal process as described by the console parameters.

If the addr command has been used, the availability of the storage area is checked. Otherwise s scans the store from the low address end and chooses the first available core area of sufficient size.

If mode=0 has been used the write limits in the process description is set to maximum, allowing the process to write in the whole storage, else the write limits will equal the storage area found above.

The ability of s to supply the specified claims is checked. After the creation s includes the process as a user of a standard configuration of peripherals, assigns the backing storage claims listed, and sets the priority of the process.

The registers of the process are initialized as required by ref. [2].

The catalog base of the process is the standard (login) base.

Example: create

Date Command

Syntax: date <year><month><day><hour><min><sec>

<year>::= <digit> <digit>

<month>::= <digit>|<digit> <digit>

<day>::= <digit>|<digit> <digit>

<hour>::= <digit>|<digit> <digit>

<min>::= <digit>|<digit> <digit>

<sec>::= <digit>|<digit> <digit>

Sets the internal clock of the monitor to the value given in the call.

If the main catalog has not yet been defined, s will continue with the "oldcat" command.

Example: date 81.05.10 9.30.0

Dump Command

Syntax: dump <dump area>

<dump area>::= <name>

The process defined by the console parameter process name is stopped (if not already stopped) and its core area is copied to the dump area. If the size of the dump area is not sufficient, only the first part of the core area is copied. The dump area must be visible from the catalog base of the internal process.

Example: dump image

Exclude Command

Syntax: exclude $\left\{ \text{<device number>} \right\}_{1}^{*}$

<device number>::= <unsigned number>

The (already created and possibly running) process, defined by the console parameter process name is excluded as a user of a sequence of peripheral devices.

Example: exclude 5

Function Command

Syntax: function $\left\{ \text{<bit number>} \right\}_1^*$

<bit number>::= <unsigned number>

The console parameter function mask is set to contain ones in the selected bits. The rest is set to contain zeroes.

Example: function 4 5 6

Get Command

Syntax: get <process name>
<process name>::= <name>

Searches the systemfile "susercat" for an entry with the given processname and initializes the console description according to the entry, but unlike the job command, no process is created.

Example:
get student run,

will be the same as

job student

As the I and O commands are not implemented in susercat, you may instead use

get student I 4 input run

I Command

Syntax: I <kind> <name>

<kind>::= <short integer>

Sets the console parameter primin to the parameters given in the call. At load time the registers of the created internal process are set as required by ref. [2], causing FP to select the given name as primary input.

Example: new job1 I 4. input run

FP will select the area (kind=4) "input" as primary input.

If FP is reinitialized, maybe because of a syntax error, FP will start reading from the beginning of the input file. The private modebits, however, will not be changed, so the following commands will prevent FP from looping:

```
if mode.1 yes
finis
mode.1 yes
```

Include Command

Syntax: include $\left\{ \text{<device number>} \right\}_1^*$

<device number>::= <unsigned number>

Includes the (already created and possibly running) process, defined by the console parameter process name, as a user of a sequence of peripheral devices.

Example: include 6 7

Init Command

Syntax: init

This command has the same effect as a create command followed by a load command.

Example: init

Internal Command

Syntax: internal <internal claim>
<internal claim>::= <unsigned number>

Assigns a number to the console parameter internal claim.

Example: internal 2

A job will only need internal processes if it acts as an operating system.

Job Command

Syntax: job <process name>
<process name>::= <name>

Searches the system file "susercat" for an entry with the process name given. If the entry is found, the console description is initialized according to the entry, and a process is started by means of the functions implementing the create, the load and the start commands in the order mentioned.

Example: job student

List Command

Syntax: list

Prints a list of all internal processes having a parent.

Each process is described on one line with the format:

<process name><first storage address><size of storage area>
<stop-count><state><parent name>

Example: list

Load Command

Syntax: load

Loads the program defined by the console parameter program name into the beginning of the internal process defined by the console parameter process name.
The program must be located on the backing store, and be described as a catalog entry with a positive size and content key (see ref. [1]) must be 3 or 8.

A possible area process for the entry must not be reserved. The number of halfwords to load and the relative entry point of the program must not exceed the size of the internal process. The program must be visible from the catalog base of the internal process.

After loading, the registers are set as required by ref. [2], and the instruction counter is set to the absolute address corresponding to the relative entry point. The catalog base of the internal process is set to the user base, given in the console description.

Example: load

Lock Command

Syntax: lock
(privileged console only)

When this command has been used, it is only possible to use following commands from a non privileged console:

break
dump
job
list
load
max
proc
prog
remove
start
stop

Example: lock

Login Command

Syntax: login <ll standard base> <ul standard base>
<ll standard base>::= <number>
<ul standard base>::= <number>

Assigns the number pair to the console parameter standard interval. This usually defines the temp scope in the sense of "File Processor".

Example: login 1025 1025

Max Command

Syntax: max

Prints the maximum of certain available resources with the format:

max <max storage area><buf claim><area claim><internal claim>

Example: max

Mode Command

Syntax: mode <integer>

If integer = 0 the internal process is allowed to write in the whole primary storage. If integer <> 0, the process can only write in it's own storage area.

Default is mode <> 0

Example: mode 0
- : mode 1

Modify Command

Syntax: modify <address><old contents><new contents>
(privileged console only)

This command changes the word with address <address> to <new contents> if <old contents> is the present contents of the word.

Example: modify 29276 30 34

Example: modify 29314 8:24700014 8:24300014

This command changes word 29314 from rl.w3 8:14 to rl.wl 8:14.

New Command

Syntax: new <process name>

<process name>::= <name>

First the console parameter process name is defined by the parameter to the new command. Then all the other console parameters are initiated by setting the name fp into the console parameter program, and some standard values into the remaining parameters.

Example: new me

The most common used standard values are:

priority	0
no. of message buffers	7
no. of area processes	6
no. of internal processes	0
function mask (octal value)	7440
core size	12800
perm entries on work device	20
perm segments on work device	800
standard work device	<:disc:>
all base values	8388605

O Command

Syntax: O <kind> <name>
<kind>::= <short integer>

Sets the console parameter primout to the parameters given in the call.

At load time the registers of the created internal process are set as required by ref. [2], causing FP to select the given name as primary output.

Example: new job1 O 8 sub010 run

The terminal "sub010" is selected for output.

Perm Command

Syntax: perm <bs device name><segments><entries>

<bs device name>::= <name>

<segments>::= <unsigned number>

<entries>::= <unsigned number>

Searches the monitor for a backing storage device with the stated name. Then the console parameters for permanent claims for that device are defined. The stated values are added to the temp claims for the device.

Example: perm disc1 1500 20

Print Command

Syntax: print <first storage><last storage>
<first storage>::= <unsigned number>
<last storage>::= <unsigned number>

Reserves the device named printer and prints the contents of the core storage starting with <first storage>. The printing ends when last storage has been passed after printing at least one word, or the current word is outside core store, or the printer could not be used properly (e.g. because of endpaper or because the printer was reserved). Then the printer is released.

The format of the printing is:

<address decimal><word decimal><halfword 1 decimal><halfword 2 decimal>
<word octal><word text>

Example: print 10246 10700

Prio Command

Syntax: prio <short integer>

The console parameter priority is set to the value given in the call.

Normally a process will be created with priority 0, which is the highest possible priority of an internal process.

Proc Command

Syntax: proc <process name>
<process name>::= <name>

Assigns a name to the console parameter process name. It may for instance be used as a preample to commands like include, exclude, start etc.

Example: proc him start proc me

This sequence may for instance be used by the main operator (sitting at a privileged console) after having mounted and called a magnetic tape. After having started the waiting process, the operator reinserts the name of his own process in the console description.

Prog Command

Syntax: prog <program name>
<program name>::= <name>

Assigns a name to the console parameter program name.

Example: prog batchsystem

Project Command

Syntax: project <ll maxbase><ul maxbase>
<ll maxbase>::= <number>
<ul maxbase>::= <number>

Assigns the number pair to the console parameter max interval. This defines the scope project in the sense of the "File Processor".

Example: project 0 2000

Remove Command

Syntax: remove

Stops and removes an internal process defined by the console parameter process name. The console must be either the console from which the process was created or a privileged console. Note that the console parameters are unchanged. If the process is created by means of the job-command, the description of the permanent bs-claims in "susercat" belonging to the process is adjusted.

Read Command

Syntax: read <area name>
<area name>::= <name>

Causes s to stack the current input pointer and start reading commands from the bs area. The last command must be "unstack" which will cause s to restore the previous input pointer and continue the command reading from there.

The area must be visible from the catalog base of s.

The stack depth is an option to s. Default is stack depth = 2, i.e. it is allowed to have one read command in an area.

Example: read pip

where pip contains

all fut run unstack,

will be the same as the command line

all fut run

Replace Command

Syntax: replace <area name>
<area name>::= <name>
(privileged consoles only)

The area must contain an independent program (contents key = 8). This program is loaded replacing the code of s (see ref. [1]).

Run Command

Syntax: run

The run command has the same effect as a create command followed by a load command and a start command.

Example: run

Size Command

Syntax: size <size of storage area in halfwords>
<size of storage area in halfwords>::= <unsigned number>

Assigns a number to the console parameter size of storage area. An odd number is decreased by one.

Example: size 20000

Start Command

Syntax: start

Starts the execution of an internal process defined by the console parameter process name.
The console must be either the console from which the process was created or a privileged console.

Example: start

Stop Command

Syntax: stop

Stops the execution of an internal process defined by the console parameter process name. The console must be either the console from which the process was created or a privileged console.

Example: stop

Temp Command

Syntax: temp <bs devicename><segments><entries>

<bs devicename>::= <name>
<segments> ::= <unsigned number>
<entries> ::= <unsigned number>

Searches the monitor for a backing storage device with the given name. Then the console parameters for temporary segments and entries for that device are defined.

Example: temp disc2 1500 3

Unlock Command

Syntax: unlock
(privileged console only)

Will establish the use of the commands excluded by the lock command.

Example: unlock

Unstack Command

Syntax: unstack

Causes s to continue reading commands from a previous stacked input source. If no input source is stacked, the command will be blind.
Input will be stacked by the read command.
The stack depth is an option to s. Default is 2.

User Command

Syntax: user <ll userbase> <ul userbase>
<ll userbase>::= <number>
<ul userbase>::= <number>

Assigns the number pair to the console parameter user interval. This usually defines the user scope in the sense of "File Processor".

Example: user 1025 1035

2. MESSAGES FROM s 2.

2.1 Console Messages 2.1

When the operating system has successfully processed a line of commands, it prints the reply:

ready

If an error is detected during processing of a command in a line then an error message is displayed, the rest of the command line is ignored and the console returns to the ready situation.

The following is a list of error messages printed by the operating systems:

error message:	meaning:
area error	input/output error during area loading or the print-command is trying to print a word outside corestore
area reserved	area reserved by another process
area unknown	area not described correctly in catalog or area name incorrect
base illegal	incorrect use of the bases remedy: examine your use of the project, user, login- and base-command
bs claims exceeded	bs claims not available; the process is removed remedy: create the process with "smaller" claims
bs device unknown	device name does not exist remedy: examine your use of the bs-command

catalog error	input/output error during catalog lookup or catalog system not initialized
device reserved	device reserved by another process remedy: wait until the device is not reserved
device unknown	device number does not exist remedy: use the correct device-number
input aborted	the buffer from the console has been received with an abnormal status remedy: send the command line to s once more
name unknown	name does not exist in "susercat" remedy: create an entrance in "susercat" with the given name
no areas	area process description not available remedy: use the area-command
no buffers	messages buffers not available remedy: use the buf-command
no core	storage area not available remedy: use the size-command
no entries in maincat	the number of temporary entry claims are not available in the main catalog remedy: create the process with less temporary entries
no internals	internal process descriptions not available remedy: use the internal-command
not allowed	command forbidden from this console, system under initialization or state of process does not permit this command

not implemented	optional command not assembled
process unknown	process does not exist remedy: use the proc-command
prog name unknown	the program name is not visible from the catalog base of the internal process remedy: examine your use of the user-and base-commands
program too big	program size or entry point exceeds primary storage area remedy: use the size-command
syntax error	illegal command syntax (or e.g. timeout)

2.2 Child Messages

2.2

When the operating systems receives a message from one of its child processes, the following is done:

a) If the waiting bit is equal to one (bit No 23 in the first word of the message) then the child process is stopped, an answer is sent to the process and a message is printed on the console from which the process was created:

pause <processname><text>

The operator can now start, break, or remove the process depending on the function in the message.

b) If the waiting bit is equal to zero, then an answer is sent to the process, and a message is printed on the console from which the process was created:

message <processname><text>

The format of a message is following:
+0: function shift 12 + Pattern shift 5 + wait
+2: integer or text portion
+4: integer or text portion
until +14

Function If function is "finis", and wait is one, s will remove the process.
If function is "replace", and wait is one, s will remove the process and start reading from the bs-area specified (see the read command).

Pattern Specifies how the message is to be displayed to the operator. The pattern contains seven bits, one to each of the words in +2 to +14 of the message.
A bit being one means that the corresponding word should be printed as an integer, otherwise the word is printed as a text portion of 3 characters.
Thus 1<11 means that the word in +2 is an integer.

Wait May be zero or one.

3. ALFABETIC LIST OF COMMANDS FOR INITIALIZATION OF THE CATALOG SYSTEM 3.

The functions of the initializing commands, which can initiate the backing storage, are described.

3.1 Introduction 3.1

If you want to overrule the automatic oldcat action, change the maincatsize or name, initialize a new system, change the kitlabel etc., you may use the following commands to the operating system s. These commands can only be used until the first process is created.

The commands may be typed as described in section 1.5, but in contrast to the reaction on 'normal' error messages, see section 2.1, the rest of the command line will not be skipped in case of errors.

3.2 Description of the Commands 3.2

AUXCLEAR Command

Syntax: auxclear $\left\{ \begin{matrix} \text{fast} \\ \text{slow} \end{matrix} \right\}_{0}^{1}$ <devno> <lowerbase>

<upperbase> $\left\{ \text{<entryname>} \right\}_{0}^{*}$

<devno>::= <unsigned integer>
<lowerbase>::= <number>
<upperbase>::= <number>
<entryname>::= <name>

This command includes temporarily, the stated device in the bs-system with anonymous auxcatname and documentname, and possibly with changed device kind (fast/slow).

All the stated catalog-entries will be removed from auxcatalog.

The discdrive will be excluded from the bs-system.

Example:

```
auxclear slow 6 8: 40000001 8388606 catalog
auxclear slow 6   -8388607  8388606 catalog
```

Both of these commands delete the catalog entry with the name <:catalog:> and the interval -8388607, 8388606 from auxcat on device 6.

Errortexts

a) syntax error

b) as the KIT-command a.-e.

c) remove aux entry <entryname> result <integer>

   <integer> is explained in the description of the monitor procedure 'remove aux entry' (see ref. [4])

d) as the NOKIT-command

BININ Command

Syntax: binin { bs / mto / nrz / tro } <docname> { <position> }*1

<docname>::= <name>

<position>::= <unsigned number>

This command reads from the stated device (cf. BININ RCSL 31-D234).

If the reading is performed from bs/mto/nrz you must as the very first name the device by using the CALL-command.

If tro the paper tape must be ready in the paper tape reader.

<position> indicates:

On disc it is the number of the first segment in the BININ-file.

On magnetic tape it is the filenumber.

On papertape it is irrelevant.

Example:

```
call 10 systemtape
binin mto systemtape 2 3 4 5 6 7 8 9
```

These commands will name the magnetic tape station and read 8 files from the tape.

Errortexts/Messages	Meaning:
a) modekind illegal	the first parameter is not bs/mto/nrz/tro
b) <docname> status <logical status-bitpattern>	status bitpattern after input (see appendix C)
c) input sum error	sum error in SLANG-segment
d) input size error	command segment > 256 words
e) <command name> syntax error	illegal command in command segment
f) create <entry name> result <integer>	create entry <> 0
g) change <entry name> result <integer>	change entry <> 0
h) rename <entryname> result <integer>	rename entry <> 0
i) remove <entryname> result <integer>	remove entry <> 0
j) perman <entryname> result <integer>	permanent entry <> 0
k) load <entryname> result <integer>	create area process <> 0
l) <entry name> status <logical status bitpattern>	status bitpattern after area output (see appendix B)

<integer> is the resultvalue described in the matching monitor procedure (see ref. [4]).

Clearcat Command

Syntax: clearcat

This command releases all discdrives/chaintables/catalogs.

Errortexts:

a) delete bs <docname> result <integer>

<integer> is explained in the description of the monitor procedure 'delete backing storage'.

b) delete entries <docname> result <integer>

<integer> is explained in the description of the monitor procedure 'delete entries'.

KIT

Syntax: kit <new docname> [<new auxcat name> [fast / slow]1 0]1 0
<device number>

<new docname>::= <name>
<new auxcat name>::= <name>
<device number>::= <unsigned number>

This command includes device <device number> in the bs-system with the documentname <new docname> and possibly with changed auxcatname <new auxcatname> and device kind (slow is used in connection with discdrives).

The document name and name of auxiliary catalog will be changed in the kitlabel (i.e. the chainhead).

If the disc drive is switched off, this command is blind.

Example: kit privkit catprivkit slow 8

When the first kit is included in the bs-system, the corresponding auxcat is searched for an entry describing a maincatalog with the name <maincatname> (see the MAINCAT command). If the entry exists, but has an incorrect size, it will be removed from the auxcat. If the entry does not exist, it will be created with the size specified by the maincat-command. This entry will be connected as the main catalog.

To indicate inclusion in the bs-system the following text is printed:

<docname> mounted on <devicenumber>

Errortexts:

a) create peripheral process wrkname on <devno> result <integer>

<integer> is explained in the description of the monitor procedure 'create peripheral process'.

b) reserve process wrkname on <devno> result <integer>

<integer> is explained in the description of the monitor procedure 'reserve process'.

c) create peripheral process documentname on <devno> result <integer>

<integer> is explained in the description of the monitor procedure 'create peripheral process'.

d) prepare bs on <devno> result <integer>

<integer> is explained in the description of the monitor procedure 'prepare backing storage'.

e) on <devno> status <logical status bit pattern>

I/O - error. <logical status bit pattern> is explained in appendix B.

f) <auxcatname> status <logical status bit pattern>

I/O - error. <logical status bit pattern> is explained in the Binin-command.

g) <auxcat-name> status <logical status bit pattern> repair not possible

I/O - error. <logical status bit pattern> is explained in the Binin-command.

h) <auxcat-name> status <logical status bit pattern>
update of entry count not possible

I/O - error. <logical status bit pattern> is explained in the Binin command.

i) insert entry <entryname> result <integer>

<integer> is explained in the description of the monitor procedure 'insert entry'.

j) auxcat to be repaired

k) auxcat to be repaired
update of entry count not possible.

l) main catalog not defined
no maincat connected
catalog error

use the MAINCAT-command or the KIT-command with the devices in the right order, to remove this errortext.

m) connect main catalog <maincatname> result <integer>
no maincat connected
catalog error

<integer> is explained in the description of the monitor procedure 'connect main catalog'.

n) remove aux entry <maincatname> result <integer>
no maincat connected
catalog error

<integer> is explained in the description of the monitor procedure 'remove aux entry'.

o) create aux entry <maincatname> result <integer>
no maincat connected
catalog error

<integer> is explained in the description of the monitor procedure 'create aux entry'.

p) (as the NOKIT-command)
no maincat connected
catalog error.

q) (errortext a-k)
no maincat connected
catalog error.

KIT

Syntax: kit $\left\{ \text{<devicenumber>} \right\}_1^*$

<devicenumber>::= <unsigned number>

This command functions like the former command, with the exception that no fields of the kitlabel (i.e. the chainhead) are changed.

Errortexts:

see the former command

KITLABEL

Syntax: kitlabel <devicenumber> <docname> <auxcatname>

$$\left\{\begin{matrix}\text{slow}\\\text{fast}\end{matrix}\right\}$$ <catsize> <slice length> <no. of slices>

<catsize>::= <unsigned number>
<slice length>::= <unsigned number>
<no. of slices>::= <unsigned number>

At present: <no. of slices> must be less than or equal to 2047
<catsize> must be less than or equal to 273

This command causes the writing of a chaintablehead and an empty auxcatalog on the device stated.

The device will be partially included in the bs-system to prevent unintentional use of the disc-kit.

If you want to use the disckit at once, then use the NOKIT-command followed by the KIT-command.

Errortexts:

intervention on <devno> (i.e. the discdrive was switched off)

and

see KIT

MAINCAT

Syntax: maincat <name of catalog> <size of catalog>

<name of catalog>::= <name>
<size of maincatalog>::= <unsigned number>|-1

At present: <size of catalog> must be less than or equal to 273.

This command defines the name and size in segments of the maincatalog.

Standard is:

<name of catalog> = catalog
<size of catalog> = -1

The size -1 indicates that the MAINCAT-command has not been used yet.

Errortexts:

None

NOKIT

Syntax: nokit <devicenumber>

<devicenumber>:: <unsigned number>

This command releases discdrive/chaintable/auxcatalog from device <devicenumber>.

Errortexts:

See CLEARCAT

OLDCAT

Syntax: oldcat

This command uses a table from monitor-options with the bs-devicenumbers, to generate a call of the KIT-command.

Example:
If the monitor is generated with bs-device 6, 7 and 8 (in this sequence), the OLDCAT-command is exactly the same as the command KIT 6 7 8.

Errortexts:

see KIT

REPAIR

Syntax: repair

This command has to be performed before a call of the KIT-commands. By doing this it is possible to suppress one errortext of the type j and k (see KIT). It is permitted to repair one catalog-segment.

3.3 Examples of Using the Initializing Commands 3.3

It is common to the succeeding examples that the monitor is loaded, and s asks for the date/time but before date/time is written on the keyboard.

3.3.1 Initializing of a New System from Magtape 3.3.1

1. Switch off all the dicsdrives. This will make the automatic OLDCAT-action completely blind.

2. Type date/time on the keyboard. Now the system is ready for KITLABEL etc.

3. Mount relevant disckits and switch on all discdrives.
   Write a label on the disckits
   e.g.

```
kitlabel  6 disc catdisc       slow 50 21 2045
kitlabel  7 disc 1 catdisc1    slow 50 21 2045
kitlabel 22 diablo1 catdiablo1 slow 20  2 1212
```

4. Perform the cleaning after KITLABEL and include the disckits in the system by giving the commands

```
clearcat
maincat  catalog 73 (or another size if wanted)
oldcat
```

5. Transfer the standardprograms from systemtape e.g.

```
call 10 systemtape
binin mto systemtape 2 3 4 5 6 7 8 9
```

Now the system is ready for creation of process etc.

3.3.2 Normal (Daily) Start with a New Specific Maincatsize 3.3.2

This can be done in several ways.

Either:

1. Mount all relevant disckits and apply power to the disc-drives.

2. Type in date/time on the keyboard. (Now an automatic oldcat-action is performed).

3. Give the commands:

```
clearcat
maincat catalog 83
oldcat
```

Or:

1. Switch off all the discdrives.

2. Type in date/time on the keyboard. (Notice: all discdrives are switched off, therefore the automatic oldcat-action is blind).

3. Mount all relevant disckits and apply power to the disc-drives.

4. Give the commands:

```
maincat catalog 83
oldcat
```

Now the system is ready for creation of processes etc.

3.3.3 Normal Start, but Maincatalog is not on the 'First' Oldcatdevice 3.3.3

1. Switch off all discdrives.

2. Type in date/time on the keyboard (notice: an automatic oldcat-action will be blind).

3. Mount all relevant disckits on and apply power to the disc-drives.

4. If necessary define the size of the maincatalog and include the disckits in the wanted order.

   e.g.

   maincat catalog 59 (if necessary)
   kit 7 6 22

4. PROGRAM SCATOP (S-CATALOG_OPERATION) 4.

The program is intended for use on operators level for s-user-catalog managing purposes, i.e. initializing the catalog, creating/changing/deleting entries or printing out contents of single entries.

The s-user-catalog is supposed to be placed on an area, named <:susercat:>, and should be present before using the scatop-program.

The program is called with a number of parameters, of which the first determines the action/subprograms to be performed. This first parameter should be one of the following names:

newcat	a new s-user-catalog will be created
insert	creation of an entry is attempted
change	changing of an entry is attempted
delete	deletion of an entry is attempted
print	an entry is attempted printed out
printcat	the usernames in susercat are printed out.

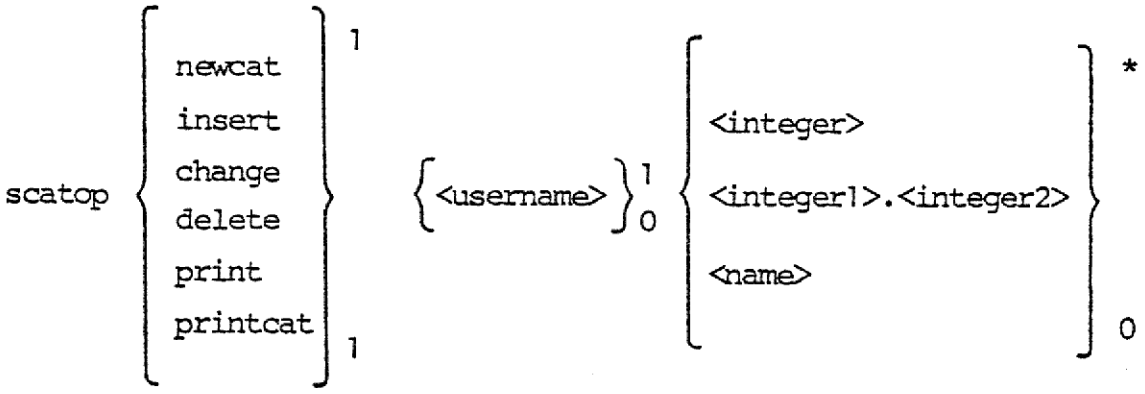
The newcat and printcat action will unconditionally be executed.

If the parameter following the action is a name the action will act upon the hereby named entry (gives no meaning for newcat and printcat). If the parameter is not a name the action will act upon the "entry0" (the catalog-describing entry).

A parameter to any action must be one of the following types:

<integer>	determines the value of one word
<integer1>.<integer2>	determines the value of one word, is interpreted as integer1 shift 12 + integer2.
<name>	determines the value of 4 words. max 11 characters, which are placed to the left in the 4 words, unused place is padded up with (binary) zeroes.

Full Syntax



All actions have in common:

-- word 0 in an entry, the hashvalue, cannot directly be set or changed, but is managed due to the built in hashfunction.

-- superfluous parameters are blind.

-- too few parameters, will be padded up with zero parameters, before the action is executed.

Referring to the format of susercat (see page 73), the detailed interpretation of the parameters in each action will be:

newcat — the whole catalog is deleted.
The parameters are interpreted strictly sequentially and laid out as entry0 in the catalog from word (+2) and up.

insert — the entry named is created and placed in the catalog. The parameters are interpreted and laid out sequentially from word (+2) and up, only the entryname and the succeeding parameter are interchanged to fit the entryformat (see page 74 and example 2).

change — changes parts or all of an entry.
Every second parameter (after an eventual entryname) indicates a word (= relative halfword address//2) to be changed. The following parameter determines the new value to be placed in this word, and succeeding words in case of a name.

delete — the indicated entry will be (marked as) deleted.
Parameters given after the entryname has no influence at all.
Note that entry0 (fortunately) cannot be deleted by this command.

print — each word in the indicated entry will be printed as one integer word, 2 integer halfwords and 3 characters. Blind, illegal and control characters are printed out as an '*'. Parameters given after the entryname have no influence at all.

printcat — each username included in susercat will be printed out, unsorted.
Parameters given after printcat have no influence at all.

UDR AF S BRUGERCAT

Calculation of entry length and 'last used of entry 0'

<last used of entry 0>: as entry 0 has the same length as a job describing entry only a part of the entry may carry information about the devices. 'last used of entry 0' is the last halfword carrying information in entry 0. It can be calculated as

6 + 12 x (no. of devices in susercat)

<entry length>: The length of an entry in susercat. If no. of devices is less than 10 the entry length is calculated as

44 + 8 x (no. of devices)

else <entry length> equals <last used of entry 0>

To avoid mistakes see to that an area named <:susercat:> is present at all before using the commands. Even before using the newcat command an area of this name (no matter the size) should be present.

Some examples (see formats of susercat page 74)

1) scatop newcat 68 42 10 disc 0 0 disc1 0 0 disc2 0 0
creates a new susercat with
+2: <entrylength> = 68 halfwords
+4: <last used> = 42 (halfwordaddress)
+6: <catsize> = 10 segments
+8: <name of device 0> = 'disc', padded with zeroes, take up 4 words.
+16: <slicelength device 0>
+18: <reference>

the given values unimportant, the proper values will be computed and inserted.

.
.
.

and so on, until the parameter list stops, the rest of the entry will be filled with zeroes.

2) scatop insert japroc 0.0 0 0 6.7 0.1056 0.0 100 200 100 100 20000 FP 100 110 20 1000 20 1000

(Observe that the two first parameters are interchanged, see page 74, hw +2 and hw +4).

An entry named "japroc" is inserted in the catalog. Most values defined by parameters are following the standard claims for a process by "new" command, but three exceptions:

a) The three bases (std, user and max base) are located to a private interval. (HW +22, +24, +26, +28, +40, +42).

b) Hw +30 <size> is expanded from normally 12800 to 20000 halfwords.

c) The bs-claims are somewhat extended as well. (hw +44, +46, +48, +50).

3) scatop change japroc 8 4.7 15 15000 16 myprog
will change the entry "japroc" by following:
word 8 (+16) <buf> changed to 4, <area> left unchanged

word 15 (+30) <size> changed to 15000.
word 16 (+32) and succeeding three words
now contain the name "myprog",
left justified, padded with binary zeroes.

4) scatop change 8 42 10 disc2
As no name is indicated as first parameter to change the action is executed upon entry0 (the susercat describing)
word 8 (+16) is changed to 42, i.e.
<slice length, device 0> is changed to 42 (there is hardly a reason for making this change)
word 10-13 (+20-26) are changed to contain 'disc2', i.e. the name of device 1 will now be 'disc2'. Note that the <slice length> and <reference> in word 14-15 will not be changed!

5) scatop delete japroc
The entry with <user name> = 'japroc' is (marked) deleted.

6) scatop delete
Will actually be executed, but has no effect at all.

7) scatop print japroc
The entry with <user name> = japroc will be printed out

8) scatop print
The catalog describing entry0 (no name indicated)
will be printed out.

9) scatop printcat
All usernames in susercat will be printed out.

10) a job with claims as a "new" process

Scatop insert name 0.0 0 0 7.6 0.1056,
0.0 8388605 8388605 8388605 8388605,
12800 FP 8388605 8388605 0 0 20 800
will correspond to new name run

11) a job with all available bs claims

Scatop insert haps 0.1024 0 0 5.5 0.1056,
0.0 10 20 10 20 10000 FP 10 20

12) Change the job in ex 10 to a job with start adr 60000

Scatop change name 1 0.2 6 60000

Possible errormessages:

Parameter Errors

Preceded by '***scatop: param no <no>:'

Succeded by one of the following:

'action missing'	scatop has been called without indication of an action.
'action unknown'	the parameter following scatop is not one of the known actionnames.
'unknown delimiter'	the parameter has a wrong format.
'too many parameters'	the parameterlist is too long.

(the following occur only when using the change command)

'<integer> index expected'	an integer was supposed as next parameter, should indicate an index.
'index outside entrybounds'	the parameter contains an index greater than or equal to the number of words in an entry.
'forbidden field, must not be changed'	the indicated word cannot be changed by the change command. Try delete followed by insert or create a new susercat.
'one more parameter expected'	after the last index given, a new value to be inserted should follow.

Format of susercat

		Entry 0		All other entries	
+ 0		not used	+ 0	<hash value>	
+ 2		<entry length>	+ 2	<prio>•<command mask>	
+ 4		<last used of entry 0>	+ 4	<process name>	
6		<size of catalog>	6	-	
8		<name device 0>	8	-	
10		-	10	-	
12	dev.0	-	12	<first address>	
14		-	14	<work>.<work>	
16		<slice length device 0	16	<buf>•<area>	
18		<reference>	18	<internal>•<function>	
20		<name device 1>	20	working location	
22		-	22	<max ll>	
24	dev.1	-	24	<max ul>	
26		-	26	<std. ll>	
28		<slice length device 1>	28	<std. ul>	
30		<reference>	30	<size>	
32		.	32	<prog>	
34		.	34	-	
36		.	36	-	
38	dev.2	.	38	-	
40		.	40	<user ll>	
42		.	42	<user ul>	
44		.	44	<entries temp>	device 0
46		.	46	<segments temp>	
48	dev.3	.	48	<entries perm>	
50		.	50	<segments perm>	
52			52	.	
54			54	.	device 1
56			56		

hashvalue = -1 empty entry
hashvalue = -2 deleted entry

<reference> points to the start of the claim list for the device in consolebuffer relative to the start of the console buffer

Number of devices in susercat is installation dependent; standard is all devices.

Explanation of a susercat entry

<hash value>	Inserted automatically. See page 71, example 2.
<prio>	Insert the priority of the process. Corresponds to the prio command.
<command mask>	Only the three bits mentioned under 1, 2 and 3 are used so the standard value of the command mask can be 0. 1. if <first address> is set <> 0 add 2 to the command mask 2. if you want mode=0 add 4 to the command mask = KØRER I MON. 3. if you want "all" bs-resources add 1024 to the command mask
<first address>	Insert the value 0 except you want the process to start in a specific address. Remember to change the command mask. Corresponds to the addr-command.
<buf>	Insert the no. of message buffers. Corresponds to the buf-command.
<area>	Insert the no. of area processes. Corresponds to the area-command.
<internal>	Insert the no. of internal processes. Corresponds to the internal-command.

<function> Insert the standard value 1056. Corresponds to the function-command.

<max ll> Lower limit of the max base.

<max ul> Upper limit of the max base. Corresponds to the project-command.

<std.ll> Lower limit of the standard base.

<std.ul> Upper limit of the standard base. Corresponds to the login-command.

<size> Insert the process size in half-words. Corresponds to the size-command.

<prog> Insert the name fp except you want another program to replace fp. Corresponds to the progcommand.

<user ll> Lower limit of the user base.

<user ul> Upper limit of the user base. Corresponds to the user-command.

<entries temp> Temporary entries.

<segments temp> Temporary segments. Corresponds to the temp-command.

<entries perm> Permanent entries.

<segments perm> Permanent segments. Corresponds to the perm-command.

Other Messages

Preceded by '***scatop: ', followed by one of the following:

'entry does not exist' or
'entry exists already'

Occurs when the desired action cannot be performed due to one of the given reasons, see the survey in chapter 1.

'devi<no> not found in nametabel'

During execution of newcat the devicename indicated for device no <no> was not found/known by the system.

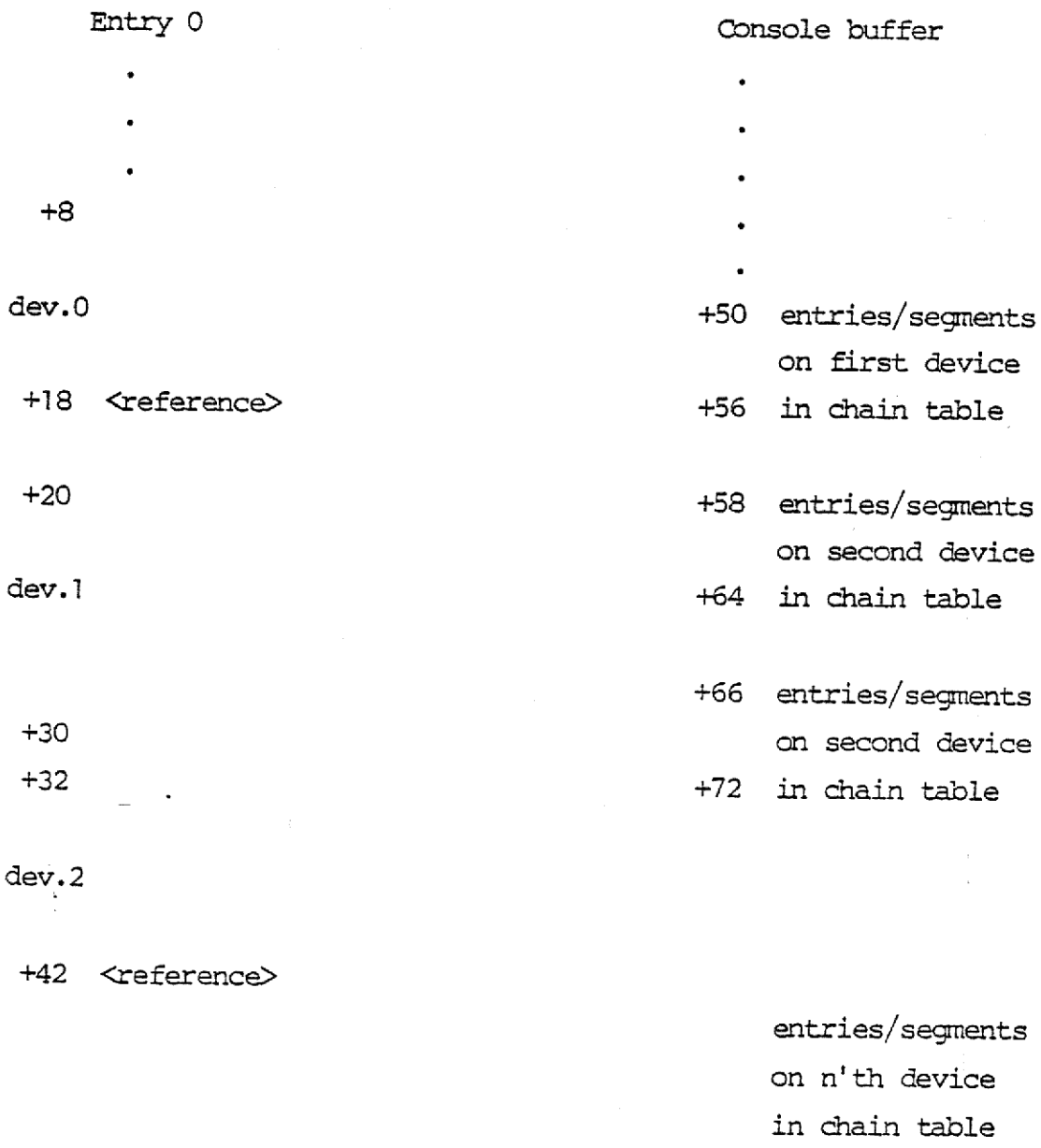
'catalog full, name not found'

Occurs e.g. when the catalog is filled up, and it is attempted to insert a new entry. Delete some entries and insert again, refuse the insertion or create a bigger susercat (and insert all the entries again). Could also occur when e.g. the whole catalog has been searched for an unknown entry. In any case, the operator is warned that the catalog is full (and the name was not found).

'catalog error, hashvalue'

This is a hard system error, save a hardcopy of the run and consult the RC support department.

A possible configuration of susercat could be:



In the consolebuffer there are four words for every possible device in chaintable, containing information about permanent/ temporary segments and entries.

A. REFERENCES

[1] RCSL No 31-D364:
SYSTEM 3 UTILITY PROGRAMS, Part One

[2] RCSL No 31-D379:
SYSTEM 3 UTILITY PROGRAMS, Part Three

[3] RCSL No 55-D1:
RC4000 REFERENCE MANUAL

[4] RCSL No 31-D477:
RC8000 MONITOR, Part Two

[5] RCSL No 31-D476:
RC8000 MONITOR, Part One

B. BACKING STORAGE B.

Backing storage files are created explicitly by means of the program 'set', but they may also be created implicitly in case an FP-command like 'p=algol' or >p=edit q> is executed and p does not exist already (or is protected against writing).

A backing store file occupies two kinds of resources: a number of backing store segments and one catalog entry. A file may change length after its creation, thereby occupying more or less segments.

1. Scope

Some files live as long as the user wants, others are cancelled. Some files are only 'visible' to one user, others are visible to all users. Altogether, we distinguish between the following kinds of scope:

Temporary File

A temporary file is cancelled on request of the user, of the autoload function, of the start up of BOSS and in some cases BOSS will cancel a temporary file not in use. A temporary file is visible from all processes with the same login base.

Login Files

A login file is cancelled on request of the user or by start up of BOSS. A login file is visible from all processes with the same login base.

User File

A user file is only cancelled on the request of the user. A user file is 'visible' from all processes created with the same jobname (user base) as when he created the file.

Project File

A project file is only cancelled on the request of a user under the same project base. A project file is visible from all processes with the same project base as when the file was created.

System File

A system file may only be created and modified by the maintenance staff. The file is visible from all projects enabling programming.

User, project and system files are named in common as permanent files, because they are never cancelled automatically. When a file is born, it will always start out as a temporary file. Later, the scope of the file may be changed by means of the scope command, which is available as an FP-command.

A login file or a temporary file may be created with the name p even if you already have a user file with that name. When you later refer to p, the login file will be used. If the login file is cleared, the user file will reappear. In the same way, a user file may be created even if a project file with the same name already exists. And finally a project file may be created with the same name as a system file.

B.1 Representation of Scope

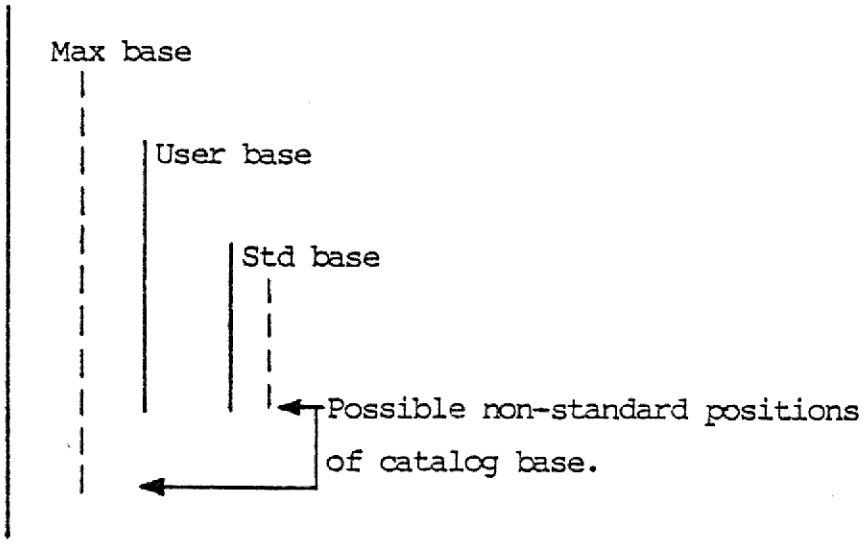
B.1

Each file in the system is described by an entry in the common file catalog (which itself is a system file). The entry associates a permkey and an interval (the so-called entry base) with the file. The catalog key may be 0, 2 or 3 and defines the 'life time' of the entry. The entry base defines an interval of integers from which the file is visible. Together the permkey and the base define the scope.

A process searching for a file in the catalog will have a catalog base (an interval) from which it looks for visible entries. The result of the lookup will be the file with the name wanted and with the smallest entry base surrounding the catalog base (or equal to the catalog base).

Within certain limits the job may change its catalog base and the entry base of files. To each job is associated 3 limiting inter-

vals specified in the user catalog and defining the access rights of the job. Typically they appear like this:



The max base, the std base and the catalog base are parts of the process description in the monitor, which maintains these rules: The job may change its catalog base to any interval which is contained in (or equals to) the max base and which contains the std base (or equal it). The catalog base may also be contained in the std base. The job may cancel or modify a file if the job could change its own catalog base to the entry base of the file. Thus files with an entry base outside the max base or adjacent to the std base are protected against the job. Files adjacent to the std base cannot even be read by the job.

Relative to the process, the scopes of files may now be described in this way:

temp files: permkey 0, entry base equals (or is contained in) the std base.

login files: permkey 2, entry base as for temp files.

user files: permkey 3, entry base equals user base.

project files: permkey 3, entry base equals max base.

system files: entry base sorrounds the max base.

Files with permkey 0 or 2 (non-permanent files) cannot be generated with an entry base surrounding the std base, as this would prevent s from distinguishing processes temporary files from other processes. Thus the std bases may never overlap, not even the std base of the maintenance project (which has the largest possible interval as its max base).

A full description of the file system is found in ref. [5].

C. LOGICAL STATUS BITS C.

The meaning of the logical status bits is:

1 shift 23: Intervention. The device was set in local mode during the operation.

1 shift 22: Parity error. A parity error was detected during the block transfer.

1 shift 21: Timer. The operation was not completed within a certain time defined in the hardware.

1 shift 20: Data overrun. The data channel was overloaded and could not transfer the data.

1 shift 19: Block length. A block input from magnetic tape was longer than the buffer area allowed for it.

1 shift 18: End of document. Means various things, for instance: Reading or writing outside the backing storage area was attempted, the paper tape reader was empty, the end of tape was sensed on magnetic tape. For further details see the description of External processes (see RCSL 31-D478: External Processes).

1 shift 17: Load point. The load point was sensed after an operation on the magnetic tape.

1 shift 16: Tape mark. A tape mark was sensed or written on the magnetic tape.

1 shift 15: Write-enable. A write-enable ring is mounted on the magnetic tape.

1 shift 14: Mode error. It is attempted to handle a magnetic tape in a wrong mode (NRZ or PE).

1 shift 13: Read error. Occurs on card reader. See the description of 'External processes'.

1 shift 12: Card reject or disc error. Occurs on card reader or disc. See the description of 'External processes'.

1 shift 8: Stopped. Less than wanted was output to a file of any kind or zero bytes were input from a backing storage area.

1 shift 5: Process does not exist. The document is unknown to the monitor.

1 shift 4: Disconnected. The power is switched off on the device.

1 shift 3: Unintelligible. The operation attempted is illegal on that device, e.g. input from a printer.

1 shift 2: Rejected. The program may not use the document, or it should be reserved first.

1 shift 1: Normal answer. The device has attempted to execute the operation, i.e. '1 shift 5' to '1 shift 2' are not set.

RETURN LETTER

Title: OPERATING SYSTEM s, Reference Manual RCSL No.: 31-D643

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

42-i 1288

. . . . . Fold here . . . . . .

. . . . . Do not tear - Fold here and staple . . . . . .

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark