GIER SYSTEM LIBRARY

Order No.: 490

Annotated Help 3 Programs

vol. I

main help	15 pages
init help	8 -
P1	8 -
algol	7 -
binin	3 -
binout, outparam	6 -
check, compress, setsum, list	15 -
clear, res, set	8 -

December 1967

REGNECENTRALEN

; slip<
0 ; STOP, CLEAR

[8.8.67 (2) contents of main help]

[Page:	track:	Track numbers in bracket refer to version with buffermedia
2-3	0-1	All external entries use track 0 and 1 which dumps registers and core if necessary. The actual entry is determined before main help is read from drum.
4	2	Standard content of core 0-9, part of the code for exit and an address print routine.
5-6	3(3-4)	Init medium which transforms an area word to a more convenient discription of an input medium in cell -6 to -2. It also used for initialisation of program call and then the description is placed in cell c-3 to 1c. If the area word describes a point on magnetic tape then the tape is positioned.
7	about 4(5)	Entry from track 1 and special treatment of all entries to help.
8	about 5(6)	The help input program which reads help information and stores it in the parameter buffer.
9	about 6(7)	The parameter interpreter which scans the parameter list and accordingly selects media and starts program call by means of init medium.
10	about 7(8)	Tables and character look-up for help input.
11	about 10(9)	Contains a few constants, a routine for reading next character from internal media and a routine for adjusting booked or work after certain aux. programs.
12	about 8(10)	Program call. Reads the program to core and checks the sum if wanted. core 0-9 is always initialised and core inhibition is removed if wanted.
13	9(11)	Search catalog. Contains several small routines for selecting and reading tracks. The main purpose is to search for a text in the help catalog.
14	10(12)	Contains textprint and several constants. If buffer media are treated then a routine is included for getting next word from buffer medium. Otherwise is the read internal routine loaded to this track.

]

```
b i=0,d55; outermost block
d55=2     ; version number
 d1=100   ; first track loaded
 d3=0     ; free kind
 d4=0     ; discblocks mod 1024
 d5=39    ; first free block
 d9=0     ; first track of final help
d11=38    ; parameter track
d16=294   ; image track 0
d17=3     ; reader
d18=512   ; by inhibit
d19=960   ; image group
d21=34    ; first catalog track
d22=1     ; no of 320 track drums
d23=4     ; no of catalog tracks
d32=1     ; work in free
d33=0     ; first work track if not in free
d34=0     ; no. of work tracks if not in free
d35=0     ; running kind of aux programs (may be changed later)
d36=0     ; aux programs in free         (may be changed later)
d41=1     ; buffermedia treated
d43=17    ; help alarm unit, includes typewr input
d44=17    ; help standard output unit, used by date print and most
          ; aux. programs (except binout and other with perforator output)
d46=0     ; first free block : 1024
d50=960   ; image group during loading
d52=0     ; discblocks : 1024
d53=400   ; block length, disc
d54=1     ; lib station
 d=0      ; debug:= false
ac        ; STOP, SUM
[d32, d36, d41 and d are booleans with true = 1, false = 0]
i redefine
s         ; allows redefinition of preceding names
0         ; STOP, CLEAR
< d       ; if debug then place help relative to d1 and image on 268.
d5=40d1, d9=d1, d11=38d1, d16=268, d21=36d1 >

 d8=d1    ; define intermediate help base
d13=80    ; core base of parameter track
d14=40    ; core base of text buffer
d42=k     ; define image during loading.

d37=10, <d41, d37=12>     ; last rel help track:= if buffermedia then
d12=d37+d37+d37+d37+d37   ; 12 else 10
d12=d12+d12+d12+d12       ; core base of help:=
d12=-d12-d12+34           ; - (last rel help track - 1) × 40 - 6;

d10=d16-294, d20=1        ; SLIP names: FB0, first slip in core.

          ; Prepare loading of aux. programs only. Changed by INIT HELP.
 d2=10    ; coreaddress for loading descriptions of aux programs
d39=1     ; aux mode:= aux only;

d45=4, <d32, d45=0>       ; work as output:= if work in free then
                          ; free area else max work;
d51=6, <d32, d51=5>       ; date word relative:= entry after work
```

```
b k=2d8, i=d12, c85, e20   ; begin help
b k=d8, i=0, a20, b10      ; begin dump program
                             [stored registers]     [actions]
i=1 [qq        , hh c25]   ; 0 exit address       ; jump to primitive input
    it  960    , pt   1    ; group, entry kind    ; ENTRY MODE 5:
< d22-3                    ;                      ; entrykind:= right or left;
    gg  1      , gk   2    ; 2 by , tk
×   qq         , gk   2    ; 2                    ; store group, by, tk
>   vy d18+d43, hv   a     ; 3 indicator          ; by inhibition;

b:  qq 64.9+29.15+60.21+18.27+20.33+36.39 ; tSUM ; inhibit pattern
    qq 64.9+57.15+36.21+49.27+55.33+53.39 ; t image
                                                 ; PROGRAMMED ENTRIES:
a:  gi  3              IPC ; 6 M, marks          ; store indicator
    gm  6              MPC ; 7 S                 ; store M
    gs  7              IOB ; 8 R, overflow       ; store s
    gr  8              MOB ; 9 Radr              ; store R
    ga  9     , tl -39     ; 10 s during prim inp ; store Radr for overflow
    pm  s              IRB ; 11 R sign           ; RB:= f-marked call;
    gr  11    , hr  r1     ; 12 p                ; store sign; store p;
    gp  12    , arn b      ; [work for lyn]      ; simulate return
    sr  -1    , tk  21     ; if core [-1] ≠ inhibition then goto dump core;
    hv  a1              NZ ; if core [-1] ≠ full inhibition then
    hv  a2              NA ; goto get help;
a3: arn 1b    , pm  -1     ; 17 write: save core [-1];
a5: ga  a4    , tk  10     ;   writetext(textword);
a4: sy  -1    , ck  -4     ;
    nc  0     , hv  a5     ;
    lyn -1    , gm  -1     ;   typechar; restore core[-1];
b1: bs  1     , hv  a6     ;   if SUM then begin store return s;
c25h:gs 10    , vyn 16d18  ; PRIMITIVE INPUT: select reader;
    tl  -6    , ca  0      ;   read and pack until 64 hole is read;
    ly  r4    , hs  r-1    ;   store instruction;
    gm  s3    t-1    M     ;   see binout for further description;
                           ;   end;
a6: ca  0     , hv  a2     ;   if char = 0 then goto get help;
    nc  17    , hv  a3     ;   if char ≠ < then goto write;
a1: vk  d19   , it  1      ; dump core:
    vk  d16   , it  40     ;   for i:= 40 step 40 until 960 do
    sk  0     , it  -1     ;   dump core[i] to core [i+39];
c54: ncn 24   , hh  a1     ;
    [core dumped]

a2: vkf 960   , vk  1d9    ; get help: [f to stop checksum]
    lk  40    , vkn 38     ;   read track 1 of help;
a8: ar  37    , pa  b1     ;   while -, b-marked do sum words;
    ar  2     D       LA   ;   SUM:= true;
[37] hv (a8)  Dt  1   NB   ;   if sum ≠ 0 then begin textword:= SUM;
    hv (a3)   Dt  -1  NZ   ;   goto write end;
    arn 1                  ;   exit:= left;

d=40, d:                   ; track 0
i=0                        ;
    qq        , hh  c25    ;   cell 0 on track, (address chain)
```

```
i=40
     lk  40d13 , ck  10     ;   read track 38; if entrykind = right hp then
     ca  0     , pi  12     ;   exit:= right;
     vk  d19   , nc  1      ;   if entrykind ≠ programmed call then
     ps (0)    , hvn a9     ;   begin addr:= r1; goto set exit end;

     arn 41d13 , ga  1      ;   stored group:= cell 1 on track 38;
     ck  10    , pm  42d13  ;   stored by-tk:= cell 2 on track 38;
     gm  2     , pm  a10    ;   if entrykind = hs 2 then
     ca  1     , cln 20     ;   begin M:= hs2 entries; goto hs entry end;
     ca  0     , hh  a11    ;   if entry kind = hs 1 then
     ca  -1    , pi  12     ;   begin M:= hs 1 entries; goto hs entry end;
     ps (9)    , arn a12    ;   if entrykind = overflow h then exit:= right;
a11h:hv  a9    , pp (7)     ;   addr:= stored Radr; goto set exit;
     bs  p512  Xt  551      ; hs entry: if stored s < 40 then
d7:  qq[balance], hv a13    ;   begin RB:= f - marked call;
     pm  p40d13       IRB   ;         simulate return
     ps  p40d13, hr  r1     ;   end;
a13: gs  7     , ps  p1     ;   stored s:= return s; addr:= call s+1;
     ck  -10          NRB   ;   if f-marked call then shift entry word;

a9:  bs  s506  t   515      ; set exit: if addr < 6 ∨ addr > 9 then
     gs  49d13        MQC   ;   image[9] := addr, exit;
a12:
c55:[gt  r     , ps  c42]   ; [loaded by main help]
i=i+1 [entry overflow]      ;   help entry:= Radr;
     ncn(c54)  , hv  a16    ;   if core dumped then
     vk  d16   , sk  40d13  ;   begin dump track 38;
     vk  25d16 , sk  1000   ;         dump core 1000-1023 and registers
a16: vk  960   , pm  b      ;   end;
     gm  -1           M     ;   inhibit core;
c53: vy  d43   , pp  1      ; READ AND CHECK HELP: select(alarm unit);
a14: pp  p1    , vk  pd9    ;   release by-inhibit;
     lk  d12-40t   40       ;   read help to core;
     bs  p-512d37 , hv a14  ;
a15: vkn d21   , ar pd12-d37;   check sum of help tracks;
     ar  1     D       LB   ;
     ar  2     D       LA   ;
     bsp  519d12-d37        ;
     pp  p1    , hh  a15    ;   if sum ≠ 0 then begin textword:= SUM;
[75] ck [balance], lk d13   ;
     hv (a3)   D t-1   NZ   ;     goto write end;
a10=i+1                     ;   read first catalog track;
c56:[vk  d11   , hv  sc41   ;   select param track; switch to help entry;
a10: hs 1/hsf1/hs2/hsf2     ; two words loaded by main help]
i=i+2
     qqf [checksum]         ;
d=80, d:                    ; track 1

e                           ; end dump program
```

```
b i=d12, a50, b10          ;   begin local help names. Load to track 2 of help
c=-86                      ;   predefine place of search track

[cell 0-9 in core after exit]
     it  -1    , it  -1    ; 0 OVERFLOW v: entry kind:= -2;
     it  0     , pt  1     ; 1 OVERFLOW h: entry kind:= -1;
< d22-3                    ;   HS 1: HSF 1: entry kind:= 0;
     gg  1     , gk  2     ; 2 HS 2: HSF 2: entry kind:= 1;
     vy d18+d43, vk  960   ; 3 save group, track and by;
x    pa  1     t   960     ; 2
     gk  2     , vy d18+d43; 3 select alarm unit and inhibit by;
>    vk  38    , sk  0     ; 4 save core 0-39 on track 38;
     vk  d9    , lk  0     ; 5 read track 0 of help; wait for track;
     vk [group], vk [track]; 6 EXIT TO CORE: select group and track;
     qq        N [hh-hv]   ; 7 [used by HP-patches]
     vy [core init] t -1d18; 8 release by inhibit;
     qq  10    , hv (r)    ; 9 final exit

     [Part of code for exit]
c59: pi  2     t  -3   NO  ; set 0: NRA:= R overflow;
     ac  512   DV 512 LRC ;   if -, stored overflow = R overflow then
     ac  512   Dt 512 NRC ;   0:= -, 0;
     pi (3)    , lk  0    ;   restore indicator;
     vk  d19   , it (86d13);   read exit to core 0;
     pa  6     , hh  0    ;   prepare restore group; goto read image;

b a8 [address print, PM]
d40: pa  ra1   t  511      ; clear count: count:= 0;
d24: tk  20                ; spacing and print 30-39:
d25: tk  10                ; spacing and print 10-19:
d26: bt (ra1)  t   1       ; spacing and print  0- 9:
     sy        , hv  r-1   ;
d27: pa  ra1   t  506      ; COUNT 5: count:= 5;
d28: hs  ra1               ; SIMPLE PRINT:
     srn(ra4)  Dt  -16     ;
     hv  ra                ;
a1:                        ;
d29: qq[anslag], ck        ; print with zero as empty:
d30: bs 570 [case], it 510 ;
     sy (r-1)  , tl  -30   ;
     pa  ra7               ;
     dk  ra2   XV          ;
a6:  sy        , it  16    ;
a3:  pa  ra4   , it  -128  ;
a7:                        ;
d31: bt 1 [slip boolean], hrn s1    ;
     mln ra5   , tk  30    ;
a4:  ar  0     D       LZ  ;
a:   hh  ra3           LZ  ;
     qq (rd29) t   1       ;
     ga  ra6   , hv  ra6   ;
b:a5:m 10                  ; [base of constants]
a2:  9999                  ;
e

d=40d12, d:                ; track, address print
```

```
b a10, b21
c28: c63=k-d8+d9              ;   define init medium track.
b20: vk  960    , vk  1c63    ; INIT MEDIUM: get second track of init medium;
     [blocks in area]
     lk (s-1)   , vk  960     ; ENTRY NO GET TRACK: R contains an area word;
[2b20]gp ra     , pp  c-1     ;   save p;
[3b20]bs 1      , pp  -4      ;   p:= if medium then -4 else search work;
     [medium]
     gr  rb3    , tl  -7      ;   area word:= R;
     hv  ra1              NT  ;   if kind > 0 then goto internal;
     nc  -1     , pp  c22     ;   if kind ≠ ly then p:= selected output - 2;
     tk  17     , ga  rb10    ;   work:= core[p+2]:=
     gt  rb10   , pm  rb10    ;   vy <bits 10-19> t <bits 20-29>;
b10: vy  -1     t    -1       ; work for ly and sy media, and for label check
     gm  p2               M   ;   select medium;
a:   pp  -1     , hrn s1      ;   restore p; return normal;

a1:  ga  rb1    , tl  -25     ; INTERNAL: store kind;
     tln 16     , gr  rb20    ;   blocks:= bits 8-23;
     arn rb4    , gr  p1      ;   char count:= 1;
b1:   [kind]
<d41, ncn -1    , hv  ra2     ;   if kind ≠ drum ∧ buffer media then goto buffer;
×     ncn -1    , hv  c57     ;   if kind ≠ drum then goto kind alarm;
>    tl  -23    , ud  ra8     ;   medium word:= bits 24 - 39,;
     pm  d14-1 D              ;
a8:  gm  p5     t   -3  MA    ;   word address:= qq <text base - 1>,;
b4:  qq  1      , hv  ra      ;   restore p; return normal;

[buffer medium state]
c82:
b3:  qq                       ; area word. END OF INIT WITHOUT BUFFER.
<d41,qq      1.21             ; block increment
[2b3]qq d53.9   +7.39         ; current block
[3b3]qq       d53.39          ; block length
[4b3]qq  16     , il  -1      ; check block, read block
[5b3]qq           1.39        ; area word increment
b2:  qq     1.19+7.39         ; transfer upper
c29:
b21: qq [b3.9]6.19+1.39       ; put state
b13: qq       1.19            ; get label
b19: qq         1543.39       ; base for program buffer
c80:
b15: tcattap;                 ; tape label
     qq 6.19+1544.39          ; get progr state

a2:  tln 4      , ck  -10     ; BUFFER:
     ga  rb5    , ac  r4b3    ;   unit:= bits 24-27; check block:= 16 + unit;
     arn c64    D             ;   save s;
     gs  ra3    , ck  10      ;   medium word:= get word track b-marks;
a4:  gr  p5     t   -3  MB    ; [s-2]  read block:= unit;
b5:  it [unit]  , pt  r4b3    ;
b6:  hs (s-1)   , udn ra4     ;   call second track; word address:= 0b;
a3:  ps  -1     , hv  ra      ;   restore s and p; return normal;

d=40b20, d:                   ;   init medium 1
```

```
b1=b1-b6, b2=b2-b6, b3=b3-b6, b5=b5-b6, b10=b10-b6 ; define s-rel
b13=b13-b6, b15=b15-b6, b19=b19-b6, b21=b21-b6, b20=b20-b6;

[rel 0] gp ra9 , pp (sb1)  ; SECOND TRACK: save address of medium table;
     can p-2   , hh  ra5   ;   p:= kind; if kind = carrusel then goto carr;
     ca  p-3   , hv  ra6   ;   if kind = tape then goto tape;

     tln 12    , tk  18    ; DISC: current block := current block + first block.21;
a5h: hv  ra7   , tln 2     ;   goto finish;

     gr  s5b3  , tk  9     ; CARR: area word increment:= grouped;
     gr  s3b3  , tk  11    ;   blocklength:= 512 × grouped;
     gt  s2b3  , tk  10    ;   current block 10-19 := grouped;
     gr  s1b3  , tl  40    ;   block increment:= grouped.9;
     ga  s2b3  , hvn ra7   ;   current block 0-9:= reel and block; goto finish;

b7:  qq  400.19+7.39       ;
a6:  arn rb7   , gr  s2b3  ; TAPE:
     tk  -20   , gr  s3b3  ;   current block:= qq 400.19+7.39; block length:= 400;
     qq (s4b3) t   144     ;   check block:= unit + 160;
     pp (sb5)  , usn p64   ;   p:= unit; rewind(unit);
     arn sb13  , il  p     ;   read block (unit) length: (1) to: (0);
     ar  sb10  D           ;   read eof (unit);
     il  p64   , il  0     ;   get label to work;
     arn sb15  , sr  sb10  ;   if label ≠ <<cattap|then
c81: hh (s1)   t   1   NZ  ;   return error;
     tln 5     , ck  -10   ;   files:= bits 28-32;
     ga  rb17  , tln 37    ;   blocks:= bits 33-39;
     ga  rb18  , grn s1b3  ;   block increment:= 0;
b17: bt  -1    t   -1      ;   for i:= 1 step 1 until files do
     il  p64   , hv  r-1   ;   read eof (unit);
b18: bt  -1    t   -1      ;   for i:= 1 step 1 until blocks do
     iln p     , hv  r-1   ;   read block (unit) length: (0) to: (0);

a7:                        ; FINISH:
     sr  s1b3  , ac  s2b3  ;   current block := current block - block increment;
     arn s5b3  , sc  sb3   ;   area word := area word - area word increm;
     can(s3b20), arn sb19  ;   if -, medium then begin
     ac  s2b3  , ac  sb21  ;     current block:= current block + program buffer;
a9:  pp  -1    , gp  sb2   ;     put state:= put state + program buffer;
     ar  sb2   , ar  s3b3  ;     transfer upper:= transfer upper + program buffer
     gr  p-2   , it  sb3   ;   end; p:= address of medium table;
     pa  sb21  , arn sb21  ;   core[p-2]:= p.9 + transfer upper + block length;
     us  0     , hh  s     ;   put state to buffer; return to first track;

× c29=i, c80=i, c81=i>     ;   define c29, c80, c81 in no buffer case
e
```

```
c51: lk  d13   , pp  d13   ; RETURN FROM AUX PROGRAM:
     vk  d11   , arn d13   ;   read parameter track;
     ca  0     , hv  c46   ;   if -, hs 1 then goto select medium;
     ca  1     , pp  e12   ;   if hs 1 ∧ no interprete then param:= exit;
a41: vy  d44   , hv  c47   ; prep interprete: select (std output); goto interprete;

c49: pp  e11   , hv  a41   ; ENTRY HSF 1: param:= hsf 1; goto prep interprete;

c50: pa  d13   t   1       ; ENTRY HS 1: hs 1:= true;
     arn 49d13 , ga  r1    ;
     pp  -1    , vk  d19   ;   p:= exit addr; select image group;
     ps  d16   , pp  p-40  ;
     bs  p552  t   551     ;   p:= track rel (exit addr);
     ps  s1    , hh  r-2   ;   s:= track no(exit addr);
     pp  p80d13, vk  s     ;   param:= address(first param) - 1;
     lk  40d13 , ncns-25d16;   read track(s); if s ≠ image track 25
     vk  s1    , lk  80d13 ;   then read track (s + 1);
     vk  960   , hv  a41   ;   goto prep interprete;

c40: it  39[p] , pa  b3    ; ENTRY HSF 2:
c41: sy  64    , sy  29    ; ENTRY HP-BUTTON: writecr; writered;
     sy  58    , arn d13+1d51; count 5 and print (run);
b3:  sy  0     , hs  d27   ;   writechar (if hsf 2 then p else space);
     vy  d44   , arn d13+d51 ;   select (std output);
     ck  10    , hs  d26   ;   spacing and print (day);
     arn d13+d51, ck 20    ;   count 5 and print (month);
     sy  59    , hs  d27   ;
     arn d13+d51, ck -10   ;   simple print (year);
     sy  59    , hs  d28   ;
     arn 1     D           ;   run:= run + 1;
     ac d13+1d51, sc 39d13 ;   compensate catalog check sum;
     vk  d21   , sk  d13   ;   write catalog track;
     vk  d21   V           ;   skip line
c42: qq  e1    , hs  c23   ; ENTRY OVERFLOW: writetext (≪overflow≫);
     sy  0     , sy  0     ;
     can(c54)  , sy  53    ;   if core dumped then writechar(e);
     sy  56              LQB ;   if exit = right then writechar(h);
     arn 49d13 , hs  d28   ;   simple print(exit addr);
c62=i-1                    ;
c45: pm  e10   , vy  d43   ; SET TYPEWR: select (alarm unit);
     gm  -2              M ;   current medium:= vy 1 t 1016;
                           ; SELECT MEDIUM: ENTRY HS 2:
c46: grn d13   , arn -2    ;   hs 1:= false ; write param track;
     nc  1     , hh  a23   ;   if current medium = type wr then
     sy  62    , sy  58    ;   begin write black; writelc; writecr end;
a23h:sy  64    , vy  d44   ;   select(std output);
     hv  a40             NC ;   if current medium -< internal then
     pmn -2    , hs  c3    ;   begin select medium track;
     lk  d14   , vk  960   ;     read medium track;
     pa  b1    Vt c44  M   ;     read and look-up:= internal
a40: ud  -2                ;   end else
     pa  b1    t  c43  NC  ;   begin select(current medium);
                           ;     read and look-up:= lyn D end;
```

```
b k=d8, i=0                    ;   finish track 1 of help
c40=c40-c41, c42=c42-c41, c46=c46-c41 ; define relative
c49=c49-c41, c50=c50-c41, c51=c51-c41 ; entries
c45=c45-c41                    ;
i=c56                          ;
     vk  d11    , hv  sc41     ; define aux return
     qq c50.9+c49.19+c46.29+c40.39 ; entry table
i=c55                          ;
     gt  r      , ps  c42      ; define overflow entry
c40=c40+c41, c42=c42+c41, c46=c46+c41 ;
c49=c49+c41, c50=c50+c41, c51=c51+c41 ; redefine
c45=c45+c41                    ;
e                              ;

b6:                            ; HELP INPUT:
a13: pp  d13    , pi  512      ;   param:= begin of line; state:= start;
a15: bs  p-38d13, hv a16       ; CHECK: if param > 38 then goto full;
a:   hv  a10            LTA    ; NEXT: if end input then goto end input;
b1:  hs [c43 or c44]           ;   read and look-up;
[s+1]pmm -1    Xt   b4         ;
     tl  20             LOA    ;   Radr:= syntax[kind] shift state;
     tl  10             LOB    ;
     ga  r1     , mb  a11      ;   state:= bits 0-1; end input:= bits 2-3;
     pi  -1     , ga  r1       ;   action:= bits 4-9;
     hvn -1     t   a          ;   switch to action;

a1:  pp  p1     , grn 2c       ; BEGIN NUMBER: param:= param + 1;
     gr  p              MB     ;   work:= 0; list[param]:= 0, number;
a2:  arn c43    , pm  2c       ; NUMBER:
     ca  16     , hhn  r1      ;   Radr:= if char = 16 then 0 else char;
     tk  -30    , ml  b        ;   if number too big then goto full;
     hv  a16           NZ      ;   work:= work × 10 + Radr;
     gm  2c     , hv  a        ;   goto next;

a3:  arn p      , nc  0        ; SHIFT NUMBER: if part 1 [param] ≠ 0 then
c61:
a16: qq  e2     , hs  c24      ; FULL: alarmprint(≮<full≯);
     ar  2c     , ck  10       ;   list[param]:= (list[param] + work)
     gr  p      V              ;   shift 10; work:= 0; goto check;
a5:  arn 2c     , ac  p        ; END NUMBER:
     grn 2c     , hv  a15      ;   list[param]:= list[param] + work;

a4:  arn(c43)   D              ; SINGLE:
     gr  p1             MA     ;   list[param + 1]:= char; single;
     pp  p1     , hv  a15      ;   param:= param + 1; goto check;

a12: qq  e4     , hs  c26      ; ANNUL: writetext (≮<annul≯);
                               ;   goto set typewr;
a14: arn 15.3   D              ; filled: list [param]:= list[param] + end word;
a6:  ac  p      , pp  p1       ; BEGIN TEXT: param:= param + 1;
     grn p      V       M      ;   list [param]:= 0, text;
a7:  arn p      , ck  -6       ; PACK:
     ar (c43)   D              ;   if list [param] -< full then
     nc (c43)   , hv  a14      ;   goto filled;
a17: gr  p      , hv  a15      ;   list [param]:= (list[param] shift -6)
                               ;   + char; goto check;
a8:  arn p      , ck  -6       ; END TEXT;
     ar  10     D              ;   while list [param] -≮ full do
     ca  10     , hh  a8       ;   list [param] := (list[param] shift - 6)
     ck  6      , hv  a17      ;   + 10; goto check;
a9:  qq  e3     , hs  c24      ; SYNTAX ERROR: alarmprint(≮<syntax≯);
```

```
a10: grn p1            MC  ; END INPUT: list[param + 1]:= end list;
     hv  a18           NTB ;   if end input = 3 then
a11: qq  63    , vk 960   ;   begin
     [mask]               ;     read track 0;
     vk  d9    , lk 0     ;     goto primitive input;
     vk  960   , hh c25   ;   end;

a18: pp  d13   , gp a13   ;   param:= begin of line:= 0;
c47:                      ;
c47h:pp  p-1   , hs c52   ; INTERPRET: get param; this will select group 0;
c58: qq  e5    , hs c24   ;   if number V single then alarm print (<<param>);
     hv  c46   , ga r1    ;   if end list then goto select medium;
     pi [bits] , vk c63   ;   indicator:= area bits;
     lk  c28   , vk 960   ;   read init medium 1;
     pa  3c28       LPA ;   if program then medium:= false;
     tl  -7    , ca -4    ;   if kind = constant V kind = -3 then
c57: qq  e8    , hs c24   ; KIND ALARM: alarm print(<<kind>);
     ca  -3    , hv c57   ;
     qq  40c28 [init 2],t17;
     hs  2c28  , qq d13   ;   call init medium; s:= parameter|base;
     hs  c24        NZ  ;   if init error then alarmprint(<<label>);
     qq  e9    , pm1 c80  ;   if -, program then goto interpret;
     hh  c47        NPA ;   if buffer media then begin
<d41,pa  c66   t   c-2    ;   modify get word to take description
     pa  c67   t   c-3    ;   from c-3 to 1c.
     pa  c68   t   c-1    ;   get state:= get progr state;
     pa  c73   t   c-1    ;   get status to c-1.
a24h:gm  c65   , ps s1    ;   end;
×
a24h:ps  s1               ;
>    pm  p2         IRC ;   move remaining parameters to
     gm  s          MRC ;   parametertrack;
     bs  s-39d13, hv a16 ;   if more than 39 params then
     vk  d11   V    LRC ;   alarmprint(<<full>);
     pp  p1    , hh a24  ;
     sk  d13   , pm 2c   ;   M:= programarea;
     arn 1c         IRC ;   RC:= marks of medium word;
     srn(c15)  , hv c48  ;   R:= -spec word; goto program call;

c52: pp  p1    , arn p1   ; GET PARAM: param:= param + 1;
     hv  s2         LC  ;   switch to end list,
     hv  s1         LB  ;             number,
     hh  s1         LA  ;             single;
     pa  a19   t   e6       ;   if name then
     pmm c69   DX       IZA ;   begin addr free word:= addr area word;
     hs  c2                 ;     search catalog (param);
c60: pa  a19   t   e7  NT   ; NOT FOUND:
     hs  c24        NZ  ;     if not found then alarm print(
a19: qq  -1    , arn p1   ;     if undef then <<undef> else <<catalog>);
     tl  -6    , ca -1    ;     param:= end of current name;
     pp  p1    , hh r-2   ;     R:= area word; switch to name;
     arn 2c    , hh s2    ;   end;
```

```
c44: hs  c27                ; READ AND LOOK UP: if internal then read internal
c43: lyn       D            ;   else begin
     [char]                 ;     char:= lyn; if tapefeed then
     ca  63    , hv  c43    ;     goto read and look up end;
     ga  a36   , mb  a37    ;
a37: sc  r3    , ga  r1     ;   shift:= char ∧ 7 m;
     [mask]                 ;
     pm  -1    Xt   b5      ;   RM:= inputtable [char ∧ 103];
     pm (r-1)  t    4       ;
[r3]
a36: tl  -1    , cl  -6     ;   Radr:= RM shift shift;
     ca  1     , hv  a12    ;   if annul then goto annul;
     ca  10    , hv  a9     ;   if end medium then goto syntax;
     ca  4     , gp  b6     ;   if CR then begin of line:= param;
     ca  11    , pan r1     ;   if LC then skip:= false;
     bs [skip] , hv (s)     ;   if skip then repeat call;
     ca  12    , gpn r-1    ;   if UC then skip:= true;
     ca  0     , hv (s)     ;   if blind ∨ UC ∨ LC then repeat call;
     ga  s1    , hv  s1     ;   kind:= Radr; return;

b5:  [inputtable]
qq  2.3+3.7+3.11+0.15+3.19+6.23+6.27+0.31+6.35+6.39 ;
;     ,   4   8   x   0   u   y   x   -   m   ;
qq  3.3+3.7+3.11+1.15+9.19+6.23+6.27+0.31+6.35+6.39 ;
;   1   5   9   aa  <   ∨   z   x   j   n   ;
qq  3.3+3.7+10.11+8.15+6.19+6.23+0.27+0.31+6.35+6.39;
;   2   6   end     s   w   x   x   k   0   ;
qq  3.3+3.7+0.11+0.11+6.19+6.23+5.27+0.31+6.35+6.39 ;
;   3   7   x   x   t   x   ,   x   l   p   ;

qq  6.4+0.8+6.12+6.16+6.20+12.24+4.28           ;
;   q   x   æ   d   h   UC  CR                  ;
qq  6.4+0.8+6.12+6.16+6.20+0.24                 ;
;   r   x   a   e   i   x                       ;
qq  0.4+0.8+6.12+6.16+11.20+0.24                ;
;   x   x   b   f   LC  x                       ;
qq  6.4+0.8+6.12+6.16+7.20+4.24                 ;
;   ø   x   c   g   .   63                      ;

b4=i-2, a1=a1-a, a2=a2-a, a3=a3-a, a4=a4-a, a5=a5-a ; define relative
        a6=a6-a, a7=a7-a, a8=a8-a, a9=a9-a          ; actions
[syntax tabel: each entry consists of state.1+end.3+relative action.9]

[state: text  number       start        underlined      input:]
qq 0.1+0.9    +1.11+0.19   +2.21+0.29   +2.31+a9.39  ; 2 space
qq 0.1+a7.9   +1.11+a2.19  +1.21+a1.29   +2.31+a4.39  ; 3 digits
qq 2.1+a8.9   +2.11+a5.19  +2.21+0.29    +2.31+a9.39  ; 4 CR
qq 2.1+a8.9   +2.11+a5.19  +2.21+0.29    +2.31+a4.39  ; 5 ,
qq 0.1+a7.9   +2.11+a9.19  +0.21+a6.29   +2.31+a4.39  ; 6 letters
qq 0.1+a7.9   +1.11+a3.19  +2.21+a9.29   +2.31+a4.39  ; 7 .
qq 2.1+a9.9   +2.11+a9.19  +3.21+0.29    +3.31+ 0.39  ; 8
qq 2.3+a8.9   +2.13+a5.19  +2.23+0.29    +3.33+ 0.39  ; 9 ≥

        a1=a1+a, a2=a2+a, a3=a3+a, a4=a4+a, a5=a5+a  ; redefine
        a6=a6+a, a7=a7+a, a8=a8+a, a9=a9+a           ;
```

```
<-d41+1, d=i, i=-26> ; if no buffermedia then load to last help track;

b a4                 ;

e:   m -1                        ;
a:   pa  -3    t   -4            ; next word: char count:= -4;
c75: nc  39d14 , hv  ra1         ;   if word address = end of drum buffer then
     srn re    , ac  -2          ;   begin medium word:= medium word + 1;
     hs  c16             M     ;     select next track; drum:= true;
     is (c75)  , lk  s-39        ;     read track to drum buffer;
     vk  960   , it  -39         ;     word address:= base drum buffer end;
a1:  pm (-5)   Vt   1   LA       ;   if drum then begin word address:= word address + 1
c76: hs  20d14                   ;     M:= current word:= word in buffer end
     hh  ra2   , hv  c70         ;   else if -, get word then
                                 ;   alarm print(<<fault>); goto unpack;

c27: bt (-3)   t   1             ; ENTRY READ INTERNAL: char count:= char count + 1;
     arn -5    , hv  ra          ;   if char count > 1 then goto next word;
a2h: pm  -4    , cln -6          ; unpack: current word:= current word shift -6;
     ck  -4    , gm  -4          ;   Radr:= core[s+1]:= next char;
     ga  s1    , hr  s3          ;   return;

<-d41+1, i=d>                    ;   continue loading

e5:  tparam;                     ;   alarm texts
e6:  tundef;                     ;
e7:  tcatalog;                   ;
e8:  tkind;                      ;
e9:  tlabel;                     ;
e2:  tfull;                      ;
c22=i-2                          ;
     qq [selected output]        ;

                                 ; ADJUST SPECIAL: R = last block + 1, M = first block
                                 ;   in = bits of input area. Catalog track is last searc
c74: hv  -9              NTB     ;   if not special then goto initialise help;
     hh  a4              LQA     ;   if work ouptut then begin
     gm  3d14  , sr  3d14        ;     work:= first block
     pa  3d14  t 1.3+d32.5       ;     + bits and kind; R:= no of blocks;
     tk  16    , ac  3d14        ;     work:= work + no of blocks end
a4h: hv  a3    , sr  d14         ;   else begin
     tl  d3.7-16, arn 1d14       ;     M.16:= last block + 1 - first free block;
     ck  d3.7+8 , tl -d3.7+16    ;       booked:= M.16;
     ck  16    , gr  1d14        ;   end;
a3:  pi  0     , hs  c8          ;   ZA:= false; sum track;
     sk  d14   , hv  -9          ;   write track to drum; goto initialise help;

e
```

```
[M:=area; R:=-spec word     ;   RC:= marks of medium word; init medium called]
c48:  qq       V       NT   ; PROGRAM CALL: if spec word ≤ 0
      hv  a25          NC   ;     ∨ -, marks 00 then
a26: cln -26 [neg]           ;     R:= -(if drum program then
      ga  e15          NRB  ;     std entry + no of tracks.9 else
      cl  26    , srn e15  ;     std entry + 10.9);
a25: mt  a26   , ga  a30   ;   R:= -R; unpack no of tracks,
      gt  a35   , ck  20    ;   entry,
      ga  a31   , ck  10    ;   core start,
      ga  a29   , vk  960   ;   first relative track;
      vk  2d9   , lk  0     ;   initialise core 0-9;
d38: pmm 1c     , hs  c3    ;   select (first track); marks:= 00;
      [max core for program call]
a29: ncn -1     , hs  a32   ;   while first rel track > 0 do
      [first rel track]     ;   skip track (first rel track);
a30: ncn -1     , hs  a31   ;   while no of tracks > 0 do
      [no of tracks]        ;   read track(no of tracks);
      grn -1           NQA  ;   if -, inhibit then release core inhibit;
      arn 3c           LQB  ;   if sum ∧ program sum ≠ 0 then
c77: hs  c24           NZ   ;   alarm print(<<sum>);
a35h:qq  e16    , hv [entry];   entry program;

a31: pp[core start]Vt 40    ; procedure read track(count);
a32: is (a31)   , pp  s40   ;   begin core start:= core start + 40;
<d41,hv  a33          NRB   ;   entry skip track: if -, drum program then
      gp  a34   , pp  -40   ;     for i:= core start - 40 step 1 until core start-1
a27: hs  c71                ;     do begin core [i]:= get word;
c70: hs  c24           NZ   ;     if fault then
      qq  e13          IPC  ;     alarm print(<<fault|);
      gm (a34)         MPC  ;
      can p1    , hv  a34   ;   end
      pp  p1    , hv  a27   ;
× c70: >                    ;   else
a33: hs  c16           LC   ;   begin if marks 11 then select next track;
      lk  p-40  , vk  960   ;     read track to (core [core start - 40]);
      gpn a34   , pp  -1    ;   end;
a34: ar  p0     , pp  p-1   ;
      ar  1     D     LB    ;   for i:= corestart - 1 step -1 until
      ar  2     D     LA    ;     core start - 40 do
      bs  p41   , hv  a34   ;     program sum:= program sum + sum(core[i]);
      ac  3c                ;   count:= count - 1; marks:= 11;
      hv (s)    Dt  -1  M   ;   return end read track;

c24: pa  -9     t c45-c41   ; ALARM PRINT: prepare help entry to set typewr;
[1c24]it -10                ;   prepare return to init help;
c26: pa  c72    t c62       ; ANNUL PRINT: prepare return to set typewr;
      vy  d43   , sy  64    ;   select(alarm unit); writecr;
      sy  29    , hh  c23   ;   writered; goto text print;

e16: tsum;                  ;   may be used late during program call.
e3:  tsyntax;               ;   the following 3 cells may be spoiled
e4:  tannul;                ;   during interprete of program.
e10: vyn 1      t   1016    ;   constant for set typewr.

<i+86, itotal length
>                           ;   check help length
```

```
i=-86                          ; load to help track d8+d37-1
     [c-3 to 1c are used during init medium in program case]
c:   qq  d18.9+d17.9+c53.19+c45.29-c41.29+d53.39; constants for
     qq  d37.9                                   ; system punch
d6:  qq [area word]            ; check sum, work for number read
     qq[area word 1] ;                 .....
                               ;   comment found = NZA;
c1:  pmn c18   DX      IZA     ; search: Raddr:= addr free word - 1; found:= f;
c2:  pm  rc12  , ga  rc17      ; get free: M:= catalog start; iparam:= Raddr;
c3:  ga  rc6   , pa  rc8       ; select: mode:= Raddr; reltrack:= 0;
     dln rc11  , ar  rc11      ;   group:= M : 960 + 960;
     ck  -10   , ga  rc5       ;   track:= M mod 960;
     cln -10   , ga  rc4       ;                    . . .
c4:  is [track], can  s-960    ; select track: if track = 960 then
     pa  rc4   , it  1         ;   begin track:= 0; group:= group + 1 end;
c5:  vk [group], vk (rc4)      ;   wait track: vk(track);
c6:  can[mode] , hr  s1        ;   if mode = 0 then return
c7=10c1[10 used by Algol]      ; read track and sum:
c7:  lk  d14   , it  1         ;   to core (place 0); reltr:= reltr + 1;
c8:  qq [reltr], arn rc7       ; sum track: isum:= iword:= place 0;
     ga  rc9   , ga  rc15      ;
     vk (rc5)  , vkn(rc4)      ;   group vk(group); vk(track); R:= 0;
c9:  ar [isum]         IPC     ; sumit: R:= set PC (store[isum]);
     ar  2     D       LA      ;   if LA then R:= R + 2 pos 9;
c10: sr  -1 [see c19-1]D LB    ;   if LB then R:= R + 1 pos 9;
     ar (rc8)  DVX     LC      ;   if -, LC then
     hv (rc9)  Dt  1           ;     begin isum:= isum + 1; goto sumit end;
c11: qq XVDN [=960.39]         ;   R:= R + reltr pos 9; R00:= R0;
c12:qqd21[catalog start].39;   ;   if R = 0 then                 . . . .
     hv (rc15) Dt  -1  LZ      ;     begin iword:= iword - 1; goto get word end;
     sc (rc9)  , pm  rc13      ;   store[isum]:= store[isum] - R; M:= 1pos 9 + noise;
c13: hr s1[see i-1] X  LPB     ; test end: if LPB then begin swap; return end;
c14: gp  rc19  , pa  rc20      ; cont search: itext:= p; equal:= t;
c15: pmn[iword]Xt   1  IPC     ; get word: iword:=iword+1; M:= 0; R:= setPC(store[iword]
c16: hv (rc4)  Dt   1  LC      ; next track: test for last word on track;
     hr  s1    X       NZA     ;   if LC then begin track:= track + 1;
                               ;   goto select track end;
c17: gr r[iparam]Vt 1  LA      ;   if found then begin swap; return end;
c18=c-1c17, c69=2c18           ;   if LA then areaword: store[iparam:=iparam+1]:= R;
     pa  rc17  V 2c18  LT      ;   if LA ∨ NT then nottext:
     pm rc10   , hv  rc13      ;     begin M:= -1.9+noise; goto test end end;
c19: sr [iname]t   1           ;   iparam:= addr area - 1; iname:= iname + 1
     pa  rc20  t 512   NZ      ;   R:= R - store[iname]; if R ≠ 0 then equal:= f;
     hv  rc15          NPB     ;   if NPB then not last text word: goto get word;
c20: pi [equal, f=512]t 511;   ;   found:= equal; goto cont search;
c21: qq d3.2+1.5, hv rc14      ;
     [value for test cancel allowed, used by algol cancel]

e                              ; end names a50, b10
```

```
b i=-46, a10, b5            ; load to help track d8+d37

c64=d9+d37                  ; define get word track
c23: gs  ra    , an  s      ; TEXTPRINT: if called from half instr then
     is  s1          NA     ;     s:= address(s) else s:= address(s+1);
a1:  ps (s)    , arn rc23   ; repeat:
     pm  s     , ps  s1     ;   Radr:= next char;
a3:  cl  -6    , ca  10.5   ;
c72:                        ;
a:   ps  -1    , hr  s1     ;   if Radr = 10 then return;
     ca  15.5  , hh  ra1    ;   if Radr = 15 then get next textword;
     tk  -4    , ga  ra2    ;
     ca  63    , it  1      ;   if Radr = 63 then outchar(64) else
a2:  sy  -1    , hvn ra3    ;   outchar(Radr); goto repeat;
d15=i-c23                   ;   [length of text print used in slip]

e13: tfault;                ; e1 to e15 is used by get word for:
e1: b: k 29, 63             ; b: area word [rel 11 on track]
     toverflow;             ; 1b: block increment
     .                      ;
e14: vy  1     t  -1d18     ; 2b: current block
e11=i-1, e12=i              ; 3b: block length
     thsf1;                 ; 4b: check block, read block
     texit;                 ;
e15: qqf 10.9+40d13.19+40d13.29,; 5b: area word increment
     [standard entry]
<d41                        ;   if buffermedia then load get word;
     qq          1.39       ; 6b: 1
c65: qq  0.9+6.19+1.39      ; 7b: get state
c73: qq -4.9+1.19           ; 8b: get status
c71=-26                     ; ENTRY GET WORD:
c71: arn r6b   , ar (rc66)  ;   [rel 20 on track] if 1+remaining words gq 0 then
c66: gr  -5    V      LT    ;   goto get block;
     pa  rb1   , hv  ra4    ;   remaining words:= remaining words + 1;
c67: ar  -6    , il  0      ; transfer: get buf word(remaining words+trans upper);
c68: pm  -4    , hrn s1     ;   M:= current word; return normal;

a4:  it  rb    , pa  r7b    ; get block: repeat:= 0;
     arn r7b   , il  0      ;   get state to area word ... area word increm;
     arn r5b   , ac  rb     ;   area word:= area word + area word increm;
     arn r1b   , ac  r2b    ;   current block:= current block + block increment;
a5:  arn r2b   , ud  r4b    ; rep: read block(current block);
     arn r7b   , us  0      ;   put state back to buffer;
     arn r8b   , il (r4b)   ;   wait for block; read status to current word;
     il  0     , arn(rc68)  ;   if no error then
b1:  bt  0     Vt -100 LT   ;   begin remaining words:= -block length;
     srn r3b   , hv  rc66   ;     goto transfer end;
     hh  s1                 ;   repeat:= repeat + 1;
     hv  ra5                ;   if repeat gr 3 then error return; goto rep;
× c65=i, c66=i, c67=i, c68=i, c71=i, c73=i>
<i+9, i get word length
>                           ;   check length of get word
i=-9                        ;
     ps c51-c41, vk 960     ; INITIALISE HELP: s:= relative entry;
     vk  1d9   , lk  40     ;   read track 1 of help;
     vk  960   , hv  c53    ;   goto read and check help;
e                           ;   end names a10, b5
a                           ; STOP, SUM, a
i main help
s
```

0[STOP, CLEAR]

[This tape may be loaded after main help and before aux. programs. The loading will then create a new help system including the loaded aux programs. The final storing of the aux. programs is controlled by means of the names d35=aux kind and d36=aux reserved.

Main help and aux programs are loaded to track d1 and onwards. Init help, special initialisation code and a primitive catalog is loaded to the core image. When Init help is executed the following takes place:

1) Special init code is executed.
2) Main help is initialised and moved to group 960 track d9 and on.
3) The primitive catalog is scanned and the entries are treated thus:
   All items are checked for proper kind if the reserve bit is set.
   Program items are moved to final place, the area word and sum word (if present) are adjusted. Moving depends on kind.
   Programs in free are moved to first free block and on, the free area is adjusted.
   Other drum programs are moved to group 960 and displaced d9-d8 tracks.
   Other disc programs are moved to block 0 and on.
   Carroussel programs are moved to reel 0, block 0 and on. Blocks grouped 3 per transport. Only one reel can be loaded.
   Tape programs are moved to the magnetic tape on station d54. A ↓<<cattap>↓ label is put on the tape and the last program terminated with EOF.
4) Core 0-9 are initialised.
5) The catalog is finished and placed on drum.
6) hsf 2 is performed.

   If Init Help is omitted then only the aux programs will be loaded to d1 and on. Calls of set, move and setsum will be loaded to the core image.]

```
b k=d42, i=0, a30, b30, e20 ; load to image
I=10                        ;
     hvf a                  ; execute special init; security end of cat.
b1=i, d49=b1, i=80i         ; define base of track buffers

b:  m 1                             ; constants and work locations
[1b] 960                            ;
[2b]  40                            ;
[3b]  qq 15.5+15.11+15.17+15.23     ; EOF constant
[4b]  qq 3b.9+1.19+4095.39          ; EOF to buffer
[5b]  qq [buf rel]                  ;
[6b]  qq 1.19+4095.39               ; EOF or label to tape
[7b]  qq b1+40.19+41.39             ; 40 words to buffer
[8b]  qq 13b.9+1.19+4095.39         ; label to buffer
[9b]  qq [work and blocks]          ; blocks in primitive catalog
[10b] qq 9b.9+1.19+0.39             ; get status
c78:                                ;
[11b] qq [sum]                      ; sum of outputted words
[12b] qq [first block]              ; first block in primitive catalog
[13b] tcattap;                      ; tape label
[14b] qq d9.39                      ; to drum base
[15b] qq d8.39                      ; from drum base
[16b] qq 2d38.39                    ; max core for program call

e5:  vy  16    , sy  64     ; ALARM: writecr; writered;
     sy  29    , pp (s)     ;
     hs  a14   , ps  b3     ;   writetext(param); prepare return to scan cat;
a12: arn(b3)   t   1        ;
     hv  a12          NT    ;   scan catalog for first following text;
     pp (b3)   V      NC    ;   writetext(catalog)
     hv  a12               ;
a14: pmn p     X            ; writetext:
a22h:cl  34    , ck  -4     ;   Radr:= next char;
     ga  a13   , ca  15     ;   if Radr = 15 then
     pp  p1    , hv  a14    ;   write next word;
     ca  10    , hh  s      ;   if Radr = 10 then return;
a13: sy  -1    , cl  -6     ;   write char;
     hh  a22                ;   get next char;

b17: tmove trouble   ;
b18: tformat   ;
b19: tprogram call   ;
b21: tkind   ;
b23: tcat length   ;
```

```
b13:  qq d53.39              ; PARAMETERS TO FREE: block length
[1b13]qq d46.29+d5.39        ;   first free block
[2b13]qq 8.27                ;   disc unit = 8

b14:  qq d53.39              ; PARAMETERS TO DISC: block length
[1b14]qq 0.39                ;   first disc block
[2b14]qq 8.27                ;   disc unit = 8
[3b14]qq d53.9+41.39         ;   d53 words to disc
e3:
a4h: grn 5b    , arn p1      ; END DISC: buf rel:= 0;
     tk  18    , ar  3b14    ; rep: write to disc(d53) words to : (first block);
     us  24    , iln 24      ;
     arn 10b   , il  0       ;   read status to work;
     arn 9b                  ;   if error then goto rep;
     hh  a4            LT    ;
     arn b     , ac  p1      ;   first block:= first block + 1;
     hr  s1                  ;   return;

b15:  qq 512.39              ; PARAMETERS TO CARR: block length
[1b15]qq 0.39                ;   first carr block, reel 0, block 0
[2b15]qq 7.27+3.29           ;   carr unit = 7, grouped = 3
[3b15]qq 1.19+41.39          ;   3 blocks to carr
[4b15]qq 0.39                ;   base next carr program, set in carr area
[5b15]qq 512.39              ;   physical block length, used in carr area
[6b15]qq 1.39                ;   increase to first carr block

e10: arn 5b    , ar 7b       ; END CARR: output track to (buffer [41+blockrel]);
     us  0     , arn 5b      ;   bufrel:= bufrel - block length;
     sr  b15   , gr  5b      ;   the remaining track words will be outputted
a2:  arn p1    , tk  30      ;   upon return;
     nc  15    , ca  14      ; rep: if first block -< 14, 15 then
a21: qq  b17   , hs  e5      ;   alarm(<<move trouble>);
     ar  3b15  , us  7       ;   write to carr(3) blocks to: (first carr block);
     iln 23    , arn 10b     ;   read status to work;
     il  0     , arn 9b      ;
     hv  a2            LT    ;   if error then goto rep;
     arn 6b15  , ac  p1      ;   first block:= first block + 3;
     hr  s1                  ;   return;

b16:  qq 400.39              ; PARAMETERS TO TAPE: block length
[1b16]qq 0.39                ;   first tape block
[2b16]qq d54.27              ;   tape unit = d54
[3b16]qq 400.19+41.39        ;   400 words to tape
```

```
e11: arn p1                    ; END TO TAPE:
     hv  a19          NZ       ;   if first tape block = 0 then
     arn 8b    , us  0         ;   begin buffer[4095]:= tprogr;
     usn 64d54 , arn 6b        ;     rewind tape;
     us  d54   , iln 160d54    ;     write to tape (buffer[4095]);
     arn 10b   , il  0         ;
     arn 9b    , vy  17        ;     if error then
     hv  a30          NT       ;     begin
     sy  64    , sy  54        ;       writetext(<<fault>);
     sy  49    , sy  20        ;
     sy  35    , sy  19        ;       typechar; goto end to tape
     ly  9b    , hv  e11       ;     end;
a30: arn 4b    , us  0         ;     buffer[4095]:= EOF;
     arn 6b    , us  144d54    ;     write EOF on tape;
a19: grn 5b    , arn 3b16      ;   end;
a20: us  d54   V               ;   buf rel:= 0; write to tape(buffer[41:440]);
     arn 3b16  , us  32d54     ; rep: if error then skip 5 inches and write block;
     iln 160d54, arn 10b       ;   read status to work; back up if error;
     il  0     , arn 9b        ;
     hv  a20          LT       ;   if error then goto rep;
     arn 6b    , us  144d54    ;   write EOF on tape;
     iln 32d54 , arn b         ;   back up;
     ac  p1    , hr  s1        ;   first block:= first block + 1; return;

e1:  hs  e4                    ; TRACK TO FREE DRUM: sum track;
     pmn p1    , dl  1b        ;
     ar  1b    , ck  -10       ;   group:= first free : 960 + 960;
     ga  r1    , cl  -10       ;
     vk[group] , ga  r1        ;   track:= first free mod 960;
     vk[track] , sk  b1        ;   write track (track buf);
     arn b     , ac  p1        ;   first free:= first free + 1;
     vk  960   , hr  s1        ;   return;

e2:  hs  e4                    ; TRACK TO BUF: sum track;
     arn p     , sr  5b        ;
     sr  2b                    ;   if buf rel + 40 > block length then
     hs (b11)          LT      ;   call (end action); will change buf rel;
     arn 5b    , ar  7b        ;   output track to (buffer[41 + buf rel]);
     us  0     , arn 2b        ;   buf rel:= buf rel + 40;
     ac  5b    , hr  s1        ;   return;

e:   hs  e4                    ; OUT DISPLACE: sum track;
     is (b2)   , vk  sd9-d8    ;   select(track read + displacement)
     sk  b1    , vk  960       ;   write(track buf)
     hr  s1                    ;   return;

c79:                           ;
e4:  gp  a5    , ppn 40        ; SUM TRACK: save p;
a6:  pp  p-1   , ar  pb1       ;
     ar  2     D       LA      ;   for p:= 39 step -1 until 0 do
     ar  1     D       LB      ;   sum:= sum + track buf[p] + marks;
     bs  p     , hv  a6        ;
a5:  pp  0     , ac  11b       ;   restore p;
     hr  s1                    ;   return;
```

[START ACTIONS]

```
e6:  arn 15b   , sr  14b   ; DISPLACE AREA: if to drum base > from drum base
     hv  a21         LT    ;   then alarm(<<move trouble>);
     sc  12b   , pm  12b   ;   first block in catalog:= first block + to drum base
     hv  s1                ;   - from drum base; return;

e7:  arn 12b   , sr  p1    ; AREA IN FREE DRUM: if first free block > first block
     hv  a21         LT    ;   in prim catalog then alarm (<<move trouble>);
     pm  p1    , hv  s1    ;   first block in catalog:= first free block; return;

e8:  pm  9b    , arn p     ; BUF AREA:
     sr  b     , ml  2b    ;   blocks:= (blocks im prim catalog × 40)
     dl  p     , gr  9b    ;   + block length - 1) : block length;
     arn p1    , ar  p2    ;   first block in catalog:= first block + unit;
     hr  s1    X           ;   return; may be called from carr area;

e12: pm  4b15  , gm  p1    ; CARR AREA: first carr block:= base next carr progr;
     hs  e8                ;   call(buf area);
     arn 9b    , ac  4b15  ;   base next carr progr:= base next carr progr
     hv  s1                ;   + blocks; return;

d47:                       ; ENTRY AFTER SPECIAL INIT:
     vk  d50   , vk  d8    ; INIT TRACK 0-1:
     lk  b1    , vk  1d8   ;   read track 0 and 1;
     lk  40b1  , vkn 960   ;
a7:  ar  37b1              ;   R:= sum of words 37 to 79 + sum of marks;
     ar  2     D      LA    ;
     hv (a7)   Dt  1   NB   ;   track 1 sum:= track 1 sum - R;
     sc  79b1  , ar  79b1  ;   R:= R + track 1 sum;
     arn 512   D       NZ  ;   if overflow then begin
     ac  d7+b1 , ac  75b1  ;   balance := balance + 512.9;
     vk  d9    , sk  b1    ;   track 1 sum:= track 1 sum + 512.9 end;
     vk  1d9   , sk  40b1  ;   write track 0 and 1;
a8:  vk  d50   , it  1     ;
     vk (b2)   , lk  b1    ;   for track read:= track 2 of help step 1 until
     vk  960   , hs  e     ;   last main help track do
     bt  d37-2 t   -1      ;   begin read track (track read);
     hv  a8                ;   out displace end;
     vk  d9+d37-1 , lk b1  ;   read search track;
     vk (r-1)  , arn 11b   ;   help sum:= help sum - sum;
     sc  d6-c+b1, sk b1    ;   write search track;
```

```
b3:  arn b4    t  1          ; SCAN CAT: current:= current + 1;
     hv  a23            LC   ;   if cat[current] a-marked then goto area;
     hv  a9             LA   ;
     hv  a10            NC   ;   if - - - 0-marked then goto text;
     hv  a11            LZ   ;   if - - - =0b then goto end scan;
a23: qq  b18   , hs  e5       ;   alarm(‡<format‡);

a10: tl  -6    , nc  -6       ; text: if cat[current] -‡ end text then
     hv  b3                   ;   goto scan cat;
     acn(b3)           MB     ;   b-mark (cat[current]);
     is (b3)   , arn s1       ;   if cat[current + 1] -‡ specification then
     qq        V       LT     ;   begin
     hv  a24   X       NC     ;     if spec required then
b20: bs  0     , hv  a25      ;       alarm(‡<program call‡);
     [spec required]          ;       goto scan cat;
     hv  b3                   ;   end;
a24: xrn       , cl  10       ;
     ck  20    , tk  30       ;   if tracks read to core × 40
     ck  10    , ml  2b       ;   + core address for first track
     xr        , sr  16b      ;   < 2 + max core for call then
     hv  b3          LT       ;   goto scan cat;
     hv  b3          NPA      ;   if -, program then goto scan cat;
a25: qq  b19   , hs  e5       ;   alarm(‡<program call‡);

a9:  ga  r1    , tl  -7       ; AREA: indicator:= area bits;
     pi  0     , nc  d3       ;
     hv  a23         LPB      ;   if reserved ∧ kind ≠ free kind then goto alarm;
     ga  b26                  ;   store kind
     ga  b5    , tl  -25      ;
     tln 16    , gr  9b       ;   work:= tracks := no of blocks;
     ck  -10   , ga  b6       ;
     tln 16    , gr  12b      ;   track read:= first track:= bits 23-39;
     ck  -10   , ga  b2       ;
     pa  b20                  ;   spec required:= false;
     hv  a15         NPA      ;   if -, program then goto end area;
     bs (b6)   t   19         ;   if tracks > 19 then
     pa  b20   t   1          ;   spec required:= true;
     ncn(b5)   , pa  b20      ;   if -, drum then spec required := false;
b5:  arn[kind] Vt  b7  NPB    ;   R:= if -, reserved then action tabel [kind]
     arn b8                   ;   else free actions;
     gt  b10   , gt  b24      ;
     ck  -10   , gt  b9       ;   unpack outtrack action, start action,
     ck  -10   , ga  b25      ;
     ga  b11   , gt  b22      ;   end action, parameter base.
b22h:grn 5b    , pp  0        ;   buf rel:= 0; p:= parameter base;
b9h: qq  b21   , hs  0        ;   call start action, prepare for alarm(‡<kind‡);
     arn(b3)   , cl  -16      ;
     pm  9b    , cl  -16      ;   cat[current]:= bits 0-7 + blocks.23
     ck  -8    , gr (b3)      ;   + first track.39;
     grn 11b   , vk  d50      ;   sum:= 0; select load image group;
b2:  vk  1d8   , lk  b1       ;   for i:= tracks step -1 until 1 do
     [track read]             ;   begin read track(track read);
b10h:vk  960   , hs  0        ;     switch to outtrack action;
     vk  d50                  ;
b6:  ncn[tracks] t -1         ;     track read:= track read + 1;
     hv (b2)   Dt  1          ;   end;
     ps  40    , ps  s-1      ;
     grn sb1         M        ;   clear track buffer
     bs  s     , hh  r-2      ;
```

```
b26: can[kind] , hv  e9     ;   if kind ≠ drum then
b11: pa [end action] Dt b25;   begin end action:= end output
b24h:ps  r-1   , hv  0      ; rep: outtrack action; goto rep;
b25: hs [end action]        ;   call end action end;
e9:  srn 11b   VX           ;   M:= -sum; skip line;
a15: is (b3)   , pm  s1     ; end area: M:= cat[current + 1];
     is (b3)   , arn s1     ;   if cat[current + 1] a-marked then
     gm (b3)   Vt   1  LA   ;   begin current:= current + 1; cat[current]:= M
     hv  a23           NT   ;   end else if cat[current + 1] ≠ text then
     hv  b3                 ;   alarm(≪format≫); goto scan cat;
```

[Format of action tabel: Start action.9 adjust first block and blocks which then are assembled in the central routine. Out track action.19 sums and outputs one track, this may call the end action to output a block. End action.29 is blind in case of drum program but outputs the last block in other cases. Param base.39 points to 3 cells containing: block length, first block, unit.]

```
[-4] qq e5                        ; constant: alarm
[-3] qq e5                        ; not used: alarm
[-2] qq e5                        ; sy-progr: alarm
[-1] qq e5                        ; ly-progr: alarm
b7:  qq e6 + e.19+ e9.29+  0.39   ; drum: displace area , out displace, blind, blind
[1]  qq e8 +e2.19+ e3.29+b14.39   ; disc: buf area      , track to buf, end disc, di
[2]  qq e12+e2.19+e10.29+b15.39   ; carr: carr area     , track to buf, end carr, ca
[3]  qq e8 +e2.19+e11.29+b16.39   ; tape: buf area      , track to buf, end tape, ta
b8: < d3                          ;
     qq e8 +e2.19+ e3.29+b13.39   ; free disc: buf area , track to buf, end disc, fr
x    qq e7 +e1.19+ e9.29+b13.39   ; free drum: area in free, track to free, blind, f
>                                 ;

a11: arn 1b13  , ac  b12    ; END SCAN: free:= free + to free.39
     tk  16    , sc  b12    ;   - to free.23;
     vk  960   , vk  2d9    ;   initialise core 0-9
     lk  0     , vk  960    ;

a16h:pp  b12   , ps  0      ;   get:= address of free word;
     srn 3     Dt   1       ; fill cat track: track buf[39]:= rel track.9c;
     gr  39b1          MC   ;   for s:= 0 step 1 until 38 do
a17: arn p             IRC  ;   begin track buf[s]:= cat[get];
     pin 1             LC   ;     if c-marked then
     gr  sb1           MRC  ;     track buf[s]:= 0b;
     ar  1     D       LRB  ;
     ar  2     D       LRA  ;     track buf[39]:= track buf[39]
     sc  39b1               ;     - track buf[s] - marks;
     pp  p1    , ps  s1     ;     get:= get + 1;
     bs  s473  , hv  a17    ;   end;
a18: vk  d21-1 t   1        ;   cat track:= cat track + 1;
     sk  b1    , vk  960    ;   write(cat track);
     ps (b3)   , it  p512   ;   if get < last scanned then
     bs  s513  , hh  a16    ;   goto fill cat track;
     pp  b23   , vy  16     ;   p:= alarm text;
     is (a18)  , bs s-d21-d23+1 ; if cat track - first cat track + 1
     ps  r1    , hv  a14        ; > no of cat tracks then alarm(≪cat length≫)
     hsf 2                      ; call help
d2=i                            ;
a:   hv  d47                    ; prepare loading of special init
```

```
i=800                                   ;
b12=i, b4=11                            ; define place of primitive catalog
<d3                                     ; free area:
qq d3.2+1.3+1.6+d52.13+d4.23+8.27,      ; special, inhibit, free blocks, first free
x                                       ; initially:
qq d3.2+1.3+1.6+d22.15+d22.17,          ; special, inhibit, free top, 0
>                                       ; finished in Init Help.
qq d23.7+d46.29+d5.39,                  ; max free:
tfree;                                  ; catalog tracks, booked, min first free

d19=d19-960                             ; work:
qq 0.2+1.3+d32.5,                       ; special, work in free, tracks, first track
<-d32+1                                 ; if -, work in free then
qq d34.23+d19.29-d19.33+d33.39,         ; load max work: work tracks, first work
>                                       ;
twork;                                  ;
qq -4.2+2.19+6.29+67.39,                ; date:
qq 0.9,                                 ; constant, day, month, year
tdate;                                  ; run number

qq 26.23+d19.29-d19.33+d16.39,          ; image:
timage;                                 ;     26 tracks, first track of image
d19=d19+960                             ;

qq -1.2+d17.19+1016.29,                 ; reader:
tr;                                     ; ly-medium, 0 or 3, bits 7-9

qq -1.2+1.19+1016.29,                   ; typewriter:
tt;                                     ; ly-medium, 1     , bits 7-9

qq -2.2+32.19+903.29,                   ; perforator:
tp;                                     ; sy-medium, 32    , bits 3-6

qq -2.2+8.19+903.29,                    ; lineprinter:
tl;                                     ; sy-medium, 8     , bits 3-6

qq -2.2+16.19+903.29,                   ; writer:
tw;                                     ; sy-medium, 16    , bits 3-6

qq -2.2+0.19+1023.29,                   ; no-alarm unit:
tx;                                     ; sy-medium, 0     , no bits active

d48=i, d39=0, d1=d1+1d37                ; prepare loading aux. programs
qqf                                     ; aux only:= false;
e                                       ; end image block
[STOP, SUM] a i init help
s
```

;slip<

[7.8.1967 P1 page 1

[This program includes the Gier algol 4 translator in the Help 3 system, or punches a paper tape version of the translator. The translator should be placed on the drum before loading of this program. If wanted, the translator may be moved to another medium. The program enters a description in the catalog. The program requires the aux programs: res, set, move and setsum]

```
b a50, b30, c20, d50          ;
iP1, include algol:
d i=15, d37=0                 ;

d24: qq [first track]         ;   first track of translator;
d i=i-1                       ;

itype: actual first track of translator
s                             ;

itype: d35 = kind of final translator medium
s                             ;

<-d35,d37=1>                  ;   if kind less 0
<d35-6,d37=1>                 ;    ∨ kind gr 6
<d35-3,<-d35+6                ;    ∨ kind gr 3 ∧ kind less 6
 d37=1                        ;   then begin message wrong kind; hsf 2 end;
>                             ;

<d37, iwrong kind
   hsf2                       ;
  ei-1                        ;
>                             ;

c:   qq  1.39                 ;
c5:  ttranslator medium;      ;
c9:  qq  40.39                ;
c10: qq  39.39                ;
c11: qq  400.39               ;
c12: qq  512.39               ;
```

```
a18: vy  17    , sy  29    ; alarm:   select(type writer);
     pt  a22   , hh  r2    ;   write red;   to help := true;  goto text;
a19: pt  a22   t   1       ; writetext:   to help := false;
a25: vy  17    , sy  64    ;   select(type writer);
     it (s)    , pa  r1    ; text:   writecr;
a20: pmn -1    X           ; next word:   M := 0;   R := next text word;
[1]  cl  34    , ck  -4    ;   RM := RM shift 34;
     ga  a21   , ca  15    ; next char:   R := R shift -4;   char := RADDR;
     hv (a20)  D   1       ;   if char=15 then goto next word;
     ca  10    , hh  a22   ;   if char=10 then goto end text;
     ca  63    , it  1     ;   if char=63 then char := char + 1;
a21: sy [char] , cln -6    ;   writechar(char);   R := 0;   RM := RM shift -6;
a22: hh  1a20  , bs[tohelp];   goto next char;
     vy  33    , hr  s1    ; end text:   select(type writer input and puncher
     hsf 2                 ;   if to help then hsf 2 else return;

a12: hv (a14)  D   1       ; next track:    track := track + 1;  goto SELECT;
a13: dln a16   , ar  a16   ; to core:
     ck  -10   , ga  a26   ;   group := M:960 + 960;
     cl  -10   , ga  a14   ;   track := M mod 960;
a14: is [track], can s-960 ; SELECT:  if track = 960 then
     pa  a14   , it  1     ;    begin track := 0;  group := group + 1 end;
a26: vk [group], vk (a14)  ;   select(group);  select(track);
a15: lk  d2    , vk (a14)  ;   from drum;   wait drum;
     qq (a15)  t   40  LZA ;   if count base then increase track place;
     hr  s1                ;   return;
a16: qq  960.39            ;   960;

d20: gs  d21   , pmn d24   ; GET GP SEGMENT:
     hsn a13          IZA  ;   count base := true;
     arn d2    , ga  a1    ;   to core(first track translator);
a1:  pp [GPsize],vy  33    ;   p := size := part 1 of first word GP;
     pp  p-40  , ps  r-1   ;   for p := p-40 while p gr 0 do
     bs  p     , hv  a12   ;     next track;
     pa  a15   t   d2      ;   restore track buffer pointer;
     arn d9    , ga  r1    ;
     pp[reltab]t   d2      ;
<-d35+6                    ;
     gp  b1    , ck  21    ;
×    gp  a10   , ck  21    ;
>                          ;
×    gp  a10   , ck  21    ;   take abs address segm table;
>                          ;
d21: hv [old s]t   1   IOB ;   OB := GP name(e44);  return;


<-d35+6                    ;
b10: grn b2    , hs  d20   ; START INTERNAL:
[1]  pp  -1    , pp  p1    ;   blocks := 0;  GET GP SEGMENT;
b1:  pm p[base], ncn p-14  ;   for p := 0 step 1 until 16 do
     tl  9     , tln -9    ;     blocks := blocks + (if std proc code segm
     tln 19    , ac  b2    ;          then bits(0,19,segm table[p])
     bs  p496  , hh  1b10  ;          else bits(10,19,segm table[p]));
```

```
        arn d9     , gt  r1    ;
        arn 184    D   [e4]    ;   specword := (184+e4) pos 19
        ck  -10    , gt  b3    ;         + ((GP size+39):40) pos 9
        ck  -10    , ac  b3    ;         + (184+e4) pos 29;
        pmn(a1)    D X t -1    ;
        cl  10     , dln c9    ;
        ar  c      , ck  -10   ;
        ga  b3     , arn b2    ;   part 1 of specword := tracks in GP;
        sr  c      X           ;   move parameter 1 :=
        dln c9     , ar  c     ;     tracks in translator pos 23
        tk  16     , ar  d24   ;   + first track translator pos 39;
        gr  b9     , grm -1    ;   remove core inhibition;
        bs  511d35 V      LOB  ;   if GP name(e44) = 1 ∧ d35 less 1
        bs  d35                 ;    ∨GP name(e44) = 0 ∧ d36 gr 0
        qq  c5     , hs  a18   ;   then alarm(translator medium);
        arn b2     , sr  c     ;
        xr         , dln d13   ;
        ar  c      , gr  b2    ;   blocks := (blocks - 1):blocklength + 1;
        arn b2     , nc  39    ;
        acn(r-1)          MB   ;
        hv (r-2)   D t 1  LZ    ;

d d36=0                         ;
<-d35+2,iif reserved then type: d36=1
s>                              ;

<-d35+6                         ;
     hs  1                      ;
     hv  b5                     ;
<d36,tres;                      ;
x<-d35+6                        ;
     tset;                      ;
     qqf d35.39                 ;
>                               ;
<-d35+6                         ;
b2:  qqf [blocks]               ;   if reserved then res else (set,kind),
     qqf 3.39                   ;
     qqf                        ;
     qqf                        ;   typein,

<-d35+1                         ; DRUM:
  d i=i-3, d13=c9               ;
<-d36+1,itype: final first track
  s                             ;
>                               ;

<d35,<-d35+2                    ; DISC:
  d i=i-3                       ;
<-d36+1,itype: disc no, first block
  s                             ;
>                               ;

<d35-1,<-d35+3                  ; CARROUSEL:
  d i=i-2, d13=c12              ;
  itype: reel, first block
  s                             ;
>                               ;

<d35-2,<-d35+4                  ; TAPE:
  d i=i-3, d13=c11              ;
  itype: unit, file, block
  s                             ;
>                               ;
```

```
<-d35+6                                ;
     qq 39    ,                        ;  p
     qq 57    ,                        ;  i
     qq 52    ,                        ;  d
     qqf 0                             ;  0

b4: itype: tname of translator;
    s                                  ;

b3: qqf [specword]                     ;  name,specword
    qqf 0,                             ;  <

b5:  arn b4-1  t   1   IRC ; move:
[1]  gr  b9    t   1   MRC ;
[2]  gr  b6    t   1   MRC ;  set name of translator as second parameter
     hv  b5            NRC ;  to move and parameter to setsum;
     grn (1b5)         MC  ;
     grn (2b5)         MC  ;

     hs  1                 ;  call move
     hv  b8                ;  and return to setsum;
     tmove;                ;
     qq  50    ,           ;  move, b
b9:  qqf [moveparameter 1] ;        actual place on drum,
     [name of translator]  ;        name of translator<

d i=50i                    ;

b8:  hs  1                 ;  call setsum
     hsf 2                 ;  and return to Help;
     tsetsum;              ;  setsum,name of translator<
d b6=i-1                   ;
     [name of translator]  ;

<d35,<-d35+2, itype: if CDC disc then 640 else 400
d13: qq [block length]     ;
     d i=i-1               ;
s                          ;
>                          ;

<-d35+6                    ;
d d2=50i, d9=1d2           ;
e b10                      ;
>                          ;
```

[The following code punches two paper tapes. The first tape consists of GP and pass1, the second of the other translator segments (passes). The sum of all segments is checked. The GP segment is modified before output: Track 5e14 (block for next segment 2) is replaced by the papertape version (see below d ff), and in tail of GP a code for writing GP on drum after input is inserted (see below d1 ff). The first tape may be read in by means of the auxiliary program algol. The code is only loaded if kind (d35) = 6]

```
b11: hs  d20                    ;  GET GP SEGMENT;  ....
     pp (a1)   V        NO      ;  if GP name(e44) ≠ 0 then
     qq  c5    , hs  a18        ;   then alarm(translator medium);
     hsn a23                    ;  p := size;  R := 0;  sum;
     pmn(d2)  DXV t -1 LZ       ;  if R≠0 then alarm(pass sum);
     qq  c4    , hs  a18        ;
     cl  10    , dln c9         ;  GPsize := GPsize - 1;
     ck  -10   , ga  d23        ;  tracks := GPsize:40;
     pp (d2)   t   d4           ;  p := size := GPsize+init code length;
     arn  d9   , pa  d12        ;
     gt  d22   , ck  10         ;  part 2 of first word GP := core base
     ga  d1    , ga  r1         ;                              + old size;
     ps [e4]   , it  sd5        ;  set e13 and e4 in Get next segment 2;
d12: it  p0    , pt  d2         ;
     it  s21   , pa  d          ;
     it  s263  , pa  d13        ;  set e16-1 in Get next segment 2;
     it  pd6   , pa  a3         ;
     it  p     , pt (a10)       ;  part 2 of GP segm word := size;
     pp  d7    , vy  33         ;  select(typewriter and puncher);
     pm (d2)   D X              ;  words := size;
     ck  10    , gr  c7         ;
a2:  pm  pd              IRC     ;  Move init code
a3:  gm  p0              MRC     ;  and block for next segment 2
     pp  p-1   , can p-39       ;  to final places;
     pa  a3    t   d8           ;
     bs  p     , hv  a2         ;
     pp (d2)                    ;
     pp  p-1   , hsn a23        ;  p := size-1; R:=0; sum;
     pp (d2)   , gr  pd11       ;
     srn pd11  , gr  pd11       ;  last word GP := -R;
     pp  -1    , hs  a17        ;  p := -1; spaces;
     qq  c2    , hs  a19        ;  writetext(tear off. first tape...);
     it (a10)  , pt  a24        ;  base := base segm table;
```

```
a4:  lyn[char] D              ;   typechar;
a5:  hs  a17                  ; PUNCH SEGMENT:
     sy  13    , grn c1       ;   spaces;  writetext(aa);  sum := 0;
a6:  pp  p1                   ; NEXT WORD:   p := p+1;
     bs  p-39  V       NZA     ;   if -,count base then begin
     hh  a7                    ;   if p gr 39 then
     pp  0     , hs  a12       ;    begin p := 0; next track end;
     arn pd2                   ;   M := R := core[track buffer base + p];
     ar  2     D       LA      ;   if mark a then R := R + 2 pos 9;
     ar  1     D       LB      ;   if mark b then R := R + 1 pos 9;
a7h: ac  c1    , pmn pd2       ;   sum := sum + R ;  R := 0; end;
     ar  2.2   D       LA      ;   if mark a then R := R + 2 pos 2;
     ar  1.2   D       LB      ;   if mark b then R := R + 1 pos 2 ;
     pa  a9    X   5           ;   swap;   RM := RM shift 32;
a8h: cl  32    , ga  r1        ;   for i := 1 step 1 until 6 do
     sy [char] , cl  -7        ;    begin writechar(bits(3,9,R));
a9:  bt  5     t   -1          ;    RM := RM shift -7
     hh  a8                    ;    end;
     arn c7    , sr  c         ;   words := words - 1;
     arn c1    V       LZ      ;   if words ≠ 0 then goto NEXT WORD;
     gr  c7    , hv  a6        ;   if sum ≠ 0 then
     pin 0               LZA   ;    alarm(pass sum);  count base := false;
a10: arn[segm] XV t1    LZ      ;   R := next segm word;
     qq  c4    , hs  a18       ;
a24: ns (a10)  , ps s[base];    ;
     ncn s14   X               ;   if not std proc code segm word
     tk  10    , ck  -10       ;    then clear part 1 of R;
     tl  -20   , gr  c7        ;   words := bits(0,19,R);
     can s2    , hv  a11       ;   if pass2 segm word then goto NEXT TAPE;
     ncn s17   , hv  a5        ;   if more segments then goto PUNCH SEGMENT;
     pt  a22   , hs  a17       ;   spaces;
     qq  c6    , hs  a25       ;   writetext(tear off. end...);   hsf 2;
a11: hs  a17                   ; NEXT TAPE:   spaces;
     qq  c3    , hs  a19       ;   writetext(tear off. next...);
     hv  a4                    ;   goto PUNCH SEGMENT;

a23: pp  p-1   , ar  pd2       ; sum:  for p:=p, p-1 while p gr 0 do
     ar  2     D       LA      ;   begin R := R + word[track buffer base + p];
     ar  1     D       LB      ;    R := R + marks pos 9;
     bs  p     , hv  a23       ;   end;
     ck  0     , hr  s1        ;   clear R00;  return;

c1:  qq [sum]
c7:  qq [words]
c2:  ttear off. first tape is punched by typing SPACE;
c3:  ttear off. next tape is punched by typing SPACE;
c4:  tpass sum;
c6:  ttear off. end paper tapes;
```

d: [Get next segment 2, paper tape version]

```
b a2, b10, c10, e28          ;
d e4=0,e13=21,e28=163e13     ;
d e16=80e28,e11=34e28        ;

c1:  gmn[e13]  V X      IZC ; Get segment:  e13 := R := M; goto fetch e4;
c2:  grn rb6                ; Get word:  segment transp := f; R := 0;
d22h:gs  rc8   , ps [e4]    ; fetch e4: save s := s;  s := e4;
     hh  rc5           LZ   ;   if R=0 then goto read start;
     gt  r     , pp [words];   sum ok := transport ok := t;
     ga  rb5   , tk  20    ;   p := bits(10,19,R); first core := bits(0,9,R
     nc  2.4   , hv  rc5   ;   M := passes text;
                           ;   if bits(20,29,R) = pass no 2 then
c3h: pm  rb8   , ud  s14   ; write pause text:
     sy  64    , sy  58    ;     begin
     sy  62    , sy  0     ;       select(type writer);
                           ;       writecr; write LC;
c4:  cln -6    , ck  -4    ;       write black; write space;
     ga  rb1   , nc  2     ;       write M as 6 characters;
b1:  sy [char] , hv  rc4   ;       comment only one word no CR;
     lyn rb1   , ud  s16   ;       lyn; select(normal);
     can p     , hv  rc7   ;       if p=0 then goto finis;
c5:                        ;     end;
c5h: grn s41e13, ud  s15   ; read start:  sum := 0;
     ca 13[aa], grn rb2    ; read word:   select(secondary reader);
b2:  lyn rb1   , hh  rc5   ;   if code fresh in core then while lyn ≠ <aa> d
                           ;   code fresh in core := f;
     pa  rb3   t   5       ;
c6h: lyn rb4   , tl  -7    ;   R := next word from reader; marks in RC;
b3:  bt[charct]V t -1  NT  ;   if parity error then
     hv  rc9           IZC ;     begin sum ok := transport ok := f; exit end;
     ly  rb1   , hh  rc6   ;
     tl  10    , ud  s16   ;   select(normal);
b4:  pi [marks]t   1020    ;   store[first core] MRC := R;
d13:                       ;
b5:  gr  e16-1 [first] MRC ;   if -,segment transp then
b6:  qq (rb5)  V t   1     ;     begin R := store[first core]; exit end;
     arn(rb5)  , hv  rc9   ;   first core := first core + 1;
     ar  2     D       LRA ;
     ar  1     D       LRB ;   sum := sum + R + RC pos 9;
     ac  s41e13, pp  p-1   ;   p := p-1;
     ncn p     , hhn rc5   ;   if p ≠ 0 then goto read word;

     arn s41e13        IZB ;   sum ok := sum = 0;
     pm  rb7   , arn se13  ;   M := ready text;
     tk  20    , ca 1.0+1.4;   if pass no 1 (e13) ∧ sum = 0 then
     hh  rc3           LZB ;     goto write pause text;
c7:  pm  se13              ; finis:   M := e13;
c8:
c9:  ps [save s],hr  s1    ; exit:  s := save s;  return;

b7:  tready;               ;
b8:  tpasses;              ;
```

```
d1:  [Init GP, paper tape version]

a2:  pp [e4]    , arn p5     ;   GP base := first track translator - 1;
     sr  p3e28  , gr  p11    ;
a:   arn p5     , hs  pe11   ;
     sk  pe28   t   40       ;   GP except fixed GP to first track ff;
     arn p3e28  , ac  p5     ;
     sc  p4     , it  -1     ;   first track := first track + GP tracks;
d23: ncn[tracks],hv  ra       ;   available tracks := available tracks
     vk (p5e11), hh  ra2-37;      ...                 + GP tracks;
     qq [GP sum]             ;   goto Init comp buffer or paper tape version
e                            ;
d d4 = i-d1                  ;   d4 := init GP code length;


a17: pa  r1     t   100      ; spaces:      ....    .....     ..
     bt [100]   t   -1       ;   for i := 1 step 1 until 100 do
     sy  0      , hv  r-1    ;     writechar(0);
     hr  s1                  ;

d d2=i, d3=1d2, d5=-d4+184   ;
d d6=d2-39d4-1, d7=39d4      ;
d d8=d2+200,d9=1d2,d11=d2-1; 
u b11                        ;
e                            ;
e                            ;
s                            ;
```

[Call: algol, <list> <

<list> ::= <name>|<lineinterval>|s|i|d|n|<empty>|<list><list>

<name> ::= <name of sy-medium>|<name of input medium>

The sy-medium, called normal out, will be used for possible output of source program, pass information and pass output. If no name in the list describes a sy-medium the output unit will be the current selected Help output medium. The translator will be searched under the name <name>. This is also true if <name> describes a ly-medium (transient translator). If no <name> in the list describes a input medium, the translator will be searched under the name : ga4.

<lineinterval> ::= <help number>

Every <lineinterval>th line in the source program will be copied to normal out unit. If no <lineinterval> is present, no output of the source program will be made.

s (skip between punch-off and punch-on)
The parameter will cause program text between punch-off and punch-on to be skipped.

i (information wanted)
Pass information will be output on normal out unit.

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

n (no indexcheck)
No check of references to subscripted variables is generated.

Source program and help input medium.

If the help input medium is typewriter without explicitly having been specified as such, then the source program medium will be reader, and after translation the help input medium will again be typewriter.
In all other cases the source input medium will be the current help input medium and after translation the help input medium will be the last used source program medium.

Error output will appear on the current selected help sy-medium.

Alarm output will appear on helps alarm output unit.

Type out: typewriter output.

Type in: typewriter input.]

```
<-d55+1,iversion
s>


[Here follows STOPCODE, CLEARCODE]

b k=d1, i=120, a25, b27                    ;
d a=i-3                                    ;

[ 3e4] qq  [lineinterval.39]               ; Translator parameters:
[ 4e4] qq  [no of tracks in work.39];      see: A Manual of Gier Algol 4
[ 5e4] qq  [first track of work.39] ;      section 13.2.5;
[ 6e4] qq  c64.9-1.9+c63.19                ;
[ 7e4] qq  -1.2+d17.19+1016.29             ;
[ 8e4] qq  1.9+d11.39                      ;
[ 9e4] qq  17.29+d17.39                    ;
[10e4] qq  1.7                             ;
[11e4] qqf -1.2+63.29+1023.39              ; also used as mask
[12e4] qqf 6.39                            ; also used as constant
[13e4] qqf 6.19+1.39                       ; also used as constant
[14e4] qqf d14.39,                         ; also used as constant

b25: pmn 1.3   D X     IZA ; NEXT WORD:   i := 0;  first char := true;
a18: tl  -7    , ly  a19   ; NEXT CHAR:   i := i + 1; RM := RM shift -7;
a19: pi [char] t   508 LZA ;   R := R + lyn pos 9;  if first char then
     hs  c24           LT  ;   begin  RC := bits(8,9,char); first char:=false e
     qq  a22   X           ;   if char<0 then alarm(‡<<parity>);
     hv  a18   X       LZ  ;   if i≠7 then goto NEXT CHAR;
     xr        , tl  3     ;   RM := RM shift 3;
     gr  p183  t  1    MRC ;   c:=c+1; core[183+c] := R mark RC;
[-1] ga  a20   , grn r     ;   if first word then
a20: ncn [words] t  -1     ;    begin words:=RADDR; first word:=false end;
     hv  s1                ;   words:=words-1; if words≠0 then goto NEXT WORD;
a13: grn r2                ; NOT BUFFER:
a12: pm  2a    X t  1  IRC ; BUFFER MEDIUM TRANSLATOR:
     pm  e1    X t  1  LRB ;  move translator parameters
a16: gr [2e4]  t    1      ;  to core[3e4:14e4]; comment in case of
     hv  a12           NRA ;  buffer medium the parameters
     gt  b7    , vk  960   ;  11e4:14e4 are fetched from help;
     vk  d11   , sk  a24   ;  to drum(help parameter track,
     vk  d11   , ps (a16)  ;          AFTER TRANSLATION);
     pp (s170) , pin 0     ; CHECK SUM GP:  R :=0;
b:   ar  s169  t   1   IRC ;  for p:=word size GP step -1 until 0 do
[1]  gi  a20-1 , ar  a20-1 ;   begin
     ck  0     , pp  p-1   ;    R := R + core[184e4+p];
     bs  p     , hv  b     ;    R := R + marks pos 9; clear R00;
     hs  c24           NZ  ;   end; if R≠0 then alarm(‡<<sum>);
     qq  a21   , hv  s170  ;  goto first word of GP;

a22: tparity;              ;
a21: tsum;                 ;
```

[The following code is placed on Help parameter track during translation and is entered at b11 with description of translated program in R]

```
                                    ; AFTER TRANSLATION:
a24:  qq [Help param(0)]            ;   This word is used by Help and is set below;
```

[The instruction in b7 is replaced by : qq
if the translator is not on tape]

```
<d41                                ;   if buffermedia treated
b7:   is  64     , us s[unit];      ;     ∧ tape then rewind translator tape;
>                                   ;
b11:  ps  rb1    V  -c41 NZ         ;   if R≠0 then set return(OK) else
[1]   ps  c45-c41                   ;   set return(SET TYPEWR);
b24:  gr  rb11   , pm [7e4]         ;   comment SET TYPEWR is an entry in Help;
      gm  r1b11  , vk  960          ;   save description of translated program;
      vk  1d9    , lk  40           ;   save description of latest input medium;
      vk  960    , hv  c53          ;   restore main Help; return to s+c41;
b1:   sy  64     , sy  38[o]        ; OK:   writecr; writetext(<<ok>);
      sy  34[k]  , hsn c2           ;   get free;  to core(first catalog track);
      lk  d14    , vk  (c4)         ;   wait drum;
      arn 3d14   , ac  39d14        ;   checksum := checksum - work area word;
      tl  -32    , tk  32           ;   work area word := bits(0,7,work area word) pos 7
      ar  rb11   , sc  39d14        ;   + description of translated program;
      gr  3d14   , sk  d14          ;   checksum := checksum + work area word;
      vk  (c4)   , arn r1b11        ;   modified catalog track back to drum;  wait drum;
b4:   bs  0      , hv  c45          ;   if restore typewriter then goto SET TYPEWR;
      ps  rb6    V       LT         ; MEDIUM DESCR:  R:= description of latest input med;
      qq  rb8    , ps  rb5          ;   if R<0 then set return(EXIT)
b5:   hh  3c28                      ;   else set return(SKIP);  ENTRY NO GET TRACK;
```

[The code 3c28 will initialize the input medium described in R
and return to s+1 in case of ly-medium or drum medium.
Otherwise return will be made by: hs (s-1). Cell c28 will hold
number of skipped characters]

```
[1b5] pm  -2     , hsn c3           ; SKIP:
      lk  d14    , vk  (c4)         ;   select track(bits(24,39,core[-2]));
      ps  r-3    , arn c28          ;   to core(place d14);  set return(r-3);
      sr  rb21   , gr  c28          ;   for i:=1 step 1 until skipped characters do
      hv  c27           NT          ;     read internal; comment return by: hr s3;

d b6=i-1                            ; EXIT:  select track(Help parameter track);
      vk  960    , vk  d11          ;   goto RETURN FROM AUX PROGRAM;
      hv  c51                       ; BUFFER INPUT MEDIUM:
<d41                                ;
b8:   arn rb20   , il  0            ;   only loaded if d41 is positive;
      arn c29-1  , ar  d14          ;   core[-6] := -4 pos 9 + 1 pos 19 + 7 pos 39
      gr  -6     , srn d14          ;   +buffer[4];
      sr  rb21   , ud  s-2          ;   core[-5] := -buffer[4] - 1;
      gp  -6     , hv  r1b5         ;   goto SKIP;
xb7: b8:>                           ;
b20:  qq  d14.9+1.19+4.39           ;
b21:  qq  1.39                      ;

<i-194                              ;
b21:>                               ; test load address;
```

```
b10:                                  ; ENTRY CALL algol: ....
<d35-2,us(-31)  t  -96>               ;   if I am on tape then rewind my tape;
      pm  d13   , gm  a24             ;   save param(0); comment used after translation;
      pp  d13-1                       ;   p:=base parameterlist;
a1:   gp  a3    , hs  c52             ; SCAN PARAMETERLIST: GET PARAM;
      hv  a5    , hh  a4              ;   if number then goto LINEINTERVAL;
      hv  a7    , tl  -7              ;   if letter then goto SPECIAL BITS;
<d41, nc  -4    , ca  -3              ;   if end list then goto AFTER SCAN;
a2:   qq  e8    , hs  c24             ; TRANSLATOR MEDIUM OR NORMAL OUT:
a3:   ps [p]    , nc  -2              ;   if kind=4 ∨ kind=5 ∨ -,buffermedia treated ∧
× a3: ps [p]    , nc  0               ;      (kind=1 ∨ kind=2 ∨ kind=3) then alarm;
a2:   hs  c24             NT          ;   address of translator medium name :=p;
      qq  e8    , nc  -2              ;   if kind≠2 then normal out := bits(10,19,areaword);
>     gs  a10   , hv  a1              ;   ....
      ck  17    , ga  9a              ;   goto SCAN PARAMETERLIST;
a4:   hv  a1    , ps  -1              ; SPECIAL BITS: s:= -1;          ...      ....
      ca  18[s] , psn s-1.2           ;   if letter s then s := s - 1 pos 2  else
      ca  57[i] , psn s-1.4           ;   if letter i then s := s - 1 pos 4  else
      ca  52[d] , psn s-1.6           ;   if letter d then s := s - 1 pos 6  else
      ca  37[n] , psn s-1.7           ;   if letter n then s := s - 1 pos 7
      hh  b23                   NZ    ;   else alarm;  ...      ....      .....
      pi (10a)  , it  s               ;   bits := (-5 pos 9) mask s;  comment 10e4;
      pi  -5    , hv  a6              ;   goto SCAN PARAMETERLIST;
a5:   gr  3a    , pi (10a)            ; LINEINTERVAL:   lineinterval := R;
      pi  1.5   t  -17  NZ            ;   if lineinterval≠0 then set bit(5,bits);
a6:   gi  10a   , hv  a1              ;   goto SCAN PARAMETERLIST;
a7:   arn 9a    , pi (10a)            ; AFTER SCAN:
      nc  0     , hv  b19             ;   if (lineprint wanted
      gk  9a                  LPA     ;       ∨ pass information wanted)
      gk  9a                  LPB     ;      ∧ normal out = 0 then
b19:  gk  a1    , it (a1)             ;   normal out := by;
      pt  9a    , pm  1c26            ;   error out := by;
      tln 6     , tk  3               ;   alarm out := Help alarm out;
      ac  9a    , hsn c2              ;   get free;
a10:  pp  a23   , ps  a15             ;   p := address of translator medium name;
      lk  d14   , vk (c4)             ;   set return(TRANSLATOR MEDIUM);
      pm d14+d45, tl  7               ;   to core(first catalog track);
      tln 16    , gr  4a              ;   set number of tracks and first track
      tln 16    , gr  5a              ;   of work in 4e4 and 5e4;
```

```
     arn -5     , pm  -2     ; INITIAL SOURCE INPUT MEDIUM:  ....
     hh  a8              LA  ;   if bits(40,41,core[-2]) = 0 then
<d41,hv  a9              LB  ; READER OR TYPEWRITER:  ...  ...
>    cln -14                 ;    begin core[7e4] := -1 pos 2 + reader pos 19;
     pa  b4     V   1    LO  ;     if core[-2] = vyn [by] t [mask] then
     it (-2)    , pt  7a    ;     part 2 of core[7e4] := part 1 of core [-2]
a8:  hv  c52    , gm  7a    ;     else restore typewriter := true;
                            ;    GET PARAM; comment return by hh s+2; ....
                            ;   end else if bits(40,41,core[-2]) = 2 then
[1a8]tl  -30    , sr  14a   ; DRUM:  begin core[7e4] := core[-2];
     pm (-3)    D X         ;   R := (part 1 of core[-5] )  pos 39;
     ar  a11    , tk  -30   ; ADD: core[7e4] := core[7e4] + ((R - d14)x6
     ml  12a    , tln 55    ;    + 5 + part 1 of core[-3]) pos 23;
     ac  7a     , hv  c52   ;     GET PARAM; comment return by hh s+2 (=a15+2);
<d41,                       ;     end else
a9:  gm  7a     X       IZA ; BUFFER MEDIUM:
     arn 13a    , il  0     ;   begin core[7e4] := core[-2]; M := core[-5];;
     arn 0      , mb  b22   ;    R := buffer[1]; clear bits(3,23,R);
     ac  7a     , srn 3     ;    core[7e4] := core[7e4] + R;
     hv  c52             LZA ;    if core[-5] = 0 then GET PARAM;
     gr  14a    , sr  b21   ;   d14 := -buffer[4];
     gr  -5                 ;    core[-5] := -buffer[4] - 1;
     hh  1a8    X           ;    swap; goto ADD;
>                           ;   end;
d a15 = i-2                 ;   define return from GET PARAM;
<d41,qq                     ;
     mb  11a    , gr  11a   ; TRANSLATOR MEDIUM:  ..
     ca  0      , it  a13   ;   core[11e4] := bits(0,2,area word) pos 9
     pt (c15)   V   a12 NT  ;   + bits(24,39,area word);  .....  .....
     grn b7     , hv  a17   ;   if kind ≠ 3 then tape := false; comment the
     nc  3.2    , grn b7    ;   instruction in b7 is replaced by: qq;
                            ;   if kind gr 5 then  ....
×    pt (c15)   V   a13 NT  ;    begin tape := false; goto TRANSIENT TRANSLATOR e
     hv  a17                ;   if kind = 0 then set return(NOT BUFFER) else
     mb  11a    , gr  11a   ;   set return(BUFFER MEDIUM TRANSLATOR);
>    arn (c15)  , nc  10    ;   if part 1 of spec word ≠10 (GP tracks)
     nc  11     , hh  b23   ;    ∧ part 1 of spec word ≠11
     ck  10     , gt  b26   ;   then alarm(param);
     grn a20-1  , hs  b26   ;   TEST;
     arn 2c     , ga  r1    ;   R := areaword;
     pi [bits]  , pp  d13-1 ;   p := base parameter list;
     hh  1c58            LPA ;   if program bit then PROGRAM CALL;
     qq  e8     , hs  c24   ;   alarm(kind);
```

```
a17: vy  d17    , lyn a19   ; TRANSIENT TRANSLATOR:  first word := true;  c := 0;
     nc  13     , hh  r-1   ;   select(reader);
     pp d14-183, hs  b25    ;   NEXT WORD;
     pa  b26    , hs  b25   ;   R := NEXT WORD;
     arn 2d14   , pm  1d14  ;   M := first word from reader;
     ps  b25-1  , gt  r1    ;   set return(NEXT WORD);
b26: it  -184   , pp [e4]   ; TEST:   p := e4 + (if TRANSIENT TRANSLATOR
                            ;                  ... then 0 else -184);
     bs  p494   t   503     ;   if p>9 ∧ p<18 then
     gr  p185   V       MC  ;   core[p+185] := R
b23: qq  e5     , hs  c24   ;   else alarm(param);
     gm  p184           MA  ;   core[p+184] := M;
[2]  it  p7     , pt  b24   ;   save 7e4;
[3]  it  p2     , pa  a16   ;   save 2e4;
     hv  s1                 ;   return;


a11: qq  5.9                ;
b12: tga4;                  ;   initial name of translator;
d a23=b12-2                 ;
b22: qq  -1.2               ;

<i-280                      ;
b23:>                       ;   test load address;
```

```
i=i+39, d=k-d1                          ;
b k=d42,i=0,a10                         ;
a1=d19-960                              ; a1=group no for image
a=d,<d35,a=1>                           ; a=no of blocks

<d39                                    ; if aux only then
i=d2,hs1                                ; begin
     hv a5                              ;  (if aux reserved then
<d36,tres;                              ;
     qqf a.39                           ;   res,no of blocks,
x<<d39                                  ;    else
     tset;                              ;    set,aux kind,no of blocks,
     qqf d35.39                         ;    type in)
     qqf a.39                           ;
[after i follows STOP,SUM and a sum character]
ia base,algol
s
[STOP,CLEAR]
><d39                                   ;
   qq  39,                              ; concat pid 0,
   qq  57,                              ;
   qq  52,                              ;
   qqf                                  ;
   talgol;                              ; algol,spec<
   qqf d.9+b10.19+120.29                ;
   qqf,                                 ;

a5: hs 1                                ;
    hv a6                               ;
    tmove;                              ; move,b loadplace,algol<
    qq 50,                              ;
    qqf d.23+d1.39+a1.29-a1.33          ;
    talgol;                             ;
    qqf,                                ;

a6: hs 1                                ;
    hv a7                               ;
    tsetsum;                            ;
    talgol;                             ;
    qqf,                                ;
d2=i                                    ; end else
a7: hsf 2                               ;
x                                       ;
i=d48                                   ;
   qq d35.2+11.7+d36.5+d.23+d1.39,      ; load to primitive catalog;
   qq,                                  ;
   talgol;                              ;
   qqd.9+b10.19+120.29                  ;
d48=i,qqf                               ;
>                                       ;
d1=d+d1   ;
e         ; end image load
e         ; end algol
[after i follows STOP,SUM and a sumcharacter]
ia algol
s
```

```
<-d55+1, iversion
s
>
[Here follows STOPCODE, CLEARCODE]0

b k=d1, i=40d13, a15, b15  ;   begin binin
b=79i                      ;   define base of working locations
     pp  d13-1 , hs  c52   ; ENTRY TO BININ: get first parameter;
     hh  r1    , hh  r-1   ;   if single then get next; if end list then alarm;
     hv  c58   , ga  rb11  ;   indicator:= bits 0-9; R:= area word;
b11: pi  0 [area bits]     ;
     qq        V      LQA  ;   if work area then R:= work as output;
     arn d45+d14      LTB  ;
     tl  -32   , tk  25    ;   if kind ≠ drum then alarmprint(<<kind>);
     nc  0     , hv  c57   ;   now track read is false (ZB = 0)
     tln 16    , gr  r3b   ;   upper:= first track + no of tracks;
     tln 16    , ac  r3b   ;
     gr  r2b   , gr  r5b   ;   track:= destination base:= first:= first track;
     gr  r1b   , ly  rb1   ;
     nc  13    , hh  r-1   ;   for i:= lyn while i ≠ aa do ;

a:   pmn r2b   , hs  c3    ; start track: select track(track);
     pp  0     , lk (rb4)  ;   read track to store buffer[0];
b6h: vk  960   , pp  0     ;
     [trackrel]              ;   p:= track rel - 1;
a1:  pp  p-1   , psn ra2   ; read label: prepare return to label;
a4:  ly  rb1               ; procedure inchar; Radr:= words:= lyn;
     ac (rb7)  DV     NT   ;   sum:= sum + Radr;
a6:  qq  rb8   , hs  c24   ;   if Radr < 0 then alarmprint(<<parity>);
     qq (rb9)  t  1        ;   characters:= characters + 1; return;
a2:  hh  s     , ps  ra3   ; label: prepare return to next word;
     bs (rb1)  Xt   63     ;   if words ≥ 64 then goto special;
a3:  hv  ra10  , it  -1    ; next word:
b1:  can -1    , hh  ra1   ;   if words = 0 then goto read label;
     [words]
     pmn 1.3   DX     IZA ;   for i:= 1 step 1 until 6 do

a5:  tl  -7    , ly  r1    ;   begin pack; inchar;
     pi  -1    t 256  LZA  ;     if i = 1 then RC:= R8 - 9;  ZA := 0;
b7:  ac  0     DVX    NT   ;   sum := sum + char;
     [sum]                 ;
     hv  ra6               ;     if Radr < 0 then alarmprint(<<parity>);
     hv  ra5   X      LZ   ;   end;
b9:  qq  0     Xt6         ;   characters:= characters + 6
     [characters]          ;
a7h: tl  3     , pp  p1    ;   R:= packed word;
b4:  gr  p130d13 X    MRC  ; outword: p:= p + 1; storebuffer[p]:= R, RC;
     hk  ra9          NZB  ;   if not busy ∧ -, trackread then readtrack;
     bs  p472  , hh  s     ;   if p < 40 then return;
```

```
a8:  hs  ra9          NZB ; procedure writetrack; if -, trackread then read tra
[1a8]arn r2b   , sr  r3b  ; writetrack no read:
     hv  c61          NT  ;   if track ≥ upper then alarmprint(‹<full›);
     pm  r2b              ;   track read:= false;
     hsn c3           IOB ;   select track(track);
     arn rb    , ac   r2b ;   track := track+1;
     arn(rb4)  , pp   0   ;   R:= last word stored; p:= 0;
     sk (rb4)  , pm   rb4 ;   write track from storebuffer[0];
     it (rb5)  , pa   rb4 ;   exchange (store buffer, drum buffer);
     ud  rb4              ;   storebuffer[0]:= R, RC;
     ga  rb5   , hh   s   ;   Raddr ≠0; return;

a9:  arn r2b   , ar  rb   ; procedure readtrack;
     sr  r3b              ;   if track + 1 ≥ upper then return;
     hr  s1           NT  ;
     ar  r3b   X          ;   select track (track + 1);
     hsn c3           IZB ;   read track to drumbuffer[0]; trackread:= true;
b5:  lk  171d13 , hr s1   ;   return;

a10: it (rb9)  , pt  rb2  ;   characters 1:= characters;
     it (rb7)  , pt  rb3  ;   sum 1:= sum;
     hsn ra4   , tk  -7   ;
     hs  ra4   , tk  -7   ;   k:= (inchar shift -14)
     hs  ra4   , xr       ;   + (inchar shift -7) + inchar;
     ca  64    , hv  ra11 ;   if words = 64 then goto repeat;
     ca  65    , hv  ra12 ;   if words = 65 then goto new destination;
b2h: tl  43    , ca  -1   ; check sum:
b3h: ck  10    , nc  -1   ;   if characters 1 ≠ k : 1024 ∨ sum 1 ≠ k mod 1024
     qq  rb10  , hs  c24  ;   then alarmprint(‹<tapesum›);
     hs  r1a8  , hs  c2   ;   write track no read; get free;
     arn 2b    , pm  5b   ;   R:= track; M:= destination base;
     pi (rb11) , hv  c74  ;   indicator:= area bits; goto adjust special;

a12: tl  43    , gt  rb6  ; new destination: trackrel:= k mod 1024;
     tl  -30   , ar  r1b  ;
     gr  r4b   , gr  r5b  ;   work:= destination base:= first + k : 1024;
     hs  r1a8  , pm  r4b  ;   write track no read; track:= work;
     gm  r2b   , hv  ra   ;   goto start track;

a11: tln 23    , ps  r1   ; repeat:
     hv  r1    , arn r4b  ;   for work:= k step -1 until 1 do
     sr  rb    , gr  r4b  ;
     hh  ra1          LT  ;   outword(last word);
     arn(rb4)  , hh  ra7  ;   goto read label;

b8:  tparity;              ; alarm messages.
b10: ttapesum;             ;

< i-b, i track, binin
> i=b

     qq  1.39              ; count
[1b: first.39              ; first track of area
 2b: track.39              ; output buffer will go to this track when full
 3b: upper.39              ; first track after area
 4b: work
 5b: destination base      ; first track changed after new destination]
e                          ; end binin code
```

```
i=i+39, d=k-d1                          ;
b k=d42,i=0,a10                         ;
a1=d19-960                              ; a1=group no for image
a=d,<d35,a=1>                           ; a=no of blocks

<d39                                    ; if aux only then
i=d2,hs1                                ; begin
     hv a4                              ;  (if aux reserved then
<d36,tres;                              ;
     qqf a.39                           ;   res,no of blocks,
x<d39                                   ;    else
     tset;                              ;    set,aux kind,no of blocks,
     qqf d35.39                         ;    type in)
     qqf a.39                           ;
[after i follows STOP,SUM and a sum character]
ia  base,binin
s
[STOP,CLEAR]0
><d39                                   ;
   qq  39,                              ; concat pid 0,
   qq  57,                              ;
   qq  52,                              ;
   qqf                                  ;
   tbinin;                              ; binin<
   qqf,                                 ;

a4: hs 1                                ;
    hv a5                               ;
    tmove;                              ; move,b loadplace,binin<
    qq 50,                              ;
    qqf d.23+d1.39+a1.29-a1.33          ;
    tbinin;                             ;
    qqf,                                ;

a5: hs 1                                ;
    hv a6                               ;
    tsetsum;                            ;
    tbinin;                             ;
    qqf,                                ;
d2=i                                    ; end else
a6: hsf 2                               ;
x                                       ;
i=d48                                   ;
   qq d35.2+11.7+d36.5+d.23+d1.39,      ; load to primitive catalog;
   qq,                                  ;
   tbinin;                              ;
d48=i,qqf                               ;
>                                       ;
d1=d+d1   ;
e         ; end image load
e         ; end binin
[after i follows STOP,SUM and a sumcharacter]
ia  binin
s
```

```
[here follows STOPCODE, CLEARCODE]0

b k=d1, i=40d13, a25, b15  ; begin binout
b a10                      ; begin outparam

     can(2c22) , vy 32     ; ENTRY OUTPARAM: if no output unit then select(p);
<d35-2, us(-31)t   -96>    ;   if aux kind = tape then rewind tape;
     sy  58    , sy  64    ;   writecr;
a:   pmn d13   t   1       ; next: param:= param + 1;
     hv  a8          LC    ;   if list[param] = end then goto end list;
     hv  a4    X     LA    ;   if list[param] -< letter then goto letter;
     hh  a6    X     LB    ;   if list[param] -< number then goto number;
a1h: arn a2    , cl  -6    ; text:
     tk  -4    , ga  a3    ;   char:= next char (list[param]);
     ca  15    , hv  a     ;   if char = end word then goto next;
a2:  ca  10          IB    ;   if char = end text then
     [1.39]                ;   begin write comma; goto next
     sy  27    , hv  a     ;   end;
     ca  63    , it  1     ;   if char = 63 then char:= CR;
a3:  sy  0     , hhn a1    ;   writechar(char); goto text;

a4:  sy  14    , ga  r1    ; letter: write underlining;
     sy  0     , hv  a     ;   writechar (list[param]); goto next;

a5:  ck  10    , gr  (a)   ; number: simple print(list[param]);
a6h: sy  59    , hs  d28   ;   for i:= 1 step 1 until 3 do
a7:  bt  3     t   -1      ;   begin list [param]:= list[param] shift 10;
     arn(a)    , hv  a5    ;     writepoint;
     pa  a7    t   3       ;     simple print (list[param]);
     sy  27    , hv  a     ;   end; writecomma; goto next;

a8:  sy  17    , hv  -9    ; end list: writetext(<); return to help;

e                          ; end outparam

d=d19-960                  ;
b:   qq d.29-d.33+d16.39   ; base: initially image base
[1b] qq 1023.9+1023.29     ; mask
[2b] qq 1023.19+1023.39    ; mask
[3b] qq 26.39              ; image length
[4b] qq 510.39-1c54.39     ; max length of bin 0 tape
[5b] qq 1.39               ; 1
b12: qq 40.39              ; length tabel: drum
[1b12]qq d53.39            ; disc
[2b12]qq 512.39            ; carr
[3b12]qq 400.39            ; tape

a=240d13, b8=6a            ; define outbuf base
[a:  length
1a:  label
2a:  previous word
3a:  alike
4a:  blocks
5a:  work, skipped words]
```

```
a18: pm  3b     , gm  4a     ; ENTRY BINOUT: blocks := 26;
<d35-2,us(-31) t -96>        ;   if aux kind = tape then rewind tape;
     can(2c22)  , vy  32     ;   if no of output selected then select(p);
b1:  pp  d13-1  , hs  c52    ; NEXT PARAM: get next param;
     hv  c58    , hv  a1     ;   if number then param alarm;
                             ;   if single then goto domain;
a21h:hv  a2     , gr  b      ;   if end list then goto write end label;
     hv  c57              LT ;   base:= area word; if kind < 0 then kind alarm;
     tl  -7     , ga  b11    ;   store kind;
     tl  -25    , tln 16     ;   blocks:= bits 8-23;
     gr  4a     , hh  b1     ;   goto next param;

a2:  ncn(b6)    , hv  -9     ; WRITE END LABEL: if only bin 0 then return to help;
     sy  66     , it  (b7)   ;   write end mark;
     pa  -1     , arn -1     ;   out label (characters x 1024 + sum);
     ck  10     , hs  a5     ;
     hv  -9                  ;   return to help;
a1:
b11: ps [kind]  , pm  p2     ; DOMAIN: interval:= next word in param list;
     ca  16     , pin 16     ;   if single = 0 then bin 0:= PB:= true;
     ca  37     , pin 0      ;   if single = n then bin 0:= PB:= false;
     qq         V       LC   ;   if next word in param list ≠ number then
     hv  a19            LB   ;   begin
     hv  c58            NZ   ;     if single ≠ 0 ∧ single ≠ n then param alarm;
     pmn 4a     , mln sb12   ;     length:= blocks x length[kind]; first word:= 0;
     hhn a20    X            ;   end else
a19: pp  p1     , ca  50     ;   begin prepare skip of next param;
     hh  a21    X            ;     if single = b then goto set base;
     hv  c58            NZ   ;     if single ≠ 0 ∧ single ≠ n then param alarm;
     xr         , mb  1b     ;
     cl  30     , arn p1     ;     M:= first word:=
     mb  2b     , ml  sb12   ;     bits 0: 9 (interval) x 40 + bits 10:19 (interval);
     cln -20    , ck  -20    ;     R:= last word:=
     gm  5a     , sr  5a     ;     bits 20:29 (interval) x 40 + bits 30:39 (interval);
a20h:ar  5b     , gr  a      ;     length:= last word - first word + 1;
     grn c81    , gm  1a     ;   end; set no label check; label:= first word;
     dln sb12   , gm  5a     ;   skipped words:= first word mod length[kind];
     vk  960    , vk  c63    ;   select and read init medium 1;
     lk  c28    , ar  b      ;   R:= first word : length[kind] + base;
     qq  40c28  , vk  960    ;
     pa  3c28   , hs  2c28   ;   medium:= false; init medium;
     pmn 1c             IPA  ;   PA:= medium = drum;
     gp  b1     , hs  c3     ;   save param address; select first block
     lk  d14    , vkn 960    ;   and read it to text buffer; prepare no fault;
     hv  c70            NZ   ;   for i:= skipped words step -1 until 1 do
     ps  r-2    , arn 5a     ;   begin if fault then fault alarm;
     sr  5b     , gr  5a     ;     call(get next word);
     hv  a7             NT   ;   end;
     pa  b3     t   -1       ;   previous marks:= not existing;
     pp  0      , grn 3a     ;   out:= 0; alike:= 0;
```

```
[Output prelude]
b5:  hv  a17          LPB ;   if bin 0 then goto output bootstrap;
     pa  b5    t   c58     ;   next bin 0 should give param alarm;
b6:  can 1     , hv  a4    ;   if -, bin normal printed then
     bt  100   t   -1      ;   begin
     sy  0     , hv  r-1   ;     outsp(100);
     sy  13    , pt  -1    ;     write aa; sum:= 0;
     pa  b7    t   1       ;     characters:= 1; [prepares final 66-mark]
a4:  sy  65    , pmn 1a    ;   end; write labelmark;
     dl  b12   , gr  1a    ;   outlabel(label : 40 x 1024 + label mod 40);
     cln -10   , ar  1a    ;
     pa  b6    , hs  a5    ;   bin normal printed:= true;

a12: hs  a7                 ; PROCESS NEXT WORD: get next word;
     hv  c70          NZ    ;   if fault then fault alarm;
     qq        X      IRC   ;   RC:= marks of word;
     hv  a6    X      LPB   ;   if bin 0 then goto out bin 0;
     sr  2a    , gi   1a    ;   if current word = previous word
     hh  a8           NZ    ;
     arn 1a    , ps   a9    ;   ∧ indicator = previous marks then
b3:  nc  -1    , hh   a8    ;   begin
     [previous marks]        ;     alike := alike + 1;
     arn 5b    , ac   3a    ;     output buffer
a8h: hv  a10   , hs   a11   ;   end else
     bs  p-62  , hs   a10   ;   begin outrepeat; if out > 62 then
     pm  c-1   , pp   p1    ;     output buffer; out:= out + 1;
     gm  pb8          MRC   ;     outbuf[out]:= previous word:= current word;
     gm  2a    , gi   b3    ;     previous marks := indicator end;
a9=i-1 [Return from outbin 0, output buffer]
     arn a     , sr   5b    ; CHECK LENGTH: length:= length - 1;
     gr  a     , ps   b1-1  ;   prepare return to next param;
     hv  a12          NZ    ;   if length > 0 then goto process next word;
[End of domain]             ;
     hv  a22          NPB   ;   if bin 0 then
     sy  64    , it (b7)    ;   begin writecr to stop primitive input;
     pa  -1    , arn -1     ;     out label(characters x 1024 + sum);
     ck  10    , hv  a5     ;     goto next param
a22: hs  a10                ;   end; output buffer;

a11: arn 3a    , sc  3a     ; OUTREPEAT: R:= alike; alike:= 0;
     hr  s1           LZ    ;   if R = 0 then return;
     sy  64    , ck  -10    ;   write repeat mark; R:= R shift -10;
a5:  ga  r1    , ck  -7     ; OUTLABEL:
     sy  -1    , ga  r1     ;   writechar(bits 3:9(R));
     sy  -1    , ck  -7     ;   writechar(bits 36:2(R));
     ga  r2                 ;   writechar(bits 29:35(R));
     qq (b7)   t   4        ;   characters:= characters + 4;
     sy  -1    , hr  s1     ;   return;
```

```
a10: can p      , hr  s1     ; OUTPUT BUFFER: if out = 0 then return;
     qq (b7)    t   1        ;   characters:= characters + 1;
     sy  p      , gp  b9     ;   writechar(out); start:= 1;
a14h:pp  1      , pmn pb8    ;   for i:= start step 1 until out + start - 1 do
     ar  1.1    D       LA   ;   begin
     ar  1.2    D       LB   ;     R:= outbuf[i];
b7:  qq  1      Xt   6       ;     M:= marks.2;
a13h:cl  32     , ga  r1     ;     characters:= characters + 6;
     sy  -1     , it  -100   ;     for j:= 1 step 1 until 6 do
     bt  0      , hv  r2     ;     writechar(char j of RM);
     cl  -7     , hh  a13    ;
     pa  r-2    , pp  p1     ;
b9:  can -1     t   -1       ; OUT SEGMENT:
     pp  0      , hr  s1     ;   end;
a15: hh  a14                 ;   out:= 0; return;

a7:  hv  c71           NPA   ; GET NEXT WORD: if -, drum medium then
     arn(c-2)  Dt  1         ;   goto get word; word address:= word address + 1;
     nc  40d14 , hh  a23     ;   if word address = top of buffer then
     hs  c16         M       ;   begin select next track;
     lk (c-2)  t   -40       ;     word address:= first in text buf;
a23h:vk  960   , pmn(c-2)    ;     read track end;
     gm  c-1   , hr  s1      ;   M:= current word:= text buf[word address];
                             ;   return with no fault;
```

[The Bootstrap program on track 0 works like this:

```
      original          2 instr. read    2 + 6 instr. read
s-1   tl -6, ca 0
s     ly r4, hs r-1
s+1   gm s3 Mt -1       gm s7 Mt-1       bs (r4)  IO
s+2                     hv s             tl 12 V IK
s+3                                      hv [513-real length|MR
s+4                                      pi 0 t -49
s+5                                      gr [510-real length|MPCt1
s+6                                      hv  s  IKC
s+7   sk dumped. This word must not be destroyed]
```

```
[Bootstrap program, 6 7-bit characters / word.      Bits of: Curr instr; prec.]
b10:   qq   2.6+ 0.13+ 0.20+  0.27+ 0.34+ 0.39  ;hv s                25-39, length
              [bits 29-34, 36-38 is length]
       qq   2.6+ 4.13+ 0.20+  0.27+ 0.34+ 7.39  ; gm s7 t-1 M       25-39, 0-24
       qqf 28.6+ 2.13+ 0.20+127.27+63.34+11.39  ; hv s     IKC     19-39, 0-24
       qq  32.6+ 6.13+56.20+ 64.27+ 0.34+ 0.39  ; gr xx t1 MPC      25-39, 0-18
[4b10] qq   0.6+ 0.13+ 0.20+  0.27+64.34+10.39,  ; pi 0 t -49       31-39, 0-24
              [bits 15-20, 22-24 is 510 - length]
       qq  28.6+ 0.13+ 7.20+103.27+49.34+ 8.39  ; hv xx    MR      19-39, 0-30
[6b10] qq   0.6+32.13+96.20+  0.27+ 0.34+ 0.39  ; tl 12 V  IK      25-39, 0-18
              [bits 22-27, 29-31 is 513 - length]
       qqf 24.6+44.13+ 1.20+ 32.27+64.34+ 2.39  ; bs (r4)  IO      19-39, 0-24
       qq   0.6+ 0.13+32.20+ 64.27+ 0.34+ 0.39  ;                   fill , 0-18
       qq             64.20+  0.27+ 0.34+ 0.39  ;                   fill
```

```
a17: arn a      , sr  4b     ; OUTPUT BOOTSTRAP:
     hv  c58          NT     ;   if length > max length then param alarm;
     sy  14     , sy  17     ;   writetext(‹< ≤ ›);
     pmn a      ; tl  36     ;
     tk  5      ; ac  b10    ;   pack length into boot[0];
     tln 4      , ac  b10    ;
a3:  arn 507    Dt   3       ;
     tk  -30    ; sr  a      ;   pack 510 - length into boot[4];
     tl  -3     ; tk  19     ;
     ac  4b10   ; tln 18     ;
     ac  4b10   ; ud  a3     ;
     tk  -30    ; sr  a      ;   pack 513 - length into boot[6];
     tl  -3     ; tk  12     ;
     ac  6b10   , tln 11     ;
     ac  6b10   , pp  b10-b8 ;   p:= start:= first boot - out buf base;
     pa  b9     t   10       ;   out:= 10;
     pa  b7     t   1        ;   characters:= 1; [prepare final 64-mark]
     pt  -1                  ;   sum:= 0;
     pa  b5     t   c58      ;   next bin 0 should give param alarm;
     ps  a12-1  , hv  a15    ;   call(out segment); goto process next word;

a6:  ar  1.1    D       LRA ; OUT BINO:
     ar  1.2    D       LRB ;
     qq (b7)    t   7        ;   characters:= characters + 7;
     ck  3      X            ;   R:= current word; M:= marks.39;
     cl  34     ; ps  -6     ;
a16: ck  -4     ; ga  r2     ;   for s:= -6 step 1 until 0 do
     bs  s1     ; it  64     ;   writechar((if s = 0 then 64 else 0)
     sy  -1     ; ps  s1     ;   + char s of RM);
     bs  s      ; hv  1a9    ;   goto check length;
     cln -6     , hv  a16    ;
```

```
i=i+39, d=k-d1              ; d = no of tracks
b k=d42, i=0, a10           ;
a1=d19-960                  ; a1 = group no for image
a2=40d13                    ;
a3=a18                      ;
a=d, <d35, a=1>             ; a = no of blocks

< d39                       ; if aux only then
i=d2, hs 1                  ; begin
     hv  a4                 ;
< d36, tres;                ;   (if aux reserved then
     qqf a.39               ;   res, no of blocks,
x < d39                     ;   else
     tset;                  ;   set, aux kind, no of blocks,
     qqf d35.39             ;   typein)
     qqf   a.39             ;

[after i follows STOPCODE, SUMCODE and a sum character]
ia  base, binout
s
[STOP, CLEAR]0
>< d39                      ;
     qq 39,                 ;   concat p i d 0,
     qq 57,                 ;
     qq 52,                 ;
     qqf                    ;
     tbinout;               ;   binout, spec, outparam <
     qqf d.9+a3.19+a2.29    ;
     toutparam;             ;
     qqf,                   ;

a4:  hs 1                   ;
     hv  a5                 ;
     tmove;                 ;   move, b load place, binout <
     qq  50,                ;
     qqf d.23+d1.39+a1.29-a1.33     ;
     tbinout;               ;
     qqf,                   ;

a5:  hs  1                  ;
     hv  a6                 ;
     tsetsum;               ;   setsum, binout <
     tbinout;               ;
     qqf,                   ;
d2=i                        ; end else
a6:  hsf  2                 ;
x                           ;
i=d48                       ;
     qq d35.2+11.7+d36.5+d.23+d1.39, ; load to primitive catalog;
     qq,                    ;
     tbinout;               ;
     qq d.9+a3.19+a2.29     ;
     toutparam;             ;
d48=i, qqf                  ;
>                           ;
d1=d+d1                     ;
e                           ; end image load
e                           ; end of binout

[after i follows STOPCODE, SUMCODE and a sum character]
ia  binout
s
```

[3.10.67. check, list, set sum, compress. page 1]
[STOP, CLEAR]^

```
b k=d1, i=40d13 + 327, a63, b19 ;
                                ;
[Variables and buffer areas:

b-4: sizes or save spec
b-3: save free
b-2: new free
b-1: out track
b=44d13
b-39b: entry buf
40b-79b: out buf
80b-322b: used by sum and move
323b =40d13+327: first word loaded.]

b=i-323
a59:
a: [first word of sum and move loaded here]
```

[sum and move]

[A subroutine used for summing and, if the area is reserved, moving an area given by the two first words of a catalog entry.

The subroutine uses 40 + n times 40 + 3 working locations in front of it self, where 1 ≤ n, as follows:

b1=i-3-n times 40 - 40;
  b1-39b1: used for first track of init medium and for:
    b1: number of blocks.
    1b1-5b1: core part of medium description
    b3=b1+c82-c28
    b3-5b3: buffer part of medium description
    6b3: words to sum
    7b3: to core word, transport word for il 0
    8b3: working location
  b4=40b1, b=b4+b2 where b2 = n times 40. b2 is originally 200
    b4 ff are used for second track of init medium and
    afterwards as core buffer for summing and for moving of tracks.
  b: new free
  1b: to free word, parameter for us unit if free kind is disc
  2b: area sum

call:
  M:= first free. Will be updated if the area is reserved.
  p:= core address of area word of catalog entry.
  LTA: namelist. Areas of kind 2 or 3 will only be summed if LTA.
  LRA: move it. If the area is reserved it will be moved to first free.
  LRB: sum it. If the area has a sumbit and a sumword the
       sum will be generated. Areas of kind 2 or 3 only if LTA however.
  The address part of first word of the subroutine must hold the
       address of select track (M). Originally set to point at
       first word after the routine
  hs first word

Normal return:
  hr s1 with:
  R = sum, only relevant if LRB
  M = updated first free
  LTB - LQB = bits(3, 7, areaword). If special area then NPB
  LRA: has been moved.
  LRB: has been summed.

Error reactions: fault alarm

Note: s and p have been used]

```
b b15, a21                  ; Core block sum and move;

b2=200[n times 40]          ; b2 = core buf size
b=i-3, b4=b-b2, b1=b4-40    ; b4 = core buf
b3=b1+c82-c28               ;

a1:  qq  a21   , hv  a2     ; sum and move:
a:   qq  1.39; 1
[1]  qq b2.39; core buf size
[2]  qq 8b3t1; check il word

a2:  arn p     , ga  b10    ;   R:= area;
b10: pi[kindbits]t  899     ;   indicator 3 to 7:= kind bits;
     pi 0.5+0.9 t 1006 LTB  ;   if special V kind > 3 then
     pi 0.5+0.9 t 1006 LT   ;     reserved := sumit:= f
     gm  b     , gm  1b     ;   new free:= to free word:= M
     tl  -32   , tln 16     ;   if reserved then
     ac  b     V       LPB  ;     new free:= new free + bits (8, 23, R)
     pi  0.8   t  1021      ;   else moveit:= f;
     tln 16    , sr  1b     ;   if to free word = bits (24, 39, R) then
     pi  0.8   t  1021 LZ   ;     move it:= f;
     pi  0.9   t  1022 NQB  ;   if -, sum then sum it:= f;
     bs (b10)  t  1.1-1.9   ;   if bits (0, 2) = 2 V bits (0, 2, R) = 3 then
     pi  0.9   t  1022 NTA  ;     begin if -, namelist then sumit:= f; end;
     arn p1    , gs  b13    ;   save s:= s; area sum:= 0
     pin 0.9   t  1022 NA   ;   if no secondary word then sumit:= f
     gr  2b    , vk  960    ;   else area sum:= secondary word;
     hv  a18          NRC   ;   if -, (sumit V move it) then goto exit;

     vk  c63   , lk  b1     ; init medium:
     vk  1c63               ;   get first medium track;
     pt  2b1   t   3b1      ;   medium:= false;
     qq  40b1  , pa  3b1    ;   core part:= 3b1;
     arn p     , hs  b1     ;   init medium (area);
     hs  c24          NZ    ;   if R ≠ 0 then label alarm;
     qq  e9    , pm  5b1    ;   if buffer medium then goto kind 1 to 3;
     hh  a10          LB    ;
a3:  grn 6b3   , pp  0      ; kind 0: words to sum:= 0; p:= 0
a4:  pm  5b1   , hs (a1)    ; next track: to core (curr track, core buf + p);
     lk  pb4   , arn a      ;   curr track:= curr track + 1;
     ac  5b1   , sc  b1     ;   blocks:= blocks - 1; p:= p + 40;
     pp  p40   , gp  b11    ;   buf size:= p;
     arn b1    , bs p-b2+512;    if p < core buf size ∧ blocks ≠ 0 then
     hv  a4           NZ    ;     goto next track;
     hv  a6           NRA   ;   if move it then
a5h: pp  0     , pp  p40    ;     begin p:= 0;
     pm  1b    , hs (a1)    ; write next: to drum(to free word core buf + p);
     sk  pb4-40, arn a      ;     p:= p + 40; to free word:= to free word + 1;
     ac  1b    , it  p      ;     if p < bufsize then goto write next
b11: bs[buf size], hh a5    ;     end;
a6:  vk  960   , hsn a14    ;   wait drum;
     hv  a3                 ;   sum p words goto kind 0;
```

```
<-d41+1 [if -,buffer media];
a10=c57, a13=c57          ; kind 1 to 3: error exit;
×    [else]               ;
a19: arn 1b    , us (4b3) ; write block (repeat):
                          ;   us(check unit, to free word)
a20: arn 2a    , il (4b3) ; check transp (repeat):
     il  0     , arn 8b3  ;   il (check unit, status to core);
b14: can[check ct]Vt 256 LT;   il (0, status to core); check ct:= check ct + 1;
     pa  b14   , hr  s1   ;   if ok then begin check ct := 0; return) end;
     arn r     , hv  c70  ;   if check count = 4 then faultalarm;
a10h:hr  s-1   , arn 1b   ;   return to (repeat);

     tk  18    , gr  1b   ; kind 1 to 3: to free word:= to free word pos 21 +
     arn 2b3   , ga  1b   ;   bits (0, 9, current block) pos 9 +
     tl  -12   , tln 12   ;   bits (28, 39, current block);

a11: ac  1b    , arn a    ; next block: no of blocks:= no of blocks - 1;
     sc  b1    , arn 2b1  ;   if left in buf = 0 then
     hv  a12         LT   ;     begin
     arn 1b3   , ac  2b3  ;     current block:= current block + block increment
     arn 2b3   , ud  4b3  ;     rep il: il (read block, current block);
     hs  a20   , qq  a11-1;     check transp (rep il); set return next block
     srn 3b3   , gr  2b1  ;     rep us: left in buf:= -block length;
     hs  a19         LRA  ;     if move it then write block (rep us);
     arn 2b3   , tl  -12  ;     to core word:= bits (28, 39, current block)
     tln 12    , gr  7b3  ;       + core buf pos 9;
     pa  7b3   t   b4     ;     end;

a12: pm  3b3   , dln 5b3  ;   words to sum:= block length : area word increment;
     gr  6b3   , ac  2b1  ;   left in buf:= left in buf + words to sum;
     arn 1b3   , ac  1b   ;   to free word:= to free word + block increment;
     hv  a16         NRB  ;   if sum it then
                          ; sum buf:
a13: arn 6b3   , sr  1a   ;     begin
     qqn             NT   ;     R:= if words to sum < core buf size
     ar  1a    , gr  8b3  ;     then words to sum else core buf size;
     ck  20    , gt  b12  ;     p:= R;
     ar  7b3   , il  0    ;     il(0, to core word + R pos 19);
     arn 8b3   , ac  7b3  ;     to core word:= to core word + R;
b12: sc  6b3   , ppn[tosum];     words to sum:= words to sum - R
> [end buffer media]      ;     end;
a14: hv  a16         NRB  ; sum p words:
a15: pp  p-1   , ar  pb4  ;   if sum it then
     ar  2     D     LA   ;     begin
     ar  1     D     LB   ;     for p:= p - 1 while 0 ≤ p do
     bs  p     , hv  a15  ;       area sum:= area sum +
     ac  2b    , srn 6b3  ;         word[core buf + p] + the marks pos 9;
     hv  a13         LT   ;     if 0 < words to sum then goto sum buf
a16: arn b1              ;     end;
     hvn s1          NZ   ;   if no of blocks ≠ 0 then goto return;

a18: arn 2b    , pm  b    ; exit: R:= area sum; M:= new free;
b13: ps[save s], hr  s1   ;   s:= save s; go right back;

a21: [here starts select track]
e    [core block sum and move];
```

```
a1:  dln a9     , ck  -10    ; select track:
     ga  b1     , cln -10    ;   group:= M : 960 + 960; select (group);
b1:  vk [group]t   960      ;   track:= M mod 960; select (track);
     ga  b2                 ;   return;
b2:  vk [track], hr  s1     ;


a2:  pp  p1     , arn r      ; get next word: p:= p + 1; found:= t;
     hs  c15            IZA ;   nextword;
     gm  pb-1           MPC ;   word [p+entry buf-1] MPC:= M;
     hr  s1     X       LZ  ;   if R = 0 then swap return;
a3:  ca  1                  ; caterr: if Raddr = 1 then
     qq  e7     , hs  c24    ;   alarm print (<<catalog>);
     hs  a40    , qq  sa5    ;   alarm(<<name>, 1);

a60: pi1.2+1.9 , hh  a61    ; check: start it (0, f, f, t);
a61: pi1.2     , hs  a4     ; list:  start it (0, f, f, f);
a62: pi1.2+5.9 , hs  a4     ; set sum: start it (1, t, f, t);
a63: pi1.2+2.9 , hs  a4     ; compress: start it (2, f, t, f);
                            ; start it(act, name list, force sum, move it, sum it);
a4:  gs  b3     , hsn a6    ;   comment namelist = LTA. force sum = LQB.
     arn(b3)    Vt  a51 NC  ;   move it = LRA. sum it = LRB; R:= 0;
b3:  arn[act]   t   a50 ITA ;   param; if last param then
     ga  b9     , gt  b4    ;     begin act:= act + 3; namelist:= f end;
     tk  20     , ga  b7    ;   R:= action table[act]; main action:= part 1(R);
b4h: gi  b8     , hv [start];   start:= part 2(R); finis:= part 3(R);
                            ;   main in:= indicator; goto start;
a5:  tname;                 ;

a6:
b5:  pp d13-1[paramaddr]t1  ; param: p:= param addr:= param addr + 1;
a7:  arn p1             LZ  ; test param: if R = 0 then R:= param[paramaddr];
     hv  s1             LC  ;   if last param then goback
     hh  s1             NC  ;   if name param then go right back;
     bs  p      t  d13  NB  ;   if not first param ∨ help number then
a8:  qq  e5     , hs  c24   ;   alarm print (<<param>);
     pp  2      , ca 41 [r] ;   print kind:= if Raddr = r then 2 else
     can(2c22)  , vy  32    ;     if Raddr = n then 1 else if Raddr = a then 3
     ca  37[n]  , ppn 1     ;     else 0;
     ca  49[a]  , ppn 3     ;   if print kind ≠ 0 then R:= 0;
     gp  a55    , hv  a6    ;   if print kind = 2 ∧ no output then select(32);
                            ;   goto param;
a9:  m 960                  ;
```

```
a10: srn 3[+rel cat] D 1    ; start compr: rel cat:= 1;
[1]  gr  79b          MC    ;   word [outbuf + 39] MC:= - rel cat - 3;
     gp  -1           MA    ;   half inhibition;
     vy  -1    t   -1d18    ;   by inhibit;
     pm  c12   , gm  b-1    ;   outtrack:= first catalog track;
a11: hs  c2                 ; start no name: get free;
     hv  a3           NZ    ;   if R ≠ 0 then goto cat err;
     arn d14   , tl d3.7-16;   newfree:=
     tk-d3.7+16, ar  1d14   ;     bottom free + if free on disc then
     tl  -16   , tln 16     ;       bits (24, 27, first free) else 0;
     gr  b-2   , grn b-4    ;   sizes:= 0; LA:= t;

a12: qq (c15)  t  -1   LA   ; get entry: if LA then i word:= iword - 1;

a13: pp  0     , hs  a2     ; start entry: p:= 0; get next word;
     ga  b12   V       NZ   ;   if R ≠ 0 then save bits:= Raddr
     hv  a13           NB   ;   else if not end of catalog then goto start entry
b7:  hv [finis]        LZ   ;   else goto finis;
     hs  a2                ;   get next word;
a14: gp  b13   , hs  a2     ; loop entry: entry sz := p; get next word;
     grn pb-1  V       LA   ;   if it is an area word then
     hv  a14           NZ   ;     word [entry buf + p - 1]:= 0
     grn pb-1          M    ;   else if R = 0 then goto loop entry;
b8:  pi[mainin], ppn b      ;   word[entry buf + p - 1] M := 0;
     arn 1.7   D       LQB  ;   indicator:= main in; p:= entry buf;
     ab  b     , gr  b     ;   if force sum then set sumbit in area word;
     pm  b-2   , hs  a     ;   M:= new free; sum and move;
     pp  b                 ;   p: entry buf; sum ok:= R = 0
b9:  hsr[main action]  IZB  ;   main action;
```

```
a16: pp (b5)   Vt  1   LTA ; next entry: if -, namelist then
     hv  a12           M   ;   begin LA:= t; goto get entry end;
     arn p     , tl  -6    ;   for param addr:= paramaddr + 1 while
     ca  -1    , hv  a16   ;     not last name word do;
                           ;
a17: hsn a7                ; get name entry: test param;
     hv (b7)   , hs  c1    ;   if last param then goto finis else search;
     hv  a3            NZ  ;   if R ≠ 0 then goto cat err;
     gm  b-4           MPC ;   save spec MPC:= M;
     hs  a19           LZ  ;   while R = 0 do back up;
     hs  a19           NA  ;   while NA do back up;
     ps  a12   , hv  a19   ;   set return(get entry); goto back up;


a18: pa  b11   t   a21     ; set write back up: changed:= t;
a19: arn c15   , ca (c7)   ; back up: if i word = place 0 then
a20h:                      ;     goto if changed then write get previous
b11: hv a22[changed],gac15 ;       else get previous track;
     pmn(c15)  Xt  -1      ; backwards: i word:= i word - 1; M:= 0;
     hr  s                 ;   R:= store[iword]; return same;

a21: hs  a23               ; write get previous: sum and write;
a22: arn(c4)   Dt  -1  M   ; get previous track:
     ca  -1    , ac  c5    ;   track:= track - 1;
     pa  c4    t  959  NB  ;   if track = -1 then
     qq (c8)   t  -2       ;     begin group:= group - 1; track:= 959 end;
     hs  c4                ;   rel track:= rel track - 2; select and read;
     arn c9    , hh  a20   ;   i word:= i sum; goto back wards;


a23: hsn c8                ; sum and write: sum track;
     pan b11   Xt  a22     ;   changed:= f; M:= R:= 0;
     sk (c7)   , hrn s1    ;   to drum (place 0); return;


a24:                       ; main set sum:
b12: pm[save bits]XD   IOB ;   if -, sumit then
     pi  1.1   VXt 767 LRB ;     begin sum ok:= f; restore bits of
     ga  b     , hv  a27   ;       area word; goto call print end;
     sc  1b    , hs  a27   ;   sum ok:= t; sum word:= sum word - R;
     hs  a19           NA  ;   call print; while NA do back up;
     arn 1b    , gr (c15)  ;   store new sum word;
     ps  r1    , hv  a18   ;   write back up;
     arn b     , gr (c15)  ;   store new area word;
     ps  a16-1 , hv  a23   ;   set return next entry; goto sum and write;


a26: pi  1.1   t   767 NRB ; main list: if -, sum it then sumok:= t;
a27: gi  b17   , hv  a55   ; call print: result in:= in; goto print entry;
```

```
a32: pp  0      , hh  a34    ; fin compr: p:= 0; goto move entry;

a33: arn b      , ck  -16    ; main compr:
     tk  16     , ar  b-2    ;   if reserved bit then
     gr  b                LPB;     area word:= (bits 0, 23, area word) pos 23
     gm  b-2    , hs  a26    ;      + newfree; newfree:= M; main list;
     arn(b13)   D            ;   sizes:= sizes + entry sz pos 23;
     ck  -14    , ac  b-4    ;   p:= entry size;
a34h:                        ;
b13: pp[entrysz], it b-1     ; move entry: i entry:= entry buf - 1;
a35h:pa  b14    , pp  p-1    ; loop move: p:= p - 1;
b14: arn[i entry] t 1  IPC   ;   i entry:= i entry + 1; i out:= i out + 1;
b15: gr 39b[i out] t 1 MPC   ;   word [i out] with marks:= word [i entry];
     ar  2      D        LA  ;
     ar  1      D        LB  ;   word [out buf + 39]:= word [out buf + 39] -
     sc  79b    , arn b15    ;     word [i entry] - marks pos 9; LB:= f
     nc  78b    , hv  a36    ;   if i out = out buf + 38 then
     pm  b-1    , hs  a1     ;     begin
     sk  40b    , arn a37    ;     to drum (outtrack, outbuf);
     ac  b-1    , ud  a10    ;     out track:= out track + 1;
     vk (b2)    , ud  1a10   ;     i out:= out buf + 39; rel cat := relcat + 1;
     pa  b15    t   39b      ;     word[i out] MC:= -rel cat - 3;
                             ;     LB:= t; end;
a36: bs  p      , hh  a35    ;   if 0 < p then goto loop move;
     bs  p1     , hv  a16    ;   If p = 0 then goto next entry;
     hv (b14)   Dt  -1  NB   ;   If -, LB then
                             ;     begin i entry:= i entry - 1; goto loop move end;
     hs  c2                  ; modify free: get free;
     arn d14    , sr  b-2    ;   R:= bits(24, 39, free area word) -
     tl  -16    , tln 16     ;     new free;
     sc  d14    , ck  16     ;   free area word:= free area word - R +
     ac  d14    , hs  a23    ;     R pos 23; sum and write;
     pp  a38    , hs  a58    ;   print text (<<words in cat, free size>);
     pm  d14    , tl  7      ;   M:= sizes + bits(8, 23, free area word);
     tln 16     , ar  b-4    ;   R:= area print format[0];
     pm  a56    X       IRA  ;
     gm  b-4    , hs  a57    ;   print group;
     sy  64     , hv  a42    ;   writecr; goto exit;

a37: m 1
a38: t
Words in cat, free size      ;
```

```
a40: arn s      , tk  10     ; alarm: mess:= part 2 [word[s]];
     ga  b18    , tk  20     ;   R:= part 4[word [s]] pos 9;
b17: a41=i-1                 ;
a42: pi[result in], vk d19   ; exit: in:= result in;
     pm2 c22    X       IZA  ;   callok:= LZA:= R = 0;
     tk  10     , vk  25d16  ;   if part 2(current output) = 1023 then
     nc  -1     , hv  a45    ;     begin
     lk  d14    , vk  25d16  ;     to core (last image track, work buf);
     hv  a44    X       NZA  ;     if call ok then
     pmn b-4                 ;       begin
     hv  a44    X       NTA  ;       if namelist then
     qq         X       NC   ;       image M:=
     qqn                LT   ;       if spec present then save spec else 0;
     gr30 d14   , arn b      ;       R:= areaword; end;
a44: gr  32d14  , tl  -39    ;     image R:= R;
     gr  34d14  , gi  27d14  ;     image in:= in
     arn d13    , nc  0      ;     if called from program then
     sk  d14    , vk  26d16  ;     to drum (last image track, work buf);
     pt  b18    t    1c24    ;     set alarm print to normal return;
                             ;     end
a45: acn -1             M    ;   full inhibition;
     vy  0      t    -1d18   ;   no by inhibition;
b18:                         ;
b18h:ncn[mess] , hs  c24     ;   if mess ≠ 0 then alarm print (mess);
     hv  -9                  ;   goto call help;


[actiontable]                ; act: main action, start, finis
a50=i-a61                    ; no name list:
qq a26-b9 t a11 + a42.19     ;  0: main list, start no name, exit;
qq        t a8               ;  1: -, param err, - ;
qq a33-b9 t a10 + a32.19     ;  2: main compr, start compr, fincompr;

a51=i-a61                    ; name list:
qq a26-b9 t a17 + a42.19     ;  0: main list, get name entry, exit;
qq a24-b9 t a17 + a42.19     ;  1: main set sum, get name entry, set sum;
qq        t a8               ;  2: -, param err, - ;


a55: [first word of print entry]
```

[print entry

Outputs a catalog entry on the media given in by.
The format of the output is determined by the entry, a preset printkind, and, a boolean sumok
printkind may be 0, 1, 2, or 3 with the following effect:

0. if sum ok then no output else as printkind 1;
1. Only kind and names will be output.
2. The entry will be output in the form of a complete call of set or, if the area is reserved, res.
3. The whole entry will be output in a format corresponding to the parameter format to set, but with two additional possible lined letters:
   x meaning special bit present. If so then the secondary word, if present, will be printed as
   <bit 0 to 7> , <bit 8 to 23> , <bit 24 to 39>
   r meaning reserved bit present.

Any output will be followed by:
if -, sum ok then message(<<sum>)

Call of print an entry:
p:= address of first word of the entry. The entry must be terminated with a zero;
print kind in address part of first word of print an entry
LZB:= sumok.
Indicator bits TB - QB must hold bits 3 - 7 of the area word.

hs first word

ZA - QB are untouched upon exit. M, p, RA and RB have been used.
R = 0 with no marks.
Returns to s1.]

```
b a45, b15                    ; core block print entry;

a1:  is[print kind],pmsa44    ; printentry: R:= action table[print kind];
     ptn b3    VXt a18 LA     ;   area pr:= part 1(R);
     grn a12   X              ;   bit pr:= part 2 (R);
     ga  b1    , gt  b2       ;   spec pr:= part 3(R);
     ck  20    , ga  b4       ;   CRact:= part 4(R); save s:= s
     gt  b14   , gs  b5       ;   if printkind = 2 then resbit:= skip single
     arn p     , cl  -7       ;     else set no < print;
     ga  b15   , ps (b15)     ;     comment no r should be output in
     pm  p     , ca  0        ;     printkind 2;
     pa  b15   t   16         ;   M:= area; kind:= s:= bits(0, 2, M);
     arn sa              IRC  ;   if kind = 0 then kind:= digit zero;
b1:  hv [area pr]             ;   R:= format[s]; point sep:= LRA:= LA
                              ;   terminated:= LRB:= f;
                              ;   goto area pr;
```

```
[area print format[kind]                                    ;
a:[0]qq  8.4+31.9+31.14                                     ;
[1]  qq  8.4+31.9+19.14+27.19                               ;
[2]  qq  8.4+31.9+ 4.14+17.19+21.24+19.29                   ;
[3]  qq  8.4+31.9+19.14+20.19+22.24                         ;
[4]  qq  8.4+17.9+25.14+25.19+25.24,                        ;
[5]  qq  8.4+17.9+25.14+25.19+25.24,                        ;
[6]  qq  8.4+17.9+25.14+25.19+25.24,                        ;
[7]  qq  8.4+17.9+25.14+25.19+25.24,                        ;

[8]  qq  8.4+31.9                ; print format: reserved area, printkind = 2.
[9]  qq 23.4+31.9+31.14          ;   -      -  : secondary word, special.
[10] qq 25.4+25.9+25.14+25.19,   ;   -      -   sumword or spec.

[11] m  10-8   ;
[12] m     10  ;


a2:  t   sum; ;


b:   qq [number];
[1]  qq [format];


a4:  arn 8a              LPB ; area pr 2: if reserved then R:= reserved format;
a5:  ps  a8    , hv  a20     ; area pr 3: set return(start bit pr); goto CRgroup;
a6:  hh  a13             LZB ; area pr 0: if sum ok then goto exit;
a7:  a8=a7-1                 ;
b2h: arn p     , hv [bitpr]; start bit pr: R:= area; goto bit pr;

a9:  qq 23 [x] , hs  a15     ; bit pr 23: start single (x);
     qq 39 [p] , hh  a16     ;   next single (p);
b3h: qq 41 [r] , hha16[resbit];  resbit (r);
     qq 57 [i] , hh  a16     ;   next single (i);
     qq 18 [s] , hh  a16     ;   next single (s);
     pm  p1    , ck  -1      ;   M:= secondary word;
     hv  a11             NA  ;   if is secondary then
     qq        V         LO  ;     begin if -, sumbit then
     qq 52 [d] , hs  a17     ;       print single (d);
     arn 9a    V         LTB ;     if special bit then R:= sec special format else
a10: arn 10a                 ; spec pr 23: R:= four bytes format;
     hs  a21             IRA ;     point sep:= LA; comma group;
                             ;     end;
a11: pp  p1    , arn p       ; name pr: p:= p + 1; R:= word[p];
     hv  a11             LA  ;   if LA then goto name pr
     hv  a12             LZ  ;   if R = 0 then goto end print;
b4:  hv [specpr] X      NT  ;   if spec then goto spec pr;
     hs  a35                 ;   newline;
     ps  a11-1 , hv  a31     ;   set return (name pr); goto printtext;
```

```
a12: sy  64    , sy  17     ; finis: if prinkind = 2 then
     pp  a2                 ;   print end help call;
     hs  a31          NZB   ;   if -, sum ok then print text(«sum»);
a13h:sy  64    , arn r-1    ;   writecr;
b5:  ps[saves] , hrn s1     ; exit: s:= saves; R:= no marks 0; return
a15: ck  3                  ; start single: R:= R shift 3;
a16h:hh  a18          NO    ; next single: if bit(0, R) = 1 then
a17: hs  a39          NRB   ; print single: begin if -, terminated then comma;
     sy  14[_] , sy (s)     ;   print lined letter from cell s followed by space end;
a18h:sy  0     , ck  1      ; skip single: R:= R shift 1;
     ps  s1    , hv  s      ;   s:= s + 1; go back;


a20: hs  a35                ; CR group: new line;
a21: hs  a39          NRB   ; comma group: if -, terminated then comma;
a22: hs  a25          IRB   ; print group: terminated:= f; print next;
     hr  s1           LZ    ;   if R = 0 then return;
     sy  59[.] V      LRA   ;   if point sep then print point
     sy  27[,] , sy  0      ;   else print comma and space;
     hv  a22                ;   goto print group;


a24: hs  a28                ; skip it: skip;
a25: gm  b     , tk  -5     ; print next: number:= M;
     ga  b7    , tk  10     ;   no of bits:= bits(0, 4, R);
     gr  1b    , xrn        ;   format:= bits(5, 39, R) pos 34; R:= M;
     bs (b7)   , hv  a24    ;   M:= 0; if no of bits < 16 then goto skip it;
b7:  cl [no of bits] t 17   ;   no of bits:= no of bits - 15;
                            ;   M:= bits(0, no of bits, R);
a26: gs  b10   , mln 11a    ; print M:
a27: mln 12a   , it  128    ;   comment prints M with layout «d»
b8:  ca 0[count], pa  b10   ;   maximum 8 digits, no sign;
     arn 16    DV     LZ    ;
     ck  -10   , pa  b10    ;
b9:  ga [digit]D            ;
b10: can[yes]  , sy (b9)    ;
     ncn(b8)   , hv  a27    ;

a28: arn b     , pm  1b     ; skip: M:= bits(no of bits, 39, number);
     tk (b7)   X            ;   R:= format;
     hr  s1                 ;   return;


a31:
a30h:pm  p     , cln -6     ; print text:
     ck  -4                 ;
     pi  0.9   Xt  1022     ;   M:= word[p];
     hh  a32   X      NZ    ;   print string in M
     xr        , ca  7      ;
     pp  p1    , hv  a31    ;   if continued then
a32h:hr  s1    , ga  b12    ;     begin p:= p + 1; goto print text end;
     ca  10    , hr  s1     ;   terminated:= f;
     ca  63    , it  1      ;   return;
b12: sy [char] , hhn a30    ;
```

```
a35: sy 64 [CR], bs (b15)  ; newline: print CR; if kind not printed then
b14h:sy  58    , hv [CRact];   begin print LC; goto CR act end;
     sy  0     , sy  0     ;   four spaces; terminated:= t; return;
a36h:hh  a40   , sy  18 [s]; set: print text(<<set>);
     sy  53 [e], sy  19 [t];   print comma;
     hs  a39               ;
a37:a38h:                  ;
b15: sy [kind] , pa  b15   ; kind pr: outchar(kind); kind:= 0;
a39:                       ;
a40h:sy  27[,] , sy  0     ; print comma: print comma and two spaces;
     pi  1.9   t   1022    ;   terminated:= t;
     sy  0     , hr  s1    ;   return;
a41: hh  a36          NPB  ; set or res: if -, reserved then goto print set;
     sy  41 [r], sy  53 [e];   printtext(<<res>);
     sy  18 [s], hh  a38   ;   goto print comma;

[Action table[print kind]          ; area pr  bit pr   spec pr  CRact
a44: qq  a6.9+a11.19+a11.29+a37.39 ; area pr0,namepr   namepr   kind pr
[1]  qq a11.9+a11.19+a11.29+a37.39 ; namepr  ,namepr   namepr   kind pr
[2]  qq  a4.9+ a9.19+a10.29+a41.39,; areapr2 ,bitpr23  specpr23 set or res
[3]  qq  a5.9+ a9.19+a10.29+a37.39 ; areapr3 ,bitpr23  specpr23 kind pr


[Define names for use in outer block]
a56=a, a57=a22, a58=a31    ; printformat 0, print group, print text.

e [print an entry]         ;
```

```
i=39i, d=k-d1                ;
b k=d42, i=0, a10            ;
a1=d19-960                   ; a1 = group no for image.
a2=a59, a3=a60, a7=a61       ; define entry points.
a8=a62, a9=a63               ;
a=d                          ; a = no of blocks.
<d35, a=1>                   ;

<d39                         ; if aux only then
i=d2                         ;
     hs1                     ; begin
     hv  a4                  ;   (if aux reserved then
<d36                         ;
     tres;                   ;   res, no of blocks
     qqf a.39                ;   else
×                            ;
<d39                         ;
     tset;                   ;   set, aux kind, no of blocks,
     qqf d35.39              ;   typein)
     qqf a.39                ;
[After i follow STOP, SUM, a, and a SPACE]
iaa base check, compress, list, setsum
s
[STOP, CLEAR]0
>                            ;
<d39                         ;
     qq  39,                 ; concat pid 0,
     qq  57,                 ;
     qq  52,                 ;
     qqf                     ;
     tcheck;                 ; check,
     qqf d.9+a3.19+a2.29     ; spec,
     tlist;                  ; list,
     qqf d.9+a7.19+a2.29     ; spec,
     tsetsum;                ; setsum,
     qqf d.9+a8.19+a2.29     ; spec,
     tcompress;              ; compress,
     qqf d.9+a9.19+a2.29     ; spec<
     qqf,                    ;
a4:  hs  1                   ;
     hv  a5                  ;
     tmove;                  ; move, b loadplace, check <
     qq  50,                 ;
     qqf d.23+d1.39+a1.29-a1.33 ;
     tcheck;                 ;
     qqf,                    ;
a5:  hs  1                   ;
     hv  a6                  ;
     tsetsum;                ;
     tcheck;                 ;
     qqf,                    ;
d2=i                         ;
a6:  hsf 2                   ; end
```

```
×                                   ; else
i=d48                               ; load to primitive catalog
     qq d35.2+11.7+d36.5+d.23+d1.39, ;
     qq,                            ;
     tcheck;                        ;
     qq d.9+a3.19+a2.29             ;
     tlist;                         ;
     qq d.9+a7.19+a2.29             ;
     tsetsum;                       ;
     qq d.9+a8.19+a2.29             ;
     tcompress;                     ;
     qq d.9+a9.19+a2.29             ;
d48=i                               ;
     qqf                            ;
>                                   ;

d1=d+d1                             ;
e  [image load]                     ;
e  [check ff]                       ;
[After i follow STOP, SUM, a, and a SPACE]
iaa check, compress, list, setsum
s
```

[STOP, CLEAR]

```
b k=d1, i=40d13, a67, b17 ;
[variables]
b:   qq                    ; length
[1]  qq d3.2+1.5,          ; area, initially reserved in free, for use by res.
[2]  qq                    ; sum word
[3]  qq                    ; start point test value

[kind table]

[-1] pi   4     , hv  a4   ; kind 4 to 7
a:   pi   2     , hv  a12  ; kind 0
[1]  pi   0     , hv  a40  ; kind 1
[2]  pi   0     , hv  a41  ; kind 2
[3]  pi   0     , hv  a42  ; kind 3

[constants]
[4]  qq          2d38.39   ; top core + 1 for programs
[5]  qq            40.39   ; track length
[6]  qq -1.7+1.23-1.39     ; mask for zeroing booked
[7]  qq      7.27          ; unit for caroussel
```

```
a10: pp  d13    , arn p1     ; set:
     ck  -10    , ga  b4     ;   p:= addr of first param - 1;
     ck  7      , gr  1b     ;   R:= param[p+1];
     pa  b4     t   -1  LO   ;   kind:= if R < 4 then R else -1;
     hsn a2                  ;   area:= R pos 2; R:= 0;
     qqf            7.39     ;   length:= test add next(0, 7, true);
     gr  b      , hsn a2     ;   test add next(dum, dum, false);
b4:  hv [kind] t   a         ;   go to table[kind];


a2:  hv  a6             LA   ; procedure test add next(lower, upper, test);
     hv  a6             NB   ;   comment test = word[s1] is f marked,
     ac  1b     , pp  p1     ;    lower = signed bits(0, 19, word[s1]),
     arn s1     , tl  -20    ;    upper = bits(20, 39, word[s1]);
     ar  p      V       LB   ;   begin if paramkind(R) ≠ help number
     arn p1     , hv  s1     ;     then go to param err;
     hv  a5                  ;   area:= area + R; p:= p + 1;
     tln 20     , sr  p      ;   if -,test then begin R:= param[p+1]; return end;
     hv  a5                  ;   if param[p] < lower ∨ param[p] > upper
     arn p1     , hv  s2     ;     then go to value err;
                             ;   R:= param[p+1]; return;
a11: arn r1     ; hs  c2     ; res: R:= not zero; get free;
     pm  1d14   ; tln 7      ;   p:= addr of first param -1
     tln 16     , pp  d13    ;   if param kind[p+1] ≠ help number then
     pm  p1                  ;     length:= booked
     gr  b      V       NB   ;   else begin length:= R:= param[p+1];
     gm  b      , hsn a2     ;     R:= 0; test add next(dum, dum, false);
     pi1.3+1.8-d3.8, srn b   ;   reserved:= t; kind is 0:= free kind = 0;
     hv  a7                  ;   if length < 0 ∨ length > free size
     pm  d14    ; tln 7      ;     then go to length err;
     tln 16     , sr  b      ;   area:= area + first free +
     hv  a7                  ;     length pos 23;
     arn b      ; tl  16     ;   go to get bits and sum;
     ac  1b     , hv  a14    ;

a4:  arn b                   ; kind 4 to 7:
     ac  1b     ; tl  -2     ;   area:= area + length;
     ca  0      , hv  a14    ;   if bits(0, 7, length) = 0 then
                             ;     go to get bits and sum;
a5:  hs  a65    ; qq  sa8    ; value err: alarm(‹<value›, 1);
a6:  qq  e5     ; hs  c24    ; param err: alarm print(‹<param›);
a7:  hs  a65    , qq (a9)    ; length err: alarm(‹<length›, 4);

a8:  tvalue;                 ; texts.
a9:  tlength;                ;
```

```
a12: gr  3b    , hs  a2     ; kind 0: start point test value:= R;
     qqf     d22.31+d22.33  ;   test add next(0, max start kind 0 + 1, true);

a13: arn b     , ck  16     ; test size:
     ac  1b    , srn b      ;   area:= area + length pos 23;
     hv  a7          NT     ;   if length + start point test value ≥
     sr  3b    , ar  s1     ;     last used value of upper then
     hv  a7          LT     ;     go to length err;

a14: arn p1    , ps  2      ; get bits and sum: R:= param[p+1];
     hv  a6          LC     ;   if param kind(R) = lined then
     hh  a16         NA     ;     begin p:= p + 1;
     pp  p1    , ca  18 [s] ;     if R = s then
     pt  b11   , hv  a15    ;       begin sum bit:= 1; go to get sum end;
     ca  39 [p], pan b10    ;     if R = p then programbit:= 1
     ca  57 [i], ptn b10    ;     else if R = i then inhibit bit:= 1
     hv  a14         LZ     ;     else if R = d then get sum:
     nc  52 [d], hv  a6     ;       begin a words:= 2;
a15: pm  p1    , gm  2b     ;       sum word:= param[p+1]; R:= 0;
     gs  b14   , hsn a2     ;       test add next(dum, dum, false) end
                            ;     else go to param err; go to get bits and sum
a16h:hv  a14   , gp  b15    ;     end;
b10: ns  1.4   , is  s-1.6  ;   j:= p; R:= programbit pos 4 +
b11: arns1.4+1.6+1.7Dt-1.7  ;     inhibit bit pos 6 + sum bit pos 7;
     ac  1b    , nc  0      ;   area:= area + R; if R ≠ 0 ∧
     hv  a6          LQB    ;     kind is 4 to 7 then go to param err;

a17: hs  c1                 ; test name: search;
     hv  a36         NT     ;   if R ≥ 0 then go to cat err res set;
a18: arn p1    , tl  -6     ; skip name: R:= param[p+1];
     ca  -1    V     NC     ;   if param kind(R) ≠ name then go to param err;
     hv  a6                 ;   if bits(0, 3, R) = 15 then
     pp  p1    , hv  a18    ;     begin p:= p + 1; go to skip name end;
     nc  -6    , hv  a5     ;   if bits(0, 3, R) ≠ 10 then go to value err;
     acn p1          MB     ;   set f mark on last name word;
     pp  p1    , pmn p1     ;   p:= p + 1; M:= 0; R:= param[p+1];
     hh  a22   X     NC     ;   if param kind(R) = name ∨
     hh  a22   X     LC     ;     param kind(r) = finis then
     hv  a6    X     LA     ;       go to test area program;
                            ;   if param kind(R) = lined then go to param err;
```

```
      gm  p1    X        M    ; test spec:
      hv  a5              LZ   ;   clear marks on spec word;
      hv  a5              LT   ;   if r < 0 then go to value err;
      cl  10    , ck  20       ;   M:= tracks part(R); p:= p + 1;
      tk  30    , pp  p1       ;   Raddr:= first core part(R); go to test progr;

a22h: hv  a23   , pm  b        ; test area program:
      arn 40d13 D              ;   if kind is 0 then
      hh  a24             NRA  ;     begin M:= length; Raddr:= fixed first core;
a23:  ck  10    , sr  4a       ; test program: R:= Raddr - top program core;
a24:  ml  5a    , pm  p1       ;   RM:= R + M x 40;
      ca  (b10) , hv  a5       ;   if program ∧ RM ≥ 0 then go to value err;
                               ;     end;
      hv  a17             NA   ;   if param kind[p+1] = name then go to test name;
      hv  a6              NB   ;   if param kind[p+1] ≠ finis V
      bs  p-36d13, hv  a6      ;     too many params then go to param err;

a25:                           ; store: comment LRC = fill up;
b14:  bt  1 [awords] t -1      ;   awords:= awords - 1; if awords ≥ 0 then
      arn b       Vt  1   IRC  ;     begin b:= b + 1; R:= set RC(word[b]) end
b15:  arn[j]      t   1   IRC  ;   else begin j:= j + 1; R:= set RC(word[j]) end;
      qq  (b15)   Vt  -1  LRC  ;   if fill up then
      gr  (c15)   V       MRC  ;     begin j:= j + 1; store[iword] MB:= 0 end
      grn(c15)            MB   ;   else store[iword] MRC:= R;
      arn(c15)    Dt  1   IZA  ;   iword:= iword + 1; found:= t;
      nc  (c9)    , hv  a25    ;   if iword ≠ i sum then go to store;
a26:  pa  c6      , hs  c8     ; full track: mode:= 0; sum track;
      hv  a27             LRC  ;   if -, fill up then
      arn 1c      , tl  -2     ;     begin
      ca  (c8)    , hv  a34    ;     if rel track = max cat tracks then
      sk  (c7)    , ps  a25-1  ;       full catalog;
      qq  (c8)    t   1        ;     to drum(place 0); rel track:= rel track + 1;
      hv  (c4)    Dt  1        ;     track:= track + 1; select track; go to store
                               ;     end;
a27:  sk  (c7)    , vk  (c4)   ;   to drum(place 0); wait drum;

a28:  hhn a66             NTB  ; set new free:
      arn r1      , hs  c2     ;   if -, reserved then go to exit;
      arn b       , ac  (c7)   ;   Raddr:= not zero;
      tk  16      , sc  (c7)   ;   free word:= free word + length - length pos 23;
      arn 1d14            ITB  ;   reserved:= f;
      mb  6a      , gr  1d14   ;   booked:= 0;
      hv  a26                  ;   go to full track;

a30:  tname;                   ;
a31:  tno clear;               ;
a32:  ca  -1                   ; cat err clear: if Raddr = -1 then
      hs  a65     , qq  sa30   ;   alarm(‹<name›, 1) else
a33:  qq  e7      , hs  c24    ;   alarm print(‹<catalog›);
a34:  hs  a65     , qq  pe2    ; full catalog: alarm(‹<full›, 3);
a35:  hs  a65     , qq a31+2.29; clear err: alarm(‹<no clear›, 2);
a36:  nc  0       , hv  a33    ; cat err res set: if Raddr = 0 then
      hs  a65     , qq a30+2.29;   alarm(‹<name›, 2) else alarm print(‹<catalog›);
```

```
<d41 [if buffer media then begin]

a40: ck  12     , hs  a2      ; kind 1: R:= disc unit pos 27;
     qqf -8.19      +15.39 ;   test add next(8, 15, true);
     gr  3b     , hs  a2    ;   start point test value:= R:= first disc block;
     qqf       d52.29+d4.39 ;  test add next(0, max disc blocks, true);
     hv  a13                ;  go to test size;

a41: ck  10     , hs  a2      ; kind 2: R:= grouped pos 29;
     qqf -1.19      + 3.39 ;   test add next(1, 3, true);
     ck  4      , gr  3b    ;   start point test value:= R:= reel pos 35;
     hs  a2                 ;  test add next(0, 63, true);
     qqf              63.39 ;   start point test value:=
     ac  3b     , arn 7a    ;     start point test value + caroussel block;
     ar  p1     , hs  a2    ;   R:= 7 pos 27 + caroussel block;
     qqf              15.39 ;   test add next(0, 15, true);
     hs  a13                ;   set last used upper to 1024;
     qq             1024.39 ;   go to test size;

a42: ck  12     , hs  a2      ; kind 3: R:= tape unit pos 27;
     qqf -1.19      + 6.39 ;   test add next(1, 6, true);
     ck  7      , hs  a2    ;   R:= file pos 32;
     qqf              31.39 ;   test add next(0, 31, true);
     gr  3b     , hs  a2    ;   start poin test value:= R:= first tape block;
     qqf             127.39 ;   test add next(0, 127, true);
     hs  a13                ;   set last used upper to 8000;
     qq             8000.39 ;   go to test size;

x    [end else]            ;

a40=c57, a41=c57, a42=c57  ; kind 1: kind 2: kind 3: alarm print(<kind>);

>                          ;
```

```
a50: arn 6a    , gr  2c    ; clear: area word:= something with special bit;
     pp  d13   , hs  c1    ;   p:= address of first param - 1; search;
     hv  a32         NZ    ;   if R ≠ 0 then go to catalog err;
     arn 2c    , ck  3     ;   if special bit is set in area word then
     hv  a35         LO    ;     clear err;
     pi  327   t   48      ;   finis:= LRC:= reserved:= t; R:= 0;
     tk  3           ITB   ;   changed:=f; if -, reserved bit then
     hvn a51         LTB   ;     begin reserved:= f; if kind = 0 then
     tk  35                ;
     hvn a51         NZ    ;       for R:=R - store[iname:=iname + 1]
[-1] arn 2c    , ud  c19   ;         while NC do
     hv  r-1         NC    ;     if R ≠ 0 then clear err;
     hv  a35         NZ    ;     end;
                           ; search end entry:
a51: pm (c15)  , ps  a51-1 ;   M:= store[iword]; set return(search end entry);
     hv  a32         NZ    ;   if R ≠ 0 then go to cat err clear;
     qq        X     NA    ;   if -, areaword then swop;
     hv  c15         NZ    ;   if R ≠ 0 then go to get word;
     qq              IQB   ;   LQB:= LB;

     ps  r1    , hv  a57   ; clear entry: back up;
     hs  a56         NPA   ;   while NPA do clear back up;
     hs  a56         LPA   ;   while LPA do clear back up;
     hs  a56         LZ    ;   while R = 0 do clear back up;
     arn 2c    , ps  a28-1 ; test if free can be changed:
     hv  a52         NTB   ;   set return(exit);
     ck  -16   , ar  2c    ;   if reserved ∧ first track(free) =
     sr  c     , cl  24    ;     first track(area) + size(area) then

     hv  a52         NZ    ; get new free descr:
a60: hs  a57         NPA   ;     begin skip to area: while NPA do back up;
     hs  a55         LPA   ;     while LPA do set area;
     arn 2c    , tk  4     ;     if bit(3, area word) = 1 then
     tk  2     V     NT    ;       R:= free word 1
     arn 1c    , hv  a61   ;     else if bit(5, area word) = 0 then
     hv  a60         NT    ;       go to skip to area
     ck  -22   , ar  2c    ;     else R:= area word + bits(8, 23, area word;
a61: sc  c     , arn c     ;     length:= bits
     tl d3.7-16,tln-d3.7+16;       (24 + free kind x 4, 39, R) - first free;
     sc  b     , ps  a28-1 ;     set return(set new free)
                           ;     end;

a52: hr  s1          LZB   ; maybe write: if -, changed then return;
a53: hsn c8          IZB   ; sum and write: changed:= f;
     sk (c7)   , hr  s1    ;   sum track; to drum(place 0); return;
```

```
a54: hs  a53          NZB ; get previous track: if changed then sum and write;
     arn(c4)  Dt  -1  M   ;   track:= track - 1;
     ca  -1   , ac  c5    ;    if track = -1 then
     pa  c4   t  959  NB  ;     begin group:= group - 1; track:= 959 end;
     qq (c8)  t   -2      ;   rel track:= rel track - 1;
     hs  c4               ;   select track and read it;
     arn c9   , hh  a58   ;   i word:= isuum; go to backwards;

a55: gr  2c   , hv  a57   ; set area back up: area word:= R; go to backup;

a56: grn(c15)         MQB ; clear back up: store[iword] MQB:= R:= 0;
     pi  0.1  t  767      ;   changed:= t;

a57: arn c15  ; ca (c7)   ; back up: if iword = place 0 then
a58h:hv  a54  , ga  c15   ;       go to get previous track;
     pmn(c15) Xt  -1  IPC ; backwards: iword:= iword -1; R:= store[iword];
     hr  s  [ not s1 ]    ;   return same;

a65: arn s    ; tk  10    ; alarm: set text addr;
     ga  b17  , tk  20    ;   Raddr:= exit value;

a66: pm  2c22 X            ; exit:
     ck  10   ; vk  d19    ;   if mask current output = -1 then
     nc  -1   , hv  a67    ;     begin
     vk  25d16; lk  d14    ;     store exit value in R cells
     vk  25d16; gm 32d14   ;     in image;
     grn 35d14; arn d13    ;     if called from programthen image to drum;
     nc  0    , sk  d14    ;     set alarm print to normal return
     pt  b17  t 1c27       ;     end;
b17: b17h:                 ;
a67: ncn 0    , hs  c24    ;   if text addr ≠ 0 then alarm print(text addr);
     hv  -9               ;   go to CALL HELP;
```

```
i=39i, d=k-d1          ; d = no of tracks
b k=d42, i=0, a3       ;
a1=d19-960             ; a1 = image group
a=d, <d35, a=1>        ; a = no of blocks

<d39                   ; if aux only then
i=d2                   ;  begin
     hs  1             ;
     hv  a2            ;
     tmove;            ;  move, b load place, set <
     qq  50            ;
     qqf d.23+d1.39+a1.29-a1.33        ;
     tset;             ;
     qqf               ;

a2:  hs  1             ;
     hv  a3            ;
     tsetsum;          ;  setsum, set <
     tset;             ;
     qqf               ;
d2=i                   ;
a3:  hsf 2             ;  end
x                      ; else
i=d48                  ;  load to primitive catalog;
     qq  d35.2+11.7+d36.5+d.23+d1.39,  ;
     qq                                ;
     tset;                             ;
     qq  d.9+a10.19+40d13.29           ;
     tres;                             ;
     qq  d.9+a11.19+40d13.29           ;
     tclear;                           ;
     qq  d.9+a50.19+40d13.29           ;
d48=i                  ;
     qqf               ;
>                      ;

d1=d+d1                ;
e [load image]         ;
e [set,res,clear.]     ;

[after i follows STOP, SUM and a sum character]
ia set,res,clear
s
```