A MANUAL OF HELP 3

as developed by

Tove Asmussen, Jørn Jensen, Søren Lauesen,
Per Mondrup, Charles Simonyi, and Jørgen Zachariassen

Edited by

Søren Lauesen

A/S REGNECENTRALEN . COPENHAGEN
1967

1. Introduction 4
1.1 Design considerations 4
1.2 Proposal for a user philosophy 5
1.3 Backing stores 5
2. Principles of operation 6
3. Help 3 information 7
3.1 External form 7
3.2 Internal form 7
4. List of auxiliary programs 8
5. Common characteristics of auxiliary programs 10
5.1 Sum check, inhibition 10
5.2 Texts and sum code 10
5.3 Current input medium 10
5.4 Length of areas 11
5.5 Area conflicts 11
6. Storage areas occupied by Help 3 12
7. Entries to Help 3 12
7.1 Core dumping 13
7.2 Specific entries 13
8. Alarm messages 15
9. Help 3 paper tapes sent to the Gier installations 16
10. Generating a Help 3 system 17
10.1 Generating Main Help + aux. programs 17
10.2 Slip names defining the system 18
10.3 Adding aux. programs to the system in the machine 19
10.4 System punch and binary tapes 20
11. Conventions for programs called by Help 3 21
11.1 Details of program call 21
11.2 Internal entries in Help 3 22
12. System track 23
13. Primitive input program on track 0 24
14. Description of auxiliary programs 25
14.1 Algol 25
14.2 Binin 27
14.3 Binout 28
14.3.1 Format of binary tapes 28
14.3.2 Format of bin 0 tapes 29
14.4 Check 30
14.5 Clear 31
14.6 Compress 32
14.7 Edit 32
14.8 Exit 35
14.9 List 35
14.10 Move 36
14.11 Outparam 36
14.12 Pair 37
14.13 Print 38
14.14 Res 40
14.15 Run 41
14.16 Set 42
14.17 Setsum 42
14.18 Slip 43
14.19 Start 45

15. Programs in paper tape form 46
15.1 Cattap 46
15.2 Check bin 46
15.3 Create new -> old 46
15.4 Punch head kompud 46
15.5 P 1, include algol 47

Appendix A: The Cat system 48
1. Catalog 48
1.1 Format of an area word 49
1.2 Secondary word 49
1.3 Names 50
1.4 Specifications 50
1.5 Free, work and date 50
2. Search 51
3. Init medium 52
3.1 Normal entry and exit 52
3.2 Area descriptions set by init medium 52
3.3 Special entries 53
3.4 Help's current input medium 53
4. Get word 54
4.1 Normal entry 54
4.2 Special entries 54

Appendix B: Help 3 global names 55

Index 56

1. INTRODUCTION.

1.1. Design considerations.

The Help 3 system is designed to help in debugging programs written in Gier Algol 4 or slip, and to enable an effective use of Gier in debug runs and smaller routine runs.

The main points determining the design are:

1) The Help system should supply a strong frame for the existing Gier Algol 4 compiler. This implies the introduction of a storage catalog holding descriptions of data areas on drum and disc. In Help 3 this principle is extended to cover descriptions of peripheral units, programs, and other areas on all kinds of backing stores.

2) Correction of paper tapes is time consuming and dangerous with the existing hardware. In fact it is a bottleneck in debugging Gier Algol 3 programs. One remedy for this is a correction program able to use all kinds of areas for input and output. The slip and algol compilers may then read the input directly from internal areas.

   Safety in correcting is obtained by use of sum code which is treated similarly in all programs.

3) The typical debug run consists of a correction run and an algol translation. Thus in small machines it would be advantageous to replace slip with a correction program.

   This explains why a simple Help language is developed in Help 3.

4) All backing stores may be utilized for program texts and executable programs to keep the drum free for working purposes.

   A dynamic allocation of areas on carrousel and magnetic tapes is not important because the tapes are easily exchangeable and relatively cheap . On the disc a dynamic area structure is introduced by means of the free area from which reservations of named areas may take place.

5) The drum disc requires that the program on track 0 is changed to store and select the group. A flexible condition for dumping of core store into the core image is also introduced for the benefit of running algol programs.

6) It is found inconvenient to make old binary Help 1 tapes directly compatible with Help 3 as they assume absolute storing on the drum and thereby violate the area structure in Help 3.

1.2. Proposal for a user philosophy.

1) It is possible to arrange complicated runs in a way that the operator only needs to insert a sequence of tapes in the reader and type the message r < for each tape.

2) Keep corrected programs inside the machine as far as possible. One method is to use the same program (on paper tape) for several test runs and make a correction tape which is updated before each run. This updating may take place at a flexowriter. A run consists then in reading the correction tape (call of the correction program should be part of the tape). Next read the program tape and store the corrected version in the free area or on a magnetic tape, etc. Finally insert the data tape which may contain call of the algol translator and call of the run program.

When the correction tape grows too big a new paper tape version of the program can easily be generated.

Larger programs may be stored permanently on carrousel or magnetic tape if available. And it may still be a good philosophy to update the correction list and not the program.

1.3. Backing stores.

Help 3 handles the following kinds of backing store (the kind numbers refers to the catalog content, see app. A):

0: Drum which physically may be a disc (called a drum disc). It is organized as a sequence of 40 word blocks. The drum disc is separated into 3 drum areas each with a group number (960, 961 ...) attached. Within Help 3 the tracks are numbered 0, 1 ... 959, 960, 961 ...

1: Disc connected to the buffer and organized as blocks of fixed length (lengths 400 and 640 exists).
The word, disc, will always mean a buffer disc.

2: Carrousel organized as 64 reels of 16 blocks of 512 words.

3: Tape stations with magnetic tapes organized as files containing 400-word blocks (detailed format in app. A).

If these backing stores are used for text input they are called internal media opposed to the external paper tape reader, etc. Disc, carrousel and tapes are called buffer media.

The named areas in Help 3 consist of a number of consecutive blocks determined by the length (in blocks) and some first block information.

Throughout this manual the word track will denote a string of 40 words, on the drum matching one block.

2. PRINCIPLES OF OPERATION.

The Help 3 program is placed on the drum and may be called into action either by HP-button interrupt or by various programmed entries. In outline the following then takes place:

1) The core store and all registers may (depending on core[1023]) be dumped into the core image on the drum.
2) An input medium is selected and Help 3 information terminating with the symbol < is read and converted to an internal form. The following is an example of Help information:

   r, edit, o free, 12 <

   This information list contains 5 elements, the names r, edit, free, the single letter o and the number 12. The unit from which the information is read is called Help's current input unit. In case of the entries hs 1 and hsf 1 no reading will take place as an internal information list is present already.
3) Help interpretes the internal list by means of a catalog which holds information about the names. Starting with the first element of the list the interpretation proceeds thus:
   3.1) If the name describes an output unit then this unit is selected and Help continues the interpretation.
   3.2) If the name describes an input unit (or an area holding text) then the unit is selected as Help's current input unit and interpretation continues.
   3.3) If the name describes a program then the remaining list is moved to the parameter track and the program is called to core and entered. The sense of the parameter list thus depends on the program called.
   3.4) If the list termination is met Help continues with step 2, reading from current input unit.
   3.5) If the element is not a legal name, then an alarm message is given.

In the example above r turns out to be the paper tape reader, edit a program. The result will be that edit is called with the parameters o free, 12 < . According to the description of edit corrections will be read from the current input unit, i.e. the paper tape reader.

As a second example consider the information line

```
free<
```

free designates an area on drum or disc and the current input unit will be this area. Assume that it contains the text

```
1,algol, 10<
  begin
  . . . some algol program
  end
run<
```

Help will now select the lineprinter, 1, and call the algol translator with 10< as parameter list. Algol will read and translate the algol program on the current input unit and output every 10th line on the line printer (because of 1 and 10<). After translation algol will return to Help with current input unit pointing to the text run < in the free area. Help will call the program, run, which will start execution of the latest translated algol program.

3. HELP 3 INFORMATION.

3.1. External form:

<Help 3 information> ::= <list> < |<list> ≤ |<list> <text> < |
<list> <Help number> <

<list> ::= <empty>|<list> <separator>|<list> <single>|
<list> <text> <separator>|<list> <Help number> <separator>

<text> ::= <letter>|<text> <digit>|<text> <letter>|<text> .

<Help number> ::= <digit>|<Help number> <digit>|<Help number> .

<single> ::= <underlined digit>|<underlined letter>|,|.

<separator> ::= ,|<CR>

<digit> ::= 0|1|2|3|4|5|6|7|8|9

<letter> ::= a|b| ... |z|æ|ø|-

The symbol å will always delete the current line and select the typewriter as input unit.
The symbol <10> marks the end of a text string and will cause an alarm.
All symbols not mentioned above are blind. Thus all upper case symbols will be skipped.

Examples: r, <
sl 13, print, b, 105, ra p 0.10.0.39 <
≤

3.2. Internal form.

The Help 3 information is transformed to an internal word by word form easily described in Slip language:

<text> -> t <text>; (0-marked words) spaces and blind symbols are removed.
<number> -> b-marked word with point replaced by slash, e.g. 10..39 -> 10//39 b
<single> -> qq <value of the underlined symbol>, (a-marked), e.g. a -> qq 49,
≤ and < -> qqf,

The termination ≤ will immediately (without interpretation) cause a jump to the primitive input program on track 0. ≤ can be used to reestablish the system if the catalog is spoilt.

4. LIST OF AUXILIARY PROGRAMS.

Beyond the standard content described in appendix A the catalog will contain some or all of the names in the list of auxiliary programs. These names should be reserved for this purpose only.

The following list contains two lengths (in tracks) for each program. The first corresponds to versions without buffer media the second to versions with buffer media.

algol 4(4) Ex: algol, i n s 10 <
Calls the compiler ga4 (or another compiler if specified) with Helps current input medium as program source.

binin 2(2) Ex: binin, image <
Reads a binary tape from current input medium to the drum area given as parameter.

binout 5(5) Ex: binout, image, n 10..1022 <
Punches the domains described in the parameter list in the binary form accepted by binin. Short domains may also be punched in a form which can be read by track 0.

check 9(10) Ex: check, algol, ga 4, sl 13 <
Sum checks either all drum and disc areas in the catalog or only the areas mentioned in the parameter list. A catalog listing can be printed meanwhile.

clear 6(6) Ex: clear, ga 4 <
Removes the catalog item given as parameter. If the area is adjecent to the free area then free will be extended.

compress (entry in check) Ex: compress, a <
Compresses all reserved areas to remove holes left by clear. Also null items in the catalog are removed.

edit 15(18) Ex: edit, i tape 1, o free, 12 <
Reads corrections from current input medium and corrects the text given in the input area, storing the result in the output area.

exit 1(1) Ex: exit, h 715 <
Restores the core and register situation by means of the image and jumps either to the instruction where interrupt took place or to the address given as parameter.

ga 4 about 170(170) Called implicitly by means of algol. Contains the Gier Algol 4 compiler.

list (entry in check) Ex: list, r slip, sl 14 <
Lists catalog items, either all or those mentioned in the parameter list.

move 2(8) Ex: move, work, free <
Moves the area given as first parameter into the area given as second parameter.

outparam (entry in binout) Ex: outparam, binin, image <
Punches its parameter list as a normal Help information list. May be used to punch head and tail on binary tapes.

pair (entry in print) Ex: pair, image, p , free, p 10..1023 <
Compares the two domains described in the parameter list and prints all deviations in the specified form.

print 18(19) Ex: print, p, tape 1, r 1.0.10.399 <
Prints the domains given in the list in the specified form.

res (entry in clear) Ex: res, 26, s 0, sec.image <
Reserves a number of blocks from the free area and sets the description of the reserved area into free. If no length is specified then <booked> blocks will be reserved (see app. A).

run 1(1) Ex: run, sl 14 <
Checks if the drum area given as parameter contains a translated algol program and executes it. If no parameter is given the program in work will be tried (usually the latest translated program).

set (entry in clear) Ex: set, 3, 1, 0, 0, d 0, tape 1 <
Inserts an item in the catalog. All kinds of not reserved items can be set.

setsum (entry in check) Ex: setsum, tape 1, sec. image <
Inserts the check sum of the areas, given as parameters, in the catalog.

slip 23(23) Ex: slip <
The symbolic language input program. Reads program text from current input medium.

start 1(1) Ex: start, 2.6.67, image <
Inserts the number given as parameter into the date item in the catalog. Drum areas in the parameter list will be filled with hsf 2 instructions.

5. COMMON CHARACTERISTICS OF AUXILIARY PROGRAMS.

5.1. Sumcheck, inhibition.

When an auxiliary program (with sum bit present, see app. A, 1.1) is called by means of Help 3 the sum of the instructions is compared with the sum word in the catalog. Only if they agree is the program entered. Long programs which are not transferred entirely by Help will make sum check of their own tail (e.g. slip and ga 4).

Most auxiliary programs are executed with the core store inhibited (for exceptions see run and compress). This means that a HP-button interrupt will not change the core image, and all information about the aux. program called will disappear.

5.2. Texts and sum code.

All programs handling texts use the following format for texts in internal areas:

cell n: qq <end>.3+<char 6>.9+<char 5>.15+ ... <char 1>.39; marks 0
n+1 : qq <end>.3+<char 12>.9+<char 11>.15+ ... <char 7>.39;
...

<end> is 15 except for the last word in the text string where <end> is 10 and unused characters are set to 10.

The programs edit, ga4 and slip treat clear code (28) and sum code (61) in input in the same way: The character following sum code is checked against the sum of the characters read after the last clear code or sum code. Full details may be found in the manual of Gier Algol 4.

In the output string produced by edit, the sum is corrected corresponding to the characters produced. This mechanism may for example be utilized thus:

All programs should, when they are first punched, be provided with one or more sum codes each followed by a space. If later corrections are made with edit an easy check for perforator and tape reader faults is obtained. The first time such a program is corrected a sum error will of course appear, but at that point of debugging the program is wrong with respect to many other things.

5.3. Current input medium.

Many auxiliary programs (e.g. ga 4 and slip) reads further information from Helps current input medium and returns later to Help which continues reading from where the auxiliary program left. The programs edit and run cannot preserve the current input medium and they will select the typewriter when they return to Help.

The detailed format of Helps current input medium may be found in App. A, 3.4.

5.4. Length of areas.

The description of an area contains information about the first block of the area and the length of the area. The length serves two different purposes:

1) If the entire area is to be moved or printed the length tells the number of blocks involved. Apart from this the length is not used when an area is used as input to a program.
2) If a program makes output to an area, the length is used to protect neighbour areas. All auxiliary programs (except slip) perform this check when needed.

Most areas have the same length in case 1 and 2, but exceptions exist:

work case 1: The primary description in the catalog is used when work serves as input area.
case 2: The place allowed for output to work is found in a special cell in the catalog. When the program has finished its output it changes the primary description of work to point to the produced output area. More details may be found in app. A, 1.5.

free case 1: The entire free area is used, but the description in the catalog contains also the variable booked, which may determine the number of blocks reserved in a call of res.
case 2: The entire free area is allowed for output and the program sets booked to the number of blocks produced. More details may be found in app. A, 1.5.

magnetic tape areas case 1: The description in the catalog is used.
case 2: An infinite number of blocks is allowed for output independent of the content of the catalog. When the output is finished an EOF-mark is written and the length in the catalog is changed to the number of blocks produced. Notice that only the programs edit and move can make output to magnetic tapes.

5.5. Area conflicts.

Auxiliary programs handling two areas simultaneously (edit, move and pair) will not accept that the areas are on the same magnetic tape or carrousel. If it is tried an alarm is given.

Special care must be taken if a magnetic tape contains both auxiliary programs and texts. Assume for example that a tape contains the programs edit, ga 4, and the text area, text. The call

```
text, edit<
```

will first position the tape to the beginning of text. Next the tape is positioned to edit, which is called and starts reading from the tape but from a wrong block because edit was called.

On the other hand

```
edit, i text<
```

will work as wanted, while

```
algol<
begin copy text<
```

will work o.k. in pass 1 but cause the message ,pass sum, later because the tape is moved to the text area.

6. STORAGE OCCUPIED BY HELP 3.

The following drum tracks are used whenever Help is called into action:

Main help	track 0 - 10 (0 - 12 in versions where buffer media are treated).
Track 38	working track
Parameter track	track 38 or another track.
Catalog	at least two tracks. Place on drum depends on installation.

A task program is a program executed without the inhibit pattern in core [1023]. See section 7 for further explanation.

If a task program runs and Help is called (HP-button or programmed call the following parts of store are affected too:

core image	26 drum tracks. Place depends on installation.
Core 0 to 9	Only cell 10 to 1023 and registers will be restored when the program, exit, is used to continue the run. For a special possibility of restoring cell 0 and cell 7-9, see the description of exit.

When Help is called during execution of non-task programs, the entire core store is spoilt by Help.

Help 3 uses this part of the buffer store:

Text buffer	cell 0 to 1542. Used when text reading from a buffer medium takes place.
Program buffer	cell 1543 to 3084. Used to call auxiliary programs from buffer media.
Working area	cell 1543 to 4095 is used as working area for some programs (e.g. algol 4, slip, edit, move).

To this list must of course be added the areas on backing store occupied by the different auxiliary programs.

7. ENTRIES TO HELP 3.

In this section only the task entries will be described. The conventions for the internal entries, used in auxiliary programs, may be found in section 11.

7.1. Core dumping.

Track 0 to 1 and, in case of programmed entry, core 0 to 6 take care of the dumping of registers and core. In more detail the following takes place:

1) by[0] is set to prevent further HP-button interrupts. Core 0 to 39 is unconditionally stored on track 38, and track 0 is read to core 0 to 39. This is done by mode 5 (HP-button entry) or core 0 to 6.
2) The content of core[1023] is investigated:

2a) Core [1023] = ann sx V t x MK, full inhibition. Help proceeds to step 3.

2b) Core [1023] = ann sx , tl (x), half inhibition (full inhibition with a-mark). The message

image

is typed. If the operator types a space Help will continue like full inhibition. If the operator types < the action is like:

2c) Core [1023] ≠ full or half inhibition. Core 40 to 1023, track 38 and all registers are stored in the core image. The exit address is stored in the core image, cell 9 address part and marks. If the exit address is 6, 7, 8 or 9, cell 9 will not be changed however, because the entry to Help took place during exit. Programmed entries will also change address and increment parts of core 0 to 2.

Notice that initialisation of core 0 to 9 is done in the auxiliary program exit which takes care of restoring of registers and core store.

3) Core [1023] is set to full inhibition to prevent destroying of the core image in case of HP-button interrupt. by[0] is cleared to release HP-button. Main help is sum checked and sum error causes the message

SUM

(see section 8). Otherwise Help executes the special action corresponding to the entry.

7.2. Specific entries.

HP-button. The following message appears:

<run> <date> <e> <exit address>

The integer <run> is increased by one each time HP-button or hsf 2 entry takes place. <exit address> describes the next instruction to be executed when the program was interrupted. <e> is the letter e if the core store was dumped and the <exit address> can be used to continue the run. <e> is blank otherwise.

<run> is printed on Help's alarm unit (typewriter), the other informations on the normal output medium (typewriter or line printer).

The typewriter is selected as Help's current input medium and reading starts.

Programmed entries:

hsf 2 or an equivalent jump to cell 2. Nearly as HP-button but a p is printed in front of run. The exit address corresponds to the return hr s1. The stored s register has the content which would be obtained in this way.

hs 2 or an equivalent jump to cell 2. Help starts reading from the input medium described in cell -6 to -2, the detailed formats of this may be found in appendix A, 3.4. Exit and stored s as for hsf 2.

hsf 1 or an equivalent jump to cell 1. Help will interprete the list:

```
hsf 1 <
```

This name may describe a patch program or any other suitable area. Exit and stored s as for hsf 2.

hs 1 or an equivalent jump to cell 1 causes a programmed call of an aux. program. The entry sets the boolean hs 1 which is cleared in all other entries and after alarm. When an ordinary auxiliary program returns control to Help (one of the internal entries) this boolean is investigated. If it is set the exit program is called; otherwise help continues reading.

The hs 1 entry interpretes the information list stored in internal form in cell s+2 and on. It is assumed that the core store has been dumped as the list is fetched from the image. The list may not contain more than 40 words including the end mark.

If the called program reads from current input medium the description in cell -6 to -2 will be used as for hs 2 entry. Exit address and stored s as for hsf 2.

Example: Programmed call of: print, p 100..110 <

```
hs 1                ; hs 1 entry
hv a                ; Help returns to this instruction
tprint;             ; 0-marked name
qq 39,              ; p, a-marked single
qqf 100.19+110.39   ; f-marked number
qqf,                ; <, c-marked end mark
```

overflow or any other jump to cell 0. The message

```
overflow <e> <exit address>
```

is printed on the alarm unit. The typewriter is selected as current input medium and Help starts reading.
The <exit address> is given as Raddr corresponding to a floating arithmetic overflow.

8. ALARM MESSAGES.

The following is a list of all alarm messages appearing in Help 3 and the auxiliary programs mentioned in section 4 (except messages from the Gier Algol 4 translator). Unless something else is explained the message is printed in red, the typewriter is selected as current input medium and control is given to Help which starts reading.

If no specific auxiliary program is mentioned in the explanation, the alarm is called general, otherwise it is a special alarm and details may be found in the description of the program.

<number><number>	Error message from Slip which continues reading.
annul	Current line of information annulled by the symbol ø. Appears in Help and edit.
catalog	Sum fails in the catalog. Appears in Help and several auxiliary programs.
char	Alarm character read. Appears in edit.
fault	Parity error on buffer medium even after 4 rereadings. Appears in Help and several auxiliary programs.
full	An information list is too long, a number too big or an area too short. Appears in Help, binin, edit, move, res and set.
image	(black) Half inhibit present when Help is called (see 7.1).
kind	An area word has a not allowed kind. Appears in Help and several auxiliary programs.
label	A magnetic tape has a wrong label. Appears in Help and several auxiliary programs.
length	The area length is outside the allowed range. Appears in res and set.
name	A name conflicts with the catalog. Appears in clear, res, set, check, list and compress.
no clear	The item must not be cleared. Appears in clear.
not present	The area contains no algol program. Appears in run.
overflow	Help is called by a jump to cell 0 (see 7.2).
overlap	Conflict between input and output area. Appears in move and edit.
param	Improper sequence of parameters. Appears in Help and several auxiliary programs.
parity	Parity error on paper tape. Appears in binin, check bin and edit. Edit continues reading after the error.
sum (red)	Sum fails in calling an auxiliary program.
sum (black)	Area sum fails. Appears in check and setsum.
SUM	Main help is destroyed. If the paper tape containing Main help is inserted in the reader and a space is typed the system will be reestablished and Help continues as if no SUM had been printed.
syntax	Syntactical error in Help information. Appears only in Help.
tapesum	Sum fails in a text string. Appears in binin, check bin and edit. Edit continues reading after the error.
termination	Unterminated correction. Appears in edit.
undef	A name is not found in the catalog. Appears in Help and several auxiliary programs.
units	It is tried to compare two areas on the same magnetic tape or carrousel. Appears in pair.
value	Information outside allowed range. Appears in res and set.

9. HELP 3 PAPER TAPES SENT TO THE GIER INSTALLATIONS.

The basic paper tape form of the Help 3 system is a set of slip tapes which may be read by the slip program in some Help 3 system. These slip tapes allow a number of different versions to be generated as explained in section 10. The slip tapes are:

Main help	
init help	Used to initialise the loaded system
P 1	May be read in when the algol merger has worked and will then move the compiler and include it in Help 3 (see 15.5)
algol	
binin	
binout + outparam	
check + compress + list + setsum	
clear + res + set	
edit	
exit	
move	
print + pair	
run	
slip	
start	

In order to read these tapes a basic 1 drum version with buffer media and all auxiliary programs is supplied. The final system or parts of it may be punched in binary form by means of the program, system punch.

These programs are punched in bin 0 form which may be read by the 3 cell input program on track 0.

Tapes in bin 0 form:

basic track 0)	the basic 1 drum version
basic Help 3)	
system punch	(see 10.4).
check bin	Proof-reads binary tapes without changing the backing stores (see 15.2).
cattap	Writes ‹<cattap› labels on tapes (see 15.1).

A few tapes are supplied to facilitate the transition from Help 1 to Help 3:

head bin 0	This tape used as head of a bin 0 tape enables it to be read by Help 1. May for instance be used to read track 0 of Help 3 by means of Help 1.
Punch head kompud	This slip program can punch 3 different head kompud tapes which enable a kompud tape from Help 1 to be read by track 0 of Help 3 (see 15.4).
Create new -> old	This slip program punches track 0 of Help 1 in bin 0 form (see 15.3).

10. GENERATING A HELP 3 SYSTEM.

10.1. Generating Main help + aux. programs.

In order to generate a Help 3 system suitable for the particular machine the following procedure may be used:

1) Put some Help 3 system including slip in Gier and push HP-button.

2) Insert Main help in the reader and type r<. When reading stops after the message

redefine

the slip names mentioned in 10.2 may be redefined before you type l to continue reading. All the slip tapes will terminate with printing the name of the tape.

3) Insert init help in the reader and type l.

4) Select the auxiliary programs wanted and load then one by one. Preceding every auxiliary program the following slip names may be redefined:

d35 = 0, 1, 2 or 3. Describes the medium to which the auxiliary program will be moved after loading.

d36 = 0 or 1. If d36 = 1 then the program will be a reserved area which may be given back to free by means of clear.

d1 determines the drum track to which the program will be loaded. In special situations it may be useful to increase d1.

Each program is loaded to the drum and a primitive catalog item is loaded to the core image. Other primitive catalog items may be loaded like auxiliary programs.

5) After loading the last auxiliary program, you release track 0 and other protected tracks and type

e 10

Init help will now compute the check sums and move the system to its final place thus:

Main help is moved to track 0 and on (displaced d8 tracks).
All primitive catalog items are checked for proper kind if they are reserved. Only program items are treated further.
Reserved programs (d35 = 0 or 1) are moved to first free block and on. The free area is adjusted accordingly.
Other drum programs (d35 = 0) are displaced d8 tracks like Main help (always to group 0). In this case d1 may be useful to avoid moving programs to track 38, etc.
Other disc programs (d35 = 1) are moved to block 0 and on.
Carrousel programs (d35 = 2) are moved to reel 0, block 0 and on.
Blocks are grouped 3 per transport. Only one reel may be loaded.
Tape programs (d35 = 3) are moved to the magnetic tape on station d54. A ⟨<cattap⟩ label is put on the tape and the last program is terminated with EOF-mark.

Each program will after moving occupate a number of full blocks. Unused words in the last of these blocks are filled with zeroes.

Init help will finally move the catalog to the drum and call the generated Help 3 system with a hsf 2 entry.

During loading of the slip tapes various error messages may be given. Here we only mention the alarm

version

showing that a too old version of Main help is used.

During the execution of init help the following alarm messages may appear:

cat length — The primitive catalog is longer than the tracks allowed for the final catalog. Execution continues.

fault — Parity error in writing tape label. Label writing is repeated when you type a space (another tape should be mounted).

format <program name> — Wrong format loaded to primitive catalog. Execution proceeds from the next catalog item.

kind <program name> — Not allowed kind in primitive catalog. Execution proceeds from the next catalog item.

move trouble <program name> — It is tried to move a program to higher track numbers or to use more than one carroussel reel. Execution proceeds from the next catalog item.

program call <program name> — The program cannot be called by Help 3 as too many words would be transferred to core.

10.2. Slip names defining the system.

A full list of the global slip names used in Help 3 is shown in app. B. Here is given a more detailed explanation of the names which may be redefined in the beginning of loading.

Name	init.value	Meaning
d1	100	The system will be loaded to this track and on before moving takes place.
d3	0	d3 = 0 means that the drum will contain the free area, d3 = 1 that it will be on buffer disc.
d4	0	1024 × d52 + d4 is used as the number of blocks on the buffer disc. Only significant if d3 = 1.
d52	0	
d5	39	1024 × d46 + d5 is used as the first free block. This may be changed by init help.
d46	0	
d9	0	First track of final help. Should only be changed for debugging purpose.
d11	38	Parameter track.
d16	294	First track of core image.
d17	3	by-value for standard paper tape reader.
d18	512	by-value for HP-button inhibit.
d19	960	Track group for core image.
d21	34	First track of catalog (catalog always in group 960).
d22	1	No. of 320-track drums. Drum disc has d22 = 30.

d23	4	No. of catalog tracks.
d32	1	d32 = 0 designates that a special work area is used. If d32 = 1 the free area is used for working.
d33	0	First track and number of tracks in the special work area (in group d19). Only significant if d32 = 0.
d34	0	
d35	0	Running kind of aux. programs (see 10.1)
d36	0	Aux. programs reserved (see 10.1).
d41	1	d41 = 1 designates that code for treating buffer media will be included. d41 = 0 otherwise.
d43	17	Help 3 alarm output unit + typewriter input. Is used for all alarm messages and HP-entry run number.
d44	17	Standard output unit + typewriter input. Is used for normal output from aux. programs and most of HP-entry message.
d46	0	See d5.
d50	960	Image group during loading. May be redefined according to d19 in the Help 3 system used for loading.
d52	0	See d4.
d53	400	Block length on the buffer disc.
d54	1	Tape station to which tape programs will be moved.

10.3 Adding aux. programs to the system in the machine.

The slip tapes containing the aux. programs may be used for adding programs to the Help 3 system in the machine. This requires the presence of:

slip, exit, res + set, move and setsum.

The tapes are loaded in the following way:

1) Read Main Help by typing r< and redefine all names to the values used when the Help system was created. Only d1 (first track loaded) may be changed.

2) Select the auxiliary programs to be added and load them one by one. Definitions of d35 and d36 may appear exactly as in 10.1.

If d36=0 (not reserved) the tapes will stop with the message

base, <program name>

Now a few b-marked slip numbers should be typed in, to define the first block of the moved program:

d35=0,	type:	<first block>b
	1	<unit>b <first block>b
	2	<grouped>b <reel>b <first block>b
	3	<unit>b <file>b <first block>b

Continue reading of the program tape.

3) Type e10 after the last aux. program. Each program has loaded a programmed call of res or set, a call of move, and a call of setsum into the core image. When these are executed the programs are included in the system.

This way of loading is espacially useful when new aux. programs are debugged or when the old ones are changed. Even the programs slip and setsum may be changed thus: when res or set is called, an alarm message

appears. Now clear the program from the catalog, set (manually) the new description and exit.

New versions of exit, res + set, and move can only be inserted in this way if the length and specifications are unchanged. Res or set will not be called at all.

Notice that programs cannot be inserted in the middle of magnetic tapes.

10.4 System punch and binary tapes.

Part or all of any Help 3 system in the machine may be punched in track 0 form by means of system punch which is loaded by r<.

System punch asks a list of questions each to be answered by y (for yes) or n (for no).

Question:	Action if y is typed:
track 0	Track 0 is punched. Tear off the paper tape.
main help	Main help is punched. The tape must not be torn off until the catalog question has been asked.
<area name>	The entire area is punched. Only drum and disc areas appear. System punch generates a modified catalog in the working area. It will include all catalog items except reserved not punched. The reserved areas punched will be loaded tight together as if compress had worked. Free is set accordingly.
catalog	The modified catalog is punched. The date cell is skipped in loading to prevent unnecessary calls of start.

System tapes created in this way will be loaded to absolute addresses as they assume an empty machine.

Binary tape versions of programs to be added to the system may be punched by means of list, outparam and binout. For example a tape containing slip may look like:

```
res,23,pis147.12.716.158,slip,20.760.0.3<
binin,slip<
a <the binary slip program>
t<
```

10.4.1 Swop tapes.

At a small 1 drum Gier the full system cannot be present in the machine. Often the users shift between Algol runs (clear, edit, algol, run, ga4 necessary) and slip runs (clear, edit, slip, print, etc. necessary).

In this case the following paper tapes would be convenient:

1) main help + start + exit + clear + edit, but without catalog.
2) algol, run, ga4 and a catalog containing descriptions of 1 + 2.
3) slip + print + binout, etc. and a catalog containing descriptions of 1 + 3.

These 3 swop tapes may be read independently of each other and tape 1 need only be loaded seldom.

11. CONVENTIONS FOR PROGRAMS CALLED BY HELP 3.

Any area with the program bit set may be called by means of Help 3. Thus one way of introducing a new aux. program is to read it by means of slip, set a suitable area description, move the program to that area and perhaps set the sum of the area.

11.1 Details of program call.

The program is called in this way:
1) Core 0 to 9 is initialised (core image is not changed)
2) If a specification word is present (i.e. follows the program name in the catalog), it determines the transfer. Else the program is transferred according to the specification
qq <blocks>.9 + 40d13.19 + 40d13.29; in case of drum
qq 10.9 + 40d13.19 + 40d13.29; in case of buffer medium
3) The specification consists of the 4 parts described in app. A, 1.4. First the specified tracks are skipped and their sum is computed. Next the specified tracks are read to core and their sum is added to the sum of the skipped tracks.
4) If the sum bit is present, it is checked that sum computed + sum in catalog = 0.
5) If the program is a task program (<inhibit>=0), then core[1023] is cleared. A jump to the entry is performed.

If the specification is improper the call program may be destroyed and peculiar things will happen. Res and set will however check that the last cell transferred is outside the call program (last core for program call is approx. 890).

Long programs must check the sum of the tracks not transferred by Help. This sum should be 0 to match the conventions for check which sums the entire area. The area word describing the program area is upon entry still in cell 2c of Help. If the program was called from a buffer medium then get word (entry in -26) is ready for transferring next word from the program area.

The parameter list may be found on the parameter track starting in cell 1. Cell 0 of the track contains the switch hs 1. The contents of the parameter track is present in the core store starting in cell d13, unless these cells are destroyed in the call.

The program may return to Help 3 by means of an external entry (hsf 2, etc.) or by a jump to core[-9] which is the return as an ordinary aux. program. In the last case core[-9] to [-7] must be intact and core[-6] to [-2] must describe the current input medium.

hs 1 is used after return to core[-9] thus:
address part = 0, read from current input medium
= 1, call exit<
= 2, interprete the internal information list on the parameter track.

11.2 Internal entries in Help 3.

Depending on the length of the transferred program, more or less of main Help is left in the core store. The following is a list of some routines in Help which has proved to be of use in the aux. programs.

Each routine is described by the entry name, the slip address, the required intact parts of Help in core store, and a short explanation of the routine.

SIMPLE PRINT, d28 cell 580 to 700

Called by hs d28. Prints Raddr as a positive integer.

INIT MEDIUM, c28 cell 580 to 700

Place of first track of init medium as used in Help. Second track is in 40c28 (see app. A, 3).

FULL ALARM, c61 cell 700 to -1

Prints the text ⟨<full⟩ and makes an alarm return to Help.

PARAM ALARM, c58 cell 750 to -1

Prints the text ⟨<param⟩ and makes an alarm return to Help.

KIND ALARM, c57 cell 750 to -1

Prints the text ⟨<kind⟩ and makes an alarm return to Help.

GET PARAM, c52 cell 750 to -1

Called by pp <address of param> -2, hs c52

Increases p to point to <address of next param> -2. Returns with:

hv s1; and the parameter in R, in case of a help number.
hh s1; and the parameter in R, in case of a single number.
hv s2; in case of end of the parameter list.
hh s2; and the area word in R in case of a name. If the name is not found in the catalog or the catalog is improper no return is made but an alarm is called. Notice that search is called and that d14 to 39d14 is destroyed.

READ INTERNAL, c27 cell 850 to -1

If current input unit is an internal medium then hs c27 will yield in Raddr the next char from the medium. The return is:

ga s1, hr s3

If the medium is drum the latest track selected by search must be the medium drum track.

Cells d14 to 39d14 must contain the track given in core[-2]. If a buffer medium fails an alarm is called.

ADJUST SPECIAL, c74 cell 850 to -1

Takes care of a possible adjustment of free and work as described in app. A, 1.5. The latest call of search must have transferred the first catalog track to d14, ff. R = last block written + 1, M = first block written, indicator = bits 0 to 9 of the area word.

The routine returns to Help 3 (hv -9) after its work.

ALARM PRINT, c24 cell 920 to -1

Is called by qq <text address>, hs c24 or by

hs c24
qq <text address>

Selects the alarm unit, writes CR, red ribbon, the text. Selects typewriter input and calls Help which starts reading.

SEARCH, c to c21 cell -86 to -47

Contains the search track (see app. A, 2).

TEXTPRINT, c23 cell -46 to -36
Called like alarm print but prints only the text on the unit in the by-register. Returns with hr s1.

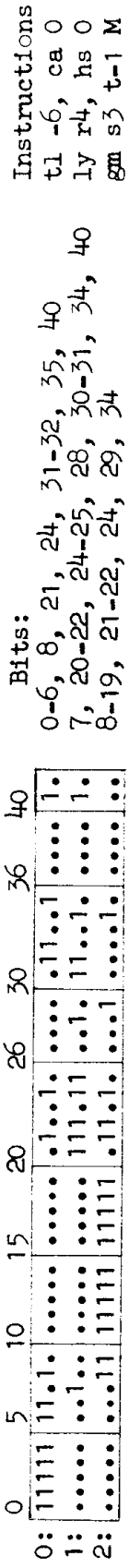
GET WORD, c71 cell -30 to -10
A modified version of get word (see app. A, 4.2).

INITIALISE HELP, -9 cell -9 to -1
Return to Help from aux. program. Help will usually be entered corresponding to an ordinary return from aux. program.
Any other entry may be obtained by
ps <entry wanted> -c41, hh -9

12. SYSTEM TRACK.

The environment description required by the Gier Algol 4 standard procedure ,system, consists of one drum track described in the catalog under the name ‹‹system›.

The track may be generated by means of slip and set in the catalog as an ordinary drum area.



13. PRIMITIVE INPUT PROGRAM ON TRACK 0.

The primitive input program may be called into action in one of the following ways:

1) Help 3 reads the termination <
2) Main Help is destroyed when entry is attempted. This causes the message SUM after which a space will start primitive input.

The sum is checked in two stages. First the sum of track 1 is checked. If it is o.k. track 1 checks the remaining part of Main Help. The paper tapes containing Main Help returns to the point where the sum error was detected or to typewriter input in case 1.

3) If Main Help is destroyed but the sum is still o.k. there is a probability that primitive input may be started by a manual jump to cell 0. In other words, cell 0 on track 0 contains an entry jump to primitive input.
4) The primitive input program may be inserted manually in the core store as described in 13.1.

Primitive input assumes that the first part of the paper tape contains a bootstrap program (e.g. as punched by binout, 0-form) working like this:

```
     Original          2 instr. read     2+6 instr. read
s-1  tl -6, ca 0       tl -6, ca 0       tl -6, ca 0
s    ly r4, hs r-1     ly r4, hs r-1     ly r4, hs r-1
s+1  gm s3 t -1 M      gm s7 t -1 M      bs(r4)     IO
s+2                    hv s              tl 12  V
s+3                                      hv <513-length>  MR
s+4                                      pi 0   t -49
s+5                                      gr <510-length> t 1 MPC
s+6                                      hv s                IKC
s+7  sk [dumped]   This word must not be changed
```

The last version of the program can read instructions with marks. It will terminate reading and jump to the loaded program when an instruction is stored in cell 512.

13.1 Manual set in of primitive input.

1) Set 0 in the by-register.
2) Insert the bit pattern, shown in the margin, into cell 0 to 2.
3) Reset and start with r1 = 0.

14. DESCRIPTION OF AUXILIARY PROGRAMS.

The descriptions attempt to conform to the following scheme:
1) The syntax of the call, written in Backus notation, but on a higher level than section 3. Rather, the syntax describes the internal form of the call.
2) Outline of the program.
3) Semantics of the parameters. The name of the parameter is written as head followed by the semantics of each possible value of that parameter.
4) Further remarks on the program, return conditions, etc.
5) Special alarms and messages. The general alarms are only described in section 8.
6) A few examples, usually demonstrating the special facilities.

Unless something else is mentioned the aux. programs are executed with inhibited core store.

14.1 Algol.

Call ::= algol, <spec list> <
<spec list> ::= <empty>|<spec> <spec list>
<spec> ::= <name of sy-medium>|<name of compiler>|s|i|d|n|<number>

Algol calls some Gier Algol 4 compiler in a way that it will read from current input unit and after a successful compilation return to Help as an ordinary aux. program.

<spec>:

<empty> An empty <spec list> will start the compiler with the name ga 4. Compilation will include text between P.OFF and P.ON. Index check of subscripted variables is generated, and no information is output during translation.

<name of sy-medium> The medium is used as normal out unit, i.e. for possible output of source program, pass information, and pass output (the output produced with KB on). If no sy-medium appears in the list the selected Help output unit is used. If none is selected, normal out will be the standard output unit.

<name of compiler> The name must describe a ly-medium (a transient compiler) or a program area. If no compiler is mentioned, the name ga 4 will be used.

s (skip between P.OFF and P.ON)
The program text between P.OFF and P.ON is skipped. It is included if no s appears in the list.

i (information wanted)
Pass information is printed on normal out unit.

d (disc mode)
The drum disc is used in a mode which may give fewer head movements during translation of large programs. Experimental facility.

n	(no index check) No bound check of subscripted variables is generated.
<number>	Every <number>th line of the source program is copied to normal out unit. If no <number> appears or <number> is 0, no line output will be made.

If the current input unit is typewriter, without explicitly having been specified as such, the source program medium will be reader, and after translation the current input unit will again be typewriter.

If an error has occurred Help will always continue reading from typewriter.

In all other cases the compiler will read from current input unit and return with the last used source program medium as current input unit. The place of the translated program is set in work. The algol compiler will use work-as-output (see app. A, 1.5) for working area during translation.

Error output appears on Help's selected output unit and alarm output appears on Help's alarm unit.

Algol uses the parameter track for storing return information during compilation.

Transient compiler.

If a ly-medium is specified as compiler name, a transient compiler is read. It will use the 12 first tracks (approx.) of the working area for storing part of the translator.

This means that the source program cannot be stored in free, and a special area must be set to hold the result of internal corrections performed by means of edit.

Examples: The typed call:

algol, n <

will read and compile the program in the reader without index check. After compilation Help will wait for typewriter input.

The call:

tape, algol <

starts compiling the program in the area, tape. Help continues reading from tape where the translator left (unless errors occurred or copy has been used).

The second example in section 2 will go wrong if free is used for working area, because the text ,run<, is spoilt during translation. Notice that the program still may be executed by typing run< after the alarm from Help.

14.2 Binin.

Call ::= binin, <name> < |binin, b <number> <

Reads a binary tape to part of the drum area given as parameter:

binin, <name> < The <name> must describe a drum area.
binin, b <help number> < The <help number> will be used as an area word. It must still describe a drum area.

Binin reads from Help's current input medium (which must be a ly-medium) and searches for the symbol ä, heading the binary segments, before any other information is interpreted.

All destination labels will be relative to the first word of the parameter area and the reading will stop when a label > 66 is met. Then the sum and number of characters are checked and binin returns to Help as an ordinary aux.program.

Notice that bin 0 tapes cannot be read by binin.

Special alarms.

full	It is tried to store outside the parameter area.
parity	Parity error on the paper tape.
tapesum	Sum or character check fails.

Examples: see binout.

14.3 Binout.

Call ::= binout, <domain> ... <domain> <
<domain> ::= <base> <form> <interval>
<base> ::= <empty>|<area name>|b <help number>
<form> ::= n|0
<interval> ::= <empty>|
<first block>.<first cell>.<last block>.<last cell>

Punches the domains given in the parameter list thus:
<form>:

n — In normal binary form.
0 — In bin 0 form suitable for reading by track 0. A maximum of 480 words may be punched in this form. Only the first domain may be of 0-form. Notice that track 0 will read the words to the core store so that the last word will go to cell 512.

<base>:
<empty> The preceding base will be used. If no base precedes, then the core image is used.
<area name> Part or all of the area will be punched.
b <help number> The <help number> is used as an area word.

<interval>:
<empty> The entire area described in the base is punched
<help number> Only the cells from <first block> × block length + <first cell> to <last block> × block length + <last cell> are punched. The cell numbers are relative to the base.

Binout punches on the selected output unit. If none is selected the unit 32 (perforator) will be used. Binout returns to Help as an ordinary aux. program.

14.3.1 Format of binary tapes.

The n-domains specified in one call of binout are punched as one binary tape with the format:

<100 spaces> å <label> <label> ...

<label>: Meaning of :
char.< 64 Normal segment: consists of <label> words from the punched area, each punched as 6 characters:
qq <char 6>.6 + <char 5>.13 + ... <char 1>.41
64 Repeat label: consists of an integer, i, punched as 3 characters:
i = qq <char 3>.25 + <char 2>.32 + <char 1>.39
The latest punched word is repeated i times more in the area.

65 Destination label: consists of an integer, i, like repeat label. The following words are printed from the address (relative to base):
i : 1024 × 40 + i mod 1024

66 End label: consists of an integer, i, like repeat label. i contains check information thus:
i : 1024 is the number of characters punched from first label to 66 of the end label.
i mod 1024 is the sum modulo 1024 of the characters from first label to 66 of the end label.

Each domain will thus be punched as one destination label followed by a number of normal segments and repeat labels. The last domain will be followed by an end label.

14.3.2 Format of bin 0 tapes.

A 0-domain is punched in the format:

< < 60 characters bootstrap program> <words> <check>

The first two characters of the bootstrap program yields the number of words in <words> thus:

number of words.9 = qq <char 2>.6 + <char 1>.12

Each word in <words> is punched as 7 characters:

qq <char 7>.4 + char 6>.10 + ... <char 1>.39; marks are <char>.45

<check> consists of the symbol 64 and an integer, i, as in end label for a binary tape. i contains the number of characters and their sum, from the first bootstrap character and to 64 of <check>.

Examples: binout, n <

This call punches the entire core image. The resulting tape may be read to the core image in any Gier thus:

r, binin, image<

The call: r, binin, b 2..150< will read the same tape to track 150 and on.

The call: r, binin, b 150 < gives a full-alarm because the length of the area is shorter than the tape content.

Assume that the core image cell 10 to 100 contains a program working like binin called thus:

r, binin, b 20..0<

It is then possible to punch a tape to be loaded by track 0 to absolutely addressed parts of the drum, for example will

binout, 0 10..100, b 0, n 150.5.175.39 <

create a tape to be loaded to track 150 cell 5 and on.

14.4 Check.

Call::= check, <form> <name list> <
<form>::= <empty>|a|n|r
<name list>::= <empty>|<name> <name list>

Performs sum check on areas described in the catalog and prints the catalog items if wanted.

<form>:

<empty>	The items are not printed.
a	All information in the item is printed in a way resembling the parameters to res or set. The notation r and x is used for the reserved bit and the special bit.
n	(names only). Only kind and names in the item are printed.
r	(res or set form). Prints in a form which may be used as Help 3 information (a call of res or set intended to re-establish the catalog item). Prints on the selected output unit. If none is selected the unit 32 (perforator) is used.

<name list>:

<empty>	The entire catalog is printed according to <form>. Areas on drum or disc with sum bit present are sum checked.
<names>	The items in the list are sum checked if the sum bit is present and printed according to <form>.

The selected output unit is used for printing. If none is selected the standard output unit is used, except for the form r.

No-alarm return.

Check returns to Help as an ordinary aux. program except if x is selected as output unit. The return will then take place as described for res; in case of a programmed call with the register contents:

R = area word of last processed item in <name list>.
M = specification word, or 0 if none is present.
indicator =
LZA, name found.
LZB, sum o.k. or sum not computed.
LRB, sum computed.
bits 3 to 7, bits from the area word (with reserve bit set to 0 in case of any special item).

Special messages.

name	Name in list not found in catalog.
sum (in black)	Sum of area fails. Check proceeds.

14.5 Clear.

Call ::= clear, <name>|clear, <name>, <help number>

Clear removes the entire item, containing <name>, from the catalog, if the following conditions all are fulfilled:
1) <name> is found in the catalog
2) The area is neither free nor work (i.e. has no special-bit)
3) The area is reserved or has kind ≠ 0 or <help number> is specified with a value matching the area word in catalog.

If the area is reserved and adjecent to first free block then the area and adjecent not used areas are given back to free and booked is set to zero.

Clear will change the item to a null item except if it was the last item in the catalog.

No-alarm return.

Clear returns to Help as an ordinary aux. program, except if x is selected as output unit. The return will then take place as described for res, and with the following values in case of a programmed call:

Raddr = 0 o.k., item removed
1 name not found
2 clear not allowed

Special alarms.

name	Name not found in the catalog
no clear	Condition 2 or 3 above violated

Example: The calls:

```
algol<
begin
...
end
clear, ga4<
run<
```

will translate the algol program, remove the translator item from the catalog, and run the translated program.

If ga4 is reserved and adjacent to free (as will be the case at small installations), the area allowed for program data is in this way increased by approx. 170 tracks.

14.6 Compress (entry in check).

Call::= compress, <form> <
For the syntax of <form>, see the description of check.

Removes all null items from the catalog and displaces all reserved areas to remove holes left by clear. Meanwhile all items are printed according to <form>, as described for check. The items appear as they are left by compress (except free, which is adjusted later).

When compress has finished, the number of words used in the catalog and the final size of free are printed.

It is important that compress is not interrupted during its work. To protect against mistakes the by-inhibition is set, and half inhibition is set in the core store.

Return. Compress returns as an ordinary aux. program. If x is selected as output unit and compress is called by a program the stored R register in the image will be set to

qq <words used in cat>.23 + <free size>.39

before the normal return takes place.

14.7 Edit.

Call ::= edit, <input area> <output area> <space tracks> <changes> <
<input area> ::= <empty>|i <name>
<output area> ::= <empty>|o <name>
<space tracks> ::= <empty>|<number>
<changes> ::= <empty>|l <char> <type> <changes>
<char> ::= <single>|<number>
<type> ::= c|s|b|a|e|r<char>|n

Edit reads a list of corrections from current input medium, scans the text in <input area>, corrects it and puts the result in <output area>. Then the typewriter is selected as current input medium and control is given back to Help which will start reading.

<input area>:	
<empty>	The paper tape reader will be used for input.
i <name>	The <name> must describe a ly-medium or a text area.
<output area>:	
<empty>	The perforator will be used for output.
o <name>	The <name> must describe a sy-medium or a text area.
<space tracks>:	
<number>	<number> tracks filled with spaces will be output to <output area> preceding the corrected text.

<changes>:

<empty> The standard character table will be used to interprete the symbols in the correction list and <input area> thus:
UC and LC have the conventional meaning. <10> will unconditionally terminate the input.
Skipped characters: TAPE FEED and ALL HOLES.
Blind characters: SPACE, END CODE, CR, BLACK R., RED R., P.OFF and P.ON will not be used in string matching.
Alarm characters: unused characters (45, 46, etc.) will give an alarm if met.
End marks: STOP CODE will terminate the input when the last correction has been inserted. Before that point it will be treated as a blind character.
Replaced characters: The symbol å in the input will be replaced by STOP CODE and treated thus. å in corrections from typewriter is treated in a special way, see later.
Normal symbols: All other symbols will be treated as normal, case dependent symbols.

l <char> <type> The character table will be changed to treat <char> in another way. <char> may be designated either by the character underlined or by its decimal value.

<type>:

c The character will be treated as case free, i.e. it will be used in string matching but with same value in upper and lower case.

s Treated as a skipped character
b - - a blind -
a - - an alarm -
e - - an end mark
r <char> The character will be replaced by <char> and treated thus. Notice that this may cause endless loops like:
l 13, r 11 , l 11 , r 13

n Treated as a normal symbol.

The effect of TAPE FEED, UC, LC and <10> cannot be changed.

In output on external media superfluous case shifts are removed and SPACE and CR take the case of the preceding symbol.

14.7.1 Correction list.

<correction list> ::= <end mark>|<correction> <correction list>
<correction> ::= <copy to> . <insert> . <skip to> .

<end mark> is an end mark character, e.g. STOP CODE. The 3 strings in <correction> may be empty and the corresponding action will then be empty. The correction:

. a b C . .

will thus immediately insert a b C without skipping anything. The corrections are executed one by one during the scan of the input text. Edit inserts case shifts in the output as needed.

Notice that only TAPE FEED's may separate _ (underlining) and . (point). The string: _ UC LC . will thus not be recognized as string termination but is treated as the two characters _ and . .

Edit stores the corrections in the top end of the working area.

If both corrections and text are read from the paper tape reader the program will wait for a space to be typed after reading the corrections.

14.7.2 Typed corrections to edit.

If the typewriter is used as input unit for corrections then 4 case shifts will cause edit to type <. It is then possible to type:

<number>. designating a lower case symbol with the value <number>.
<number>: designating an upper case symbol

ø from typewriter is always treated thus: before a correction starts ø will terminate the correction list. Inside a correction ø annuls the current line.

14.7.3 Messages and alarm situations.

Edit types the number of corrections read (in red) when correcting starts. If the corrections or the text for some reason do not terminate as expected the RESET and then the START button should be pushed. This will work as an unconditional end mark for the corrections or the text. The number of executed corrections are typed when edit terminates.

Special alarms:

char	Alarm character read. Edit skips the character and continues.
full	The length of <copy to> or <skip to> exceeds 400 characters or the corrections exceeds the working area.
overlap	The output text exceeds the space allowed for it, or the output overtakes the input (input and output area identical), or the input and output area are on the same magnetic tape.
termination	Unterminated correction.

Examples: A program on paper tape containing many STOP CODE's may be read to the free area thus:

```
edit, o free, l 11,b <
ø
a
```

The paper tape reading must be terminated by means of RESET and START and then booked will be set to the number of blocks written.

Assume that the text in free contains

```
table[ALFA]
```

which should be corrected to

```
table[alfa + 1]
```

This may be done by the call:

```
edit, i free, o free, l 11 , b <
table[. alfa + 1 . FA .
ø
a
```

The copying will terminate on the <10> previously inserted as end of the program text.

14.8 Exit.

Call ::= exit < |exit, <number> < |exit, h <number> <

The exit program re-establishes the core store and all registers by means of the core image. Then a jump is performed depending on the call of exit:

exit< Jumps to the exit address set when the image was created.
exit, <number> < Jumps to the left instruction in cell <number>.
exit, h <number> < Jumps to the right instruction in cell <number>.

Before the core store is re-established cell 0 to 9 of the image may be initialised. This can take place in two ways:
1) If cell 8 of the image contains

```
vy  1  t  511
```

and cell 9 contains (x are arbitrary numbers):

```
qq  x  ,  hv (rx)  or  qqfx  ,  hh (rx)
```

then only cell 1 to 6 will be set as below.
2) In all other cases will cell 0 to 9 be set to:

```
[basic dump program]
0:  it  -1     , it  -1     ; overflow entry:
1:  it  0      , pt  1      ; hs 1 and hsf 1 entries:
2:  pa  1      t    960     ; drum disc version: gg 1 , gk 2
3:  gk  2      , vy d18+d43; drum disc version: vy d18+d43, vk 960
4:  vk  38     , sk  0      ; store cell 0 to 39, get track 0 of Help;
5:  vk  0      , lk  0      ; both entry and exit takes place in cell 6.
6:  vk[group] , vk[track] ; reestablish group and track
7:  qq                 N   ; used by patch programs, etc.
8:  vy  0      t -d18 -1    ; release HP-inhibit.
9:  qq[exit addr], hv(r)    ; or : qqf, hh(r)
```

Example: It is sometimes useful in debugging to stop immediately before the exit jump. This may be achieved thus:

```
slip <          ; call slip
i = 7           ;
zq              ; insert stop
vy 1 t 511      ; prevent initialisation of cell 7
e               ; exit
```

The exit program can only be placed on drum and a message will be given during loading if other media are attempted.

14.9 List (entry in check).

Call ::= list, <form> <name list> <
For the syntax of <form> and <name list> see check.

List works exactly like check, except for the sum check, which is never performed.

14.10 Move.

Call ::= move, <base> <base> <tracks> <
<base> ::= <area name>|b <help number>
<tracks> ::= <empty>|<number>

The program moves the words in the area given as first base to part of the area given as second base.
<base>:

<area name>	The description in the catalog is used.
b <help number>	The <help number> is used as an area word.

<tracks>:

<empty>	All the words in the first area will be moved.
<number>	<number> tracks (of 40 words) will be moved.

It will be checked that the moved words do not extend farther than the second area.

A full number of blocks in the second area will be changed. Not used words in the last block will be filled with zeroes.

Special alarms:

full	Second area too short.
overlap	It is tried to move from a magnetic tape or carrousel to the same tape or carrousel. Or it is tried to move from a disc area to an overlapping area.

Examples:

```
move, work, free <
```

will move the words stored in work (not the entire work area, see 5.4) to the beginning of the free area and set booked to show the number of blocks written.

```
move, free, tape, 6 <
```

will move 6 × 40 words from free to one block in the tape area, tape, and fill the last 4 × 40 words in the block with zeroes. Then an EOF-mark is written and the length 1 is inserted in the catalog description of tape.

14.11 Outparam(entry in binout).

Call ::= outparam, <arbitrary parameter list> <

Outparam punches the parameter list in a form suitable for reading by Help 3. The list is punched on the selected output unit. If none is selected the unit 32 (perforator) will be used.
Outparam returns to Help as an ordinary aux. program.

Example: The call binin, image < may be punched as head of a binary tape thus:

```
outparam, binin, image <
```

14.12 Pair (entry in print).

Call ::= pair, <pair list> <
<pair list> ::= <empty>|<pair> <pair list>
<pair> ::= <domain> <domain>
For the syntax of <domain> see the description of print.

Pair compares the two domains described in <pair> and prints the deviations in the <form> contained in the second <domain>. The length of the compared areas is also taken from the second <domain>.

Pair prints on the selected output unit. If none is selected the standard output unit is used.
The next <pair> is treated in the same way, and so on until the end of the list.

Pair returns to Help as an ordinary aux. program.

The layouts - , l , and w have no effect in pair. Apart from this the rules of print hold as if the second <domain> was printed.

Special alarms:
units it is tried to compare two areas on the same magnetic tape or carrousel.

Example: pair, p 10.., sl 15, p 10..1023 <

This call compares the core image with the area sl 15. The comparison is extended from cell 10 to 1023 and deviations are printed in program form.

14.13 Print.

```
Call ::= print, <domain list> <
<domain list> ::= <empty>|<domain> <domain list>
<domain> ::= <base> <layout> <form> <interval>|c
<base> ::= <empty>|<area name>|b <help number>|b|<base> <base>
<layout> ::= <empty>|m <list of help numbers>|l <number>|w <number>|
             -|<layout> <layout>
<form> ::= p|i|r|f|g|t|pn|ix|rx|fx
<interval> ::= <empty>|<help number>|a|c|<help number> <number>
```

Print prints the contents of the storage sections given in the <domain list> on the selected output unit. If none is selected the standard output unit is used. Print returns to Help as an ordinary aux. program.

<base>:

<area name>	Part of the corresponding area is printed.
b <help number>	The <help number> is treated as an area word describing the area to be printed.
b	Part of the buffer store is printed.
<empty>	The latest <base> is used. If none has occurred part of the core image is printed.

<layout>:

m <list of help numbers>

The numbers are treated as a list of 10-bit bytes each specifying a number of bits within a word thus:

byte = 0	blind specification.
1≤byte≤40	the following 'byte' bits are printed as an integer.
101≤byte≤140	the following 'byte-100' bits are skipped.

The bits are specified from left to right in the cell. m has only significance in connection with g-form.

l <number>	At most <number> positions are printed on one line. This deletes the effect of any preceding w-layout and vice versa.
w <number>	<number> words are printed on each line.
-	No cell number is printed in the beginning of the lines.
<empty>	The latest m, l, w and - layouts are used. Print starts parameter scanning with printing of cell numbers active and as if the following layout list had been present: m 10.10.10.10, w 1

<form>:

p	program form. Notice that w-layout also works in this case.
i	integer form corresponding to the algol layout ⟨-dddddd⟩ or ⟨-ddddₙd⟩
r	real form corresponding to ⟨-d.dddₙ-ddd⟩
f	fractional form (unit in position 0) corresponding to ⟨-d.dddₙ-ddd⟩
g	group form corresponding to the latest m-layout.
t	text form assuming a Help 3 text format.
pn	program form without effective address value for r-marked instructions.
ix	integer form with extra accuracy, corresponding to ⟨-ddd,ddd,ddd,ddd⟩
rx	real form with extra accuracy, corresponding to ⟨-d.ddd,ddd,dddₙ-ddd⟩
fx	fractional form with extra accuracy, corresponding to ⟨-d.ddd,ddd,dddₙ-ddd⟩

<interval>:

<empty>	the entire area is printed.
<help number>	if the area is core image or buffer store the number is treated as <first cell> .. <last cell>, else the number is treated as <first block>.<first cell>.<last block>.<last cell> The words in the area from relative cell '<first block> × block length + first cell' to cell '<last block> × block length + last cell' are printed.
a	the arithmetic registers (R and M) are printed.
c (or <domain> = c)	all the control registers are printed in a special way independent of the form and layout.
<help number> <number>	<help number> works as above and <number> is used as the starting value of the printed running address. If <number> is not specified the running address starts with <first cell>.

Each <domain> is printed as a head stating the used base and a list of lines containing the words. Each line starts with two numbers (unless - has appeared), the first is the running address the second the relative address within the block.

Example: 1, print, p 10..20
c ra
tape, m 4.132.4, w 10, g 10.399 <

This call prints on the line printer the instructions 10 to 20 in the core image, the control registers, R, M, and RF, the first 11 blocks of the area, tape, with 10 words a line and only the first and last 4 bits of each word printed.

14.14 Res (entry in clear).

```
Call ::= res, <length> <bits> <entries> <
<length> ::= <number>|<empty>
<bits> ::= p|i|s <help number>|d <help number>|<empty>|
           <bits> <bits>
<entries> ::= <name>|<name> <help number>|<entries> <entries>
```

Res reserves a part of free starting with first free block and inserts a description of the reserved area in the catalog. Free is updated accordingly.

<length>:

<number>	<number> tracks are reserved.
<empty>	<booked> tracks are reserved (see app. A, 1.5).

<bits>:

p	sets the program bit in the area word.
i	- - inhibit - - - - - -
s <help number>	- - sum - - - - - - and inserts <help number> as a secondary word.
d <help number>	inserts <help number> as a secondary word.

<entries>:

<name>	<name> is inserted in the catalog item. It is checked that <name> is not already in the catalog. If p appears it is checked that the program may be called by Help 3 (see 11.1).
<name> <help number>	<name> is treated as above and <help number> is inserted as a specification word. If p appears it is checked that the specification allows the program to be called by Help 3 (see 11.1 and app A, 1.4).

Booked is always cleared by res. The return to Help is as an ordinary aux. program except when the output unit x is selected.

No-alarm return.
If x is selected, res will return as an ordinary aux. program, even if one of the special alarm terminations should have occurred. If res is called from a program (hs 1 ≠ 0), one of the following values is set in Raddr (i.e. in the stored R in core image) before return:

Raddr = 0 o.k., the item is inserted
1 error in param value
2 name already in catalog
3 catalog filled
4 length ≤ 0 or too large

In case 1 to 4 an error message will appear on the selected output unit (another than x may be effective because x has the mask -1). Notice that the general alarms (e.g. param) still uses the alarm return.

Special alarms.

full	catalog filled
length	area length outside allowed range, or free too small.
name	<name> already in catalog.
value	information outside allowed range.

14.15 Run.

Call ::= run< | run, <name> <

Run starts execution of a translated Gier Algol 4 program. The execution terminates with a hsf 2 entry to Help.

run< The program described in work is executed (usually the latest translated program).

run, <name> <
<name> must describe a drum area containing the translated algol program to be executed.

Run checks cell 2 on the first track of the area to see if it contains an algol program.

If part of the algol program is placed in the core image the execution takes place with half inhibit in core[1023] to allow the operator to choose between dump of the core store (type < after ⟨<image⟩) or rerun of the program (type space after ⟨<image⟩). In other cases the core store is not inhibited.

As the algol program changes cell 0 (spill action), run will set vy 1 t 511 in cell 8. This prevents initialisation of cell 0, if the HP-button is used during run. Cell 8 is set back again when the run is completed in the normal way.

If the executed program is part of free the execution is performed with a smaller free area to prevent destroying the program itself.

Special alarm.

not present	The area contains no algol program.

14.16 Set (entry in clear).

```
Call ::= set, 0, <length> <first block> <bits> <entries>|
         set, 1, <length> <disc unit> <first block> <bits> <entries>|
         set, 2, <length> <grouped> <reel> <first block> <bits>
         <entries>|
         set, 3, <length> <tape unit> <file> <first block> <bits>
         <entries>|
         set, <kind 4 to 7> <rest of area word> <entries>
```

For the format of <bits> and <entries>, see res. All remaining syntax elements are help numbers.

Set inserts a catalog item describing a non-reserved area. The first parameter is the kind of the area.

<length>: Set checks as far as possible that the area is inside the capacity of the backing store concerned. In case of tape (kind 3) a maximum of 8000 blocks are allowed and <file> and <first block> must fit into the area word format (see app. A, 1.1).
<first block>: First block of the area. See remarks on <length>.
<bits> and <entries> are treated as in res.
<disc unit>: unit number, between 8 and 15.
<grouped>: number of blocks per transport, 1, 2 or 3.
<reel>: first reel of the area, between 0 and 63.
<tape unit>: the station number, between 1 and 6.
<file>: file number between 0 and 31.
<rest of area word>: the last 32 bits of the number are inserted in the area word.

Set returns to Help in the same way as res, and with the same values in Raddr if x is selected.

14.17 Setsum (entry in check).

```
Call ::= setsum, <form>, <names> <
<names> ::= <name>|<name> <names>
```

The sum bits are set and the complementary sums of the areas in <names> are computed and set in the catalog items. The items are printed according to <form> as described for check. Return and no-alarm return takes place as for check.

Special messages.

name	Name not found in catalog.
sum (in black)	No secondary word in the catalog. Setsum proceeds.

14.18 Slip.

Call ::= slip < |slip, <group>.<first track>.<first cell>.<start> <

Only deviations from 'A Manual of Gier Programming II' will be described. The syntactical elements defined there will be used below.

Slip starts reading from current input medium and may assemble instructions into the 3 drum area given by <group>:

Call:
Slip< Slip starts loading as if the following block head had been read:
[select group <d19>]
b k= <d16> , i=0 ; d16 and d19 are names in Help
I=10
n
m

slip, <group>.<first track>.<first cell>.<start> <
Slip starts loading thus:
[select group <group>]
b k=<first track>, i=<first cell>
I=<start>
n
m
Slip may only load to one group of the drum disc in each call.

List of new syntactical elements:

< <pre-def. address>
is syntactically a new <definition line>
Slip tests <pre-def address> (by means of a bs-instruction). If 0 < <pre-def address> then the following text is read in normal way, else slip goes into skip mode during which only × and > are treated.

× may appear everywhere outside comments in read mode and is always treated in skip mode.
Slip shifts mode from reading to skip mode or vice versa.

> has the same syntactical position as × . Slip starts reading in normal way. The slip condition:

```
< d1 - 3
text 1
×
text 2
>
```

has some similarity to an algol construction:

```
if 0 < d1 - 3 then load (text 1) else load (text 2);
```

But notice that slip conditions cannot be nested and that × and > will be interpreted even in comments and texts in skip mode.

< Not used, i.e. works as s

e has the usual significance. If e causes an exit from slip the auxiliary program, exit, will be called with suitable parameters.

g is a new <control code>.
The current input medium will be the latest used internal medium. If none is present a 10-error appears.

h is a new <control code>. Causes a return to Help 3 which continues reading from the current input medium. Slip forgets all about the loaded program.

i <arbitrary string without CR> CR
is a new <line>.
The text string will be printed on the typewriter during loading.

j is a new <symbolic address> like i and k.
j is the current relative address on the track to which slip loads. $0 \leq j < 40$

k <character list>.
is a new <constant line>.
<character list>::= <empty>|<integer> <term.> <character list>|
t <arbitrary string not including ;>; <character list>|
T <arbitrary string not including ;>; <character list>
<term.> ::= ,|<line end>
The <character list> is packed into one text string:
<integer> is packed as the symbol with this value. t . . .; is packed as the string starting in LC. T . . . ; is packed as the string starting in UC.

p <arbitrary string without '<' > <
is a new <control code>.
The string between p and < is assembled as a text and is looked up in the catalog. The resulting description must be a text area and the area will be selected as current input unit. Notice that space, etc., should not appear in the string.

<sum code> <character>
may appear everywhere. <character> is checked in the same way as Gier Algol 4 and edit performs sum check. Apart from this the construction is blind.

t <arbitrary string without;>;
The stored format of the text is changed to conform to Gier Algol 4 and Help 3. Superfluous case shifts are removed but space, CR, etc. are regarded as case dependent symbols.

w <pre-def. address>, <pre-def. address>
is a new <definition>.
The two addresses are put in the catalog in the area word for work. The first is inserted as <blocks written>; to the second is added the group information and it is inserted as <first written>.

x Not used, i.e. works as s.

z Not used, i.e. works as s.

New error messages from slip.

9 <serial address>
Too many labels or blocks. May appear at p which uses 40 words of the label table. Typewriter is selected as input unit.
10 <serial address>
Illegal use of p or g. Typewriter is selected as input unit.
<bit pattern> sum
The <character> following <sum code> does not match the character read. <bit pattern> shows the corresponding correct <character>. Slip continues reading.

Example: The following slip program will load the integers 1, 2, 3 ... into cell d1 to 300 of the core store.

```
b d1          ;
i=0           ; load the following text into core image
t d=d+1
  i=d+d1-1
  qq d.39
  <i-300,pimage<
  >1;         ; end of text.
i=20          ;
. . .         ; Some ordinary slip program.
d1:           ; The integers will be loaded from here and on.
d=0,pimage<   ; Read the text in image. It will repeat reading itself
e             ; until cell 300 is reached.
```

14.19 Start.

Call ::= start, <param list> <
<param list> ::= <empty>|<param> <param list>
<param> ::= <number>|<area name>

Start clears cell 1023 (inhibit) and clears the address part of cell 8 in the core image so that the next call of exit will initialise cell 0 to 9. Depending on the parameter list the following takes place:
<param>:
<number> The 10 first bits of <number> are inserted as run number , the remaining bits as date.
<area name> The area must be a drum area. It will be filled with hsf 2 instructions.

Start returns to Help as an ordinary aux. program.

Example: start, 9.6.67, image <
This call sets the date to 9.6.67 and fills image with hsf 2.

15. PROGRAMS IN PAPER TAPE FORM.

The following programs must be read in each time they are needed.

15.1 cattap.

Read by r< or < . Prints the message ‡<<cattap› and waits for a digit followed by comma to be typed. Writes a ‡<<cattap› label and an EOF-mark on the corresponding tape station. After that it is ready for a new station number to be typed.

If a parity error occurred during writing the message ‡<<fault› is given.

15.2 Check bin.

Read by r< or < . Prints the message ‡<<check bin› and waits for a space. It will then read and check the first binary section of the tape in the reader (both bin 0 tapes and normal binary tapes may be checked).

If the tape was o.k. the program returns to the start situation. Otherwise an alarm is given. If reading stops (because of a parity error) or the reader is emptied, the program may be restarted by pushing RESET and START.

15.3 Create new -> old.

Read by Slip in the old Help 1 version. Waits for a space to be typed and will then punch the current track 0 in bin 0 form. Returns then to the start situation.

The punched tape enables the transition from Help 3 to Help 1.

15.4 Punch head kompud.

May be read by 1 in Help 1 or r< (and slip) in Help 3. After a moment the tape stops. It is now possible to redefine d = <number of drums> before 1 is typed. d = 1 from the beginning.

After loading the program punches the head kompud tape corresponding to the d-value. Then it waits for a symbol to be typed.

If c is typed, the program starts copying from reader to punch (usually a kompud tape is copied). Else a hsf 2 is performed.

15.5 P 1, include algol.

The aux. programs slip, exit, and (if a permanent translator is wanted) res + set, move and setsum must be present when P 1 is used. Further the drum must contain a Gier Algol 4 translator (not necessarily a drum version).

P 1 is loaded by r< . During loading a number of further information must be typed in, to define the actual place of the translator, the place to which it should be moved, and the name of the final area (usually the name tga4; should be typed).

Each piece of information to be typed is explained by a message. In most cases the information resembles what is required when programs are added to the system (see 10.3).

If the kind (d35) is defined to sy-medium (6) the two paper tapes of a transient compiler are punched. This requires however that the translator on the drum is a drum version.

Alarm messages from P 1.

pass sum	Translator not present (only tested in case of a transient compiler).
translator medium	Translator version does not match the final place typed in.
wrong kind	d ≠ 0, 1 , 2, 3 or 6.

APPENDIX A: THE CAT SYSTEM.

The Cat system consists of the catalog tracks and the tracks necessary for basic handling of the catalog. The Algol translator requires only the presence of some Cat system to make full use of the catalog facilities, but usually Cat will be part of the Help 3 system. The four constituents of Cat are then placed on the drum thus (the Slip names are defined in Help 3):

1) Catalog	tracks	d21 to d21 + d23 - 1
2) Search	track	c64 - 1
3) Init medium	tracks	c63 and 1c63
4) Get word	track	c64

Either the drum or the disc (unit 8) is chosen to be the free store in which reservations of areas can take place. Free kind is 0 if the free store is drum, 1 otherwise.

1. CATALOG.

The catalog holds descriptions of peripheral units and named areas on the various backing stores. Named constants may also be placed in the catalog. The last word on each catalog track is a c-marked check word satisfying

sum of words on track + sum of marks on track.9 + c.9 = 0

where c is the catalog track number, 1, 2, 3 ...

The string of catalog words, omitting the check words, consists of a sequence of catalog items, the first of which have standard names:

Items in catalog:	describes:
free	the free area in the free store
work	a working area on the drum
date	the current date and run number
image	the core image on the drum
r	the standard paper tape reader
t	the typewriter input unit
p	the perforator output unit
l	the lineprinter
w	the typewriter output unit
x	no-alarm output unit, used in check, etc.
<optional items>	peripheral units, areas and constants
<0b>	a b-marked zero terminates the catalog

An item has the format:

1) An area word, a-marked
2) A secondary word, a-marked. This word may be omitted.
3) A name of arbitrary length, the last word b-marked, the other words 0-marked.
4) A specification word, 0-marked. May be omitted.
5) ..An arbitrary number of names like 3) and specification words like 4) may follow at this place.

Null items, consisting entirely of zeroes, may be present in the catalog as results of clear actions.

1.1 Format of an area word:

qq <kind>.2 + <special>.3 + <program>.4 + <reserved>.5 + <inhibit>.6 + <sum>.7 + <various>.39,

The content of <various> depends on the value of <kind> which describes the kind of storage medium concerned.

<kind>	corresponding format of <various>
0 drum area	<blocks>.23 + <first block>.39
1 disc area	<blocks>.23 + <unit>.27 + <first block>.39

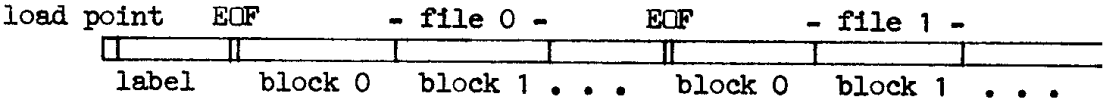
<unit> is disc file number, usually 8.

2 carroussel	<blocks>.23 + 7.27 + <grouped>.29 + <reel>.35 + <block>.39

<reel> and <block> describes the first block in the area.
<grouped> is the number of 512-word blocks to be transferred by one il or us instruction when transport from or to the area takes place.
<special> and <reserved> will always be 0.

3 tape area	<blocks>.23 + <unit>.27 + <file>.32 + <block>.39

<unit> is the tape station number, 1 ≤ unit ≤ 6. <file> and <block> determines the first block of the area thus:



All blocks, except the label, are assumed to be 400 words. The first word of the label block must contain the text ‡<<cattap>‡ in order to be accepted by init medium. <special> and <reserved> will always be 0.

4 constant	<arbitrary bits>.39

Bits 3 - 7 in the area word will be 0.

6 sy-medium	<by>.19 + <mask>.29
7 ly-medium	<by>.19 + <mask>.29

<by> and <mask> will be used to execute vy <by> t <mask>. bits 3 - 7 in the area word will be 0 when <kind> = 6 or 7.

<special>	is 1 in case of the items free and work, 0 otherwise.
<program>	is 1 in case of an executable program. If <program> is 0 the item is a text area.
<reserved>	is 1 if the area originally has been a part of the free area. Then it may again be included in free by means of a cancellation and a clean up of the backing store. If special = 1, see 1.5.
<inhibit>	is only significant if <program> is 1. <inhibit> = 1 indicates a program to be executed with inhibited core store. <inhibit> = 0 corresponds to a task program. If special = 1, see 1.5.
<sum>	is 1 if the secondary word contains the negative sum of all the words in the area (including marks.9).

1.2. Secondary word.

Usually the secondary word will contain the checksum of the area, but, in case of free, work, and date, the content is different. See 1.5.

1.3. Names.

The names in the catalog must have the form of a Slip text without any spaces, case shifts, or 'invisible' symbols. Furthermore the last word of the name must be b-marked.

1.4. Specifications.

In text areas a specification word is normally not used, but when a program area with specification is called by means of Help 3, the specification will be interpreted as follows:

```
qq <tracks read to core>.9 + <entry address>.19
   + <core address for first track>.29
   + <tracks skipped in start of area>.39
```

1.5. Free, work and date.

Format of free item:

```
qq <free kind>.2 + 1.3 + 1.6
   + <various corresponding to free kind>.39,
qq <no of catalog tracks>.7 + <booked>.23 + <min first free>.39,
tfree;
```

<booked> is the number of blocks transferred to free in the latest call of an auxiliary program with free specified as output area. If all reserved areas were cancelled the first free block would be <min first free>.

The area word of work describes the drum area produced in the latest call of an auxiliary program with work specified as output area. There are two possible forms of the work item:

Work in free:

```
qq 0.2 + 1.3 + 1.5 + <blocks written>.23 + <first written>.39,
twork;
```

Work not in free:

```
qq 0.2 + 1.3       + <blocks written>.23 + <first written>.39,
qq <max work blocks for output>.23 + <first block for output>.39,
twork;
```

The working area necessary for the Algol translator and a few other programs is called work-as-output. In some machines it may be the free area, in other machines the area described in the secondary word of work. If work is explicitly specified as output area (e.g. edit, o work <) then work-as-output will be used. The description of work-as-output is always on the first track of the catalog, cell d45.

Format of date:

```
qq 4.2 + <day, month and year as 3 ten-bit groups>.39,
qq <run number>.9,
tdate;
```

2. SEARCH.

The Search routine searches for a name in the catalog and yields the area word, the secondary word, and the specification word (if present). Several other entries on the track are useful for selecting a track, correcting a sum, etc.

The search track is present in cell - 86 ff when an auxiliary program is entered. In other cases it may be read from drum to an arbitrary place. The routine will use the cells d14 to 39d14 for work.

All the addresses in the following assumes that the track is placed in -86 ff. The exit is always performed with hr s1; R, M, ZA and PC spoiled.

Entry conditions	Exit conditions
pp <first word of name> - 1 hs c1 [The name should not be ≠<free>]	1. R > 0 catalog error 2. R < 0 name not found 3. R = 0 cell c free area word - 1c secondary free word - 2c area word of named item - 3c secondary word (if present). cell c15 points to the item word after the name. This word and its marks is also in M, marks.
pp <first word of name> - 1 pmn 2c18 DX IZA hs c2 [the name may be ≠<free>]	1. R ≠ 0 as above 2. R = 0 cell 2c, 3c, c15 and M as above.
Raddr ≠ 0 ZA := 0 hs c2	The first track of the catalog will be selected and read to d14. c4 will contain the selected track number, c5 the selected group
pmn <general trackno> hs c3	The track is selected, c4 and c5 contain track no. and group.
ZA:= 0 hs c8	The sum word of the track in d14 is corrected. (c5) and (c4) are selected.
marks:= 11 hs c16	c4 and c5 are adjusted to point to the next track. This track is further selected.

Change of catalog and work place.

The routine may work on other catalogs (placed on drum), if the general track no. in c12 is changed to the first track of the catalog. The track place d14 may be changed by correcting the address in cell c7.

3. INIT MEDIUM.

The routine Init medium requires an area word as input parameter and initialises a few cells (the area description) useful for handling the words in the area. If the area is on magnetic tape then the tape will be positioned to the start of the area.

3.1. Normal entry and exit.

Normally the routine is entered thus:

```
transfer first track of Init medium to core (every time).
arn <area word>                ; or the equivalent.
qq <place for second track>    ; this address always in s-1.
hs <cell 0 of first track>     ; or the equivalent.
```

The normal exit will take place with hrn s1; only R, M spoilt.
An error exit can occur if a magnetic tape has not the label ≠<cattap>. The exit conditions are: hh s1; R ≠ 0 and M, s and p spoilt.

3.2. Area descriptions set by Init medium.

The description will be placed in core from <core part> - 2 to <core part> + 2 and in case of buffer media further from <buffer part> + 1 to <buffer part> + 6 in the buffer store.

The normal entry corresponds to <core part> = -4, <buffer part> = 0.

Description in core depends on kind:

cell	kind=7	kind=0	kind= 1, 2, 3
<core part>-2	not changed	not changed	qq<core part>.9+1.19 +<buf>.39
- -1	not changed	qq d14-1,	qqf 0
- +0	not changed	not changed	not changed
- +1	not changed	qq 1 , xx	qq 1 , xx
- +2	vy<by>t<mask>	qq<first block>.39,	qqf<get word track>.39

The result is unpredictable if kind is 4, 5 or 6.
<buf> is <buffer part> + <transport length> + 7, where <transport length> means the number of words transferred by one il x or us x instruction.

Description in buffer in case of buffer medium

cell	kind=1	kind=2	kind=3
<buffer part>+1	area word	area word	area word
- +2	qq 1.21	qq<grouped>.9	qq 0
- +3	current block	current block	current block
- +4	qq d53.39	qq<grouped>.30	qq 400.39
- +5	qq<unit>+16,il<unit>	qq 7+16,il 7	qq<unit> +160,il<unit>
- +6	qq 1.39	qq<grouped>.39	qq 1.39

Current block is:
kind=1 qq d53.9+<first block-1>.21+<buffer part+7>.39
kind=2 qq <reel>.5+<block>.9-<grouped>.9+<grouped>.19+<buffer part+7>.39
kind=3 qq 400.19+<buffer part+7>.39
Area word is the original area word minus the word in <buffer part> + 6.

The Help 3 routines Read internal and Get word update the area description as suggested by the following description:

<core part>-2		transfer top of block to core (buffer only)
	-1	current word address (drum only) - <left in block> - 1 (buffer only)
	+0	current word
	+1	character count: -4 if first character of the word is fetched, 1 if the last character is fetched.
	+2	current track number (drum only). Fixed otherwise.
<buffer p>	+1	area word describing the block from which the latest word was fetched.
	+2	block increment. Added to current block to read next block.
	+3	current block. The instructions: arn <buffer part+3> , il <unit> were used to read the current block
	+4	transport length. Fixed
	+5	qq <check transport> , il <unit>. Fixed
	+6	area increment, used to update area word. Fixed.

3.3. Special entries.

If the address part of cell 3 of init medium is cleared before entry the area description will be stored according to

<core part> = c - 1 , <buffer part> = 1543.

In case of a tape area, the label must be equal to the content of cell c80-c28 of init medium. This cell usually contains ⟨<cattap⟩, but it may be changed before entry.

Init medium may be entered at cell 2, if the second track already is in core store.

Check of magnetic tape label can be avoided by clearing of cell c81-40c28 on the second track before entry, but the label file will still be skipped.

The area description may be stored somewhere else by placing <core part> in p, <buffer part> in c80-1c28 on first track and entering cell 4 of Init medium. The address part of cell 3 of init medium must be cleared ahead. The p-register will be spoilt upon return.

3.4. Help's current input medium.

The selection of current input medium is obtained by a normal call of init medium. If the input medium is not explicitly selected (e.g. after an alarm message) the description in core is:

cell -2(=<core part>+2) : vyn 1 t 1016

4. GET WORD.

This routine transfers one word from a buffer medium selected by Init medium.

4.1. Normal entry.

The get word track is fetched from the drum, and if a buffer area description is present in the normal place the routine may then be called several times thus:

```
hs <cell 20 on track>
```

The normal exit is performed by hrn s1 with the next word from the medium both in M and cell <core part>.

An error exit can occur if a block fails (parity error) even after 4 rereadings. The return is: hh s1 with R = status word.

In each call the area description is updated.

4.2. Special entries.

The routine may handle area descriptions placed elsewhere if the following cells are changed:

```
cell 18 on track := qq 6.19 + <buffer part+1>.39;
cell 19 address  := <core part>;
cell 21 address  := <core part> - 1;
cell 23 address  := <core part> - 2;
cell 24 address  := <core part>;
```

When an auxiliary program is called by means of Help 3, a Get word routine initialised according to <core part> = c-1 and <buffer part> = 1543 is present in the core store. Its entry is in cell -26.

APPENDIX B: HELP 3 GLOBAL NAMES.

s-marked names may be changed in the start
d-marked names may be changed during loading of aux. programs

```
  d d   work name
s+d d1  first track loaded
  d d2  load init code
  s d3  free kind
  s d4  disc blocks mod 1024
  s d5  first free block mod
        1024
    d6  sum cell help
    d7  balance track 1
    d8  intermediate first help
    d9  first track final help
   d10  [FB-294, SLIP]
  s d11 parameter track
   d12  first core help
   d13  first core param track
   d14  first core texttrack
        and catalog track
   d15  length text print
  s d16 image track 0
  s d17 reader (0 or 3)
  s d18 by inhibit
  s d19 image group
   d20  [first SLIP in core]
  s d21 first catalog track
  s d22 no of 320 track drums
  s d23 no of catalog tracks
   d24
   ...  print entries
   d31
  s d32 work in free
  s d33 first work track
        (group d19)
  s d34 work tracks
  d d35 aux kind
  d d36 aux reserved
   d37  last relative help track
        (relative to d9)
   d38  last core for program
        call
   d39  aux only
   d40  print entry
  s d41 buffermedia treated
   d42  image track 0 during
        loading
  s d43 help alarm unit,
        including typewr input
  s d44 standard output unit
        including typewr input
   d45  work as output area,
        relative on track
  s d46 first free block:1024
   d47  return to init help
  d d48 load primitive catalog
   d49  first of 80 work cells
        in init help
  s d50 image group during
        loading
   d51  date word, relative
        on track
  s d52 discblocks:1024
  s d53 block length, disc
  s d54 lib station
   d55  version number


    c   search results
    c1  search entry
    c2  get free
    c3  select track in M
    c4  track selected
    c5  group selected
    c6  search mode
    c7  read track and sum
    c8  sum track
    c9  sumit
   c10  (search)
   c11  960
   c12  catalog start
   c13  test end
   c14  cont search
   c15  last searched
   c16  select next track
        (marks 11)
   c17  (search)
   c18  (search)
   c19  (search)
   c20  (search)
   c21  (search)
c22+2   selected output
   c23  text print entry
   c24  alarmprint entry
   c25  primitive input entry
   c26  annull print entry
   c27  read internal entry
   c28  init medium entry
   c29  put state
   c30
   ...  [SLIP names]
   c39
   c40  entry hsf 2
   c41  entry hp-button
   c42  entry overflow
   c43  char , ly D
   c44  read look-up entry
   c45  entry set typewr
   c46  entry hs 2,
        select medium
   c47  interprete
   c48  program call
   c49  entry hsf 1
   c50  entry hs 1
   c51  return from aux
        program
   c52  get param entry
   c53  read and check help
   c54  core dumped
   c55  finish track 1
   c56  finish track 1
   c57  kind alarm
   c58  param alarm
   c59  part of exit
   c60  not found alarm
   c61  full alarm
   c62  = c45 - 1
   c63  init medium track
   c64  get word track
   c65  get state
   c66  modify get word
   c67   -   -   -
   c68   -   -   -
   c69  = 2c18
   c70  fault alarm
   c71  get word entry
   c72  exit from text print
   c73  get status
   c74  adjust special
   c75  modify get word
   c76  modify get word
   c77  sum alarm
   c78  sum, variable in init
        help
   c79  sum track, procedure
        in init help
   c80  label in init medium
   c81  clear label check
   c82  buffer part, init med.


    e   -1
    e1  toverflow;
    e2  tfull;
    e3  tsyntax;
    e4  tannull;
    e5  tparam;
    e6  tundef;
    e7  tcatalog;
    e8  tkind;
    e9  tlabel;
   e10  vyf 1 , qq 1016
e11+1   thsf1;
e12+1   texit;
   e13  tfault;
   e14  vy 1 t 511
   e15  std. entry constant
   e16  tsum;
```

°a, 7
alarms, general, 15, 25
alarms, special, 15, 25
area description, 52, 54
area length, 11
area word, 48, 52
backing stores, 5
bin 0 form, 16, 28, 29, 46
binary tape, 27, 28, 36, 46
block, 5
booked, 11, 31, 40, 50
bootstrap, 24, 29
buffer media, 5
carrousel, 5, 49
case shifts, 7, 33
catalog, 6, 12, 48
cattap, 46
check bin, 46
compiler, 25, 47
constants, 48, 49
core image, 5, 12, 13, 35
corrections, 5, 32
create new -> old, 46
current input medium, 6, 10, 53
date, 13, 45, 48, 50
disc, 5, 49
domain, 28, 37, 38
drum, 5, 49
dump, 5, 13, 35, 41, 45
entries to Help 3, 13
exit address, 13
external media, 5
file, 49
free, 31, 32, 36, 40, 41, 50
free kind, 48
generating Help 3, 17
get word, 54
group number, 5
half inhibition, 13, 32, 41
head bin 0, 16
head kompud, 46
help 3 information, 6, 7
help number, 7
hp-button, 13
hs 1 (switch), 14, 21
hs 1, hsf 2, hsf 1, hs 2, 14
image, 48
include algol, 47
inhibit bit, 21, 40, 49
inhibition, 10, 13, 25, 45
init help, 16, 17
init medium, 52
internal entries, 22
internal form of parameters, 7
internal media, 5
item, 48, 51
l-item, 48
label, 46, 49, 52
ly-medium, 49
main help, 12, 16, 17, 19
move system, 17
name, 50
no-alarm, 30, 31, 40, 42
normal out, 25
null item, 31, 32, 48
overflow, 14
p-item, 48
P 1, 47
parameter list, 6
parameter track, 6, 12, 21, 26
primitive catalog, 17
primitive input, 7, 13, 24
program bit, 40, 49
programmed call, 14
programmed entries, 14
r-item, 48
redefine, 18
registers, 13, 35, 39
reserved areas, 32, 49
return to help, 21
search, 51
secondary word, 40, 48
selected output, 6
single, 7
source program, 26
special bit, 49
specification word, 21, 40, 48, 50
sum bit, 21, 30, 40, 42, 49
sum check, 10, 21, 30
sum code, 10
swop tapes, 20
sy-medium, 49
system punch, 20
system track, 23
t-item, 48
tape, 5, 11, 49
tape label, 46, 49
task program, 12, 21, 49
text format, 10
track, 5
track-38: 12, 13
translated progr, 26, 41
transient compiler, 25, 26, 47
version, 18
w-item, 48
work, 36, 41, 44, 50
work-as-output, 26, 50
x-item, 30, 31, 32, 40, 48