Order No.: 494

Class: 0.2.2

Type: Book

Author: T. Asmussen, J. Jensen,
S. Lauesen, P. Lindgreen,
P. Mondrup, P. Naur, and
J. Zachariassen

Ed.: December 1967 (E)

THE COMPLETE ANNOTATED

PROGRAMS OF GIER ALGOL 4

Volume I

Tape identification	Contents	Page identification	Number of pages
T1,L1	Block head	T1, L1 Gier Algol 4	1
	– –	check load parameters	1
	General pass adm.	GP, page 3–13	11
	Running system	RS, – 1–18	18
T2	Tape test	T2 Gier Algol 4	1
	Pass 1	pass 1, page 1–35 11, 14, 22 (intended for comments) missing	32
	Pass 2	pass 2, page 1–6	6
	Pass 3, phase 3.1	pass 3 (phase 1) (std proc) page 1–3	3
	Sum check	tape number := 3	1
T3	Tape test	T3 Gier Algol 4	1
	Pass 3, phase 3.2	Pass 3, page 1–22	22
	Pass 4	pass 4, page 1–12	12
	Pass 5, phase 5.1	pass 5, page 1–11	11
	Pass 5, phase 5.2	pass 5 std. proc descr., page 1–3	3
	Sum check	tape number := 4	1

```
;slip<

[31.8.67                                                T1,L1 Gier Algol 4
                                                        1]


b c82, e69 ; Outermost block


[Definition of loading parameters
           Possible values    , meaning]

e14=90   ; Depending on available drum, first track for reading by slip.
e4=15    ;     10 ≤ e4  ≤ 17  , first core used by translator.
e20=1022 ; 1005e4 ≤ e20 ≤ 1022, last   –    –   –      –
e38=15   ;     10 ≤ e38 ≤ 200 , first core used by translated program.
e37=1022 ;    800 ≤ e37 ≤ 1022, last   –    –   –      –         –
         ;                       Note below: e37 = 1e37.
e27=0    ;   0, arrays in core, e18 and e40 must also be 0.
         ;   1,   –    –  buffer.
e44=0    ;   0, translator medium is drum.
         ;   1,     –        –    – a buffer medium.
e18=0    ;   0, backing store is drum.
         ; 400,    –      –   –  disc with block length 400.
         ; 640,    –      –   –   –    –     –     –    640.
e40=0    ; < 0, no tape units , SAMBA record   handling procedures are included.
         ; = 0, –   –     –   ,   –     –          –        –       – not  –   .
         ; > 0, e40 –     –   ,   –     – and tape –        –       –      –   .
e69=4    ; 4 ≤ e69 ≤ 40 , SAMBA record segment length.

i redefine wanted loading parameters
s

[here follows stop code and clear code]

<e40+1,<–e40+1
×
e27=1
>
```

```
[check load parameters]
<e4−17,e4=0>
<−e4+10,e4=0>
<e20−1022,e4=0>
<−e20+1005e4,e4=0>
<e38−200,e4=0>
<−e38+10,e4=0>
<e37−1022,e4=0>
<−e37+800,e4=0>
<e27−1,e4=0>
<−e27,e4=0>
<e44−1,e4=0>
<−e44,e4=0>
<e40−15,e4=0>
<−e69+4,e4=0>
<e69−40,e4=0>
<−e27+1,<−e18,e4=0>
<−e27+1,<e18,e4=0>
e3=1
<e18+1,<−e18+1,e3=0>
<e18−399,<−e18+401,e3=0>
<e18−639,<−e18+641,e3=0>
<e3,e4=0>
<−e4+1
i wrong redefinition, restart the slip reading
e
s

>


[define some general names]
e13=21e4    ; start translator buffer areas
e28=163e13  ; start adress of GP fixed part
e3=40e28    ; byte input−output code
e16=40e3    ; normal first word of a pass
e37=1e37    ;
e41=1       ; GP seg size
<e44        ; if translator medium is buffer medium then
e41=40      ;   GP seg size := 40;
>           ;
e26=41      ; trackplace length for RS

[version and tape number definitions]
d e48=0     ; translator is in := false
d e51=1     ; version number
d e50=e51   ; max version number := version number
            ; set version number for :
d e61=e51   ; T1,L1
d e62=2     ; T2
d e63=3     ; T3
d e64=4     ; T4
d e65=5     ; T5
d e66=6     ; T6,L2
d e67=7     ; T7,L3
d e68=8
```

```
b k=e14, i=e28, a25,b35,c7 ; GP drumblock. GP fixed part, track e14.

e28: qq [loaded again at end GP] ; start compiler: goto init GP;
[164e13]                        ;
     qq [loaded again at end GP] ;
a16:
a:   1₁₀−3                  ; constants:
[1a] 1                      ;
[2a] 10                     ;
[3a] 960                    ;

b1:
e6:  bt474[print count]t−55; test print: print count:= print count + 1;
     hs  a4                 ;   if print count ≥ 10 then newline;
e7:  gr  e13   , gm  41e13 ; print: save R:= R; save M:= M;
     ck  0     , tl  −69    ;   RM:= part 1(R);
e8:  mln a     , pt  b2     ; print 1: RM:= RM × ₁₀−3; zero:= 0;
a1:  ck  −10   , ga  b3     ; next: char:= Raddr:= part 4(R);
     bs (b4)           LZ   ;   if digits = 3 ∨ R ≠ 0 then zero:= 16;
     pt  b2    t   16       ;
b2:  pa  b3    t [zero] LZ  ;   if R = 0 then char:= zero;
b3:  sy [char] , ncn sa10   ;   outchar(char); if −, call from byte out then
a2:  pa  b1    t   474      ;     print count:= 10;
b4:  ncn 0[digits]t−256     ;   digits:= (digits + 1) mod 4;
     mln 2a    , hv  a1     ;   if digits ≠ 0 then begin R:= RM × 10 goto next end;
     arn e13   , pm  41e13  ;   R:= save R; M:= save M; marks:= marks A;
a3:                         ; space exit:
b5:  sy  59 [point], pa b5  ;   outchar (point); point:= 0;
     sy  0     , hr  s1     ;   outchar(0); return;
a4:                         ; new line: outchar(64) print count:= 0
e9:  sy  64    , pa  b1     ;   out char (passno); passno:= 0;
b6:  sy 1[passno], pa b6    ;   goto space exit;
e29h:hv  a3    , it  a17    ; next segm: GPseg:= next seg 1; goto set GP pl;
e5:  pa  b11   , it  123e13 ; mess: GPseg:= mess1; set GP pl: GPpl:= if outaddr ≥
     bs (b15)  , it  83e13  ;     outbuf 2 − (if backw pass then 1 else 0) then
b10:pa124e13[GPpl]DXt124e13;     outbuf 1 else outbuf 2; swop;
e42: hs  b11                ; next GP seg: get GP seg; goto GP pl;
e30:                        ;
b11: arn a19[GPseg] Dt e41  ; get GP seg: GP seg:= GP seg + GP seg size;
< e41−1                     ;   if GP in buf then
     ck  10    , ar  11e4   ;
     ar (b10)  D            ;    buf to core (40) words from: (GP base + GP seg)
e43: qq 0[segno] , vk (b9)  ;      to: (GP pl);
     il  0     , hv (s−1)   ;
×                           ;   else
     hs  a5                 ;     begin
     lk (b10)  , vk (b9)    ;     track (GP base + GP seg); lk (GP pl);
e43: qq 0[segno] , hv(s−1)  ;     end;
a5:  ck  10    , ar  11e4   ;   wait drum; go indirect (s−1);
>                           ;
e11: gm  41e13 X            ; track: save M:= M;
     dln 3a    , ck  −10    ;
     ga  b8    , cl  −10    ;   group:= 960 + R : 960;
     ga  b9    , pm  41e13  ;   the track:= R mod 960; M:= save M;
b8:  vk [group]t   960      ;   select group;
e17:
b9:  vk[the track], hr s1   ;   select the track; return;
e3: d e22=1, e24=0, e10=i   ;   Define GP loading parameters
```

```
b c7                        ; mess 1, track 1e14

c1:  hrn r1     X           ; message: restore s from original call; M:= 0;
     arn s1    , it 41e13   ; R:= paramword in s1;
     bs (b17)  , it  −41    ;   text:= GPpl:= if inaddr ≥ inbuf 2 then inbuf 1
     it (rb21) , pa  b10    ;     else inbuf 2;
     gt  rb21  , cl  −2     ;   rel:= part 2(R);
     tk  38    , ck  −18    ;   by no:= bits(36, 37, R); M pos 2:= bits(38, 39, R);
     ac  r, ud13e4[byaddr]  ;   do(select by no);
     sy  58    , arn e4     ;   write LC;  R:= M;  M:= linecount;
     hs  e9    X            ;   new line;
     nc  0     , gs  rb23   ;   if Raddr ≠ 0 then return addr:= s;
     ca  3.2   , hv  rc2    ;   if Raddr ≠ 3.2 then
     arn 16e4  , ab  rb22   ;     begin  error occurred:= true;
     ga  16e4  , sy 29[RED];     write char (RED)  end;
c2:  sy  35[l] , sy  57[i]  ;   writetext (≠<line≠);
     sy  37[n] , sy  53[e]  ;

c3h: qq  rc5   , hsn e8     ;   print 1 (CR count)EP seg := 0;
                            ;     comment to get this GPseg again;
c4h: pa  b11   , hs  e30    ; next text segm: get GP seg;
c5:                         ;
b21: pmn42e13[text]Xt [rel]; next text word:
     pt  rb21  t   −40      ;   text:= text + rel; rel:= −40
     is (b10)  , it  s39    ;   if text > GPpl + 39 then goto next text segm;
     bs (rb21) , hh  rc4    ;
                            ;
c6h: cl  34    , ck  −4     ;   write text (word[text]);
     ga  rb22  , ca  15     ;   if more words then
     hv (rb21) Dt  41       ;     begin text:= text + 41; goto next text word end

b22: sy1.3[char], cln −6    ;
     [see 3b20]             ;
     nc  160   , hh  rc6    ;
     pa  b11   t  a19       ;   GPseg:=  end comp;
     arn 8e4   , hs  e11    ;   from drum (input track, GP pl);
     lk (b10)  , sy  62     ;   writechar (BLACK);
b23: ps e42−2  , ud  16e4   ;   s:= return addr;  select(normal;
     [return addr]          ;
[−2] grn rc1−1 VX  1   M    ;   clear this track for marks
     hr  s1                 ;   return; comment if stop then to
     hv  r−2                ;     next GP seg which will be end comp;

d e31=k, e32=j              ; Define text loading parameters
d e10=120e10, e22=3e22      ; Reserve room for mess 2 and 3, tracks 2 and 3 e14.
d e24=e24+e41+e41+e41       ;

e                           ; end mess 1
```

```
b k=e31, i=0               ; load common texts.
i=e32                      ;
e33: tprogram too big;     ;
e34: tstack;               ;
e45: tpass sum;            ;
e46: tpass medium;         ;
e32=i;
e    ;
```

```
b c11                       ; Block for next seg 1, track 4e14.
i=e10, a17=e24              ; Define GP seg for next seg 1.
e23=k                       ; Define abs track for segment table
c1:  it (e43)  t   1        ; next segment: seg no:= seg no + 1;
     arn rc3   X       IRC ;   R:= M; M:= set RC (seg table[seg no]);
     hh  rc10          NZ  ;   if R ≠ 0 then goto segment driver only;
     hh  rc9           LRB ;   if LRB then goto segment only;
c2:  arn rc4   t   1   IRC ; prepare new pass:
     gr  e13   t   1   MRC ;   The code from 1c4 to c10 is moved to inbuf1 ff,
     hv  rc2           NB  ;   this place is cleared for marks and
     hv (rc11) DVt c5      ;   the moved code is entered.
[segment table]             ;
c3: e21=j, a8=c3−e28        ; Define relative addresses for segment words.
    ; Format
qq  ; GP, not segment word
qq  ; 2   pass 1         Each segment of the compiler is described by one word
qq  ; 1   pass 2         which is loaded by the segment itself or, for the
qq  ; 1   pass 3.1       std pr segment,  by merger.
qqf,; 3   std.pr.ident.    Three main formats are used:
qq  ; 2   pass 3.2
qq  ; 1   pass 4         1. Words which describe a new pass, i.e. byte input−
qq  ; 1   pass 5.1          output is adjusted:
qq  ; 2   pass 5.2          qq first core.9 + words.19 + pass no.24 + pass bits.29 +
qqf,; 2   std.pr.descr.                                    entry in core.39;
qq  ; 1   pass 6            pass bits: change direction: 1.29 else 0.29
qq  ; 1   pass 9                       backward pass   : 1.28 else 0.28
qq  ; 1   pass 7
qq  ; 1   pass 8.1       2. Words which describe next part of current pass:
qqf,; 2   std.pr.code       qq first core.9 + words.19 + 1.20 + entry in core.39 f;
qq  ; 2   pass 8.2
qq  ; 2   pass 8.3 (RS)  3. Words which describe a std.proc segm:
    ;                       qq words.19 f,;

c4=i−1, c5=1e13−e30         ; terminate byte output
     pi (16e4) , hsn e3     ;   in:= modebits; byte out (0);
     grn a9    , hsn e3     ;   prevent testoutput; byte out(0); byteout(0);
c6:  pp  rc7   , hsn e3     ;   change return from full track to c7;
     gp  b16   , hh  rc6    ;   while track not full do byte out(0);

c7:  hh  rc9           NPA  ; if pass inf wanted then
     gm  rc7   , hs  e9     ;   begin newline; print 1(used tracks);
     pmn 1e4   , hs  e8     ;   if inf 1 ≠ 0 then print (inf 1);
     arn 2e4   , pm  3e4    ;   if inf 2 ≠ 0 then print (inf 2)
c8:  nc  0     , hs  e7     ;   end;
     hvn rc8   X       NZ   ; Segment only: set return to get GP seg;
c9h: pm  rc7   , ps  e42    ;   goto next GP seg;

c10h:hvf e42   , ga  b10    ; segment driver only: GP pl:= Raddr;
[−2] grn rc3   Vt  1   M    ;   clear this place for marks;
c11: hr  e30                ;   comment also used from c2 ff;
[1]  hv  r−2                ;   return to (get GP seg);
e
e10=40e10, e22=1e22         ; Set GP load parameters
e24=e24+e41                 ;
e10: [check load addr]      ;
```

```
b c11                       ; Common block for next seg 2, track 5e14
                            ;   only one of the following versions is loaded:

< −e44+1                    ; Drum version:

c1:  gmn e13   XV      IZC  ; Get segment: e13:= R:= M; goto start;
c2:  arn 12e4  , hv  rc7    ; Get word: R:= track rel pos 9 + track no pos 39;
                            ;   goto move word;
     gt  r     , pp [words]; Start: sumok; transport ok:= t;

c3h: ga  rb25  , arn 12e4   ; next: R:= track rel pos 9 + track no pos 39;
     bs  p−512 t   −473     ;   if 39 < p ∧ trackrel = 0 then
     ca  0     , hv  rc6    ;     goto whole tracks;
     hs  rc7                ;   move word;
     pp  p−1   , it  1      ;   p:= p − 1; first core:= first core − 39;
c4:  qq (rb25) t   40       ; after whole tracks: first core:= first core + 40;
     ncn p     , hh  rc3    ;   if p ≠ 0 then goto next;
     pa  rb26  , pmn 8e4    ;   new:= true;
     hs  rc9   X            ;   fill buf (input track);
c5:                         ;
b25: ar e16−1[firstcore]t−1 ; sumit:
     ar  2     D       LA   ;   R:= sum of read words including marks;
     ar  1     D       LB   ;   M:= e13;
     pm  e13   X            ;   sumok:= R = 0;
     ca (rb25) X            ;   return;
     hr  s1            IZB  ;
     hv  rc5                ;

c6:  hs  rc10               ; whole tracks:
     lk (rb25)              ;   from drum (first core, trackno);
     pp  p−40  , arn 1a16   ;   p:= p − 40; trackno:= trackno + 1;
     ac  12e4  , hv  rc4    ;   goto after whole tracks;
c7:
b26: bs [new]  , hh  rc8    ; move word:
     hs  rc9                ;   if new then fill buf(trackno);
c8h: arn 12e4  , ga  rb27   ;   rel:= trackrel;
     ar  1     D            ;   track rel:= track rel + 1;
     ca  40    , ar  rc11   ;   if trackrel = 40 then
     ga  rb26  , it (rb28)  ;     begin track rel:= 0; trackno:= trackno + 1 end;
b27: pm [rel]  X       IRC  ;   new := track rel = 0;
     gr (rb25)         MRC  ;   store[first core] marks := R:= store[rel+trackbuf];
     gm  12e4  , hr  s1     ;   return;

c9:  hs  rc10               ; fill buf:
     pa  rb28  , it  41e13  ;   track buf:=
     bs (b17)  , it  1e13   ;     if inbuf 2 < inaddress then inbuf 1 else inbuf 2;
b28: lk[trackbuf] t 42e13   ;   from drum (track buf, bits (10, 39, R);
     vk (e17)  , hrn s1     ;   wait drum; R:= 0; return;

c10: tk  10    , ck  30     ; subroutine for track select;
     hv  e11                ;

c11: qq  −40   t   1.29     ; constant used in 2c8;
```

```
×                          ; Buffer version:
                           ;
c1:  gmn e13   XV      IZC ; Get segment: e13:= R:= M; M:= 0; goto start;
c2:  arn 1a16  , hv  rc3   ; Getword: R:= 1 goto
     gm  rb26  , ga  rb25  ; start: get word exit:= false; first core:= bits(0, 9, R);
     ck  10    , tl  −30   ;   R:= bits(10, 19, R); sum ok:= transport ok:= t;
c3:  sr  12e4  , gr  41e13 ; get R words: words:= R;
     qqn               NT  ;   if rest in buf < R then R:= rest in buf;
     sc  41e13 , ar  12e4  ;   words:= words − R;
     hh  rc7           LZ  ;  if R = 0 then goto get next block;
     tk  20    , ar  rb33  ;   il (R) words from: (topbuf − restinbuf) to:
     ar (rb25) D           ;     (firstcore);
     sr  12e4  , il  0     ;
b26: arn 1a16  , hv  rc6   ;   if get word exit then goto exit 1;
     ck  20    , tk  30    ;   first core:= first core + R;
     ac  rb25  , ck  10    ;   rest in buf:= rest in buf − R;
c4h: sc  12e4  , arn 41e13 ; test more words: R:= words;
     hv  rc3           NZ  ;   if R ≠ 0 then goto get R words;
c5:
b25: ar e16−1[firstcore]t−1; sumit:
     ar  2     D       LA  ;   R:= sum of read words including marks;
     ar  1     D       LB  ;   M:= e13;
     pm  e13   X           ;   sum ok:= R = 0;
     ca (rb25) VX          ;   return;
c6:  sc  12e4  , arn e16−1 ; exit 1: rest in buf:= rest in buf − 1;
     hr  s1            IZB ;   R:= store[first core]; return;
c7h: hv  rc5   , pa  rb29  ; get next block: err ct:= 0;
     arn rb30  , ac  rb31  ;   curr block:= curr block + block in cr;
c8:                        ; repeat block:
b27: arn rb31  , il [unit] ;   il block (curr block, unit);
b28: arn rb34  , il [check];   il status (check word, check);
     il  0     , pm  rb32  ;   us (1) word from: (addr of curr block) to:
c9:  ar  rc10  D           ;     (addr of curr block in buffer);
     us  0     , arn 12e4  ;   if status < 0 then
b29: bt [errct]Vt−200  LT  ;     begin err ct:= err ct + 1; if err ct < t then
     gm  12e4  , hh  rc4   ;       goto repeat block;
     hr  s1            IZC ;     sumok:= transport ok := f return
     hv  rc8               ;     end;
     qq  [fill]            ;   rest in buf:= block length; goto test more words;
b30: qq [block incr]       ; These parameters are set by
b31: qq [current block]    ;   init buf compiler
b32: qq [block length]     ;
b33: qq [top buf]          ;
b34: qq 12e4.9+1.19+b31.39−e28.39 ; word for transport of status word and
                                  ; curr block, top buf is added by init buf comp
c10=b31−12e4               ; address diff. for transport word;

>                          ; End buffer version
e                          ;
e10=40e10, e22=1e22        ; Set GP load parameters
e24=e24 + e41              ;
e10: [ check load address] ;
```

```
b b3, c5                  ; Block for next seg 3, track 6e14

c:   cln −20   , ga  rb1   ; Unpack rest of segment word from M:
     gt  rb2   , tl −5     ;   pass bits:= bits(20,29,M);
     ps  e13−1 , pm  rc5   ;   exit:= bits(30,39,M); Raddr:= bits(20,24,M);
     cl  −10           NZA ;   if −, sum ok then mess(if −, transport ok
b1:  pi[passbits]Vt508 LZB ;     then ↤<pass medium↦ else ↤<pass sum↦, 2, 0);
     gm  e13   , hv  e5    ;   comment mess is called in a funny way because
     hh  rc4           LZA ;   it overwrites this track; pass bits 089 to in;
d c6=e20+i−c−40            ;   if LZA then goto possible pause;
     ca r−c6−i+9,hhn e29   ;   if this place is at e20−40 then
                           ; skip pass 9: goto next segm;
     ga  b6    , ud  14e4  ;   pass no:= Raddr;  select(typewriter);
                           ; prepare new pass:
     pm  8e4   , arn 9e4   ;   M:= input track; R:= output track;
     hv  rc1   X       LRA ;   if −, change direction ∧ discmode then
     hv  rc1           NQA ;     begin R:= last bit (R − first track);
     sr  5e4   , mb  1a    ;     M:= R + first track; R:= first track − R − 1
     gr  8e4   , ar  5e4   ;     end
     pm  5e4   X           ;     else if change direction then swap;
     sr  8e4   , sr  1a    ;   input track:= M; output track:= R;
c1:  gm  8e4   , gr  9e4   ;
     qq (b11)  t  e41  LRB ; Get byte input output segm:
     qq  r2    , it  e3    ;   if backward pass then GP seg:= GP seg + GP segsz;
     pa  b10   , hs  e30   ;   GPpl:= byte out; get GP seg;
     pa  2e4   , pa  3e4   ;   inf 1:= inf 2:= 0;
c2:  arn 8e4   , hs  e11   ;   track(input track);
     pa  b5    t   59      ;   point:= 59; print count:= 10;
     lk  1e13  , ud  a2    ;   from drum(the track, base inbuf 1)
     arn 8e4   , ar  1a    ;   if backward pass ∧ discmode then
     grn e4    V       NRB ;     output track:= input track + 1
     gr  9e4           LQA ;   else if −,backward pass then CRcount:= 0;
     grn a12           LQA ;   if discmode then no release used tracks;
     grn 1e4           LQA ;   if discmode then used tracks := 0;
     arn a9    , gt  e5    ;   set test value to find GP pl;
c3:  arn(b18)  t   1       ;   while next byte = 0 do;
     hs  a14           LA  ;   byte addr:= byte addr − 1
     qq (b18)  V   −1  NZ  ;
c4h: hv  rc3   , ud  14e4  ; possible pause: select (typewriter);
     lyn       D       LKA ;   if LKA then lyn;
     pi (16e4) , ud  16e4  ;   in:= mode bits;  select(normal);

[−2] grn rc−1  Vt  1   M   ; finis: Clear this place for marks;
b2:  hv  0     t  [exit]   ;   goto exit;
     hv  r−2               ;

c5:  qqn e46   t e45+1.26  ; packed text params for use if −, sumok;
e                          ; end next seg 3

e10=40e10, e22=1e22        ; Set GP load parameters
e24=e24+e41                ;
e10: [check load addr]     ;
```

```
b k=e22+e14, i=e3, a1, b1  ; Forward pass track, track 7e14, core e3
a18=e24 [GP seg forward]   ;   comment must follow next seg 3;

e3:  hv 1e3[i] t   2   NKB ; byte out: comment 1e3 corresponds to i = 0;
a9:  a10=−a9[used in a2−1]       ;
     hs e6 t 123e13 [see endpass] ;   if LKB then test print;
     hv (e3)   t   2       ;   i:= i + 1; goto pack [i];
e12:                       ;
b15: gr 82e13[out addr] t 1; pack[1]: out addr:= out addr + 1;
     hr  s1                ;   buf[out addr]:= R; return;
     ck  −10   , gt (b15)  ; pack[2]: part 2 buf[out addr]:= Raddr;
     hr  s1                ;   return;
     ck  20    , ac (b15)  ; pack[3]: R:= R shift 20;
     hr  s1                ;   buf[out addr]:= buf[out addr] + R; return;
     ck  10    , ac (b15)  ; pack[4]: R:= R shift 10;
     pa  e3    t   a9      ;   buf[out addr]:= buf[out addr] + R; I:= 0;
     hr  s1            NA  ;   if −, mark A then return;
     arn 10e4  , ac  1e4   ; track out: used tracks:= used tracks + increm;
     arn 4e4   , sr  1e4   ;   if used tracks > available tracks then
     hs  e5            LT  ;     mess (ϕ<program too big>ϕ, 2, 0);
     hv  r1    , qqn e33   ;
     hs  a1                ;   next track (output track);
     qq  9e4   , arn(b15)  ;   R:= buf[out addr];
     is (b15)  , sk  s−40  ;   to drum(the track, out addr − 40);
     qq (b15)  t −82   LB  ;   if mark B then out addr:= out addr − 82;
b16: hv  e12 [see end pass];   goto pack[1];
     [a12 is cleared by end pass when discmode]
a12: srn 10e4  , ac  1e4   ; track in: if −, discmode then
     hs  a1                ;     used tracks:= used tracks − increm;
     qq  8e4   , arn(b1)   ;   next track(input track); R:= buf[in addr];
     is (b1)   , lk  s−40  ;   from drum (the track, in addr − 40);
     qq (b1)   t −82   LB  ;   if mark B then in addr:= inaddr − 82
e2:b1:a14:
b17: arn 81e13 [inaddr] t 1; unpack next: in addr:= inaddr + 1;
[1e2]ga  e13−4 V       NA  ;   R:= buf[in addr];
    [NA is removed by init comp for use in type in in pass 1]
     hv  a12               ;   if mark A ∧ −, pass 1 then goto track in;
     tk  10    , ga  e13−3 ;   for j:= 1 step 1 until 4 do
     tk  10    , ga  e13−2 ;     byte buf[j]:= part (j) of: (R);
     tk  10    , ga  e13−1 ;
b18:                       ;
e1:  arne13−1 [byteaddr]t−4;   byte addr:= byte addr − 4;
     hr  s1                ;   R:= byte buf [byte addr]; return;
a13:                       ;
a1:  arn 10e4  , ac (s1)   ; next track: comment addr of track no in s1;
                           ;   track no:= track no + increm
e15: [pass 1 starts here and replaces the next 4 words by
     arn(s1)   y hv  e11   ;   R:= track no; goto track
     ...                   ;   ...                      ]
     arn 6e4   , sr (s1)   ;   if track no > last track then
     arn (s1)  V       NT  ;     track no:= track no − available tracks;
     srn 4e4   , hh  a1    ;   R:= track no; goto track
     hv  e11               ;
     qq [fill]             ;
d e24=e24+e41, e10=40e10   ; Set GP load parameters
d e22=1e22                 ;
e                          ;
```

```
b k=e22+e14, i=e3, a1      ; Backward pass track, track 8e14, core e3;

e3:  hh  e3[i] t   2   NKB ; byte out: comment e3 corresponds to i = 0;
                           ;   if LKB then test print;
     hs e6 t 122e13 [see endpass];   i:= i + 1; goto pack[i]
     hv  a1    , ck  10    ; pack[1]: R:= R shift 10;
e12:                       ; store first: out addr:= out addr − 1;
b15: gr123e13[outaddr]t−1  ;   buf[out addr]:= R; return;
     hr  s1    , ck  20    ; pack[2]: R:= R shift 20; buf[out addr]:=
     ac (b15)  , hr  s1    ;     buf [out addr] + R; return
     ck  −10               ; pack[3]: part 2 buf [out addr]:= Raddr;
     gt (b15)  , hr  s1    ;   return;
     ac (b15)              ; pack[4]: buf [out addr]:= buf[out addr] + R;
     pa  e3    V   e3      ;   i:= 0;
a1:  hh (e3)   t   2       ; comment a1 used after test print;
     hr  s1           NA   ;   if −, mark A then return;
e25: arn 10e4  , ac  1e4   ; track out: used tracks:= used tracks + increm;
     arn 4e4   , sr  1e4   ;   if used tracks > available tracks then
     hs  e5          LT    ;     mess (‡<program too big›, 2, 0);
     hv  r1    , qqn e33   ;
     hs  a13               ;   next track (output track);
     qq  9e4   , arn(b15)  ;   R:= buf[out addr];
     is (b15)  , sk  s1    ;   to drum (the track, out addr + 1);
     qq (b15)  t   82  LB  ;   if mark B then out addr:= out addr + 82;
b16: hv  e12 [see end pass];   goto store first;
     [a12 is cleared by end pass when discmode];
e35:
a12: srn 10e4  , ac  1e4   ; track in: if −, discmode then
     hs  a13               ;     used tracks:= used tracks − increm;
     qq  8e4   , arn(b17)  ;   next track (input track); R:= buf[in addr];
     is (b17)  , lk  s1    ;   from drum (the grack, in addr + 1);
     qq (b17)  t   82  LB  ;   if mark B then in addr:= inaddr + 82;
e2:
b17: arn 42e13 [inaddr] t−1; unpack word: in addr:= in addr − 1:
     ga  e13−1 V       NA  ;   R:= buf[in addr];
     hv  a12               ;   if mark A then goto track in;
     tk  10    , ga  e13−2 ;   for j:= 4 step −1 until 1 do
     tk  10    , ga  e13−3 ;     byte buf[j]:= part (j) of: (R);
     tk  10    , ga  e13−4 ;
e1:  arne13−1[byteaddr] t−4;   byte addr:= byte addr − 4:
     hr  s1                ;   R:= byte buf [byte addr]; return;

a13: srn 10e4  , ac (s1)   ; next track: comment addr of track no in s1;
     arn(s1)   , sr  5e4   ;   trackno:= trackno − increm;
     arn(s1)   V       NT  ;   if trackno < first track then
     arn 4e4   , hh  a13   ;     trackno:= trackno + available tracks;
     hv  e11               ;   R:= track no; goto grack;
     qq [fill]             ;
e16:                       ; first instruction of a pass is read in here;

d e24=e24+e41, e10=40e10   ; Set GP load parameters
d e22=1e22                 ;
e                          ;
```

```
[6.4.67]
[Gier Algol 4, GP, page 12]
i=e10                       ;
b c7                        ; Block for init comp, track 9e14;
a19=e24                     ; Define GP seg for init comp;

     bs (e43)  , hv  rc6    ; End comp or init comp drum version:
     arn 11e4  , ar  rc5    ;   if segno > 0 then goto end comp;
                            ;   trackno:= base GP + tracks in GP; track rel:= 0;
a21: gr  12e4  , pm  rc4    ; Init comp buffer or paper tape vesion:
c1h: pp  86    , pp  p-1    ;   for i:= 39 step -1 until 0 do
     can p-4   , it  -43    ;     inbuf 2[i] mark0:= outbuf 1[i]mark0:= 0;
     grn123e13 t   -1  M    ;   for i:= 4 step -1 until 0 do
     bs  p-13  , hh  rc1    ;     byte buf[i] mark0:= 0;
     acn 13e4  t   -1  M    ;   for i:= 12 step -1 until 0 do
     bs  p-4   , hh  rc1    ;     marks of e4 params:= 0;
     gm  17e4  t   -1  MA   ;   for i:= 3 step -1 until 0 do
     bs  p-2   , hh  rc1    ;     e4 param[i+13]:= instr(qq 1023, vy);
     grn 205e13t   -82 MA   ;   buf limit 3 mark A := buf limit 1 mark A:= 0;
     grn 164e13t   -82 MC   ;   buf limit 2 mark C := base in mark C:= 0;
     bs  p-1   , hh  rc1    ;

     pa  b11   t   a18      ;   GP seg:= forward pass; GP pl:= byteout;
     qq  r2    , it  e3     ;   get GP seg;
     pa  b10   , hs  e30    ;
[2]  arn 8e4   , ga  15e4   ;   save return point addr in 15e4
     tk  14    , gr  2e4    ;   Save return point track in pos 25 in 2e4;
     arn 6e4   , gr  -2     ;   Save search and start medium track in -2;

c2h: arn 9e4   , pp  p-1    ;   for i:= 1 step -1 until -2 do
     ck  -10   , gt  p16e4  ;     by select[i]:= part [i, units];
     bs  p3    , hh  rc2    ;
     grn e4    , arn 5e4    ;   CRcount:=0;
     gr  9e4   , ar  4e4    ;   output track := first track;
     sr  1a16  , gr  6e4    ;   stringtrack:= inputtrack := lasttrack:=
     gr  8e4   , hs  e11    ;     first track + available tracks - 1;
     sk  42e13 , pi (10e4)  ;   to drum(string track, inbuf 2);
     gi  16e4  , arn 1a16   ;   modebits:= part 1(param10e4)
     ar  1a16          LQA  ;   increment:= if discmode then 2 else 1;
     gr  10e4  , sc  9e4    ;   output track:= output track - increment;
     grn 1e4   , hhn e29    ;   used tracks:= 0; goto next segment;

c4:  qq  1023  , vy         ; instruction for initalization of by select;
c5:  qq  1e22.39            ; tracks in GP. Only used by drumversion

c6:  pa  2e4   , arn 2e4    ; end comp: return track to  byte out;
     tk  -14   , hs  e11    ;   goto byteout[relative return]
     lk  e3    , vk (e17)   ;
     it (15e4) , hvn e3     ;

     qq [fill]              ;
     qq [GPsum if drum]     ;

d e24=e24+e41, e10=40e10    ; Set GP load parameters
d e22=1e22                  ;
e10: [check load addr]      ;

e                           ; End init comp
```

```
[30.10.67]
[Gier Algol 4, GP, page 13]

[Init GP: 3 versions, entered with GP in core]

< −e44+1                    ; Drum version
d a20=e42                   ;   Define init GP. No code.

×                           ; Buffer version

a20: arn 14e4  , gt  b27    ;   Set parameters in next seg 2:
     ck  −10   , gt  b28    ;   Set il [unit], il [check]
     arn 11e4  , gr  b30    ;   Set block incr
     arn 13e4  , gr  b32    ;   Set block length
     arn 12e4  , gr  b31    ;   Set current block;
     tk  24    , ck  16     ;   topbuf:= bits(24, 39, current block) + blocklength;
     ar  b32   , gr  b33    ;   GP base:= top buf + GP seg sz pos 19;
     ac  b34   , ar  a22    ;   check transport word:=
     gr  11e4  , ar  a23    ;     check transport word + topbuf;
     us  0     , srn a24    ;   GP to buf ( top buf );
     ar  b32   , gr  12e4   ;   for rest in buf:= − GP size,
     hv  r−1           LT   ;     GP size + block length while rest in buf < 0 do;
     gr  12e4  , hh  a21    ;   goto Init comp buffer version;

a22: qq        t   e41      ;   GP seg sz
a23: qq  e28  t e28−e10−e41;   param word for GP to buf;
a24: qq 40e10.39−e28.39     ;   GP size;

i=39e10                     ;
     qq [GPsum if buffer]   ;

d e24=e24+e41, e10=40e10    ; Set GP Load parameters, fill up this track
d e22=1e22, i=e10           ;

>                           ; End buffer version

i=e28                                   ;
     qq e10−e28, hv  a20                ;   load 2 final words of GP
     qqf a8.9+e4.19+e27.20+e44.21,      ;   a8 = relative address of first segment.

e [ final end GP ]          ;
```

```
[This segment consists of: End pass 8, Initrun, RS code, RS error tracks.

   RS error is loaded as a seperate drumblock inside the drumblock
   holding Endpass 8, Init run, and RS code.
   End pass 8 and Init run are loaded with internal references
   r−marked. RS code is absolute addressed and must have the last
   instruction in e37 − 1.
   The start address for this drumblock during loading is
   therefore computed below as: e36 = e37 − 40 × number of tracks in
   RS code − 40 × number of tracks in Initrun − 4 × number of tracks
   in Endpass 8;
   When the segment is taken to core after pass 8 it will include
   the tracks for RS error with the last instruction in e37−1 and
   the first in e36 − 40 × number of tracks in RS error]

e39=190          ; define size of RS code:
< e27            ;   e39:= basic size of RS code
e39=e39+11       ;   if buffer mode then e39 :=e39+11;
>                ;

<e40+1 , <−e40+1;   if number of units ǂ 0 then
×                ;
e39 = e39 + 51   ;   e39 := e39 + 51;
>                ;
<e40              ;   if number of units > 0 then
e39 = e39 + e40  ;   e39 := e39 + number of units;
>                ;

e36=e37−200      ; define core start for RS:
< e39−200        ;   e36 = first free after RS during run − 5 × 40;
e36 = e36−40     ;   if 5 × 40 < e39 then e36:= e36−40;
>                ;
< e39−240        ;   if 6 × 40 < e39 then e36:= e36−40;
e36 = e36−40     ;
>                ;


e36=e36−40       ; reserve core for init run
e36=e36−40       ; reserve core for end pass 8

b k=e22+e14, i=e36, a50, b50, d40 ; drumblock head for RS;
                                  ;
d33 = e40 + e40 + e40 + e40       ;
d33 = d33 + d33 + 1        ; reserve buffer cells for SAMBA file tables

d1=e38           ; define first track place
d3=4             ;   minimum number of places
d4=164           ;   number of track places from start × length of trackplace
d8=19            ;   max number of trackplaces
```

```
[END PASS 8]

a42: nt (13e4) , qq (rb34) ; Modify init run:
     arn 1e4   , ck −10    ;   stdprtr:= −run track;
     sc  13e4              ;   run track := run track − used tracks;
     arn 3e4   , gt  rb35  ;   start rel:= part 2 (inf 2);
     arn (13e4) D          ;   first tr:= run track;
     ga  rb32  , ck  10    ;   comment also to initial jumb in b35;
     ac  rb35  , ps (2e4)  ;
     nt  s3    , pa  rb33  ;   displsz:= −rel stackref 0 − 3;
     ps  s2    , it  sd9   ;   last used:= addr (displ [0]) + 2 + rel stack ref 0;
     pt  rb33  , pi (16e4) ;

     srn d28   D           ; Add RS tracks to compiled code:
     ac  13e4  , tl  −30   ;   run track:= run track − tracks in RS;
     ar  9e4   , gr  9e4   ;   output track:= output track − tracks in RS;
     sr  5e4   , pp r40a42 ;   if output track < first track then
     hs  e5            LT  ;     mess(‹<program too big›, 2, 0)
     ac  5e4   , qqn e33   ;   else first track:= output track;

a43: arn 9e4   , hs  e11   ;   for p:= first RS place step 40 until last RS place do
     sk  p     , pp  p40   ;     begin to drum (output track, p);
     arn rd21  , ac  9e4   ;     output track:= output track + 1;
     ncn p−e37+1,hv  ra43  ;     end;

     hv  ra44          NPA ; Pass information:
     hs  e9                ;   if pass information wanted then
     pmn 1e4   , hs  e8    ;     begin
     srn(13e4) D           ;     new line; print 1 (used tracks, −run track);
     hs  e7                ;     comment pass 8 used and total used;
     pmn 5e4   , dl  rd20  ;     print (first track : 960);
     ck  −10   , hs  e7    ;     print (first track mod 960);
     cln −10   , hs  e7    ;     end;

a44: pa  2e4   , arn 2e4   ; Exit from translation:
     tk  −14   , hs  e11   ;   to core (exit track , byteout );
     lk  e3    , vk (e17)  ;
     srn(13e4) D           ;   R:= total size.23 + first track.39;
     ck  −14   , ar 5e4    ;
     it (15e4) , hv  e3    ;   goto byteout [rel exit];

     qq                    ;   fill;
     qq                    ;     −
     qq                    ;     −
     qq                    ;     −
     qq                    ;     −
     qq                    ;     −

d26 = k − e14              ;   first relative RS track;
```

```
[INIT RUN track: May be read into core anywhere between 81e38 (e38=first
 track place) and lower limit for sr0 (which probably is at least 512).
 It is entered in relative 0 which in core must be preceded by four or,
 if buffer mode, six parameters:
 −6: lower limit in buffer        .39; only used in buffer mode
 −5: initial last used in buffer  .39;   −    −   −    −    −
 −4: used in top of free          .39;
 −3: abs track for look up        .9 ;
 −2: abs return track             .39;
 −1: abs track for init run       .39;
by specifies the units to be selected in case of error during run.
Note: Parameters which are set on this track by end pass 8 are marked ×]

a40: pp  a40   , hs  ra41  ; Read RS code tracks:
     pp  p40   , ncn p−e37 ;   for p:= first RS track step 40 until last RS place do
[ 2] lk  p     , hh  ra40  ;     to core (next track, p);
     it (rb37) , pt  d6    ;   next track; set error group in RS code;
     pp (rb36) , arn d25   ;   set error track in RS code;
     ac  ra40−1, hs  ra41  ;   param [−1]:= param [−1] + no of error track − 1;
     gp  d6    , pp (rb36) ;   next track; p:= abs group;
b32: qq [×firsttr], hs a34 ;   set track select parameters and init display;
b33: qq [×displsz], ps[×last used]; comment hsa34 has 4 parameters;
                                  ; Raddr = absfirst track, p = abs group,
a35: pp  p−1   , ar   d23  ;        (s) = first track, (s1) = displsz − 2.
     gr  d2    t −1    MA  ;        Will return to s1h with p = tracktablesize
     bs  p−1   , hv  ra35  ;        and R = track table [1];
<e27,<−e40+1 [buffer mode,no tape units] ; Init track table:
     arn ra40−6, gr  d13    ;   for i:= 2 step 1 until track table size do
×<e27        [buffer mode, tape units ]
     arn b44   , gr d13     ;
>
<e27         [buffer mode]
     ac  d14   , arn ra40−5;     track table[i] MA:= R:= R + place size.19;
     tk  15    , gr  b20   ; if buffermode∧−,tape units then lower buf:=R:=param[−6];
×   [end buffer mode]      ; if buffermode∧ tape units then lower buf:=R:=cell[b44];
                           ; if buffer mode then begin bufword:=
                           ;   bufword+R; last used in buf:=param[−5] end;
     qq                    ;   fill
     qq                    ;   fill
     qq                    ;   fill
>                          ;
     gs  b20   , pmn ra40−2;
     arn ra40−4, gr  d32   ;   set used in top of free;
     dln rd20  , ck  20    ; Set last used in core and
     ac  d17   , cln −10   ;   exit group and
     ga  d18   , pmn rd24  ;   exit track;
     arn ra40−3 , ga d31   ; Set look up track;
[−2] gm  0             MA  ; Set spill exit in abs core 0;
     pm  rd22  , ud  ra36  ;   error unit:= by;
     gk  r−2   , it  (r−2) ;
     pt  d19   , ud  ra36  ; Set track place word in
     gm  d1            MC  ;   first 4 track places;
a36: gm  d1    t e26   MA  ;
     it (rb32) , pt  d15   ; Set first track;
b34: it [×stprtr], pt  d16 ;        ; Set first std proc track;
b35: hsa8,qq[×start rel+×starttr.29]; Jump to start compiled program;
a41: arn rd21  , ac  ra40−1; next track:
     pm  ra40−1, dln rd20  ;   M:= param [−1]:= param [−1] + 1;
     ck  −10   , ga  rb36  ;   abs group:= M : 960 + 960;
     cln −10   , ga  rb37  ;   abs track:= Raddr:= M mod 960;
b36: vk[absgr] t  960      ;   select (abs group); select (abs track);
b37: vk[abstr] , hr  s1    ;   return;
```

[Note: parameters set by init run are marked ×]

```
d20: qq          960.39    ; constants: 960
d21: qq            1.39    ;   1
d22: qq 512    , hs  a10   ;   track place word
d24 = 39a40 [check load]   ;
d24: qq        , hh  d31   ;   spill exit

     pa  8     , hsf 3     ;
     qq  [fill]            ;   fill track with zeros:
     qq  [fill]
     qq  [fill]
     qq  [fill]
```

```
[26.8.67]
[GIER Algol 4, RS, page 5]
[Note: parameters set by init run are marked ×]

< e40+1, < −e40+1          ;   if number of tape units ≠ 0 then
×                          ;   begin
e60 = i                    ;
     arn b13   , sr  b29   ; INITIALIZE BUFFER:
     hv  r3           LT   ;   comment all words are initialized
     gr  b13   , ar  b38   ;    to zero with marks 00;
     us  0     , hv  r−3   ;
     arn b45             ;   insert unit numbers in the locations
     arb 29    , it  1     ;   in the buffer which correspond to
     it (b23)  , bs e40+1  ;   the address part of 3c69 in the filetables;
     us 0      , hv r−2    ;
e60 = −e60 + i             ;   SAMBA check;
>                          ;   end;
b31: ps [save s], arn d2   ; initialize display:
a31: gm  d9    t   −1  MA  ;   comment display size − 2 is in (s1);
     pm  a32   , it  −1    ;   display [0]:= display [−1]:= display word 1;
     bt (s1)   , hv  a31   ;   for i:= −2 step −1 until −display size do
     pp  d8    , hh  s1    ;     display [i] := display word 2;
                           ;   return to s1h;
                           ;   comment R = track table[1]; p = track table size;

a32: arn 0     , ga  r1    ; display word 2
a33: qq  0     , hh  a8    ; display word 1

[The code above is part of init run and is overwritten by the track table.
 Core picture when block 0 is entered:
 0:  qq        , hh  c64   ; alarm for spill.
 ––
 d1: qqf 512   , hs  a10   ; 1. track place.
 ––
 d1+e26:
     qq  512   , hs  a10   ; 2. track place.
 ––
d1+e26+e26:
     qq  512   , hs  a10   ; 3. track place.
 ––
d1+e26+e26+e26:
     qq  512   , hs  a10   ; 4. track place.
 ––
     arn 0     , ga  r1    ; display [−maxblockno − 1|
     arn 0     , ga  r1    ; ..
     ..                      ..
     qq  0     , hh  a8    ; display[−1|
 c:d9:qq 0     , hh  a8    ; display [0|
 d2−d8+1:
     ca  0     , hh  d1+(d8−1) × e26; track table [last place|
     ..
     ..
 d2: ca  0     , hh  d1            ; track table: [1|

          RS−code                  ;

 The track table is further used for run error indication:
   error(n) will set part 2 (track table [d2−r|) to 0 for
   use by the error track]

c49:
d2:  ca  0     , hh  d1    ;  base track table

d9=d2−d8, c =d9            ;  define display [0]
```

```
     [RS code]             ; Is entered here when track not found in table:

     hh  b28            NRC;   if get place then goto search min priority;
a1: c23:                   ; get track: s:= track; vkgr (group);
b1:  ps[×track], vk[×group];   if s−512 < base − 512 then
b26: it s−512,bs[×base−512];     begin vkgr (group − 1); vk (s − base + 960) end
b27: vk[×group−1], is s960 ;   else vk (s − base);
b28: vks[×−base],ps (b2)   ;   comment base = trackno corresp. to track 0 in group;
                           ; search min priority: s:= i:= top place;
a2:  gs  r1    , arn s     ;   for i:= i − placelength while
     sr  −1    t   −e26    ;   −, mark B[i + placelength] do
     ps (r−1)          NT  ;     if priorpart [i] ≤ priorpart [s] then
     hh  a2            NB  ;       i:= s;

     arn d10   , ac  b14   ; read track: if −,get place then
     hh  r1            NRC ;     begin tracks transferred:= tracks transferred+1;
     lk  s1    , gs  b11   ;     from drum (track, s+1) end;
                           ;   curr place:= s;
a3:  bs s−d1+512e26t512e26 ; track to table:
     ps  s−e26−1,hv  a3    ;   while s > first place ∨ s < first place −
     it (b1)   , pa s+d2−d1;     place length do s:= s − place length − 1;
     ps (b11)  , arn(b7)   ;   track part [s − first place + base table]:=
                           ;     track; s:= curr place;
                           ; try to reserve new place:
     ca  0     , hv  a4    ;   if track part [table top] ≠ 0 ∧ last used
     is (b20), it s−e26−512;     −512 − place length > top program then
     bs (b3)   , hv  a4    ;     begin table top:= table top − 1;
     gan(b7)   t   −1      ;     track part [table top]:= 0;
     arn b10   , ar  a5    ;     top place:= top place + place length;
b2:  gr d1+d4−e26 t e26 MA ;     store [top place ] mark A:= pack 4
     [top place]           ;       (priority − 4, track is in, 0, op((hs));
b3:  qq d1+d4−512 t e26    ;     top program:= top program + place length
     [top program]         ;     end;

a4:  hh  a10           NRC ;   if −,get place then wait drum;
d30: vk[×group], hh  a10   ;   goto set exit addr;

a5:  qq −4.9+123.26+42.35, qq d7 ;   constant: −(arn4D 1), hs a10
                           ;
a6:  ga  b4    , it  0     ; clean up:
a7:                        ;   for i:= top place, i − place length while
b4:  arn −1    Vt −e26 NB  ;     −, markB [i + placelength] do
     hv (a11)  D   −1004   ;     prior part [i]:=
     ar  21    D           ;       if prior part [i] < 491 then −512
     arn 512   D       NO  ;       else prior part [i] − 1003;
     ga (b4)   , hv  a7    ;   priority:= priority − 1004;
                           ;   goto set priority;
```

```
     [next param track: R:= hsc3, qq <rel> + <line no>.29]
c3:  pm  a16   , ud  a16   ;   RC:= 01; track return:= stack param − 1; skip next;
     [jump to next track:  hs c1, qq <rel> + <line no>.29 (f)]
c1:  gr  b21   , arn s     ;   save R:= R;
     gt  b9            IRC ;   rel addr:= set RC (relpart(R));
     arn(b1)   D   1       ;   Raddr:= track:= track + 1;
d19h:hv (b7) , vy[×errunit];   goto table search [top table]
      [err unit is only used by error tracks]

     [call std proc:   hs c4, <drum point>]
c4:  it (b1)   , pt  b8    ;   return track:= track;
     it (b11)  , pa  b5    ;   return place:= curr place;
b5:c32: qq[return place],gsb6 ;   save s:= s;

     [goto local:      ps<op>, hv c2      ]
     [goto track:      hs c2 , <program point>]
c2:                        ;   save R:= R;
a8:  gr  b21   , arn s     ; goto track no save: R:= param [s];
c26:
a9:  gt  b9            IRC ; point in R: reladdr:= set RC (part 2 (R));

     ck  −10   , ga  b1    ;   Raddr:= track:= part 4 (R);
                           ;   goto table search [table top];
                           ;   comment table search will end as follows:
                           ;   When track already in core: At track is in
                           ;     with track place in s;
b7:                        ;   When track taken from drum: At set exit
c5:  hv  d2−3  , arn b5    ;     addr with track place in curr place
     [table top]           ;     and s;

     [return from std proc: hh c5]
b6:[1c5]ps[saves] , ga b11;   curr place:= return place; RC:= 10;
c62:   [return track]      ;   track:= return track;
b8:  pa  b1    V  −1   IRC ;   go to set priority

a10: d7=a10−2[used by a5]  ;
b9:  gs  b11   , ps  s −1  ; track is in: curr place:= s;
a11:         [rel addr]    ; set exit addr: s:= s + rel addr;
b10: arn−492[priority] D 1 ; set priority: priority:= Raddr:= priority + 1;
c65:                       ;
b11: ga −1[curr place] VNO ;   if priority = 512 then
     arn b2    , hv  a6    ;     begin Raddr:= top place; goto clean up end;
     arn b21               ;   R:= save R;
     hh  s1            LRC ;  if go then go to if LRB then right s1
     hv  s1            LRA ;                    else left s1;
b12: hv[track return] t 1  ;   track return:= track return + 1;
                           ;   goto store [track return];

     [get rel track: qq <track no relative to track>, hs c6]
c6:  gr  b21   , gs  b12   ;   save R:= R; track return:= s;
     arn a16   , it  (s)   ;   track:= Raddr:= track + part 1 (store[s]);
     arn(b1)   D       IRC ;   go:= LRA:= f; rel addr:= 0;
     pt  b9    , hv  (b7)  ;   goto table search [table top];

     [get place:  arn c42, hs c63]
c63: gs  b12   , hv  a9    ;   track return := s; goto point in R;
                           ;    comment marks = 00;
```

```
     [prepare block or call: qq <appetite> , hs c7]
c7:  it (s)    , pa  b24   ;   appetite:= part 1 (store [s]);
     arn b20   , ga  b16   ;   stack addr:= last used; mark A:= f;
                           ;   last used:= last used − appetite;
a15: nt (b24)  , is (b20)  ; test last used:
     it  s−512 , bs (b3)   ;   if last used − 512 ≥ top program then
a16: paf b12   V d5    IRC ; start paramtr:
a17: gs  b17   , hv  a27   ;    begin param addr:= s; goto Next end;
d d5=a17−1                 ;   track return:= start paramtr − 1;  RC:= 01;
                           ; release place:
a18: qq (b3)   t   −e26    ;   top program:= top program − place length;
     qq (b2)   t   −e26    ;   top place:= top place − place length;
     it (b7)   t   1       ;   table top:= table top + 1;
     bs d2−d3+2, hv  a19   ;   if table top ≤ max table top then
     pt  d2−1  , hv  d6    ;     goto testcore; error (1);

     [expression as formal: ud <formal> , hs c8]
c8:  arn b1    , ck  10    ;   last used:= last used − 1
     gr (b20)  t   −1  MA  ;   stack [last used] mark A:=
     gp (b20)  , nt  s     ;     pack 4 (sr, s − curr place, vk instr, track);
     nt (b11)  , pt (b20)  ;   mark A:= t;

a19: is (b20)  , it  s−512 ; test core: if last used − 512 < top program then
     bs (b3)   , hv  a18   ;   goto release place;
     hh  a22           NA  ;   if −, go then goto test param track;
                           ;   s:= stack addr of formal;
a20: ps (s)    , arn  s    ; end call: R:= stack [s]; p:= sr part (R);
     pp (s)    , hh  p     ;   goto update [p];

     [goto computed: UA:=if undef then 0 else absaddrlabel; ncn (c30) , hs c13]
c13: pp (b22)  , ps  a12   ;   p:=UA; set return( exit to label);

     [exit block: hs c9]
<e27 [buffer version]      ;
c9:  pp (p)    , arn p1    ;   p:= sr part [store[p]];
     mb  d12   , gr  b20   ;   last used word:= store[p+1] ∧ 26 m −
     sr  p1    , tk  15    ;     bits(26,39,store[p+1]) × 2↑15;
     ac  b20   , hv  s1    ;   go back;
×    [core version]
c9:  pp (p)    , pm  p1    ;   p:= sr part [store[p]];
     gm  b20   , hv  s1    ;   last used word:= store[p+1]; go back;
>                          ;

     [mult:   mln <op> X  IZA
              hs  c14  X  NZA]
c14: ar  d10   X       IZA ;   R:= R + 1; swap;
     ar  512   DV      LZA ;   if M ≠ 0 then error (2) else
     pt  d2−2  , hv  d6    ;     R:= R + bit 0; go back;

a22h:hr  s1    , it  s−512 ; test param track: if top program > s − 512 then
     bs (b3)   , hv  a17   ;   param track ok: goto start param tr;
     nt  s     , nt (b11)  ;   curr place:= curr place − s;
     pt  b9    , hv  a1    ;   rel addr:= −curr place; goto get track;
```

[The actions on this page are all part of the parameter transformation
 mechanism. Each action is headed by the relevant parameter format]

```
     [block information: hv c11, hh displ − <block no> − 1]
c11: gr (b16)  t   −2  MA  ;   stack addr:= stack addr − 2;
     gp (b16)  , pp (b16)  ;   stack [stack addr] mark A:= R;
     gt  r     , is  −1    ;   sr part [stack addr]:= p; p:= stack addr;
     gp  s1    , pm  b20   ;   sr part [displ − block no]:= p;
     gm  p1    , hv  a27   ;   stack [stack addr + 1]:= last used cell;
                           ;   goto Next;

     [array: ps (<displ ref>), hv  c12]
c12: arn s     D           ;   stack [stack addr]:= stack [stack addr] + s;
     qq (b24) t1           ;
     ac (b16)  , hv  a27   ;   goto Next;

d11: qq 1.26+1.36          ;   mask, used by 8a27
b25: pa  b22   t [abs addr];   param constant, used by 9a27

     [end call declared or formal: is (<displ ref>), ps s<block rel> f
      other formal or
      procedure identifier:;       is (<displ ref>), pmns<block rel> f]
a23: hh  a20           NZ  ; described in stack: if R = 0 then goto end call;
     hv  a26   X       IRC ;   set RC (swap); goto Store;

     [program point: <2 half words, address part ≥ 0>]
a24: ar  p     DV          ; program point; R:= R + sr; goto Store;
a25: [constant: <40 bits> no marks]
c57:
b24: qq [appetite] t −1    ; constant: appetite:= appetite − 1;
a26: c10:                  ;
b16: gr [stack addr]t −1MRC; Store: stack addr:= stack addr − 1;
a27: c38:                  ;   Stack [stack addr] mark RC:= R;
b17: pmn[param addr]X t1IRC; Next: param addr:= param addr + 1;
                           ;   R:= set RC (store[param addr]);
     hv  a25           NC  ;   if mark 00 then goto constant;
     hv  a24           NT  ;   if R≥0 then goto program point;
     gr  b18               ;   param:= R;
c51:
b18: qq [parameter] ,      ;   execute parameter as two instructions;
     pt  b25               ;   part 2 [param const]:= abs addr;
     hv  a23           LRC ;   if RC then goto described in stack;
     [absolute addressed:
      simple: ps(n) (<displ rel>) (,) it (n) s<blockrel> (f)
      literal:ps n (stack addr)   (,) it (n) s<stack addr rel> (f)]
     arn b18   , mb  d11   ; Abs addr: R:= param const +
     ar  b25           IRA ;     bit (26, parameter) + bit (36, parameter);
d15: hv  a26, ps[×firsttr] ;   RA:= 1; goto Store
     [first track only used by error track]
```

```
     [case i of : Raddr:= value of i;
                  qq <no of params> , hs c15]
c15: ar (s)    D       LO  ;   index:= Raddr
     ga  b19   V       LT  ;   if index > no of params ∨ index ≤ 0 then
     hv  s1                ;     go back; comment to alarm exit;
b19:                       ;
c16: arn s[index] t 3 IRC  ;   R:= UV:= case param:= set RC (param [index]);
     gr  b23   , gr  b15   ;   if NC then goback 1;
     hv  s2            NC  ;     comment constant;
c52:
b15: qq [execute param] ,  ;   execute case param;
     pa  b22           NRA ;   if label then UA:= abs addr of label;
     gr  b23   V       LRB ;   if float case then
     nkf 39    , grf b23   ;     UV:= RF:= float (value(simple))
d16: hv  s2, ps [×stdprtr] ;   else UV:= value (simple); go back 1
     [stdprtr only used by error track]

     [formal addr: ud<formal> , hs  c28
                   qq w       , arn b22]
c28: arn(s1)               ;   R:= description word;
     hv  s2            NZ  ;   if R ≠ 0 then go back 1;
c25: pt  d2-9  , hv  d6    ; assign error: error (9);

     [end subscr in buffer: UV:= value; R:= description, hv c18]
c18: [end UV, R, RF - expression: UV:= value, hvn c18]
a28: pa  b22   t   b23     ;   UA:= address (UV);

     [end subscr in core: UA:= R:= address of value, hv c19]
     [end addr expr: UA:= address of value, hvn c19]
c19:                       ;   s:= last used;
a29h:ps (b20)  , gr  b21   ; expr return: save R:= R;
     it  s1    , pa  b20   ;   last used:=s+1; p:=sr part[s];
     pp (s)    V           ;   goto update and go;
a12 = i - 1                ; exit to label:
     ps (b22)              ;   s:= UA;
     arn s     , hh  p     ;   R:=stack[s]; goto update[p];

     [next in: ud c37, hs c37]
c37: lyn p 0   D           ;

     [next out: qq <char to output> , hs c39]
c39: sy (s)    , hv  s1    ;
     [c37 and c39 are used for all input and output and may be
      replaced by other instructions having the wanted effect]
```

```
[26.08.67]
[GIER Algol 4, RS, page 11]

[The actions below are loaded in one of two versions: one for
 arrays in buffer and another for arrays in core]

< e27 [buffer versions]

     [reserve array: arn length , hs c20]
c20: ac  p1    , pm  d12   ;   stack[sr+1]:= stack[sr+1] + R;
     cm  0     D        NZ ;   if R ≤0 ∨ R > 2↑14 − 1 then
a30: pt  d2−3  , hv  d6    ;     error(3);
     ck  15    , sc  b20   ;   last used word:= R := last used word −
     arn b20   , tk  11    ;    R × 2↑15; R:= bits with first as sign
     tk  −26   , sr  d13   ;    (10,25,R) − lover buf limit;
     hv  a30           LT  ;   if R<0 then error(3);
     ar  d14   , hv  s1    ;   R:=R + buf array word; go back;

d12: qq  −1.25             ; mask , used by 1c9 , c20 , and 1c22
c36:
d13: qq 0[x+lower buf.39]  ; set by init run
d14: qqb23t1[x+lowerbuf.39]; set by init run

     [exit func: ps p + number of formals + 2, hv c21]
c21: pm  p−1   , ud  a28   ;   UV:= stack [p−1]; UA:= addr(UV);

     [exit proc: ps p + number of formals + 2, hh c22]
c22h:gm  b23   , arn p1    ;   last used word:=
     mb  d12   , gr  b20   ;     stack[sr+1] ∧ 26 m; R:=0;
     hhn a29               ;   goto expr return;

     [assign to formal subscripted: ud formal , hs c24
                                    qq  w     , store op (b23)]
c24: pa  b22   t  b23      ;   UA:= addr (UV);
     gr  b21   , ud  s1    ;   save R:= R; execute (store op(UA));
     arn(s1)               ;   R:= description of buffer word;
     us  0     V       NZ  ;   if R = 0 then
     pt  d2−9  , hv  d6    ;     error (9)
     arn b21   , hv  s2    ;   else us 0; R:= save R; goback 1;

×    [core versions]

     [reserve array: srn length, hs c20;
                     arn last used, hv s2]
c20: arn b20   V       LT  ;   if R < 0 then error (3);
a30: pt  d2−3  , hv  d6    ;
     ck  10    , sr (s)    ;   R:= (last used shift 10) − length;
     hv  a30           LT  ;   if R < 0 then error (3);
     ck  −10   , ga  b20   ;   last used:= stack [p+1] := R shift − 10;
     ga  p1    , hh  c7    ;   goto test last used;
                           ;     comment ensure NA, therefore jump to c7;
```

```
         [core versions ...]

     [exit func: ps p + number of formals + 2, hv c21]
c21: pm  p−1   , ud  a28   ;   UV:= stack [p−1]; UA:= addr(UV);

     [exit proc: ps p + number of formals + 2, hh c22]
c22h:gm  b23   , hhn a29   ;   R := 0; go to expr return;

     [assign to formal subscripted: ud formal , hs  c24
                                    qq  w     , store op (b23)]
c24: gr  b21   , arn(s1)   ;   save R:= R; R:= description of actual;
     ga  b22   V       NZ  ;   if R = 0 then
     pt  d2−9  , hv  d6     ;     error (9)
     arn b21   , hh  s1     ;   R:= save R; go right back;

>    [end core versions]

d32:
c31: qq[× used in top of free];
d31:
c64: qq[×catalog], pt d2−10; d31h: spill entry:  error (10);
c27: [error: pt<base table − error number, hv/hs c27]
d6:  vk[×errgr],vk[×errtr] ; error: to core (error track, first place);
     lk (d17)  , ud  d17   ;   goto first place;
c29: [exit: hhn c29]
d17: hv1d1,vk960[×exitgr]  ; exit: to core (exit track, first place);
d18: vk [×exittr], hv 1d6  ;   goto first place;


     [SAMBA−sequenses]

<e40+1 , <−e40+1           ;   if number of units ≠ 0 then
×                          ;   begin
     [get filetable : R := unit.9; hs c66]
c66: ga r2     , tk −27    ;   paramword := 3×unit+ c56+8.19−7.39;
     ar d35    , gr  b47   ;
c67: bt[unit]  t −e40+511  ; if unit<1 ∨ unit > number of units
     il  0     , hr  s1    ;   then error(≠<param≠);
     pt  d2−17 , hs  d6    ;   transfer filetable; return;
```

```
     [transfer and calculate sum: R :=paramword1; M := paramword2;
                                                  hs c48; ]

c48: xr                LOC ;   if array∧ outrec then change RM;
     il        X       LTB ;   if array∨ inrec  then transfer in and change RM;
     us        X       LTA ;   if array∨ outrec then transferout and change RM;
     hv  s1            NPB ;   if −, sumcheck then return;
     xr                    ;   change RM;
     ga  b41   , gt  r1    ;   b41 := base transfer addr in core; save return,
     gs  b40   , psn[length];  R := 0; s:= number of words transferred;
     ps s−1    , ar(b41)   ; NEXTWORD: s:= s−1; R := R + core[base +s];
     bs  s     , hv r−1    ;   if s>0 then go to NEXTWORD;
b41: qq s[base], ac  b44   ;   sambasum := sambasum + R;
b40: hv  [save s]t  1      ;   return;


     [exit SAMBAproc  :   hv c40  or hv c46 ]

c40: arn b47   , us  0     ;   restore filetab;
c46: hh  rc5   , ud  c18   ;   if −, in error then go to EXIT ST PROC;
     pa  c46   t  c5 − c46 ;   UA := address(UV); in error := false;
     ps  p+2   , hh  c22   ;   s := stackref + 2; go to EXIT PROC;

     [get statusword :  hs c60]

c60: arn  d34  , is (b45)  ;   R := b46.9+1.19+0.39;
     il  s224  , il  0     ;   get statusword to buffer0(unit);
     arn b46   , hr  s1    ;   R := b46 := buffer[0]; return;

>                          ;   end samba sequenses;

     [constants]

c41: d10:                  ;
c42: qq 1.39               ;   1
c58: qq 10.39              ;  10
c43:                       ;
c44: qq −1 t 256           ;    .5
c53: qq  0 t 256           ;   1.0

     [variables]

c35:b20: qq[×lastused in core.9+last used in buffer.25] f ; last used
c55:     vy[select bits] 48     t  [mask] 768             ; select
c61: b14: qq               ; tracks transferred
c30: b22: qq               ; UA
c33:      qq               ; address
c54:      qq               ; char
```

```
<e40+1 , <−e40+1          ;   if number of units ǂ 0 then
×                         ;   begin
c45: b46: qq              ;   status word location

c82:      qq              ;   sambaerror;
c47: b44: qq d33.39       ;   sambasum; constant used by init run;
c81: b47: qq              ;   paramword for current filetables;
c73:      qq              ;   working cell; constant used by init run;
c74: b38: qq b23  t 8     ;      −      −       −      −   −   −   − ;
c79: b29: qq 8.39         ;      −      −       −      −   −   −   − ;
c80: b13: qq 4096.39      ;      −      −       −      −   −   −   − ;
i = c73 + e69             ;   reserve place for SAMBA record segment ;
>                         ;   end;


c50: b21: qq              ;   save R;
c34: b43: qq              ;   UV−4
c68:      qq              ;   UV−3
c70: d25: qq              ;   UV−2 ; number of errortracks.39.
                          ;   loaded by errortracks. Used by init run;
c72: d23: qq te 26        ;   UV−1 ; placelength.19 .  Used by init run;
c17: b23: qq              ;   UV
c56:      qq              ;   UV + 1 ; filetab[1];

<e40+1 , <−e40+1          ;   if number of units ǂ 0 then
×                         ;   begin

[1c56:]   qq              ;            filetab[2];
[2c56:]   qq              ;            filetab[3];
[3c56:]   qq              ;            filetab[4];
c69:      qq              ;            filetab[5];
[1c69:]   qq              ;            filetab[6];
[2c69:]   qq              ;            filetab[7];
[3c69:]                   ;            filetab[8];
c71: b45: qq  b23  t 1    ;   constant used by init run;
>                         ;   end;


<e40                      ;   if number of units > 0 then
                          ;   begin
          qq              ;   unit 1 error;
          qq              ;   unit 2 error;
          qq              ;   unit 3 error;
          qq              ;   unit 4 error;
          qq              ;   unit 5 error;
          qq              ;   unit 6 error;
          qq              ;   unit 7 error;
          qq              ;   unit 8 error;
i = c71 + e40 + 1         ;   delete superflous error−cells;
>                         ;   end;
```

```
<e40+1 , <−e40+1           ;   if number of units ≠ 0 then
×                          ;   begin

     [samba constants]

c75: d34: qq b46   t 1     ;
c76:      qq  1            ;
c77:      qq       t 1     ;
c78:      qq  1.25−1.39    ;   mask;
d35:      qq  c56.9+8.19−7.39; filetab base addr;

e60=e60+c60+c40+c46+c71+c72+c56+c71  ;
e60=−e60+c66+c67+c48+c45+c81+c80+c45 ; SAMBA check;

>                          ;   end;


c59=1023                   ;   output sum cell

e37:                       ;   control definition of top RS in core;
e19=c+c1+c7+c11+c13+c20+c24+c17 ; RS check entries
e19=e19+c1+c11+c20+c17          ; system
```

```
[Error tracks. Two tracks which are taken in to trackplace 1 and 2]
d27=k−e14 [define rel first err tr];
b k=d27+e14, i=1d1, c18            ; Error track 1, track place 1.

[Upon exit 1e1−5d1 will hold the saved registers and 6d1−9d1 err. inf:]

[1d1]gi  1d1           IRC ;  gi <indicator>   IRC
[2]  gs  2d1   , ud  d19   ;  gs <s> , ...
[3]  gr  3d1           IOA ;    < R >
[4]  gp  4d1   , arn d6    ;  gp <p> , ...
[5]  sy  64    , gm  5d1   ;    < M >
[6]  gt  6d1   , ps [errtr];  qq <error no> , qq
[7]  ga  9d1   , can s−959 ;  ga <error track> , ...
[8]  ps  −1    , it  1     ;    < from line >
[9]  vk [errgr], vk  s1    ;    < to line >
     lk  42d1  , vk (9d1)  ; Start error: 1d1: save registers; select(errunit);
                           ;  writecr; to core(error track 2, track place 2);
c1:  pp  −1    , arn pd2   ; Get error no: p:= −1;
     ck  10    , nc  0     ;   while part 2(table[table base+p]) ≠ 0 do p:= p+1;
     pp  p−1   , hh  c1    ;
     arn p1d2  , ar  r1    ;   comment p now holds − error no  (<0);
     gt  pd2   , xrn e26   ;   restore part 2 of error table word;
     bs  p11   , nt  p     ; Write error text:
     arn c3    , sy  58    ;
     sy  29    , cl  40    ;   write char(<LC>); write char(<red>);
c2:  cl  −6    , ck  −4    ;
     ca  63    , ar  r1    ;   write text(error text [if −p > max err no then
[1]  ga  r1    , nc  10    ;                              0 else −p]);
[1]  sy[charac], hvn c2    ;
     srn p     DX          ;   save error no;
     gm  6d1   , bs p−502  ;   if −p >max err no then
     cl  −30   , hsn c17   ;     write integer (≠p≠, −p);
< e40                      ;   if number of tape units > 0 then
     sy  0     , sy (3c69) ;     incorporate( write integer(≠dd≠, tape unit));
>                          ;
     sy  0     , hv  42d1  ;   write char (0); goto track place 2;

c3: terror ;;[n] error texts:
    tstack;; 1
    tmult; ; 2
< e40+1, < −e40+1          ;   if number of tape units = 0 then
    tarray;; 3             ;    incorporate(≠<array≠)
×                          ;    else
    tbuffer; ; 3           ;    incorporate(≠<buffer≠)
>
    tcase; ; 4
    tindex;; 5
    tln;   ; 6
    tsqrt; ; 7
    texp;  ; 8
    tformal;; 9
    tspill;;10

     qq[fill]
     qq[fill]
< e40                      ;   if number of tape units ≤ 0 then
×
     qq[fill]              ;    incorporate(<fill>);
>
```

[Error track 2, loaded frm 41d1 but executed from 42d1]

```
c4=41d1 [check load address]
c4:  arn b8    , pp (b1)   ; Find line numbers:
     qq        , ud  d16   ;
     nt  p−511 , bs  s−512 ;
     gt  r     , pp[ret.tr];   p:= if curr track ≥ std pr tr then
     sy  0     , ud  d15   ;     return track else curr track;
                           ;   write char (0);  s:= first track;
c5:  it  p−576 , bs  s−512 ; loop 1: if s > p−64 then
     ps  p−65  , pa  rc8   ;     begin fount:= t; s:= p−1 end
                           ;   else s:= s+64;
c6:  ps  s+64  , ud  a1    ; Get line number on track s:
     ud  1a1   , ud  1a1   ;
     ud  2a1   , ud  2a1   ;   to core (run track(s), track place 3);
     ud  3a1   , lk  83d1  ;
     vk (2a1)  , pmn 122d1 ;   Raddr:= line mod 1024;
     gp  7d1   , cln −10   ;   if Raddr < 512 then
c7:  qq  0     V  0    NO  ;     begin times 1024:= times1024+incr; incr:=0 end
     it  1                 ;   else incr:= 1;
     pt  rc7   , ck −10    ;   error track := p;
     ar (rc7)  DX          ;   M:= to line:= times 1024 × 1024 + Raddr;
     cln−20    , gm  9d1
c8:  bs  1     , hv rc5    ;   if −, found then goto loop 1;
c9:  bs  1     , gm  8d1   ;   if −, found 1 then from line:= M;
     hs  rc12              ;   write integer(⟨p⟩, M);
     can(rc9)  , hv  rc10  ;   if −, found 1 then
     ps  p     , sy   32   ;     begin write char(⟨−⟩);  s:= p;
     pa  rc9   , hh  rc6   ;       found 1:= t; goto Get line number end;

c10: pmn 7d1   X           ; Finish: write char (⟨,⟩); write char (0);
     sy  27    , sy  0     ;
     cl  10    , hs  rc12  ;   out integer (⟨p⟩, error track);
     ps  1d1   , arn 6d1   ;   s:= address of first information ;
     hh (d17)  D   41      ;   Raddr:= error no;
                           ;   exitplace:= exitplace + 41; goto exit;

c11:  m  10−3              ; constants for write integer
[1]      10                ;

c12: mln rc11  , pa  rc15  ; procedure write integer;
c13: ck  −10   , bs (rc16) ;
c14: pa  rc15  t  16       ;
c15: arn 0     D       LZ  ;
     ga  r1    , nc  0     ;
[1]  sy  0     , ud  rc14  ;
c16: ncn 0     t  −256     ;
     mln r1c11 , hv  rc13  ;
     hr  s1                ;

c17=c12+1                  ; define entry for write integer from error track 1
c18=81d1                   ; check load address
c18: qq [RS sum]           ;

e22=k−e14, e47=j           ; set load parameter
d29=e36−80                 ; define RS segment start
e [error tracks]           ;   for two error tracks (80 words)
```

[26.8.67]
[GIER Algol 4, RS, page 18]

[Finish loading of RS segment]

```
d28=e22−d26          ; number of RS tracks  (Init run + RScode + error tracks)
i=d25                ; load number of error tracks.39 in init run
qq e22.39−d27.39−1.39;

b k=e23, i=0         ; load RS segment word
i= 16e21             ;
qq d29.9 − 1.9 + e37.19 − d29.19 + 3.20 + d29.39 − 1.39 f ;
e                    ;
e [RS]               ;

d e49=2              ; set next tape number;
```

[after i follows STOPCODE, SUMCODE and a sum character]

```
ib  T1,L1

s
```

[Here follows STOPCODE and CLEARCODE]

```
<e49−1, <−e49+3, ×  ;   test tape number
i wrong tape
s
                    ;
>                   ;

d e48=1             ;   translator is in:=true
d e52=9             ;   version number

<e52−e50,           ;   if version number T2 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=e52           ;   define version number T2
d e63=3             ;   define version number T3
d e64=4             ;   define version number T4
d e65=5             ;   define version number T5
d e66=6             ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e52
>                   ;
```

[10.4.67]
[Gier Algol 4, pass 1, first page]

[CONTENTS OF TAPE:

	Page
1. Indicator usage, common address variables	2
2. Definitions of values (d−names)	3
3. Constants, Variables, Init pass 1, reservation lower case input table	4
4. Program, main block	
4.1. Central input	5
4.2. Central actions	6
4.3. Normal actions	8
5. Program, special actions	
5.1. Code, local block	11
5.2. Medium change and input drivers, local block	14
5.4. Layout, local block	20
5.5. String, local block	22
6. Error output, texts	25
7. Tables and Code for compound symbols, local block	26
7.1. Compound table	29
7.2. Input table lower case	31
7.3. Input table upper case	33
8. End tape pass 1	35

Special requirements: Between input table lower case [0| and input table upper case [0| must be at least 128 words as this is the range of an input symbol from reader.
Allowed input bytes are in the table flagged by an f−mark.
All input bytes outside tables are forbitten. Consequently no word outside tables from input table lower case [0| to input table lower case [127| and from input table upper case [0| to input table upper case [127| may be f−marked.
end contents .....]

[Drum block head, pass 1]

```
b k=e22+e14, i=e2−e47, a33, b29, c57, d97;
i=e2                                     ;

e2:  arn 81e13 t   1       ; Last part of forward pass track repeated
     ga  e13−4 V           ;   with 1e2 and 2e2 slightly changed.
     qq                    ;
     tk  10    , ga  e13−3 ;
     tk  10    , ga  e13−2 ;
     tk  10    , ga  e13−1 ;
e1:  arn e13−1 t   −4      ;
     hr  s1                ;
     arn 10e4  , ac (s1)   ;
     arn (s1)  , hv  e11   ; Last 4 words replaced by this one;
```

[1. INDICATOR USAGE, COMMON ADDRESS VARIABLES

[INDICATOR USAGE:
NZA = normal. Initialized to false.
LZB = comm ok. In strings: LOB = set case. Initialized to false.
LTA = Poff active, initialized in next seegm 3 from mode bits.
LTB = sum. Initialized to false. Set to true by SUMCODE and CLEAR CODE.
LPA = print. Initialized to true.
LPB = line test. Indicates that each line interval lines should be
      printed. Initialized in next segm 3 from mode bits.
LQA = lined. Indicates that the present character is underlined.
LQB = comp. Indicates that the present character is underlined or barred.
 RA Used internally in code
 RB   –   –   –   –

[b–NAMES COMMON TO WHOLE PASS 1, SPECIFYING ADDRESS VARIABLES:
 Form of description:
   Slipname: part no of word, variable name, initial value, comment.

b10: 1, char, – , last character input.
b10: 2, case, d50, address of lower or upper case input table
b11: 2, action63 , 0, set by copy to: c2(reader), 2c2(drum),or, a21(keys)
b12: 2, print medium, 16e4, during message set to 15e4
b13: 1, outCR, 0, after first begin set to d22, incode to d2 or d39.
b14: 1, case1, 0, set by UC to d53, by LC to 0.
b16: 1, sem value, d12, d1 from code to;
b17: 1, end level, 0, increased by 1 by begin, decreased by end.
b18: 1, colon value, d27, d16 from code to ;
b19: 1, comma value, d16, d27 from code to ;

[b20 – 4b20: Full word variables, see page 4.

b22: 2, exit address from store string, next string word, changed by
         finis pass 1.
b24: 1, const kind , – , used by output of layouts and strings.
b25: 1, str case , – , current case in strings LC = –1, UC = –3.
b26: 1, i copy, d50, stack pointer for copy.
b27: 1, search, set to abs track for search by init pass 1.
b28: 1, init medium, – , set to abs track for init medium by init pass 1.]

[2. DEFINITIONS OF OUTPUT VALUES AND STRING VALUES REFERRED TO
OUTSIDE TABLES.
Kinds: v: normal output; c: in code; s: string; cs: string in code

Kind Meaning]

d1 = 1008; v begin code
d2 = 76; c blind CR
d3 = 249; v code
d4 = 74; c :
d5 = 61; c _

[d6–d11, see tables, page 28]

d12 = 167; v ;
d13 = 172; v end
d14 = 1009; v end pass
d15 = 957; v <lined digit> − <digit>. 0 − 9 is output as 1014 − 1023.
d16 = 220; v ,
d17 = 14; scs _
d18 = 63; scs CR
d19 = 10; s ≠
d20 = 1011; v short string
d21 = 1012; v long string
d22 = 1010; v CR
d23 = 291; v)
d24 = 93; v begin
d25 = 287; v fat comma
d26 = 1013; v layout
d27 = 194; v :
d28 = 273; v =
d29 = 76; v −
d30 = 228; v :=
d31 = 280; v =>
d32 = 82; v −,
d33 = 98; v for
d34 = 176; v else
d35 = 184; v (
d36 = 32; c forbidden
d37 = 59; c t1
d38 = 60; c t2
d39 = 63; c CR
d40 = 15; s word end
d41 = 271; v <
d42 = 72; v +
d43 = 275; v >
d44 = 86; v goto go to go to
d47 = 62; c f
d48 = 77; c e

d53 = 128; special, UC1, used to test current case.

[3. CONSTANTS, VARIABLES, INIT PASS 1, RESERVATION LOWER CASE TABLE]

```
[CONSTANTS                    Name(s)                 Used in action
                                                      −  when more than once]
a30: qq 1.4 + 1.23 + 1.27  ; d increment[0]            c44
[1]  qq 1.23               ; b bit                     c44
[2]  qq 1.34               ; n bit                     c44
[3]  qq 1.29               ; sign bit                  c44
[4]  qq −1.7+1.23−1.39     ; mask used to copy         c40
[5]  qq 1e13   t  1.29 + 4 ; il−param for buf medium   c40
[6]  qq 1.4 + 1.23 + 1.33  ; d increment[6]            c44
[7]  qq 1.39               ; bit 39                    c40,c44,c45,c10
[8]  pmn(e1)   X   1       ; input instr, drum or keys c40
[9]  ga  b10               ;   −     −  ,   −  −    −  c40
[10] lyn b10   V           ;   −     −  , reader       c40
[11] hv  a3           LT   ;   −     −  ,   −          c40
[12] qq 1.3 + 1.37         ; d increment[12], bit 37   c44,c45
[13] qq  6.23              ; char pr word              c40

[Variables.  Full word variables and working areas overwrite the code
 for init pass 1 (see below).
        Name(s)            Used by
 b20:   word 0, spaceword  c40, c44, c45, c50
1b20:   word 1, layoutword c44, c45
2b20:   word 2             c45
3b20:   word 3             c45
d49: to 6d49:
        copy name[0−6|     c40
7d49: to d50−1:
        copy stack         c40
The copy stack starts at d50−1 and works towars d49. It is bounded by
d50 and 6d49 which both are f−marked.]

a31:                       ; Init pass 1:  normal:= commok:= sum:=
b20: pi  544   t   144     ;     lined:= comp:= false;  print:= true;
[1]  arn −2    , ga  b27   ;   set search and init medium track numbers;
[2]  tk  10    , ga  b28   ;
[3]  arn  7e4  , hs  c52   ;   new medium(first medium);
a32:                       ;     comment return with first character read;
d49: nc  2[b]  , hs  c1    ; read first begin: if Raddr ≠ tb then
                           ; start begin:  next 1;
     qq  5[e]  , hsn a33   ;   begin test (5);
     qq  7[g]  , hsn a33   ;   begin test (7);
     qq  9[i]  , hsn a33   ;   begin test (9);
     qq 14[n]  , hsn a33   ;   begin test (14);
     pa  b13   t   d22 IQC ;   comp:= lined:= false;  outCR:= vCR;
[7]  pif 0     t   991 NPB ;   if −,linetest then print:= false;
     pa  14e4  , hv  c28   ;   string rel:= 0;  goto begin;

a33: hs   c1          LQA ; begin test:  if lined then next 1;
     hh  a32          NQA ;   if −,lined then goto start begin;
     nc (s)    , hv  a32   ;   if  Raddr ≠ (test value)  then goto
     arn a33   , hr  s1    ;    read first begin;  Rmarks:= 0;  return;

     qq [first medium is stacked here];

d50:       ; Input table lower case[0].  Limit copy pointer + 1.
i = 65i    ; reserve input table lower case [0:64], loaded later.

d52:       ; Input table lower case[65], secondary table words.
i = 3i     ; reserve secondary tablewords.
```

```
[4.1 CENTRAL INPUT]

c1:  pm  r     V       IQC ; next 1: comp:= lined:= f;
                , ps  i     ;   comment ps i will ensure return to
                            ;   next as the outermost return level;

[Input instructions: The next 3 instructions depend on the current
 input medium as follows: They are set from copy as follows:

     reader input:      key input:         drum input:         buf medium:
c2:  lyn b10 V          pmn(e1) X 1        pmn(e1) X 1         pmn(e1) X 1
     hs irrelevant LA   hs  e2    LA       hs  a14  LA         hs  a16  LA
     hv  a3  LT         ga  b10            ga  b10             ga  b10
                                           note: a14 and a16 are local in c40]

c2:  zq [input instr 1]        ; next: Raddr:= char:=
     hs [next word action]  LA ;       next character from input
     zq [input instr 2]        ;   if reader input ∧ R < 0 theN
                               ;      goto test it

b2:  ac[sum 1] Dt  1  LTB  ;   if sum then sum 1:= sum 1 + Raddr + 1;
b10: pm  −1    X   d50     ;   char:= char + case; M:= R;
     [char]      [case]    ;   R:= input table[char];
     hv  a3    X       NB  ;   if −,markB then begin swap; goto test it end;
     hv  a6    X       LPA ;   if print then begin swap; goto print it end;
                           ; after print:
a2:  hv (b10)          LA  ;   if LA then goto central action [Raddr];
                           ; after central: if inpoff then goto next;
c3:  hr  s1            LZA ;   if −,normal then return;
     [modified by pon,poff]; in normal:
c4:  hs  c50           LQB ;   if comp then compound;
                           ; treat normal:
c5:  ga  b1    V       LT  ;   if R ≥ 0 then
                           ; normal out:
c6:  mb  1023  DV          ;   begin Raddr:= part 1(R); normal:= t;
b1:  hv  −1    t   c   IZA ;   commok:= f; goto output end;
     [normal action]       ; normal act: normal:= t;
     hv  e3            IZC ;   goto normal action (R addr);
c7:  hsn c1            IZA ; comp or simple:
     hv  c50           LQB ;   next 1; if comp then goto compound;
     hr  s1                ;   return;
```

[4.1. cont. and normal action for SPACE]

```
a3:  ck  4     , nc  63.5  ; test it: if bits(4,9,R) ≠ 63 then
c9:  hs  c49               ; forbidden: mess 1(≠<character≠, 0, 1);
c19: hv  c2    , qq  sd54  ; space: blind: goto next;

a6:  ga  b4    X           ; print it:
b12: ck  −5    , ud  16e4  ;   begin select(print medium);
           [print medium]  ;   if −,no printing then out char(Raddr);
b4:  sy  −1            NO  ;   select (normal medium);
     arn(b10)  , ud  16e4  ;   swop;  goto after print
                           ;   end;
b11: hv a2, hv [action 63] ; action 63: goto
    [set by start medium   ;   if −, input from keys then next
     and input from keys]  ;   else if during input from keys then new key line
                           ;   else finis;
```

[4.2. CENTRAL ACTIONS]

```
c10:
b13: pmn[outCR]DX          ; CR: if out CR = 0 then goto next;
     hv  c2            LZ  ;     comment pre− or post−lude;
     hs  e3                ;   output(outCR);
     arn 7a30  , ac  e4    ;   CRcount:= CRcount + 1, print:= f;
     pm  e4            IPA ;   if linetest ∧
     dln 3e4   X       LPB ;      CRcount mod line interval = 0 then
     hv  a7            NZ  ;        begin
     sy  58    , pmn e4    ;        outchar (58); print:= t;
     hs  e9                ;        newline;
     hs  e8                ;        print1(CRcount); comment LA on return;
b14: bs  0     , sy  60    ;        if case 1 > 0 then
       [case 1]            ;          outchar(60);
     sy  0             IPA ;        outchar(0);
                           ;        end;
a7:  pm d18    DV      LZA ;   if −, normal then M:= s CR
     hv  c2            NQB ;   else if −, comp then goto next;
     arn 2d52  , hv  c3    ;   R:= table [second CR word]; goto after central;
```

[4.2. cont.]

```
c11: it  d53   , pa  b14   ; UC: case:= baseUC; case1:= UC1; goto next;
     gt  b10   , hv  c2    ; LC: case:= baseLC; case1:= LC1; goto next;

c12: arn d52   V       LQB ; Line:
     hv  c2            IQC ;   if comp then R:= table[second lineword]
     hv  c3            NQA ;   else begin comp:= lined:= t; goto next end;
a8:  hv  c2[used by poff]  ;   goto if lined then next else after central;
[+1] hr s1[used by pon]LZA ;

c13: arn 1d52  V       LQA ; bar: if −, lined then
     hv  c2            IQB ;   begin comp:= t; goto next end;
     hv  c3                ;   R:= table[second bar word];
                           ;   goto after central;

c14: hv  c2            NTA ; poff:  if −, Poff active then goto next;
     pm  a8    , gm c3     ;
     hs  c49               ;   action after central:= goto next
     hv  c2    , qq  pd55  ;   mess 1 (ǂ<off>, 0, 3); goto next;

c15: hv  c2            NTA ; pon:  if −,Poff active then goto next;
     pm  1a8   , gm  c3    ;   action after central:= conditional return;
     hs  c49           IQC ;   mess1 (ǂ<on>, 0, 3);
     hv  c2    , qq  pd56  ;   comp:= lined:= false;  goto next;

c16: arn(b2)   D t  −62    ; Sum code:  sum:= sum − 62; comment the sum code;
     ck  −5    , ar  b2    ;   sumcheck:= (sum : 32 + sum) mod 32
     mb  31    D           ;
     ga  b6    , pm  1c2   ;   store second input instruction below;
     gm  r1    , ud  c2    ;   execute first input instruction;
     qq[sec. input instr]  ;   second input instruction;
b6:  nc  −1    t   31      ;    comment next character now in Raddr;
     [sum check]           ;    if Raddr ǂ sum check + 31 then
     hs  c49               ;      mess1(ǂ<sum>, 0, 3);
c17: pa  b2    , qq  pd68  ; clear code: sum 1:= 0;
     pi  1.3   t  959      ;   sum:= t;
     hv  c2                ;   goto next;
```

```
[4.3. NORMAL ACTIONS]      ; comment Note the return to s in
                           ;   read comment;
c18: arn r     , ud  15e4  ; message: print medium:= alarm print;
     gt  b12   , sy  64    ;   select(alarm medium); outchar(CR);
     hs  a9            IPA ;   print:= t; read comment;
     arn r2    , ud  15e4  ;   select(alarm);
     sy  58    , ud  16e4  ;   outchar(LC); select(normal); print:= f;
[2]  pt  b12   Vt  16e4IPA ;   printmedium:= normal; goto fin comm;

c20: hs  a9                ; comment: read comment;
     hr  c1            IZA ; fin comm: normal:= t; return to next 1;
a9:  hs  c49           NZB ; read comment: if −, commok then
     hv  r1    , qq  sd57  ;   mess1 (ǂ<commentǂ, 0, 1);
     hsn c1            IZC ;   commok:= t; normal:= f; next 1;
     nc  d8            NQB ;   if Raddr ǂ t; ∨ comp then
     pmn r     , hr  s     ;     begin set mark A; return same end;
     hr  s1                ;   return;

c21:
b16: pmn d12[v;] DX    IZB ; semicolon: commok:= t; Raddr:= sem value;
     [sem value]           ;   if Raddr = vbegcode then goto code 1;
     ca  d1    , hv  c31   ;   normal:= t;
     hv  e3            IZA ;   goto output;

a10: hsn c1            IZA ; rep end comm: normal:= f; next 1;

a11: hs  c50           LQB ; test sem: if comp then compound;
     ca  d8    , hv  c21   ;   if Raddr = t; then goto semicolon;
     nc  d34   , ca  d33   ;   if Raddr = tfor ∨ Raddr = telse then
     hv  c5                ;     goto treat normal;
     nc  d7    , hv  a10   ;   if Raddr ǂ tend then goto rep end comm;

c22: pmn d13   DX          ; end: comm ok:= f;
     hs  e3            IZB ;   output(vend);
b17: bt   −1    t   −1     ;   endlevel:= endlevel − 1;
     [endlevel]            ;   if endlevel ≥ 0 then
     hv  a10               ;     goto rep end comm;

     pmn(14e4) D       IZA ; finis pass 1: M:= str rel; normal:= f;
     pt  b22   t   i       ;   set return from string track out to here
     pp  0                 ;   ensure return;
     bs (14e4) , hv  c51   ;   if str rel > 0 then goto string track out;
     arn(13e4) D           ;   inf 1:= str rel track;
     ga  2e4   , gm  3e4   ;   inf 2:= M;
     pm  8e4   , gm  6e4   ;   last track:= string track;
     pm  d14   DX          ;   output(vendpass);
     pm  5e4   , hs  e3    ;   input track:= first track
     gm  8e4   , hs  c53   ;   start medium:= pack medium;
     gr  7e4   , hhn e29   ;   go to end pass;
```

[4.3. cont.]

```
c23: hs  c7                ; colon: comp or simple;
     ca d9 [t=], hvn a15   ;   if Raddr = t = then goto colon equal;
b18: pmd27[v:] DXV         ;   swap; Raddr:= colon value;
     [in code heading d16] ;     goto double out;
a12: pm d28[v=]DX          ; equal1: swap; Raddr:= v = ;
a13: hs  e3            IZC ; double out: normal:= t; commok:= f;
     hv  c4    X           ;   output(Raddr); swap; goto in normal;

c24: hs  c7                ; equal: comp or simple;
     nc d43[t>], hv  a12   ;   if Raddr ≠ t> then goto equal 1;
     pm d31[v=>] DXV       ;   Raddr:= v =>  goto combined out;
c25:                       ;
b19: pm d16[v,]  DVX       ; comma: Raddr:= comma value
     [in code heading d27] ;   goto combined out;
a14: pmd25[vfat,]DXV  LZ   ; fat com: Raddr:= v fat, ; goto combined out;
a15: pm d30[v:=] DXV  LZ   ; colon eq: Raddr:= v:= ; goto combined out;
a16: pm d32[v−,] DX   LZ   ; negate: Raddr:= v −, ;
     hv  e3           IZC  ; combined out: normal:= t; commok:= f;
                           ;   goto output;

c26: hs  c7                ; minus: comp or single:
     ca d10[t,], hvn a16   ;   if Raddr = t ,  then goto negate;
     pm d29[v−]DXV         ;   swap; Raddr:= v − ;  goto double out;

a17: pm d23[v)]DX          ; right parent 1:
     hv  a13               ;   swap; Raddr:= v )  ;  goto double out;

c27: hsn c2              IZA ; right parent: normal:= f; next;
a18: ck  −2              NQB ;   if −, letter or blink ∨ comp
     hh  a19             LO  ;     goto right parent 1;
     ck  2               NQB ;
a19: hv  a17   , ck  −2      ;   if −,letter then goto right parent;
     hv  c27             LO  ;
a21: hs  c2                  ; in fat: next;
     nc  d6[t:], ud  a18     ;   if letter or blind ∧ −,comp then
     hv  a21             LO  ;       goto in fat;
     nc  d6[t:], hv  a23     ;   if Raddr ≠ t:  ∨ comp then goto fat error;
a22: hs  c2                  ; end fat: next;
     ca d35[t(]              ;   if Raddr = t ( ∧ −,comp then
     hvn a14             NQB ;     goto fat com;
     ck  −4              NQB ;   if blind ∧ −, comp then
     hv  a22             LO  ;     goto end fat;


a23: hs  c49                 ; fat error:
     pm (b10)  , qq  sd58    ;   mess1(≠<)<improper>≠, 0, 1); R:= table[char];
     hv  a11   X             ;   goto test sem;
```

[4.3. cont.]

```
c28: qqn(b17)  t   1   IZB ; begin: end level:= endlevel + 1;
     arn d24   D       IZA ;   Raddr:= v begin ; commok:= normal:= t;
     ps  c2 −1 , hv  e3    ;   set return (next); goto output;


c29: arn d15   D           ; lined digit: Raddr:= lined digit diff +
     ar (b10)  , hv  c6    ;     table char;  goto normal out;
```

[5.1. cont.]

```
b a10, b2;                ; begin block for code

c30: pm  d3    DX         ; code: Raddr:= v code;
     pa  b16   t   d1     ;   sem value:= v begcode;
     pp  d27   , it  d16  ;   p:= v:  ;  colon value:= v,  ;
                          ;   goto code beg end;
a1:  pa  b18   t   d27 IQC ; code end: colon value:= v: ; comp:= lined:= f;
     gp  b19   , hv  c6   ; code beg end: comma value:= p;
                          ;   goto normal out;

c31: pm  e4    , gm  b2    ; code 1: code CR count:= CR count;
a2:  pi1.1+1.8+0.9 V 764   ; end code text: first on line:= t; inbracket:= f;
                          ;   code comm:= t; goto set blind CR;
c36: pi 1.1+0.8t  765  NRB ; code CR: if −, in bracket then
                          ;     begin first on line:= t; code comm:= f end;
     pa  b13   V   d2     ; set blind CR: out CR:= cv blind CR;
                          ;   goto code byte out;
c37: pi  1.8   t   1021    ; code sem: code comm:= t;

c32: hs  e3           NZ   ; code byte out: if R ≠ 0 then output;
c33: hsn c1           IZA  ; next code: normal:= f; next1;
     hv  a5           LQB  ;   if comp then goto code comp;
a3:  ck  20    , ga  b1    ; after code comp: code byte:= part 3(R);
b1:  hvn −1    t   c   LO   ;   if code byte < 0 then
     [code byte]           ;     begin R:= 0; goto code action[codebyte] end;
     hv  c33          LRA  ;   if code comm then goto next code;
     arn(b1)   DV     IZB  ;   Raddr:= code byte; first on line:= f
                           ;   goto finis code byte;
c34: pi  0     t  1020 LRB  ; code right bracket:
                           ;   if in bracket then code comm:= in bracket:= f;
a10: pa  b13   t  d39  NZB  ; finis code byte: if −, first on line then
     hv  c32               ;   out CR:= v CRcode; goto code byte out
c38: arn d4    D           ; code colon: Raddr:= cv;
                           ; code left bracket:
c35: hv  c33          LRA  ;   if code comm then goto next code;
     pi 1.8+1.9t  1020 LZ  ;   if R = 0 then code comm:= in bracket:= t;
     hv  c36               ;   goto code CR;
```

[5.1. cont.]

```
                              ; code comp:
a5:  hv  a3            LRA    ;   if code comm then goto after code comp;
     hv  a6            NQA    ;   if −, lined then goto code comp err;
     hv  a3            NZB    ;   if −, first on line then
                              ;     goto after code comp;
     ca  4 [td],  hv c33      ;   if Raddr = td then goto next code;
     ca  5 [te], hv  a7       ;   if Raddr = te then goto finis code;
     ca 20 [tt], hv  a8       ;   if Raddr = tt then goto codetext;
     ca  6 [tf], psn d47      ;   if Raddr = tf then Raddr:= c f
     ca 13 [tm], psn d5       ;   else if Raddr = tm then Raddr:= c m
     pm  s     DXV     LZ     ;   else
a6:  pm  d36   DX             ; code comp err: Raddr:= c forbidden;
     hh  a3                   ;   R := R shift −20; goto after code comp;

a7:  pm  d48   DX             ; finis code:
     hs  e3                   ;   output(c e);
     pan b16   X   d12        ;   sem value:= v; ; M:= 0
     arn e4    , sr  b2       ;   MR pos 48:= CR count − code CR count;
     tl  −9    , pp  d16      ;   p:= v, ;
     hs  e5            NZ     ;   if R ≠ 0 then mess (≮<code≯, 2, 0);
     ps  c2−1  , qq  nd59     ;   set return (next);
     pa  b13   X   d22        ;   out CR:= v CR;
     mt  4a30  , hv  a1       ;   Raddr:= − MR pos 48; goto code end;
b2:  qq [code CR count]       ;

a8:  qq  c55   , hs  c54      ; code text: init str (test code char);
     bs  p−41  , it  d38      ; after code str word: const kind:=
     pa  b24   t   d37        ;      if last word then cv string else cv last str;
     tl  −6    , hs  c56      ;   output 5 of code;
     bs  p−41  , hvn a2       ;   if last word then goto end code text;
     ps  a8    , hv  c48      ;   set return (after code str word);
                              ;   goto get str word (test code char);

e                             ; end block for code
```

[5.2. MEDIUM CHANGE AND INPUT DRIVERS]

```
b a27, b9, d1              ; begin

c40: arn 8e4   , hs  e11   ; Copy:
     sk  42e13 , vk  960   ;   to drum (string stack, inbuf 2);
b27: can[search], hh  a2   ;   if search = 0 then goto copy err 1;
     vk (b27)  , lk  42e13 ;   from drum (search, inbuf 2);
     pa  b1    t   d49     ;   i name:= i name 0;
     gi  b2    , vk  960   ;   save in:= in;  wait drum;
     qq  c57   , hs  c54   ;   R:= init str (test copy char);
b1:  gr [iname], arn b1    ; rep name: store[i name]:= R;
     bs  p-41  , hv  a1    ;   if lastword then goto search name;
     ca  6d49  , hv  a3    ;   if i name = iname 0 + 6 then goto copy err 2;
     qq  c57   , hsn c48   ;   R:= get str word (test copy char);
     hv (b1)   D   1       ;   i name:= i name + 1; goto rep name;

a1:  pa  a10   , hs  c53   ; search name:  pack medium;
     pa  56e13 t  1e13     ;   place 0 in look up:= inbuf 1;
     qq (46e13)t   2       ;   free is treated as normal area;
     pp  d49-1 , hs  46e13 ;   p:= i name 0 - 1; search;
a2h: nc  0     , hs  e5    ;   if Raddr ≠ 0 then
b2:  pi [save in], qqn d66 ; copy err 1: mess(≠<copy≠, 2, 0)
     arn 8e4   , hs  e11   ;   in:= save in; R:= areaword
     arn 44e13 , lk  42e13 ;   from drum (string stack , inbuf 2);
     vk (e17)  , ps  c2-1  ;   wait drum; set return (next);
     mb  4a30          NT  ;   if R > 0 then clear bits(8,23,R);

c52:                       ;
b26: gr d50 [i copy] t -1  ; new medium:  stack[i copy-1]:= R;
     gs  b7    , it  0     ;   i copy:= i copy -2;  save s := s;

c41: pmn(b26)  X t 1       ; finis:  i copy:= i copy + 1;  R:= stack[i copy];
     tl  -7    , ga  b3    ;   kind:= bits(0, 2, R);
     tl  -25   , it (b3)   ;   if kind < -1 then goto copy err 1;
     bs  -1    , hh  a2    ;   char no:=
     tln 16    , tk  16    ;     bits(8,23,R) pos 23;
     gr  b20   , tln  16   ;   M:= bits (24,39,R);
b3:  pm [kind] XV t a25 NB ; start medium:  if LB then
a3:  ps  a4    , hv  e5    ; copy err 2: mess(≠<stack≠, 2, 0);
     ga  1c2   , gt  b11   ;   R:= descriptor[kind];
     gm  -2    , ck  20    ;   next word action:= part 1(R);
     ga  b4    , gt  b5    ;   action 63:= part 2(R);
a4=i-1                     ;   medium track:= M;  start action := part 4(R);
b4:  pm  -1    , qqn e34   ;   input instr 1 := instr [part 3(R)];
     gm  c2    , it  1     ;   input instr 2 := instr [part 3(R) + 1];
     pm (b4)   , gm  2c2   ;   M:= stack[i copy];  R:= 0;
b5h: pm (b26)  , hvn[start];   goto action [start];
```

[5.2. cont.]

```
c53: arn(b26)   , tl  −37   ; pack medium:  R:= stack[i copy];
     hhn a8             LT  ;   if kind part > 3 then return;
     arn(e2)    DVt−1e13LZ  ;   if kind part = 0 then
     arn 5a30   , hv  a6    ; pack drum medium:
     pm  −2     , gm (b26)  ;     begin stack[i copy]:= medium track;
     ck  10     , hv  a7    ;     R:=inaddr − inbuf1; comment wordno;
a6:  il  0      , arn 1e13  ;     end
     mb  4a30   , gr (b26)  ;   else pack buf medium:
     arn 4e13   , ar  −5    ;     begin stack[i copy]:= area word from buf 1 ∧
a7:  pm (e1)    DXt d1      ;       8 m 16 0 16 m;
     ck  −14    , ml  13a30 ;     R:= block length from buf 4 − left in buf end;
a8h: xr         , ar (b26)  ;   stack[i copy]:= R:= stack[i copy] +
     gr (b26)   , hr  s1    ;     (R × 6 + byte address − charbuf0 + 1) pos 23;
                            ;   return;
                            ; start medium:
a10: hh  r8     , vk  960   ;   if called by copy then begin
b28: can [init med.], hh a2 ;   if init medium = 0 then goto copy err 1;
     vk (b28)   , lk  1e13  ;   init medium 1 to inbuf 1;
     arn 8e4    , hs  e11   ;   to drum(string track, inbuf 1);
[s−1]sk  42e13  , arn(b26)  ;   R:= stack[i copy];
     hs  1e13               ;   init medium; comment init medium 2
                            ;     to inbuf 2;
     hh  a2              NZ ;   if R ǂ 0 then goto copy err 1;
     arn 8e4    , hs  e11   ;   from drum(string track, inbuf 2);
[8]  lk  42e13  , pmn r     ;   end;  M:= not zero;  R:= 0;

a11: ar  −2     , gr  −2    ; get medium track:
     hs  e11                ;   medium track:= medium track + R;
     pa  e2     X t  e13    ;   from drum (medium track, inbuf 1);
     lk  1e13   , vk  (e17) ;   in address:= inbuf 1 − 1; wait drum;
     hh  a13            NZ  ;   if M ǂ 0 then goto skip to char;

     arn −2     , sr  5e4   ; check against input ouput overlap:
     hv  a15            LT  ;   if (medium track ≥ first track ∧
     arn  9e4   , sr  −2    ;       medium track ≤ output track) ∨
     hv  r4             NT  ;      (medium track ≥ stringtrack ∧
     arn  −2    , sr  8e4   ;       medium track ≤ last track)
     hv  a15            LT  ;   then
     arn  6e4   , sr  −2    ;     mess (ǂ<copy overlapǂ, 2, 0);
[4]  hs  e5             NT  ;
     hv  a15    , qqn d62   ;   goto get drum word;

a12:                        ; skip to char:
a13h:gr  b20    , hs (1c2)  ;   get next word;
     arn b20    , sr  13a30 ;   if char no − 6 pos 23 ≥ 0 then
     hv  a12            NT  ;     begin charno:= charno − 6; goto skip to char end;
     arn b20    , ck  14    ;   byte address:= byte address + char no;
     ac  e1     , arn(e1)   ;   s:= save s;  R:=store[byte address];
b7:  ps [save s], hv 1c2    ;   goto central input test if next word;
```

[5.2. cont.]

```
                          ; Next word from drum: M:= 0;
                          ;   if in address ≥ inbuf 1 + 39 then
a14: bsn (e2)  X t  39e13 ;     begin  R:= 1; goto get medium track  end;
     arn 7a30  , hv a11   ; get drum word:  in address:= in address + 1;
a15: pmn (e2)  V t  1     ;   R:= 0;  M:= store[in address];  goto common;
a16: hs  21e13            ; Next word from buf medium:  next buf word;
     pa  e1    V t   b8 LZ ;   if R ≠ 0 then
     ps  r1    , hv  e5    ; copy err 3: mess(≠<copy medium≠, 2, 0);
     arn  a19  , gr  a17   ;   set unpack instruction;
[2]  hvn  a18  , qqn  d67  ;   goto unpack;

b8:  qq  ; char buf 0:  coment a word containing
     qq  ;   6 characters is unpacked here by the
     qq  ;   instructions below
     qq  ;   The loop stops when the sixth character
     qq  ;   overwrites the instruction in a17;
a17: zq [see 3a16]        ;
     arn (e1)  , hr  s1   ; unpack: unpack R and replace 63 by 64;
a18: cln −6               ;   R:= store [byte address];  return;
     ca  63.5  , arn 7a30 ;
     ck  −4    , hv  a17  ;

a19: gr  b8−1  V t 1      ; unpack instruction.
d1 = −b8+1 [see 9c53]     ;
```

[10.4.67]
[Gier Algol 4, pass 1, page 18]

[5.2. cont.]

```
b a8, b3                  ; begin block for input from keys:

a20: hs  c49              ; start keys:
     hvn r1    , qq (pd63) ;   mess 1(ǂ<type inǂ, 2, 3); R:= 0;
     bs (b14)  , it  60    ;   set part 1 (save case and 3 blinds,
     pa  b3    t   58      ;     if case 1 > 0 then 60 else 58);

a21: gp  b1    , ud  14e4  ; new key line: save p:= p; select(typewriter);

a1:  pp  e13   , gp  e1    ; after no: inaddress:= byte address := basein;
     gp  e2    , pp  −119  ;   p:= −119; if R ǂ 0 then from new keyline;
     ppn −159  V       NZ  ;   begin p:= −159; R:= 0 end
     arn a7    , sy 64 [CR];   else begin writecr; R:= four case end;

a2:  ca  64    , hh  a6    ; pack key: if Raddr = 64 then goto end line;
     ck  10    , bs  p     ;   R:= R shift 10; if p > 0 then
     gr  pe13  , pp  p−159 ;     begin inbuf 1[p]:= R; p:= p − 159 end;
     ly  b2    , pm  a8    ;   no clear Raddr:= type char; M:= four 63;
     pp  p40   , bs  p474  ;   p:= p + 40; if p < 38 ∧ R ǂ four case then
     cm  a7    , hv  a2    ;     goto pack key;

     sy 29[RED], sy 17[<]  ; stop line: write char(w RED); writechar(w <);
a3:  lyn b2    , bs  p474  ; read stop inf: Raddr:= typechar;
     ca 54[f]  , hh  a4    ;   if p < 38 ∧ Raddr = wf then goto finis;
     nc 37[n]  , hv  a3    ;   if Raddr = wn then goto read stop inf;
     sy 38[o]  , sy 62[blk]; no: write char(wo); writechar (w BLACK);
                           ;   R:= 0; goto after no;
a4:  hvn a1    , sy 57[i]  ; finis: write char (wi); writechar(wn);
     sy 37[n]  , sy 57[i]  ;   writechar(wi); writechar(ws);
     sy 18[s]  , sy 62[blk];   write char (w BLACK);
     pt  b11   t   c41     ;   action 63:= finis;
     arn b3    , hh  a6    ;   R:= save case and 3 blinds; goto end line;

a5:  pp  p40   , tk  10    ; fill word: p:= p + 40; no clear R addr:= 63;
a6:  cl  −10   , ck  10    ; end line: R:= R shift 10;
     bs  p512  , hv  a5    ;   if p < 0 then goto fill word;

     gr  pe13  , gm p1e13  ;   inbuf 1[p]:= R; inbuf 1[p+1]:= M;
b1:  pp  −1    , ud  16e4  ;     comment give a 63−byte if no fill word;
     [save p]              ;   p:= save p; select (normal);
b2:  qq[for ly], hv  1c2   ;   goto next plus 1; comment LA;
                           ;     constants:
b3:  qq[save case]+127.19+127.29+127.39,; save case and 3 blinds
a7:  qq 58.9+60.19+58.29+60.39         ,; four case
a8:  qq 63.9+63.19+63.29+63.39         ,; four 63

e                         ; end block for input from keys
```

[5.2. cont.]

```
a23: tl  49    , ca  1     ; start ly medium:  if bits(10,19,M) = 1 then
     hv (b3)   D t   a26   ;   begin kind:= −2; goto  start medium end;
     arn 16e4  , ck  −23   ; start reader: normal input unit:=
     tk  23    , ar (b26)  ;   bits (17, 19, copy stack[i copy];
[−2] gt  16e4  , ud  16e4  ;
a24h:hv  c2    , ud  14e4  ; after pause: select(normal unit);
     lyn b10   , hh  r−2   ;   goto next;

c42: hs  c49               ; end code: mess 1(ǂ<pauseǂ, 2, 3);
     hh  a24   , qqn pd65  ;   select (type writer); lyn; goto after pause;
```

[Input medium descriptors.   Accessed through kind and unpacked innext medium.

```
Kind   Next word   63     Variable  Start
        action   action   instruct. action ]

[−2] qq  e2.9 +  a21.19 + 8a30.29 + a20.39 ; keys, accessed via kind −1
[−1] qq           c2.19 +10a30.29 + a23.39 ; reader
a25: qq a14.9 +   c2.19 + 8a30.29 + a11.39 ; drum
[1]  qq a16.9 +   c2.19 + 8a30.29 + a10.39 ; buffer medium
[2]  qq a16.9 +   c2.19 + 8a30.29 + a10.39 ; buffer medium
[3]  qq a16.9 +   c2.19 + 8a30.29 + a10.39 ; buffer medium

d a26= −1a25 [see 1a23];
e                      ; end block for medium change and input drivers;
```

[5.4. LAY OUT, comments on next page]

```
b a19, b9;                ; begin block for layout:
                          ; layout or string:
c44: qq  i+1   , hs  c54  ;   init str (here);
     ps  0     , ca d41[<] ; here: s:= 0; if −, comp ∧ Raddr = t< then
     hv  c45          NQB ;     goto string;
     pa  b24   X   d26     ;   const kind:= v layout; M:= 0;
     pan b1    X   −19     ;   position:= −19; space word:= 0;
     gm  b20   , hv  a13   ;   goto start layout;

a1:  ar  3a30             ; lined plus: R:= R + sign bit;
a2:  ar  3a30             ; plus:       R:= R + sign bit;
a3:  ar  3a30  , pp  p1   ; minus:      R:= R + sign bit; p:= p + 1;
     bs  s−6   , ck  −10  ;   if s > 6 then R:= R shifts − 10
     ac  1b20  , hv  a12  ;     comment exponent sign;
                          ;   layout word:= layout word + R;
                          ;   goto next layout;

a4:  pp  p7    , ps  6    ; point: p:= p + 7; s:= 6;
     qq (b1)   V   1      ;   position:= position + 1; goto clear check;
                          ; ten: p:= 13; s:= 12;
a5:  pp  13    , ps  12   ; clear check: clear part 1(layout);
     pa  1b20  , hv  a12  ;   goto next layout;

a6:  bs (b1)   , hv  a11  ; space: if position  > 0 then goto set alarm;
     arn 7a30  , ns (b1)  ;   space word:= space word + bit(19−position);
     ck  s−20  , ac  b20  ;   R:= 0;
     can p     , hv  a10  ;   if p = 0 then goto increment
     pp  p+1   , hv  a12  ;   p:= p + 1; goto next layout;

a7:  arn 2a30  V          ; n: R:= n bit; goto d;
a8:  srn 1a30  , is  s2   ; zero: R:= − b bit; p:= s + 5 goto commond;
a9:  pp  s3    , ar  sa30 ; d: p:= s + 3;
                          ; common d: R:= R + d increment [s];
a10: ac  1b20         IOA ; increment: layout word:= layout word + R;
b1:  qq  −1    V   1  NOA ;   if bit (0, layout word) = 0 then
     [position]           ;     position:= position + 1;
a11: pp  16               ; set alarm: p:= 16; comment error state;
a12: hsn c1           IZA ; next layout : normal:= f; next 1;
a13: hh  a14          NQB ; start layout: layout class:=
     pa  b2    t   10 NQA ;     if comp then
     nc  d43[>]V      NQA ;        (if −, lined ∧ Raddr = t > then 10
     ca  d42[+], it  9    ;         else if lined ∧ Raddr = t+ then 9
     pa  b2    , ca  d10[,];         else if lined ∧ Raddr = t, then 5
     pa  b2    t   5  LQA ;         else 0)
a14: hv  a15   , ck  −10  ;     else bits(30, 33, R);
     tk  −6    , ga  b2   ;
a15:                      ;   R:= layout table [layout class];
b2:  arn −1    t a17  IZA ;   layout act:= Raddr; normal:= t;
     [layout class]       ;   if bit (p + 10, R) = 1 then
     ga  b3    , ck  p10  ;     begin R:= 0; goto layout act end;
b3:  hvn[layout act]  LO  ;   if −,mark A then
     hv  a11          NA  ; not terminator: goto set alarm;
     pp  16    , hv  a16  ;   p:= 16; goto terminator;
```

[5.4. cont.]

```
a16: pa  1b20  , arn 1b20  ; terminator: clear part1(layout word);
     ac  b20   , tk  19    ;   space word:= space word + layout word;
     pp  16            LO  ;   if b overflow then p:= 16;
     bs  p−15  , hs  c49   ;   if p > 15 then mess1(ɬ<string>ʇ, 0, 1);
     hv  c46   , qq  sd69  ;   goto const out;
```

[During layout scanning p holds the current state (0−16) and s holds a
function of this state indicating before point (0), after point (6), or,
after $_{10}$ (12). s is used to set p and to reference the proper constant which
is added to the layout word for d, r, or, 0. Initially, s = p = 0
The actions are governed by a state table which contain one word for each
allowed layout symbol and a common word (word 0) for forbidden ones.
Each word consists of an action address in bit 0 to and check bits in
bit 10 to 26, corresponding to state 0 to 16, A one indicate that the
symbol is allowed in the corresponding sate.
The table is referenced via bits 30 to 33 of the main table which gives
the address relative to first word (slipname a17) or, for compound
symbols, from the code.

```
       action.  state bits.  symbol  allowed in states.      new states.
                                                              p    s]
a17: qq                    ; forbidden                        16   −
[1]  qq a2   t 1.0+1.13    ;   +      0,13                    p+1  −
[2]  qq a3   t 1.0+1.13    ;   −      0,13                    p+1  −
[3]  qq a4   t 53.5        ;   .      0,1,3,5                 p+7  6
[4]  qq a5   t 5.5+5.11    ;   10     3,5,9,11                13   12
[5]  qq a6   t 37.5+5.11   ;   ←      0,3,5,9,11              see code
[6]  qq a7   t 3.1         ;   n      0,1                     s+3  −
[7]  qq a8   t 15.6+15.12  ;   0      3,4,5,6,9,10,11,12      s+5  −
[8]  qq a9   t 217.7+231.15;   d    0,1,3,4,7,8,9,10,13,14,15 s+3  −
[9]  qq a1   t 1.0+1.13    ;   ±      0,13                    p+1  −
[10] qq a16,qq 5.5+163.16  ;   ʇ      3,5,9,11,15,16      terminates layout
                           ;     Note ,mark.
```

[Formation of layout: Bits 20 − 39 of the final layout is accumulated
in word 1, the layout word. Format:
dh−check.4 or s−check.3 + b overflow.19 + b.23 + h.27 + fn.29 + d.33 +
n.34 + s.37 + fe.39.
dh−check or s−check is incremented and tested for each d, zero, or n in
the layout input an gives alarm when d > 15 ∨ h > 15 ∨ s > 7
b overflow is tested when the layout is completed and gives alarm
if b > 15.
Bits 0 − 19 of the layout is accumulated in word 0, the spaceword,
and the final layout is formed here before it is output]

```
e                          ; end layout
```

[5.5. cont.]

```
b a10, b1                 ; begin block for string

c45: pa  b24   t   d20    ; string: const kind:= vshort string;
b21: pa[string level]Dt 1 ;   stringlevel:= 1;
     qq  a5    , hs  c48  ;   get str word (test str char);
     bs  p-41  , hh  c46  ;   if last wrd then goto end const

     pm (13e4) DX         ; set long string description:
     ck  10    , gr  b20  ;   word 0:= str rel track + (str rel shift 20);
     it (14e4) , pt  b20  ;   const kind:= vlong string;
     pa  b24   XV  d21    ;   go to store string:

a1:  qq  a5    , hs  c48  ; next string word: get str word(test str char);

     is (14e4) , gr s42e13 ; store string: str buf [str rel]:= M;
     arn(14e4) D    1     ;   str rel:= str rel + 1;
     nc  40    , hv  a2   ;   if str rel = 40 then
                          ;     begin string track out:
c51: arn 8e4   , hs  e11  ;     to drum (string track, str buf);
     sk  42e13 , pa  14e4 ;     str rel:= 0;
     srn 7a30  , ac  8e4  ;     string track:= string track - 1;
                          ;     available tracks:= available tracks - 1;
     arn 4e4   , sr  7a30 ;     if used tracks > available tracks then
     gr  4e4   , sr  1e4  ;
     qq (13e4) t   -1     ;         mess(‡<program too big‡);
     hs  e5          LT   ;     str rel track:= str rel track - 1;
     vk (e17)  , qqn e33  ;     wait track
a2:                       ;     end;
b22: bs p470   , hv  a1   ;   if -, last word then goto next string word;
               [see finis pass 1]
c46: arn b20   , hs  a10  ; const out: R:= word 0;
     ps  c2-1  , hv  c1   ; end const: output 5; set return (next)
                          ;   goto next 1;

c56: pt  b1    t   -1     ; output 5 of code: cut := -1;
a9:                       ; next of 5:
b1:  tl  10    , ck 0 [cut];  RM:= RM shift 10; R:= R shift cut;
a10:                      ; output 5:
b24: pm [const kind] DX   ;   swap; Raddr:= const kind;
     hs  e3               ;   output (Raddr); swap;
     ncn 0[i]  X   256    ;   i:= (i+1) mod 4; comment i starts at 0;
     ga  b24   , hv  a9   ;   if i ≠ 0 then
     pt  b1    , hv  e3   ;     begin const kind:= Raddr; goto next of 5 end
                          ;   cut:= 0; goto output;
```

[5.5. cont.]

```
c54: grn 1b20  , pp  0     ; init str: word 1:= p:= 0;
     pa  b25   t   −1  IZA ;   str case:= −1; normal := f;

c48: grn 2b20              ; get str word: word 2:= 0

a3:  hs  c1            LZA ; next str char: if −, normal then
     hv  (s)           LZA ;     begin next 1; goto char test [s] end;
a4:  arn d19   DX      IZA ; terminator: Maddr:= s⫞; R:= bit 37; normal:= t;
     arn 12a30 , hv  a7    ;   goto pack it;

a5:  hv  a7            NQB ; test str char: if −, comp then goto pack it;
     nc d43[>] , hh  a6    ;   if Raddr = t> ∧ −, lined then
     bt (b21)  t   1   NQA ;     begin stringlevel:= stringlevel + 1;
     hv  a4            NQA ;     if stringlevel > 1 then goto terminator end;

c57: hh  a6            LQB ; test copy char: if comp then goto comp char;
     ca d11[SP], hv  a3    ;   if Raddr = tSP then goto next str char;
     ca d41[<] , hv  a4    ;   if Raddr = t< then goto terminator;
     hv  a7                ;   goto pack it;

c55: hh  a6            LQB ; test code char: if comp then goto comp char;
     ca  d8[;] , hv  a4    ;   if Raddr = t; then goto terminator;
                           ;   goto pack it;

a6:  hv  a7    , ca d10[,] ; comp char: if Raddr = t, ∧ lined then
     hvn a7    X       LQA ;     begin R:= 0; swap; goto pack it end;
     gm  3b20  , ca d41[<] ;   word 3:= M; if Raddr = t< ∧ −, lined then
     hs  c43           NQA ;     string nest (⫞<⫞in string⫞, 0, 3);
     pm  12a30 , qq  pd60  ;   Maddr:= s ;
     arn d17   DX          ;   R:= if lined then bit 37 else bit 39;
     arn  7a30         NQA ;
     hs  c47               ;   pack char;
     arn(b10)  , pm  3b20  ;   M:= word 3; R:= table[char];

a7:  hs  c47               ; pack it: pack char;
     bs  p475  , hv  a3    ;   if p ≤ 36 then goto next str char;
     pm  2b20  , arn 1b20  ;   R:= word 1; word 1:= word 2;
     gm  1b20  , bs  p469  ;   if p ≤ 42 ∧ normal then last word;
     ar  d19.3 DV      NZA ;     R:= R + s⫞ shift 36
     ar  d40.3 DV          ;   else
     hr  s1                ;     begin R:= R + s wordend shift 36
     pp  p−36  , hr  s1    ;       p:= p − 36 end;
                           ;   return;

c47:
b25: ck  −1    X       IOB ; pack char: set case:= bit(40+str case, R) = 1;
     [str case]            ;   swap; if set case then
     pm (b25)  DX  61  LOB ;     begin swap; R:= 0; Raddr:=
                           ;     str case:= str case + 61 end;
a8:  ck  p−30  , bs  p481  ; pack after case: R:= R shift p − 30;
     ac  1b20  V           ;   if p ≤ 30 then word 1:= word 1 + R
     ck  4     , ac  2b20  ;   else word 2:= word 2 + (R shift 4);
                           ;   p:= p + 6;
     srn(b25)  DV −57  LOB ;   if −, set case then return;
     pp  p6    , hr  s1    ;   str case:= 57 − str case; swap;
     ga  b25   X       IZB ;   set case:= f; goto pack after case;
     pp  p6    , hv  a8    ;

e                          ; end block for string;
```

[6. MESS 1 AND LOADING OF TEXTS]

```
c43: qq (b21)  t  −1       ; string nest: stringlevel:= stringlevel − 1;
c49: arn 8e4           IPA ; mess 1: to drum (string track, str buf);
     hs  e11               ;   print:= f;
     sk  42e13 , hv  e5    ;   goto mess;



b k=e31,  i=0              ; load messages:
i=e32                      ;

d54: tcharacter;           ;
d55: toff;                 ;
d56: ton;                  ;
d57: tcomment;             ;
d58: t)<improper>.;        ;
d59: tcode;                ;
d60: t∤ in string;         ;
d61: tcompound;            ;
d62: tcopy overlap;        ;
d63: ttype in;             ;
d65: tpause;               ;
d66: tcopy;                ;
d67: tcopy medium;         ;
d68: tsum;                 ;
d69: tstring;              ;

e32=i                          ;
e                              ;
```

[7. TABLES.

The pass 1 input table consists of two areas:
Input table lower case, starting at d50 and
Input table upper case, starting at d51.
Each area holds in principle one word for each possible character input value from 0 to 127.; however, as all proper input characters have values from 0 to 64, only these values are represented explicitly in the tables, the rest beeing represented by words belonging to the code of pass 1. This is possible because the admissible tablewords all are flagged with an f−mark and no word of the code is f−marked.

Format of tablewords:
  3 different formats, characterized by the flagbits alone are used:

1. Not f−marked: Totally forbidden characters. The rest of the word may contain anything.

2. f−marked and comma−marked: The word describes a central action, i.e. an action which is performed by the central input mechanism independent of context; e.g. UC, LC, CR, _ |
   The action itself is specified by a jump instruction in the left half of the tableword.
   The word also contains the boolean no printing (see below).

3. f−marked and not comma−marked: The word contains 5 parts as follows:

   Part1.9
     Action (>511, relative to c) or output value (≤511) in normal mode, i.e. outside compound symbols, code, layouts, strings, and, comments. Part 1 is also used for recognization of specific characters in other modes.

   Part 2.19
     Compound table reference. When an underlined or barred character is met in normal mode, part 2 is used as index to a table which describes compound symbols with this first character.

   Part 3.29
     Action (>511, relative to c) or output value (≤511) in code.

   Part 4.33
     Reference to the action table for layouts.

   Part 5.39
     Part 5 contains six booleans (1=true):
     Skipped in compound error.34:
       Is skipped, even if not underlined, during the reading of the rest of a compound symbol.
     No printing.35:
       Some characters, e.g. TAB, are not printed.
     Blind in a contruction which may be a fat comma.36
     Requires lower case in strings.37
     Part of a fat comma, i.e. letters.38
     Requires upper case in strings.   ]

[7. cont.]

```
d51=i          ; Input table upper case[0].
c50=65i        ; reserve input table upper case[0:64].
i=d50          ; start block at input table lower case [0]

b  a15, b5                ; begin compound and input table.

[procedure compound; comment is called by hs c50 with comp = t and
 main table word in R.
 Exit:  hr s1 with: comp = lined = normal = f.
        R contains compound table word, or, in case of unidentified
        compound, main table word of first non-compound character.
        Possible error message has been given.]

i = c50                   ; begin
c50: ck  10    , ga  b2   ; comp index:= part 2(R);
     hv  a1           LQA ; if -, lined then
                          ;   begin comp index:= comp index + 1
     arn(b2)   t   1  IQB ;   R:= table[comp index]; comp:= f;
     pa  b2    t  d96 NA  ;   if -,LA then comp index:= undef bar comp;
                          ;   end;
a1:  arn(b2)   , hv  a6   ; R:= table[comp index]; goto comp action;

a2:  hv  a9           NQA ; check lined: if -, lined then
                          ;   goto comp undefined;
a3:  hsn c1           IZA ; next comp in: normal:= f; next 1;
     ga  b3    , arn(b2)  ;   test char:= Raddr; R:= table[comp index];
a4:  ck  20    , nc (b3)  ; compare next: R:= R shift 20;
                          ;   if Raddr ≠ test char then
a5:                       ; next comp word:
b2:  arn -1    t   1      ;   begin comp index:= comp index + 1;
    [comp index]          ;   R:= table comp index;
                          ;   end;
a6:  gt  r     , hv  -1   ; comp action: goto part 2(R)
                          ;
a7:  hsn c1           IZA ; comp in must agree: normal:= f; next 1;
     ga  b3    , arn(b2)  ;   test char:= Raddr; R:= table[comp index];
                          ;
a8:  ck  20               ; compare must agree: R:= R shift 20;
b3:  ca  -1    , hv  a6   ;   if Raddr = test char then goto comp action;
     [test char]          ;
a9:  hsn c1           IZA ; comp undefined: normal:= f; next 1;
     hv  a9           LQB ;   if comp then goto comp undefined;
a10: pa  b2    t   b20    ;   comp index:= addr of (word 0);
     gr  b20   , hv  a14  ;   word 0:= R; goto comp error;
```

[7. cont.]

```
a11: hsn a15           IZA ; comp rest 7: rest;
[1]  hsn a15           IZA ; comp rest 6: rest;
[2]  hsn a15           IZA ; comp rest 5: rest;
[3]  hsn a15           IZA ; comp rest 4: rest;
[4]  hsn a15           IZA ; comp rest 3: rest;
[5]  hsn a15           IZA ; comp rest 2: rest;
[6]  hsn a15           IZA ; comp rest 1: rest;
[7]  hv  a14           NQA ; comp rest 0: if −, lined then goto comp err;
a12: arn(b2)           IQC ; finis comp: R:= table[comp index];
a13:                       ;   lined:= comp:= f;
b4:  bs  1     , hr  s1    ; exit comp: if −, rest err then return;
     [rest err]            ;   comment > 0 = false;
a14: pa  b4    t   1       ;   rest err:= f;
     hs  c49               ; comp err: mess 1(⫞<compound⫟, 0, 1);
     hv  a12   , qq  sd61  ;   goto finis comp;
a15: hv  c1            LQA ; rest: normal:= f;
                           ;   if lined then goto next 1;
     arn(b10)  , ck  −6    ;   if bit(<letter>, table[char]) = 0 then
     hr  a14           NO  ;     return to comp err;
     pa  b4    , hv  c1    ;   comp alarm:= t; goto next 1;
                           ; end;
```

[Redifinition of actions in normal mode so that they are negative
 and relative to c = c44 + 1 (last normal action)]

```
c = c44+1,  c9= c9−c, c18=c18−c, c19=c19−c, c20=c20−c, c21=c21−c
c22=c22−c, c23=c23−c, c24=c24−c, c25=c25−c, c26=c26−c, c27=c27−c
c28=c28−c, c29=c29−c, c30=c30−c, c32=c32−c, c33=c33−c, c34=c34−c
c35=c35−c, c36=c36−c, c37=c37−c, c38=c38−c, c40=c40−c
c41=c41−c, c44=c44−c
```

[Definition of d names which are used to test table values which are
 actions in normal mode]

```
 d6=c23[:],  d7=c22[end],  d8=c21[;],  d9=c24[=]           ;
d10=c25[,], d11=c19[SP],  d45=c18[message],  d46=c2[next]−c;
```

[30.3.66]
[GIER Algol 4, pass 1, page 29]

[7.1. COMPOUND TABLE]

```
     [  OUTPUT    COMP.     COMP.     COMP.       COMP         OUTPUT
        VALUE     ACTION    CHAR      ACTION      SYMBOL       VALUE
        or        BEFORE    TO        WHEN                     WHEN
        NORMAL    CHAR      TEST      CHAR                     PART 1
        ACTION    TEST                FOUND                    IS
        NAME                                                   ACTION]

d:   qq d46.9 + a9.19                          ;  undefined lined, skip comp
[1]  qq d46.9 +a10.19                          ;       −      −  , single char
d75: qq 279.9 +a12.19                          ;  =
[1]  qq 276.9 +a13.19                        , ;  ≠
[2]  qq 272.9 +a12.19                          ;  ≤
[3]  qq c44.9 +a13.19                        , ;  ≮         d20, d21, d26
[4]  qq 277.9 +a14.19                          ;  △
[5]  qq 269.9 +a13.19                        , ;  ⍋
[6]  qq 274.9 +a12.19                          ;  ≥
[7]  qq 270.9 +a12.19                          ;  :

d76: qq c29.9 +a12.19                          ;  <digit>        d41

d77: qq 245.9 + a3.19 +   2[b].29 + 6a11.39    ;  abs
     qq 144.9 + a8.19 +  18[r].29 + 4a11.39    ;  array

d78: qq c28.9 + a3.19 +   5[e].29 + 4a11.39    ;  begin          d24
     qq 130.9 + a8.19 +  15[o].29 + 2a11.39    ;  boolean
d79: qq 130.9 + a7.19 +  15[o].29 + 2a11.39    ;  Boolean

d80: qq 256.9 + a3.19 +   1[a].29 + 5a11.39    ;  case
     qq         a8.19 +  15[o].29 +   a5.39    ;  co..
     qq c30.9 + a2.19 +   4[d].29 + 6a11.39    ;  code           d3
     qq c20.9 + a4.19 +  13[m].29 + 3a11.39    ;  comment        none
     qq c40.9 + a4.19 +  16[p].29 + 6a11.39    ;  copy           none
     qq 251.9 + a8.19 +  18[r].29 + 6a11.39    ;  core

d81: qq 243.9 + a3.19 +  15[o].29 + 7a11.39    ;  do
d82: qq d34.9 + a3.19 +  12[l].29 + 5a11.39    ;  else
     qq         a8.19 +  14[n].29 +   a5.39    ;  en..
     qq c22.9 + a2.19 +   4[d].29 + 7a11.39    ;  end            d13
     qq 262.9 + a8.19 +  20[t].29 + 4a11.39    ;  entier

d83: qq 266.9 + a3.19 +   1[a].29 + 4a11.39    ;  false
     qq d33.9 + a4.19 +  15[o].29 + 6a11.39    ;  for
     qq c41.9 + a8.19 +   9[i].29 + 4a11.39    ;  finis           none

d84: qq         a7.19 +  15[o].29 +   a5.39    ;  go..
     qq d44.9 + a2.19 +  20[t].29 + 6a11.39    ;  goto
     qq         a8.19 + c19[ ].29 +   a5.39    ;  go . . or go  . .
     qq d44.9 + a7.19 +  20[t].29 + 6a11.39    ;  go to or go to
```

[7.1. cont.]

```
     [  OUTPUT    COMP.     COMP.     COMP.       COMP         OUTPUT
        VALUE     ACTION    CHAR      ACTION      SYMBOL       VALUE
        or        BEFORE    TO        WHEN                     WHEN
        NORMAL    CHAR      TEST      CHAR                     PART 1
        ACTION    TEST                FOUND                    IS
        NAME                                                   ACTION]

d85: qq 106.9 + a3.19 +   6[f].29 + 7a11.39    ;  if
     qq 116.9 + a8.19 +  14[n].29 + 2a11.39    ;  integer

d86: qq 155.9 + a7.19 +   1[a].29 + 4a11.39    ;  label

d87: qq c18.9 + a3.19 +   5[e].29 + 2a11.39    ;  message        none
     qq 281.9 + a8.19 +  15[o].29 + 6a11.39    ;  mod

d88: qq 258.9 + a3.19 +   6[f].29 + 7a11.39    ;  of
     qq 109.9 + a8.19 +  23[w].29 + 6a11.39    ;  own

d89: qq 137.9 + a7.19 +  18[r].29 +  a11.39    ;  procedure

d90: qq 123.9 + a3.19 +   5[e].29 + 5a11.39    ;  real
     qq 260.9 + a8.19 +  15[o].29 + 4a11.39    ;  round

d91: qq 282.9 + a3.19 +   8[h].29 + 4a11.39    ;  shift
     qq 149.9 + a4.19 +  23[w].29 + 3a11.39    ;  switch
     qq         a8.19 +  20[t].29 +   a5.39    ;  st ..
     qq 196.9 + a2.19 +   5[e].29 + 6a11.39    ;  step
     qq 153.9 + a8.19 +  18[r].29 + 4a11.39    ;  string

d92: qq 231.9 + a3.19 +   8[h].29 + 5a11.39    ;  then
     qq 265.9 + a8.19 +  18[r].29 + 5a11.39    ;  true

d93: qq 198.9 + a7.19 +  14[n].29 + 4a11.39    ;  until

d94: qq 157.9 + a7.19 +   1[a].29 + 4a11.39    ;  value

d95: qq 200.9 + a7.19 +   8[h].29 + 4a11.39    ;  while

d96: qq d46.9 +a14.19                          ;  undef bar comp
                                               ;
d74:   ; DEFINE LAST INSTR PASS 1              ;
```

[23.11.67]
[Gier Algol 4, pass 1, page 31]

[7.2. INPUT TABLE LOWER CASE]
i=d50

```
 [ Normal  Comp.    Code     Layout  Skip in comperr  U   Input      Output
   output  table    output   table    No printing     n   value      value
   or      refer.   or       index     Blind in fat, d   un =       when
   action  d=err    action   0=err      LC in string e   unused     part 1
   If central                           Ok in fat,   f              is an
   action then ,                         UC in str.  .              action ]

qq c19.9 +  1d.19 + c33.29 + 5.33 + [fftftf=] 10.39   f ;  0  BLANK  none
qq  58.9 + d76.19 +  65.29 +        [ffftff=]  4.39   f ;  1  1
qq  59.9 + d76.19 +  66.29 +        [ffftff=]  4.39   f ;  2  2
qq  60.9 + d76.19 +  67.29 +        [ffftff=]  4.39   f ;  3  3
qq  61.9 + d76.19 +  68.29 +        [ffftff=]  4.39   f ;  4  4
qq  62.9 + d76.19 +  69.29 +        [ffftff=]  4.39   f ;  5  5
qq  63.9 + d76.19 +  70.29 +        [ffftff=]  4.39   f ;  6  6
qq  64.9 + d76.19 +  71.29 +        [ffftff=]  4.39   f ;  7  7
qq  65.9 + d76.19 +  72.29 +        [ffftff=]  4.39   f ;  8  8
qq  66.9 + d76.19 +  73.29 +        [ffftff=]  4.39   f ;  9  9
qq                                                    t ; 10  un
hv  c2         ,  qq                [-f----=]         f ; 11  STOP   none
hv c42         ,  qq                [-f----=]         f ; 12  END    none
qq  c9.9 +   d.19 +  32.29 +        [fffttf=]  6.39   f ; 13  aa     none
hv c12         ,  qq                [-f-t--=]  4.29   f ; 14  _      none
qq                                                    t ; 15  un
qq  57.9 + d76.19 +  64.29 + 7.33 + [ffftff=]  4.39   f ; 16  0
qq d41.9 +2d75.19 +  32.29 +        [ffftff=]  4.39   f ; 17  <
qq  19.9 + d91.19 +  41.29 +        [tfftff=] 38.39   f ; 18  s
qq  20.9 + d92.19 +  42.29 +        [tfftff=] 38.39   f ; 19  t
qq  21.9 + d93.19 +  43.29 +        [tfftff=] 38.39   f ; 20  u
qq  22.9 + d94.19 +  44.29 +        [tfftff=] 38.39   f ; 21  v
qq  23.9 + d95.19 +  45.29 +        [tfftff=] 38.39   f ; 22  w
qq  24.9 +   d.19 +  46.29 +        [tfftff=] 38.39   f ; 23  x
qq  25.9 +   d.19 +  47.29 +        [tfftff=] 38.39   f ; 24  y
qq  26.9 +   d.19 +  48.29 +        [tfftff=] 38.39   f ; 25  z
qq                                                    t ; 26  un
qq c25.9 +   d.19 +  30.29 +        [ffftff=]  4.39   f ; 27  ,      d16
hv c17         ,  qq                [-t----=] 16.29   f ; 28  CLEAR  none
qq                                                    t ; 29  RED
hv  c2         ,  qq                [-t----=] 16.29   f ; 30  TAB    none
hv c14         ,  qq                [-t----=] 16.29   f ; 31  P OFF  none
qq c26.9 +   d.19 +  28.29 + 2.33 + [ffftff=]  4.39   f ; 32  -      d29
```

[7.2. cont.]

```
 [ Normal  Comp.    Code     Layout  Skip in comperr  U   Input      Output
   output  table    output   table    No printing     n   value      value
   or      refer.   or       index     Blind in fat,  d   un =       when
   action  d=err    action   0=err      LC in string  e   unused     part 1
   If central                           Ok in fat,    f              is an
   action then ,                         UC in str.   .              action ]

qq  10.9 +   d.19 +  58.29 +        [tffttf=] 38.39   f ; 33  j
qq  11.9 +   d.19 +  33.29 +        [tffttf=] 38.39   f ; 34  k
qq  12.9 + d86.19 +  34.29 +        [tffttf=] 38.39   f ; 35  l
qq  13.9 + d87.19 +  35.29 +        [tffttf=] 38.39   f ; 36  m
qq  14.9 +   d.19 +  36.29 +        [tffttf=] 38.39   f ; 37  n
qq  15.9 + d88.19 +  37.29 +        [tffttf=] 38.39   f ; 38  o
qq  16.9 + d89.19 +  38.29 + 6.33 + [tffttf=] 38.39   f ; 39  p
qq  17.9 +   d.19 +  39.29 +        [tffttf=] 38.39   f ; 40  q
qq  18.9 + d90.19 +  40.29 +        [tffttf=] 38.39   f ; 41  r
qq                                                    t ; 42  un
qq  28.9 +   d.19 +  32.29 +        [tffttf=] 38.39   f ; 43  ø
hv c15         , qq                 [-t----=] 16.29   f ; 44  P ON   none
qq                                                    t ; 45  un
qq                                                    t ; 46  un
qq                                                    t ; 47  un
qq  27.9 +   d.19 +  32.29 +        [tffttf=] 38.39   f ; 48  æ
qq   1.9 + d77.19 +  49.29 +        [tffttf=] 38.39   f ; 49  a
qq   2.9 + d78.19 +  50.29 +        [tffttf=] 38.39   f ; 50  b
qq   3.9 + d80.19 +  51.29 +        [tffttf=] 38.39   f ; 51  c
qq   4.9 + d81.19 +  52.29 + 8.33 + [tffttf=] 38.39   f ; 52  d
qq   5.9 + d82.19 +  53.29 +        [tffttf=] 38.39   f ; 53  e
qq   6.9 + d83.19 +  54.29 +        [tffttf=] 38.39   f ; 54  f
qq   7.9 + d84.19 +  55.29 +        [tffttf=] 38.39   f ; 55  g
qq   8.9 +   d.19 +  56.29 +        [tffttf=] 38.39   f ; 56  h
qq   9.9 + d85.19 +  57.29 +        [tffttf=] 38.39   f ; 57  i
hv  c2         ,  qq                [-f----=]         f ; 58  LC     none
qq  67.9 +   d.19 +  27.29 + 3.33 + [ffftff=]  4.39   f ; 59  .
hv c11         ,  qq  d51           [-f----=]         f ; 60  UC     none
hv c16         ,  qq                [-t----=] 16.29   f ; 61  SUM    none
qq                                                    t ; 62  BLACK
hh b11         ,  qq                [-t----=] 16.29   f ; 63  TF     none
hv c10         ,  qq                [-f----=]         f ; 64  CR     (b13)
d52: [secondary table words, used by the actions]       ;
qq d46.9 +   d.19 +  32.29 +        [---t-f=]  4.39     ; 65  _      none
qq d46.9 +   d.19 +  32.29 +        [---f-t=]  1.39     ; 66  |      none
qq d46.9 +  1d.19 + c36.29 +        [f-tftf=] 10.39     ; 67  CR     none
```

[7.3. INPUT TABLE UPPER CASE]
i=d51

```
 [ Normal  Comp.    Code     Layout  Skip in comperr  U   Input      Output
   output  table    output   table    No printing     n   value      value
   or      refer.   or       index     Blind in fat,  d   un =       when
   action  d=err    action   0=err      LC in string  e   unused     part 1
   If central                           Ok in fat,    f              is an
   action then ,                         UC in str.   .              action ]

qq c19.9 +  1d.19 + c33.29 + 5.33 + [fftftf=] 10.39   f ;  0  BLANK  none
qq 278.9 +   d.19 +  32.29 +        [ffffft=]  1.39   f ;  1  ∨
qq 267.9 +   d.19 +  32.29 +        [ffffft=]  1.39   f ;  2  ×
qq 268.9 +   d.19 +  32.29 +        [ffffft=]  1.39   f ;  3  /
qq c24.9 + d75.19 +  75.29 +        [ffffft=]  1.39   f ;  4  =      d28
qq c21.9 +   d.19 + c37.29 +        [ffffft=]  1.39   f ;  5  ;      d12
qq 210.9 +   d.19 + c35.29 +        [ffffft=]  1.39   f ;  6  [
qq 202.9 +   d.19 + c34.29 +        [ffffft=]  1.39   f ;  7  ]
qq d35.9 +   d.19 +  31.29 +        [ffffft=]  1.39   f ;  8  (
qq c27.9 +   d.19 +  78.29 +        [ffffft=]  1.39   f ;  9  )      d23
qq                                                    t ; 10  un
hv c2          ,  qq                [−f−−−=]          f ; 11  STOP   none
hv c42         ,  qq                [−f−−−=]          f ; 12  END    none
qq  c9.9 +   d.19 +  32.29 +        [fffftt=]  3.39   f ; 13  AA     none
hv c13         ,  qq                [−f−−t=]  1.29    f ; 14  |      none
qq                                                    t ; 15  un
qq 277.9 +4d75.19 +  32.29 +        [ffffft=]  1.39   f ; 16  ∧
qq d43.9 +6d75.19 +  32.29 +        [ffffft=]  1.39   f ; 17  >
qq  47.9 +   d.19 +   9.29 +        [fffftt=]  3.39   f ; 18  S
qq  48.9 +   d.19 +  10.29 +        [fffftt=]  3.39   f ; 19  T
qq  49.9 +   d.19 +  11.29 +        [fffftt=]  3.39   f ; 20  U
qq  50.9 +   d.19 +  12.29 +        [fffftt=]  3.39   f ; 21  V
qq  51.9 +   d.19 +  13.29 +        [fffftt=]  3.39   f ; 22  W
qq  52.9 +   d.19 +  14.29 +        [fffftt=]  3.39   f ; 23  X
qq  53.9 +   d.19 +  15.29 +        [fffftt=]  3.39   f ; 24  Y
qq  54.9 +   d.19 +  16.29 +        [fffftt=]  3.39   f ; 25  Z
qq                                                    t ; 26  un
qq  68.9 +   d.19 +  32.29 + 4.33 + [ffffft=]  1.39   f ; 27  10
hv c17         ,  qq                [−t−−−=] 16.29    f ; 28  CLEAR  none
qq                                                    t ; 29  RED
hv  c2         ,  qq                [−t−−−=] 16.29    f ; 30  TAB    none
hv c14         ,  qq                [−t−−−=] 16.29    f ; 31  P OFF  none
qq d42.9 +   d.19 +  29.29 + 1.33 + [ffffft=]  1.39   f ; 32  +
```

[7.3. cont.]

```
 [ Normal  Comp.    Code     Layout  Skip in comperr  U   Input      Output
   output  table    output   table    No printing     n   value      value
   or      refer.   or       index     Blind in fat,  d   un =       when
   action  d=err    action   0=err      LC in string  e   unused     part 1
   If central                           Ok in fat,    f              is an
   action then ,                         UC in str.   .              action ]

qq  38.9 +   d.19 +  26.29 +        [ffffft=]  3.39   f ; 33  J
qq  39.9 +   d.19 +   1.29 +        [ffffft=]  3.39   f ; 34  K
qq  40.9 +   d.19 +   2.29 +        [ffffft=]  3.39   f ; 35  L
qq  41.9 +   d.19 +   3.29 +        [ffffft=]  3.39   f ; 36  M
qq  42.9 +   d.19 +   4.29 +        [ffffft=]  3.39   f ; 37  N
qq  43.9 +   d.19 +   5.29 +        [ffffft=]  3.39   f ; 38  O
qq  44.9 +   d.19 +   6.29 +        [ffffft=]  3.39   f ; 39  P
qq  45.9 +   d.19 +   7.29 +        [ffffft=]  3.39   f ; 40  Q
qq  46.9 +   d.19 +   8.29 +        [ffffft=]  3.39   f ; 41  R
qq                                                    t ; 42  un
qq  56.9 +   d.19 +  32.29 +        [ffffft=]  3.39   f ; 43  Ø
hv c15         ,  qq                [−t−−−−=] 16.29   f ; 44  P ON   none
qq                                                    t ; 45  un
qq                                                    t ; 46  un
qq                                                    t ; 47  un
qq  55.9 +   d.19 +  32.29 +        [ffffft=]  3.39   f ; 48  Æ
qq  29.9 +   d.19 +  17.29 +        [ffffft=]  3.39   f ; 49  A
qq  30.9 + d79.19 +  18.29 +        [ffffft=]  3.39   f ; 50  B
qq  31.9 +   d.19 +  19.29 +        [ffffft=]  3.39   f ; 51  C
qq  32.9 +   d.19 +  20.29 +        [ffffft=]  3.39   f ; 52  D
qq  33.9 +   d.19 +  21.29 +        [ffffft=]  3.39   f ; 53  E
qq  34.9 +   d.19 +  22.29 +        [ffffft=]  3.39   f ; 54  F
qq  35.9 +   d.19 +  23.29 +        [ffffft=]  3.39   f ; 55  G
qq  36.9 +   d.19 +  24.29 +        [ffffft=]  3.39   f ; 56  H
qq  37.9 +   d.19 +  25.29 +        [ffffft=]  3.39   f ; 57  I
hh c11         ,  qq  d50           [−f−−−−=]         f ; 58  LC     none
qq c23.9 +7d75.19 + c38.29 +        [ffffft=]  1.39   f ; 59  :      d27
hv  c2         ,  qq                [−f−−−−=]         f ; 60  UC     none
hv c16         ,  qq                [−t−−−−=] 16.29   f ; 61  SUM    none
qq                                                    t ; 62  BLACK
hh b11         ,  qq                [−t−−−−=] 16.29   f ; 63  TF     none
hv c10         ,  qq                [−f−−−−=]         f ; 64  CR     (b13)

e              ; end compound and tables;
```

```
[14.4.67]
[Gier algol 4, pass 1, page 35]

[8. END TAPE PASS 1]

d i=d74
qq [pass sum]

d e22=k−e14,e47=j       ; Definition of running pass track for next pass.

b k=e23,i=0             ; load pass 1 segment word;
d i=1e21                ;
qq e2.9+1d74.19−e2.19+1.20+1.24+a31.39 f  ;
e                                         ;

e [final end pass 1]                      ;

s
```

[31.10.66]

[GIER Algol 4, pass 2, page 1]

[Synopsis
Pass 2 recognises byte strings representing identifiers and substitutes a unique byte for each such string. This is done regardless of block structure so that the same identifier will be represented by the same byte throughout the text.
The values of the bytes will be in the range 1023 > <byte> > 511.

Tables.
Pass 2 uses two tables: first letter table[1:28| and table [first free after pass 2: top of store|;
Table holds, starting at top of store, one word (referred to as short word) for each distinct identifier. All words representing identifiers with the same first letter form a chain whose links are the address parts. The end of the chain has address part = 0. The starting point of each chain is given in the addresspart of the correspoding word in the letter table if the first letter is ≤ 28 else in the counting part of first letter table [first letter − 28|. A letter which does not start a chain is represented in the letter table by a reference to itself.

The rest of the identifier is assembled as an integer in a base 67 number system.
If this integer < 2↑30 it will be stored in the remaining 30 bits of the short word and bit 40 will be 1.
Otherwise the last 20 bits of the integer will be stored in bit 20 to 39 of short word and bit 40 will be 0. The rest of the integer is stored in one or more words (long words) starting at the first free word at the bottom of table and the address + 1 of the last of these words will be stored in bit 10 to 19 of short word. The first of these words will have bit 0 = 1, the others, if any, will have bit 0 = 0.
The two tables are kept in core after pass 2 for use by the std proc look up at the start of pass 3.

Table look up
Each identifier encounted by pass 2 is added to the table but the linking is not performed.
Pass 2 now compares each entry in the corresponding chain with the new identifier. If the identifier is found in the table the new identifier is removed from the table otherwise the necessary linking is performed.
Finally the address of the short word corresponding to the identifier is put out.

Program:]

```
b k=e22+e14, i=e16−e47, a34,b14,c5,d6; drum block head pass 2;
i=e16                                 ;

d d2=e20−1 ; first identifier value  ; initial i short;

b k=e31, i=0; Load texts
i=e32        ;
d3: tidentifier overflow; ;used by 1a12
d4: tpattern;           ;   −   −  1a26
e32=i       ;
e           ;

e16: [First letter table starts here. Firstr letter 1−28 uses part 1,
    29−56 uses part 2. From start part 1 and part 2 points at them
    selves (values < 512). First time an identifier with a given first
    letter is encountered the corresponding address in the first letter
    table is changed to point at the short word for that identifier
    thereby starting a new chain (values ≥ 512)]

qq         ; Final i short is stored here for use by the next segment
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
qq i, psn i;
```

```
                                ; constants. Used by
c1:  qq  1.19−1.39              ; mask long  5a4
c2:  qq  1.9−1.39 f             ; mask short a5
c3:  m            1             ;           a33
c4:  m          67a             ;           2b3
c5:  m          10              ;           a23
                                ; start pass 2:
a1:  pmn(e1)   X   1   IZC      ; next: fchar:= next byte; short:= ZB:= t;
     hs  e2            LA       ;   if f char > 512 then
     ga  b1                     ;     begin s:= fchar; goto special actions end;
b1:  bs [f char]Vt 56  NT       ;   if f char > 56 then
     ps (b1)   , hv  a34        ;     begin set return(next − 1);
     ps  a1−1  , hv  e3         ;     Raddr:= fchar; goto output end
b2:  ppn d1 [ilong],ps(b1)      ;   p:= ilong; s:= fchar; R:= 0;

a2:  pm (e1)   X   1            ; next char: swap;
     hs  e2            LA       ;   char:= Raddr:= next byte; finis:= LA:= t;
     ga  b3                     ;   if char > 66 then
b3:  bs [char] t   66  NT       ;     end identifier:
     hvn a4    X                ;     begin R:= 0; swap; goto ident finis end;
     ck  10    , ml  c4         ;   RM:= M × 67 + byte; finis:= LA:= f;
     hv  a2    X       LZ       ;   if R = 0 then begin swap; goto next char end;
b4:  bsp−510t d2−512[itest]     ; check room: if p−510 > itest then
     ps  a12   , hv  e5         ;     mess(‹<no room for identifiers›, 2, 0);
                                ;
                                ;   if −, short then store long word:
     gm  p     V       NZA      ;     begin table [p]:= M; p:= p + 1;
                                ;       swap; goto next char end;
a3:  cl  19    XV      IZA      ; store first long: short := ZC:= f;
     pp  p1    , hv  a2         ;   table[i short] mark ZC:= RM rem 2↑20;
     ck  −19   , ud  a6         ;   RM:= RM : 2↑20; swap;
     hv  a2            LA       ;   if −, finis then goto next char;
                                ;
                                ; finis ident:
a4:  pm  512   DXV     NZA      ;   if short then
     ca  0     , hh  a5         ;     begin if Raddr = 0 then goto store short;
     hv  a3    X       LZA      ;       swap; goto store first long end;
     gm  p     , ac (b2)        ;   table[p]:= R; table[ilong]:=
     pp  p1    , it  p          ;     table[ilong] + bit 0; p:= p + 1;
     pt (b6)   , arn c1         ;   set part 2 of (table [i short]) to: (p);
                                ;   R:= mask long; goto end ident;
a5:  hv  a7    , pm  c2         ; store short: table [ishort] mark ZC:= R;
a6:  gr (b6)   X       MZA      ;   R:= mask short;
       [executed from 2a3]      ;

a7:  pm (e1)   XD −1   IPB      ; end ident: set PB; reset input; swap;
```

[31.10.66]

[GIER Algol 4, pass 2, page 4]

```
[Now an identifier has been read and the situation is as follows:
 s = first char, range 1-56. p := address of first free word after long words.
 store [i short| = short word, part 1 = 0. R ǂ 0 (address of last byte).
 if short identifier: Marks[ishort| = 11, no long words stored.
p = ilong. PB = 1. M = mask short = 2↑30 - 1;
 if long identifier: marks [ishort| = 01 part 2 = p. store [ilong| =
 first long word, bit 0 = 1. store [p-1| = last long word, may be
 the same as first long word. PB = 0. M = mask long = 2↑20 - 1]

                          ; start look up:
     bs  s-28  , ud se16-28;   if s ≤ 28 then s:= part 1(first letter[s]) else
     ps (se16)         NZ  ;     begin R:= 0; s:= part 2 (first letter[s-28]) end;
     bs  s-511 , hh  a11   ;   if s ≥ 512 then goto look up;
                          ; first identifier in chain: chain:= s;
     gs  b5    , it (b6)   ;   comment address of first letter [f char].
     pt (b5)   V       LZ  ; connect to chain:
a8:  it (b6)   , pa (b5)   ;   if R = 0 then part 2[chain] = i short else
     pm (b6)   DX          ;not found: part 1 [chain]:= i short;
     ps (b6)   V   -1  LT  ;   Raddr:= i short; i short:= s:= i short -1 ;
     ps  a12   , hv  e5    ;   if Raddr < 512 then
     it s-512  , pt  b4    ;     mess(ǂ<no room for identifiersǂ, 2, 0);
     gp  b2    , ps  a1-1  ;   itest:= s-512; ilong:= p; set return(next);
                          ;   goto output;
a9:  hv  e3    , arn(b5)   ; after long failed: R:= table [chain]
a10: ga  b5    V       LT  ; next test: if Raddr < 512 then goto not found;
                          ;   chain:= Raddr; goto test it;
a11: hv  a8    , gs  b5    ; look up: chain:= s;

b5:  arn -1 [chain]    IPA ; testit: R:= set PA (table[chain]);
b6:  cm d2[ishort], hva10  ;   if R masked M ǂ table[ishort] masked M
                          ;     then goto next test;
     hv  a14           LPC ;   if LPC then goto found;
     gp  b7    V       NPC ;   if -, NPC goto next test;

d a12=i-1 [mess descr]     ; test rest of long: ilong new:= p;
     hv  a10   , qqn d3    ;   i long old:= part 2 (R);
     ck  10    , ga  b8    ;
a13:                       ; test next long: i long new:= i long new - 1
b7:  arn[ilong new]t-1 ITA ;   i long old:= i long old - 1;
b8:  sr [i long old]t-1    ;   if table [ilong new] ǂ table [ilongold] then
     hh  a9            NZ  ;     goto after long failed;
     hv  a13           NTA ;   if bit 0 (table[i long new]) = 0 then
                          ;     goto test next long;
a14: pm (b5)   DX          ; found: Raddr:= chain; set return (next)
     ps  a1-1  , hv  e3    ;   goto output;
```

```
[Patterns:
 This logic uses four states (as an address in s). The state table
 contains for each state four jump instructions packed in two
 words. Each jump instruction corresponds to an input byte
 class. The comments below describe for each state the new
 state and the action for each of the four input byte classes in
 the following order:                digit     lined digit
                                     letter m  other]

a15: pan b12   , ud  a21   ; pattern: pos:= word:= M:= R:= 0;
     grn(b6)   , hv  a24   ;   reset input; goto end term;

a16: arn(e1)   t   1   IOA ; get next: digit := next byte;
     hs  e2            LA  ;   copy:= f;
     ga  b13   , ca  13    ;   if digit = letter m then
     srn c3    , hv  s1    ;     goto m action [state];
     pp (b13)  , it  501   ;   if digit is normal digit then
     bs  p445  , hv  s     ;     goto digit action [state];
     bs  p512  t   501     ;   if digit is lined digit then
     arn b11   , hh  s     ;     goto lined digit action [state];
                           ;   goto other action [state];
a17h:hh  s1    , ns (b11)  ; mval: R:= 2↑bits − 1 shift 40 − bits;
     tk  s40   , hv  a23   ;   goto acc;

a18: tk  2     , ar  b11   ; mult bits:
     tk  1     , ga  b11   ;   bits:= bits × 10 + lined digit value;
     it  p10   , pp (b11)  ;   if bits > pos + 40 then state:= 0;
     it (b12)  , bs  p−40  ;   goto get next;
                           ; test: if R ≠ 0 then state:= 0; goto next;
a19: hs  a16           NZ  ;          state 0:    0, get next       0, get next
     hv  a16   , hs  e5    ;          in error    0, get next       alarm exit

a20: pm (b6)   , qq  sd4   ; alarm exit: mess (≠<pattern≠, 0, 1);
a21: qqn(e1)   Xt   −1     ; exit: reset; R:= word; goto output R;
     [exc. by a15, a22]
a22h:hv  a31   , ud  a21   ; reset acc: reset; swap; M:= 0;
b11: ns [bits] , cl  s40   ;   RM:= R long shift 40 − bits;
     [b11 has 00 in bits 10−11]
a23:                       ; acc: word:= word + R shift pos;
b12: ck [pos]  , ac (b6)   ;   pos:= pos − bits;
     nt (b11)  , qq (b12)  ;   comment M ≠ 0 indicates now too big value;
                           ; end term: R:= M; M:= bits:= 0;
a24: pan b11   X           ;   state:= 1;  goto test;

a25: hs  a19   , hh  a26   ;          state 1:   0, get next     2, mult bits
     hv  a19   , hv  a20   ;          after term 0, get next     exit

a26: hv  a27   , hs  a18   ;          state 2:   3, mult val     2, mult bits
     hh  a17   , hh  1a19  ;          in bits    1, mval, acc    alarm exit

a27: hs  a28   , hh  a22   ;          state 3:   3, mult val     1, reset acc
     hv  a19   , hh  a22   ;          in value   0, get next     1, reset acc
a28:
b13: arn[digit]Dt   −57    ; mult val: RM:= M × 10 + digit value;
     ck  10    , ml  c5    ;   goto test;

a29h:hv  a19   , ns (b6)   ; end pass 2:
     it  sd2   , pa  2e4   ;   inf 1:= initial ishort − ishort;
     it (b6)   , pa  e16   ;   save final i short for use by next segment;
     nt  d1    , it (b2)   ;   inf 2:= ilong − initial i long;
     pa  3e4   , hhn e29   ;   R:= 0; goto next segm;
```

```
[output four outputs 4 bytes, either from R (copy = f) or from
 input (copy = t).]
a30: arn(e1)   t   1   LZA ; output four:
     hs  e2            LA  ;   if copy then Raddr:= next byte;
     ga  b14   , ck  10    ;   out byte:= Raddr; R:= R shift 10;
b14: pm  −1    DX          ;   output (outbyte);
     hs   e3               ;   i:= i + 1 mod 4;
     ncn 0[i]  X  −256     ;   if i ≠ 0 then
a31: pm  r−1   , hv  a30   ; output R: begin LA:= f; goto output four end
                           ;   goto next;
a32: hv  a1    , ps  i     ; beg code:
     arn(e1)   t   1       ;   for Raddr:= next byte while Raddr < 512 do
     hs  e2            LA  ;     output (Raddr);
     hv  e3            NT  ;
     tk  −30   , sc  e4    ;   CRcount:= CRcount + R × 2↑(−30);
     tk  30    , ps  a1−1  ;   set return (next); goto output;
a33: hv  e3    , arn c3    ; CR: CR count:= CRcount + 1;
     ac  e4    , hv  a1    ;   goto next;

a34: bs s−1014 , ps  1014  ; special actions: if s > 1014 then s:= 1014;
     arn(sd5)  D           ;   output part 1 (act table[s − 1008);
     ps  sd6   , hv  e3    ;   goto act part (act table[s − 1008]);


[Action table: used for action bytes > 511: part 1 is unconditionally
 output and the action specified in right half entered
                           input. action.]
     qq  264   , hh  a32   ; 1008 begcode
     qq  283   , hh  a29   ; 1009 end pass
     qq  77    , hh  a33   ; 1010 CR
     qq  78    , hv  a30   ; 1011 short str
     qq  79    , hv  a30   ; 1012 long str
     qq  80    , hv  a30   ; 1013 layout
     qq  80    , hv  a15   ; 1014 − 1023 pattern

d d5=i+9, d6=d5−1          ; define act table bases.
d1:  qq [pass sum]         ; define initial i long = first free after pass 2;

d e22=k−e14, e47=j; Define load parameters.
b k=e23, i=0      ; Load segment word 2.
i=2e21            ;
qq e16.9+1d1.19−e16.19+2.24+a1.39;
e                              ;

e [end pass 2]    ;

s
```

[27.4.67]
[Gier Algol 4, pass 3 phase 1, std proc, page 1]

```
[This segment assumes that the first letter table and the table
 from pass 2 still is in core.
 It takes in thetable of standard procedure identifiers word by word
 using the word driver in GP. Each identifier is looked up in
 the pass 2 table and if found 2 bytes are output:
   1. The running number (T) in the standard procedure table.
   2. The pass 2 identifier.

 Format of a standard procedure identifier:
   1. A short word corresponding to short word in pass 2 with:
      part 1 = first letter (range 1−56).
      If long identifier then part 2 = 0 and bit 40 = 0
      otherwise bits 10−39 holds the rest of the identifier and
      bit 40 = 1.
   2. If long identifier:
      One or more long words as in pass 2 but in such an order
      that the one with bit(0) = 1 comes last i.e. opposite
      the order in which pass 2 generates them.

 The byte pair output (see above) is preceded by a byte 0
 (to stop on in pass 4) and followed by <first free ident>
 which is stored by pass 2 in part 1 of letter table[0|.

 This segment is loaded at 29e16, which is just after
 the first letter table in pass 2.]

b k=e22+e14, i=29e16−e47, a16, b8, c3, d7; drumblock head
i=29e16                                   ;

b k=e31, i=0  ; load texts
i=e32         ;
d1: tstd proc format;
d2: tzero;    ;
d7: tdouble std proc;
e32=i         ;
e             ;
```

```
a:   hs  e5                ; zero mess: mess (⫞<zero⫟, 2, 3);
     hh  r1    , qqn pd2   ;   pause;
     qq        , ud  14e4  ;
     lyn r−1   , ud  16e4  ;
a1:  arnf b7   , srf b7    ; start: if wrong floating point zero then
     grf  b8               ;
     ncn(b8)   , hv  a     ;   go to zero mess;
     hsn e3                ;   output (0);
     qq  r2    , arn b1    ;   word driver to core d4ff;
     ps  r     , hh  e29   ;   words:=bits(10,19,M)−1;
     cln −30   , ac  b6    ;

a2:  pp  0     , hs  a15   ; next proc: p:=0; next word;
     gr  b7    , ga  b1    ;   short word:= R; first letter:= s:= Raddr;
b1:  ps d4 [first letter]  ;   if s > 56 ∨ s < 1 then goto proc err;
       [see 1a1]
     bs  s−513 t  −457     ;
     hv  a6                ;   if −,mark A then
     hv  a4            LA  ;     begin take long: P:= 1; loop long:
     pp  1                 ;     long word[p]:= R:= next word;
a3:  hs  a15               ;     if p > 5 then goto proc err;
     bs  p−5   , hv  a6    ;     p:= p + 1;
     gr  pb7   , pp  p1    ;     if R ≥ 0 then goto loop long;
     hv  a3            NT  ;     M:= mask long; PB:= f,
     pm  c1    V       IPB ;     end
a4:  pm  c2            IPB ;   else begin M:= mask short; PB:= t end;
                           ; start look up: s:=
     bs  s−28  , ud se16−28;   if s ≤ 28 then part 1 (first letter table[s]);
     ps (se16)         NZ  ;   else part 2 (first letter table[s]);
     bs  s−511 , hh  a9    ;   if s ≥ 512 then goto look up;
a5:  arn(b5)   D   1       ; not found: T:= T + 1;
     hv  a2            NT  ;   if T < 512 then goto nexxt proc;
a6:  hs  e5                ; proc err:
     hv  a13   , qq  sd1   ;   mess (⫞<std proc format⫟, 0, 1); goto skip;
a7:  arn(b2)               ; after long failed: R:= table[chain];
a8:  ga  b2    V       LT  ; next test: if Raddr < 512 then goto not found;
                           ;   chain:= Raddr; goto test it;
a9:  hv  a5    , gs  b2    ; look up: chain:= s;
b2:  arn [chain]       IPA ; test it: R:= set PA(table[chain];
     cm  b7    , hv  a8    ;   if R ∧ M ≠ short word ∧ M
                           ;     then goto next test;
     hv  a11           LPC ;   if LPC then goto found;
     pa  b3    V b7    NPC ;   if −,NPC then goto next test;
     hv  a8                ;   i1:= 0;
     ck  10    , ga  b4    ;   i2:= part 2 (R);
a10:                       ; test rest of long: i1:= i1 + 1; i2:= i2 − 1;
b3:  arn[i1]   t  +1   ITA ;   if table [i2] ≠ long word [i1] then
b4:  sr [i2]   t  −1       ;     goto after long failed;
     hv  a7            NZ  ;   if bit 0 (long word[i]) = 0 then
     hv  a10           NTA ;     goto test rest of long;

a11: pm (b2)           IPA ; found: if markB(table[chain]) then
b5:  arn 0 [T] DVt 1   NB  ;   mess(⫞<double std proc⫟,0,1)
     ps  a12   , hv e5     ;   else
     pm (b2)   D       M   ;     begin T:= T+1;  output(T);
     hs  e3            IPB ;     output(chain);  set mark B on (table[chain])
     acn(b2)   X       MPC ;     end;
     ps  a2−1  , hv  e3    ;   goto next proc;
a12=i−1
     hv  a5    , qq  sd7   ;   mess descr for double std.proc;
```

```
a13:  pp  0    , ps  i     ; skip: p:=0; set return(next word);

a15:  hsn d5           IZC ; next word: sum ok:=transport ok:=true;
      ar  2    D        LA ;   Get word; proc sum:=procsum+marksR+R;
      ar  1    D        LB ;   restore R;
      ac  b8   , arn e16−1 ;   words:=words−1;
b6:   bt −1[words] Vt−1LZA ;   if −,transport ok then
      hs  e5               ;    mess(ǂ<pass mediumǂ,2,0);
      hr  s1   , qqn e46   ;   if words≥0 then return,

      bs  p    , hv  a6    ; finis: if p>0 then goto proc err;
      arn e16  , hs  e3    ;   output(final i short);
      arn 8e4  , hs  e11   ;   restore inbuf2;
      lk 42e13 , arn b8    ;   if proc sum≠0 then
      hs  e5            NZ ;    mess(ǂ<pass sumǂ,2,0);
      hhn e29  , qqn e45   ;   R:=0; goto next segm;

b7:  qq 1 [short word] t256;   Also used for test of zero arithmetic.
     [see a1−1]
     qq [long words 1:5]   ;
     qq  ;
     qq  ;
     qq  ;
     qq  ;
b8:  qq [proc sum]         ;


c1:  qq 1.19−1.39   ; mask long
c2:  qq 1.9−1.39f   ; mask short
c3:  m          1   ;

d4:  qq [pass sum]         ;   word driver is taken in to d4.
d5:  [entry to word driver];

d e22=k−e14, e47=j         ;   define load parameters
b k=e23, i=0               ;
i=3e21                     ;
 qq a.9+d5.19−a.19+3.24+a1.39;
e                            ;

e [final end]              ;
s
```

```
d e49=3      ; tape number := 3;

[After i follows STOPCODE, SUMCODE and a sum character]
ix T2
s
```

```
[Here follows STOPCODE and CLEARCODE]


<e49−2, <−e49+4, ×  ;   test tape number
i wrong tape
s                   ;
>                   ;

d e53=10            ;   version number

<e53−e50,           ;   if version number T3 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=9             ;   define version number T2
d e63=e53           ;   define version number T3
d e64=4             ;   define version number T4
d e65=5             ;   define version number T5
d e66=6             ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e53
>                   ;
```

```
[3.8.66]

[GIER ALGOL 4, Pass 3, page 1]

b k=e22+e14,i=e16−e47, a30, b26, c87, d60;
i=e16                                    ;
d d4=e20 [max stack]                     ;

[Input byte values]
d d17=167[;], d18=184[(]                 ;
d d7=241 [trouble]                       ;

[Output byte values]
d d5=39[beg func]                                          ;
d d9=134[end else exp], d10=128[proc;], d11=2[litreal]     ;
d d12=1[lit integer], d15=22[do]                           ;
d d16=e20[dummy identifier]                                ;
d=5[value−non value spec], d13=51[undeclared]              ;
d33=68[unspec], d35=18[end proc no type]                   ;
d36=46[decl switch], d43=e20[switch dummy ident]           ;
d44=69[specgen], d45=30[end spec]                          ;
d46=19[end type proc], d47=52[spec integer]                ;
d48=166[case exp], d49=133[delete call]                    ;
d37=57[spec value int]                                     ;

[Stack representations]
d d3=12[beg block], d6=32[func], d8=1[else exp]            ;
d d19=15[beg proc], d34=1[do−single do]                    ;

b:   qq 1.3−1.39                    ; mask. c10,
[1b] qq t 256                       ; 1.0 floating, c36, c40
[2b] qq 3 t 320                     ; 10 floating. c38, c39, c41, c56
m                                   ;
[3b] qq 10.39                       ; c56,
[4b] qq 57                          ; 57.c39,
                                    ; c43,
[5b] qq 1.39                        ; 1
[6b] hs c7                          ; instruction, c7,
[7b] qq 1020.9+409.19+614.29+410.39 ; 0.1 floating
[8b] qq 1023.39                     ; 1023, 4a9
b1:  qq                             ; decl.c13, c15, c16, c17, c20
[1b1]qq                             ; N. c60, c36, c39, c40, c48, a19, a9, c67, c59,
                                    ; c56,
[2b1]qq                             ; factor. c36, c38, c48
                                    ; 1c52, 1c53, c41, c48, a16
[3b1]qq                             ; ×. c39,
d d40=1                             ; If redefined, d40=0: output of
s                                   ; state, KA, KB=10, or stack = 00
```

```
b k=e31, i=0                    ;
d i=e32                         ; Alarm texts, used in:
d51: t-delimiter;               ;    1c5,
d42: toperand;                  ;   c10-1, 2a12
d e32=i                         ;
e                               ;

c2: pi   0        V   -5   LQB  ; AFTER TROUBLE: if first after trouble then
    pi   0        t   -9        ;        first after trouble:= false else
    hv   c3                     ;        introuble:= false; go to NEXT 1;
c66:hs   e3       ,   ps  i     ; OUT: output(par M);
c1: pa  [operand] DV      NQA   ; NEXT:  operand:= 0;
    pa   c1       ,   hv  c2    ;    ifintrouble then go to AFTER TROUBLE;
c3: pmn (e1)      X   1         ; NEXT 1: byte:= input;
    hs   e2               LA    ;    if byte < 512 then go to NOT OPERAND;
    hv   c5               NT    ;    if -, introuble then
a6: hs   e3[or a5]        NQA   ;    begin if -, output identifier then
   [output identifier: true=e3, false=a5]; keep else output(byte) end;
    pa   c1       t   1   NQA         ; if -, introuble then operand := 1;
c6: pmn (e1)      X   1               ; AFTER OPERAND: byte:= input;
    hs   e2               LA    ;    if byte ≥ 512 then alarm(d41);
c5: ga   a1       V           NT  ; NOT OPERAND: R:= table[byte];
    hs   c7       ,   qq  s+d51 ;    marks:= marksof(table[byte]);
    hv   i+5              LKB   ;    skip output for KB = 1
    hs   e7               LKA   ; Special state output: delimiter
    sy   27               LKA   ;          comma
    sy  (c1)              LKA   ;          operand
    sy  (i+3)             LKA   ;          state
d i=i-d40-d40-d40-d40-d40       ; remove special output unless d40=0
a1: pmn [byte]    Xt  d2        ;    if marks > 1 then go to SPECIAL;
    hv   c9               LA    ;    OA:= bit(state, R); if marks = 0 then
a2: ck   27 [state]       IOA   ;    begin byte:= byte - 1;
    hv   a3               LB    ;    R:= table[byte]; marks:= marksof(table[byte]);
    pmn (a1)      X   -1        ;    OR:= bit(state, R);
    ck  (a2)              IOB   ;    if marks  = 0 then
    hv   a4               LB    ;        begin byte:= byte -1;
    pmn (a1)      X   -1        ;        R:= table[byte];
    ck  (a2)                    ;        if OB then byte := byte - 4
    qq  (a1)      t   -4  LOB   ;        end;
a4: qq  (a1)      t   -2  LOA   ;    if OA then byte:= byte - 2
a3: pmn (a1)      XV  -1  LO    ;    end; if bit(state,R) then byte:=byte-1;
a22:pmn (a1)      X             ;CONTROL WORD FOUND:
    ck  (c1)      V           NC  ;    R:= table[byte]; marks:= marksof(table[byte]);
    hv   c8                     ;    if marks > 0 then go to SEARCH;
    hv   c10              LQA   ;    if -, bit(operand, allowed operand part(R))
    pmn (a1)      XV      LO    ;        ∧ -, introuble then alarm (d42);
    hs   c7       ,   qq  s+d42 ;
c10:mb   b        ,   ga  a2    ; NORMALACTION: state:= newstatepart(R);
c9: hv   c35              LC    ; SPECIAL: if marks = 3 then go to INITIALIZE NUMBER;
    gt   a        ,   cl  -9    ;    parM:= outpar(R); parR:= stackpart(R);
a:  ck   -11, hv [switching part];    go to instruction[switchingpart(R)];
a5: ga   b6       ,   hr  s1    ;    procedure keep; ident:= byte;
b6: ca [ident]-1  ,   hs  s1    ;    ident word
```

```
b k=e31, i=0                    ;
d i=e32                         ; Alarm texts, used in:
d53: tdelimiter;                ;    a14-5, 2c29
d54: t-operand;                 ;    a14
d55: ttermination;              ;    1a30
d52: thead;                     ;    1b4
d e32=i                         ;
e                               ;

c63:pi   0        t   -13       ; START PASS 3: introuble:= first after trouble:=
    pm   a20      V               ;       stackidentifier:= false;
a20:hv   c51      ,   hv  c51   ;    store[0]:= jump to error 2;
    gm   0                MA    ;    for i:= upper stack limit step -1 until 0 do
a21:grn  d4       t   -1   M    ;        store with marks(stack[i], 0, 0);
    bs  (a21)     t   d1        ;    ds:= 0; state:= 27;
    hv   a21                    ;    go to NEXT;
    pp   d1       ,   hh  c66   ;
c62:hhn  e29                    ; END PASS 3: R:= 0; goto new segm;
c7: arn  s        ,   ps  c66   ; ALARM: procedure alarm(n); value n; integer n;
    hv   c6               LQA   ;    begin if introuble then go to AFTER OPERAND;
    sr   6b       ,   ac  a10   ;    errormessage(n);
    hs   e5                     ;    byte address:= byte address - 1;
a10:qq   d7       ,   qq        ;    byte:= trouble;
    pa   c1       ,   grn a10   ;    introuble:= first after trouble:= true;
    pa   a10      t   d7        ;    operand:= 0; go to NOT OPERAND
    qq  (e1)      t   -1        ;    end;
    pi   12       t   -13       ;    comment errormessages: 34: stack, 51: -delimi-
    arn  a10      ,   hv  c5    ;    ter, 52: head, 42: operand, 53: delimiter, 54:
                                ;    -operand, 55 termination, 56: number
c8: sy   64               NKC   ; Special output, stack: CARRET
    arn  (a1)      D      NKC   ;
    hs   e7                NKC   ;                byte
    sy   27                NKC   ;                comma
    sy  (c1)               NKC   ;                operand
    pmn (a1)      X             ; reset A and M
d i=i-d40-d40-d40-d40-d40-d40   ; remove special output
    gr  (a30)     DV       LA   ; SEARCH: if marks = 1 then alarm(d43);
    hs  c7        ,   qq  s+d53 ;    comment delimiter;
    ck  10        ,   gt  a11   ;    lim 1:= part 1(R); lim 2:= part 2(R);
                                ;    base:= part 3(R);
    pm  (c1)      DX            ;    if marks = 3 then go to a12;
    hv   a12               LC   ;    if operand = 0 then alarm(54);
    pm   d8       DVX      NZ   ;    comment - operand;
a14:hs   c7       ,   qq  s+d54 ;    if stack[ds] ǂ elseex then go to a13;
    nc  (p)       ,   hv  a13   ;
    sy  (p)               NKC   ; Special output: top of stack
d i = i - d40                   ; remove special output unless d40 = 0
    pm   d9       DV            ;    R:= end else expr;
```

```
c55:cl   −9                      ; c55:
    hs  e3        X              ;    output(R);
c54:pp  p−1                      ; c54: ds:= ds − 1;
a13:sy  (p)                NKC   ; Special output: top of stack
d i = i − d40                    ; remove special output unless d40 = 0
    is  (p)                      ; a13: st:= stack[ds];
a30:bs   s0       t   0          ;    if st+lim 1 < −512 ∨ st > lim 2−lim 1 then
    hs   c7       ,   qq  s+d55  ;    alarm(d55); comment termination;
a11:is  (p)       ,   pmn s[base];
    hv   c10      X        NC    ;    if marks = 0 then go to NORMAL ACTION;
    hv   c55      X        LA    ;    if marks = 1 then begin R:= part 4(R);
    hv   c54      X              ;    go to c55 end; go to c54;
a12:hv   a13                LZ   ; a12: if operand = 0 then go to a13;
    nc   1                       ;    if operand ≠ 1 then
    hs   c7       ,   qq  s+d42  ;    alarm(d42);
    pm   d10      DX             ;
    hs   e3                      ;    output(proc;);
    hv   a13                     ;    go to a13;
c11:pt   b26      t   d35        ; SETBLOCKPROC: end proc:= no type end proc
b2: pa   a6       t   a5         ;    output identifier:= false;
    pa   b4       ,   ud  b5     ;    head alarm:= false; ident:= dummy;
c12:ca  (p)                      ; SET BLOCK: if stack[ds] = parR then
    pa   p        t   d3         ;                 stack[ds]:= begblock;
c13:gm   b1       ,   hv  c1     ; SET DECL: decl:= parM; go to NEXT;
c14:pt   b26      t   d46        ; ADD DECL PROC: end proc:= type end proc;
b5: pa   b6       t   d16        ;    ident:= dummy ; head alarm:=
    pa   b4       ,   ud  b2     ;    false; output identifier:= false;
c15:xr            ,   ac  b1     ; ADD DECL: decl:= decl + parM; go to NEXT;
    hv   c1                      ;
c17:pm   b1                      ; DECL ENT: parM:= decl;
c18:pp   p1       ,   bs  p−d4   ; ENT OUT: ds:= ds + 1;
    hs   c7       ,   qq  e34    ;    if ds > maxds then alarm(38); comment stack;
c19:ga   p        VX             ; CH OUT: stack[ds] := parR; go to OUT;
c20:pm   b1       X              ; DECL: parM:= decl; go to OUT;
    hv   e3                      ;
c21:pp   p1       ,   bs  p−d4   ; ENT: ds:= ds + 1; if ds > maxds then
    hs   c7       ,   qq  e34    ;      alarm(38); comment stack;
c70:ga   p        ,   hv  c1     ; CH: stack[ds]:= parR; go to NEXT;
c28:hs   ne3      X              ; TROUBLEPROCEND: output (par M);
c22:arn  p        ,   cl  −9     ; PROCEND: R:= stack[ds];
    tk   19                      ;    par M:= par t4(R);
    hs   e3                      ;    output(part 2(R));
c61:qq  (2e4)     t   1          ; BLOCK COUNT: information 1:= information 1 + 1;
c23:pp   p−1                     ; AN OUT: ds:= ds − 1; go to OUT;
c4: hv   e3       X              ;
c24:pp   p−1      ,   hv  c1     ; AN: ds:= ds − 1; go to NEXT;
c64:hs   e3       X              ; DO: output(par M);
    arn  d15      DX             ;    parM:= do;
    bs  (c34)     ,   ar  r1     ;    if count ≠ 0 then par R:=
    qq   d34      ,   hv  c19    ;    par R + do_difference; goto CHOUT;
c26:bs  (c1)                     ; LEFT PARENT: if operand > 0 then
    pmn  d5       D              ;        begin parR:= func; parM_begcall_ end;
    arn  d6       D        LZ    ;    go to ENT OUT;
    hv   c18
```

```
c68:hs   e3       X              ; RIGHT CALL: output(par M);
    arn  d49      DX             ;    par M:= delete call;
c25:pp   p−1      ,   ga  c1     ; RIGHT: ds:= ds − 1; operand:= parR;
    hsn  e3       X              ;    output(parM); go to AFTER OPERAND;
    hv   c6                      ;
c69:                             ; BOUNDS:
    mb   1023     D              ;    R:= part 1(R);
    hs   e3       X              ;   output(par M); M:=R;
    hvn  c25                     ;    R:= 0; goto RIGHT;
c27:ga   b22      ,   can(c1)    ; PLUSMINUS: monadic:= par R;
b22:pm [monadic]−1 D             ;    if operand = 0 then par M:= monadic;
    hv   e3       X              ;    goto OUT;
c29:bs  (c1)                     ; BINARY: if operand = 0 ∨ state > 7 ∨ introuble
    bs  (a2)      t   7    NQA   ;        then alarm(d43);
    hs   c7       ,   qq  s+d53  ;    comment delimiter;
    ga   a2       X              ;    state:= parR; go to OUT;
    hv   e3                      ;
c30:pi   8        t   −9         ; AN TROUBLE: introuble := true; ds:= ds − 1;
    pp   p−1      ,   hv  c6     ;    go to AFTER OPERAND;
c32:arn  5b       ,   ac  e4     ; CR: CRcounter:= CRcounter + 1;
    hs   e3       X              ;    output(parM);
    bs  (c1)             NQA     ;    go to if operand = 0 then NEXT 1
    hv   c6                      ;        else AFTER OPERAND;
    hv   c1                      ;
b8: qq 1.19+50.35+1.39−1.36      ; instruction modifier (3c16)
c16:pp  p+1       ,   gp  b7     ; FORMALLIST: ds:= stack entry:=
    arn b1        ,   ck  −10    ;     ds+1; stack[ds]:= instruction
    ar  b6        ,   sr  b8     ;
b7: gr  −1 [stack entry]   MA    ;   (ca <ident>, qqn decl) ; cf. 1b26
    gp  b17       ,   hh  c71    ;    goto SET STOP;
c72:arn   b6      ,   hs  (b7)   ; FORMAL: R:= ident word; call
c47:                             ;   (store [stack entry]);
[1] gp   b4       ,   hh  c66    ; return 1: head alarm:= true;
                                 ;   goto NEXT;
b9: qq            ,   hs  s3     ;    stop instruction, 1c71
[3] hv   c47             LZ      ; return 3: if R = 0 then goto return 1;
                                 ;    stack [ds]:= R;

c71:gr   p        ,   pp  p1     ; SET STOP: ds:= ds + 1; if
    pm   b9       ,   bs  p−d4   ;    ds > limit then ALARM (38);
    hs   c7       ,   qq  e34    ;    stack[ds]:= instruction(qq,
    gm   p             MA        ;    hs s3); goto NEXT;
    hh   c66                     ;
```

[When we enter FORMAL the top of the stack has the following form:

```
     stack entry: ca procedure identifier, qqn decl
                  ca formal ident 1      , hs  s1
                  ca formal ident 2      , hs  s1
                  . . . . . .
              p : qq                     , hs  s3
```

No two identifiers are the same]

```
c46:pt   b10      V   b11        ; VALUE: mod:= m2; goto SETSPEC;

c74:pt   b10      t   b13        ; FIRST SPEC: mod:= m1;
    pt   b13                     ; SETSPEC: Store [m1]:= 0;

c75:ga   b12      X              ; SECOND SPEC: value
    ck   −10      ,   ac  b13    ;    allowed:= Rpar; store[m1]:=
    hv   c1                      ;    store[m1] + M par; goto NEXT;

[The current for of the top part of the stack:
     stack entry:  ca procedure identifier, qq n decl
For each formal identifier, one word in one of three formats:
(1) No value, no spec. before     ca formal identifier , hs  s1
(2) Value, no spec.               ca formal identifier , hss c73 − c76
(3) Already specified             ca formal identifier , qqn specif.
At end of list:                   qq                   , hs  s3
No two identifiers in the list are the same]

c76:arn  b6       ,   hs (b7)    ; SPEC COMMA: R:= ident word;
b10:                             ;    Call (store [stack entry]);
[1] qqn           ,   ar  [mod]  ; return 1: R:= 0;
                                 ; modify: stack [s]:= stack [s] +
[2] ac   s        ,   hh  c66    ;    R + store[mod]; goto NEXT;
[3] hv   c47           LZ        ;    if R = 0 then begin head alarm:= true;
                                 ;    goto NEXT end;
    gp   b4       ,   ar  b13    ;    head alarm:= true; R:= R + mod 1;
    hv   c71                     ;    goto return 3;
c73:gr   s        ,   arn b14    ;    stack[s]:= R; R:= value modifier;
b12:nc   0        ,   gpn b4     ;    if value no then begin R:= 0; head alarm:=
   [value allowed: yes = d, no=0];    true end;
    tk   −10      ,   hh  b10    ;    goto modify;
b14:qq   d                       ;
b26: qq[no.9+end.19+] d45.29+d19.39;
[1b26] gr 0       t   5  MA      ;
c77:ga   b16      ,   srn  b7    ; COMPL HEAD: no spec:= par R;
    ar   p−1      D      IQA     ;    R:= ds − 1 − stack entry;
    ar   1b26     ,   ga  b26    ;    no of par:= R;
    ca   0        ,   ac (b7)    ;    if R = 0 then add 5 to (
                                 ;      part 2 [stack entry]);
    ncn (b4)                     ;    introuble:= false; if head
    pt  (b7)      t   d13        ;      alarm then part 2 [stack entry]:=
                                 ;      undeclared;
b17:arn [i]       ,   ga  b18    ;    for i:= stack entry step 1 until ds−1 do
    ck   −10           IOA       ;    begin R:= stack[i]; id:= part 1(R);
    ck   20                      ;    OA:= op2(R) ǂ qqn; R:= part 2(R);
b16:arn [no spec] D        LOA   ;    if −, OA then R:= no spec;
    ga   b19      ,   ca  d33    ;    spec:= R;
    gp   b4                      ;    if R = unspec then head alarm:= true;
b18:pmn  −2 [id]  DX             ;    output(id);
    hs   e3                      ;
b19:pmn  −4 [spec]DX             ;  output(spec);
    hs   e3                      ;
    arn (b17)     Dt   1         ;
```

```
    nc   p        ,   hv  b17    ;    end for i;
    pan  a6       X   e3         ;    output identifier:= true;
    arn  b26      ,   ck  −10    ;    The top word of the stack
    cl   −20      ,   ck  10     ;    is set to qq beginproc.9+
    cl   −29      ,   pp  (b7)   ;    no of param.19 + endproc.39;
    gm   p        ,   hs  e3     ;    output (end spec);
    qq  (e1)      t   −1         ;    byte address:= byte address − 1;
b4: ncn[head alarm],  hs  e5     ;    if head alarm then
    ps   c66      ,   qq  s+d52  ;    ALARM(‡<head›);
    pm  (b6)      D              ;    if operand = 1 then
    arn  c1       ,   ca  1      ;    output(ident);
    hs   e3       X              ;
    hv   c6                      ;    goto AFTER OPERAND

c78:ga   b16      ,   nsn(a6)    ; SEMICOLON: code:= par R; if −, output
    ca   sa5      ,   hh  c77    ;    identifier then goto COMPL HEAD;
    pa   a1       t   b25        ;    byte:= b25;
    hv   a22                     ;    goto CONTROL WORD FOUND;
c31:pa   c34      ,   hv  c4     ; FOR: for count:= 0; goto OUT;

c36:it   −1                      ; WHILE COUNT: count:= count − 1;
c34:qq [for comma count] t 1     ; COUNT CHOUT: count:=
    hv   c19                     ;    count + 1; goto CHOUT

c82:ck   10       ,   tk  30     ; CODE: output(par R);
    hs   e3       ,   ps  i      ;
    arn (e1)      t   1          ; Q: R:= input;
    hs   e2                LA    ;    if R ≥ 0 then
    hv   e3                NT    ;    begin output (R); goto Q end;
    tk   −30      ,   sc  e4     ;    CR counter:= CR counter
    tk   30       ,   hs  e3     ;    + R; output (R);
    ps   c66      ,   hv  c4     ;    goto OUT;
b23:qq 1.19+d35.39               ;    begin label proc, stack word
c83:ca  (p)                      ; SWITCH: if stack [ds] = par R then
    pa   p        t   d3         ;    stack [ds]:= beg block;
    arn  b23      ,   gr  p1     ;    stack[ds + 1]:= begin label proc;
    hv   c21      X              ;    goto ENT;

c84:                             ; SWITCH ASSIGN:
    qq   d36      ,   hs  a29    ;    output(decl label proc with par);
    qq   d43      ,   hs  a29    ;    output(i);
    qq   d37      ,   hs  a29    ;    output(spec value integer);
    qq   d45      ,   hs  a29    ;    output(end spec);
    qq   d48      ,   hs  a29    ;    output(case expr);
    qq   d43      ,   hs  a29    ;    output(i)
    arn  17       D              ;    par R:= :=switch
    ps   c66      ,   hv  c18    ;    goto ENTOUT;
a29:arn (s)       D              ;
    hv   e3                      ;
```

```
c37:pmn (e1)      X   1          ; NEXT OF NUMBER: s:= input(byte);
    hs   e2               LA    ;    s:= if 56 < s ∧ s < 67 then 57
    ga   r1                     ;        else if 66 < s ∧ s < 77 then s − 9
    ps                          ;        else 56;
    bs   s445     t   501       ;    go to instruction[numbers +
    ps   57       ,   hh  r3    ;        bits(numberstate, numberstate + 4,
    bs   s435     t   501       ;        numberstatetable[s])];
    ps   s−9      ,   hh  r1    ;
    ps   56       ,   arn sd2   ;
a15:ck [numberstate], tk  −5    ;
    ga   r1                     ;
    hv [switchingpart]t  c38    ;
c58:srn  57       D             ;    procedure mult 10;
    ar  (e1)      ,   ck  10    ;    MR:= MR × 10 +
    ml   3b       ,   hv  s1    ;        symbol − 57;
c60:gm  (a17)     D             ; LOGIC VALUE: kind:= outpart
    cl   9        ,   gm  2b1   ;    n2:= bit 8 (par R);
    tk   1        ,   tk  −40   ;    n1:= bit 9 (par R) from 0 to 39;
    hh   a8                     ;    goto a8;
c33:gm  (a17)     D             ; LITERAL: kind:= outpart;
    pa   a7       t   3         ;   for i:= 0 step 10 until 30 do
    arn (e1)      t   1         ;    pack(M, i, i+9, input);
    hs   e2               LA    ;
a7: bt   0        t   −1        ;
    cl   10       ,   hv  r−3   ;
a8: cl   −30      ,   gr  1b1   ; a8: n1:= M;
    hv   c67                    ;    goto OUTPUT LITERAL;
a26:ga   a23      ,   arn 1b1   ; PACK REAL: exp 10:= Radr;
    nkf  39       ,   dkf 3b1   ;    R:= N/factor; for exp 10 :=
a23:bt [exp 10]   t   −1        ;   exp 10 − 1 while exp 10 > 0 do
a27:mkf [q]       ,   hv  a23   ;    R:= R × (if pos exp then 10 else 0.1);
    grf  1b1                    ;    N:= R;
c49:qq  (e1)      t   −1        ; OUTPUT NUMBER: byte address:=
                                ;    byte address − 1; OUTPUT LITERAL:
c67:hv   a28              LQA   ;    if introuble then goto AFTER OPERAND;
    bsn (a2)      Xt  7         ;    if state ≥ 7 then
    hv   a25                    ;    goto AFTER CONST OUT;
    arn  1b1      ,   pa  a18   ;    for i:= 0 step 10 until 30 do
a19:cl   10       X             ;       output (bits(i, i+9, n1));
    ck   −10      ,   hs  e3    ;
a18:btn           t   −150      ;
a17:pmn [kind]    DXV           ;
    hv   a19      X             ;
    hs   e3                     ;    output(kind)
a25:pm  (c1)      DXt  3        ; AFTER CONST OUT: operand:=
    ca   3                      ;    R:= operand + 3; if R = 3 then
a28:ps   c66      ,   hv  c6    ;    goto AFTER OPERAND;
    qq  (e1)      t   1         ;    byte address:= byte address + 1;
    hs   c7       ,   qq  s+d56 ;    error message(ǂ<const.ǂ);
```

```
c35:qq  (e1)      t   −1          ; INITIALIZE NUMBER:
                                  ;    byte addres:= byte address − 1;
    grn  1b1      ,   pa  a15     ;    n1:= 0; nstate:= 0;
    pm   1b       ,   gm  3b1     ;    factor:= 1;
    hv   c37                      ;    goto NEXT OF NUMBER;
c38:arnf 3b1      ,   mkf 2b      ; DIGIT AFTER POINT:
    grf  3b1      ,   it  15      ;    factor:= factor × 10;
                                  ;    nstate:= 15; goto DIGIT 12;
c39:pa   a15      t   5           ; DIGIT BEFORE POINT: nstate:= 5;
    pm   1b1      ,   hs  c58     ; DIGIT 12: M:= n1 × 10 + (symbol − 57)
    gm   1b1                      ;    n1:= M
    pa   a15      t   35   NZ     ;    if overflow from n1 then
                                  ;       nstate:= 35;
c81:hv   c37                      ; BLIND: goto NEXT OF NUMBER;
c52:pa   a15      t   30          ; DIGIT 3: nstate:= 30;
    pm   2b1      ,   hs  c58     ;    n2:= n2 × 10 + (symbol − 57);
                                  ;    if overflow from n2 then
c53:pa   a15      t   35   NZ     ; ERROR 1: nstate:= 35;
    gm   2b1      ,   hv  c37     ;    goto NEXT OF NUMBER;
b k=e31, i=0                      ;
d i=e32                           ;    Alarm text used in:
d56: tconst.;                     ;    c48, 2a28
d e32=i                           ;
e                                 ;

c40:pm   5b       ,   gm  1b1     ; TEN 1: n1:= 1;
c41:pa   a27      t   2b          ; TEN 2: q:= 10;
    grn  2b1      ,   it  20      ;    n2:= 0; nstate:= 20;
                                  ;    goto NEXT OF NUMBER;
c42:pan  a15      Vt  10          ; POINT: nstate:= 10;
                                  ;    goto NEXT OF NUMBER;
c43:pa   a27      t   7b          ; EXPMINUS: q:= 0.1
c44:pa   a15      t   25   NZ     ; EXPPLUS: nstate:= 25;
    hv   c37                      ;
c51:grn  1b1      ,   hs  e5      ; ERROR 2: n1:= 0;
                                  ;    error message(ɬ<const.ɬ);
c48:grn  2b1      ,   qq  s+d56   ; FINISH 2: n2:= 0;
c79:pmn  2b1      ,   cl  30      ; FINISH 3: Madr:= n2;
    hv   c51           NZ         ;    if n2 > 511 then goto ERROR 2;
    pa   a17      Xt  d12         ; FINISH 1: kind:= integer;
    hv   c49                      ;    goto OUTPUT NUMBER;
```

```
[3.8.66]
[GIER ALGOL 4, Pass 3, page 10, stack words 1]
[New state.9+switching part.19+stack part.29+output part.39]

      [stack: switching part, stack part, output]
      [; 4 and ;7]
      d d20=i-3                          ;
[3]   qqf                                ; trouble: -, -, -
[4]   qq                          136.39,; thenst: -, -, end thenst
[5]   qq                          137.39,; goto: -, -, end goto
[6]   qq                           75.39,; assign: -, -, end assign
[7]   qq                          171.39,; loop: -, -, end loop
[8]   qq                          142.39,; single do: -, -, end single do
[9]   qq                          141.39,; do: -, -, end do
[10]  qq                          135.39,; elsest: -, -, end elsest
[11]  qq 27.9+ c4.19       +       21.39 ; begclean: OUT, -, ;
[12]  qq 27.9+ c4.19       +       21.39 ; beg block: OUT, -, ;
[13]  qq 27.9+ c4.19       +       21.39 ; beg body: OUT, -, ;
[14]  qq 27.9+ c4.19       +      170.39 ; of st: OUT, -, case semicolon
[15]  qq 28.9+c22.19       +        0.39 ; beg proc: PROC END, -,
[16]  qq 28.9+c23.19       +       29.39 ; core: ANOUT, -, core code end
[17]  qq                   +      155.39,; := switch: -, -, end switch

       [end 1 and end 2]
      d d21=i-3                          ;
[3]   qqf                                ; trouble: -, -, -
[4]   qq                          136.39,; thenst: -, -, end thenst
[5]   qq                          137.39,; goto: -, -, end goto
[6]   qq                           75.39,; assign: -, -, end assign
[7]   qq                          171.39,; loop: -, -, end loop
[8]   qq                          142.39,; single do: -, -, end single do
[9]   qq                          141.39,; do: -, -, end do
[10]  qq                          135.39,; elsest: -, -, end elsest
[11]  qq 20.9+c23.19       +       15.39 ; beg clean: AN OUT, -, end clean
[12]  qq 20.9+c61.19       +       16.39 ; beg block: BLOCK COUNT, -, end block
[13]  qq 10.9+c24.19                     ; beg body: AN, -, -
[14]  qq 20.9+c23.19       +       26.39 ; of st: AN OUT, -, end case st

       [else 1 and else 2]
      d d22=i-2                          ;
[2]   qq  3.9+c19.19+  1.29+      132.39 ; thenex: CH OUT, else ex, else ex
[3]   qq 34.9+c30.19                     ; trouble:  AN TROUBLE, -, -
[4]   qq  9.9+c19.19+ 10.29+       24.39 ; thenst: CH OUT, else st, else st
[5]   qq                          137.39,; goto: -, -, end goto
[6]   qq                           75.39,; assign: -, -, end assign
```

```
[3.8.66]
[GIER ALGOL 4, Pass 3, Page 11]
[stack words 2]

              [,7]
         d d23=i−17                    ;

[17]  qq  2.9+ c4.19+          169.39 ; := switch: OUT, −, case comma
[18]  qq  2.9+c19.19+ 38.29+   152.39 ; next colon: CHOUT, array comma, not first bound
[19]  qq  2.9+c19.19+ 38.29+   151.39 ; first colon: CHOUT, array comma, first bound
[20]  qq  2.9+c19.19+ 23.29+    35.39 ; [left: CHOUT, left, comma 1
[21]  qq  2.9+c19.19+ 24.29+    35.39 ; [subscr: CHOUT, subscr, comma 1
[22]  qq  2.9+c19.19+ 25.29+    35.39 ; [left or subs: CHOUT, left or sub, comma 1
[23]  qq  2.9+ c4.19+           36.39 ; , left: OUT, −, comma 2
[24]  qq  2.9+ c4.19       +    36.39 ; , subscr: OUT, −, comma 2
[25]  qq  2.9+ c4.19+           36.39 ; , left or subs: OUT, −, comma 2
[26]  qq  2.9+c34.19+ 27.29+    70.39 ; single comma: COUNTCHOUT, := for, simple for
[27]  qq  2.9+c34.19+ 26.29+    70.39 ; := for: COUNT CHOUT, single comma, simple for
[28]  qq  2.9+c34.19+ 27.29+    72.39 ; until: COUNT CHOUT, := for, stepelem
[29]  qq  2.9+c34.19+ 27.29+    73.39 ; while: COUNT CHOUT, := for, while elem
[30]  qq  2.9+ c4.19+          169.39 ; of exp: OUT, −, case comma

              [, 7 and parameter delimiter]

         d d41=i−31                    ;

[31]  qq  2.9+ c4.19+           34.39 ; (call: OUT, −, call param
[32]  qq  2.9+ c4.19+           34.39 ; (func: OUT, −, call param

       [right bracket 1]
      d d24=i−18                     ;
[18]  qq 13.9+c69.19+ 17.29+   152.39 ; next colon: BOUNDS, end bounds, not first bound
[19]  qq 13.9+c69.19+ 17.29+   151.39 ; first colon: bound base, first bound
[20]  qq 27.9+c25.19+  2.29+    32.39 ; [left: RIGHT, NEW OPERAND, ] one
[21]  qq  1.9+c25.19+  2.29+    32.39 ; [subscr: RIGHT, NEW OPERAND, ] one
[22]  qq  4.9+c25.19+  2.29+    32.39 ; [left or subs: RIGHT, NEW OPERAND, ] one
[23]  qq 27.9+c25.19+  2.29+    33.39 ; , left: RIGHT, NEW OPERAND, ] more
[24]  qq  1.9+c25.19+  2.29+    33.39 ; , subscr: RIGHT, NEW OP, ] more
[25]  qq  4.9+c25.19+  2.29+    33.39 ; , left or subscr: RIGHT NEW OP, ] more

       [do 1]
      d d25=i−26                     ;
[26]  qq 27.9+c64.19+  8.29+   162.39 ; single comma: DO, single do, simple for do
[27]  qq 27.9+c64.19+  8.29+   162.39 ; := for: DO, single do, simpel for do
[28]  qq 27.9+c64.19+  8.29+   163.39 ; until: DO, single do, step elem do
[29]  qq 27.9+c64.19+  8.29+   164.39 ; while: DO, single do, while elem do

       [right parenthesis 2]
      d d26=i−30                     ;
[30]  qq  1.9+c25.19+  3.29+   168.39 ; of exp: RIGHT, NEW OPERAND, end case exp
[31]  qq 38.9+c68.19       +    31.39 ; (call: RIGHT CALL, NEW OPERAND, end call
[32]  qq  1.9+c25.19+  3.29+    31.39 ; (func: RIGHT, NEW OPERAND, end call
[33]  qq  1.9+c25.19+  3.29+   161.39 ; (subex: RIGHT, NEW OPERAND, )
```

```
       [then 1]
      d d27=i-34                      ;
[34]  qq  5.9+c19.19+  2.29+  131.39 ; ifex: CH OUT, thenex, thenex
[35]  qq  8.9+c19.19+  4.29+   23.39 ; ifst: CH OUT, thenst, thenst

       [step 1]
      d d28=i-26                      ;
[26]  qq  2.9+c19.19+ 36.29+  139.39 ; single comma: CHOUT, step, step
[27]  qq  2.9+c19.19+ 36.29+  139.39 ; := for: CH OUT, step, step

       [until 1]
      d d29=i-36                      ;
[36]  qq  2.9+c19.19+ 28.29+  140.39 ; step: CH OUT , until, until

       [while 1]
      d d30=i-26                      ;
[26]  qq  2.9+c36.19+ 29.29+   74.39 ; single comma: WHILE COUNT, while, while
[27]  qq  2.9+c19.19+ 29.29+   74.39 ; := for: CH OUT, while, while

       [:3]
      d d31=i-37                      ;
[37]  qq  2.9+c19.19+ 19.29+   37.39 ; [arr: CH OUT, first colon, boundcolon
[38]  qq  2.9+c19.19+ 18.29+   37.39 ; array comma: CH OUT, next colon, bound colon

       [of]
      d d38=i-39                      ;
[39]  qq 11.9+c23.19+         167.39 ; case exp: ANOUT, -, of exp
[40]  qq 26.9+c23.19+          25.39 ; case st: ANOUT, -, of st

       [for 3]
      d d39=i-8                       ;
[8]   qq 20.9+c19.19+  7.29+  142.39 ; single do: CHOUT, loop, end single do
```

[trouble 1]

```
d d32=i−1                              ;

[1]   qqf                              ; else ex: −, −, −
[2]   qqf                              ; then ex: −, −, −
[3]   qqf                              ; trouble: −, −, −
[4]   qq 34.9+c18.19+  3.29+   41.39 ; then st: ENT OUT, trouble, trouble
[5]   qq 34.9+c19.19+  3.29+   41.39 ; goto: CH OUT, trouble, trouble
[6]   qq 34.9+c19.19+  3.29+   41.39 ; assign: CH OUT, trouble, trouble
[7]   qqf                              ; loop:
[8]   qq 34.9+c18.19+  3.29+   41.39 ; single do: ENTOUT, trouble, trouble
[9]   qq 34.9+c18.19+  3.29+   41.39 ; do: ENT OUT, trouble, trouble
[10]  qq 34.9+c18.19+  3.29+   41.39 ; elsest: ENT OUT, trouble, trouble
[11]  qq 34.9+c18.19+  3.29+   41.39 ; beg clean: ENT OUT, trouble, trouble
[12]  qq 34.9+c18.19+  3.29+   41.39 ; beg block: ENT OUT, trouble, trouble
[13]  qq 34.9+c18.19+  3.29+   41.39 ; beg body: ENT OUT, trouble, trouble
[14]  qq 34.9+c18.19+  3.29+   41.39 ; of st: ENTOUT, trouble, trouble
[15]  qq 32.9+c28.19       +   41.39 ; beg proc: TRPROCEND, −, trouble
[16]  qq 32.9+c23.19       +   41.39 ; core: ANOUT, −, trouble
[17]  qqf                              ; := switch: −, −, −
[18]  qq 32.9+c23.19+          41.39 ; next colon: ANOUT, −, trouble
[19]  qq 32.9+c23.19       +   41.39 ; first colon: AN OUT, −, trouble
[20]  qqf                              ; [left: −, −, −
[21]  qqf                              ; [subscr: −, −, −
[22]  qqf                              ; [left or subs: −, −, −
[23]  qqf                              ; , left:
[24]  qqf                              ; , subscr:
[25]  qqf                              ; , left or subscr:
[26]  qqf                              ; single comma:
[27]  qqf                              ; := for: −, −, −
[28]  qqf                              ; until: −, −, −
[29]  qqf                              ; while: −, −, −
[30]  qqf                              ; of exp
[31]  qqf                              ; (call: −, −, −
[32]  qqf                              ; (func: −, −, −
[33]  qqf                              ; (subex: −, −, −
[34]  qqf                              ; if ex: −, −, −
[35]  qqf                              ; if st: −, −, −
[36]  qqf                              ; step: −, −, −
[37]  qq 32.9+c23.19       +   41.39 ; [arr: AN OUT, −, trouble
[38]  qq 32.9+c23.19       +   41.39 ; array comma: ANOUT, −, trouble
[39]  qqf                              ; case exp:
[40]  qqf                              ; case st
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 14, control table for numbers]

[Input byte values from 56 to 68 have their control words c−marked (i.e.
f and comma marked) and call for a special logic for analyzing numbers.
While this logic is operating the input byte values are converted so as
to give the appropriate entry of the following table. This is an action
table having the current numberstate as the other argument. The action is
given in the table as the machine address of the code relative to DIGIT 2
= c38. The number states are:

Number state	Position	
0	4	Before number
5	9	Following digit before point
10	14	Following point
15	19	Following digit after point
20	24	Following ten
25	29	Following exponent sign
30	34	Following digit after ten
35	39	In erroneous number

When an error has been detected the remaining part of the number is
skipped. The error message is given on the following terminator.]

[Entry to table: direct converted]

```
d d2=i−56
d c39=c39−c38, c81=c81−c38, c52=c52−c38 ;
d c53=c53−c38, c40=c40−c38, c41=c41−c38 ;
d c42=c42−c38, c43=c43−c38, c44=c44−c38 ;
d c51=c51−c38, c48=c48−c38              ;
d c79=c79−c38, c80=c80−c38              ;
```

```
[56]  qqf c51.4+c80.9+c51.14+c48.19+c51.24+c51.29+c79.34+c51.39,;        <terminator>
[57]  qqf c39.4+c39.9+  0.14+  0.19+c52.24+c52.29+c52.34+c81.39,;    0   <digit>
[58]  qqf c42.4+c42.9+c53.14+c53.19+c53.24+c53.29+c53.34+c81.39,;    1      .
[59]  qqf c40.4+c41.9+c53.14+c41.19+c53.24+c53.29+c53.34+c81.39,;    2      10
[60]  qqf,                                                           ;    3
[61]  qqf,                                                           ;    4
[62]  qqf,                                                           ;    5
[63]  qqf c51.4+c80.9+c51.14+c48.19+c44.24+c51.29+c79.34+c51.39,;    6      +
[64]  qqf,                                                           ;    7
[65]  qqf,                                                           ;    8
[66]  qqf,                                                           ;    9
[67]  qqf c51.4+c80.9+c51.14+c48.19+c43.24+c51.29+c79.34+c51.39,;    .      −
[68]  qqf,                                                           ;    10
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 15, main control table]

[Each input byte gives access to a word in the following table. If this word is NOT COMMA−MARKED the table contains NORMAL action words arranged as follows:

```
               Delimiter action word p
                   −       −     −   p−1
               . . . . . . .
                   −       −     −   1
               Delimiter meaning word q
                   −        −     −   q−1
byte value:        −        −     −   1
               Delimiter meaning comment.
```

The delimiter meaning words in their bit no. w contain the number of the action word appropriate to the state w, in binary form, thus:

```
                                          Multiplier
q=1, p=1      Delimiter meaning word 1:         1

q=2, p=2 or 3     −        −     −   2:         1
                  −        −     −   1:         2

q=3, p=4 to 7     −        −     −   3:         1
                  −        −     −   2:         4
                  −        −     −   1:         2
```

The action word number corresponding to the various states is also given in the delimiter meaning comment. Delimiter meaning word q is f−marked.

Delimiter action words come in two formats:

NO MARKS: SIMPLE ACTION WORDS

qq allowed operand.3+new state.9+switching part.19+stack part.29+output.39

When the program indicated in the switching part is entered we have

R: qq stack part.9+(output:512).10+new state.29+switching part.39
M: qq (output mod 512).9

qq(f) lim 1, qq lim 2 + base address.19

For a delimiter admitting the stack delimiters LOW to UP inclusive we have: lim 1 = −(512+LOW), lim 2 = lim 1 + UP.

Not f−marked words perform TEST FOR ELSE EXPRESSION. f−marked words perform TEST FOR PROCEDURE CALL. The base address points to the table of stack words.

If the word indicated by the input byte value is COMMA−MARKED we have a SPECIAL ACTION:

Not−f marked words supply:

qq switching part.19+parameter.29+output.39,

When the program indicated in the switching part is entered we have

R: qq parameter.9+(output:512).10+switching part.39
M: qq (output mod 512).9

f−marked words enter the number reading program INITIALIZE NUMBER, controlled by the control table for numbers.]

```
[3.8.66]
[GIER ALGOL 4, Pass 3, Page 16, main control table]

qq   7.3+ 6.9+ c4.19         +  144.39 ; +2: OUT, −, +
qq  15.3+ 6.9+c27.19+  504.29+  144.39 ; +1: PLUSMINUS, pos, +
qqf 31.5+ 1.7                          ; 1
qq   1.6                               ; 2: 72
[01111 12100 1 00000 00000 2 00000 00000 3 00000 00000]
qq   7.3+ 6.9+ c4.19         +  145.39 ; −2: OUT, −, −
qq  15.3+ 6.9+c27.19+  505.29+  145.39 ; −1: PLUSMINUS, neg, −
qqf 31.5+1.7                           ; 1
qq   1.6                               ; 2: 76
[01111 12100 1 00000 00000 2 00000 00000 3 00000 00000]
qq            c32.19         +   0.39, ; 77 CARRET: CR, −, CR
qq            c33.19         +   4.39, ; 78 short string: LITERAL , −, lit string
qq            c33.19+  512.29+   4.39, ; 79 long string: LITERAL, −, lit string
qq            c33.19         +   3.39, ; 80 boolean: LITERAL , −, lit string
qq  8.3+ 7.9+  c4.19         +  500.39 ; −,1: OUT, −, not
qqf 31.5+                              ; 1: 82
[01111 10000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  8.3+27.9+ c77.19  +d33.29          ; goto 2: COMPL HEAD, unspec
qq  8.3+ 2.9+ c21.19+    5.29          ; goto 1: ENT, goto, −
qqf 12.11+3.28                         ; 1
qq  13.33                              ; 2: 86
[00000 00011 1 00000 00000 2 00000 00110 3 22020 00000]
qq  8.3+27.9+ c21.19+   14.29          ; begin 4: ENT, of st
qq  8.3+29.9+ c77.19+  d33.29          ; begin 3: COMPL HEAD, unspec
qq  8.3+28.9+ c21.19+   13.29          ; begin 2: ENT, beg body, −
qq  8.3+28.9+ c18.19+   11.29+   20.39 ; begin 1: ENT OUT, beg clean, begin
qqf 12.11+3.28+31.34                   ; 1
qq  1.26                               ; 4
qq  7.31+1.33                          ; 2: 93
[00000 00011 1 00000 00000 2 00000 04112 3 33131 00000]
qqf −520, qq d39.19−512                ; for 3: SEARCH STATEMENT
qq  8.3+27.9+ c77.19+  d33.29          ; for 2: COMPL HEAD, unspec
qq  8.3+22.9+ c31.19         +  138.39 ; for 1: FOR, −, for
qqf 12.11+3.28+1.20                    ; 1
qq  13.33+1.20                         ; 2: 98
[00000 00011 1 00000 00000 2 30000 00110 3 22020 00000]
qq  8.3+27.9+ c77.19+  d33.29          ; if 5: COMPL HEAD, unspec
qq  8.3+ 2.9+ c70.19+   34.29          ; if 4: CH, ifex, −
qq  8.3+ 2.9+ c18.19+   34.29+  129.39 ; if 3: ENT OUT, ifex, ifex
qq  8.3+ 2.9+ c70.19+   35.29          ; if 2: CH, ifst, −
qq  8.3+ 2.9+ c18.19+   35.29+  130.39 ; if 1: ENT OUT, ifst, ifst
qqf 5.4+27.31+1.33                     ; 1
qq  1.3+13.33                          ; 4
qq  5.4+1.9                            ; 2: 106
[00343 00002 1 00000 00000 2 00000 00110 3 55050 00000]
```

```
b11: qq c73.19−1.19−c76.19          ; modifier 2: value (c46)
qq  8.3+25.9+ c12.19+11.29          ; own 1: SET BLOCK, beg clean
qqf 17.32                           ; 1: 109
[00000 00000 1 00000 00000 2 00000 00010 3 00100 00000]
qq  8.3+ 6.9+  c4.19+        508.39 ; integer 4: OUT, −, opint
qq 8.3+15.9+c74.19+ d.29+d47.39−2.39; integer 3: FIRST SPEC, value yes, spec int−2
qq  8.3+17.9+ c13.19      +    9.39 ; integer 2: SET DECL, −, own integ
qq  8.3+19.9+ c12.19+11.29+    5.39 ; integer 1: SET BLOCK, beg clean, decl integ
qqf 1.28+15.33                      ; 1
qq  1.0−1.7                         ; 4
qq  1.25+13.33                      ; 2: 116
[04444 44400 1 00000 00000 2 00000 20010 3 33130 00000]
qq  8.3+ 6.9+  c4.19+        509.39 ; real 4: OUT, −, opreal
qq  8.3+15.9+ c74.19+ d.29+   51.39 ; real 3:FIRST SPEC, value yes, spec real−2
qq  8.3+17.9+ c13.19      +   10.39 ; real 2: SET DECL, −, own real
qq  8.3+19.9+ c12.19+11.29+    6.39 ; real 1: SET BLOCK, beg clean, decl real
qqf 1.28+15.33                      ; 1
qq  1.0−1.7                         ; 4
qq  1.25+13.33                      ; 2: 123
[04444 44400 1 00000 00000 2 00000 20010 3 33130 00000]
qq  8.3+ 6.9+  c4.19+        510.39 ; Boolean 4: OUT, −, op boolean
qq  8.3+15.9+ c74.19+ d.29+   52.39 ; Boolean 3: SET DECL, value yes, spec bool − 2
qq  8.3+17.9+ c13.19+         11.39 ; Boolean 2: SET DECL, −, own bool
qq  8.3+19.9+ c12.19+11.29+    7.39 ; Boolean 1: SET BLOCK, beg clean, decl bool
qqf 1.28+15.33                      ; 1
qq 1.0−1.7                          ; 4
qq  1.25+13.33                      ; 2: 130
[04444 44400 1 00000 00000 2 00000 20010 3 33130 00000]
qq  8.3+14.9+c75.19+ 0.29+   12.39  ; procedure 4: SEC SPEC, val no, proc−simple spec
qq  8.3+14.9+c74.19+ 0.29+   61.39  ; procedure 3: FIRST SPEC, value no, spec proc−2
qq  8.3+16.9+c14.19      +   38.39  ; procedure 2: ADD DECL PROC, −, simple−proc
qq  8.3+16.9+c11.19+11.29+   42.39  ; procedure 1: SET BLOCK PROC, beg clean, proc
qqf 1.28+15.33                      ; 1
qq  1.15                            ; 4
qq  1.19+13.33                      ; 2: 137
[00000 00000 1 00000 40002 2 00000 00010 3 33130 00000]
qq  8.3+14.9+c75.19+ 0.29+    8.39  ; array 4: SEC SPEC, value no, array−simple spec
qq  8.3+14.9+c74.19+ 0.29+   59.39  ; array 3: FIRST SPEC, value no, spec array−2
qq  8.3+24.9+c15.19      +    7.39  ; array 2: ADD DECL, −, simple−array decl
qq  8.3+24.9+c12.19+11.29+   13.39  ; array 1: SET BLOCK, beg clean, real array
qqf 1.28+15.33                      ; 1
qq  1.15                            ; 4
qq  1.19+13.33                      ; 2: 144
[00000 00000 1 00000 40002 2 00000 00010 3 33130 00000]
b13: qq 1.19−50.35+7.39             ; modifier 1: specification
                                    ; spec−2 is placed in pos.19 (1c74, 1c75)
```

```
[3.8.66]
[GIER ALGOL 4, Pass 3, Page 18, main control table]

qq  8.3+14.9+c74.19+ 0.29+   65.39  ; switch 2: FIRST SPEC, value no, spec switch−2
qq  8.3+23.9+c83.19+11.29+  d19.39  ; switch 1: SWITCH, beg clean, begin proc
qqf 1.28+1.32                       ; 1
qq  13.33                           ; 2: 149
[00000 00000 1 00000 00000 2 00000 00010 3 22120 00000]
qq  8.3+ 6.9+ c4.19+        511.39  ; string 2: OUT, −, opstring
qq  8.3+14.9+c74.19+ 0.29+   53.39  ; string 1: FIRST SPEC, value no, string−2
qqf 13.33                           ;
qq  1.1−1.7                         ; 2: 153
[02222 22200 1 00000 00000 2 00000 00000 3 11010 00000]
qq  8.3+14.9+c74.19+ 0.29+   54.39  ;label 1: FIRST SPEC, value no, spec label−2
qqf 13.33                           ; 1: 155
[00000 00000 1 00000 00000 2 00000 00000 3 11010 00000]
qq  8.3+12.9+c46.19+ 0.29+   66.39  ; value 1: VALUE, value no, undeclspec − 2
qqf 9.33                            ; 1: 157
[00000 00000 1 00000 00000 2 00000 00000 3 10010 00000]
qq 12.3+27.9+c78.19+d33.29          ; ; 7: SEMICOLON, unspec
qq  4.3+31.9+c76.19                 ; ;6: SPEC COMMA, −, −
qq  8.3+28.9+ c1.19                 ; ;5: NEXT, −, −
qq  −515, qq d20.19−498             ; ;4: SEARCH IN EXPRESSION
qq  8.3+30.9+ c1.19                 ; ;3: NEXT, −, −
qq  4.3+31.9+c16.19                 ; ;2: FORMALLIST, −,
qq  4.3+28.9+c20.19                 ; ;1: DECL, −, −
qqf 3.9+9.13+15.20+7.29−1.35+1.38   ; 1
qq  15.4+31.10+15.15+1.20+3.28+7.32+1.34+1.38 ; 4
qq  7.10+5.14+13.18+1.20+3.28+3.31+7.35+1.38  ; 2: 167
[04444 04477 1 70656 62131 2 70000 00770 3 77537 30070]
qq   8.3+28.9+c77.19+d44.29         ; end 3: COMPL HEAD, spec gen
qq  −515, qq d21.19−501             ; end 2: SEARCH IN EXPRESSION
qqf −515, qq d21.19−501             ; end 1: SEARCH STATEMENT
qqf 3.9+1.20+3.28+7.34+1.38         ; 1
qq  15.4+3.7+1.33                   ; 2: 172
[02222 02211 1 00000 00000 2 10000 00110 3 00131 00010]
qq  −514, qq d22.19−508             ; else 2: SEARCH IN EXPRESSION
qqf −514, qq d22.19−508             ; else 1: SEARCH STATEMENT
qqf 1.8+1.20+1.38                   ; 1
qq  15.4+7.7                        ; 2: 176
[02222 22210 1 00000 00000 2 10000 00000 3 00000 00010]
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 19, main control table]

```
qq  4.3+27.9+c77.19+d33.29          ; (5: COMPL HEAD, unspec
qq  8.3+ 2.9+c21.19+ 30.29          ; (4: ENT, of exp
qq 12.3+ 2.9+c26.19+ 33.29+  160.39 ; (3: LEFT PARENT, subex, (
qq  4.3+ 2.9+c18.19+ 31.29+   38.39 ; (2: ENT OUT, (call, beg call
qq  4.3+21.9+c16.19                 ; (1: FORMAL LIST, −, −
qqf 31.5+3.7+1.16+3.31              ; 1
qq  1.11+3.31                       ; 4
qq  1.0−1.9+3.28                    ; 2: 184
[03333 33322 1 04000 01000 2 00000 00220 3 55000 00000]
b25: qqf −515, qq d20.19−498        ; semicolon 7: SEARCH STATEMENT
qq  4.3+27.9+c77.19+d33.29          ; :6: COMPL HEAD, unspec
qq  4.3+36.9+ c1.19                 ; :5: NEXT, −, −
qq  4.3+ 9.9+ c4.19       +    8.39 ; :4: OUT, −, decl label
qq  −549, qq d31.19−511             ; :3: SEARCH IN EXPRESSION
qq  4.3+ 8.9+ c4.19       +    8.39 ; :2: OUT, −, decl label
qq  4.3+27.9+ c4.19       +    8.39 ; :1: OUT, −, decl label
qqf 7.3+1.6+3.28+1.36               ; 1
qq     + 1.9+3.31+1.36              ; 4
qq  7.3+5.8+3.31                    ; 2: 194
[03330 03024 1 00000 00000 2 00000 00110 3 66000 05000]
qq  −538, qq d28.19−511             ; step 1: SEARCH IN EXPRESSION
qqf 7.3+1.6                         ; 1: 196
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  −548, qq d29.19−512             ; until 1: SEARCH IN EXPRESSION
qqf 7.3+1.6                         ; 1: 198
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  −538, qq d30.19−511             ; while: SEARCH IN EXPRESSION
qqf 7.3+1.6                         ; 1: 200
[01110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  −530, qq d24.19−505             ; ] 1: SEARCH IN EXPRESSION
qqf 7.3+1.6                         ; 1: 202
[101110 01000 1 00000 00000 2 00000 00000 3 00000 00000]
qq  4.3+27.9+c77.19+d33.29          ; [5: COMPL HEAD, unspec
qq  4.3+ 2.9+c18.19+ 21.29+   40.39 ; [4: ENT OUT, [subscr, [
qq  4.3+ 2.9+c18.19+ 22.29+   40.39 ; [3: ENT OUT, [left or subs, [
qq  4.3+ 2.9+c18.19+ 20.29+   40.39 ; [2: ENT OUT, [left, [
qq  4.3+ 2.9+c17.19+ 37.29          ; [1: DECL ENT, [arr, −
qqf 1.4+1.24+3.31                   ; 1
qq  7.3+7.7+3.31                    ; 4
qq  1.4+3.9+3.28                    ; 2: 210
[04443 44422 1 00000 00000 2 00001 00220 3 55000 00000]
```

```
qq  −529, qq d23.19−497             ; ,7: SEARCH IN EXPRESSION
qq  8.3+24.9+ c1.19                 ; ,6: NEXT, −, −
qq  4.3+14.9+c76.19                 ; ,5: SPEC COMMA, −, −,
qq  4.3+24.9+ c1.19                 ; ,4: NEXT, −, −
qq  4.3+21.9+c72.19                 ; ,3: FORMAL, −, −
qq  4.3+12.9+c76.19                 ; ,2: SPEC COMMA, −, −
qq  4.3+17.9+ c1.19                 ; ,1: NEXT, −, −
qqf 7.3+3.7+3.15+21.21              ; 1
qq  7.3+3.7+7.15+1.24               ; 4
qq  7.3+3.7+3.13+1.21               ; 2: 220
[07770 07700 1 00265 50101 2 03004 00000 3 00000 00000]
qq  4.3+27.9+c77.19+d33.29          ; := 5: COMPL HEAD, unspec
qq  4.3+ 2.9+c18.19+ 27.29+   71.39 ; := 4: ENT OUT, := for, := for
qq  6.3+ 4.9+ c4.19       +   76.39 ; := 3: OUT, −, :=
qq  6.3+ 4.9+c18.19+  6.29+   77.39 ; := 2: ENT OUT , assign, first:=
qq  4.3+ 2.9+c84.19       +  153.39 ; := 1: SWITCH ASSIGN, −, of switch
qqf 1.4+1.23+3.31                   ; 1
qq  1.22+3.31                       ; 4
qq  1.4+3.9+3.28                    ; 2: 228
[00003 00022 1 00000 00000 2 00410 00220 3 55000 00000]
qq                                  ; not used
qq  −546, qq d27.19−511             ; then 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                         ; 1:231
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]
qq  8.3+32.9+c22.19       +    0.39 ; trouble 7: PROC END, − , end proc
qq  8.3+33.9+c16.19                 ; trouble 6: FORMAL LIST, −, −
qq  8.3+32.9+c23.19       +   41.39 ; trouble 5: AN OUT, −, trouble
qq  8.3+32.9+c20.19                 ; trouble 4: DECL, −, −
qq  8.3+33.9+ c1.19                 ; trouble 3: NEXT, −, −
qq  8.3+32.9+ c4.19       +   41.39 ; trouble 2: OUT, −, trouble
qqf −513, qq d32.19−473             ; trouble 1: SEARCH STATEMENT
qqf31.5+31.10+27.15+1.18+15.23+59.31+7.38; 1
qq  1.10+3.17+1.19+3.24                 ; 4
qq  1.10+31.16+9.21+1.25+3.31       ; 2: 241
[01111 1111 1 71323 36434 2 13154 21110 3 33000 01110]
qq  −538, qq d25.19−509             ; do 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                         ; 1: 243
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]
```

[3.8.66]
[GIER ALGOL 4, Pass 3, Page 21, main control table]

```
qq  8.3+ 6.9+ c4.19+        506.39  ; abs 1: OUT, −, abs
qqf 1.0−1.7                         ; 1:245
[01111 111 00 1 00000 00000 2 00000 00000 3 00000 00000]

qq  8.3+27.9+c77.19+d44.29          ; code 2: COMPL HEAD, specgen
qq  8.3+36.9+ c4.19       +  154.39 ; code 1: OUT, −, code
qqf 3.9+3.28+1.37                   ; 1
qq  13.33                           ; 2: 249
[00000 00011 1 00000 00000 2 00000 00110 3 22020 00100]

qq  8.3+37.9+c18.19+ 16.29+   28.39 ; core 1: ENT OUT, core, core code
qqf 1.28                            ; 1: 251
[00000 00000 1 00000 00000 2 00000 00010 3 00000 00000]

qq  8.3+27.9+c77.19+d33.29          ; case 3: COMPL HEAD, unspec
qq  8.3+ 2.9+c18.19+ 40.29+  165.39 ; case 2: ENT OUT, case st, case st
qq  8.3+ 2.9+c18.19+ 39.29+  d48.39 ; case 1: ENT OUT, case exp, case exp
qqf 14.5+13.33                      ; 1
qq  1.9+3.28+13.33                  ; 2: 256
[00111 00002 1 00000 00000 2 00000 00220 3 33030 00000]

qq  −551, qq d38.19−511             ; of 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                         ; 1: 258
[01110 01100 1 00000 00000 2 00000 00000 3 00000 00000]

qq  8.3+ 6.9+ c4.19+        507.39 ; round 1: OUT, −, round
qqf 1.0−1.7                         ; 1:260
[01111 11100 1 00000 00000 2 00000 00000 3 00000 00000]

qq  8.3+ 6.9+ c4.19+        501.39 ; entier 1: OUT, −, entier
qqf 1.0−1.7                         ; 1: 262
[01111 11100 1 00000 00000 2 00000 00000 3 00000 00000]

qq  4.3+38.9+c82.19+999.29+   27.39 ; beg code 1: CODE, code begin, code end
qqf 7.34+1.36                       ; 1: 264
[00000 00000 1 00000 00000 2 00000 00000 3 00111 01000]
```

```
qq  c60.19+ 3.29+    3.39, ; 265: true: LOGIC VALUE, true, lit bool
qq  c60.19+ 0.29+    3.39, ; 266: false: LOGIC VALUE, false, lit bool
qq  c29.19+ 6.29+  146.39, ; 267: × : BINARY, −, ×
qq  c29.19+ 6.29+  147.39, ; 268: / : BINARY, −, /
qq  c29.19+ 6.29+  149.39, ; 269: ↑ : BINARY, −, ↑
qq  c29.19+ 6.29+  148.39, ; 270: : : BINARY, −, :
qq  c29.19+ 1.29+  492.39, ; 271: < : BINARY, −, <
qq  c29.19+ 1.29+  493.39, ; 272: ≤ : BINARY, −, ≤
qq  c29.19+ 1.29+  494.39, ; 273: = : BINARY, −, =
qq  c29.19+ 1.29+  495.39, ; 274: ≥ : BINARY, −, ≥
qq  c29.19+ 1.29+  496.39, ; 275: > : BINARY, −, >
qq  c29.19+ 1.29+  497.39, ; 276: ≠ : BINARY, −, ≠
qq  c29.19+ 1.29+  156.39, ; 277: ∧ : BINARY, −, ∧
qq  c29.19+ 1.29+  157.39, ; 278: ∨ : BINARY, −, ∨
qq  c29.19+ 1.29+  159.39, ; 279: ≡ : BINARY, −, ≡
qq  c29.19+ 1.29+  158.39, ; 280: => : BINARY, −, =>
qq  c29.19+ 6.29+  143.39, ; 281: mod: BINARY, −, mod
qq  c29.19+ 1.29+  150.39, ; 282: shift: BINARY, −, shift
qq  c62.19,                ; endpass: ENDPASS, −, −
qq  4.3+21.9+c72.19        ; param delim 2: FORMAL, −, −
qq  −543, qq d41.19−511    ; param delim 1: SEARCH IN EXPRESSION
qqf 7.3+3.7                ; 1
qq  1.21                   ; 2: 287
[01110 01100 1 00000 00000 2 02000 00000 3 00000 00000]
qq  −542, qq d26.19−509    ; )2: SEARCH IN EXPRESSION
qq  1.1+18.9+c72.19        ; )1: FORMAL, −, −
qqf 1.21                   ; 1
qq  7.3+3.7                ; 2: 291
[02220 02200 1 00000 00000 2 01000 00000 3 00000 00000 ]

d1: qq [pass sum]          ;

d e22=k−e14,e47=j          ; set load parameters
b k=e23,i=0                ; load segment word 5
i=5e21                     ;
qq e16.9+1d1.19−e16.19+1.20+c63.39f ;
e                                   ;

e                          ; final end pass 3
s
```

[13.6.66]

[GIER Algol 4, pass 4, page 1]

```
b k=e22+e14, i=e16−e47, a22, b15, c48, d20 ; drum block head pass 4
i=e16                                      ;

[Use of indicator:

Name:       true=   init to:   comment:

in proc   : NTA      f         t whenever inside a proc declaration
in head   : LTB      f         Set to f on end proc or after begin.
                                  Tested on any declaration, bounds,
                                  end stack code, begin block, or, specs and
                                  if f then butput (end head); in head:= t;
active    : LZA      f         Used by the bypass logic which generates
                                  jumps around procedure declarations and
                                  sets the return point from local declaration
in block  : LOB      f         The word end bits is used as a stack of bits which
                                  keeps track of the current interpretation of
                                  begin. On ends endbits is shifted right 1 and
                                  bit 0 set to 1 if endblock or endproc, to 0
                                  if end clean, and in block is set accordingly.
                                  A begin is a begin clear if −, in block.
                                  After begins are processed endbits is
                                  shifted 1 left and inblock:= bit (0) = 1;
warning   : LPB      f         Used by the logic which generates while label
                                  and prepass.
in trouble: LQB      f         Used to prevent output of literals when in
                                  trouble.
marks of current actionword are set in RC by the central logic]

[Predefinitions, mostly output values]
d3 = 194 ; goto bypass
d4 = 195 ; bypass label
d5 = e20 ; initial top of use stack, not output
d8 = 16  ; end decl
d9 = 511 ; search, stack value
d11= 64  ; take array
d12= 80  ; formal array
d13= 15  ; end head
d14= 16  ; end block, input value
d19= 2   ; end pass
d20= 24  ; end bound head

[The following a, b, and c names do not appear in natural order
c43  (after c9),  c44  (after c17),  c45  (after c31),  a17  (after c22),
b15  (after c14),  c47(after c13), c48(after c19), a2, a22(after a)]
```

```
b:   qq       [endbits]    ; c21, 1c21, c26, 1c26
[1]  qq       [owns]       ; table
[2]  qq       [variables]  ; 2c20,16c20,c25,3c25,3c31,5c31,6c31,table
[3]  qq       [locals]     ; c18,1c20,2c20,15c20,1c25,6c36,table
[4]  qq       [counter]    ; 7c22,1c23,2c23,c27,2c28,c29,1c29,c30,2c31

c1:
b1:  pmn [par 3]DX         ; stack out next:
     hs  c7                ;   stack(par 3);
c2:
b2:  pmn [par 1]DX         ; out next:
     hs  e3           NZ   ;   Raddr:= par 1; if R ≠ 0 then output (R);
c3:  pmn(e1)   X   1       ; next:
     hs  e2           LA   ;   Raddr:= next byte;
a:   ga  b3    V       NT  ; after CR: if Raddr > maxinterest then
     ps  c3−1  , hv  e3    ;     begin set return (next); goto output end;
b3:  bs  d14   V d1    NZ  ;   if R = 0 then
     ps  a−1   , hv  a19   ;     begin set return(after CR); goto count CR end;
     ps  c3−1  , hv  e3    ; start: R:= set RC (table[Raddr]);
a21: pmn(b3)   X d2    IRC ; rep: par 1:= part 1 (R);
a22: ga  b2                ;   if LA then
     gt  b4    V       LA  ;     begin
     gt  r     , hvn −1    ;     par 3:= part 2 (R); par 4:= part 4 (R)
     ck  20    , ga  b1    ;     end;
b4:  gt  b13   , hvn  −1   ;   M:= R:= 0; goto part 2 (R);

c4:  pmn d3    DXV     IZA ; set active: output (goto bypass);
                           ;   active:= t; return;
c5:  arn d4    D       IZA ; set not active: output (bypass label);
     hv  e3                ;   active:= f; return;


c6:  arn(b1)   D           ; output par 3: output (par 3);
     hv  e3                ;   return;


c7:  gr  p1           M    ; stack: stack top:= stack top + 1;
     pp  p1    , it (b5)   ;   core[stack top] M:= R;
                           ;   if stack top + 2 > use top then
c8h: bs  p2    , hs  e5    ; stack alarm: mess (⊰<stack⊱, 2, 0);
     hr  s1    , qqn e34   ;   return;


c9:  arn p     , pp  p−1   ; byte unstack: R:= store [stack top];
     hr  s1                ;   stack top:= stack top − 1 return;


c43: arn −d13  D       ITB ; set in head: in head:= t;
     mt  c43   , hv  e3    ;   Raddr:= end head; goto output;
```

```
[Search use stack: search in use stack from use top to first block stop
 for identifier given in Raddr. Goes back with:
 if not found: R unchanged, entry = address of first blockstop, s = 0,
 if     found: R = 0, entry = addr where found, s = number of subscr]

c10: gs  b6    , ps  a1    ; search use stack: save s; s:= subcr table;
b5:  hv  d5[usetop]        ;   goto core [use top]; comment return to here
b6:  hv  −1    t   1       ;     with entry, R and s set; goback;
                           ; subscr table: comment
a1:  gs  b7    , hr  b6    ; 0   A search in the use stack terminates
     gs  b7    , hr  b6    ; 1   always in the entry in this table
     gs  b7    , hr  b6    ; 2   coresponding to the number of indices
     gs  b7    , hr  b6    ; 3   found (or 0).
     gs  b7    , hr  b6    ; 4   The following 3 formats may be encountered
     gs  b7    , hr  b6    ; 5   in the use stack during search:
     gs  b7    , hr  b6    ; 6   blockstop:  qq       , hs  s
     gs  b7    , hr  b6    ; 7   cancelled entry  qq
     gs  b7    , hr  b6    ; 8   normal entry: ca <ident>, hsn s<no of subscr>

a2:  ca        , hsn s −1  ; entry word
a3:  qq        , hs  s     ; use block stop


c11: ps  c2−1              ; literal: set return (out next);

c12: hs  c13               ; copy lit: copy 1;
     hs  c13               ;   copy 1;
     hs  c13               ;   copy 1;

c13: arn(e1)   t   1       ; copy 1:
     hs  e2            LA  ;   Raddr:= next byte;
     hv  e3            NQB ;   if −, in trouble then goto output;
     hr  s1                ;   return;
```

[GIER Algol 4, pass 4, page 4]

```
b  a5                      ; block for finis pass 4

c47: grn e20           MC  ; finis pass 4: stack [top core] MC:= 0; inf 1:= 0;
                           ;   M:= 0;
a1:  pm  d15   t   1       ;   for i:= base stack,
     psn(a1)   VX  d18 LC  ;        i+1 while marks [i] ≠ Cmark do
     hv  a1                ;       inf 1:= inf 1 + 1;

     gs  2e4   , hs  c16   ;   Raddr:= next byte;
                           ;     comment firstident − 1;

a2:  gm  1e20  t  −1   MB  ;   for i:= topcore step −1 until first ident−1 do
     nc (a2)   , hv  a2    ;     pass 5 table [i] MB:= 0;
     hs  e3            IQB ;   output (Raddr); trouble:= true;

a3:  hs  c13               ; set std proc:
     hv  a5    X       NT  ;   for Raddr:= next byte while Raddr > 511 do
     ga  a4    , ck  10    ;     begin
a4:  gm[ident] X       MC  ;     ident:= Raddr;
     is (a4)   , it  s512  ;     pass 5 table[ident] MC:=
     pa (a4)   , hs  c13   ;       M + inpos 29 (next byte) +
     ck  20    , ac (a4)   ;       in pos 9 (ident − 512);
                           ;     M:= in pos 39 (ident);
     hv  a3                ;     end;
a5:  ck  −10   , hs  e3    ;   output (M shift 10);
     srn(1b)   D   2       ;   owns:= owns + 2;
     sr  3e4   , hs  e3    ;   output(−owns − max block level);
     arn 1b    , hs  e3    ;   output(owns);
     hhn e29               ;   R:= 0; goto new segm;


e
```

```
c14: pa  4b            IQB ; trouble: counter:= 0; introuble:= t;
                           ;
a4:  hs  c16               ; loop trouble: Raddr:= next not CR;
     hv  a4            LT  ;   if Raddr > max out of trouble then
     ga  b15   , it  d6    ;     goto loop trouble;
b15: bs  −1    , hv  a4    ;   if Raddr ≤ max literal then
     it (b15)  , bs  d7    ;     begin set return (loop trouble)
     ps  a4−1  , hv  c12   ;       goto copy lit end;
     qq (e1)   t   −1  IQB ;   reset input; in trouble := f;
     hv  c3                ;   goto next;




c16: arn(e1)   t   1       ; next not CR:
     hs  e2            LA  ;   Raddr:= next byte;
     hr  s1            NZ  ;   if R ≠ 0 then return;
a19: arn a5    , sc  e4    ; count CR:
     hsn e3                ;   output (0);
     hv  c16               ;   goto next not CR

a5: m                  1   ;

c17: pm  d8    DV      LOB ; begin: if −, inblock then goto decrease ends
     pa  b2    , hv  c21   ;   begin par 1:= 0; goto decrease ends end;
     hs  c43           NTB ; unstack block: if −, in head then set in head;
     hs  e3    X           ;   output(end decl);
a6:  arn p     , pp  p−1   ; loop unstack: Raddr:= unstack byte;
a7:  hh  a10          LZ   ; test block byte: if R = 0 then goto block stop;
     nc  d9           NT   ;   if Raddr ≠ search then
     ps  a6−1  , hv  e3    ;     begin set ret(loop unstack); goto output end;


a8:  arn p     , pp  p−1   ; search use: Raddr:= unstack byte
     hv  a7            NT  ;   if Raddr < 512 then goto test block byte;
     hs  c10               ;   search use stack;
     hh  a9            NZ  ;   if found then
b7:  gr [entry]        M   ;     begin cancel(entry); Raddr:= stack byte end;
a9h: arn p1    , ps  a8−1  ;   set return (search use); goto output;

a10h:hv  e3    , hs  c9    ; block stop:
     ga  b14   , hs  c9    ;   last decl:= byte unstack;
     ga  b10   , hs  c9    ;   locals 1:= byte unstack;
     pi  0.3   V 959   NRB ;   variables 1:= byte unstack;
     ga  b11   , hv  c1    ;   if LRB then goto stack out next;
     ga  b11   , hv  c20   ;   inhead:= f; goto out of block
```

```
c44: pp  p−1               ; no par type pr: stack top:= stack top − 1;
c18: hs  c13               ; par type pr: copy 1:
     qq (e1)   t   −1      ;   reset input; locals := locals + 1;
     qq (3b)   V   1       ;   goto out of block;

c19: pp  p−1               ; no par pr: stack top:= stack top − 1;
c48: arn(b12)   D −1   ITA ; par pr: proclevel:= proclevel − 1;
                           ;   in proc:= proclevel ≥ 0;
c20: hs  c5            LZA ; out of block: if active then set not active
     srn 3b    , sr  2b    ;   output (−locals − variables);
     hs  e3                ;   output (−locals);
     srn 3b    , hs  e3    ;   comment base work and variable address
b8:  can 0[block level]t−1 ;   block level:= block level − 1;
     hv  c47               ;   if blocklevel = 0 then goto finis pass 4;
     hv  a12           LTA ;   if in proc then collaps usage stack
                           ;     begin
     hsn c10               ;     R:= 0; search use stack; found:= t;
     grn(b5)   t   −1  MB  ;     stack inuse (special stop);
     ps (b7)   , gs  b9    ;     i:= entry; comment last block stop;
     ps  s1    , gs  b5    ;     top use:= i + 1
                           ;   loop collaps: if R ≠ 0 then search use stack;
a11: hs  c10           NZ  ;     if −, found then stack en use (R);
     ud  a14           NZ  ;     i:= i − 1;
b9:  arn −1    t   −1      ;     R:= use stack[i];
     hv  a11           NB  ;     if R ≠ special stop then goto loop collaps;
a12:                       ;     end;
b10: it [locals 1] , pa 3b ;   locals:= locals 1;
b11: it [variables1],pa 2b ;   variables:= variables 1;

c21: arn b     , cl  1     ; decrease ends: unstack end bit;
     gr  b             IOB ;   block:= first end bit = 1;
     hv  c32           LRA ;   if LRA then goto declare proc;
     hvn c2            IZA ;   active:= t; goto outnext;
                           ;   comment NRA = begin;

c22: hs  c16               ; beg subscr: Raddr:= next not CR;
a17: qq (e1)t−1[excbya16]  ;   reset input;
     hv  c23           LTA ;   if −, in proc then goto counter out;
     hs  c10               ;   search use stack;
     hv  c23           LZ  ;   if found then goto counter out;
     is (4b)   , it  s1    ;   part 2 of use entry word:= counter + 1;
     pt  a2    , ar  a2    ;   R:= R + use entry word;
   [a14 exec. by 1a11,7c25];   use top:= use top − 1;
a14: gr (b5)   t   −1  MA  ;   store[use top] MA:= R;
     it (b5)   , bs  p2    ;   if usetop < stack top + 2 then
     hh  c8                ;   goto stack alarm;


c23: hs  c9    , qq  c2−1  ; counter out: M:= byte unstack;
     pm (4b)   DX   1      ;   R:= counter + 1; set return(out next);
     gm  4b    , hv  e3    ;   counter:= M; goto output
```

```
c24: hs  c43           NTB ; end proc: if −, in head then set in head
     hs  c5            LZA ;   if active then set not active;
     hs  c13               ;   copy 1; in head:= f;
b12: arn −1[proclev]D 1ITC ;   proc level:= proc level + 1; inproc:= t

c25: arn 2b    , hs  c7    ; end block:
     arn 3b    , hs  c7    ;   stack(variables); stack(locals);
     arn(b14)  D           ;   stack (last decl);
     pa  2b    , hs  c7    ;   variables:= locals:= last decl:= 0;
     psn(b8)   t   1   IZA ;   block level:= block level + 1;
     arn a3    , it (3e4)  ;   if block level > max block level then
     bs (b8)   , gs  3e4   ;     max block level:= block level;
     ud  a14           NTA ;   active:= t;
     pa  3b    , hsn c7    ;   if inproc then stackuse (useblock stop);
     pa  b14   , arn a5    ;   stack(0); R39:= 1;
                           ;
c26: ar  b     , ck  −1    ; end clean: stack end bit (R39);
     gr  b             IOB ;   block:= first end bit = 1;
     ck  −1                ;   if endbit 40 = 1 ∨ block level > 32 then
     bs (b8)   t   32  NO  ;     mess(⟨<begin ends⟩, 2, 0);
     hs  e5                ;   goto outnext;
     hv  c2    , qqn d10   ;

c27: arn 4b    , pa  4b    ; start count:
     ps  c2−1  , hv  c7    ;   stack(counter); counter:= 0; goto outnext;


c28: hs  c43           NTB ; bounds: if −, inhead then set inhead;
     hs  c4            NZA ;   if −, active then set active;
     pa  4b    , hv  c2    ;   counter:= 0; goto outnext;

c29: bs (4b)   t   7       ; count index:
     pa  4b    , hs  e5    ;   if counter > 7 then
     hv  c30   , qq  sd16  ;     begin counter:= 0; mess(⟨<index⟩, 0, 1) end;

c30: qq (4b)   t   1       ; count param: counter:= counter + 1;
     hv  c2                ;   goto out next;
```

```
c31: pa  b14   , can(4b)   ; declare array: last decl:= 0;
     arn a15   , hv  a22   ;   if counter = 0 then
     hs  c35   , qq  d20   ;     begin R:= is undeclared; goto rep end;
     gs  b1    , hs  c6    ;   test decl; par 3:= endboundhead;
     arn 4b    , hs  c7    ;   output par 3; stack counter;
     ac (2b)   D   1       ;   variables:= variables + counts + 1
     ps  c2−1  , hv  c34   ;   set return(outnext); goto stack and copy

c45: hs  c35   , qq  i+2   ; declare label: test decl;
     gs  a16   , hv  c34   ;   set return to here after one stack and copy in a16;
     pa  a16   t  c34      ; here: reset a16;
     hv  c2                ;   goto outnext;

c32: pa  b5    t d5    LTA ; declare proc:
     arn(b2)   D           ;   if −, in proc then usetop:= initial usetop;
     pa  b14   , hs  e3    ;   last decl:= 0; output (par 1);

c33: hs  c35   , qq  c3−1  ; declare: test decl; set return (next);
     hs  c43          NTB ;   if −, in head then set in head;

c34: hs  c16               ; stack and copy: Raddr:= next not CR;
     hh  a16           NT  ;   if R < 0 then
     hs  c7                ;     begin
     hs  e3                ;     stack (Raddr); output (Raddr);
b13h:it  1     , qq ([par 4]);   counts [par 4]:= counts [par 4] + 1;
                           ;   goto stack and copy;
a16: hv  c34   , ud  a17   ;     end; comment a16 is changed and reset by c45.
     hr  s1           NRB ;   reset input;
     arn d9    D      NTA ;   if LRB ∧ inproc then stack (search);
     hv  c7           NTA ;     comment array and switch has LRB;
     hr  s1                ;   return;


c35: arn(b1)   D           ; test decl: Raddr:= par 3;
b14: ca [last decl], hr s1 ;   if Raddr = last decl then return;
     ga  b14   , hv  c7    ;   last decl:= Raddr; goto stack;
a15: qq c33.19+8.29+2b.39, ;   action word for undeclared array
```

```
c36: hs  c13               ; spec1: comment array, switch, or unspec;
     arn(e1)   , hs  c10   ;   copy 1; Raddr:= last byte;
     grn(b7)         LZ    ;   search usage;
     bs  s511        LRB   ;   if found then cancel (entry);
     hv  c1                ;   if number of indices = 0 ∨ −, array spec then
     hs  c4          NZA   ;     goto stack outnext;
     arn s     D           ;   if −, active then set active;
     ac (2b)   D   2       ;   Raddr:= number of indices;
     qq (b2)   t d11  −d12 ;   variables:= Raddr + 2 + variables;
     ps  c1−1  , hv  e3    ;   par 1:= par 1 + take array − formal array
                           ;   set return (stack out next 1); goto output;

c37: hs  c4            NZA ; spec 2: comment value;
                           ;   if −, active then set active;
c38: ps  c1−1  , hv  c13   ; spec 3: comment therest;
                           ;     set return (stack out next); goto copy 1


c39: hs  c13               ; copy code: copy 1:
     arn(e1)   , tk  −30   ;   CRcount:= CRcount + last byte × 2↑(−30);
     ac  e4                ;
                           ;
a18: arn(e1)   t   1       ; loop code:
     hs  e2          LA    ;   Raddr:= next byte;
     hv  c2          LT    ;   if Raddr > 512 then goto outnext;
     ps  a18−1 , hv  e3    ;   set return (loop code); goto output;


                           ; forelem:
c40: hs  c6            LPB ; assign: if warning then output(par 3);

c41: pm  r                 ; simple for: warning:= f; goto outnext;

c42: hv  c2            IPB ; set warning: warning:= t; goto outnext;

s                          ;
```

```
[CONTROL TABLE: Each word holds up to four byte. The central input (c3:)
 unpacks all four if the word is ,marked otherwise only the first two, it
 leaves always the marks in RC when they are used to distinguish variants
 of actions common to more input bytes.
 The 4 bytes are usually used as:
    output, action, stack value or extra output, where to count identifiers.
 In the comments ++ indicates same meaning as input and _ indicates
 extra output. Marks are given as A (,), B (f), C(f, ), or N (none).]

d d2= i-1 [base table]          ; Input:        out,act,stack,count,marks

qq  84.9+c11.19                 ;  1 literal in: ++, literal, -, -, N
qq  85.9+c11.19                 ;  2    -    re: ++, literal, -, -, N
qq  86.9+c11.19                 ;  3    -    bo: ++, literal, -, -, N
qq  87.9+c11.19                 ;  4    -   str: ++, literal, -, -, N

d d7= i-d2 [input byte from here to d6 terminates trouble]

qq       c33.19+ 52.29+ 2b.39  , ;  5 decl simpin: -, declare  , ++, var, A
qq       c33.19+ 53.29+ 2b.39  , ;  6     -    re: -, declare  , ++, var, A
qq       c33.19+ 54.29+ 2b.39  , ;  7     -    bo: -, declare  , ++, var, A
qq  21.9+c45.19+  7.29+ 3b.39  , ;  8 decl label : tab:, decl lab, ++, loc, A
qq       c33.19+ 56.29+ 1b.39  , ;  9 decl own in: -, declare    , ++, own, A
qq       c33.19+ 57.29+ 1b.39  , ; 10     -    re: -, declare    , ++, own, A
qq       c33.19+ 58.29+ 1b.39  , ; 11     -    bo: -, declare    , ++, own, A
qq  28.9+c31.19+ 60.29+ 2b.39f , ; 12 decl arr in: begbnds in,decl arr, ++, par 3, C
qq  29.9+c31.19+ 61.29+ 2b.39f , ; 13     -    re: begbnds re,decl arr, ++, par 3, C
qq  30.9+c31.19+ 62.29+ 2b.39f , ; 14     -    bo: begbnds bo,decl arr, ++, par 3, C

qq       c26.19                 ; 15 end clean  : -, end clean, -, -, N
qq  17.9+c25.19                 ; 16 end block  : ++,end block, -, -, N
qq  14.9+c28.19                 ; 17 end bounds : ++,bounds   , -, -, N
qq  18.9+c24.19                 ; 18 end proc   : ++,end proc , -, -, N
qq  19.9+c24.19                 ; 19 endtypeproc: ++,end proc , -, -, N

qq   1.9+c17.19                 ; 20 begin      : beg block, begin, -, -, N
qq        c3.19                 ; 21 ;          : -, next, -, -, N
qq 172.9+ c2.19                 ; 22 do         : ++,outnext, -, -, N
qq 173.9+ c2.19                 ; 23 then st    : ++,outnext, -, -, N
qq 174.9+ c2.19                 ; 24 else st    : ++,outnext, -, -, N
qq 175.9+ c2.19                 ; 25 of st      : ++,outnext, -, -, N
qq 176.9+ c2.19                 ; 26 end case st: ++,outnext, -, -, N
qq   4.9+c39.19                 ; 27 code end   : beg code, copy code, -, -, N
qq  12.9+ c2.19                 ; 28 core code  : ++,outnext, -, -, N
qq   5.9+c28.19                 ; 29 corecodeend: ++,bounds , -, -, N
qq       c17.19+ 20.29       f , ; 30 endspec    : -, begin, specs, -, C

d d6= i-d2-1 [input bytes from d7 to here terminates trouble]

qq 177.9+c27.19                 ; 31 end call   : ++, start count, -, -, N
qq 178.9+c27.19                 ; 32 ] one      : ++, start count, -, -, N
qq 179.9+c27.19                 ; 33 ] more     : ++, start count, -, -, N
```

```
                                  ; input         : out,act,stack,count,marks

qq 180.9+c30.19                   ; 34 call param : ++, count param, −, −, N
qq 181.9+c29.19                   ; 35 comma 1    : ++, count index, −, −, N
qq 182.9+c29.19                   ; 36 comma 2    : ++, count index, −, −, N
qq 183.9+c29.19                   ; 37 boundcolon : ++, count index, −, −, N
qq  22.9+c23.19                   ; 38 beg call   : ++, counter out, −, −, N
qq   3.9+c23.19                   ; 39 beg func   : ++, counter out, −, −, N
qq  23.9+c22.19                   ; 40 [          : ++, beg subscr , −, −, N
qq       c14.19                 f  ; 41 trouble    : − , trouble    , −, −, B

qq  36.9+c48.19+ 51.29+ 3b.39   , ; 42 dclparprno : begpr no, par pr,++ , loc, A
qq  37.9+c18.19+ 48.29+ 3b.39   , ; 43     −   in : begpr in, par tppr, ++, loc, A
qq  38.9+c18.19+ 49.29+ 3b.39   , ; 44     −   re : begpr re, par tppr, ++, loc, A
qq  39.9+c18.19+ 50.29+ 3b.39   , ; 45     −   bo : begpr bo, par tppr, ++, loc, A
qq  35.9+c48.19+  6.29+ 3b.39 f, ; 46 decl switch: beg swch, par pr  , ++, loc, C
qq  40.9+c19.19+ 47.29+ 3b.39   , ; 47 decl procno: begpr no, no parpr, ++, loc, A
qq  41.9+c44.19+ 44.29+ 3b.39   , ; 48     −    in: begpr in, nopartppr, ++, loc, A
qq  42.9+c44.19+ 45.29+ 3b.39   , ; 49     −    re: begpr re, nopartppr, ++, loc, A
qq  43.9+c44.19+ 46.29+ 3b.39   , ; 50          bo: begpr bo, nopartppr, ++, loc, A
qq  40.9+c19.19+  8.29+ 3b.39   , ; 51     −    un: begpr no, no par pr, ++, loc, A

qq  76.9+c38.19+1023.29         , ; 52 spcsimp in : form in   , spec 3, ++, −, A
qq  77.9+c38.19+1022.29         , ; 53    −    re : form re   , spec 3, ++, −, A
qq  78.9+c38.19+1021.29         , ; 54    −    bo : form bo   , spec 3, ++, −, A
qq  79.9+c38.19+1020.29         , ; 55    −   str : form st   , spec 3, ++, −, A
qq   9.9+c38.19+1019.29         , ; 56    −    la : form la   , spec 3, ++, −, A
qq  68.9+c37.19+1018.29         , ; 57 specval in : val  in   , spec 2, ++, −, A
qq  69.9+c37.19+1017.29         , ; 58    −    re : val  re   , spec 2, ++, −, A
qq  70.9+c37.19+1016.29         , ; 59    −    bo : val  bo   , spec 2, ++, −, A
qq  80.9+c36.19+1015.29       f, ; 60 specarr in : formarr in, spec 1, ++, −, C
qq  81.9+c36.19+1014.29       f, ; 61    −    re : formarr re, spec 1, ++, −, C
qq  82.9+c36.19+1013.29       f, ; 62    −    bo : formarr bo, spec 1, ++, −, C
qq  75.9+c38.19+1012.29         , ; 63 spcproc no : form pr no, spec 3, ++, −, A
qq  72.9+c38.19+1011.29         , ; 64    −    in : form pr in, spec 3, ++, −, A
qq  73.9+c38.19+1010.29         , ; 65    −    re : form pr re, spec 3, ++, −, A
qq  74.9+c38.19+1009.29         , ; 66    −    bo : form pr bo, spec 3, ++, −, A
qq  13.9+c36.19+1008.29         , ; 67    −    la : form pr la, spec 1, ++, −, A
qq  11.9+c36.19+1007.29         , ; 68 un spec    : form un   , spec 1, ++, −, A
qq  10.9+c36.19+1006.29         , ; 69 spec gm    : form gm   , spec 1, ++, −, A

qq 184.9+c41.19                   ; 70 simple for : ++, simple for, −, −, N
qq 185.9+c40.19+192.29          , ; 71 := for     : ++, forelem, whilelabel, −, A
qq 186.9+c40.19+192.29          , ; 72 step elem  : ++, forelem, whilelabel, −, A
qq 187.9+c40.19+192.29          , ; 73 while elem : ++, forelem, whilelabel, −, A
qq 188.9+c42.19                 f  ; 74 while     : ++, set warning, −, −, B
qq 189.9+c42.19                 f  ; 75 end ass   : ++, set warning, −, −, B
qq 190.9+c40.19+193.29          , ; 76 :=         : ++, assign, prepass, −, −, A
qq 191.9+c40.19+193.29          , ; 77 first:=    : ++, assign, prepass, −, −, A
d d1= i−d2−1 [max interest]       ;
```

```
[30.6.66]

[GIER Algol 4, pass 4, page 12]

d15= i−1, d18= −d15        ; define stack base addresses
                           ; init stack:
a20: grn sc46          MC  ;   for i:= top core step −1 until stack base + 4 do
     bs  s511  , hv  a21   ;     store[i] MC:=0; goto first byte is read;
     ps  s−1   , hv  a20   ;   comment end block;
c46: pa  3e4   t   2       ; start pass 4: max block level:= 2;
     qq (e1)   t   1       ;   stack top:= stack base;
     pi  128   , pp  d15   ;   inproc:= inhead:= active:= warning:=
     arn d19   D           ;     in trouble:= in block:= f;
     hs  e3                ;   output (end pass);
     ps e20−c46,  hvn a20  ;   goto init stack;

b k=e31, i=0    ; load texts
i=e32           ;

d10: tbegin ends;
d16: tindices;  ;

e32=i           ;
e               ;

d17: qq [pass sum]         ;
d e22=k−e14, e47=j         ; set load parameters
b k=e23,i=0                ; load segment word 6
i=6e21                     ;
     qq e16.9+1d17.19−e16.19+4.24+3.29+c46.39 ;
e                                             ;

e                          ; final end pass 4
s
```

```
b k=e22+e14, i=e16−e47, a27, b27, c34, d28 ; drum block head;
d i=e16                                    ;

[Specification of address variables. b1 − b7, see page 4
                    referred by:
b8h:  outrel        1c10, exc. from a13h
b9:   n             1a5
b10:  ident act     c8, c9, c23, c26
b11:  byte          c10, 2c12, 7c12,1a8,3a14,1c20,c21,6c21,2c22,5c22,6c22,11c22
b12   entry         9c12, 2a6
b13   decl act      13c12
b14   stack ref     1a10, a13,
b15   L             b15, c33
b16         unused
b17         unused
b18         unused
b19         unused
b20h  rel ref       12c12
b21   min ident     16c24, 6c34
b22   i             c24, 1c24, 5c24, 10c24
b23   decl top      7c22, 14c24, 2c25, 2c26, 3c26, c30
b24   anon          7c24, 8c24, 11c24
b25   specrel       6c28, c32
d26   doperel       3c28
b27   n subscr      1c28 ]


[Predefinitions:]

d1=196   ; CR outputvalue
d2=404   ; Base for kind type. Added at output by outputdescr.
d3=87    ; Maximum value of input bytes which are treated by pass 5.
d6=190   ; Input value for := Used by output descr to test
d7=191   ;    −    −    −  first:= for assign to procedure value
d9=471   ; Base for specifications in output
d11=12   ; Kindtype = undeclared
d15=445  ; Simple integr. Outputvalue used by take value and
         ;   take array.
d18=199  ; Take array. Output value
d19=48   ; Kind type = dope descr.
d20=27   ; Miminum input byte where two last bits gives type − 1.
d21=198  ; begin proc.
d22=132  ; output (take value int) − input (take value int)
d25=40   ; input value begproc no
d27=197  ; beg block
d28=96   ; undeclared.4
```

[The input table and the central logic of pass 5.

When entered at next the central logic inputs a byte and treats it in one of 4 ways depending on size:

1. byte = 0: CARRET action, return to next.

2. byte > 511 (i.e. negative): Jumps to the current identifier action. There are 3 possible actions on an identifier:
   1. It is declared, i.e. entered in the table with the current description given in the variable decl.
      This identifier action is set by begin block or begin proc.
   2. The table entry is checked for double declaration.
      This action is set by enddecl, endbounds, end proc, and end core code, and is explicitly performed by label colon
   3. The corresponding description is output from the table.
      This action is set by end head and core code.

3. byte > max interest. The byte is output, return to next.

4. byte ≤ max interest. The byte byte refers to the input table and is used as follows:
   If byte ≥ min with type then byte : 4 is used as index in the table otherwise byte itself is used.
   The input table word is added to the byte and gives hereby a word in R in one of two formats:
   1. Declaration word (R positive):
      <kind type>.9 + <decl act>.19 + <how to get relative address>.29 + <marks to store in table>. marks.
      The variable decl is set to kind type.39 and curr block.33 is added to decl.
      The marks are saved in PC, relref is set to part 3, M is cleared and a jump to decl act is performed.
   2. Output and action word (R negative):
      qq <output value> + 512, hv <action>
      output value is output and a jump to <action> (via the table) is performed.]

[14.7.66]
[GIER Algol 4, pass 5, page 3]

[THE DECLARATION TABLE. Each identifier is represented by one word which
 is referenced by the pass 2 representation of the identifier and which
 holds the current valid declaration of that identifier.
 Three main formats are distinguished:

 Standard identifiers:
   qq 0.0 + identpart.9 + Lpart.19 + Tpart.29 + chainpart.39, f;
      identpart: pass 2 representation − 512. Only used during unstacking
         of a redeclared identifier.
      Lpart: Representation of the std proc in pass 5 output. Is initialized
         to 0 and set to L before it is output first time.
         L is initialized to e20 and decreased by 2 before each time it
         is used to set a Lpart.
     Tpart: Number in identifier table. Used by finis pass 5
     chainpart: Reference to identifier with next higher Tpart.
         Used by finis pass 5.
   These words are initialized by finis pass 4 after it has initialized:

 Not used:
   qq 0 f;
     At each endblock or end proc all identifiers declared in that block
     are reset to this value.

 All declared or used identifiers:
   qq 1.0 + identpart.9 + relpart.19 + refpart.28 + blockpart.33 +
                                       declpart.39 (marks: see below)
     identpart: As above.
     relpart: Block relative address
     refpart: 0 or reference to a word in the stack which holds
         For arrays with knownno of subscr: Desciption of the dope vector
         For procedures with parameters   : The specificationlist.
         For procedure values   : Description of the procedure
     block part: Block number
     decl part: Kind and type
   These words are set whenever an identifier is declared after d possible
     stacking of a declaration in an outer block.

 DISTRIBUTION OF IDENTIFIERS: six primary formals are recognized
 by the marks and bit 0 as follows:

                                  Output: bit0 marks
 (1) Normal identifier: <base+decl part><relpart><blockpart>   1 0   1
 (2) Own or
 (2a) Undeclared: <base+declpart><relpart>0   1 1   0

 (3) Not used: Is replaced by 1a and then output after alarm   0 0   1

 (4) Array with subscr: As 1 followed by dope descr (also as 1)or
 (4a) Procedure with param: As 1 followed by specification list   1 0   0

 (5) Proc value: If following delimiter is:=or first:= then as (1)
 (5a)   else the word referenced by ref. part (as 1 or 4a) 1 1   1

 (6) Standard procedure: Lpart   0 1   1]

```
e16: qq [stackref block0.9+ min L .19+ std proc chain.39]   ; set by start and
     qq [silly, must be 0, see 1c33]                        ; fin pass 5

c1:  hs  c2               ; copy 4: copy 1;
     hs  c2               ;   copy 1;
     hs  c2               ;   copy 1;

c2:  hs  c3               ; copy 1:
     hv  e3               ;   output (take byte); return


c3:  arn(e1)   t   1      ; take byte:
     hr  s1           NA  ;   Raddr:= next byte; return;

a1h: hv  e2    , ac  e4   ; treat CR: CR count:= CR count + 1;
     arn d1    DV         ;   output (out CR); return;

c5:  arn(s)    D          ; byte out: Raddr:= part 1 (store[s])
     hv  e3               ;   goto output;

c4:                       ; end bound head:
bh:  pm  b6    , is[arr rel];   output (arr rel + 2);
     arn s2    D           ;     comment addr of coeff;
     hs  e3                ;   output (array count)
     qq (b16)  , hs  c5    ;    output (kind type pt);
     cln −6    , ud  c9    ;      comment array type
     ck  −4    , hs  e3    ;    set descr;

c6:  pa  b16   , hh  c11   ; beg bounds: array count:= 0; goto next 1;

c7:  hs  c3    , qq i−1    ; copy code:
     hv  e3           NT  ;   for Raddr:= take byte while Raddr < 512 do
     tk  −30   , sc  e4    ;     output (Raddr);
     tk  30    , hs  e3    ;   CRcount:= CRcount − Raddr×2↑(−30);
     hh  c11               ;

c8:  pa  b10   V   c21     ; set check: ident act:= check decl; goto next;
c9:  pa  b10   t   c10     ; set descr: identact:= output descr; goto next;
a19: qq  d23   , hr  s1    ;   d23 is used from 8c12

[b1 − b4 contains the current (i.e. last used) relative address for the 4
 kinds of addressing]


                          ; referred by:
b1:  ps [local addr] t −1  ; 1c6, 13c24, 1c26, 6c26, b8
b2:  ps [var addr]   t −1  ; 6c26, 1c28, 2c28, 3c28, 7c28, b8
b3:  ps [own addr]   t −1  ; b8, 1c34
b4:  ps [form addr]  t  1  ; 5c26, 1c27, 5c27, b8
b5:  qq [currblock]  −1.33 ; curr block 14c12, a8, 10c22, 15c24, 3c27, 7c34
b6:  qq [decl=ident pt.9+relpt.19+refpt.28+blockpt.33+kindtypept.39];
                          ; 14c12, 4c25, 1c29, 6c6
b7:  qq [dopedescr=relptofdope.19+dopedescr.39] d19.39 ; 2c28, 4c28  ;
```

```
c10: arn(b11)          ITA ; output descr: R:= set TA (table[byte]);

a2:  gt  b8    X       IRC ; normal out: outrel:= set RC (relpart(R)); swap;
     hv  a9            LRC ;   if LRC then goto proc val or std;

a3:  cln −6    , ck  −4    ; cont out: Raddr:= decl part of (M);
     ar  d2    DV      LTA ;   if NTA then
     arn(e1)   , hv  a8    ;     begin R:= same byte; goto undeclared end;
a4:                        ;   Raddr:= Raddr + base kind type;
b8:  hs  e3    , nc[outrel]; cont out 1: output (Raddr);
     arn s     D           ;   comment a4h executed from a13h;
     hs  e3                ;   output (outrel);
     cln −5    V       NRA ;   Raddr:=
     qqn       V           ;     if NRA then (− block part of (M)) else 0;
     ck  −5    , mt  r     ;
     hs  e3            LTA ;   if LTA then output (Raddr);
     hh  a10           NRC ;   if NRC then goto cont from stack;
a5:
b9:  bt [n]    t   −55     ; count output: n:= n + 1; if n = 10 then
     pa  b9    , ud  a7    ;     begin n:= 0; inf 1:= inf 1 + 1 end;
c11h:qq        , ps  i     ; next 1: set return (next);
                           ;   comment a return to next will do as next 1;
c12: pmn(e1)   X   1       ; next:
     hs  e2            LA  ;   Raddr:= next byte;
     ga  b11   V       NZ  ;   if R = 0 then goto treat CR;
     arn a17   , hh  a1    ;   byte:= Raddr;
b10: hv [ident act]    LT  ;   if Raddr > 511 then goto ident act;
b11: bs [byte] t d3        ;   if byte > of interest then goto output;
     hv  e3                ; unpack from intab: entry:= base in tab +
     bs (b11)  t d20       ;     (if byte > min with type then byte:4+d23 else byte);
     ck  −2    , ar  a19   ;   R:= byte + set QC (intab [entry]);
     ga  b12   , arn(e1)   ;   if NT then set declaration:
b12: ar [entry]t  d4   IQC ;     begin
     hv  a6            LT  ;     set PC; comment marks from in tab [entry];
     cl  10    , gt  b20   ;     decl act:= part 2 (R);
     ga  b13   X       IPC ;     relcount:= part 3 (R);
     ar  b5    , gr  b6    ;     decl:= part 1 (R) shift (−30) + curr block;
b13: hvn[decl act] X       ;     R:= R × 2↑10; M:= 0;
                           ;     goto decl act
                           ;     end;
a6:  mb  511   D           ;   Raddr:= bits (1, 9, R);
     hs  e3            NZ  ;   if Raddr ≠ 0 then output (Raddr);
     hh (b12)              ;   goto right half (in tab [entry]);


a7:  qq (2e4)  t   1       ;   comment executed from 1a5
```

```
a8:  ar  a18   , ar  b5    ; undeclared:
     gr (b11)  , hs  e5    ;   table [byte]:= R + curr block + undecl;
     hv  c10   , qq  sd5   ;   mess(‹<undeclared›, 0, 1);
                           ;   goto output descr;

a9:  hh  a14           NTA ; proc val or std: if NTA then goto std proc;
     hs  c3    , qq  i−1   ;   for Raddr:= take byte while R = 0 do
     qq (e1)   V   −1  NZ  ;     treate CR
     arn a17   , hh  a1    ;   reset input;
     nc  d6    , ca  d7    ;   if Raddr = v:= v Raddr = vfirst := then
     hv  a3            IRA ;     begin RA:= f; swap; goto cont out end;

a10h:cln −11   , cln −9    ; cont from stack:
     ck  −1    , ga  b14   ;   stack ref:= ref part of (M);
a11:                       ;
b14: pmn[stack ref]t d8 IRC; next word: M:= set RC (stack [stack ref]);
     hv  a2    X       NRA ;   if −, RA then begin swap; goto normal out end;
a12: cln −5    , ck  −5    ; next spec: Raddr:= bits (35, 39, M);
     hv  a13           LZ  ;   M:= M : 2↑5;
     ar  d9    D           ;   if R ≠ 0 then
     ps  a12−1 , hv  e3    ;     begin Raddr:= Raddr + basespec;
                           ;     set return (next spec); goto output end;
a13: hv (b14)  D d17   NRB ;   if NRB then more words:
                           ;     begin stackref:= stack ref+1; goto next word end
                           ;   goto count output;
a14h:hv  a5    , udn a4    ; std proc:
     ps  a4    , hr  s1    ;   if outrel ≠ 0 then
                           ;     begin set return(contout 1+1); return end;
b15: it e20[L] t   −2      ;   relpart of table [byte]:= L:= L − 2;
     pt (b11)  , hv  c10   ;   goto output descr;

                           ; Constants:
a15: qq 1.0 + 31.33        ;   block part mask
a16: qq        1.33        ;   unit for block count
a17: qq             1.39   ;
a18: qq           d11.39   ;   undeclared
```

```
c20: hs  c3                ; label colon:
     ga  b11               ;   byte:= take byte;

c21: arn(b11)  , gt  b     ; check decl: R:= table[byte]; arr rel:= part 2(R)
b16: qq[array count] Vt1NC ;   comment for use by end bound head;
     cl  −6    X       NB  ;   if array or proc then array count:= array count + 1;
     ca  d28   , hs  e5    ;   if decl part (R) = undecl then mess(ǂ<+declǂ,0,1);
     hh  c11   , qq  sd12  ;   goto next 1;


c22:                       ; declare:
b20h:ga  b6  , ud [rel ref];   ident part := Raddr; rel part:=
     it  s     , pt  b6    ;     rel addr[rel ref] + rel incr[rel ref];
     arn(b11)  V       IRC ;   R:= set RC (table [byte]);
a20: qq (3e4)  t   1       ;   if R = 0 then
     pm  b6    V       LZ  ; new declaration: table [byte] MPC:= decl;
     pm  a15   , hh  a21   ;   else
     gm (b11)          MPC ;     begin
a21: hh  c11   , cm  b6    ;     if block part and bit 0 (table[byte]) ǂ
                           ;        block part and bit 0 (decl) then
     gr (b23)  V   1   MRC ;     redeclaration:
     arn a18   V           ;       begin decl top:= decl top + 1; inf2:= inf 2 + 1;
     ps  a20−1 , hvn c30   ;       stack[decl top] MRC:= R; R:= 0;
                           ;       set return (new declaration); goto check stack
     ar  b5    , ar (e1)   ;       end;
     gr (b11)          MA  ;     double declared:
     ps  c12−1 , hv  c12   ;       table[byte] MA:= byte + curr block + undecl;
                           ;     end
                           ;   set return (next); goto next;
```

```
c23: pa  b10   t   c21     ; end proc: ident act:= check decl;
     hs  c2                ;   copy 1;

c24: pa  b22   V   e20+1   ; end block:
a22: grn(b22)          MB  ;
b21: arn[min ident], pm a15;   for i:= max ident step −1 until min ident+1 do
b22: cm [i]    t   −1      ;     if block part (table[i]) = curr block ∧
     hv  r−1               ;       bit 0(table[i]) = 1 then table[i] MB:= 0;
     nc (b22)  , hv  a22   ;
b23: arn[decl top] d8  IRC ; un stack declaration:
     ga  b24   V       LT  ;   for R:= set RC [stack[decl top]) while LT∨LRC do
     ga  b24   , hv  a23   ;     begin
b24: gr [anonymous]    MRC ;     table [identpart (R) + (if LT then 0 else 512]
     hv (b23)  D   −1      ;       NRC:= R;
a23: hv (b24)  D  512  LRC ;     decl top:= decl top − 1
                           ;     end;
     gt  r     , pp  −1    ; block stop now in R: spectop:= part 2(R)
     qq (b23)  t   −1      ;   decl top:= decl top − 1; R:= in pos block (−1);
a24h:srn a16   , ac  b5    ; set block no: curr block:= curr block + R;
     ac (b21)  , hv  c12   ;   table [min ident]:= table [min ident] + R;
                           ;   goto next;

c25: qq  d21   , hs  c5    ; beg proc: byte out (beg proc);
     srn d25   D           ;   out (byte − beg proc no)
     ar (e1)   , hs  e3    ;   ref part:=
     is (b23)              ;     decl top − decl base + 2;
     arn sd14  D           ;
     ck  −19   , ac  b6    ;
c26: pan b10   X   c22     ; beg block: ident act:= declare; decl top:= decl top+1
     gm (b23)  t   1   MA  ;   stack [decl top] MA:= 0
     it  p     , pt (b23)  ;     + in part 2 (spec top);
     hs  c30   , qq  1     ;   check stack; form addr:= 1;
     gs  b4    , hs  c3    ;   local addr:= 0; var addr:= take byte;
     ga  b2    , pa  b1    ;   copy 1; comment base work;
     hs  c2    , qq  c12−1 ;   set return (next);
     arn a16   , hh  a24   ;   R:= inpos block (1); goto set block no;


                           ; take formal:
c27: qq  d15   , hs  c5    ;   byte out (simple formal);
     arn(b4)   D   1       ;   form addr:= form addr + 1;
     hs  e3                ;   output(form addr);
     srn b5    , tk  24    ;   output (−curr block);
     hs  e3    , qq  c12−1 ;   set return (next)
     qq (b4)   V   −1  LPB ;   if NPB then array specification:
     ps  d16   , hv  c28   ;     begin set return (take array); goto array decl end;
     arn d22   D           ;   form addr:= form addr − 1; Raddr:= last byte +
     ar (e1)   , hv  e3    ;     take value; goto output;
```

```
c28: hs  c3                ; array decl: n subscr:= take byte;
     ga  b27   , sc  b2    ;   var addr:=
     it (b2)   , pt  b7    ;     rel part of dope:= doperel:=
     it (b2)   , pa  b26   ;     var addr − n subscr;
     ck  16    , ar  b7    ;   stack [spec top − 1] MB:=
     gr  p−1           MB  ;     dope descr + inpos block no (n subscr);
     gp  b25   , pp  p−1   ;   spec ref:= spec top; spec top:= spec top − 1;
     qq (b2)   t   −1      ;   var addr:= var addr − 1;
c29:                       ;
b25: arn[spec ref] Dt d17  ; par pr decl:
     ck  −19   , ac  b6    ;   ref part := spec ref − stack base − 1;

c30: is (b23)  , it  s−512 ; check stack:
     bs  p−512 , hr  s1    ;   if decl top < spec top then return;
     ps  a25   , hv  e5    ;   mess (‹<stack›, 2, 0);


c31: qq  d18   , hs  c5    ; take array: byte out (take array);
b27: qq [n subscr], hs c5  ;   byte out (n subscr);
b26: qq [doperel], hs c5   ;   byte out (doperel);
a25=i−1 [mess descr]       ;   goto next 1;
     hh  c11   , qqn e34   ;
d16=c31−1[ret. to take arr];


a26: bs  s511  , hv  a27   ; pack spec: if spec pos > 0 then
                           ;     begin spec top:= spec top − 1;
     pp  p−1   , gm  p     ;     store[spec top]:= 0;
     hs  c30   , qq  −30   ;     check stack; spec pos:= −30
                           ;     end;
a27: tk  s     , ps  s5    ;   stack [spec top] MC:= stack [spec top] −
     sc  p     V       MA  ;     R × 2↑spec pos; spec pos:= spec pos + 5;
                           ;   goto next spec;
c32: gp  b25   , ps  5     ; specs: speec ref:= spec top; spec pos:= 5
     hsn c3    X           ;   R:= take byte;
     hv  a26           LT  ;   if R < 0 then goto pack spec;
     acn p             MC  ;   stack [spec top] MC:= stack [spec top];
     qq (e1)   t   −1      ;   reset input;
     ps  c11−1 , hv  c11   ;   set return (next); goto next;

c33: it (b15)  , pt  e16   ; fin pass 5: min L:= L;
     arn 1e16  , hh  e29   ;   R:= descrp pass 5 segm 2; goto new segm
```

[input table: Entries starting qq − are declaration words (see page 2),
 the others are output action words]

```
d4 = i−1 [table base]              ; Input value        comment

qq 511.9+197.9, hv  c26            ;  1 begblock
qq 510.9+213.9, hv  c33            ;  2 end pass
qq 509.9+214.9, hv  c2             ;  3 beg func
qq 508.9+210.9, hv  c7             ;  4 begcode
qq 507.9+211.9, hv  c8             ;  5 end core code
qq  −6.9+ 20.9+ c12.19+ b1.29 f    ;  6 decl switch
qq  −7.9+ 16.9+ c12.19+ b1.29 f    ;  7 decl label
qq  −8.9+d11.9+ c12.19+ b1.29 f    ;  8 decl pr undef

qq  −9.9+ 24.9+ c12.19+ b4.29 f    ;  9 form label
qq −10.9+  8.9+ c12.19+ b4.29 f    ; 10 form gener
qq −11.9+d11.9+ c12.19+ b4.29 f    ; 11 form unsp
qq 500.9+212.9, hv  c9             ; 12 core code
qq −13.9+ 28.9+ c12.19+ b4.29 f    ; 13 form switch

qq 498.9+203.9 , hv   c8           ; 14 endbounds
qq −15.9+  0.9+  c9.19             ; 15 end head    15, 16 and 20 are decla−
qq −16.9+  0.9+  c8.19             ; 16 enddecl     ration words to prevent
qq 495.9+204.9 , hv  c24           ; 17 end block   output action and decl
qq 494.9+205.9 , hv  c23           ; 18 end pr      is free at these inputs.
qq 493.9+206.9 , hv  c23           ; 19 end tppr
qq −20.9+  0.9+ c32.19             ; 20 specs
qq 491.9+207.9 , hv  c20           ; 21 labcolon
qq 490.9+208.9 , hv   c2           ; 22 beg call
qq 489.9+209.9 , hv   c2           ; 23 [
qq 488.9+215.9 , hv   c4           ; 24 end bound head
d23=i−d4−7                         ;
[table entry for the following input bytes is computed as
 d4+d23+byte:4]
qq −28.9+  1.9+  c6.19             ; 28 beg bounds  kind tp pt is array type
qq −32.9+103.9+ c25.19             ; 32 beg switch  32, 36, 40 must be together
qq −36.9+103.9+ c25.19+ b1.29 f,   ; 36 beg par proc  proc type is output as
qq −40.9+103.9+ c25.19+ b1.29 f,   ; 40 beg proc      input − beg proc no
qq −44.9+ 32.9+ c12.19+ b1.29 f    ; 44 decl tppr
qq −48.9+ 36.9+ c29.19+ b1.29      ; 48 declpartppr
qq −52.9+ 40.9+ c12.19+ b2.29 f    ; 52 decl simpel
qq −56.9+ 40.9+ c12.19+ b3.29  ,   ; 56 decl own
qq −60.9+ 44.9+ c28.19+ b2.29      ; 60 decl array
qq −64.9+ 44.9+ c27.19+ b2.29      ; 64 take array
qq −68.9+ 40.9+ c27.19+ b4.29 f    ; 68 take value
qq −72.9+ 52.9+ c12.19+ b4.29 f    ; 72 form tp pr
qq −76.9+ 56.9+ c12.19+ b4.29 f    ; 76 form simpel
qq −80.9+ 60.9+ c12.19+ b4.29 f    ; 80 anon array
qq 428.9+468.9 , hv   c1           ; 84 literal
```

```
d8=i−1 [Define stack base]        ;
d10=−d8+1, d14=−d8+2, d17=−d8−1   ; other stack addresses

[Start pass 5 is overwritten by the stack]

c34: hs  c3                ; start pass 5:
     ga  b3    , hs  c3    ;   own address:= take byte
     gr  e16   , hs  e3    ;   stack ref bl 0:= R:= take byte; output (R);
     hs  c3                ;   std proc chain:= take byte;
     ck  10    , ac  e16   ;   min ident:= Raddr:= take byte
     hs  c3                ;
     ga  b21   , pp (b21)  ;   spectop:= p:= min ident;
     bs  p−d8−510, pp510d8 ;   if spec top − stack base > 510 then
     it  p     , bs d24    ;     spectop:= stack base + 510;
     ps  a25   , hv e5     ;   if spectop < last word pass 5 then
                           ;     mess(⊰<stack⊱, 2, 0);
     ar  b5    , gr (b21)  ;   table[min ident]:= R + curr block;
     qq  d27   , hs  c5    ;   byte out(beg block);
     hv  c26               ;   goto beg block


b k=e31, i=0               ; Load texts
i=e32                      ;
d5:  tundeclared;          ;
d12: t+ decl.;             ;
e32=i                      ;
e                          ;

d24:  qq [pass sum]        ;
d e22=k−e14, e47=j         ; set loading parameters
b k=e23, i=0               ; load segment word 7
i=7e21                      ;
qq e16.9+1d24.19−e16.19+5.24+1.28+c34.39;
e                                       ;

e [final end pass 5]                    ;
s                                       ;
```

[17.10.1967]
[Gier Algol 4, pass 5 std. proc descr., page 1]

[This code is taken in to 1e16 as a segment.
 It sets up the standard identifiers description table to pass 6:
 The chain of standard identifiers set up by pass 4 is followed
 starting with the address left in bits 30 − 39 of e16 by start pass 5
 and ending when a chain 0 is encountered.
 All words which have been used (Lpart ≠ 0) are stored in a list
 of used std identifiers as qq <Lpart>.9 + <Tpart>.19.
 The next segment is now taken to core; it contains section2 of the
 standard procedure library.
 Each of the above words are then treated using Tpart as index in section 2
 of the standard procedures and Lpart as address of where to store
 the description in the pass 6 table.
 Descriptions which refer to a proper standard procedure (NTR part ≠ 0)
 are further used to build up a table of tracks to be encluded in
 the generated code. The track no entered in these descriptions is
 the position in the table of tracks of the entry track number for
 the procedure.
 Descriptions which refer to std variables (NTRpart = 0) will instead
 of entry track get the relative address of the variable as if it
 were in block 0.
 Finally the track list is output.

 Format of an entry in standard procedures section 2:
   For a procedure:
      qq <FTR>.9+<NTR.14+>ETR>.19+<code ref>.23+<rel>.29+<ref>.33+<kind type>.39
      qq <spec n>.9+<spec n−1>.14+<spec n−2>.19 −−−
   For a variable:
      qq <Address relative to displ block 0>.9+<code ref>.23+1.29+<ref>.33>+
         <kind type>.39
      qq <specn>.9 ...

   FTR: first track of the procedure relative to first track of section 3
   ETR: entry track relative to FTR.
   NTR: number of tracks
   rel: entry address on track ETR.
   ref and kind type are references to pass 6 tables

Format of an entry in the pass 6 table:
   For a procedure:
   L: qq <ref>.9+<entry track>.19+<rel>.27+<code output>.33+<kind type>.39
   L+1: qq <pec n>.9 ...
   For a variable:
   L: qq <ref>.9+<block 0 rel>.19+1.27+<kind type>.39
   L+1 qq <specn>.9...]

```
b k=e22+e14, i=1e16−e47, a12, b10, d9 ; drumblock head
i=1e16                                ;

d4=200 ; max no of tracks allowed
d5=50  ; max no of std proc references alowed

a:   grn e20   , arn e16   ; start pass 5 std proc descr:
     ck  10    , ga  e20   ;   min L:= part 2 (e16);
     tk  20    , ga  b2    ;   chain:= part 4 (e16); i := 0;
     it  e20−d5−d5         ;   if min L < top L < 2×max allowed std procs then
     bs (e20)  , hv  a3    ; error: mess(ǂ<std.procs.ǂ, 2, 0);
a1:  ps  d7    , hv  e5    ;   else goto start chain;

a2:  arn(b2)   , ck  −10   ; loop chain: R:= store [chain];
     ga  b2    , tk  20    ;   chain := chainpart (R);
     nc  0                 ;   if L part (R) ǂ 0 then
b1:  gr  d1[i] t   1       ;     begin i:= i + 1; used list [i]:= R shift 10 end;
a3:                        ; start chain:
b2:  ncn[chain], hv  a2    ;   if chain ǂ 0 then goto loop chain;
     grn  d2   , hhn e29   ;   tracks[0]:= 0; goto next segment;
d3:  ppn 0     , ud  b1    ;     comment will come back here;
                           ;   i:=i+1; used list [i]:= 0; i:= 1; t:= 0;
                           ;   p:= number of std proc tracks:= 0; entry := −2;
a4:  pmn d1[i] X   1       ;   for R:= used list[i] while R ǂ 0 do
     hh  a8            LZ  ;   begin i:= i + 1;
     ga  b5    , ck  10    ;   L:= part 1(R); T:= part 2(R);
     ga  b4                ; loop t:
a5: b10: qqd8[entry] t 2   ;   entry:= entry + 2;
b3:  it [t]    t   1       ;   t:= t + 1 if t < T then
b4:  bs [T]    , hv  a5    ;     goto loop t;
     is (b10)  , arn s1    ;   pass 6 table[L+1] := std proc descr [entry + 1]
b5:  is [L]    , gr  s1    ;   R:= std proc descr [entry];
     arn(b10)  , ga  b6    ;   etr:= FTR:= part 1(R);
     ga  b8    , cl  34    ;   pass 6 table kind type [L]:= bits (34, 39, R);
     ck  16    , cl  −4    ;   pass 6 table ref [L]      := bits (30, 33, R);
     ck  −18   , cl  −6    ;   pass 6 table rel [L]      := bits (24, 29, R);
     ck  12    , cl  −4    ;   pass 6 table code ref [L] := bits (20, 23, R);
     ck  10                ;   marks [L]:= 0;
     gr (b5)           M   ;   etr:= etr + bits (15, 19, R);
     cln −5    , ck  −5    ;   Rpos4:= bits (10, 14, R);
     ac  b8    , cln −5    ;   if R = 0 then
     hv  a6            NZ  ;     begin
                           ;     not reference to std proc on tracks:
     ac (b5)           MA  ;     marks [L]:= 2;
     nt (e16)  , it (b8)   ;     pass 6 table block 0 rel [L]:=
     pt (b5)   , hv  a4    ;       etr + rel stack ref 0;
                           ;     end
                           ;   else
a6:  ga  b7                ;     begin
a7:                        ;     for Rpos4:= Rpos4 − 1 while Rpos4 ≥ 0 do
b6:  arn[FTR]  Dt  1       ;       begin
     it (pd2)  , bs (b6)   ;       FTR:= FTR + 1;
     pp  p1    , gr  pd2   ;       if FTR > tracks [p] then
b7:  ncn[Rpos4]t   −32     ;         begin p:= p + 1; tracks [p]:= FTR end
     hv  a7    , qqn d6    ;       end;
d7=i−2[address of message] ;
```

```
     it  p     , qq (b8)   ;     pass 6 table entry track [L]:=
     nt (pd2)  , it (b8)   ;       p + etr − tracks [p]
     pt (b5)   , hv  a4    ;     end
a8h:                       ;   end for R;
b8:  qq [etr]  , is (e20)  ;   marks[minL −2 ]:= 1;
     grn s−2           MB  ;   output (p+1); run track:= run track − p + 1;
     arn p1    D         ;
     sc  13e4  , hs  e3    ;
     arn(e16)  D         ;   output (rel sr 0);
     hs  e3              ;
     arn(13e4) D   2       ; modify entry track so that first track will
     ck  −10             ;   will be run track:
a9:                        ;   j:= top L + 2;
b9:  pm  e20[j]t   −2      ;   for j:= j − 2 while marks [j] ≠ 1 do
     ac (b9)           NC  ;     pass 6 table entry track [j]:=
     hv  a9            NB  ;     pass 6 table entry track [j] + run track;

                           ; output track list:
a10: bs  p511  , hhn e29   ;   for p:= p step −1 until 0 do
     arn pd2   , hs  e3    ;     output (tracks [p]);
     pp  p−1   , hv  a10   ;   R:= 0;   goto next segm;

d1:  qq [used list[0];
d2=d1+d5 [tracks[0]  ;
d8=d2+d4 [stdproc descr[−2];

b k=e31, i=0       ; load text
i=e32              ;
d6: tstd.procs.;   ;
e32=i              ;
e                  ;

d:   qq [pass sum] ;

d e22=k−e14, e47=j ; Set load parameters;
b k= e23, i=0      ; Load segment word 8 and 9
i=8e21             ;
qq 1e16.9+1d.19−1e16.19+1.20+a.39f  ;
qq 2d8.9+63.19+     1.20+ d3.39   f,; length 63 is for debugging, can stay in;
e                  ;

e [final end]      ;
s
```

```
d e49=4     ; tape number := 4;

[After i follows STOPCODE, SUMCODE and a sum character]
if T3
s
```

Order No.: 494

Class: 0.2.2

Type: Book

Author: T. Asmussen, J. Jensen,
S. Lauesen, P. Lindgreen,
P. Mondrup, P. Naur, and
J. Zachariassen

Ed.: December 1967 (E)

THE COMPLETE ANNOTATED

PROGRAMS OF GIER ALGOL 4

Volume II

Tape identification	Contents	Page identification	Number of pages
T4	Tape test	T4 Gier Algol 4	1
	Pass 6	pass 6, page 1–19	19
	Pass 9	pass 9, page 1–20	20
	Sum check	tape number := 5	1
T5	Tape test	T5 Gier Algol 4	1
	Pass 7	pass 7, page 1–21	21
	Pass 8, phase 1	pass 8.1, page 1–6 2,3 (a↑n, a↑x)	6
	Pass 8, phase 2–4	Pass 8, page 1–24 18 (comments) missing	23
	Sum check	tape number := 6	1
T6,L2	Tape test	T2 and L2 Gier Algol 4	1
	Library Processor	Library Processor, page 1–6	6
T7,L3	Tape test	T7 and L3 Gier Algol 4	1
	Library	read integer – system(A)	30
	Sum check	<–e40+1,<e40+1	1
T8,L4	End loading	T8 and L4 Gier Algol 4 page 1–3	3
T9,L5,M1	Merger	T9,L5,M1 Gier Algol 4 page 1–6	6

[Here follows STOPCODE and CLEARCODE]

```
<e49−3, <−e49+5, ×  ;   test tape number
i wrong tape
s                   ;
>                   ;

d e54=13            ;   version number

<e54−e50,           ;   if version number T4 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=9             ;   define version number T2
d e63=10            ;   define version number T3
d e64=e54           ;   define version number T4
d e65=11            ;   define version number T5
d e66=12            ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e54
>                   ;
```

```
[30.6.67]
[GIER ALGOL 4, pass 6, page 1]

b k=e22+e14, i=e16−e47, a30, b40,c70,d20;
i = e16                                  ;
[Input byte values, used otherwise than for entry into the INPUT CONTROL
TABLE:
196  CAR RET 4a18
210  begin code 5a18
412  smallest operand 1a19
467  largest      −   1a1
                      4c

Output byte values in code:
468 − ... 6a17, c66, c9, 1a18]

[Output byte values]
d9=107 [std 2 call]                ;
d11=1 [end call], d15=0 [beg call] ;

b7: qq 1.1−1.21          ; mark 1, bits 2 to 21 (1c2)
b8: qq 1.3−1.15          ; mark 2, bits 4 to 15 (3, 4c2)
b k=e31, i=0             ; Alarm texts
i=e32                    ;
d6: ttype;               ;
d7: tcall;               ;
d8: tsubscripts;         ;
e32=i                    ;
e                        ;
d=1                      ; If d is redefined = 0 we get for KA,KB = 00:
s                        ; operator output, for KA,KB = 10: indicator output

c65:arn(b)    , ga r1    ; STD IDENTIFIER: R:= STD TABLE CONTROL[inbyte];
    pmn       Xt d10  IPC ; go to UNPACK CONTROL WORD;
    hv  c2               ;

c25:pmn(e1)   X    1     ;   procedure outin;
    hs  e2           LA  ;    output(input);
    hv  e3               ;

a12:pa  p     t   d2     ; TYPE ALARM: OPERAND STACK [top]:=
    hs  e5               ;    undeclared; outtext(⟨<type⟩);
    arn(e1)   , qq  s+d6 ;    select error by;
    qq        , ud  13e4 ;    output (last input);
    hs  e7               ;    select normal by;
    qq        , ud  16e4 ;    goto SET NEW OPERAND;
    hv  a3               ;
```

```
b25: qq 1.39              ;
c5: arn b25   , ac  e4    ; CARRET: CRcount:= CRcount + 1;
c:  pmn(e1)   X   1       ; NEXT: inbyte:= input;
    hs  e2            LA  ;    if R ≤ 511 then go to STD IDENTIFIER;
    ga  b     V       NT  ;
    ga  b     , hv  c65   ;
    bs (b)    t   300     ;    if inbyte > limit then
    ck  −2    V           ;      begin
    hh  a1                ;      variant:= in byte mod 4;
    ga  b     , ck  2     ;      in byte:= inbyte : 4
    mb  3     D           ;
a1: ga  b6    , it  d1    ;      end; inbyte:= inbyte + base;
c1:                       ; REPEAT:
b:  pmn[inbyte]tX         ;
b12:sr[top operator]d5IPC ;    if CONTROL TABLE [in byte] <
    pmn(b12)  XVt      LT ;      OPERATOR STACK [top operator] then
                          ;      begin R:= top operator; top:= top − 1 end else
c3: pmn(b)    VXt     IPC ; UNPACK INPUT: R:= TABLE [inbyte];
    hv  r4                ; special output:
    sy  65            NKC ;    65 <inbyte>
    sy(b)             NKC ;
    hv  r4                ;
    sy  66            NKC ;    66 < top operator>
    sy(b12)           NKC ;
    i=i−d−d−d−d−d−d       ;
a26:qq(b12)   t−1         ;
c2: ga  b5    , gt  b1    ; UNPACK CONTROL WORD:
    ck  18    , mb  b7    ;    PA, PB := Marks
    ga  b2    , ck  6     ;    QA, QB, RA, RB:= Bits 6−9
    mb  b8    , ga  b3    ;    b1:= bits 10−19
    ck  6     , mb  b8    ;    b2:= bits 20−27
b6: ar[variant]D      LPB ;
    ga  b4                ;    b3:= bits 28−33
b5: pi  −1    t   1008    ;    b4:= bits 34 to 39 + (if PB then variant else 0);
    hv  r3            LKB ; special output:
    gi  r1            LKA ; indicator
    sy                LKA ;
    i=i−d−d−d             ;
    hv  a3            NRB ;    if RB then CHECKTOP: begin
    sy  67            NKC ; special output:
    sy(p)             NKC ;    67 <stack top>
    i=i−d−d               ;
    arn(p)                ;      if bit(0, OPERAND STACK
    qq (b2)   t  512  LT  ;      [top]) then par 2:= par 2 + 512;
b3: ck [par 3]            ;    if bit (par 3, OPERAND STACK[top])
    hv  a3            LO  ;      = 1 then goto SET NEW OPERAND;
    ck  1                 ;    if bit (par 3 + 1, OPERAND STACK[top])
    hv  a12           NO  ;      ≠ 1 then goto TYPE ALARM;
    arn b3    , ca  1     ;    if par 3 = 1 then goto a13;
    hv  a13               ;
    nc  0     , hv  a12   ;    if par 3 ≠ 0 then goto TYPE ALARM;
    pmn 30    DV          ;    M:= float top; skip;
a13:pmn 541   D           ; a13: M:= round top;
    hs  e3    X           ;    output (M);
    qq (b2)   t   512     ;    par 2:= par 2 + 512;
a3:                       ;    if RA then top operand:= top operand −1;
    pp  p1            LRA ;
```

```
a2: hv  a21           NQB ; SET NEW OPERAND:
b4: pmn[par 4]XDt d2      ; R:= OPERAND STACK [top operand];
a28:pp  p−1   , it  p−1   ;    if QB then begin top operand:=
    bs (b12)  , hs  e5    ;      top operand + 1; Test for stack
    hv  r1    , qqn e34   ;      overflow;
b9: gr  p             MA  ; OPERAND TABLE[par 4]:= R; end;
a21:hv  a4            NQA ;    if QA then begin
a6: hs  c28   , qq  d3    ; SET AND STACK(operator base);
    qq (b12)  t 1         ;    top operator:= top operator + 1;
a4:
b2: arn −1 [par 2] D  LPA ; OUTPUT: if no relation then
                          ;      output (par 2) else
b21:hs  e3[or a8]     LPA ;      goto OUTRELATION;
b1: qq        , hv [par 1];    goto action [par 1];
a8: ga  b26               ; OUTRELATION:
b22:arn[rel]  D           ; output(rel);
    hs  e3                ; NORELOUT; goto action [par 1];
a27:pa  b21   t e3        ; procedure NORELOUT;
b26:arn       D           ; begin no relation:= true;  output(b26)
    hv  e3                ; end;
c10:                      ; PROC WITH PAR: pseudo top:= top operator + 1;
    hs  c25               ;    output (input);  base:= d3;
    hs  c28   , qq  d3    ;    output (input);
    ps  c−1   , hv  c25   ;    goto NEXT;
c11:pm(p)     t   d4 IRC  ; BEGIN CALL:  RC:= marks(AUX TABLE[top operand]);
    pmn(e1)   X 1         ; R:= input;  comment number of actual parameters;
    hs  e2            LA  ; if RA = 0 then begin
    hv  a5            LRA ;   if RA = 1 ∨ R + top operator + 1 = pseudo top
    ar (b12)  D       NRB ;   then  top operator:= pseudo top
    ga  r1    , arn b10   ; else
    ca        t 1     NRB ; begin OPERAND STACK[top operand]:= undeclared;
    ga  b12   , hv  a5    ;       alarm (⫲<call⫳)
    pa  p     t d16       ; end;
    hs  e5                ; end;
a5: arn(p)    , qq sd7    ; R:= AUX TABLE[top operand];
    tk −5     , gt  b37   ;
    ga  b38   , ck  2     ;
    mb  3     D           ;
b37:ga  b14   , pi        ;
    pt  b1    t  c        ;
b38:pm        Dt d2       ;
    gm  p     , it  b15   ;
b14:arn       , hv  a28   ;

[OPERANDSTACK words, jumped to at end call]

b15:qq  d11   , hv  c20   ;     output(end call); goto NEXT;
[1b15] qq  d9 , hv  c20   ;     output(std2call); goto NEXT;
[2b15] pp  p1 , hv  c     ;     no output; goto NEXT;
b16:qq        , hs  c21   ;     output (address); goto STACK;
```

```
c20:ps  c−1              ;    output (par); goto NEXT;
c21:arn(p)    D          ;    procedure out (u);
    pp  p1    , hv  (b21) ;      begin top operand:= top operand − 1;
                         ;      output (u) end;
c14:qq (b12)  t   1      ; ACCEPT: top operator:= top operator + 1;
c12:arn(p−1)  t    d4    ; SPEC:
    tk  17    , tk  −7   ;
    mb  1023  D          ;    output (bits (16, 19, part 2 (
    ps  c3−1  , hv  e3   ;      OPERAND STACK [top + 1]));
                         ;    goto Input action
c13:qq (b12)  t   −1     ; END CALL: top operator:=
    hv  p                ;      top operator − 1; goto OPERAND
                         ;      STACK;

b13:qq 1.19−1.27+1.33−1.39;    mask 3
b19:[0]qq 4   , qq  c17  ;    std. unpack control
[1] qq  4     , qq  c    ;    char−control word
[2] qq  4     , qq  c66  ;
[3] qq  4.9+2.33 t c17   ;
[4] qq  4.9+1.33, qq c17 ;
[5] qq  4.9+2.33, qq c17 ;
c16:arn(b)    , tl −20   ; STD UNPACK:
    tk  30    , hs  e3   ;
c19:arn(b)    , mb  b13  ;    qq trackrel.27 + operand.39
    ar (b3)   t   b19 IPC ;     + unpack table [par 3];
    hv  c2               ;    goto UNPACK CONTROL WORD;

[Stack operator call
                     hs  c15  ,  qq base
The operator in location base+par4 is stacked and the stack is tested]

c27:ps  c−1   , hv  c18  ; ENTER SPEC:
c28:it (b12)  , pa  b10  ; proc SET AND STACK(q);
c15:arn s     , gt  c18  ; proc STACK(q);
c18:pm (b4)  [base]   IRC ;
b10:gm[pseudo top]t 1 MRC ;
    sy  68            NKC ;
    sy (b10)          NKC ;
    sy (b4)           NKC ;
    i=i−d−d−d             ;
    it  p−1   , bs (b10)  ;
    hs  e5                ;
    hv  s1    , qqn e34   ;
c17:it (b3)   , pa  b4   ; UNPACK STD SPECS: par4:= par3;
    hs  c28   , qq  d17  ;     SET AND STACK (d17);
    arn(b)    t 1        ;     R:= STD TABLE [std id+1];
a16:ga  b4    , tk  10   ;     for par4:= part 1 (R) while R≠ 0 do
    pm (b4)   Xt  d17    ;        begin  M:= R ; R:= STD SPEC TABLE[par4]
    hv  a31   X       NRA ;        if LRA then
    tk  28    , ck  12   ;           begin  slow proc: R:= bits (28,39,R);
    ar (b10)         LRB ;           if LRB then  variable list:
    qq (b10)  t −1   LRB ;              begin R:= R+(c14−c12) pos 19;
    ar  d17   X       IRB ;             pseudotop:= pseudotop − 1 end
a31:ck −5     , hs  b10  ;           R:= R + slow proc spec; LRB:= f end
    hv  a16          NZ  ;        swop;  R:= R shift − 5;  pseudostack:
    arn r2           NA  ;        end;
    ac (b10)  , hv  c    ;     if fast proc then add 3pos27 to
[2] qq  3.27             ;        OPERATOR STACK[pseudo top]; go NEXT;
```

```
c23:arn(b4)   D t  100    ; STD SPEC 2:
    hs (b21)              ;    output (par 4 + 100);
c26:                      ; GIER:
    arn(b2)   D           ;    OPERAND STACK [top operand]:=
    ar  b16   ,  ud  b9   ;      qq par 2, hs c21;
    hv  c3                ;    goto UNPACK INPUT;

c31:hs  c25               ; INOUT 4: output (input);
c6: hs  c25               ; INOUT 3: output (input);
c24:hs  c25               ; INOUT 2: output (input);
c7: ps  c-1   , hv  c25   ; INOUT 1: output (input); goto NEXT;

c30:hs  c25               ; DOPE: output (input);
    hs  c25               ;    e1:= input; output(e1);
    arn(e1)               ;    subs:= e1;
    ga  b17   , hv  c     ;    goto NEXT;
c39:pmn(e1)   Xt 1        ; LEFTBR:
    hs  e2           LA   ;    e1:= input;
    pmn(p)    Xt  d4      ;    R:= word 2(OPERAND STACK)
    tk -5     , ga  b23   ;      [top operand];
    ck  12           IOA  ;    OA:= bit 7(R); par4:= bits 23 to 32(R);
    ck  16    , ga  b4    ;    OPERAND STACK [top]:=
b23:pm  -1    D   d2      ;      bits 1 to 4 (R) + base;
    gm  p     , srn(e1)   ;      if OA then
    pa  b17   t -1    LOA ;      subs:= 1;
b17:ca [subs] , hv  a11   ;    if -e1 ≠ subs then
    hs  e5                ;      begin alarm (≠subscripts≠);
    pi  0     , qqs+d8    ;      OA:= false; par4:=
    pa  b4    t 17        ;
a11:qq (b12)  t -1    LOA ;    if OA then top operator:= top operator + 1;
    hs  c28   , qq  d3    ;    STACK OPERATOR(d3);
    qq (b12)  t 1         ;
    hv  c            NOA  ;    if OA then
    pmn d15   DX          ;    output (begin call);
    ps  c-1   , hv  e3    ;    goto NEXT;
c8: arn 1     D           ;  SWITCH DESIG: output(integer);
    ps  c3-1  , hv  e3    ;    goto UNPACK INPUT;
```

[GIER ALGOL 4, pass 6, page 6]

[When the following procedure is entered the operands in top and top + 1
(i.e. p and p − 1) have both been checked to be arithmetic or undeclared.
If necessary the procedure produces output of float top or float next top
to make them of the same type, makes sure that par 2 is 512−marked if the
result is real]

```
c32:arn(p)    , pm (p−1)  ;    procedure ARITHPAIR;
    ck  0     X       IOA ;
    ck  0             IOB ;
    hv  s1            NOC ;    if −, (OA ∨ OB) then return;
    bsn(b2)               ;    R:= 0; if par 2 < 512 then
    qq (b2)   t   512     ;      par 2:= par 2 + 512;
    arn 31    D       NOA ;    if −, OA then R:= float next top;
    arn 30    D       NOB ;    if −, OB then R:= float top;
    hs  e3            NZ  ;    if R ≠ 0 then output (R);
    hv  s1                ;    return;
c33:hs  c32               ; RELATION: ARITH PAIR;
    it (b2)   , pa  b22   ;    rel:= par 2;
    pa  b21   t   a8      ;    no relation:= false;
    pa  p     t   d14     ;    OPERAND STACK [top]:=
    hv  c1                ;      bool result; goto REPEAT

c34:bs (b2)               ;    POS, NEG, ABS
    pa  p     t   d12     ;    if par 2 < 512 then OPERAND
    hv  c1                ;    STACK [top]:= real res; goto NEXT;
c44:it  −1                ; TAKE OPERATOR:
c41:qq (b12)  Vt   1      ; KEEP OPERATOR:
c35:pa  p     t   d13     ; SET REAL RESULT: OPERAND
    hv  c                 ;    STACK [top]:= real res; goto NEXT;
c36:hs  c32               ; PLUS, MINUS: ARITH PAIR;
a10:bs (b2)   , it  d12   ; a10: if par 2 < 512 then OPERAND
a14:pa  p     t   d13     ;    ST [top]:= intres else
a15:arn(b2)   D           ; SET REAL OUT: OPERAND ST [top]:= realres;
    hs  e3                ;    output (par 2);
    hv  c1                ;    goto REPEAT;
c37:hs  c32               ; MULT: ARITH PAIR
    bs (b2)               ;    if par 2 < 512 then
a30:qq (b2)   t   −1      ;      par 2 := par 2 − 1;
    hv  a10               ;    goto a10;

c38:bs (b2)   , ud  a30   ; POWER: if par 2 < 512 then
                          ;    par 2:= par 2 − 1;
    hv  a14               ;    goto SET REAL OUT;
b24:qq −9.4+3.9+c3.19+0.33;    for real
c40:bs (b2)   , hv  c     ;      := for; if par 2 > 512 then
    pm  b24   , gm  (b12) ;      OPERATOR STACK [top]:= for real;
    hv  c                 ;    goto NEXT;
c43:qq (b12)  t   −1      ; STEP ELEMENT DO: topoperator:=topoperator−1;
c42:hs  c32               ; STEP ELEMENT: ARITH PAIR; go to SET REAL OUT;
    arn(b2)   D           ;    output (par2);
    ps  c−1   , hv  e3    ;    goto NEXT;
```

[The following procedure takes the top operand and (1) adds the input parameter to par 2 in case of label type, (2) puts the operator number in bits20−22 into b29, (3) outputs par 2, and (4) outputs the type given in bits 16−19]

```
c46:arn s     , gt  b30   ;
    pmn(p)    Xt  d4      ;    R:= AUX OPERAND [top];
    ck  15                ;    if bit 15 (R) = 1 then
b30:qq (b2)   t   1   LO  ;    par2:= par 2 + q;
    mb  508   D           ;    Extract bits 16 to 22
    ck  −5    , ga  b28   ;    type:= bits 16 to 19
    tk  10    , ck  −7    ;    rator:= bits 20 to 22
    ga  b29               ;
    pmn(b2)   XD          ;    output(par 2);
    hs (b21)              ;
b28:pmn −1[type] DX       ;    output (type)
    hv  e3                ;    end CASEELSE;
c45:bs (b2)   , hv  c3    ; CHECK INTEGER: if par 2 > 512 then
c62:pa  p     Vt  d13     ;    OPERANDSTACK[top]:= real;
c63:pa  p     t d20       ; CHECK UNDECLARED:
    hv  c3                ;    goto UNPACK INPUT;
                          ; CASE COMMA: base:= case;
c22:hs  c46   , qq  1     ; ELSE EXPR: CASEELSE(1)
b29:pm[rator] Vt b27  IRC ;    base:= else; SET OPERATOR:
c47:pm (b29)  t  b31  IRC ; CASE
    gm (b12)         MRC ;    OPERATORSTACK [top]:= TABLE
    pp  p+1   , hv  c     ;    [rator+base]; top operand:=
                          ;      top operand − 1; goto NEXT;
c53:                      ; END CASE:
c48:hs  c46   , qq  1     ; ENDELSEEXP: CASEELSE(1);
    qq (p)    t   d19     ;
    hv  c                 ;    goto NEXT;
                          ; CASE UNDECLARED: t:=undeclar; skip;
c49:bs (b2)   ,   hv c52  ; CASE INTEGER: if par2<512 then skip;
c50:pa  p     Vt  d13     ; CASE REAL: t:=real; OPERAND STACK[top]:=t;
c51:pa  p     t   d20     ;
c52:hs  c46   , qq  0     ; CASE OTHER: CASE ELSE(0);
    qq (p)    t   d19     ;
    hv  c3                ;   go to UNPACK BYTE;
c54:arn  5    D           ; END SWITCH: output (labeltype);
    hs  e3                ;
    arn 36    D           ;    output(proc;);
    ps  c−1   , hv  e3    ;   go to NEXT;
```

```
c55:pmn(p−1)  X   d4      ; ASSIGN: top operator:=
    tk  20    , ck  −7    ;      top operator + 1;
    ga  b4                ;    OPERATOR STACK[top operator]
    hs  c15   , qq  b32   ;      := CHECK ASSIGN [bits 20 to
    qq (b12)  t 1         ;      22 (AUX TABLE [top operand
    hv  c                 ;      + 1])]; Check stack;
                          ;    goto NEXT;
b33:qq  7.33+7.39         ;    mask
[1b33]qq−9.4+[1+2+4]7.9+c1.19+42.27, ; Operator: prep assign
c56:arn(b12)  , mb  b33             ; PREP ASSIGN:
    ar  1b33          IRC ;
    gr (b12)          MRC ;
    hv  c                 ;
```

[Code parameters. Following code we may meet:
1) Any identifier description, including specifications and dope description
2) Std identifiers
3) CAR RET = 196
4) begin code = 210]

```
c57:pmn(e1)   X   1       ; MORE: R:= input;
    hs  e2            LA  ;
    hv  a17           LT  ;    if R > 511 then goto STD;
    ca  196   , hh  a18   ;    if R = CR then goto CR;
    nc  210   , hv  a19   ;    if R ǂ begin code then goto RAND;
    pa  b20               ;    no code:= false
c58:pmn 104   DX          ; BEGCODE:
    hs  e3    , ps  i     ;    output(begin code);
    pmn(e1)   X   1       ;    for R:= input while R < 512
    hs  e2            LA  ;      do output(R);
    hv  e3            NT  ;
    tk −30    , sc  e4    ;    CRcount:= CRcount − R;
    tk  30    , ps  c−1   ;    output (R); goto NEXT;
a18h:hv e3    , arn b25   ; CR: CRcount:= CRcount+1;
    pa  b35   t 96        ;    output(CAR RET out);
    ac  e4    , hv  a29   ;    goto MORE;
```

```
a17:ga  r1                ; STD: R:= STD DESCR [R];
    arn[stdid], ck  10    ;    trno:= bits 10 to 19 (R);
    ga  b34   , ck  10    ;
    tk  −2    , ga  b35   ;    trrel:= bits 20 t0 27(R);
    ck  6                 ;
    mb  15    D           ;    output (bits 30 t0 33(R) + 468);
    ar  468   D           ;
    hs  e3                ;
b34:arn[tr.no]D           ;    output (trno);
    hs  e3                ;
a29:                      ;
b35:arn[tr.rel]D          ;    output (trrel);
    ps  c57−1 , hv  e3    ;    goto MORE;
a19:ga  b36   , is (b36)  ; RAND: b36:= R;
    bs  s100  t   567     ;  if R ≥ 412 then
    hv  c57               ;  begin  output (b36);
b36:arn [ ]   D           ;    outin;
    hs  e3                ;    outin;
    hs  c25               ;  end;
    ps  c57−1 , hv  c25   ;  goto MORE;

c59:pm (b22)  D −6        ; THEN: M:= rel:= rel−;
    arn b21   , ca  e3    ;    if no relation then
    pm (b2)   D           ;        M:= par2;
    ps  c−1   X           ;    b26:= M; NORELOUT;
    ga  b26   , hv  a27   ;    goto NEXT;

c64:grn(e20)  , hs  c25   ; END PASS 6: n:= input; output(n);
    arn(e1)   , ga b40    ;    for n:= n−1 while n≥0 do
b40:bt  −1    t  −1       ;        output (input)
    ps  b40−1 , hv  c25   ;
    grn d5                ;

a22:arn d5    t   1       ;    Count depths of 2
    qq (2e4)  t   1       ;    stacks
    hv  a22           NZ  ;
a23:arn(e20)  t  −1       ;
    qq (3e4)  t  1        ;
    hv  a23           NZ  ;

b20:it  1.1 [no code], qq (16e4);  mode bits:= mode bits ∨ no code;
    pin(16e4) , bs (b20)  ;        if no code ∧ _error occurred then
    arn e20−40 D      NTB ;        skip pass 9: R:= top core−40 else R:= 0;
    ps  e42   , hh  e29   ;        set return (get GP seg); goto next segm;

c66:arn 107   DV          ; STD 2 CALL WITHOUT PAR: output(std 2 call); skip
c9: arn 103   D           ;    output (end case st);
    ps  c−1   , hv  e3    ;    goto NEXT;
```

[7.7.67]
[GIER ALGOL 4, pass 6, page 10]

[Input control table] [add explanation of table format]

```
i=i−103      ;
d1:          ;
i=103d1      ;
qq  7.3+4.9+c24.19+71.27+      0.39,  ; 412 general
qq  7.3+4.9+c24.19+71.27+      0.39,  ; 416 undeclared
qq  7.3+4.9+c24.19+69.27+      8.39,  ; 420 label
qq  7.3+4.9+c24.19+73.27+     32.39,  ; 424 switch
qq  7.3+4.9+c24.19+72.27+      8.39,  ; 428 formal label
qq  7.3+4.9+c24.19+73.27+     32.39,  ; 432 format switch
qqf 7.3+4.9+c24.19+73.27+     16.39,  ; 436 no par proc, in, re, bo, no
qqf 7.3+4.9+c10.19+73.27+     20.39,  ; 440 par proc, in, re, bo, no
qqf 7.3+4.9+c24.19+71.27+     24.39,  ; 444 simple, in, re, bo
qqf 7.3+4.9+c24.19+70.27+     10.39,  ; 448 array, in, re, bo
qq  7.3+0.9+c30.19                    ; 452 dope descr
qqf 7.3+4.9+c24.19+73.27+     28.39,  ; 456 formal proc, in, re, bo, no
qqf 7.3+4.9+c24.19+72.27+     24.39,  ; 460 formal simple, in, re, bo, str
qqf 7.3+4.9+c24.19+72.27+     13.39,  ; 464 anon array, in re bo
qqf 7.3+4.9+c31.19+75.27+      4.39,  ; 468 literla, in, re, bo, str
qqf 7.3+0.9+c27.19+           20.39   ; 472 spec: simple in, re, bo, str
qqf 7.3+0.9+c27.19+           24.39   ; 476 − label, val: in, re, bo
qqf 7.3+0.9+c27.19+           28.39   ; 480 − array, in, re, bo, proc
qqf 7.3+0.9+c27.19+           32.39   ; 484 − proc, in, re, bo, switch
qqf 7.3+0.9+c27.19+           36.39   ; 488 − unsp, gen
qqf 4.4+9.9+  c.19+      2.33+38.39   ; 492 < ≤ = ≥
qqf 4.4+9.9+  c.19+      2.33+42.39   ; 496 > ǂ

qqf   7.3+          8.9+  c.19+             44.39  ; 500 not, entier
qqf   7.3+          8.9+  c.19+             46.39  ; 504 pos, neg, abs, round
qqf   7.3+          8.9+  c.19+             50.39  ; 508opint,opreal,opbool,opstr
qq   14.4+     [1,2]3.9+  c.19+ 36.27+ 8.33      , ; 128 proc;
qq   14.4+          8.9+  c.19+ 10.27+       9.39, ; 129 ifex
qq   14.4+          0.9+  c.19+ 10.27            , ; 130 ifst
qq   −8.4+     [1,2]3.9+c59.19+ 41.27+ 5.33       ; 131 thenex
qq  −10.4+     [1,8]9.9+c22.19+ 37.27+ 7.33+ 0.39  ; 132 elseex
qq   14.4+          2.9+  c.19+ 36.27            , ; 133 end else ex
qq  −10.4+          1.9+c48.19+ 39.27+ 7.33       ; 134 end else ex
qq   14.4+          0.9+  c.19+ 12.27            , ; 135 end else st
qq   14.4+          0.9+  c.19+ 12.27            , ; 136 end then st
qq  −10.4+     [1,2]3.9+  c.19+ 44.27+11.33      , ; 137 end go to
qq   14.4+          0.9+  c.19+ 13.27            , ; 138 for
qq  −10.4+          1.9+c41.19+ 20.27+ 2.33      , ; 139 step
qq  −10.4+          1.9+c41.19+ 21.27+ 2.33      , ; 140 until
qq   14.4+          0.9+  c.19+ 24.27            , ; 141 end do
qq   14.4+          0.9+  c.19+ 77.27            , ; 142 end single do
```

```
qq    8.4+         11.9+  c.19+       41.33+16.39  ; 143 mod
qq    6.4+     [1,8]9.9+  c.19+        2.33+54.39  ; 144 +
qq    6.4+     [1,8]9.9+  c.19+        2.33+55.39  ; 145 −
qq    8.4+     [1,8]9.9+  c.19+        2.33+56.39  ; 146 ×
qq    8.4+     [1,8]9.9+c35.19+        0.33+57.39  ; 147 /
qq    8.4+  [1,2,8]11.9+  c.19+       41.33+59.39  ; 148 ÷
qq   10.4+  [1,2,8]11.9+  c.19+        0.33+58.39  ; 149 ↑
qq   −8.4+  [1,2,8]11.9+  c.19+        5.33+63.39  ; 150 shift
qq  −10.4+         0.9+c41.19+ 80.27            ,  ; 151 first bound
qq  −10.4+         0.9+c41.19+ 81.27            ,  ; 152 not first bound
qq   −8.4+  [1,2,8]11.9+  c.19+105.27+ 1.33+12.39,  ; 153 of switch
qq   14.4+         0.9+c57.19+117.27+ 0.33      ,  ; 154 code
qq  −10.4+         2.9+c54.19+102.27            ,  ; 155 end switch
qq    0.4+  [1,2,8]11.9+  c.19+        5.33+60.39  ; 156 and ∧
qq   −2.4+  [1,2,8]11.9+  c.19+        5.33+61.39  ; 157 or ∨
qq   −4.4+  [1,2,8]11.9+  c.19+ 34.27+ 5.33+61.39,  ; 158 imply =>
qq   −6.4+  [1,2,8]11.9+  c.19+        5.33+62.39  ; 159 ≡
qq   14.4+         8.9+  c.19+             14.39  ; 160 (
qq  −10.4+         0.9+c44.19                     ; 161 )
qq  −10.4+       [1]1.9+  c.19+ 16.27+ 2.33      ,  ; 162 simple for do
qq   −8.4+     [1,2]3.9+c43.19+ 23.27+ 2.33        ; 163 step element do
qq   −8.4+     [1,2]3.9+c44.19+ 19.27+ 5.33      ,  ; 164 while element do
qq   14.4+         0.9+  c.19+ 97.27            ,  ; 165 case st
qq   14.4+         0.9+  c.19+ 97.27            ,  ; 166 case expr
qq   −8.4+  [1,2,8]11.9+  c.19+ 98.27+41.33+13.39,  ; 167 of expr
qq  −10.4+         1.9+c53.19+101.27+12.33        ; 168 end case expr
qq  −10.4+         8.9+c47.19+              0.39  ; 169 case comma
qq   14.4+         0.9+  c.19+100.27            ,  ; 170 case semicolon
qq   14.4+         0.9+  c.19+115.27            ,  ; 171 end loop
qq   14.4+         2.9+  c.19                     ; 172 do
qq   −8.4+     [1,2]3.9+c59.19+ 41.27+ 5.33        ; 173 then st
qq   14.4+         0.9+  c.19+ 11.27            ,  ; 174 else st
qq   −8.4+     [1,2]3.9+  c.19+ 98.27+41.33      ,  ; 175 of st
qq   14.4+         0.9+ c9.19+100.27            ,  ; 176 end case st
qq  −10.4+         0.9+c13.19                     ; 177 end call
qq  −12.4+         0.9+  c.19+ 76.27            ,  ; 178 ] one
qq  −12.4+         0.9+  c.19+ 76.27            ,  ; 179 ] more
qq  −10.4+         0.9+  c.19                     ; 180 call param
qq  −10.4+         8.9+  c.19+ 27.27+      18.39,  ; 181 comma 1
qq  −10.4+         8.9+  c.19+ 27.27+      18.39,  ; 182 comma 2
qq  −10.4+         0.9+c41.19                     ; 183 bound colon
qq  −10.4+         1.9+c41.19+ 15.27+ 2.33      ,  ; 184 simple for
qq   −8.4+     [1,8]9.9+c40.19+ 14.27+ 2.33+15.39,  ; 185 := for
qq   −8.4+     [1,2]3.9+c42.19+ 22.27+ 2.33        ; 186 stepelem
qq   −8.4+     [1,2]3.9+  c.19+ 18.27+ 5.33      ,  ; 187 while elem
```

```
qq  −10.4+          1.9+c41.19+ 17.27+ 2.33      , ; 188 while
qq  −10.4+          2.9+  c.19                     ; 189 endass
qq  −10.4+    [2,8]10.9+c55.19+116.27+      11.39, ; 190 :=
qq   14.4+  [1,2,8]11.9+c55.19+116.27+ 4.33+11.39, ; 191 first:=
qq   14.4+          0.9+  c.19+  9.27            , ; 192 while label
qq   14.4+              c56.19                     ; 193 prepass
qq   14.4+          0.9+  c.19+  7.27            , ; 194 goto bypass
qq   14.4+          0.9+  c.19+ 86.27            , ; 195 bypass label
qq   14.4+          0.9+ c5.19+ 96.27            , ; 196 CARRET
qq   14.4+          0.9+ c7.19+  5.27            , ; 197 begin block
qq   14.4+          0.9+c24.19+  2.27            , ; 198 begin proc
qq   14.4+          2.9+c24.19+ 85.27            , ; 199 take array
qq   14.4+          2.9+  c.19+ 84.27            , ; 200 take value int
qq   14.4+     [1,2]3.9+  c.19+ 83.27+ 2.33      , ; 201  −     −   real
qq   14.4+          2.9+  c.19+ 84.27            , ; 202  −     −   bool
qq   14.4+          0.9+c44.19+ 82.27            , ; 203 end bounds
qq   14.4+          0.9+  c.19+  6.27            , ; 204 end block
qq   14.4+          0.9+ c7.19+  3.27            , ; 205 end proc
qq   14.4+          0.9+ c7.19+  4.27            , ; 206 end tp pr
qq   14.4+          0.9+  c.19+  8.27            , ; 207 label colon
qq   14.4+          1.9+c11.19+d15.27+13.33+10.39  ; 208 beg call
qq   14.4+     [1,8]9.9+c39.19+        3.33+19.39  ; 209 [
qq   14.4+          0.9+c58.19+117.27+ 0.33      , ; 210 beg code
qq   14.4         + 0.9+  c.19                     ; 211 end core code
qq   14.4+          0.9+  c.19+118.27            , ; 212 core
qq   14.4+          0.9+c64.19+87.27             , ; 213 end pass
qq   14.4+          1.9+c11.19+d15.27+14.33+10.39  ; 214 beg func
qq   14.4+          8.9+ c6.19+ 79.27+      15.39, ; 215 begin bounds
```

[Operator table]

```
i=i-8     ;
d3:       ;
i=8d3     ;
qq   -9.4+          3.9+ c8.19+ 78.27+ 1.33      , ;  8 switch design
qq   -9.4+               c3.19                     ;  9 if ex
qq   -9.4+     [1,2]3.9+c14.19+ 78.27+12.33      , ; 10 accept 208, 214
qq   -9.4+          1.9+ c1.19+ 43.27+ 7.33      , ; 11 := 191,190
qq   -9.4+          1.9+c52.19+ 99.27+11.33        ; 12 of switch
qq   -9.4+          1.9+c52.19+ 99.27+ 7.33        ; 13 of case
qq  -13.4+          0.9+c64.19+222.27            , ; 14 (
qq   -9.4+     [1,2]3.9+ c3.19+        1.33        ; 15 for integer
qq    9.4+   [1,2,4]7.9+ c1.19+ 51.27+41.33+ 4.39, ; 16 mod
qq   -9.4+     [1,2]3.9+ c1.19+ 25.27+ 1.33      , ; 17 first subscr. 209
qq   -9.4+     [1,2]3.9+ c1.19+ 26.27+ 1.33      , ; 18 not first
qq  -11.4+          0.9+  c.19+ 28.27            , ; 19 last subscr  452
qq   -9.4+     [1,2]3.9+c12.19+ 78.27+41.33      , ; 20 spec int      name
qq   -9.4+          3.9+c12.19+ 78.27+40.33      , ; 21  -   re        -
qq   -9.4+          3.9+c12.19+ 78.27+ 5.33      , ; 22  -   bool      -
qq   -9.4+          3.9+c12.19+ 78.27+ 9.33      , ; 23  -   string
qq   -9.4+          3.9+c12.19+ 78.27+11.33      , ; 24  -   label
qq   -9.4+          3.9+c12.19+ 78.27+ 2.33      , ; 25  -   value int
qq   -9.4+          3.9+c12.19+ 78.27+ 2.33      , ; 26  -     -   re
qq   -9.4+          3.9+c12.19+ 78.27+ 5.33      , ; 27 spec value bo
qq   -9.4+          3.9+c12.19+ 78.27+15.33      , ; 28  -   array int
qq   -9.4+          3.9+c12.19+ 78.27+16.33      , ; 29  -     -   re
qq   -9.4+          3.9+c12.19+ 78.27+17.33      , ; 30  -     -   bo
qq   -9.4+          3.9+c12.19+ 78.27+18.33      , ; 31  -   proc no type
qq   -9.4+          3.9+c12.19+ 78.27+19.33      , ; 32  -     -   int
qq   -9.4+          3.9+c12.19+ 78.27+20.33      , ; 33  -     -  re
qq   -9.4+          3.9+c12.19+ 78.27+21.33      , ; 34  -     -  bo
qq   -9.4+          3.9+c12.19+ 78.27+ 6.33      , ; 35  -   switch
qq   -9.4+          3.9+c12.19+ 78.27+12.33      , ; 36  -   unsp
qq   -9.4+          3.9+c12.19+ 78.27+12.33      , ; 37  -   general
qq    5.4+     [1,2]3.9+c33.19+ 60.27+ 2.33        ; 38 <
qq    5.4+     [1,2]3.9+c33.19+ 61.27+ 2.33        ; 39 ≤
qq    5.4+     [1,2]3.9+c33.19+ 62.27+ 2.33        ; 40 =
qq    5.4+     [1,2]3.9+c33.19+ 63.27+ 2.33        ; 41 ≥
qq    5.4+     [1,2]3.9+c33.19+ 64.27+ 2.33        ; 42 >
qq    5.4+     [1,2]3.9+c33.19+ 65.27+ 2.33        ; 43 ≠
qq    3.4+   [1,2,4]7.9+ c1.19+ 34.27+ 5.33+ 6.39, ; 44 not
qq   13.4+   [1,2,4]7.9+ c1.19+ 33.27+40.33+ 4.39, ; 45 entier
qq    7.4+   [1,2,4]7.9+c34.19+  1.27+ 2.33+ 5.39  ; 46 pos
qq    7.4+   [1,2,4]7.9+c34.19+ 35.27+ 2.33+ 5.39, ; 47 neg
qq   13.4+   [1,2,4]7.9+c34.19+ 32.27+ 2.33+ 5.39, ; 48 abs
qq   13.4+   [1,2,4]7.9+ c1.19+ 29.27+40.33+ 4.39, ; 49 round
qq   13.4+          7.9+ c1.19+ 94.27+10.33+ 4.39, ; 50 opint
qq   13.4+          7.9+ c1.19+ 93.27+10.33+ 5.39, ; 51 opreal
qq   13.4+          7.9+ c1.19+ 94.27+10.33+ 6.39, ; 52 op bool
qq   13.4+          7.9+ c1.19+ 94.27+10.33+ 7.39, ; 53 opstring
qq    7.4+     [1,2]3.9+c36.19+ 45.27+ 2.33        ; 54 +
qq    7.4+     [1,2]3.9+c36.19+ 46.27+ 2.33        ; 55 -
qq    9.4+     [1,2]3.9+c37.19+ 48.27+ 2.33        ; 56 ×
qq    9.4+     [1,2]3.9+c36.19+ 49.27+ 2.33        ; 57 /
qq   11.4+       [1]1.9+c38.19+ 53.27+ 2.33        ; 58 ↑
qq    9.4+   [1,2,4]7.9+ c1.19+ 50.27+41.33+ 4.39, ; 59 :
qq    1.4+   [1,2,4]7.9+ c1.19+ 66.27+ 5.33+ 6.39, ; 60 ∧
qq   -1.4+   [1,2,4]7.9+ c1.19+ 67.27+ 5.33+ 6.39, ; 61 ∨
qq   -5.4+   [1,2,4]7.9+ c1.19+ 68.27+ 5.33+ 6.39, ; 62 ≡
qq   -7.4+   [1,2,4]7.9+ c1.19+ 91.27+41.33+ 6.39, ; 63 shift
```

```
[Check operators: case]
b31: qq −9.4+    c51.19+99.27        ; 0 undeclared
     qq −9.4+1.9+c49.19+99.27+ 2.33  ; 1 integer
     qq −9.4+1.9+c50.19+99.27+ 0.33  ; 2 real
     qq −9.4+1.9+c52.19+99.27+ 5.33  ; 3 boolean
     qq −9.4+1.9+c52.19+99.27+ 9.33  ; 4 string
     qq −9.4+1.9+c52.19+99.27+11.33  ; 5 label

[Check operators: else]
b27: qq −9.4+    c63.19              ; 0 undeclared
     qq −9.4+1.9+c45.19+ 1.27+ 2.33  ; 1 integer
     qq −9.4+1.9+c62.19+       0.33  ; 2 real
     qq −9.4+1.9+ c3.19+       5.33  ; 3 boolean
     qq −9.4+1.9+ c3.19+       9.33  ; 4 string
     qq −9.4+1.9+ c3.19+      11.33  ; 5 label

[Check assign operators]
b32: qq −9.4+1.9+c3.19+10.33+0.39 ; 0 undeclared
     qq −9.4+1.9+c3.19+41.33+4.39 ; 1 integer
     qq −9.4+1.9+c3.19+40.33+5.39 ; 2 real
     qq −9.4+1.9+c3.19+ 5.33+6.39 ; 3 boolean
```

```
[Std. proc table, at high address end of store, The entry at L contains:

L:    qq ref  .9+track no.19+track rel.27+code output.33+operand.39
L+1:  qq specn.9+spec(n−1).14+ ...

Operand is the entry in the OPERAND TABLE of the procedure identifier,
spec is the entry in the OPERATOR TABLE of the specifier operator.
Code output = output value − 468.
Ref will point to one of the action words of a table:]


d10: [Ref table]                      ; Procedure kind
qq  7.3+    c16.19+74.27+5.33      , ; 0  fast with parameters
qq  7.3+    c16.19+74.27+2.33      , ; 1  fast without parameters
qq  7.3+    c16.19+74.27           , ; 2  slow, fixed parameter list
qq  7.3+    c16.19+74.27+4.33      , ; 3  slow, variable parameter list
qq  7.3+    c16.19+71.27+1.33      , ; 4  std. variable
qq  7.3+    c19.19+      3.33        ; 5  in program with parameter
qq  7.3+4.9+  c.19+90.27+     6.39 , ; 6  kb on
qq  7.3+4.9+  c.19+89.27+     4.39 , ; 7  lyn
qq  7.3+4.9+  c.19+106.27+   19.39 , ; 8 writecr


d17: [Std proc parameter specifiers]   ; spec. Parameter  kind  type
qq −9.4+3.9+c12.19+ 78.27            , ; 0  used internally, slow
qq          c14.19−c12.19          f, ; 1   −      −      ,  −    var. list
qq −9.4+1.9+c26.19+109.27+41.33        ; 2  integer expression
qq −9.4+1.9+c26.19+109.27              ; 3  real          −
qq −9.4+1.9+c26.19+109.27+ 2.33        ; 4  int. or real  −
qq −9.4+1.9+c26.19+109.27+ 5.33        ; 5  boolean       −
qq −9.4+1.9+c26.19+111.27+41.33        ; 6  short         −
qq −9.4+1.9+c23.19+110.27+23.33+16.39  ; 7  integer variable
qq −9.4+1.9+c23.19+110.27+26.33+16.39  ; 8  real        −
qq −9.4+1.9+c23.19+110.27+22.33+16.39  ; 9  boolean     −
qq −9.4+1.9+c23.19+109.27+24.33+ 8.39  ; 10 array identifier, any type
qq −9.4+1.9+c26.19+109.27+ 9.33        ; 11 string
qq −9.4+1.9+c26.19+       25.33        ; 12 variable or array, any type
qq −9.4+1.9+c26.19+       19.33        ; 13 integer proc. with parameters
qq −9.4+1.9+c26.19+       11.33        ; 14 label
qq −9.4+1.9+c26.19+       12.33        ; 15 anything
qq −9.4+1.9+c26.19+ 95.27+22.33      , ; 16 gier
qq −9.4+1.9+c26.19+ 92.27+41.33      , ; 17 select
qq −9.4+1.9+c26.19+ 88.27+41.33      , ; 18 writechar
qq −9.4+1.9+c26.19+ 32.27            , ; 19 abs
qq −9.4+1.9+c26.19+ 33.27            , ; 20 entier
```

[Primary operand table 1]

```
d2:
d20:qq 31.4+31.9+31.14+31.19+31.24                          ; 412, 416
[11111 11111 1 11111 11111 2 11111 11111 3 11111 11111    0 undeclared]
    qq 13.4+ 4.9+20.14+10.19+2.24+16.29                     ;
[01101 00100 1 10100 01010 2 00010 10000 3 00000 00000    1 subs. var. int]
    qq 21.4+ 4.9+20.14           +24.29                     ;
[10101 00100 1 10100 00000 2 00000 11000 3 00000 00000    2 subs. var. real]
    qq  1.4+20.9+20.14+4.24      +16.29                     ;
[00001 10100 1 10100 00000 2 00100 00000 3 00000 00000    3 subs. var. bool]
d12:qq 12.4+ 6.9+20.14                                      ; 16.45,49,50,59,468
[01100 00110 1 10100 00000 2 00000 00000 3 00000 00000    4 result integer]
d13:qq 20.4+ 6.9+20.14                                      ; 46,−48, 51
[10100 00110 1 10100 00000 2 00000 00000 3 00000 00000    5 result real]
d14:qq      22.9+20.14                                      ; 44, 52, 60−63
[00000 10100 1 10100 00000 2 00000 00000 3 00000 00000    6 result bool]
    qq       5.9+20.14                                      ; 53
[00000 00101 1 10100 00000 2 00000 00000 3 00000 00000    7 string]
    qq       4.9+12.14                                      ; 420,428
[00000 00100 1 01100 00000 2 00000 00000 3 00000 00000    8 label]
    qq           4.14                                       ;
[00000 00000 1 00100 00000 2 00000 00000 3 00000 00000    9 no type]
    qq  2.4+     4.14+16.19+1.24+16.29                      ; 448
[00010 00000 1 00100 10000 2 00001 10000 3 00000 00000   10 array, sub. int]
    qq  2.4+     4.14+ 8.19+1.24+16.29                      ;
[00010 00000 1 00100 01000 2 00001 10000 3 00000 00000   11 array. sub. real]
    qq  2.4+     4.14+ 4.19+1.24+16.29                      ;
[00010 00000 1 00100 00100 2 00001 10000 3 00000 00000   12 array, sub. bool]
    qq           4.14+16.19+1.24+16.29                      ; 464
[00000 00000 1 00100 10000 2 00001 10000 3 00000 00000   13 array, anon. int]
    qq           4.14+ 8.19+1.24+16.29                      ;
[00000 00000 1 00100 01000 2 00001 10000 3 00000 00000   14 array, anon. real]
    qq           4.14+ 4.19 +1.24+16.29                     ;
[00000 00000 1 00100 00100 2 00001 10000 3 00000 00000   15 array, anon. bool]
    qq 12.4+ 6.9+20.14+ 1.19                                ; 436
[01100 00110 1 10100 00001 2 00000 00000 3 00000 00000   16 proc. no. par, int]
    qq 20.4+ 6.9+20.14+      16.24                          ;
[10100 00110 1 10100 00000 2 10000 00000 3 00000 00000   17 proc. no. par, real]
    qq      22.9+20.14+       8.24                          ;
[00000 10110 1 10100 00000 2 01000 00000 3 00000 00000   18 proc. no. par, bool]
    qq       2.9+ 4.14+ 2.19                                ;
[00000 00010 1 00100 00010 2 00000 00000 3 00000 00000   19 proc. no. par, no typ]
    qq           7.14+ 1.19                                 ; 440
[00000 00000 1 00111 00001 2 00000 00000 3 00000 00000   20 proc. w. par. int]
    qq           7.14+      16.24                          ;
[00000 00000 1 00111 00000 2 10000 00000 3 00000 00000   21 proc. w. par. real]
    qq           7.14+       8.24                          ;
[00000 00000 1 00111 00000 2 01000 00000 3 00000 00000   22 proc. w. par. bool]
    qq           6.14+ 2.19                                 ;
[00000 00000 1 00110 00010 2 00000 00000 3 00000 00000   23 proc. w. par. no. typ]
```

[Primary operand table 2]

```
    qq 13.4+ 4.9+20.14      +2.24+16.29                    ; 444, 460
[01101 00100 1 10100 00000 2 00010 10000 3 00000 00000   24 simple integer]
    qq 21.4+ 4.9+20.14           +24.29                    ;
[10101 00100 1 10100 00000 2 00000 11000 3 00000 00000   25 simple real]
    qq  1.4+20.9+20.14      +4.24+16.29                    ;
[00001 10100 1 10100 00000 2 00100 10000 3 00000 00000   26 simple bool]
    qq       5.9+20.14                                    ;
[00000 00101 1 10100 00000 2 00000 00000 3 00000 00000   27 formal string]
    qq 12.4+ 6.9+ 7.14+ 1.19                              ; 456
[01100 00110 1 00111 00001 2 00000 00000 3 00000 00000   28 form. proc. int.]
    qq 20.4+ 6.9+ 7.14+     16.24                         ;
[10100 00110 1 00111 00000 2 10000 00000 3 00000 00000   29 form. proc. real]
    qq      22.9+ 7.14+      8.24                         ;
[00000 10110 1 00111 00000 2 01000 00000 3 00000 00000   30 form. proc. bool]
    qq       2.9+ 6.14+ 2.19                              ;
[ 00000 00010 1 00110 00010 2 00000 00000 3 00000 0000   31 form. proc. no type]
    qq  2.4+ 8.9+ 4.14                                    ; 424, 432
[00010 01000 1 00100 00000 2 00000 00000 3 00000 00000   32 switch]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   33 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   34 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   35 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   36 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   37 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   38 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   39 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   40 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   41 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   42 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   43 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   44 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   45 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   46 std. proc.]
    qq            7.14                                    ;
[00000 00000 1 00111 00000 2 00000 00000 3 00000 00000   47 std. proc.]
```

```
[6.11.67]
[GIER ALGOL 4, pass 6, page 18]

[AUXILIARY OPERAND TABLE
Bit
1−4     Corresponding operand. 7c39
5−6     end call output. b39
   Value   Output on end call
     0           end call
     1           std 2 call
     2           None
7       Switch identifier 5c39
9−11    Begin call. 1.11 = Set accept operator
                    4.11 = output begin call
15      Label type. c22
16−19   Type, for output. c12, b30
20−22   Check operator. 4b30, 1c55
23−32   Left bracket operator]
d16:             ;
i=d16−d2         ;
d4:              ;
i=d16            ;
d19=−d4          ;

qq     0.4+0.6+5.11+         0.19+0.22+17.32, ; 0 undeclared
qq                           1.19+1.22 ; 1 subs. var. int
qq                           2.19+2.22 ; 2   −    −   real
qq                           3.19+3.22 ; 3   −    −   bool
qq                           1.19+1.22 ; 4 result int.
qq                           2.19+2.22 ; 5   −    real
qq                           3.19+3.22 ; 6   −    bool
qq                           4.19+4.22 ; 7 string
qq                      1.15+5.19+5.22 ; 8 label
qq                           2.19+2.22 ; 9 no type
qq     1.4+                  1.19+     17.32  ;  10 array subs. int.
qq     2.4+                  2.19+     17.32  ; 11    −    −   real
qq     3.4+                  3.19+     17.32  ; 12    −    −   bool
qq                           1.19      ; 13 array anon. int.
qq                           2.19      ; 14    −    −   real
qq                           3.19      ; 15    −    −   bool
qq                           1.19+1.22 ; 16 proc. no. par. int
qq                           2.19+2.22 ; 17   −    −   −   real
qq                           3.19+3.22 ; 18   −    −   −   bool
qq                           0.19      ; 19   −    −   −   no type
qq     4.4+0.6+4.11+        15.19      ; 20 proc. w. par. int.
qq     5.4+0.6+4.11+        14.19      ; 21  −    −   −   real
qq     6.4+0.6+4.11+        13.19      ; 22  −    −   −   bool
qq     9.4+0.6+4.11+         0.19      ; 23  −    −   −   no type
qq                           1.19+1.22 ; 24 simple integer
qq                           2.19+2.22 ; 25   −    real
qq                           3.19+3.22 ; 26   −    bool
qq                           4.19+4.22 ; 27 formal string
qq     4.4+0.6+5.11+         1.19+1.22,; 28 formal proc. int.
qq     5.4+0.6+5.11+         2.19+2.22,; 29    −     −   real
qq     6.4+0.6+5.11+         3.19+3.22,; 30    −     −   bool
qq     9.4+0.6+5.11+         0.19     ,; 31    −     −   no type
qq     8.4+    1.7+          0.19+      8.32  ; 32 switch   −
```

[6.11.67]
[GIER ALGOL 4, pass 6, page 19]

[Auxiliary operand table 2]

[Standard procedures]
```
qq  4.4+1.6+0.11 ; 33 integer fast
qq  5.4+1.6+0.11 ; 34 real     −
qq  6.4+1.6+0.11 ; 35 boolean  −
qq  9.4+1.6+0.11 ; 36 no type  −
qq  4.4+0.6+4.11 ; 37 integer slow, fixed list
qq  5.4+0.6+4.11 ; 38 real      −     −     −
qq  6.4+0.6+4.11 ; 39 boolean   −     −     −
qq  9.4+0.6+4.11 ; 40 no type   −     −     −
qqf 4.4+0.6+4.11 ; 41 integer   −   variable list
qqf 5.4+0.6+4.11 ; 42 real      −      −       −
qqf 6.4+0.6+4.11 ; 43 boolean   −      −       −
qqf 9.4+0.6+4.11 ; 44 no type   −      −       −
qq  4.4+2.6+0.11 ; 45 integer in program
qq  5.4+2.6+0.11 ; 46 real          −
qq  9.4+2.6+0.11 ; 47 no type       −
```

```
    qq 15.4+c64.19+111.27,;   dummy operator
d5: qq−13.4+c64.19+ 99.27,;    operator stack bottom
c61:hs  c25               ; START PASS 6:  output(input);
a25:pp  a24   , pp  p1    ;    p:= a24; rep: p:= p + 1;
    arn e20   , ca  p     ;    if p = p0 then goto NEXT;
    grn a24   , hv  c     ;    stack[p]:= 0; goto rep;
a24:grn p     , hh  a25   ;


d18: qq [pass sum]        ;
d e22=k−e14, e47=j        ; Set load parameters
b k=e23, i=0              ; Load segment word 10
i=10e21                   ;
qq e16.9+1d18.19−e16.19+6.24+c61.39 ;
e                                   ;
e [final end]             ;

s
```

[9.11.1967]
[GIER ALGOL 4, Pass 9, page 1]

```
b k=e22+e14, i=e16−e47, a60, b50, d40 ; drumblock head pass 9
d i=e16                               ;

[Predefinitions, when no comment then input byte value]

d10= 76 ; bl CR
d11= 63 ; CR
d12= 30 ; ,
d13= 77 ; e
d14= 59 ; t word
d15= 60 ; last t word
d16= 75 ; =
d17=104 ; beg code
d18= 96 ; CR outside code
d19= 52 ; dope spec
d20= 68 ; std spec
d21= c  ; display 0, not input value
d22= 69 ; forbid spec
d23=104 ; code, outputvalue
d24= 46 ; boolean simple spec

b c70   ; core block head pass 9. To shield c−names


b k=e31, i=0 ; load texts
i=e32        ;

d33: tnumber;    ;
d34: tsyntax;    ;
d35: taddr;      ;
d36: tcode head; ;
d37: tcode size; ;
d27: tundef;     ;
d25: tsorry;     ;
e32=i            ;

e                ;
```

```
[constants]

d:   qq              1.39   ;
[1]  qq 1.14−1.19+1.29−1.35 ;
[2]  qq 63.10+15.39         ;
[3]  qq 31.4                ;
[4]  qq 63.25               ;
[5]  qq 1.1+3.16+7.24+5.32  ; modifbits
[6]  qq 1.26+1023.39        ;
[7]  qq 3.29                ;
[8]  qq 2.29                ;
[9]  qq              10.39  ;
[10] qq 1.9−1.25            ;
[11] qq 320                 ; 10/16
[12] can s409 , cm (r−410)  ; 0.8
[13] qq 49.18 [a]−400.32    ;
[14] qq  3.25 [a−d]         ;
[15] qq d21 [displ 0]       ;
[16] qq 1.18 [b−a]+400.32   ;
[17] qq d22.10              ;

[full word working locations]

b:   qq ; instr
[1]  qq ; i
[2]  qq ; addr
[3]  qq ; term




c1:  hs  c8            NQA ; cond store: if must store then store;
c2:  arn d     , ac  e4    ; CR next slip: CRcount:= CRcount + 1;

c3:  grn b     , pi  8     ; next slip: instr:= 0; must store:= fmark:=
                           ;   comma mark:= false; comma allowed:= true;
c4:  hs  c7    , qq  0     ;   next; s:= 0;
     ga  b13   , hv (pd2)  ;   name byte:= Raddr, goto first[p];
```

```
c5:  tk 1      , cl  9     ; shift next: R:= R shift 1; RM:= RM shift 9

c6:  arn(e1)   t   1       ; next byte:
     hv  e2           LA   ;   Raddr:= input byte;
     hr  s1                ;   return;


a:   arn d     , ac  e4    ; blCR: CRcount:= CRcount + 1;
c7:  hs  c6                ; next: Raddr:= next byte;
     ca  d10   , hv  a     ;   if Raddr = <blind CR> then goto blCR
     ga  b20   , it  d1    ;   p:= bits(15, 19, table[byte]);
b20: arn −1    , mb  1d    ;
b1:  gt  r     , pp  −1    ;   modif:= bits(30, 35, table[byte])
     ck  −14   , ga  b6    ;
     arn(e1)   , hr  s1    ;   R:= byte; return;



c8:  arn b                 ; store: R:= instr;

c9:  gr (b2)   V       MPC ; store R: store[i] MPC:= R; goto count i

c10: pa  b2    t   d9      ; start slip: it:= base code;

b2:  arn[i1]0  D   1       ; count i: i:= i1 := i + 1;
     ca  e20   , hv  c48   ;   if i = top core then goto stack alarm;
     ck  10    , gr  1b    ;   return;

c11h:hr  s1    , arn(e1)   ; check CR if curr byte ≠ CR then
c12h:nc  d11   , hs  e5    ; err 1: mess(⊰<code syntax>⊱, 0, 1);
     hv  a1    , qq  sd34  ;        goto skip to next CR;
c13: hs  e5                ; err 2: mess(⊰<code addr>⊱, 0, 1);
a1:  ps  c1−1  , qq sd35   ; skip to next CR; set return (cond store);
                           ;
c70: arn(e1)   V           ; search CR: R:= curr byte; skip next;
a2:  hs  c7                ; rep search: next
     ca  d11   , hr  s1    ;   if Raddr = <code CR> then return;
     ca  d13   , hv  c65   ;   if Raddr = <end code> then goto end code
     nc  d14               ;   if Raddr = text 1 ∨ Raddr = text 2 then
     nc  d15   , hv  a2    ;     begin
     hs  c6                ;     next byte;
     hs  c6                ;     next byte;
     hs  c6                ;     next byte;
     hs  c6                ;     next byte;
     hv  a2                ;     end
                           ;   goto rep search
```

```
c14: hsn c7    X           ; first[1]: abcde: next;
     bs  p−15  , hv  a3    ;   if p < 16 then
     bs  p     , hv  c24   ;     begin if p > 0 then goto op 1;
     hs  c45               ;     cont int;
a3:  bs  p496  , hh  c12   ;     end
     ps (pd5)  , hv  c47   ;   if p < 16 then goto err 1;
                           ;   set return (def act[p]); goto look up;


c15: hs  c7                ; first[2]: idef:
     nc  d16   , hv  c24   ;   if next ǂ <=> then goto op 1;
     hsn c34               ;   read defined:= t; new i:= read addr 2;
a4:  ga  r1    , arn b2    ;   comment 5 < s < 512
     ca  −1    , hh  c11   ;   while i1 ǂ new i do store;
     hs  c8                ;   goto check CR;
     hh  a4                ;
c16=i−1
     arn(b2)   D       NRA ; def act[16]: colon def:
     nc (b2)   , hv  c13   ;   if undefined then Raddr:= i1;
     ps  c3−1  , hv  c44   ;   if Raddr ǂ i1 then goto err 2;
                           ;   set return (next slip); goto define;


c17=i−1
     it (b14)  , pt  b3    ; def act[17]: equal def: save where:= where
     hs  c34               ;   read defined:= t; read addr 2;
b3h: pa  b14   t [save wh] ;   where:= save where;
     hs  c44               ;   set return(test CR); goto define
     hh  c11               ;

c18: pm (b6)   DX          ; first[18,19]: text word: last text word:
     hs  c5                ;   Raddr:= modif; shift next;
     hs  c5                ;   shift next;
     hs  c5           IPC  ;   shift next; PC:= 0;
     hs  c5    , qq  c3−1  ;   shift next; set return(next slip);
     tk  1     , cl  −31   ;   RM := RM shift −30;  goto store R;
     hv  c9                ;

c19: it  1                 ; first[20]: m: float:= f; goto next slip;
c20: pa  b18   , hv  c3    ; first[21]: f: float:= t; goto next slip;
```

```
                              ; mod[7]
c21: hh  c12           LTB    ; first[7]: comma: if −, comman allowed ∨
     bs  s−2   , hh  c12      ;     s > 2 then goto err 1;
     pi  32    t   −33        ;   comma mark:= t;
     hs  c7    , qq  1        ;   next; s:= 1;
     ga  b13   , ca  d11      ;   name byte: Raddr;
                              ;   if Raddr = <CR> then begin
c22: ps  c2−1  , hv  c8       ; mod[6]: CR: set return(CR next slip);
                              ;   goto store end;
c23: hs  c7                   ; first[8, 9, 10, 11, 12, 14]: op part: next;

c24: arn(b20)  , mb  2d       ; op 1: no of ops:= bits(36, 39, table[byte]);
     ck  1     , ga  b5       ;   i:= bits(5,/0, table[byte]); comment
     ck  −11   , ga  b4       ;     relative start of sub table for second
     arn b13   , pm  3d       ;     letter = byte;
     ck  5     , pp  d6       ;   for no of ops:= no of ops while no of ops > 0 then
a6:                           ;     begin i:= i + 1;
b4:  bt[no of ops] t −1       ;     if bits(0, 4, table[byte]) = name byte
b5:  cm  p[i]  V   1          ;       then goto op found;
     hh  c12                  ;     end;
     hv  a6                   ;   goto err 1;

     arn(b5)   , mb  4d       ; op found: R:= bits(20, 25, table[byte]) pos 25;
c25: ck  −10           LPA    ; accinstr 1: if comma mark then
                              ;   R:= R shift −10;
c26: ab  b     , gr  b        ; accinstr 2:
     hsn c7    X              ;   instr:= instr ∨ R; next;
     hv (pd3)                 ;   goto mod [p];


c27: hh  c12           LPA    ; mod[8]: XVD IMNL KOTZPQR ABC:
     pi  64    V   −65        ;   if comma mark then goto err 1;
                              ;   comma allowed:= false goto mod[10];
c28: pin 16    V   −17        ; mod[9]: Ff: fmark:= true; R:= 0;
                              ;   goto acc instr 1;
c29: pm  5d    , tln(b6)      ; mod[10]: Sn): R:= modifications bits shift modif;
     mb  6d    , hv  c25      ;   R:= R ∧ 27 1 13 1023; goto acc instr 1;


c30: gm (b9)   D              ; mod[11]: t: srp bits:= indir:= 0;
     bs  s510  , ps  s2       ;   if s < 2 then s:= s + 1;
     hv  c34                  ;   goto read addr 2

c31:
b6:  it[modif] , pa b9        ; mod[12]: srp: srp bits:= modif;
     bs  s510  , hv  c34      ;   if s ≤ 1 then goto read addr 2

c32: bs  s−1   , hh  c12      ; mod[13]: indirect: if s > 1 then goto err 1
     pt  b10   t   4          ;   indir:= 4;
     hs  c7                   ;   next
     can p−12  , hv  c31      ;   if p = 12 then goto mod 12;
```

```
                              ; mod[0, 1, 2, 4, 5]
c33: qq (e1)   t   −1         ; read addr 1: reset;

c34: hsn c7    X              ; read addr 2: next; sign:= 1; R:= addr:= 0;
     pa  b8    t   1          ;   require defined:= s > 2; comment ≥ 0 = true;
     grn 2b    , it  s−3      ;   empty addr:= true; comment ≠ 0 = true;
a7:                           ;   goto start term;
a8h: pa  b7    , gr  3b       ; read term: require defined:= t; empty addr:= f;
     bs  p506  , hv (pd4)     ; start term: term:= R;
a9:  bs  s509  , hv  c43      ;   if p < 6 then goto addr act[p];
     bsn s507  , hv  c13      ; finis addr: is s ≤ 2 then goto end addr;
b7:  ncn −1    , hv  c13      ;   if s < 5 ∨ empty addr then goto err 2;
     [empty addr require defined]
     arn 2b    , hr  s1       ;   R:= addr; return;


c35: hsn c45   X              ; addract[0]: digits: cont int;
     bs  p509  , hv  a7       ;   if p < 3 then goto read term;
     bs  p508  , hh  a10      ;   if p = 3 then goto point term;
     tk  30    , mt  b8       ;   addr:= addr + (R shift 30) × sign;
     ac  2b    , pa  b7       ;   require defined:= t; empty addr:= f;
                              ;   goto finis addr;
a10h:hv  a9    , gr  3b       ; point term: term:= R;
     bs  s−3   , hv  c13      ;   if read defined then goto err 2;
     hs  c7                   ;   next;
     bs  p     , hv  c13      ;   if p > 0 then goto err 2;
     hsn c45   X              ;   cont int;
     ck  20    , gt  r1       ;   R:= term shift 39 − integer;
[1]  arn 3b    , ns  0        ;   goto add to addr;
     cls 39    , hh  a11      ;


c36: arn(e1)   , ga  b13      ; addract[1]: abcde: name byte:= byte;
     hsn c46   X              ;   read int;
     hs  c47                  ;   look up;
     hh  c42           NRA    ;   if NRA then goto undef exit;
     tl  −30   , is (b9)      ;   R:= defined addr pos 39;
     ca  s−2   , sr  1b       ;   if srp bits = 2 then R:= R − i;

c37: ar  3b    , it  510      ; add term: R:= R + term;
     bs  p508  , hh  a10      ;   if p = 3 then goto point term;
                              ;   R:= R shift 30;
a11h:tk  30    , mt  b8       ; add to addr: R:= R × sign;

c38: bs  p508  , hv  c13      ; addract[3]: point: if p < 4 then goto err 2;
     ac  2b    , hvn a7       ;   addr:= addr + R; R:= 0; goto read term;


c39: hsn c7    X              ; addract[2]: i: next;
     arn 1b    , hv  c37      ;   R:= i; goto add term;
```

```
c40: pa  b7    , it  −1    ; addract[4]: minus: require defined:= true;
c41:                       ;   sign:= −1; goto sign act;
b8:  pa  0[sign]D   1      ; addract[5]: plus: sign:= 1;
     hsn c7    X           ; sign act: next;
     bs  p509  , hv (pd4)  ;   goto if p < 3 then addract[p] else err 2;

c42h:hv  c13   , ga  2b    ; undef exit: part 1 addr:= Raddr;
     bsn p−5   , arn b7    ;   if p < 5 ∨ defined required then
     hv  c13           NT  ;   goto err 2;
     arn b2    , bs  s     ;   value part [where]:= i1 +
     ar  512   D           ;     if s > 0 then 512 else 0;
     ga (b14)          IQA ;   must store:= true;

c43: b9:                   ;
b10h:pmn[srpbits]DX[indir] ; end addr:
     ck  20    , ab  2b    ;   R:= addr ∨ (srp bits + indir);
     pt  b     , ud  c30   ;   part 2 instr:= 0; rps bits:= indir:= 0;
     bs  s     , cl  −10   ;   if s > 0 then RM:= RM shift −10;
     ps  s2    , ud  c33   ;   s:= s + 2; reset;
     hv  c26               ;   goto acc instr 2;
```

```
c44: pm (b14)          IRA ; define: comment list entry [where] to
     ga (b14)          MA  ;     value in Raddr;
     ga  b12   X           ;   chain 1:= set RA (value part[where]);
     ga  b21   , pm  7d    ;   value part [where] MA:= defval:= Raddr;
     hr  s1            LRA ;   if RA then return; comment defined;
                           ;   while chain 1 ≠ 0 do
a12: can(b21)  , hr  s1    ;     begin chain:= chain 1
b21: it [chain1], pa b11   ;     p:= if chain < 512 then 10 else 0;
     pp  0     , bs (b11)  ;     if chain < 512 then chain:= chain + 512;
     pp  10    , it  512   ;     R:= store[chain] shift p;
b11: arn[chain], ck  p     ;     chain 1:= part 1 (R);
     ga  b21   , tk  10    ;     clear R part 1;
     ck  −10   , bs  p     ;     if bits(28, 29, R) = 2 ∧
     hv  a13           NA  ;       (p = 0 ∨ half word instruction) then
     cm  8d    , hv  a13   ;       R:= R − chain pos 9;
     sr (b11)  D           ;
a13:                       ;     R:= R + def val pos 9;
b12: ar[defval]D           ;     store [chain]:= R shift −p;
     ns  p     , ck  s     ;     goto loop def;
     gr (b11)  , hv  a12   ;     end;
                           ;   return;

c45: arn(e1)           IZA ; cont int: digits:= true;
     tk  4     , ck  6     ;   RM:= M × 10 + bits(4, 9, curr byte);
     ml  9d                ;   if R ≠ 0 then
     qq (b16)  t   1   NZ  ;     begin RM:= RM : 10; exp10:= exp10 + 1 end
     dl  9d    X       NZ  ;   if after point then exp10:= exp10 − 1;
     qq (b16)  t   −1  LZB ;     comment only used during number reading;
c46: hs  c7                ; read int: next;
     bs  p511  , hv  c45   ;   if p = 0 then goto cont int;
     hrn s1    X           ;
c47:
b13: pm[name byte] D       ; look up: R:= namebyte pos 25 +
     tl  33    , ck  −10   ;     bits(33, 39, R) pos 16;
a14: pm  10d   , is (b22)  ; repeat look: where:= top list + 1;
     it  s1    , pa  b14   ;   list [stop list]:= R;
     gr  d30   , it  −1    ; loop look: where:= where − 1;
b14: cm[where] , hh  r−1   ;   if ident part [where] ≠ R then goto loop look;
     pmn(b14)  X       IRC ;   M:= 0; R:= set RC(list [where]);
     bs (b22)  V   d31 LB  ;   if −, RB then
     ck  0     , hr  s1    ;     begin remove bit 0; return end;
c48: ps  d32   , hv  e5    ;   top list:= top list + 1
b22: gr[toplist] t 1   M   ;   list [top list] M:= R;
d32=i−1                    ;   free := free − 1; if free  < 0 then goto stack;
     hv  a14   , qq  ne34  ;   goto repeat look;
```

```
                          ; Number reading:
c50: qq (e1)   V   −1     ;
c51:                      ; first[0, 3]: digit: point: reset; goto plus;
b15: qq [neg]  , it  −1   ; first[4]: minus: neg:= t; goto set exp 10;
c52: pa  b15   , pa  b16  ; first[5]: plus: neg:= f; set exp 10; exp 10:= 0;
     hsn c46   X          ;   read int; swap;
     bs  p508  X   510    ;   if p = 3 then
     hsn c46          IZC ;     begin digits:= before point:= f; read int end;
     hv  a20          LZA ;   if −, digits then goto err 3;
     pm  256   DV     LZB ;   if −, before point then swap;
     arn 256   DX         ;   swap; M:= .5;
     mt  b15   , gr  b    ;   instr:= if neg then −R else R;
b16: pp [exp10], bs (b18) ;   p:= exp 10; comment float = ≤ 0
     hv  a18          NZB ;   if before point ∧ −, float then goto integer;
     hv  a19          LZ  ;   if R = 0 then goto terminate number;
a15: can p     , hh  a17  ;   while p ≠ 0 do
     pp  p−1   , bs  p1   ;     if p > 0 then
     mk  11d   , hh  a16  ;       begin M:= normalize(M×10116) to : (n);
     mk  12d   , pp  p+2  ;       p:= p − 1; s:= s + 4 + n end
a16h:ps  s−7   , nk  b17  ;       comment s holds exp 2 and starts at 0;
b17: ps  s[n]  X   4      ;     else begin M:= normalize(M×.8) to: (n);
                          ;       p:= p + 1; s:= s − 3 + n end;
a17h:hv  a15   , mln  b   ;
     nl  b17   , ps (b17) ;   R:= normalize(M×instr) to: (n); s:= s + n;
b18: bs1[float], hv  a21  ;   if −, float then goto store fractional;
     nkf s40   , grf b    ;   instr:= shift float(R, s);
a18: bs  p     , hv  a20  ; integer: if p > 0 then goto err 3;
                          ;
a19: pi (b6)   t   −49    ; terminate number: PC:= modif;
     arn(e1)   , ud (b1)  ;   p:= curr p;
     ca  d11   , hv  c22  ;   if curr byte = <code CR> then goto store it;
     bs  p−1   , hv  a20  ;   if p > 1 then goto err 3;
     bs (b6)   , hs  c7   ;   if modif > 0 then next;
     ca  d11   , hv  c22  ;   if curr byte = <code CR> then goto store it;
a20: hs  e5               ; err 3: mess(⫞<number⫟, 0, 1);
     hv  a1    , qq  sd33 ;   goto skip to CR;


a21: tl  1             ITA ; store fractional: comment the instructions here
     tk  −1    , gr  b    ;   are copied from SLIP, they may make sense
     ann b             NTA ;   to somebody;
     bs  s39   , hv  a20  ;   if s > 40 then goto err 3
     tk  s40              ;
     sr  d             LO  ;
     hv  a20           LO  ;   if overflow then goto err 3;
     gr  b     , hv  a19  ;   instr:= R; goto terminate number;
```

```
c60: pi (16e4) , hs  a27   ; endpass 9: copy 1 more;  in := mode bits;
     arn(e1)   , ga  r1    ;   for i:= curr byte step −1 until 0 do
[1]  bt  −1    t   −1      ;     copy 1 more;
     ps  r−2   , hv  a27   ;
c62: hs  e5            LTB ; finis:  if error occurred then
     hhn e29   , qqn d25   ;   mess(‡<finis‡,2,0);  goto next segm;


     hs  a27               ; copy 4 more: copy 1 more;
     hs  a27               ; copy 3 more: copy 1 more;
a26: hs  a27   , ps  i+2   ; copy 2 more: copy 1 more; set return(skip actions);
a27: hs  c6                ; copy 1 more: Raddr:= next byte; goto output;
     hv  e3                ;   comment hr s1 to a27 will on next hrs 1 goto c61;
c61: hs  c6    , ps  i−1   ; skip actions: next byte; set return(skip actions);
     ga  b23               ;   byte:= if Raddr > 512 then Raddr − 512 else Raddr;
     qq (b23)  t 512   LT  ;   swap;
b23: pm  −1    X   d1      ;   R:= table[byte];
     ps  a26−1         LB  ;   if LB then set return(copy 2 more);
     hv  e3    X       NA  ;   if NA then goto output;
     tk  11    , ck  −16   ;     comment copy 1 or 3 bytes;
     gt  r     , pp  −1    ;   p:= bits(11, 14, R);
     hv  pa28          LB  ;   if LB then goto lsk[p];
     arn d     , hh  pa28  ;   R:= 1; goto rsk[p];


a28: gs  b35   , hv  c61   ; lsk[0]: core: core code:= t; goto skip actions
                           ;
[1]  hv  c63   , ps  c60   ; lsk[1]: goto code;
                           ; rsk[1]: end pass: set return(endpass9); goto copy;
[2]  ps  s1    , hv  r4    ; lsk[2]: copy 4: set return(copy 3 more); goto copy;
                           ; rsk[2]: copy 2: set return(copy 1 more); goto copy;
[3]  ps  s1    V           ; lsk[3]: begblock: set return(copy 1 more);

[4]  ps  s1    , it  1     ; lsk[4]: end proc: setreturn(copy 1 more);
                           ; rsk[4]: end block: t:= 1; goto block count;
b24: qq[curr displ] 1d21V−1; lsk[5]: beg proc: t:= −1;
                           ; block count: curr displ:= curr displ + t; goto copy;
[6]  psn s−2   , ac  e4    ; lsk[6]: copy 5: set return(copy 4 more); R:= 0;
                           ; rsk[6]: CR: CRcount:= CR count + R;
     hvn e3    X           ; copy: swap; goto output;


d40: hv  c62           LZB ; start pass 9:  if no code then goto finis;
     hs  c6    , qq c61 −1 ;   Raddr := next byte;
     ac  b26   , hv  e3    ;   sr0:= displ 0 + Raddr;
                           ;   set return (skip actions); goto output;
```

```
c63: ps (b32)  , gs  b22   ; code: top list:= where spec:= top c;
     gs  b28   , pp  0     ;   p:= 0;
                           ;
a30: hs  c6    , qq  a33   ; loop head: Raddr:= next byte; head act:= 1;
     ca  d18   , hh  a36   ;   if Raddr = <CR> then goto count head CR;
     ca  d17   , hv  a37   ;   if Raddr = <begin code> then goto test for core
     ca  400d19            ;   if Raddr= <dope spec> then
     ps  a32   , pp  p-1   ;    begin headact:=2; p:=p-1 end;
     ca  400d20, ps  a34   ;   if Raddr = <std spec> then headact:= 3;
     ck  17    , pp  p1    ;   Raddr:= Raddr shift 17; p:= p + 1;
                           ;
a31: ar  p     D           ; store head name: R:= R shift -7;
     ar  13d   , ck  -7    ;   R:= R + p pos 16 + name(a) pos 25;
     gr (b22)  t   1   MA  ;   top list:= top list + 1;
     hs  c6    X           ;   list[top list] MA:= R + next byte pos 9;
     ac (b22)  , hvn  s1   ;   goto case headact of (normal op, dope op,
a32=i-1                    ;     std op, test head room;
     arn 14d   , ac (b22)  ; dope op: list [top list]:= list[top list] +
     ps  a35   , hv  c6    ;   namediff(d-a) pos 25; next byte;
a33=i-1, a34=i             ;   goto test head room;
     arn 15d   , ck  7     ; normal op: R:= displ 0 pos 9 shift 7;
     ar  16d   , hs  a31   ; std op: R:= R + namediff(b-a) pos 18;
a35=i-1                    ;   head act:= 4; goto store head name;
     bs (b22)  t   d31     ; test head room: if toplist > max top then
     ps  d32   , hv  e5    ;     mess(⫞<stack⫟, 2, 0);
                           ;   goto loop head;
a36h:hv  a30   , hs  e3    ; count head CR: output Raddr;
     arn d     , ac  e4    ;   CR count:= CR count+1;
     hv  a30               ;   goto loop head;

a37: can(b35)  , hv  c64   ; test for core: if core code then
     ps (b32)  , arn s1    ;     begin
     ga  b43   , ck  -10   ;     if value part [top c + 2] ≠ curr displ ∨
     ca  d24   , arn s2    ;     part 4(list[top c + 1]) ≠ <boolean simple>
     ca (b24)  , hv  c64   ;       then
     arn s1    , ck  -10   ;       set part 4 (list [top c + 1]) to: (<forbid>)
     pm  17d   , tl  10    ;     rel A:= value part [top c + 1];
     gr  s1    , hv  c64   ;     end
                           ;   goto read spec;
```

```
a40:
b25: arn[doperel modif] D  ; dope spec: list [where spec]:= list [where spec]
     ac (b28)  , hv  c64   ;     + dope rel modif; goto read spec;


a41: nc (15d)  , hh  a45   ; abs addr: if Raddr ≠ displ 0 then goto head err;
     gan(b28)          M   ;   value part [where spec] M:= 0; Raddr;= sr0;
b26: arn d21 [sr 0] D      ;   value part [where spec − 1]:=
     is (b28)  , ac  s−1   ;   value part [where spec − 1] + Raddr;

a42: ga  b25   , arn(e1)   ; addr ok: doperel modif := 0; next of spec:
     ca  d11   , hv  a44   ;   if curr byte = code CR then goto test ok;
     nc  d12   , hh  a45   ;   if curr byte ≠ comma then goto head err;
     hsn c46   X           ;   read inf;
     ck  −10               ;   if part 4(R) = kdtype then kd type:= 0;
b27: ca [kdtype], pa b27   ;   goto next of spec;
     hh  a42               ;

a43:                       ; next spec:
b28: pmn[where spec] 0X 1  ;   where spec:= where spec + 1;
     ck  −10   , ga  b27   ;   Raddr:= kdtype:= part 4 (list[where spec];
     ca  d19   , hv  a40   ;   if kdtype = <dope> then goto dope spec;
     ca  d22   , hh  a45   ;   if kdtype = <forbid> then goto head err;
     hsn c46   X           ;   Raddr:= read init;
     pm (b28)  t   1       ;   where spec:= where spec + 1; M:= list[where spec];
     ck  −10   ,  ca  2    ;   if Raddr = <abs addr> then
     hv  a41   X           ;     begin swap; goto abs addr end;
     ca  1     , hvn a42   ;   if Raddr = <s addr> ∨ Raddr = <std proc> ∨
     ca  4     , hvn a42   ;     (Raddr = <p addr> ∧ part 1(M) =
     ca  3     X           ;     curr displ) then
     ca (b24)  , hvn a42   ;       begin R:= 0; goto addr ok end;
a44:
a45h:bs (b27)  , hs  e5    ; test ok: if kdtype > 0 then
                           ; head err: mess(≠<code head ≠, 0, 1)
     hs  c70   , qq  sd36  ;   search CR;
     arn d     , ac  e4    ;   CRcount:= CRcount + 1;
c64: arn b28   , ca (b22)  ; read spec: if where spec ≠ top spec then
     ps  c3−1  , hv  c10   ;     goto next spec;
     hv  a43               ;   set return(next slip); goto start slip;
```

```
c65: it (b32)  , pa  b30   ; end code: collaps name table:
                           ;   n:= top c;
a47: arn b30   , ud  16e4  ;   while n ≠ top list do
     ca (b22)  , hv  a49   ;     begin
b30: arn[n]    t   1       ;     n:= n + 1; select (normal medium)
     hv  a48           LA  ;     R:= list[n];
     ca  0     , hv  a47   ;     if undefined ∧ used then
     hs  e5                ;       begin
     arn(b30)  , qq  sd27  ;       mess(≠<undef.␣≠, 0, 1);
     ck  6     , ud  13e4  ;       select (error medium);
     gt  r     , sy  −1    ;       write char(name char);
                           ;       print(name number);
     ck  4     , tl  −72   ;       select (normal medium);
     ps  a47−1 , hv  e8    ;       end
                           ;     else if defined ∧ name char = c then
a48: ck  17    , ca  102   ;       begin
     ck  −17   , ud  b32   ;       top c:= top c + 1;
     hv  a47               ;       list [top c] MA:= R;
b32: gr[top c] d38 t 1 MA  ;       end
                           ;     end;


a49: hs  c6                ; output CRs:
     mt  a50   , ga  b33   ;   for nCR:= −next byte step −1 until 1 do
     pm  d18   DX          ;     output(CR);
b33: bt  −1    t   −1      ;
a50: ps  r−3   , hv  e3    ;
```

```
c66: ps  d9    , gs  b37   ; output code: code pointer:= base code;
     it (b2)   , pa  b34   ;   Ntail:= size:= i1 − base code − 1;
     nt  s1    , it (b34)  ;
     pt  b42               ;
a51:
b34: pp [size] , it (2e4)  ; next track: p:= size;
     bs  p     , gp  2e4   ;   if p > inf 1 then inf 1:= p;
     qq  d17   , hs  a57   ;   outbyte(code); Raddr:= 0;
b35: can[core code], hva52 ;   if core code then
                           ;     begin
     arn a53   , bs  p−29  ;   Raddr:= 10;
     pp  p−29  , arn a54   ;   if p > 29 then begin p:= p − 29; Raddr:= 6 end;
     bs  p−33              ;   while p > 33 do
     pp  p−33  , hv  r−1   ;     p:= p − 33;
a52: ga  b36   , gp  b41   ;     end;
     bs  p−39  , hs  e5    ;   nhead:= p;
a53: qq  10    , qq  sd37  ;   if p > 39 then mess(⊰<code size⊱, 0, 1);
b36: qq  p−1   , hs  a57   ;   outbyte(p + Raddr);
     nt  p     , qq (b34)  ;   size:= size − p;
     can(b35)  , hv  a55   ;   if core code then
                           ;     begin
     it  p2    , pt  b40   ;     n1head:= p + 2; p:= 6;
a54: pp  6     , hs  a56   ;     code words out (head pointer);
     pa  d29   D   d29     ;     reset (head pointer); p:= nhead;
     pp (b41)  , hs  a56   ;     code words out (i1);
     qq (b2)   , pp  4     ;     p:= 4; if size > 0 then goto next track;
     bs (b34)  , hv  a51   ;     code pointer:= tailstart;
     pa  b37   t   d28     ;     end;
a55: pa  b35   , hs  a56   ;   code words out (code pointer); core code:= f;
b37: qq[codep] , hv  c61   ;   goto skip actions


a56: gs  b39   , pi  0     ; procedure code words out(from where);
     can p     , hr  s1    ;   comment from where in s1, is counted;
     can(b35)  , it  1     ;   while p ≠ 0 do
     pmn(s1)   X   −1  IPC ;     begin
     gi  b38   , pp  p−1   ;     from where:= from where +
     cl  30    , hs  e3    ;       (if core code then −1 else 1);
     cln −10   , hs  e3    ;     R:= core[from where]; p:= p − 1;
     cln −10   , hs  e3    ;     output(part 4 (R)); output (part 3(R));
     cln −10   , hs  e3    ;     output(part 2 (R)); output (part 1(R));
b38: qq  −1    , hs  a57   ;     byteout (bits(41, 42, R) pos 5);
b39: ps  −1    , hv  a56   ;     end;


a57: arn(s)    D           ; procedure byte out(byte value);
     hv  e3                ;   comment byte value in s;
                           ;   output (byte value);
```

[Input table|, op part table]

```
[              copy                  copy
               actions               actions
op 1.op 2.op         pval modif no of
   lttr lttr                   ops in     Input values
        table                  2.lttr
        start                  table        Code   Between code]
d1: [base table] d6=i−1[base op table]  ;          Bytes copied when ǂ 1
qq                                      ;  0         Name when special action
ca 19.4+17.10+       8.19+ 3.35+10.39   ;  1 K
ga 23.4+27.10+ 5.14+ 8.19+21.35+ 7.39 f,;  2 L     3 beg proc
pa  6.4+34.10+ 4.14+ 8.19+ 6.35+ 3.39 f,;  3 M     2 end proc
ab 17.4+37.10+ 4.14+ 8.19+ 7.35+ 2.39 f,;  4 N     2 end type proc
mb  3.4+ 0.10+ 3.14+ 8.19+18.35+ 0.39 f,;  5 O     2 beg block
ac 17.4+39.10+ 4.14+ 8.19+34.35+ 2.39  ,;  6 P       end block
gc 23.4+41.10+       8.19+19.35+ 2.39   ;  7 Q
nc  4.4+43.10+       8.19+26.35+ 5.39   ;  8 R
pc  6.4+48.10+      10.19+14.35+ 7.39   ;  9 S
sc  9.4+55.10+       8.19+ 5.35+ 6.39   ; 10 T
ud 11.4+ 0.10+      14.19+       0.39   ; 11 U
gg 23.4+61.10+       8.19+ 9.35+ 1.39   ; 12 V
hh 24.4+ 0.10+      14.19+       0.39   ; 13 W
gi 23.4+ 0.10+       8.19+10.35+ 0.39   ; 14 X
pi  6.4+62.10+      14.19+       3.39   ; 15 Y
zj 16.4+ 0.10+       8.19+ 4.35+ 0.39   ; 16 Z
ck 19.4+ 1.10+       8.19+ 2.35+ 3.39   ; 17 A
dk 20.4+ 4.10+       8.19+ 1.35+ 2.39   ; 18 B
gk 23.4+ 6.10+       8.19+16.35+ 5.39   ; 19 C
hk 24.4+11.10+       8.19+ 8.35+ 1.39   ; 20 D
lk  2.4+ 0.10+      14.19+       0.39   ; 21 E
mk  3.4+ 0.10+       9.19+ 0.35+ 0.39   ; 22 F
nk  4.4+12.10+      14.19+       1.39   ; 23 G
sk  9.4+13.10+      14.19+       1.39   ; 24 H
tk 10.4+14.10+       8.19+ 0.35+ 2.39   ; 25 I
vk 12.4+16.10+      14.19+       1.39   ; 26 J
cl 19.4+             3.19              ; 27 .
dl 20.4+             4.19              ; 28 −
il 25.4+             5.19              ; 29 +
ml  3.4+             7.19              ; 30 ,
nl  4.4+            13.19              ; 31 (
tl 10.4+            15.19              ; 32 forbid
```

[21.10.1967]
[GIER Algol 4, Pass 9, page 16]

```
[              copy                 copy
               actions              actions
op 1.op 2.op         pval modif no of
   lttr lttr                    ops in    Input values
        table                   2.lttr
        start                   table       Code   Between code]

                                        ;            Bytes copied when ǂ 1
zl 16.4+17.10+      14.19+      10.39   ; 33 k        Name when special action
cm 19.4+27.10+      14.19+       7.39   ; 34 l
gm 23.4+34.10+      14.19+       3.39   ; 35 m
pm  6.4+37.10+      10.19+14.35+ 2.39   ; 36 n
an 17.4+ 0.10+ 2.14+14.19+       0.39  ,; 37 o       2 else Rt expr
sn  9.4+39.10+ 2.14+12.19+ 3.35+ 2.39  ,; 38 p       2 else addr expr
gp 23.4+41.10+ 2.14+14.19+       2.39  ,; 39 q       2 end else Rt expr
pp  6.4+43.10+ 2.14+12.19+ 2.35+ 5.39  ,; 40 r       2 end else addr expr
qq  7.4+48.10+      12.19+ 1.35+ 7.39   ; 41 s
zq 16.4+55.10+      11.19+       6.39   ; 42 t
ar 17.4+ 0.10+      14.19+       0.39   ; 43 u
gr 23.4+61.10+      14.19+       1.39   ; 44 v
hr 24.4+ 0.10+      14.19+       0.39   ; 45 w
sr  9.4+ 0.10+      14.19+       0.39   ; 46 x
xr 14.4+62.10+      14.19+       3.39   ; 47 y
bs 18.4+ 0.10+      14.19+       0.39   ; 48 z
gs 23.4+ 1.10+       1.19+32.35+ 3.39   ; 49 a
hs 24.4+ 4.10+       1.19+16.35+ 2.39   ; 50 b
is 25.4+ 6.10+       1.19+48.35+ 5.39   ; 51 c
ns  4.4+11.10+       1.19+       1.39   ; 52 d
ps  6.4+ 0.10+       1.19+       0.39   ; 53 e
us 11.4+ 0.10+       9.19+ 0.35+ 0.39   ; 54 f
bt 18.4+12.10+      14.19+       1.39   ; 55 g
gt 23.4+13.10+      14.19+       1.39   ; 56 h
it 25.4+14.10+       2.19+       2.39   ; 57 i
mt  3.4+16.10+      14.19+       1.39   ; 58 j
nt  4.4+            18.19+15.35         ; 59 t1
pt  6.4+            19.19+10.35         ; 60 t2
hv 24.4+            20.19               ; 61 m
ly  2.4+            21.19               ; 62 f
sy  9.4+             6.19               ; 63 CR
vy 12.4+             0.19               ; 64 0
```

```
[  actions first         p value for     Input values
   on line                 code bytes
   (switch on p value)                      Code    Between code]

d2: ; base first     ;                            Bytes copied when ǂ 1
                     ;                              Name when special action
qq c50     + 0.19    ;  0 digit            65 1
qq c14     + 0.19    ;  1 abcde            66 2
qq c15     + 0.19    ;  2 i                67 3
qq c50     + 0.19    ;  3 .                68 4
qq c51     + 0.19 f  ;  4 −                69 5       3 label
qq c52+6.14+ 0.19 f ,;  5 +                70 6       5 array
qq  c2     + 0.19 f  ;  6 CR               71 7       3 simple
qq c21     + 0.19 f  ;  7 ,                72 8       3 format
qq c23     + 0.19 f  ;  8 XVDIMNLKQTZPQ    73 9       3 procedure
qq c23     +16.19 f  ;  9 Ff       RABC    74 :       3 std proc
qq c23+6.14+17.19 f ,; 10 Sn)              75 =       5 constant
qq c23               ; 11 t                76 bl CR
qq c23     +22.19    ; 12 srp              77 e
qq c12+2.14+10.19   ,; 13 (                78 )       2 param comma
qq c23+2.14       f ,; 14 other letters    79         4 beg bounds
qq c12               ; 15 forbidden        80
qq c12               ; 16 :                81
qq c12               ; 17 =                82
qq c18               ; 18 t word           83
qq c18               ; 19 last t word      84
qq c19            f  ; 20 m                85         3 take array
qq c20               ; 21 f                86
qq c65+1.14        , ; 22 e                87         special endpass
```

```
[ actions              p value for       Input values
  mod, addract, defact   code bytes
  (switch on p value)                                Between code]

d3: ; base mod       ;                                Bytes copied when ≠ 1
                     ;                                Name when special action
qq c33               ;  0 digit           88
qq c33               ;  1 abcde           89
qq c33               ;  2 i               90
qq c12               ;  3 .               91
qq c33               ;  4 −               92
qq c33               ;  5 +               93
qq c22               ;  6 CR              94
qq c21               ;  7 ,               95
qq c27+6.14         ,;  8 XVDIMNLKQTZPQ   96          CR
qq c28               ;  9 Ff       RABC   97
qq c29               ; 10 Sn)             98
qq c30+2.14         ,; 11 t               99         2 case param
qq c31               ; 12 srp            100
qq c32+2.14         ,; 13 (              101         2 end case
qq c12+2.14         ,; 14 other letters  102         2 end addr case
qq c12               ; 15 forbidden      103
qq c12               ; 16 :              104 beg code (recognised by code:)
qq c12               ; 17 =              105
qq c12               ; 18 t word         106
qq c12               ; 19 last t word    107
qq c12               ; 20 m              108
qq c12               ; 21 f              109
qq c65               ; 22 e              110

d4: ; base addr act  ;

qq c35               ;  0 digit          111
qq c36               ;  1 abcde          112
qq c39               ;  2 i              113
qq c38               ;  3 .              114
qq c40               ;  4 −              115
qq c41               ;  5 +              116

d5=i−16; base defact ;

qq c16+1.14        f,; 16 :              117         0 code
qq c17+0.14        f,; 17 =              118         0 core

e                    ;  core block
```

```
[Head core code: Output in reverse order as first 6 instructions of
 each core code track]

     ps  s1    , hv  r-3   ;
b40: ca (c10)  ,hvr[n1head];
     gm (c10)  t   -1  MRC ;
     pm  s6            IRC ;
     arn c35   , hv  s2    ;
b41: qq[nhead] , hs  c7    ;
d29: [head pointer]        ;


[Tail core code: Output in normal order as last 4 instructions]

d28=i-1 [Tail pointer]     ;
     ps (c35)  , pm  r2    ;
b43: gm p[relA]V       MA  ;
b42: hv  -1    , qq[Ntail] ;
     gs  p1    , gs (r-2)  ;
d30: qqf [stop list]       ;


d9=e20-121[base code]      ; max code length = 119 words
d31=d9 [max list]          ;
```

[9.11.1967]
[GIER Algol 4, Pass 9, page 20]

[List of predefined c−names:
 qq<value> + <c−index>.16+<letter representation of c>.25,]

```
qq  c0+ 0.16+51.25, ;
qq  c1+ 1.16+51.25, ;
qq  c2+ 2.16+51.25, ;
qq  c3+ 3.16+51.25, ;
qq  c4+ 4.16+51.25, ;
qq  c6+ 6.16+51.25, ;
qq  c7+ 7.16+51.25, ;
qq  c8+ 8.16+51.25, ;
qq  c9+ 9.16+51.25, ;
qq c13+13.16+51.25, ;
qq c17+17.16+51.25, ;
qq c18+18.16+51.25, ;
qq c19+19.16+51.25, ;
qq c20+20.16+51.25, ;
qq c24+24.16+51.25, ;
qq c26+26.16+51.25, ;
qq c27+27.16+51.25, ;
qq c30+30.16+51.25, ;
qq c33+33.16+51.25, ;
qq c35+35.16+51.25, ;
qq c37+37.16+51.25, ;
qq c39+39.16+51.25, ;
qq c42+42.16+51.25, ;
qq c44+44.16+51.25, ;
qq c49+49.16+51.25, ;
qq c53+53.16+51.25, ;
qq c54+54.16+51.25, ;
qq c55+55.16+51.25, ;
qq c57+57.16+51.25, ;
qq c58+58.16+51.25, ;
qq c61+61.16+51.25, ;
qq c63+63.16+51.25, ;
qq c64+64.16+51.25, ;
qq c65+65.16+51.25, ;


d38=i−1 ; define initial top c

d39: qq ; first free

d e22=k−e14, e47=j   ;  running pass track and track relative
b k=e23,i=0          ;  load segment word
i=11e21              ;
qq e16.9+1d39.19−e16.19+9.24+1d40.39 ;
e    ;
i=d39;
e    ; final end
s    ;
```

```
d e49=5    ; tape number := 5;

[After i follows STOPCODE, SUMCODE and a sum character]
ia T4
s
```

[Here follows STOPCODE and CLEARCODE]

```
<e49−4, <−e49+6, ×  ;   test tape number
i wrong tape
s                   ;
>                   ;

d e55=11            ;   version number

<e55−e50,           ;   if version number T5 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=9             ;   define version number T2
d e63=10            ;   define version number T3
d e64=4             ;   define version number T4
d e65=e55           ;   define version number T5
d e66=6             ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e55
>                   ;
```

```
b k=e22+e14, i=e16−e47, a32, b20, c120, d50; begin pass 7
d i=e16                                     ;
d d37=−e20                                  ; max opand top:= 1023;

[outputvalues]
d10=124 [grt          ], d11= 70 [pm         ], d12= 71 [gm           ]
d13=123 [srt          ], d14=121 [arnt       ], d15= 97 [var to UA    ]
d16= 67 [xr           ], d17=102 [take formal], d18=104 [make assign  ]
d19=103 [contr formal ], d21= 45 [tk 1       ], d22= 36 [take forlabel]
                                                d32= 56 [pm UV        ]
d45= 86 [pm D         ], d46=112 [carret     ], d47=101 [move formal  ]
d48=116 [end pass     ]

a:    qq                ; current work table
[1a]  qq                ; byte 1
[2a]  qq                ; byte 2
[3a]  qq 15.9+1023.25   ; mask for outopand
[4a]  qq 97.9+1.23      ; VinW for get work
[5a]  qq  1.9           ;
[6a]  qq  4.9+4.23+4.39 ; VinUA
[7a]  qq 37.9+5.23+2.39 ; VinUV
[8a]  qq  1.39          ;
[9a]  qq  1.19          ;
[10a] qq  2.39          ;
[11a] qq 57.9+3.23      ; simple var
[12a] qq −3.9+6.23      ; stdproc, e.g. stdproc − simpel var
[13a] qq 33.9+1.23      ; var local for bounds
[14a] qq                ; work for outopand
[15a] qq −3.3−1.9−2.19−59.29+1.39+58.9; move array
[16a] qqf 1.23, qq      ; clear R and release
[17a] qq 40.29          ; multiplier for appetite
[18a] qq 0.39           ; for track
[19a] qq 21.12          ; (21/4).10, appetite word ratio
[20a] qq 320.39         ; appetite limit
[21a] qq −1.39−1.3      ; mask for no work variables
```

[Central interpreter

The central interpreter executes the actionbytes in the actionstack in this sequence:

```
                bottom, a−marked
                5 6 7 8   first word, no marks
                1 2 3 4   top word, no marks
```

A jump to NEXT will cause the execution of actionbyte 1.
4 actionbytes (one actionword) may be stacked in several ways:

1) When bottom is reached the next inputbyte is used to look−up the input control table. The tableword is stacked in the actionstack and table−marks are placed in RA.
2) a−actions cause a look−up in the auxiliary table and stacking of the tableword.
3) Certain c−actions perform explicit stacking of actionbytes.

The interpretion sequence may be changed by 1) stacking of actions, 2) clearing of top actionword (clear actions) 3) clearing of first action−byte (skip), 4) clearing of actionbyte 2, 3, 4 (nextskip).

An actionbyte is interpreted in one of 4 ways:

$-512 \leq$ byte ≤ -401	a−names	aux. look−up and stacking of tableword.
$-200 \leq$ byte ≤ -1	b−names	output (byte + 200)
$0 \leq$ byte ≤ 511	c−names	prepare action and jump to byte + c
$-400 \leq$ byte ≤ -201	d−names	output top operand, output (byte + 400 + 512 × mode) release top operand.

c−names will prepare the action by placing top operand in R, top marks in QC, nexttop in M, nexttop marks in marks. PA:= top −< const;
PB:= nexttop −> const

The notation operand(i) −< c will be used meaning class (operand(i)) = c. The outputbytes art, grt, etc. means ar, gr, etc. when mode is 0, arf, grf when mode is 1. The mode information is always added to outputvalues caused by d−names. This mode information is only of interest to pass 8 in a few cases.]

```
[c−1]pp d6    , ps  i     ; p:= stackbottom; always return to next;
c:
b1: pmn d7    VX   ITA    ; NEXT:
[current actionword]
[1c] pm d6    V           ; entry after nextin: M:= 0; marks:= 0;
    hv  c5         LA     ;    if actionword in bottom then goto nextin;
    hv (c)    Dt −1LZ     ;    if actionword = 0 then
a7: ga  b2    , cl  10    ;    begin unstack action word; goto next end;
    gr (c)    V    NZ     ;    actionbyte:= part 1(actionword);
a8: qq (c)    t−1         ;    actionword:= shift(actionword);
b2: pmn 0     DVXt200 LTA ;    if actionword = 0 then unstack actionword;
[actionbyte]
    hv  a1         IPC    ;    if actionbyte ≥ 0 then goto prepare action;
    pm (b2)   DVXt200 LT  ;    if actionbyte ≥ −200 then
                           ;    begin
c1: hv  e3                 ; OUTPUT: output(actionbyte + 200); return end;
    hv  a2         LT     ;    if actionbyte ≥ −400 then goto stack auxword;
c2:[mode]
b3: bs  0                  ; OUTINSTR REL: if floatmode then
    qq (b2)   t  512       ;    actionbyte:= actionbyte + 512;
c3: hs  c9                 ; OUTINSTR NO TYPE: call(outopand);
    pm (b2)   DX           ;    R:= actionbyte + 400;
c54:hs  e3                 ; OUTPUT RELEASE: output(R);
c4: pm  p     VX   NZB     ; RELEASE: if no release then
    arn 16a   V    IOB     ;    begin no release:= false; R:= 1.23; return end;
    pp  p−1                ;    [1.23 used in clear R]
    hr  s1         LC      ;    p:= p − 1; if operand[p+1] −< constant
    pi (p1)   t   −65      ;    then return;
    pa  b5    t d7 NC      ;    if operand[p+1] −< R then Rused:= not used;
    hr  s1         NTB     ;    if operand[p+1] −≠work then return;
    gt  a4    , udn a5     ;
b4:                        ;    release work location in
a4h:ns  0     , ck  s0     ;    worktable[blockrel[p+1] − wbase 1];
[wbase 1]
    sc  a     , hr  s1     ;    return

c5: pmn(e1)   Xt  1        ; NEXTIN: R:= input;
    hs  e2         LA      ;
    sr  512   DV   LT      ;    if R < 0 then begin R:= R + 512;
    pa  b3    Vt  0        ;    mode := 1 end
    pa  b3    t   1        ;    else mode:= 0;
    ga  a6                 ;    stack actionword(inputtable[R + base]);
a6: pmn 0     Xt d1 ITA    ;
    hv (c)    DVt 1 IRA    ;    RA:= inputtablemarks; goto after nextin;

a1: pm  p     X    IQC     ; prepare action: R:= top; QC:= marks;
    pm  p−1                ;    M:= nexttop;
    pi  16    t −17 LC     ;    PB:= nexttop −< constant;
    pi  32    t −33 LQC    ;    PA:= top −< constant;
    hv (b2)   t  c         ;    switch to action[actionbyte];
```

```
a2: pm (b2)   t  d3       ; stack aux word: M:= auxtable[actionbyte + 512];
c6: xrn(b1)   t  1 ITA    ; STACK ACTION:
    hv  a7                ;    stack actionword(M); goto next;

c7: pmn(b1)   X    ITA    ; NEXTSKIP: actionbyte:= part 1(actionword);
    ga  b2    , hvn a8    ;    goto unstack actionword;

c8: pm  p     VX          ; OUTPARAM: Radr:= part 1(top);
c9: arn p     , tk  14    ; OUTOPAND: Radr:= part 3(top); R25:= blockno;
    hv  a9        LC      ;    if top -< constant then goto outconst;
    pi (p)    t  -49      ;    PA:= output blockrel; PB:= output blockno;
    mb  3a    , ga  14a   ;    clear R0 - 5, 26 - 39; byte := Radr;
    nc  3     , hv  a10   ;    if Radr ǂ variabel then goto outvar;
    tk  16    , nc (b9)   ;    if blockno ǂ current block
    hv  a10       LO      ;    ∧ blockno < 0 then goto outvar;
    pa  14a   t  2 IPB    ;    PB:= false; byte := if blockno = 0 then
    pa  14a   t  1 LO     ;    varabs else var local;
a10:pm  p     , tl  -20   ; outvar:
    cln -10   , pm  p     ;    if PA then output(blockrel);
a11:hs  e3        LPA     ;
    cln -10               ;    if PB then output(blockno);
    hs  e3        LPB     ;
    arn 14a   , ca  11    ;    if byte  = array word 1 then
    pp  p-1   , hv  c8    ;    begin p:= p - 1; goto outparam end;
    hv  e3                ;    output(byte 1); return;

a9: pan 14a   Xt 8        ; outconst: byte := constant
    arn p     , cl  30    ;
    hs  e3                ;    output(top 30-39)
    cln -10       M       ;    PC:= true;
    hs  e3        IPC     ;    output(top 20-29);
    cln -10   , hv  a11   ;    goto output 2 bytes and byte ;

c10:                      ; CLEAR R:
b5: pm  d7    X    ITA    ;    TA:= operand[Rused] = buf R;
[Rused]
    hr  s1        LA      ;    if R not used then return,
    ca  1     , it 512d10 ;    actionbyte:= if operand[Rused] = VinRF
    pa  b2    t  d10      ;    then grf else gr;
    gp  a12   , pp (b5)   ;    save p; p:= Rused;
    hs  c11               ;    get work;
a30:gm  p         MB      ;    operand[Rused]:= VinW;
                          ;    [executed from c16-4]
    hsn c3        IZB     ;    no release:= true; outinstr no type; R:= 1.23
    qq        V   NTA     ;
<e27[bufmode]             ;    if TA then
    pa  p     t 609       ;    operand[Rused]:=
×   [FL mode]             ;    if bufmode then bufw else AinW, clear;
    sc  p         MA      ;
>   pa  b5    t  d7 M     ;    Rused:= not used; marks:= R not used;
a12:pp  0     , hr  s1    ;    p:= saved p; return;
```

```
[The worktable consists of a single machine word. Bit 0 is always 1. Bit
n is 1 if working location n is occupied, 0 otherwise. The instruction nk
with work table in R will shift R until first free working location is in
bit 1.]

c11:arn a     , nk  a13   ; GET WORK: i:= first free workbit;
a5: ar  256   D           ; [executed from b4-1]
a13:tk  0     , gr  a     ;    work[i]:= occupied;
b6h:it (a13)  t  0        ;    blockrel:= i:= i + wbase 2;
[wbase 2]
    pt  4a    , pm  4a    ;    M:= final word descr;
    it (a13)  , bs (b7)   ;    if blockrel < min work then
b7: qq  0     t  -1       ;    min work:= min work - 1;
[min work]
    hr  s1                ;    return;

c12:pa  2a    , hv  c15   ; CLEAR 0: byte 2:= 0; goto clear byte 2;
c13:pa  2a    Vt 1        ; CLEAR UAUV: byte 2:= 1; goto clear byte 2;
c14:pa  2a    t  -512     ; TOTAL CLEAR: byte 2:= min;
c15:hs  c10               ; CLEAR BYTE 2: clear R;
    gp  a14   , gp  b8    ;    save p; clear count:= p;
b8: pmn 0     X           ;    if operand[clear count] -< no clear then
[clear count]
    hv (b8)   Dt -1 LB    ;    begin clear count:= clear count - 1; goto clear end;
    ck  20    V    LA     ;    if operand[clear count] -< R then
a14:pp  0     , hr  s1    ;    begin p:= saved p; return end;
    gt  r1    , pp (b8)   ;    p:= clear count;
    it (2a)   , bs  0     ;    if blockno [p] ≤ byte 2 then
    pa  b2    Vt d11      ;    begin
    hv (b8)   Dt -1       ;    clear count:= clear count-1; goto clear end;

    hs  c3                ;    outinstr no type(pm);
    hs  c11               ;    get work; p:= p + 1;
    pp  p1    , ud  a30   ;    operand[p]:= VinW;
    pa  b2    t  d12      ;
    hsn c3         IZB    ;    no release:= true; outinstr no type(gm);
    hv (b8)   Dt -1       ;    clear count:= clear count - 1; goto clear;

c16:qq             IPC    ; EXCHANGE: PC:= marks(nexttop);
    gm  p          MPC    ;    top:= M;
    gr  p-1        MQC    ;    nexttop:= R;
    qq (b5)   Vt 1 NPC    ;    if top-< R then Rused:= Rused + 1 else
    qq (b5)   t -1 NQC    ;    if nexttop -< R then Rused:= Rused - 1;
    hr  s1                ;    return;

c17:hv  c4         LZ     ; SRT: if top = 0 then goto release;
    pa  b2    t  d13      ;    actionbyte:= srt;
    hv  c2                ;    goto outinstr rel;
```

```
c18:arn 5a    , hv  a15   ; STACK VINRF: R:= VinRF; goto stack;
c19:hvn a15               ; STACK VINR: R:= VinR; goto stack;
c20:pm 514    DVX         ; STACK BUFR: R:= bufR; goto stack;
c21:pmn(b3)   DX          ; STACK VINRT: R:= mode;
a15:gr  p1         M      ; stack: operand[p+1]:= R, marks 0;
    pp  p1    , gp  b5    ;    p:= p + 1; Rused:= p;
    hr  s1                ;    return;

c22:pm  6a    V           ; STACK VINUA: M:= VinUA; skip line;
c23:pm  7a                ; STACK VINUV: M:= VinUV;
    gm  p1         MA     ;    operand[p+1]:= M; marks clear;
    pp  p1    , hr  s1    ;    p:= p + 1; return;

c24:pm  p     , gi  a28   ; TOP TO RT: save ZB;
    hv  c4         NC     ;    if top -< R then goto release;
    hs  c10               ;    clear R;
    pa  b2    t  d14      ;    actionbyte:= arnt
a28:pi  0     , hv  c2    ;    reset ZB; goto outinstr rel;

c25:hv  c4         NQC    ; TOP TO R: if top -< R then goto release;
    pa  b2    t  d14      ;    actionbyte:= arn;
    hv  c3                ;    goto outinstr no type;

c26:ca  4     , hv  c37   ; TOP TO UA: if top = UA then goto count p1
    pa  b2    t  d15      ;    actionbyte:= var to UA;
    hv  c3                ;    goto outinstr no type;

c27:pm  d16   DX   NQC    ; TOP TO M: if top -< R then
    hv  c54        NQC    ;    begin R:= xr; goto outputrelease end;
    hs  c10               ;    clear R;
    pa  b2    t   d11     ;    actionbyte:= pm;
    hv  c3                ;    goto outinstr no type;

c28:hsn c24        IZB    ; TO FIX AND RT: no release:= true; top to Rt;
    hr  s1         LQC    ;    if top -< const then return;
    can(b3)   , hv  c10   ;    if fixmode then goto clear R;
    hs  c4                ;    release;
    hs  c19               ;    stack VinR;
    hv  c10               ;    goto clear R;

c29:ptn b10   Vt d17      ; TAKE FORMAL: take:= take formal;
c30:pt  b10   Vt d18      ; TAKE ASSIGN: take:= take assign;
c31:pt  b10   t d19 NZ    ; TAKE CONTROLLED: take:= controlled formal;
c104:arn p    , ck -10    ; TAKE OTHER:
b9: ca  1     , hv  a16   ;    if blockno[p] = current block then goto take;
[current block]
    pa  b2    t  d15      ;    actionbyte:= var to UA;
    hs  c3                ;    call(outinstr no type);
    hs  c22               ;    stack VinUA;
a16:
b10h:pa b2    t  0        ; take: actionbyte:= take;
[take]
    hv  c3                ;    goto outinstr no type;
```

```
c32:qqn            IZB    ; WHILE ASSIGN: no release:= true;
c33:pmn d8    VX          ; COLON EQUAL: R:= gm, grn, grt; skip line;
c34:pmn d9    X    IZB    ; FOR ASSIGN: R:= gm M, grn M, grt M;
b11:ck  0     , ga  b2    ;    actionbyte:= R[assign kind];
[assignkind]
    hv  c2                ;    goto outinstr rel;

c56:nc  37          NPA   ; SPARE ASSIGN: if top = VinUV then
c35:pa  b19   Vt   12     ;     begin from UV:= only UV:= true; goto next end;
    pa  b19   , hv  c4    ; PREPARE ASSIGN:
    pa  b11   t 20 IZA    ;    from UV:= only UV:= false; ZA:= top = 0;
    hv  c4          NQC   ;    assignkind:= 20; if top -< R then goto release;
    arn(b5)   , ca  1     ;    if RF used then
    hs  c10               ;    clear R; comment used in move value;
    pa  b11   Vt 10 LZA   ;    if top = 0 ∧ R not used then
    qq        V           ;    begin assignkind:= 10;
    hv  c4          LA    ;    goto release end;
    pa  b2    Vt d11 NRA  ;    actionbyte:= if RA then
    pa  b2    t  d45      ;    pm D else pm;
    pa  b11   , hv  c3    ;    assignkind:= 0; goto outinstr no type;

a17:arn 8a    , ac  e4    ; count line: linecount:= linecount + 1;
    pm  d46   DX          ;
    hs  e3                ;    output(carret);
c36:pmn(e1)   Xt  1       ; READ CARRET:
    hs  e2          LA    ;    if input = carret then
    ca  d20   , hv  a17   ;    goto count line;
    qq (e1)   t  -1       ;    save inputbyte;
    sr  512   D    LT     ;    if R < 0 then R:= R + 512;
    ga  r1    , it  d1    ;    M:= inputtable[R + base];
    pm  0     , hr  s1    ;    return;

c37:pp  p-1   , hr  s1    ; COUNT P1: p:= p - 1; return;

c38:pmn(e1)   Xt  1       ; INOUT 1:
    hs  e2          LA    ;    output(input);
    hv  e3                ;    return;

c39:pmn(e1)   Xt  1       ; INPUT 2:
    hs  e2          LA    ;    byte 1:= input;
    ga  1a                ;
c40:pmn(e1)   Xt  1       ; INPUT 1:
    hs  e2          LA    ;    byte 2:= input;
    ga  2a    , hr  s1    ;    return;

c41:hrn s1          IZB   ; NO RELEASE: no release:= true; return;

c42:pa  b16   Vt  -5      ; SET UV rel: UV rel:= -5; return;
c43:pm  d21   DVX  NQC    ; JUMP TO BOOLEAN: if top -< R then
c44h:hr s1    , hs  e5    ;    output(tk1); return;
    hv  e3    , qqn e34   ; STACK: alarmprint(ŧ<stackŧ);
[c45, see page 8]
c109:pa b3    Vt  1       ; SET FLOATMODE: mode:= float; return;
c46:hv  c16         NC    ; PLUS EXCHANGE: if nexttop -< R then
    hv  c                 ;    goto exchange; goto next;
```

```
c47:pm  a     , gm  p1    ; BLOCKSTACK: stack wordtable;
    pm (b7)   D           ;    stack min work,
    gm  p2        M       ;    wbase 2, shield;
    arn b6    , gt  p2    ;
    it (2a)   , pa  b7    ;    min work:= byte 2;
    it (2a)   t  −1       ;    wbase 1:=
    pa  b4    , it (2a)   ;    wbase 2:= byte 2 − 1;
    pt  b6    , pp  p2    ;
    qq (b9)   t  −1       ;    current block:= current block − 1;
    pm  512   D           ;    worktable:= no reserved;
    gm  a     , hr  s1    ;    return

c48:gm  a     , gt  b6    ; BLOCKUNSTACK: worktable:= nexttop;
    pm (b7)   Dt −2       ;    wbase 2:= part 2(top); M:= min work − 2;
    ga  b7    , ck  10    ;    minwork:= part 1(top);
    ga  b4    , pp  p−2   ;    wbase 1:= part 2(top); count p2;
    hs  e3    X           ;    output(M); comment old min work − 2;
    pm (b9)   DX          ;    output(current block);
    qq (b9)   t  1        ;    current block:= current block + 1;
    hv  e3                ;    return;

c49:pan b2    t d22 IZB   ; GET FORLABEL: no release:= true;
    pp  p−1   , hs  c3    ;    p:= p − 1; outinstr no type(take forlabel);
    pm  9e4   , gm  18a   ;    for track:= outputtrack;
    it (e12)  , pa  b17   ;    for word:= out addr;
    pp  p1    , hh  c−1   ;    p:= p + 1; goto next;

c50:hs  c11               ; FOR: M:= get work;
    gm  p1        MB      ;    stack for label work;
    pp  p1    , hv  c     ;    goto next;

c51:ca  55    , hv  c     ; PREP FOR ASSIGN: if top = formal then goto next;
c52:ca  55                ; ADDRESS: if top = formal then
    hvn c29       IZB     ;    begin no release := true; goto take formal end;
    acn p         MB      ;    top:= no clear;
c45:grn(b1)   , hv  c     ; CLEAR ACTIONS: current actionword:= 0;

c53:pp  p1    , hs  c11   ; UA TO WORK: p:= p + 1; get work;
    tl  −20   , tl  20    ;
    gm  p         MB      ;    operand[p]:= AinW;
    hs  c8    , qq  c−1   ;    outparam; prepare return to next;
    pm  d12   DX          ;    output(gm);
    hv  e3                ;    return;
[c54, see page 3]
c55:arn 8a    , ac  e4    ; CAR RET: linecount:= linecount + 1;
    hr  s1                ;    return;
[c56, see page 7]
```

```
c57:pm  d25   V    LQC    ; UNTIL: aux 0:= if top −< const then
    pmn d26               ;    release step else mt − step;
    gm  d27   , bs (b3)   ;    if floatmode ∧ top −< const then
    arnf p        LQC     ;    RF:= stepvalue;
    pa  d27   t c16  −c LT;    if step −< const ∧ step < 0 then
    hs  c28               ;    part 1(aux 0) := exchange; to fix and Rt;
    pmn p−1   X    IZB    ;    no release:= true;
    mb  21a               ;
    gr  p1        MB      ;    operand[p+1]:= operand[p+2]:=
    gr  p2        MB      ;    operand[p+3]:= controlled variable ∧ no work;
    gr  p3        MB      ;
c58:pp  p3    , hv  c     ; P + 3: p:= p + 3; goto next;

c59:arn p−2   , gr  p−1   ; FIRST SUBSCRIPT:
    gr  p1        MB      ;    nexttop:= operand[p+1]:= var, blockno, no clear;
    cln −20   , gt  p−1   ;    blockrel[nexttop]:= doperel;
    ar  9a    , gt  p1    ;    blockrel[p+1]:= coef 2 rel;
    pp  p1    , hv  c     ;    p:= p + 1; goto next;

c60:grn p1    V    M      ; BEGIN CALL: no of literals:= 0; stack shield;
c61:arn 9a    , ac  p1    ; NOT LAST SUBSCR: coefrel:= coefrel + 1;
    pp  p1    , hv  c     ;    p:= p + 1; goto next;

c62:arn p−2   , sr  9a    ; LAST SUBSCR:
    gr  p−1               ;    nexttop:= constant term;
    hs  c36               ;    read carret;
    hv  c         NB      ;    if −, special byte then goto next;
    ca  d28   , pmn d29   ;    current actionword:=
    pm  d30       NZ      ;    if next byte = address then subscr:=
    gm (b1)   , hv  c     ;    else subscrparam; goto next;

c63:hv  c         NPA     ; ROUND: if top −≮ const then goto next;
    arnf p    , tkf −29   ;    top:= round(top);
a18:gr  p     , hv  c45   ;    goto clear actions;

c64:hv  c         NPA     ; FLOAT: if top −≮ const then goto next;
    nkf 39    , grf p     ;    top:= float(top);
    grn(b1)   , hv  c     ;    clear actions; goto next;

c65:hv  c         NPA     ; NEGATIVE: if top −≮ const then goto next;
    srnf p    , bs (b3)   ;    if floatmode then
    grf p     , hv  c45   ;    top:= − floattop else
    srn p     , hv  a18   ;    top:= − fixtop; goto clear actions;

c66:
b19:pi  −1    t   −13     ; ASSIGN: QA:= −, from UV; QB:= −, only UV;
    hv  a32       LT      ;    if top −< buffer then goto buf assign;
    pm  d32   DX  NQB     ;    if only UV then
    hs  e3        NQB     ;    begin output(pmUV); only UV:= false;
    qq (b19)  Xt 4 NQB    ;    assignkind:= gm
    pa  b11       NQB     ;    end;
    ca  55    , hv  c30   ;    if top = formal then goto take assign;
    grn(b1)   , hv  c33   ;    current actionword:= 0; goto colon equal;
a32:pm  d31   , gm (b1)   ; buf assign: current actionword:= buf assign;
    qq (b19)  Vt −8LQA    ;    if from UV then goto skip;
    pa (b1)   , hv  c     ;    from UV:= true;
    hv  c23               ;    goto stack VinUV;

c67:bs(b11)   t   10      ; AFTER BUF ASSIGN:
    pa  b19               ;    if assignkind = grt then only UV:= from UV:= true;
    hv  c                 ;    goto next;
```

```
c68:ck  −10   , ca (b9)   ; GOTO: if blockno(top) = current block then
    hv  c7                ;    goto nextskip;
c69:pa (b1)   , hv  c     ; SKIP: clear next action; goto next;

c70:can(b3)         LPC   ; PLUSMINUS: if fix mode
    hv  c24               ;    ∨ not const operands then goto top to Rt;
    arnf p−1  V     NRA   ;    nexttop:= if add then top + nexttop
    srnf p−1              ;    else top − nexttop;
    arf p     , grf p−1   ;
a19:pp  p−1   , hv  c45   ;    p:= p − 1; goto clear actions;

c71:arnf p    V     LPC   ; MULTDIV: if not const operands then
    hv  c24               ;    goto top to Rt;
    mkf p−1   V     NRA   ;    nexttop:= if mult then top × nexttop
    dkf p−1               ;    else top / nexttop;
    grf p−1   , hv  a19   ;    p:= p − 1; goto clear actions;

c72:sr  10a   V     LQC   ; INTEG EXPON:
    hv  c111              ;    if top −∤ constant then goto set↑track;
    pm  d33         IZA   ;    ZA:= top = 2;
    sr  8a          NZ    ;    if top ∤ 2 ∧ top ∤ 3 then
    hv  c111        NZ    ;    goto set↑track;
    tl  59    X     LZA   ;    current actionword:= if ZA then
    gm (b1)   , pp  p−1   ;    mkf − word else mkf − mkf − word; p:= p − 1;
    pa  b3    t  1        ;    mode:= float;
    pm  d34   , hv  c6    ;    M:= ↑ integer first; goto stack action;

c73:pa  b12   Vt  2       ; EQUAL: relationkind:= ∤ = ∤; goto greater;
c74:pa  b12   t  4        ; NOT EQ: relationkind:= ∤ ∤ ∤;
c75:                      ; GREATER:
b12:pmn 0     Vt d35 NC   ;    M:= if nexttop −< R then table [relkind+1]
[relationkind]
    pm (b12)  Vt d36 LZ   ;    else if top = 0 then table[rel kind +2]
    pm  d2    V     NZ    ;    else table[0];
    pm  d2          NQC   ;    if top −< R then M:= table[0];
    pa  b12   , hv  c6    ;    relkind:= ∤ > ∤; goto stackaction;

c76:ca  0     V           ; REAL: if top = R then clear R;
c77:ca  1                 ; INTBOOL:
    hs  c10         NQC   ;    if top = RF then clear R;
    hs  c36               ;    read carret;
    hv  c45         NB    ;   if −, address operator then goto clear actions;
    hv  c                 ;    goto next;

c78:hv  c7          LQC   ; NEXTSKIP ON CONST: if top −< const then
    pa (b1)   , hv  c24   ;    goto nextskip; skip; goto top to Rt;

s
```

```
c79:bs  pd37  , hh  c44   ; OPAND: if p ≥ max opandtop then goto stack;
    pp  p1    , arn 2a    ;    p:= p + 1;
    ck  10    , ar  11a   ;    operand[p]:= simpel, byte 1, var, byte 2;
    qq        V    NRA    ;
    gr  p     V    MA     ;    marks[p]:= if RA then clear
    gr  p          MB     ;    else no clear;
    it (1a)   , pt  p     ;
    hr  s1                ;    return;

c80:hs  c36               ; PROC: read carret
    pa  p     Vt   55     ;    part 1(top):= descr or call; skip line;
c81:arn 12a   , ac  p     ; STDPROC: top:= stdproc; goto next;
    hv  c          LB     ;    if special byte then goto next;
    pm  d38   , it  1     ;    stack actionword (proc no param);
    gm (b1)   , hv  c8    ;    goto outparam;

c82:pm  1a    , nt (2a)   ; ARRAY: M:= byte 1;
    pa  1a    , gm  2a    ;    byte 1:= − byte 2; byte 2:= M;
    hs  c79               ;    stack operand;
    pa  p     t  59       ;    part 1(top):= array word 1;
    pa  p−1   t  58       ;    part 1(nexttop):= array;
    hv  c13               ;    goto clear UAUV;

c83:pa  p     t  55       ; FORMAL: part 1(top):= formal;
    hs  c36               ;    read carret;
    hv  c          LB     ;    if special byte then goto next;
    hs  c29               ;    take formal;
    hv  c22               ;    goto stack VinUA;

c84:bs  pd37  , hh  c44   ; CONST 1: if p ≥ max opand top then goto stack;
    pp  p1    , grn p     ;    p:= p + 1; top:= 0;
c85:arn 2a    , ck  −10   ; CONST 2:
    ar  1a                ;    top:= top + (if const 1 called then
    nt  20    , ck  60    ;    byte 1, byte 2, 0, 0 else 0, 0, byte 1, byte 2);
                          ; [nt 0 must never appear]
    ac  p          MC     ;    marks:= constant; goto next;
c87h:hv c     , hs  e5    ; OVERFLOW:
    qq        , qqn a21   ;    alarmprint(ǂ<spill>);

c86:qqn(p−1)  Vt 1 LQC    ; PARAM: if top −< const then
    pm  d41   V    NQC    ;    no of literals:= no of literals + 1;
    nc  59    , hv  a31   ;    if top = VinR ∨ top = VinRF then
    hv  c          LT     ;    begin M:= Rt expr word; goto stack action end;
    nc  59    , hv  c6    ;    if top ǂ array then goto check block no;
    arn 9a    , sc  p−1   ;    blockrel:= blockrel − 1;
    tl  9     , tln 10    ;    no of indic:= no of indic + 1;
    sr  9a    , ar  10a   ;    doperel:= doperel to arrayw;
    sc  p     , hv  c     ;    goto next;
a31:ck  −10   , ca  1     ; check blockno:
    arn 8a    , sc  p     ;    if blockno = 1 then top:= top − 1;
    hv  c                 ;   goto next;
```

```
b k=e31, i=0              ;    storing of text for errorprint
i=e32                     ;
a21: tspill;              ;
e32=i                     ;
e                         ;

c88:it  p6    , pa  b13   ; BEGIN BOUNDS: length position:= p + 6;
    pp  p10               ;    p:= p + 10;
    bs  pd37  , hh  c44   ;    if p ≥ max opandtop then goto stack;
    grn p           M     ;    shield;
    arn 13a   , pm  8a    ;
    gr  p−9         MB    ;    array address:= var local;
    gr  p−8         MB    ;    term address:= var local;
    gr  p−4         MB    ;    length address:= var local;
    gr  p−1         MB    ;    running coef:= var local;
    gm  p−3         MC    ;    length:= 1;
    grn p−6         MC    ;    term:= 0;
    arn 1a    , ck  −10   ;    blockrel(length address):= byte 1;
    gt  p−4   , sr  9a    ;    blockrel(running coef)
    gt  p−1   , gt  p−9   ;    := blockrel(array address)
    gt  p−8   , it (2a)   ;    := blockrel(term address):= byte 1 − 1;
    pa  b14   , hv  c     ;    no of arrays:= byte 2; goto next;

c89:it (2a)   t  d4       ; SET TYPE: array type:= byte 2 + base;
    pa  b15   , hv  c     ;    goto next;

c90:gm  p−9         MC    ; BOUNDS: li2:= li, constant;
    sr  p−1   , ar  8a    ;
    gr  p−2         MC    ;    ci:= ci 1:= ci 2:=
    gr  p−4         MC    ;    upper − lower + 1, constant;
    gr  p−7         MC    ;
    arn 9a    , ac  p−3   ;    increase running coef address;
    hv  c93         LPC   ;    if top −< const ∧ nexttop −< const then
                          ;    goto count p2;
    pm  p−3               ;
    gm  p−7         MB    ;    ci 2:= running coef;
    grn p−4         M     ;    ci 1:= VinR;
    grn p−2   , srn 8a    ;    ci:= if li −< const then 0 else −1;
    ac  p−1   V     LPB   ;    if li−< const then li:= li − 1;
    gr  p−2               ;
    pm  d39   , hv  c6    ;    M:= aux var bounds; goto stack action;

c91:arn 21a   , mb  p2    ; END VAR BOUNDS:
    hv  c21         LC    ;
    gr  p−6         MA    ;    if li −∤ const then li 2:= li ∧ no work, clear;
    hv  c21               ;    goto stack VinRt;

c92:pm  p−7         IQC   ; FIRST BOUND:
    gm  p−5         MQC   ;    ci 2:= li 2; comment p:= address(ci 1);
c93:pp  p−2   , hr  s1    ; COUNT P2: p:= p − 2; return;
```

```
c94:mln p     V    LPC    ; LENGTH MULT:
    hv  c                 ;    if ci 1 −∤ const ∨ length −∤ const then goto next;
    tl  39    , gr  p−1   ;    length:= length × ci 1;
    pp  p−3   , hv  c45   ;    p:= address(ci2); goto clear actions;

c95:mln p     V    LPC    ; TERM MULT:
    hv  c                 ;    if ci2 −∤ const ∨ term −∤ const then goto next;
    tl  39    , ar  p−2   ;    term:= term × ci2 + li 2;
    gr  p−1   , pp  p5    ;    p:= address(ci);
    grn(b1)   , hv  c     ;    clear actions; goto next;

c96:gr  p1          MB    ; STORE LENGTH: length:= length address;
    pa  b2    t  d10      ;    actionbyte:= gr;
    hv  c3                ;    goto outinstr no type;

c97:gr  p2          MB    ; STORE TERM: term:= term address;
    pa  b2    t  d10      ;    actionbyte:= gr;
    hs  c3                ;    outinstr no type;
    pp  p9    , hv  c     ;    p:= address(ci); goto next;

c98:arn d40   , pm  p−3   ; END BOUNDS: R:= if length −< const then
    tk  10          NA    ;    aux end bounds else aux end bounds shift 10;
    pp  p−3         NA    ;    p:= if length −< const then address(length)
    pp  p−3   , pm  p     ;    else address(term),
    hv  c6    X     LC    ;    if term −< constant then goto stack action;

c99:arn 9a                ; DECLARE ARRAY: p:= length position;
b13:pp  0     , sc  p−5   ;    decrease array address;
[length position]         ;
b14:bt  0     t  −1       ;    no of arrays:= no of arrays − 1;
[no of arrays]            ;    if no of arrays ≥ 0 theN
b15:pm  0     , hv  c6    ;    begin M:= declare word[array type];
[array type]              ;    goto stack action end;
    pp  p−6   , hv  c     ;    release boundvariables; goto next;

c100:srn 2a   , mt  b18   ; ENDPASS: output(if expontrack then
b18:qq  −1    , hs  e3    ;    − no of std tracks − 1 else
    [expon track]         ;    no of std tracks + 1);
a22:pm  1e20  Xt  −1      ;
    hv  a22         LZ    ;    pass inf 1:= address of last nonzero opand;
    nt  d6    , it (a22)  ;    comment a few more locations may be
    pa  2e4   , hhn e29   ;    used for 0 and VinR; goto endpass;

c101:bt 0     t  −100     ; CODE: for i:= 1 step 1 until 5 do
     pa  r−1  , hv  e3    ;
     hs  c102             ;    copy byte 2;
     arn 2a   , hv  c101  ;    output(no of 5 bytes); return;

c102:it(2a)   , pa  a23   ; COPY BYTE 2:
a23:can 0     , hr  s1    ;    for i:= 1 step 1 until byte 2 do
    pm (e1)   Xt 1        ;    output(input);
    hs  e2          LA    ;
    hs  e3                ;
    hv (a23)  Dt −1       ;    return;
```

```
c103:hr s1         LQC    ; CASEPARAM: if top −< const
     ca  57   , hr  s1    ;    ∨ top = simpel then return;
     pm  d42  , hv  c6    ;    M:= caseparamword; goto stack action;

[c104: take other, see c31]

c105:hs c23               ; STACK UV rel: stack VinUV;
b16:it  −5     t   1      ;    UV rel:= UV rel + 1;
[UV rel]                  ;
    pt  p      , hv  c    ;    part 2(top):= UV rel; goto next;

c106:ca 55     , pa (b1)  ; TEST FIRST FORMAL: if top = formal then skip;
    hv  c69        NQC    ;    if top −< R then goto skip;
c107:nc 55     , hv  c    ; TEST FORMAL ADDRESS:
    pt  b10    t  d47     ;    if top ǂ formal then goto next;
    hs  c104              ;    take:= move formal; take other;
    pm  d44    , hv  c6   ;    M:= move formal word; goto stack action;

c108:nc 59     , hv  c45  ; ARRAY PARAM: if top ǂ array then
    ck  20     , ar  15a  ;    goto clear actions;
    gr  p          MC     ;    top:= −1.9 + (doperel − arrayrel − 2).19
    tln 19     , tk  20   ;          + (no of indic+1).39, constant;
    sc  p      , hv  c    ;    goto next;

[c109: set float mode, see c46]

c110:arn 9e4   , sr  18a  ; APPETITE:
    pi (16e4) t   −17     ;    R:= outputtrack − for track;
    tl  −1         LPB    ;    if discmode then R:= R : 2;
    ar  4e4        LT     ;    if R < 0 then R:= R + available;
    pm (e12)  DX          ;    M:= R;
    bs (e12)  t 123  e13  ;    R:= if outaddr > buflimit 3 then
    sr  41    D           ;    outaddr − 41 else outaddr;
b17:sr [for word] D       ;
    bs (b17)  t 123  e13  ;    R:= R − (if for word > buf limit 3 then
    ar  41    D           ;    for word − 41 else for word);
    tk  −20   , ml  17a   ;    M.29 := R + M × 40;
    mkn 19a   , sr  20a   ;    R:= M × appetite word ratio;
    hvn e3         NT     ;    if R > appetite limit then
    ar  20a   , ck  −10   ;    output(0) else output(R);
    hv  e3                ;    return;

c111:pa b18   , hr  s1    ; SET ⅄ TRACK: expontrack:= true; return;
```

```
<c111−c−511   ; all actions within 511 words
i actions too big
>
d7: qq  ,  qq ; bottom of ACTIONSTACK
    qq
    qq
    qq
    qq

[form actionaddresses relative to c]
  c1=  c1−c
  c2=  c2−c,   c3=  c3−c,   c4=  c4−c,   c5=  c5−c,   c6=  c6−c,   c7=  c7−c
  c8=  c8−c,   c9=  c9−c,  c10= c10−c,  c11= c11−c,  c12= c12−c,  c13= c13−c
 c14= c14−c,  c15= c15−c,  c16= c16−c,  c17= c17−c,  c18= c18−c,  c19= c19−c
 c20= c20−c,  c21= c21−c,  c22= c22−c,  c23= c23−c,  c24= c24−c,  c25= c25−c
 c26= c26−c,  c27= c27−c,  c28= c28−c,  c29= c29−c,  c30= c30−c,  c31= c31−c
 c32= c32−c,  c33= c33−c,  c34= c34−c,  c35= c35−c,  c36= c36−c,  c37= c37−c
 c38= c38−c,  c39= c39−c,  c40= c40−c,  c41= c41−c,  c42= c42−c,  c43= c43−c
 c44= c44−c,  c45= c45−c,  c46= c46−c,  c47= c47−c,  c48= c48−c,  c49= c49−c
 c50= c50−c,  c51= c51−c,  c52= c52−c,  c53= c53−c,  c54= c54−c,  c55= c55−c
 c56= c56−c,  c57= c57−c,  c58= c58−c,  c59= c59−c,  c60= c60−c,  c61= c61−c
 c62= c62−c,  c63= c63−c,  c64= c64−c,  c65= c65−c,  c66= c66−c,  c67= c67−c
 c68= c68−c,  c69= c69−c,  c70= c70−c,  c71= c71−c,  c72= c72−c,  c73= c73−c
 c74= c74−c,  c75= c75−c,  c76= c76−c,  c77= c77−c,  c78= c78−c,  c79= c79−c
 c80= c80−c,  c81= c81−c,  c82= c82−c,  c83= c83−c,  c84= c84−c,  c85= c85−c
 c86= c86−c,  c87= c87−c,  c88= c88−c,  c89= c89−c,  c90= c90−c,  c91= c91−c
 c92= c92−c,  c93= c93−c,  c94= c94−c,  c95= c95−c,  c96= c96−c,  c97= c97−c
 c98= c98−c,  c99= c99−c, c100=c100−c, c101=c101−c, c102=c102−c, c103=c103−c
c104=c104−c, c105=c105−c, c106=c106−c, c107=c107−c, c108=c108−c, c109=c109−c
c110=c110−c, c111=c111−c
```

```
[prepare symbolic names used in tables]
a=−512, b=−200, d=−400 ;

     [relationtable]

[all relations]
d2:  qq c24 t c17                   ; a rel b, etc: top to Rt, srt
[>]
d35: qq c17 t   c4.9+ 53b.19        ; R rel b, R rel 0: srt, release, mt neg
d36: qq  c4 t  c10.9+125d.19        ; a rel 0, 0 rel 0: release, clear R, srnt
[=]
     qq c17 t   c4                  ; R rel b, R rel 0: srt, release
     qq  c4 t  c10.9+ 77d.19+ c69.29; a rel 0, 0 rel 0: release, clear R, ann, skip
[‡]
     qq c17 t   c4                  ; R rel b, R rel 0: srt, release
     qq  c4 t  c10.9+ 76d.19+ c69.29; a rel 0, 0 rel 0: release, clear R, snn, skip

     [auxiliary table 1]
d3=i+112                            ;
d27: qq 0                           ; 0, stepelem: aux 2 or aux 3
     qq c32 t  c49.9+ 29b.19+ 31b.29; 1, simple for: while assign, get forlabel,
                                    ;                do abs, bypasslabel
d25: qq   0 t  c24.9+ c17.19+  c4.29; 2, until: next, top to Rt, srt, release
d26: qq   0 t  c24.9+ c17.19+ 75d.29; 3, until: next, top to Rt, srt, mt
<e27 [bufmode∧check:]               ;
d29: qq  6a t  c20.9+ c40.19        ; 4, subscr:= : aux 6, stack buf R, input 1
d30: qq  6a t  c20.9+ 50b.19        ; 5, subscr param: aux 6, stackbuf R, il 0
×    [FL mode∧check:]               ;
d29: qq  6a t  c20.9+ c10.19        ; 4, subscr:= : aux 6, stackbuf R, clear R
d30: qq  6a t  c20.9+ 59b.19        ; 5, subscr param: aux 6, stackbuf R, ga UA
>                                   ;
     qq c28 t  c37.9+ c17.19+  7a.29; 6, checkbounds 1: to fix and Rt, count p1
                                    ;                   srt, aux 7
     qq100d t  c58.9+ 99d.19+  8a.29; 7, checkbounds 2: indexlower, p+3
< e27 [bufmode:]                    ;                   indexupper, aux 8
     qq c93 t 122d.9                ; 8, checkbounds 3: count p2, art
     qq c24 t  c93.9+122d.19        ; 9, no boundcheck: top to Rt, count p2, art
×     [FL mode:]                    ;
     qq c93 t 122d.9+ 63b.19        ; 8, checkbounds 3: count p2, art, ck −10
     qq c24 t  c93.9+122d.19+ 62b.29; 9, no boundcheck: top to Rt, count p2,
>                                   ;                 art, tk 30
     qq c64 t  c24.9+ 69b.19+ c18.29; 10, float: float, top to Rt, nkf 39, stack VinRF
     qq 74d t  c22.9+ c33.19+ 37b.29; 11, formal assign: qq, stack VinUA,
                                    ;                    colon equal, formal assign
d31: qq c33 t  c25.9+ 52b.19+ c67.29; 12, buf assign: colon equal, top to R,
                                    ;                 us 0, after bufassign
     qq c10 t  c16.9+ c41.19+ 82d.29; 13, : mod: clear R, exchange,
                                    ;           no release, ann X
     qq c16 t  81d.9+ 66b.19        ; 14, : : exchange, dln, ar eps LT
     qq c16 t  c41.9+ 83d.19+ 84d.29; 15, mod: exchange, no release,
                                    ;          dln X, sr_LT
d33: qq c41 t  72d.9+ 72d.19+ c21.29; 16, ↓int2: no release, mkf, mkf, stack Vin Rt
d34: qq c41 t  c24.9+ c10.19        ; 17, ↓int 1: no release, top to Rt, clear R
```

```
     [auxiliary table 2]
d4=i-1                                  ; define base of declare words
     qq 30a t 127d.9+ c99.19            ; 18, declare int array, aux 30, gr_M, decl array
     qq 30a t  90d.9+ c99.19            ; 19, declare real array: aux 30, gr_MB, decl array
     qq 30a t  89d.9+ c99.19            ; 20, declare bool array: aux 30, gr_MA, decl array
d38: qq c37 t  c23.9+  0b.19+ 14b.29; 21, proc no param: count p1, stack VinUV,
                                        ;                    0, end call
d39: qq c24 t  c17.9+ c17.19+ c91.29; 22, bounds: top to Rt, srt, srt, end var bound
d40: qq 24a t  70d.9+ c37.19+ 71d.29; 23, end bounds: aux 24, pm, count p1, gm
     qq 70d t  71d.9+ c37.19            ; 24, end bounds: pm, gm, count p1
     qq c94 t  c24.9+ c37.19+ c96.29; 25, first bound: length mult, top to Rt,
                                        ;                   count p1, store length
     qq c95 t  c93.9+ c24.19+ c97.29; 26, first bound: term mult, count p2,
                                        ;                   top to Rt, store term
     qq 28a t  c95.9+ c27.19+ 29a.29; 27, not first bound: aux 28, term mult,
                                        ;                      top to M, aux 29
     qq c94 t  c27.9+ 44a.19+ c96.29; 28, not first bound: length mult, top to M,
                                        ;                      aux 44, store length
     qq 44a t 122d.9+ c97.19            ; 29, not first bound: aux 44, art, store term
     qq 96d t  c93.9+ c37.19+123d.29; 30, end bounds: reserve array, count p2,
                                        ;                  count p1, srt
     qq c16 t  c24.9+106d.19+ c19.29; 31, shift const: exchange, top to Rt, ck,
                                        ;                   stack VinR
     qq c24 t  61b.9+ c19.19            ; 32, shift var: top to Rt, ck(addr), stack VinR
     qq  c8 t  c38.9                    ; 33, param output: out param, inout 1
     qq c41 t 102d.9+ c41.19+ 70d.29; 34, take value: no release, take_formal,
                                        ;                  no release, pm
d8:  qq  71 t   80.9+ 124.19            ; 35, colon equal: gm, grn, grt
d9:  qq  91 t   92.9+ 127.19            ; 36, for assign: gm M, grn M, grt M
d41: qq  c4 t  c23.9+ c41.19+124d.29; 37, Rt expression: release, stack VinUV,
                                        ;                    no release, grt
d42: qq c24 t  c23.9                    ; 38, caseparam expr: top to Rt, stack VinUV
     qq c51 t  c31.9+ c22.19+ c52.29; 39, gier: prep for assign, take controlled,
                                        ;          stack VinUA, address
d44: qq 74d t  58b.9+ 37b.19+ c19.29; 40, move formal: qq, arn UA,
                                        ;                  formal assign, stack VinR
     qq 43a t  42a.9                    ; 41, move↑opands: aux 43, aux 42
     qqc109 t  43a.9+ c42.19+ c13.29; 42, move↑opands: set float, aux 43,
                                        ;                   set UV rel, clear UAUV
     qq c35 t c105.9+ c33.19            ; 43, move↑opands: prep assign, stack UV rel,
                                        ;                   colon equal
     qq 93d t  48b.9                    ; 44, integer mult: mln X IZA, hs X NZA
     qq 44b t  c21.9                    ; 45, select: select2, stack V in Rt
     qq c78 t 108d.9+ 64b.19+ 60b.29; 46, outchar: nextskip on const, outchar const,
                                        ;              int to addr, outchar var
     qq 65b t  95d.9                    ; 47, equiv: ab DX, mbx
```

[Tablewords corresponding to an address operator are f−marked. The a−mark is transferred to RA for use in a few special actions. For further table−description see page 2.]

[input control table 1]

```
d1:  qqf c8 t  c37.9+ c23.19+ c60.29;  0, begin call: outparam, count p1,
                                    ;                 stack VinUV, begin call
     qq  c1 t  c37.9+ 14b.19        ;  1, end call: output, count p1, end call
     qq c38 t  c40.9+ 11b.19+ c47.29;  2, begin proc: inout, input 1, beg proc,
                                    ;                 blockstack
     qq c38 t 109b.9+ c48.19+ 16b.29;  3, end proc: inout 1, ps p, blockunstack,
                                    ;                 endproc
     qq c38 t 109b.9+ c48.19+ 17b.29;  4, end typeproc: inout 1, ps p,
                                    ;                 blockunstack, end typeproc
     qq c40 t  10b.9+ c47.19        ;  5, begin block: input 1, begblock, blockstart
     qq c48 t  18b.9                ;  6, end block: blockunstack, end block
     qq 30b t                       ;  7, goto bypass: goto bypass
     qq110b t                       ;  8, label declar: label declar
     qq c49 t  31b.9                ;  9, while label: get forlabel, bypass label
     qq c14 t  21b.9                ; 10, if: total clear, if
     qq  0b t  32b.9                ; 11, else statement: 0, else
     qq  0b t  33b.9                ; 12, end else statement: 0, end else
     qq c50 t  21b.9                ; 13, for: for, for
     qqfc51 t  c31.9+ 57b.19+ c53.29; 14, :=for: prep for assign, take controlled,
                                    ;                 pm UA, UA to work
     qq c35 t   1a.9                ; 15, simple for: prepare assign, aux 1
     qq c35 t   1a.9+ 26b.19+ 31b.29; 16, simple for and do: prepare assign, aux 1,
                                    ;                 bypass abs, bypasslabel
     qq c35 t  c32.9                ; 17, while: prepare assign, while assign
     qq c43 t  c24.9+ 23b.19        ; 18, whileelement: jump on boolean, top to Rt,
                                    ;                 hop LT
     qq c43 t  c24.9+ 27b.19+ 31b.29; 19, whileelement and do: jump on boolean,
                                    ;                 top to Rt, bypass NT, bypass label
     qq c35 t  c34.9+ c49.19+ 31b.29; 20, step: prep assign, for assign,
                                    ;                 get forlabel, bypass label
     qq c57 t 122d.9+128d.19+ 94d.29; 21, until: until, art, grt VLA, acn MA
     qq  0a t  22b.9                ; 22, stepelement: aux 0, hop NT
     qq  0a t  28b.9+ 31b.19        ; 23, stepelement and do: aux 0,
                                    ;                 bypass LT, bypass label
     qq  c4 t   c9.9+  c4.19+ 34b.29; 24, enddo: release, outopand, release, enddo
     qq c59 t  c16.9                ; 25, first subscript: first subscript, exchange
     qq c46 t  c24.9+122d.19+ c21.29; 26, not first subscr: plus exchange, top to Rt,
                                    ;                 art, stack VinRt
     qq c27 t  44a.9+ c61.19+ c21.29; 27, not last subscr: top to M, aux 44,
<e27 [buffer∧check]                 ;                 not last subscr, stack VinRt
d49: qq c62 t   6a.9+ 50b.19+ c23.29; 28, last subscr: last subscr, aux 6,
×    [FL    ∧check]                 ;                 il 0, stack VinUV
d49: qq c62 t   6a.9+ c20.19+ c10.29; 28, last subscr: last subscr, aux 6,
>                                   ;                 stack buf R, clear R
```

[input control table 2]

```
     qq c63 t  c24.9+ 68b.19+ c19.29; 29, round top: round, top to Rt
                                    ;                 tkf−29, stack VinR
     qq c64 t  c24.9+ 69b.19+ c18.29; 30, float top: float, top to Rt,
                                    ;                 nkf 39, stack VinRF
     qq c16 t  10a.9+ c16.19        ; 31, float next top: exchange, aux 10, exchange
     qq c10 t 126d.9+ c21.19        ; 32, abs: clear R, annt, stack VinRt
     qq c24 t  55b.9+ 68b.19+ c19.29; 33, entier: top to Rt, srf half,
                                    ;             tkf−29, stack VinR
     qq c24 t  65b.9+ c21.19        ; 34, −, : top to Rt, ab 0 DX, stack Vin RT
     qq c65 t  c10.9+125d.19+ c21.29; 35, negative: negative, clear R,
                                    ;               srnt, stack VinRt
     qq  c4 t                       ; 36, proc: release
     qq c24 t  c38.9+ 32b.19        ; 37, else Rt expr: top to Rt, inout 1, else
     qq c26 t  c38.9+ 32b.19        ; 38, else addr expr: top to UA, inout 1, else
     qq c24 t  c38.9+ 33b.19+ c21.29; 39, end else Rt expr: top to Rt, inout 1,
                                    ;                 end else, stack VinRt
     qq c26 t  c38.9+ 33b.19+ c22.29; 40, end else addr expr: top to UA, inout 1,
                                    ;                 end else, stack VinUA
     qq c43 t  c24.9+ 22b.19        ; 41, then: jump on boolean, top to Rt, hop NT
     qq c56                         ; 42, prepare assign: spare assign
     qq c66 t  11a.9                ; 43, := : assign, aux 11
     qq c68 t  98d.9+ c26.19+ 42b.29; 44, goto: goto, goto local,
                                    ;           top to UA, goto computed
     qq c46 t  c70.9+122d.19+ c21.29; 45, + : plus exchange, plus minus,
                                    ;         art, stack VinRt
     qqc16,qq  c70.9+123d.19+ c21.29; 46, − : exchange, plus minus,
                                    ;         srt, stack VinRt
     qq c46 t  c27.9+ 44a.19+ c21.29; 47, integx: plus exchange, top to M,
                                    ;             aux 44, stack VinRt
     qq c46 t  c71.9+ 72d.19+ c21.29; 48, realx: plus exchange, mult div,
                                    ;           mkf, stack VinRt
     qq c16,qq c71.9+ 73d.19+ c21.29; 49, / : exchange, mult div,
                                    ;         dkf, stack VinRt
     qq 13a t  14a.9+ 75d.19+ c21.29; 50, : : aux 13, aux 14, mt, stack VinRt
     qq 13a t  15a.9+ 75d.19+ c21.29; 51, mod: aux 13, aux 15, mt, stack VinRt
     qq c72 t  41a.9+ c23.19+113b.29; 52, ↓int: integ expon, aux 41,
                                    ;           stack VinUV, ↓int
     qq 41a t  c23.9+111b.19+c111.29; 53, ↓real: aux 41, stack VinUV, ↓real, set↓track
     qq c16 t  c75.9+ 22b.19        ; 54, <then: exchange, > , hop NT
     qq c75 t  23b.9                ; 55, <then: >, hop LT
     qq c73 t    0.9+ 24b.19        ; 56, =then: =, next, hop NZ
     qq c16 t  c75.9+ 23b.19        ; 57, >then: >, hop NT
     qq c75 t  22b.9                ; 58, >then: >, hop NT
     qq c73 t    0.9+ 25b.19        ; 59, ǂthen: =, next, hop LZ
     qq c16 t  c75.9+ c19.19        ; 60, < : exchange, >, stack VinR
     qq c16 t  c75.9+ 54b.19+ c19.29; 61, ≤ : exchange, >, sr eps, stack VinR
     qq c73 t  51b.9+ 54b.19+ c19.29; 62, = : =, mt LT, sr eps, stack VinR
     qq c75 t  54b.9+ c19.19        ; 63, ≥ : >, sr eps, stack VinR
     qq c75 t  c19.9                ; 64, > : >, stack VinR
     qq c74 t  49b.9+ c19.19        ; 65, ǂ : ǂ, mt NT, stack VinR
```

[input control table 3]

```
     qq c46 t  c24.9+ 78d.19+ c21.29; 66, ∧ : plus exchange, top to Rt,
                                    ;         mb, stack VinRt
     qq c46 t  c24.9+ 79d.19+ c21.29; 67, ∨ : plus exchange, top to Rt,
                                    ;         ab, stack VinRt
     qq c46 t  c24.9+ 47a.19+ c21.29; 68, = : plus exchange, top to Rt,
                                    ;         aux 47, stack VinRt
     qq c39 t  c79.9                ; 69, label: input 2, opand
     qq c39 t  c79.9+ c39.19+ c82.29; 70, array: input 2, opand, input 2, array
     qq c39,qq c79.9                ; 71, simple: input 2, opand
     qq c39 t  c15.9+ c79.19+ c83.29; 72, formal: input 2, clear byte 2,
                                    ;              opand, formal
     qq c14 t  c39.9+ c79.19+ c80.29; 73, proc: total clear, input 2, opand, proc
     qq c12 t  c39.9+ c79.19+ c81.29; 74, stdproc: clear 0, input 2, opand, stdproc
     qq c39 t  c84.9+ c39.19+ c85.29; 75, constant: input 2, const 1, input 2, const 2
     qq  c1 t  c93.9+ 14b.19+ c22.29; 76, end switc call: output, count p2,
                                    ;                      endcall, stack VinUA
     qq  c4 t   c9.9+  c4.19+ 35b.29; 77, end single do: release, outopand, release,
                                    ;                    end single do
     qqfc86 t  33a.9+  c4.19+ 19b.29; 78, paramcomma: param, oux 33,
                                    ;                 release, call param
     qq c39 t  c88.9+ c40.19+ c89.29; 79, begin bounds: input 2, begin bounds,
                                    ;                   input 1, settype
     qq c90 t  c92.9+ 25a.19+ 26a.29; 80, first bound: bounds, first bound,
                                    ;                 aux 25, aux 26
     qq c90 t  c24.9+124d.19+ 27a.29; 81, not first bound: bounds, top to Rt,
                                    ;                      grt, aux 27
     qq c98 t  c99.9                ; 82, end bounds: end bounds, decl array
     qq 34a t  38b.9+124d.19        ; 83, take real value: aux 34,
                                    ;                      take real value, grt
     qq 34a t  39b.9+124d.19        ; 84, take int value: aux 34,
                                    ;                     take int value, grt
     qq121d t  c38.9+ c38.19+ 40b.29; 85, take array: arnt, inout 1,
                                    ;                 inout 1, move array descr
     qq 31b t                       ; 86, bypass label: bypass label
     qq c40 t c102.9+c100.19        ; 87, end pass: input 1, copy byte 2, end pass
     qq 46a t  c21.9                ; 88, outchar: aux 46, stack VinR
     qq c10 t  46b.9+ c21.19        ; 89, lyn: clear R; lyn, stack VinRt
     qq c10 t  47b.9+ c21.19        ; 90, kbon: clear R, kbon, stack VinRt
     qq c78 t  31a.9+ 64b.19+ 32a.29; 91, shift: nextskip on const, aux 31,
                                    ;            int to addr, aux 32
     qq c24 t  63b.9+ 43b.19+ 45a.29; 92, select: top to Rt, ck−10
                                    ;               select 1, aux 45
     qq c76 t c109.9+ c24.19+ c18.29; 93, real: real, set float, top to Rt,
                                    ;            stack V in RF
     qq c77 t  c25.9+ c19.19        ; 94, int bool: int boolean, top to R,
                                    ;                stack V in R
     qqf39a t  c14.9+ 85d.19+ c21.29; 95, gier: aux 39, total clear, hs, stack VinRt
d20=i−d1                            ;
     qq c55 t 112b.9                ; 96, carret: carret, carret
     qq c14 t                       ; 97, case: total clear
     qq c10 t 125d.9+ 63b.19+ 12b.29; 98, begin case: clear R, srn,
                                    ;                  ck−10, begin case
     qqc103 t  33a.9+  c4.19+ 20b.29; 99, case param: case param, aux 33,
                                    ;                 release, case param
```

[input control table 4]

```
     qq  3b t   0b.9+ 20b.19        ; 100, case;: 3, 0, case param
     qq c38 t  15b.9+ c21.19        ; 101, end case: inout 1, end case,
                                    ;                 stack VinRt
     qq c38 t  15b.9+ c22.19        ; 102, end addr case: inout 1, end case,
                                    ;                      stack VinUA
     qq  0b t  15b.9                ; 103, end statement case: 0, end case
     qq c40 t c101.9+114b.19        ; 104, code: input 1, code, code
     qq c10 t 125d.9+ 63b.19+ 13b.29; 105, begin switch case: clear R, srn, ck 10
                                    ;                          begin switch case
     qq 41b t  c21.9                ; 106, writecr: writecr, stack VinRt
     qq  c9 t  c37.9+ c23.19+ c42.29; 107, std 2 call: outopand, count p1,
                                    ;                  stack VinUV, set UV rel
     qqfc108t  c24.9+ 88d.19+ c21.29; 108, array param: array param, top to Rt,
                                    ;                   ar D, stack VinRt
     qq c35 t c105.9+ c33.19        ; 109, move value: prep assign, stack UV rel,
                                    ;                  colon equal
     qqc107,qq c35.9+c105.19+ c33.29; 110, move address: test formal address,
                                    ;      prep assign, stack UV rel, colon equal
     qq c24 t  62b.9+c105.19+124d.29; 111, move short: top to Rt, tk 30,
                                    ;                  stack UV rel, grt
     qq c24 t                       ; 112, first value: top to Rt
     qqc106 t  87d.9                ; 113, first address: test first formal, arn D
     qq c24 t  62b.9                ; 114, first short: top to Rt, tk 30
     qqc110 t 115b.9                ; 115, new track: appetite, new track
d28=i-d1
     qqfc52 t  c19.9+ c10.19+ c16.29; 116, address: address, stack VinR,
                                    ;      clear R, exchange

d6:  qq                             ; BOTTOM OF OPERAND STACK
a25: grn 1e20 t-1 M                 ; clear operand stack:
     bs (a26) , hv  a25             ;    for i:= core top step -1 until a26 do
     pi  0    , hv  c-1              ;   operand[i]:= shield; goto start pass 7;
a26: pm  d48  DX                    ;ENTRY PASS 7:
     hs  e3                         ;    output (end pass);
     pm  a29  , pi (16e4)           ;    cell 0:= overflow jump;
     gm  0         MA               ;    if -, indexcheck mode then
     pt  d49  t 9a  NQB             ;    begin last subscr word:= no check;
     pa  d29  t 9a  NQB             ;      subscr ass word:= no check;
     pa  d30  t 9a  NQB             ;      subscr par word:= no check;
     ps  a25-1, hv  c40+c           ;    end; input 1; goto clear operand stack;

a29: hh  c87+c, hh  c87+c           ;    overflow jump

a24: qq [pass sum]    ;
e22=k-e14, e47=j      ; set load parameters
b k=e23, i=0          ; load segment word
i=12e21               ;
qq e16.9+1a24.19-e16.19+7.24+a26.39 ;
e                                   ;
e [end pass 7]        ;
s                     ;
```

[Pass 8, segment 1 performs the following tasks:

1. All tracks used for output from pass 7 are rearranged to the lower end of the working area. A table, done[1:30, 0:31| consisting of 30 words, contains 1 in locations corresponding to tracks which have to be moved. Two track buffers each containing 6 tracks are used for track exchange. 6 tracks are read at a time from nearly consecutive tracks on the disc before any output is made, with the intention of minimi−zing disc head movements. The buffer area looks like this:

```
waiting  1,0     1 word
buffer   1,0    40 words
waiting  2,0     1 word
buffer   2,0    40 words
...
waiting  7,0    stopword

waiting  1,1     1 word
buffer   1,1    40 words
...
waiting  7,1    stopword
```

2. The code for ↓ which is part of pass 8.1 is outputted into the final programarea. But only if the sign of the first inputbyte i −.

3. Standard procedure tracks described in the tracklist (pass 8 output) are transferred to the final programarea.

State of e4 parameters:

Initially:	Finally:
1e4: used tracks	changed to drummode before task 1
4e4: available	updated after task 1
8e4: inputtrack	updated before task 1
9e4: output track	updated after task 1
10e4: increment	changed to drummode (1) before task 1
13e4: running track	running track − ↓ length

Testprint from 8.1 consists of every tracknumber selected.

Drumpicture after pass 8:

```
|pass 7 output|  |RS|algol progr|↓code|std procs used|long strings|
 ^                                                              ^
first track                                                track base
```

8.1 core picture:

```
|GPA|↓code|tracksort|move↓|move std|init done|          |done|
                                  |track buffers  |
```

]

```
b k=e22+e14, i=e16-e47    ;     pass 8.1 drumblock
d i=e16                   ;
                          ;
b a9, b4                  ;     comment entry a↑x, x real;
                          ;       a↑x = 2↑(x×log2(a)); first log2(a) is computed;
    arnfc17-3 X           ;    RF:=a;
    ga  ra5   , tl  48    ;    exp2:= exponent(a); z:=R:= mantissa(a)/4;
    hv  ra4           LZ  ;    if z=0 then goto exit;
    hh  c64           LT  ;    if z<0 then alarm(≠<spill≠);
                          ;
    gr  c50   , gr  c51   ;    comment the method used is that described in
    arn ra6   , ac  c50   ;    C. Hastings: Approximations for Digital
    sr  c51   , dk  c50   ;    Computers, page 166;
    gr  c50   X           ;    z:= (sqrt(2)/4-z)/(sqrt(2)/4+z);
    mkn c50   , gr  c51   ;    z2:=z↑2;
                          ;
    it  ra6   , pan r1    ;    R:=0;
    ar [b(i)] X   t 1     ;    for y:=b[7],b[5],b[3] do
    grn c17   V       LA  ;    R:=(R+y)×z2;
    mkn c51   , hv  r-2   ;    log2:=((R+b[1])×z+exp2+0.5);
    mkn c50   , ar  ra5   ;    UV:=0;
    nkf 11    , mkf c17-4 ;    log2:=log2×2↑9×x;
                          ;
    gm  c17-1 , pm  c17   ;    if exponent(log2)≥8 then
    bs (c17-1)    t 9  NZ ;    alarm(≠<spill≠);
    hh  c64           NZ  ;
                          ;    comment the following Code computes 2↑log2 using
    tl (c17-1), ga  ra3   ;    the Method described in G.N. Lance: Numerical
    tl  10    , cl -2     ;    Methods for high speed Computers, page 32ff;
    sr  128 D X           ;    exp:=entier(log2); M:=(log2-exp-0.5)/2;
    gm  c17   , mkn c17   ;    UV:=M; M:=UV↑2;
    pm  384 D X           ;    R:=0.75;
    mk  ra8   , gr  c51   ;    work1:= R+b1×M;
    gr  c52   , arn ra7   ;    work2:=work1;
    mk  ra9   X           ;    M:= c1×UV↑2+a1;
    mkn c17   , sc  c52   ;    work2:= work2-M×UV; R:= M×UV;
    ar  c51   X           ;    M:= R+work1;
    mln ra6   , dl  c52   ;    R:=M/work2/sqrt(2);
a3: nkf[exp]      t 2     ;    RF:=R×2↑(exp+1);
a4: grf c17   , hh  c5    ; exit: UV:=RF; goto exit std proc;
                          ;
a5: qq [exp2]     t 512   ;
a6: 181 /  19/ 819/ 254   ;    sqrt(2)/4  (= 0.3535 5339 059)
    1023/ 579/ 325/ 663   ;    b[7]       (=-0.4342 5975 129×2↑(-9))
    1023/ 433/ 591/ 509   ;    b[5]       (=-0.5765 8434 206×2↑(-9))
    1023/  39/ 118/ 823   ;    b[3]       (=-0.9618 0076 229×2↑(-9))
    1021/ 117/ 369/ 224a  ;    b[1]       (=-2.8853 9007 274×2↑(-9))
                          ;
a7: 266 / 172/ 574/ 725   ;    a1         (= 0.5198 6038 444)
a8: 73  / 797/ 660/  37   ;    b1         (= 0.1440 9951 127)
a9: 8   / 524/ 902/ 710   ;    c1         (= 0.0166 2613 208)
```

```
                              ; comment entry a↑n, n integer;
    pm   c53   , gm   c50     ;     result:=1;
    arn  c17-4           ITA  ;     TA:=n<0;
    ann  c17-4                ;     R:=abs n;
b1: hv   rb2   X          NZ  ; next bit: if n=0 then
    arnf c50   V         NTA  ;             begin RF:= if -,TA then result
    arnf c53   , dkf  c50     ;                         else 1/result;
    grf  c17   , hh   c5      ; exit:        UV:=RF; goto exit std proc
                              ;             end;
b2: cln -1     , gm   c52     ;     n:= (n shift -1)∧(1 0 39 -1);
    hh   rb3              LZ  ;     if last bit of n≠0 then
    arnf c17-3 , mkf  c50     ;      result:=result×a;
b3: grf  c50   , arnf c17-3   ;
    mkf  c17-3 , grf  c17-3   ;     a:=a↑2;
    arn  c52   , hv   rb1     ;     goto next bit;
```

e

```
b a12, b10, c11, d5        ; Pass 8.1 core block

a: qq [testtrack]          ; initialised to: (if discmode then inputtrack−used+2
[1a] qq [↑ tracks] 2       ;
[2a] qq [save track, track];

c2:  gr  2a    , sr  5e4   ; procedure clear track (R);
     ck  −15   , ga  a3    ; begin save track:= R;
     tk  10    , ck  −5    ;
     ga  a4    , it  d3    ;   if done[R−first track] then
a3:  arn 0     , ck (a4)   ;   return;
     hr  s1            NO ;
     qq (b1)   t   −1      ;   inputlack:= inputlack − 1;
     ar  512   D           ;
a4:  ns  0     , ck  s     ;   done[R − first track]:= true;
     gr (a3)   , arn 2a    ;   R:= if discmode then
     sr  9e4   V       NQA ;   (save track − first track) : 2
     sr  5e4   , tl  −1    ;   else save track − outputtrack
     ar  4e4           LT  ;   + (if save track < output track then available
     ar  5e4               ;   else 0);
b2:  gr  p0    Xt  41  MA  ;   store final:= store final + 1;
     [store final]         ;   final place[store final, 1−out]:= R + firsttrack;
     pm  2a    , gm  a     ;   waiting [store final, 1−out]:= true;
     hs  c11   X           ;   test track:= save track;
     is (b2)   , lk  s1    ;   from drum (save track,
     hr  s1                ;   buffer [40 × store final, 1 − out]);
                           ; end clear track;
     [TRACK SORT]
c1:  pp  d     , gp  b4    ; for out:= 0, 1−out while outputlack > 0 do
b:   can 0     , hv  c3    ; begin i:= j:= 0;
     [outputlack]          ;
     gp  b3    , ns  p     ;
     pp  sd1   , pa  b2    ;   store final:= 0;

b4:  arn 0     t   41  IPA ;   for i:= i+1 while waiting [i, out] do
     hs  c2            LPA ;   clear track (final place[i, out]);
     hv  r−2           LPA ;

b1:  can 0     , hv  a2    ; CLEAR NEXT: if inputlack > 0 then
     [input lack]          ;   begin
     arn a     , sr  6e4   ;   if testtrack ≥ last track then
     arn 5e4   V       NT  ;   R:= first track else
     arn a     , ar  10e4  ;   R:= testtrack + 1;
     ps  b1−1  , is (b2)   ;   if store final < top then
     it  s512  , bs  pd2   ;      begin test track:= R; clear track(R);
     gr  a     , hv  c2    ;      goto clear next end;
                           ;   end inputlack > 0;
a2:  grn(b2)   Vt  41  M   ;   store final:= store final + 1;
a1:  is (b3)   , sk  s1    ;   waiting [store final, 1−out]:= false;
b3:  arn 0     t   41      ;   for j:= j + 1 while waiting [j, out] do
     hh  c1            NA  ;   begin outputlack:= outputlack − 1;
     qq (b)    t   −1      ;         to drum (final place[i, out],
     ps  a1−1  , hv  c11   ;                  buffer[40×j, out]);
                           ; end j loop end out loop;
```

```
     [Adjust e4 parameters]
c3:  arn 10e4  , ac  4e4   ; available:= available + 1;
     ar  6e4   , gr  a     ; track:= last track + 1;
     ar  10e4  , gr  9e4   ; output track:= last track + 2;
     arn(e1)   t  1        ; std tracks:= abs( input);
     hs  e2           LA   ; comment std tracks = number of std tracks + 1;
     pa  1a    V      NT   ; if −, expon track then ↑length:= 0;
     mt  −1    D           ;
     ga  b5    , ar  1a    ; track:= track − std tracks − ↑length;
     ck  10    , sc  a     ; available:= available − std tracks − ↑length;
     sc  4e4   , sc  9e4   ; output track:= output track − std tracks − ↑length;
     nt (1a)   , qq (13e4) ; running track:= running track − ↑length;
     arn 4e4   , sr  1e4   ; if used tracks > available then
     hs  e5           LT   ;   mess(≠<program too big≠, 2,0);
     hv  r1    , qqn e33   ;

     [Output ↑tracks]
c4:  btn(1a)   t  −1       ;   for ↑length:= ↑length step − 1 until 1 do
     arn a     , ar  10e4  ;     begin
     hh  a12          LZ   ;     track:= track + 1;
     gr  a     , hs  c11   ;     to drum(track, ↑code[i]);
     sk  e16−40  t  40     ;     i:= i + 40
                           ;     end;
a12h:hv  c4    , grn  2a   ; Move std procs: proc sum:=n:=0;
     qq  r2    , arn  a8   ;   segment driver to e16ff;
     ps  r     , hh  e29   ;   std proc words:=M;
     gm  1a    V           ;
a6:  qq (e1)   t  −1       ;
a7:  arn 1a    , sr  a11   ; get procs: R:=if std proc words<480 then
a8:  qqn e16[see 1a12] NT  ;   std proc words else 480;
     ar  a11   , sc  1a    ;   std proc words:=std proc words−R;
     hv  a10          LZ   ;   if R=0 then goto finis;
     ar  d4    DX          ;
     vk  0     , hs  e16   ;   wait drum; get segment(R) words to:(std buf[0]);
     ac  2a    , pp  −40   ;   proc sum:=proc sum+R; p:=−40;
     hs  e5           NZA  ;   if −,transport ok then
     hv  r1    , qqn e46   ;     mess(≠<comp.medium≠,2,0);
b5:  can 0[std.tracks],hva7; next std track:
     pmn(e1)   X   t1      ;   if std tracks=0 then goto get procs;
     hs  e2           LA   ;   Raddr:=next byte;
a9:  bs p−439  , hv  a6    ; skip: if p>439 then
     pp  p40   , it  1     ;   begin reset input; goto get procs end;
     nc 0 [n]  , hv  a9    ;   n:=n+1; if Raddr≠n then goto skip;
     arn a     , ar  10e4  ;   track:=track+1;
     gr  a     , hs  c11   ;   to drum(track, std buf[p]);
     sk  pd4               ;   std tracks:=std tracks−1;
     hv (b5)   D  t−1      ;   goto next std track;
a10: arn 2a    ,           ; finis: if proc sum≠0 then
     hs  e5           NZ   ;          mess(≠<comp.sum≠,2,0);
     hhn e29   , qqn e45   ;   goto next segment;
a11: qq  480.19            ;   constant;

c11: hv  e11          NKB  ; procedure select track (R);
     gr  e13   , gm  41e13 ;   if NKB then goto track else
     bt  474   t  −55      ;     begin lines:= lines + 1;
     hs  e9                ;     if lines = 10 then
     bs (r−2)  , pa  r−2   ;       begin new line; lines:= 0 end;
     hs  e8    X           ;     save R and M; print 1; goto track
     hv  e11               ;     end select track;
```

```
[23.10.1967]
[GIER Algol 4, pass 8.1, page 6]

d=i−40, d1=d+d+247, d4=i   ; define base of buffer 1 and buffer 2 and std buf
d2=758, d3=e20−29          ; d2:= 512 + bufferlength, d3:= base of done

     [START OF BUFFER AREA:]
c5:  grn d3    t   −1  M   ; CLEAR BUFFERS:
     bs (r2)   t   −1      ;   for i:= base of done − 1 step −1 until 3c5 do
     hv  c1                ;   marks[i]:= 0;
a5:  qq  −1    , hv  c5    ;
[1a5]qq  1.39              ;

     [ENTRY TO 8.1, clear done]
c6:  pi (16e4) , ps  30    ;   QA:= discmode;
     ps  s−1   , grn sd3   ;   for i:= 29 step −1 until 0 do done[i]:= 0;
     bs  s     , hv  r−1   ;

     arn 9e4   , sr  1a5   ;   if −, discmode then track:= test track
     hv  r4            NQA ;   := outputtrack − 1 else
     sr  5e4   , tl  −1    ;   begin used tracks:= outputtrack − 1
     gr  1e4   , tk  1     ;         − first track;
     sr  8e4   , mt  a5    ;   track:= testtrack:=
[r4] gr  a     , gr  2a    ;   inputtrack − 2 × used tracks
     arn 1e4   , ck  −10   ;   end;
                           ;   inputlack:= outputlack:= used tracks;
     ga  b     , ga  b1    ;
     ga  c7                ;
                           ; INITIALISE DONE:
c7:  can 0     , hv  c8    ;   for i:= 1 step 1 until used tracks do
     arn 6e4   , sr  2a    ;   begin
     sr  10e4              ;     track:= if last track < track + increm
     arn 5e4   V       LT  ;        then first track
     arn 2a    , ar  10e4  ;        else track + increm;
     gr  2a    , sr  5e4   ;
     ck  −15   , ga  b6    ;
     tk  10    , ck  −5    ;     done [track − first track]:= false;
     ga  b7                ;
     arn 512   D           ;     comment true is 0, false is 1;
b7:  ns  0     , ck  s     ;
b6:  ac  0     t   d3      ;
     hv (c7)   Dt   −1     ;     end;

c8:  arn 10e4  , tl  −1    ;   if discmode then increm:= increm : 2;
     gr  10e4          LQA ;
     arn 5e4   , ar  1e4   ;   inputtrack:= first track + used tracks − 1;
     sr  10e4  , gr  8e4   ;
     pm  b9    , gm  e35   ;   set −; discmode word in trackin;
     hv  c5                ;   goto clear buffers;

b9:  srn 10e4  , ac  1e4   ;   −, discmodeword in trackin;

d5:  qq [pass sum]         ;

d e22=k−e14, e47=j         ;   set load parameters
b k=e23, i=0               ;   load segment word 13
i=13e21
qq e16.9+1d5.19−e16.19+8.24+3.29+c6.39   ;
e                                        ;

e  [pass 8.1 core block]
e  [pass 8.1 drum block]
s
```

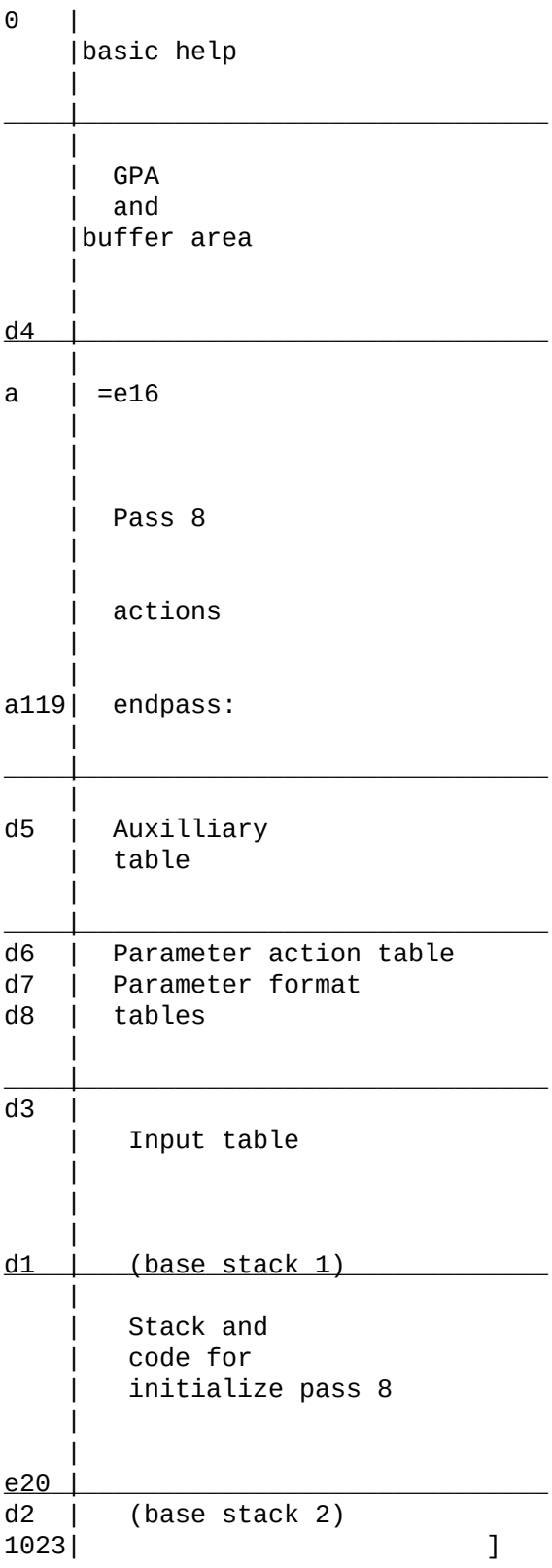
```
bk=e22+e14, i=e16−e47, a131, b50, d70  ;
i=e16                                  ;

  d27=40                               ; define tracklength

d0=−d27−d27−d27−d27+3                ; maxapp:= −(4×tracklength − 3);
d31=c4−c26                           ;
d54=163e13                           ; last word in outbuf:
d55=d54−d27                          ; first word in outbuf − 1:
d56= −d55−1                          ; relative base (used in b8):

b1: qq                               ; instr:
b3: ps (),                           ; ps instr:
b8: qq −1.9+d56.19+1.29−1.39,        ; base:
b12: qq                              ; const: [also used as work]
b13: qq [case count], hs c15         ; used in action a79
b26: qq ,                            ; work   [only address must be used]
b43: qq                              ; word   [working loc in a22]
```

[Memory layout:



```
     [address operand:]

a:   hs  a14   , ps  a21   ;   input lit; set return to (next);

     [add byte and store:]

a1:  hs  a20               ;   R:= nextbyte;

     [add R and store:]

a2:  ac  b1    , hv  a3    ;   instr:= instr + R; goto store;

     [check and store]

a127:bs  p5                ;   if t > 5
     hs  a11               ;     then changetrack

     [store:]

a3:  arn b1            IPC ;   R:= instr; PC:= markC;
     hv  a4            NTB ;   if −, newhalf then goto store R;
     hv  a87           NRA ;   if −, oldhalf then goto store half in next;
     gi  b26   , arn b26   ;   extract PC and old C;
     mb  60    D       IPA ;   PA:= 1;
     ca  16    , hv  a86   ;   if marks are not compatible
     ca  4     , hv  a86   ;     then goto not compatible;
     pi  16    t −17   LQB ;   if old B = 1 then PB:= 1;
     arn(b2)   , ck  −10   ;   R:= word [w] shift −10;
     pi  40    t   −43     ;   PA:= old A:= 1; oldhalf:= false;
     ar  b1    , hv  a109  ;   R:= R + instr; goto store in word;

     [not compatible:]

a86: hs  a48           LRA ;   fill up in left half;
     arn b1                ;   R:= instr; goto store half in next;

     [store half in next:]

                           ;   oldhalf:= true; PA:= 0;
a87: pi  2     t  −35  IQC ;   old C:= mark C;
     it  −1                ;   w:= w − 1;

     [store in word:]

a109:
b2:  gr [w] d54        MPC ;   word [w] mark PC:= R;
     pp  p2    , hr  s1    ;   t:= t + 2; return;

     [store R]

a4:  hs  a48           LRA ;   if oldhalf then fill up in lefthalf;
     gr (b2)   t −1    MPC ;   w:= w − 1; word[w] markPC:= R;
     arn(b2)           IQC ;   QC:= marks[word[w]];
     pp  p4    , hr  s1    ;   t:= t + 4; return;
```

```
     [s reloperand]

a5:  hs  a16              ;   R:= pinstr; input staddr; comment address saved in M;
     ga  b4    , ck  10   ;   new S:= part 1(R); part 1(instr):= part 2(R);
     ga  b1    , arn b3   ;   if news ǂ address of (ps instr)
     nc (b4)              ;     then store ps;
     hs  a6               ;
     qq (b5)   V   3   NTB ;   app:= app + (if newhalf then 1
     qq (b5)   t   1       ;                              else 3);
     hs  a10              ;   maybe change track;
b4:  it[news]   , pa b3   ;   psinstr:= new s;
     pp  p1           NRB ;   if −, sets then t:= t + 1;
     pm  r                ;
     hs  a3           IRB ;   sets:= false; store;
     pmf r            IRB ;   sets:= true;
     hr  s1               ;   goto next;

     [store ps:]

a6:  hr  s1    X      NRB ;   if −, sets then return;
     pi  8     V  −12 LRA ;   if oldhalf then old A:= 1 else olda:= oldB:= 0;
     pin 2     V  −16     ;   sets:= false; oldhalf:= −, oldhalf;
     arn(b2)   , ck  −10  ;   R:= (if oldhalf then 0 else instr shift −10)
     ar  b3    , pp  p1   ;     + psinstr; t:= t + 1;
     it  −1           LRA ;   if oldhalf then w:= w − 1;
     gr (b2)   X      MQC ;   word[w] markQC:= R; R:=M;
     hr  s1           IQA ;   old A:= 1; return;

     [<p rel operand> take forlabel]

a7:  arn d11   , hs  a34   ;   R:= hs c2; add stack 2;
     ps (b7)   , ud  a112  ;   s:= top 2; const markPC:= R; R:= stack[s−1] −
     arn s−1   , sr  s1    ;     stack[s+1]; if part 4(R) ǂ 0 then
     ck  −10   , nc  0     ;     begin comment enddotrack ǂ forlabeltrack;
     arn d47   , hv  a115  ;   R:= table[gmMA]; goto not local end;
     tk  20    , ud  a85   ;   const:= R shift 20 + (if PB = 1 then
     ar  d34   , gr  b12   ;     hh−hv else 0) + hvs;
     can(s1)   , hh  a116  ;   if part 1 (stack[s+1]) = 0 then goto local;
     arn(s1)   , sr  d35   ; localsingle: R:= word[part 1(stack[s+1])] − hsr
     tk  10    , ck  −10   ;   part 1 (R):= 0;
     ar  b12   , gr (s1)   ;   word[part 1(stack[s+1])]:= R + const;
                           ;   comment the previous stored hs p<work> is
                           ;     changed to local
     ps  a21   , hv  a18   ;   set return to (next); go to skip
                           ;                                 two bytes;
a115:ar  d10           LPB ; not local: if PB = 1 then R:= table[gm MC];
a116h:hv  r1   , arn d36   ;   goto store gm; local: R:= table[gm];
     hs  a23   , qq  a21   ; store gm: simulate input; set return to next;
     arn d37   , hv  a23   ;   R:= table[pm]; goto simulate input;
```

```
     [<p rel operand> end do / end single do]

a8:  hs  a30               ;   stack point 2;
b42: bs p[do app] 5, hsa11 ;   if doapp + t > 0 then change track;
     pa  b42   t   5       ;   doapp:= 5;
     hs  a15               ;   store with operand
     hs  a30               ;   store point 2;
     it (b2)          LOA  ;   part 1 (stack[top 2]):= if byte = enddo then 0
     pa (b7)   , hv  a22   ;     else w; goto next

     [<do app> new track]

a9:  hs  a120  , ac  b42   ;   doapp:= nextbyte
a105:hv  a22               ;   goto next

     [maybe change track:]

a10:b5:it p [app], bs 1    ;   if t + app < 0 then
     hr  s1                ;   return

     [change track:]

a11: hs  a6            LRB ;   if sets then store ps;
     hs  a26               ;   point;
     tl  −10   , ar  b37   ;   pack(R, 30, 39, linecount);
     mb  d14   , ck  10    ;   M:= R +
     ar  d12   VX      LOB ;     (if changemode = 1 then hs c3
     ar  d29   X           ;     else hs c1);
     sy  23            LKB ;   testprint(X);
     hs  e25               ;   trackout;

b18: srn 41    D           ;   length:= −length; R:= length shift −10;
     ga  b18   , tk  −10   ;   constw0:= constw0 + R;
     ac  b32   , ac  b33   ;   w0:= w0 + R
     ac  b19   , ar  d10   ;   const 0:= const 0 + R;
b33: pa  b10   t d55       ;   wconst:= constw0; R:= R + 40 1;

     sc  b8                ;   pointbase:= pointbase − R;
b32: pa  b2    t d54       ;   w:= w0;
     gm (b2)           MPC ;   word[w] markPC:= M;
     pi  8     t   −16     ;   oldhalf:= sets:= false; old A:= 1;
     pp [tmax]d0, hr  s1   ;   oldB:= 0; t:= t max;
```

```
     [constantoperand]

a12: it  s     , pt  r1    ;   input lit;
     hs  a14   , ps [s]    ;   set return;
     sr  d10   , pm  d19   ;   if const = 1 then
     hv  a2    X       LZ  ;     begin R:= −rbit; goto addR and store end;

     [look up constant]

a13: arn b12           IPC ;   PC:= marks(const); comment execute from a47, a129;
b19: pa  b11   t d55       ;   constaddr:= const 0;
a89h:gi  b15   , arn b12   ;   PCconst:= PC;
b11: sr[constaddr]t 1  IPC ; test next lit: constaddr:= constaddr + 1;
     it (b10)  , bs (b11)  ;   PC:= marks(constaddr); if constaddr > w const
b15: qq[PC−const], hh a92  ;     then goto constant not on track;
     hh  a89           NZ  ;   if constant ≠ word[constaddr]
     gi  r1    , arn b15   ;     ∨ PC const ≠ PC
     nc[indik] , hh  a89   ;       then goto test next lit;
a91: hs  a3                ; constant on track: store;
     arn b11   , sr  b2    ;   part 1(instr):= constaddr − w;
     ga (b2)   , hr  s1    ;   return;

     [input lit]

a14: hsn a17   X           ;   M:= 0;
     cl  −30           IPC ;   const markPC:= input 4;
     gr  b12           MPC ;   right return

     [constant not on track:]
a92h:hh  s     , it (b5)   ;   if t + app + 4 > 0
     bs  −4    , hh  a129  ;     then
     hs  a11               ;     begin change track; goto look up constant end;
a129:hv  a13   , ud  a13   ;   wconst:= wconst + 1;
b10: gr  d55   t   1   MPC ;   word[wconst] markPC:= const;
     pp  p4    , hv  a91   ;   t:= t + 4; go to constant on track;
```

```
     [store with operand]

a15: hs  a20               ;   nextbyte;
     pm (b)    X   d3      ;   R:= operandtable [byte];
     gt  r1    , mb  d14   ;   opaction:= 0; part 2(R):= 0;
     ac  b1  , hv[opaction];   instr:= R; goto (opaction);
                           ;

     [input st addr]

a16: hs  a18               ;   input 2;
     cl  -10   , ar  d20   ;   longshift - 10; R:= R + displbase;
     hr  s1                ;   return; comment part 1 = displ.ref,
                           ;                   part 2 = rel addr;

     [input 4]

a17: hs  a20               ;   nextbyte;
     hs  a19               ;   shiftnextbyte;

     [input 2]  [comment return with M30-39=firstbyte and R0-9=secondbyte]

a18: hs  a19               ;   shiftnextbyte;
a19: cl  10                ; shiftnextbyte: longshift 10;

     [nextbyte]

a20: arn(e1)   t   1       ;   R:= input;
     hs  e2            LA  ;
     hs  e7            LKB ;   if test then print the byte;
     arn(e1)           LKB ;
     ga  b     , hr  s1    ;   byte:= R; return;

     [nextbyte and right return]

a120:hs  a20               ;   nextbyte;
     hh  s                 ;   rightreturn;
```

```
     [carret]

a110:arn d10   , sc  e4    ;   CR count:= CR count − 1;
b37: qq [linecount] 0V −1  ;   linecount:= linecount − 1; goto next;

     [release for/if]

a21: qq (b7)   t   3       ;   top2:= top2 + 3;

     [next:]

a22: hs  e9            LKB ;   if testmode then reset print;
     hs  a20   , ps  a21   ;   nextbyte; set return to (next);
     hv  a98           NT  ;   if byte > 511 then
     sr  d40   D       ITA ;     begin byte:= byte − 512; R:= table[byte]
     pm (b)    XV  d41     ;       + (if byte > may be f then fmask else 0)
b:a98:pm0[byte]XV  d3      ;     end
     ar  16    DV      NTA ;   else R:= table [byte];

     [simulate input:]

a23: hv  a24           LC  ;   if mark C then goto fullword;
     ga  r2            IPC ;   PC:= mark C;
     gr  b43           MPC ;   const mark PC:= R; insert store parameters
     pi [new instr] t 15   ;   in (changemode, destroys, newhalf, PC);
     gr  b1            MPC ;   instr mark PC:= R;
     mb  d69   , ga  b5    ;   app:= bits (6, 9, R);
     hs  a6            LTA ;   if destroys then store ps;
     hs  a10               ;   maybechange track;
     pm  b43   X       IPC ;   R:= const; PC:= marks [const];
     ck  10    , ga  b14   ;   action:= part 2 (R);
     pa  b1    , pt  b1    ;   part 1 (instr):= part 2 (instr):= 0;
     hh  a104          NPC ;   if PC = 0 then goto test action;
     arn b1    , ck  −10   ;   work:= bits (30, 39, instr);
     ga  b26               ;   if PA = 0 then instr:= auxtable [work]
     pmn(b26)  XV      NPA ;     else begin part 1 (instr):= work;
     tk  10    ,  ar  b26  ;                 part 4 (instr):= 0 end;
a104h:gr b1    , arn b14   ; test action:
b14: is [action], bs s+d4  ;   if action < base tables then
     pm (b)    , hv (b14)  ;     begin M:= table [byte]; goto (action) end;
     ga  b     , hs  a3    ;   byte:= action; store;
d69: qq  15    , arn(b)    ;   R:= word [action];
a103:ps  a21   , hv  a23   ;   set return to (next); goto simulate input;
                           ; fullword:
a24: pm (b)    t   1       ;   changemode:= 0; destroys:= newhalf:= false
     pi  0     t  63   IPC ;   PC:= marks [table[byte + 1]];
a100:gr  b1            MPC ;   instr mark PC:= R;
     hv  a127              ;   goto check and store;
```

```
     [st.proc. simple]

a25: hs  a39   , qq  a21   ; take st. proc: comment rightreturn s2;
                           ;   set return to (next); goto addR and store;

     [↓real/↓integer]

a101:pm (b)    X  d33      ;   R:= programpointword[byte];
     hv  a2    , hr  a2    ;   goto add R and store;

     [point:]

a26: arn(b2)   D           ;   R:= base + w shift -10;
     ck  -10   , ar  b8    ;   comment trackrel.19 + trackno.39
     ar  d16           LRA ;   PB:= oldhalf;
     hr  s1            IPC ;   PA:= 1; return;

     [testpoint:]

a27: sr  b8    , ck  -10   ;   if part 4 (R) ≠ current track then
     ca  0     , hv  a88   ;     begin
     ck  10    , ar  b8    ;     R:= R + hs c2, ; return(s-1)
     ar  d11   , hr  s-1   ;     end
a88: tk  20    , ar  d17   ;   else R:= (R-base) shift 10 + (if PB = 1 then hhr
a85: ar  d18           LPB ;     else hvr) - inpart 1(w); comment executed from a7;
     sr (b2)   D           ;   comment hv/hh r<addressdifferens on track>;
     hr  s1                ;   return;

     [stack-actions]

                           ; stack point 1:
a28: hs  a26               ;   point
                           ; stack R1:
b6:a29:gr[top1] d1 t 1 MPC ;   top 1:= top 1 + 1; stack [top 1] markPC:= R;
     hv  a32               ;   goto test stack;
                           ; stack point 2:
a30: hs  a26               ;   point;
                           ; stack R2:
b7:a31:gr[top2] d2 t -1 MPC;   top2:= top 2 - 1; stack[top 2] markPC:= R;
a32: it (b6)   , bs (b7)   ; test stack: if top 2 > top 1 then return;
     hr  s1    , qq  e34   ;   alarm (≠<stackoverflow≠);
     ps  r-2   , hv  e5    ;

     [unstack-actions]

                           ; take stack 1:
a33: arn(b6)   V       IPC ;   R:= set PC(stack[top1]); goto count 1;
                           ; add stack 2:
a34: ar (b7)   V       IPC ;   R:= R + set PC(stack[top2]); goto count 2;
     qq (b6)   V   -1      ; count 1: top 1:= top 1 - 1; return;
     qq (b7)   t    1      ; count 2: top 2:= top 2 + 1; return;
     hr  s1                ;   comment hrs1 used in a35;

     [prepare goto:]

a35: hs  a27           IPC ;   PC:= mark C; testpoint; comment return s-1;
     ar  1     D       NRA ;   if not on same track;
     hr  s1                ;   if -, oldhalf then R:= R + input 1 (1);
                           ;   return;
```

```
     [do abs]

a36: is (b7)   , arn s1    ;   R:= stack [top 2 + 1]
     hv  a38               ;   goto generate goto;

     [goto bypasslabel]

a37: hsn a34   , ps  a21   ;   take stack 2; set return to (next)
a38: hs  a35               ; generate goto: prepare goto
     pi  64    V 911   NA  ;   if mark A then goto store R
a90: hv  a4                ;   newhalf:= true; PC:= 0;
     gr  b1    , hv  a3    ;   instr:= R; goto store;

     [take st.proc:]

a39: hs  na18              ;   M:= 0; input 2; comment part 1 (R) = trackno,
     ck  −10   , cl  −20   ;   part 4(M) = trackrel; shift to progr.point in R;
     ar  d28   , hh  s2    ;   R:= R + hs c 26, ; right return 2;

     [condgoto: hv LT/NT/LZ/NZ]

a40: arn(b7)   , ps  a21   ;   R:= stack [top 2]
     hs  a48           LRA ;   if oldhalf then fill up in lefthalf;
     hs  a35               ;   prepare goto;
a112:gr  b12           MPC ;   const markPC:= R; comment execute from a14 and a7;
     arn d17   V       LA  ;   if −, mark a then
     ac  b1    , hv  a3    ;     begin instr:= instr + R; goto store end;
     ac  b1    , hv  a13   ;   instr:= instr + hv r ; goto look up constant

     [bypass LT/NT]  [112]

a41: ps  a40   , hh  a42   ;   set return to (condgoto); goto plus 2;

     [bypass abs]  [114]

a42: ps  a37   , is (b7)   ;   set return to (goto bypass label);
     arn s2    , hh  s     ; plus 2: R:= stack[top 2 + 2]; right return;

     [<trackno><trackrel><proctype> st.proc.param]

a43: hs  a120  , ga  b17   ;   nextbyte; type:= byte;
     hs  a39   , qq  0     ;   take st proc; s:= false;
     sr  d28   , hh  a107  ;   R:= R − hs c26 ; goto add format;

     [<elsetype> endelse]

a44: hs  a30   , hr  r−1   ;   stack point 2; comment rightword used by a43;
     hs  a120  , hs  a31   ;   nextbyte; stack R2;
     ps  a21   , hv  a30   ;   set return to (next); goto stack point 2;
```

```
     [<thentype> else]

a45: hs  a120  , hs  a26   ;   nextbyte;
     gr (b7)          MPC ;   stack[top 2] markPC:= point;
     ps (b7)   , arn s2    ;   s:= top 2; R:= stack[s + 2];
     hs  a38               ;   generate goto;
a113:arn s1    , nc (b)    ; checktype: if byte ≠ stack [s+1]
     arn d13   , hs  a23   ;     then simulate input (nkf 39);
     hv  a22               ;   goto next

     [<number of indices><array rel> move array]

a46: bs  p22   , hs  a11   ;   if t > −22 then change track;
     pa  r2    t   d5      ;   index:= simulatebase; store;
     hs  a3    , qq  i     ; simulateloop: set return to (simulatelook);
     arn[d5]   t   2       ;   index:= index + 2;
     it (r−1)  , pa  b     ;   byte:= index;
     hv  a23          LT   ;   R:= simulatetable [index]; if R < 0
     arn d21   , hv  a103  ;     then goto simulate input;
                           ;   R:= ck−10; goto set next and simulate;

     [<appetite><−blockno> endblock / endproc / end typeproc ]

a47: hs  a49   , hs  a16   ;   stack paraminf; input st addr;
     pa  b24   , ck  10    ;   pointcount:= 0; endinf:= part 2 (R);
     ga  b21   , tk  20    ;   comment appetite; part 2 (R):= part 1 (R);
     ar  d45          IPC  ;   part 1(R):= 0; R:= R + hvc11, hh
     hs  a29   , qq  a21   ;   stack R1; set return to (next);
     arn b1                ;   R:= instr; if part 1(R) = exit block
     ca  c9    , hv  a86   ;     then goto not compatible;
     ca  c21   , hsn a99   ;   if part 1(R) = exit func then
     ps  a21   , hv  a3    ;     begin R:= 0; count point end; goto store;

     [formal assign]

a131:hr  s1           NRA  ;   if −, old half then return;

     [fill up in lefthalf:]

a48: hv  a6    X      LRB  ;   if set s then
a130:pi  0     t   −3      ;     save R and store ps else t:= t + 2;
     pp  p2    , hr  s1    ;   oldhalf:= false; return;

     [stack param inf:]

a49: arn b21   , cl  10    ;   pack (R, 0, 9, endinf,
     arn b22   , cl  10    ;         10, 19, paramcount,
     arn b23   , cl  10    ;         20, 29, pwork1,
     arn b24   , cl  −30   ;         30, 39, pointcount,
     bs (b25)  , pm  r1    ;         40, 40, if inexpr then 0 else 1);
     hs  a31          IPA  ;   stack R2;
     pa  b22   , hh  s     ;   paramcount:= 0; rightreturn;
```

```
     [test expr:]

a50:b25:can[inexpr], hh  s ;   if −, in expr then rightreturn;
     pa  b25   , arn(b7)   ;   inexpr:= false; mark C:= C [stack[top 2]];
     hv  a83           NA  ;   if mark A = 0 then goto add point 2;
     hh  a82           LB  ;   if mark B = 1 then goto test right;
     hs  a48           LRA ;   if oldhalf then fill up in lefthalf; goto addpoint 2;
a82h:hv  a83   , pm  r     ; test right:
     hv  a83           LRA ;   if oldhalf then goto add point 2;
     hsn a106          LQA ;   if old A = 1 then blind half;
     hs  a26               ;   stack[top 2]:= stack[top 2] + point;
     ac (b7)   , hh  s     ;   right return;
a83: hs  a26               ; add point 2:
     ac (b7)           MPC ;   stack [top 2] markPA:= stack[top 2] + point;
     hh  s                 ;   rightreturn;
a106:bs  p5                ; blind half:
     hs  a11               ;   if t > 5 then change track;
     arn r     , hvn a87   ;   marks:= 10; R:= 0; goto store half in next;

     [<further param inf><kind><type> call param / case param]

a51: cl  −10   , ga  b35   ;   in case:= part 4 (M);
     hs  a120  , ga  b17   ;   type:= nextbyte;
     hs  a50   , hs  a20   ;   test expr; nextbyte;
     arn b17   , ps  6     ;   casetype:= not float case;
b21: nc[endinf], ps  8     ;   if type ≠endinf then casetype:= floatcase;
     ca  5     , ps  7     ;   if type = label then casetype:= labelcase;
     pm (b)    X d   6     ;   R:= kindtable [byte];
b35: bs [in case]          ;   if in case then
     ck  −10   , gs  b17   ;     begin R:= R shift −10; type:= casetype end;
     gt  b29   , ga  b31   ;   parmact:= part 2 (R); subscrparam:= part 1(R);
b29: tk  20   ,hv[paramact];   R:= R shift and clear 20; gotolabel(paramact);

     [<number of literals> end call]

a52: hs  a49   , hs  a20   ;   stack paraminf;
     ck  −10   , gt  b1    ;   instr:= instr + inpart 2 (nextbyte);
     hs  a3              NZ ;   if byte > 0 then store;
     pa  b23   , hs  a26   ;   p word 1 := 0; inexpr:= false;
     pa  b25   , ar  d51   ;   R:= point + arn;
a124:ps  a21   , hv  a31   ;   set return to (next); goto stack R2;

     [<endtype> end case]

a53: hs  a49   , hs  a30   ;   stack paraminf; stackpoint 2;
     it (b7)   , pa  b23   ;   p work 1:= top 2;
     pa  b25   , hs  a20   ;   in expr:= false;
     ga  b21   , ca  2     ;   endinf:= nextbyte;
     hs  a3                ;   if byte = real then store;
     ps  a21   , hv  a30   ;   set return to (next); stack point 2;
```

```
     [<five byte words><number of words> code]

a54: hs  a120  , ga  b38   ;   nextbyte; words:= byte;
     ck  2     , ga  r1    ;   if t > −4 × byte
a108:bs p[app] , hs  a11   ;     then start new code track: change tracks;
b38: bt[words] t   −1      ; nextword: words:= words − 1;
     ps  r1    , hv  a120  ;   if words < 0 then goto next;
     hv  a22   , ga  r2    ;   PC:= nextbyte;
     hs  a14   , ps  a108  ;   input lit; set return to (nextword);
     pi [PC]   t   −49     ;   store R;
     hv  a4                ;

     [generate formal instruction]

a55: arn b1                ;   R:= instr;
     ck  −10   , ar  d23   ;   instr:= R shift − 10 + ud;
     gr  b1    , hv  a15   ;   goto store with operand;

     [generate index check instruction]

a56: hs  a15               ;   store with operand;
     arn(b2)   , gr  b1    ;   instr:= word[w];
     tk  28    , ck  −8    ;   if bits(28, 29, instr) = 2 then
     pm  d24   , ca  2     ;     part 1(instr):= part 1(instr) + 1;
     qq (b1)   t   1       ;
     gm (b2)         MA    ;   word[w]:= pt c49−5, hs c 27
     hv  a3                ;   goto store;

     [statement param:]

a57: bs  p5    , hs  a11   ;   if t > −5 then change track
     is (b23)  , arn s−1   ;   R:= stack [p work 1 − 1];
     ps  a111  , hv  a38   ;   set return to (stack case expr);
                           ;   goto generate goto;
```

```
     [<no of indices><dope rel><array rel><block no> array param]

a58: hs  a14   , ar  d20   ;   input lit; R:= R + display base;
     ga  d49   , cl  30    ;   part 1 (arrayformat):= part 1 (R);
     tk  −10   , cl  −20   ;   pack arrayword; comment arrayrel,
     hs  a31   , qq  a21   ;     doperel, 0, no of indices; stack R2;
     arn d49   , hv  a65   ;   R:= arrayformat;
                           ;   set return to (next); goto finparam IPC;

     [expr/subscr var as param]

                           ; UV expr:
a59: ac  b1    , hs  a20   ;   instr:= instr + R; skip one zerobyte;
                           ; subscr variable:
a60: sr  d15           LA  ;   if mark A then R:= R − nbit;
                           ; UA−expr:
a61: ac  b1    , hs  a127  ;   instr:= instr + R; store; s:= true; R:= 0;
b31: can[subs] , sr  d15   ;   if subscrparam then R:= R − nbit; add format:
a107:ps  1     , it  d8    ;   R:= R + stackformat [type + base expr];
     ar (b17)  , hh  a102  ;   goto set in expr;

     [expr as case param]

a62: bs  p5    , hs  a11   ;
     arn(b23)  , hs  a38   ;   generate goto (stack2[p work 1]);
     bs (b17)  t   7       ;   if type = floatcase
     arn d13   , hs  a23   ;     then simulate input (nkf 39);
a111:hs  a20   , qq  1     ;   skip one byte; s:= true; stack case expr:
a102h:pmnr−3   , gs  b25   ;   mark C:= 0; R:= 0; set in expr: in expr:= s;
a117:ps  a21   , hv  a65   ;   set return to (next); goto f in param IPC;

     [lit in case]

a63: hs  a14   , ps  a21   ;   input lit; set return to (next); PC:= 0;
     bs (b17)  t   7   IPC ;   if type = floatcase
     nkf 39    , grf b12   ;     then const:= float (R);
     arn b12   , hv  a68   ;   R:= const; goto fin param;

     [lit in call]

a64: hs  a14   , ps  a65   ;   input lit; set return to (count call lit);
a65: hv  a68           IPC ; finparam IPC: set PC; goto finparam;
     qq (b23)  D   1       ; count call lit: pwork 1 := pwork 1 + 1;
     arn b22   , ck  −10   ;   R:= in part 2 (paramcount) +
     ar  d50   , hh  a67   ;     stackformat [lit in call]; goto set next;

     [descr in stack]

a66: pa  b17   t   9       ;   type:= descr in stack;

     [simple]

a67: hs  a16   , ps  a21   ;   input st addr; set next: set return to (next);
b17: ab[type]  t d7    IPC ;   R:= R ∨ stackformat [type + base simple];
a68: b22: qq [paramcount]t1;   RC:= mark C; fin param:
     hv  a31               ;   paramcount:= paramcount + 1; goto stack R2;
```

```
     [tk 1]

a69: bs  p516  , hv  a3    ;   if t + 5 ≤ 0 then goto store;
     pp  1     , hv  a22   ;   t:= 1; goto next;

     [store stack 1]

a70: bs  p5                ;   if t > 5 then change track;
     hs  a11               ;
     hs  a33               ;   take stack 1;
a71: hs  a4            IPC ; store R and right return: store R;
     hh  s                 ;   return;
a128:ar  c26−c2D           ;
     hv  a4                ;

     [begin block]

a72: it  1                 ;   procact:= false; goto local decl;

     [<proctype> begin proc]

a73: pa  b27   , nt (b21)  ;   procact:= true; local decl:
     pa  b1    , hh  r1    ;   part 1 (instr):= end inf; comment appetite;
     hv  a128  , hsn a34   ;   take stack 2; testpoint; set return to (store points);
     hs  a27   , qq  a94   ;   if not on same track then goto store R IPC;
                           ;   comment a27 returns to s − 1 if not local;
     pm (b2)   , ca  0     ;   if part 1 (R) ≠ 0 ∨ point is righthalf then
     hh  a93   X       LRA ;     begin R:= R shift −10 + is (c38), qq<s−r> 1;
     ck  −10   , ar  d43   ;     comment this addition replaces the r−mark by
a93h:hv  a4    , ck  −10   ;     an s−mark; goto store R IPC end;
     pi  0     t   −3  IPB ;   M:= R; oldhalf:= false; PB:= old B;
     ar  d44           IPA ;   R:= R shift −10 + hh (c38), ;
a94: hs  b2                ;   store in word;
     hs  a70   , it  −1    ; store points: store stack 1; pointcount:=
b24: bt[pointcount], hv r−1;   pointcount−1; if pointcount≥0 then goto store point;
     bs  p5    , hs  a11   ;   if t > 5 then change track;
     hs  a3    , qq  a75   ;   store; set return to (finish local);
```

```
     [unstack param inf:]

a74: hsn a34                ;   take stack 2;
     pa  b25   t   1   NA   ;   if mark A then in expr:= true;
     ga  b21   , tk  10     ;   end inf := part 1  (R);
     ga  b22   , tk  10     ;   paramcount:= part 2 (R);
     ga  b23   , tk  10     ;   pwork 1:= part 3 (R);
a75: hh  s     , ga  b24    ;   return;
                            ; finish local: pointcount:= part 4 (R);
b27: bs[procact], hv a22    ;   if −, procact then goto next;
     hs  a120  , hs  a26    ; prepare procedure point: nextbyte;
     ar (b)    V d8    IPC  ;   R:= point + set PC (formattable[byte]);
                            ;   goto blockpoint;

     [declare label]

a76: hs  a26                ;   R:= point;
     ps  a21                ;   set return to (next);
a99: qq (b24)  t   1        ; blockpoint: pointcount:= pointcount + 1;
     hv  a29                ;   goto stack R1;

     [<trackno><trackrel> call st proc]

a77: hs  a50   , hs  a39    ;   test expr; take programpoint and return right 2;

     [<block rel><−blockno> begin call]

a78: hs  a50   , hs  a16    ;   test expr; R:= input st addr + is( )f,pss;
     ar  d9    , hs  a65    ; stack exit param: finparam IPC;
     it (b22)  , pa  b1     ;   part 1(instr):= paramcount;
     pi  2.2   t   767      ;   change mode := 1
a95: bs  p516  , hv  a125   ;   if t > 5 then
     gi  r1    , hs  a11    ;     begin save QC; change track;
     pi  0     t   −13      ;     restore QC end;
a125:hsn a34   , qq  a96    ;   take stack 2; set return to (test next param);
     qq        V       LQC  ;   if −, old C ∧ −, mark C then goto stack R1;
     hv  a29           NC   ;   comment constant value;
     qq (b22)  t   1   NC   ;   paramcount:= paramcount + 1;
a96: hv  a71   , ps  a97    ;   store R; test next param: set return to (test lit);
     bt (b22)  t   −1       ;   paramcount:= paramcount − 1; if paramcount ≥ 0 then
a97: hv  a95   , it  −1     ;     goto loop call; loop lit: if p work 1 > 0 then
b23: ncn[pwork1], hv a70    ;     goto store stack 1 else
                            ; test lit: begin pwork 1:= pwork 1 − 1;
     ps  a114  , hv  a127   ;   goto loop lit end;
                            ;   set return to (finish case or call); goto store;
```

```
     [begin case]

a79: hs  a50   , arn b22   ;   test expr; casecount:= paramcount;
     ga  b13   , tk  −8    ;   paramapp:= 4 × paramcount
     gt r , nt[paramapp]   ;   if t + 15 > − paramapp
     bs  p17   , hs  a11   ;     then changetrack;
     bs (b13)  t   34      ;   if case count > 35 then
     ps  a84   , hv  e5    ;     mess(⊰<case too big⊱, 2, 0);
a80: hsn a34   , ar  d46   ; take case param: take stack 2;
     nc  0             LA  ;   if mark A ∧ part 1 (R) = 0 then
     ps  a81   , hv  a4    ;     begin testpoint; comment return s − 1 if
     hv  a128          LT  ;     not on same track;
     hs  a27   , qq  a81   ;     R:= R shift −10 + it(c16), end;
     ck  −10   , hh  a80   ;   store R; paramcount:=
a81=i−1                    ;   paramcount − 1; if paramcount > 0
     ncn(b22)  t   −1      ;     then goto take case param;
     hv  a80   , qqn d68   ;
a84=i−2 [mess descr]       ;
     arn(b7)   , hs  a35   ;   R:= stack[top 2]; prepare goto;
     hs  a4                ;   store R
     hs  a3                ;   store
     qq (b7)   t   2       ;   R:= qq[paramcount], hs c15; top 2:= top 2 + 2;
     arn b13   , ps  a114  ;   store RIPC;
a114:hv  a4            IPC ; finish case or call:
     hs  a74   , hv  a22   ;   unstack param inf; goto next;

     [end pass]

a119:hs  a11               ;   change track;
     arn(b2)   , gt  3e4   ;   inf 3:= w;
     hhn e29               ;   goto init run;

b k=e31, i=0               ; load text
i=e32                      ;
d68: tcase too big;        ;
e32=i                      ;
e                          ;
```

```
d5=i−1 [simulate− and auxtable]     [used as: by entry or in action]
d4=−d5+512

d67: [1] pa  c30   , tk  10             ; continueword[12d5]
     [ 2] bt (c33) f     , qq −1        ; simulateword          a46
d32: [ 3] qq  1.29                      ; addword               a101
     [ 4] udn c17      f,qq 18.27       ; simulateword          a46
     [ 5] qq  1.29+1.39                 ; addword               a101
     [ 6] gm (c30)     f , qq 63.29+1   ; simulateword          a46
d34: [ 7] hv  s                         ; addword               a7
     [ 8] pm (c17)     f , qq 31.29+1   ; simulateword          a46
d35: [ 8] hs  r                         ; addword               a7
     [10] gr  c17      f , qq 1.24      ; simulateword          a46
d43: [11] qq c38+1.19+36.25+1.27−1.39   ; addword               a73
     [12] qq  1.0+2.5+7 ft a3+d67.29    ; simulateword          a46
     [13] qq             , hs  c7       ; auxword[7d3,10d3,11d3]
     [14] itp 1.0+6.5+3  , qq a1+1021.29; simulateword          a46
d11: [15] hs  c2         ,              ; addword         a7/a11/a27
     [16] ga  1.0+6.5+3  , qq a3+c33.29 ; simulateword          a46
d24: [17] pt  c49−5      , hs  c27      ; auxword [18d5]
     [18] qq  1.0+2.5+7f t a3+d24.29    ; simulateword          a46
     [19] qq (c33)       , hs c39       ; auxword [60d3]
     [20] nc  1.0+6.5+3  , qq a1+1.29   ; simulateword          a46
d44: [21] hh (c38)       ,              ; addword               a73
     [22] qq  0          , hs c39       ; auxword[105d3]/stopworda46
d45: [23] hv  c11−1      , hh−1         ; addword               a47
     [24] ncn(c30)       , hs c13       ; auxword[42d3]
d23: [25] ud                            ; addword               a55
d46: [26] qq  37.25+1.27−1.38+c16, qq1  ; addword               a79
     [27] abn(c30)       , nkf 39       ; auxword [38d3]
     [28] arnf(c30)      , tkf −29      ; auxword[39d3]
d58:      pt  c49−4      , hs c27       ; auxword [12d3]
d30:      qq  512        , hs 1c15      ; addword               a79
d52:      ps  6.5        t  a15         ; continueword[98d3]
d53:      it  6.5        t  a15         ; continueword[97d3]
d38:      qq  64         , hs  c39      ; auxword[41d3]
d9:       is (  )      f , ps s         ; addword               a78
d10:      qq  1.39                      ; constant       a7/a101/a11
d14:      qq  −1.9+1.19−1.29            ; mask                  a15
d17:      hvr                           ; addword               a27
d18:      qq  4.25 [=hh−hv]             ; constant              a27
d19:      qq  −1.28[−rbit]              ; constant
d20:      qq  c0                        ; addword               a16
d28:      hs  c26        ,              ; addword               a39
d49:      ps ([dref])   f, hv c12       ; addword               a58
d50:      qq c10+1.26−1.29              ; addword               a64
d42:      ud  c37       f, hs  c37      ; continueword[46d3]
d63:      srn 4.5+4      , qq a3+c41.29 ; continueword[47d3]
d65:      hs  c14        X        NZA   ; auxword[48d3]
< e27 [buffermode]
d25:      arn            , hs c20       ; auxword[96d3]
×     [coremode]
d25:      arn  c35       , hv s2        ; auxword[d22]
d26:      srn            , hs c20       ; auxword[d48]
d48:      qq  2.5+5     ft a15+d26.29   ; continueword[d22]
d22:      qq  2.5+5     ft d48+d25.29   ; continueword[96d3]
>
```

```
d6=i−2 [Parameter−action−table]

[case act.|call act.|expr exit|<further parameter information>|kind]

< e27 [Buffer mode]

  qq , qq  a60+c18.19 ; <expr code                                subscr.var     2

×     [core mode]

  qq , qq  a60+c19.19 ; <expr code                                subscr.var    2
> zq a57 t  i         ; <code>                                    statement      3
  qq i   t a61+c19.19 ; <expr code>                               UA−expr        4
  qq a62 t a59+c18.19 ; <expr code><UV−rel>                       UV−expr        5
  zq i   t a43        ; <track no><trackrel><type>                st.proc        6
  zq i   t a66        ; <block rel><block no>                     descr.in stack 7
  qq a63 t a64        ; <4 bytes>                                 constant       8
  qq a67 t a67        ; <block rel><block no>                     simple         9
  zq i   t a58        ; <no of ind.><doperel><blockrel><blockno> array       10


     [Parameter−format−table]

d7=i−1 [simple formats]

ps  ( )   , it s                   ; 1 integer
ps  ( ) f t [it s] 37.25      +1.29; 2 real
ps  ( )   , itns                   ; 3 boolean
psn ( ) f t [itns] 37.25+1.26+1.29 ; 4 string
ps  ( ) f t [itns] 37.25+1.26+1.29 ; 5 label
is  ( ) f , arns                   ; 6 non float case
is  ( ) f t [its]  37.25     +1.29 ; 7 label case
is  ( )   , arns                   ; 8 float case
is  ( ) f , pmns                   ; 9 described in stack


     [expression− and procedure−formats]

d51: arn   , [also used in a52] ; −5 label procedure with parameters
     zqn f ,                    ; −4 no type   −      −       −
     zq    ,                    ; −3 integer   −      −       −
     zq  f ,                    ; −2 real      −      −       −
     zqn   ,                    ; −1 boolean   −      −       −
d8:  zqn f , [base expr/proc]   ;  0 notype expr or proc without param.
     psn   ,                    ;  1 integer  −   −   −     −      −
     psn f ,                    ;  2 real     −   −   −     −      −
d15: qqn   , [also used in a60] ;  3 boolean  −   −   −     −      −
     qqn f ,                    ;  4 string   −   −   −     −      −
d16: qq  f , [also used in a26] ;  5 label    −   −   −     −      −
```

[Input−table]

```
d41=i−512 [used in action a22]
d3: qq (p)        , qq  a1         ;          <blockrel>           addr local      0
    qq  p         , qq  a1         ;          <blockrel>           var local       1
d60:qq  c [stackref],qq a1         ;          <blockrel>           var abs         2
    qq  s         , qq  a5         ; <blockrel><−blockno>          var block       3
    qq (c30)      , qq  a3         ;                               (UA)            4
    qq  c17       , qq  a1         ;          <UV−rel>             UV              5
    qq 1.2+2.5   ft    a77+13d5.29 ; <trackno> <trackrel>          stproc call     6
    qq 1.2+2.5   ft    a78+13d5.29 ; <blockrel><−blockno>          begin call      7
    qq r c42+i    , qq a12         ;<bits30−39><20−29><10−19><0−9> constant        8
    qq 1.2+2.5+5  , qq a25+d31.29  ; <trackno><trackrel>           stproc simple   9
    qq 2.2+2.5+5 ft    a72+13d5.29 ;                               begin block    10
    qq 2.2+2.5+5 ft    a73+13d5.29 ;          <proctype>           begin proc     11
    qq 3.2+2.5   ft    a79+ d58.29 ;                               begin case     12
    pa 3.2        , qq a79+ c30.29 ;                               begin sw case  13
    qq(1.2    +5) , qq a52+ c35.29 ;          <number of literals> end call       14
    arn1.2+5.5+4  , qq a53+ c17.29 ;          <case type>          end case       15
    hh 1.2+6.5+5  , qq a47+ c22.29 ; <appetite><−blockno>          end proc       16
    hv 1.2+6.5+5  , qq a47+ c21.29 ; <appetite><−blockno>          end typeproc   17
    hs 1.2+6.5+5  , qq a47+  c9.29 ; <appetite><−blockno>          end block      18
    hvn1.2+6.5    , qq a51         ; <further p.inf><kind><type>   call param    19
    qq 1.2+6.5+3  t    a51+   1.29 ; <further p.inf><kind><type>   case param    20
    qq            t    a21         ;                               if/for         21
    qq 1.2+    5  t    a40      NT ;                               hop NT         22
    qq 1.2+    5  t    a40      LT ;                               hop LT         23
    qq 1.2+    5  t    a40      NZ ;                               hop NZ         24
    qq 1.2+    5  t    a40      LZ ;                               hop LZ         25
    qq 1.2+6.5+5  t    a42         ;                               bypass abs     26
    qq 1.2+    5  t    a41      NT ;                               bypass NT      27
    qq 1.2+    5  t    a41      LT ;                               bypass LT      28
    qq 1.2+6.5+5  t    a36         ;                               do abs         29
    qq 1.2+6.5+5  t    a37         ;                               goto bypass    30
    qq 1.2        t    a30         ;                               bypass label   31
    qq 1.2+6.5+5  t    a45         ;                <thentype>     else           32
    qq 1.2        t    a44         ;                <elsetype>     endelse        33
    hs 1.2        t     a8         ;                <p−rel op>     enddo          34
    hs 5.2        t     a8         ;                <p−rel op>     endsingledo    35
    qq            t     a7         ;                <p−rel op>     take forlabel  36
    qq         9  t    a131        ;                               formal assign  37
    qq 3.5+27d5.39+13 ft d61       ;                               take real value38
    qq 3.5+28d5.39+13 ft d62       ;                            take integer value39
    hv r        0 , qq a46+1020.29 ;<no of indices><array rel>  move array descr  40
    qq 1.2+2.5+5 ft a3+d38.29      ;                               writecr        41
    qq 1.2+2.5+ 5ft     a3+24d5.29 ;                               goto computed  42
    pm c55       f, ga  c55        ;                               select 1       43
    tln  9       f, ud  c55        ;                               select 2       44
    tk     4.5    , qq a69+   1.29 ;                               tk 1           45
```

[Input−table]

```
    ck 1.2+6.5+3  , qq d42+  10.29 ;                                lyn            46
    qqn        5  t d63        NKB ;                                kbon           47
    qq         5 ft a3+     d65.29 ;                          hs   multiply SNZA
    mt −1       f , qq 1.22+ 20.27 ;                               mt−1DNT        49
    il     6.5+3  t    a3          ;                               il             50
    mt −1       f , qq 1.22+ 28.27 ;                               mt−1DLT        51
    us     6.5+3  t    a3          ;                               US             52
    mt     6.5+3  , qq a3  + c44.29;                               mt neg         53
    sr     4.5+4  , qq a3  + c41.29;                               sr eps         54
    sr     5.5+4  , qq a3  + c43.29;                               srf half       55
    pm     6.5+3  , qq a3  + c17.29;                               pm UV          56
    pm     6.5+3  , qq a3  + c30.29;                               pm UA          57
    arn    4.5+4  , qq a3  + c30.29;                               arn UA         58
    ga     6.5+3  , qq a3  + c30.29;                               ga UA          59
    qq 1.2+2.5+5f t a3+19d5.29     ;                               outchar var    60
    ck (   6.5+3) , qq a3  + c33.29;                               ck (address)   61
    tk     4.5+4  , qq a3  +  30.29;                               tk 30          62
d21:ck     6.5+3  , qq a3  +1014.29; [used by 64d3]                ck − 10        63
    ga     6.5+3  , qq d21 + c33.29;                               integer to addr64
    ab         5  D X  a3          ;                               ab 0 DX        65
    ar c41      f , qq        28.27;                               ar eps LT      66
    xr     6.5+3  t    a3          ;                               xr             67
    tk     5.5+4  , qq a3  + 995.29;                               tkf −29        68
d13:nk     5.5+4  , qq a3  +  39.29;                               nkf 39         69
    pm     6.5+3  t    a15         ;                         <op>  pm             70
d36:gm     6.5+3  t    a15         ; [used by action a7] <op>      gm             71
    mk     5.5+4  t    a15         ;                     <op>      mkf            72
    dk     5.5+4  t    a15         ;                     <op>      dkf            73
    qq     6.5+3  t    a15         ;                     <op>      qq             74
    mt     6.5+3  t    a15         ;                     <op>      mt             75
    snn    4.5+4  t    a15         ;                     <op>      snn            76
    ann    4.5+4  t    a15         ;                     <op>      ann            77
    mb     6.5+3  t    a15         ;                     <op>      mb             78
    ab     6.5+3  t    a15         ;                     <op>      ab             79
    grn    4.5+4  t    a15         ;                     <op>      grn            80
    dln    6.5+3  t    a15         ;                     <op>      dln            81
    ann        5  X    a15         ;                     <op>      ann X          82
    dln        5  X    a15         ;                     <op>      dln X          83
    sr         5  t    a15     LT  ;                     <op>      sr LT          84
    hs 1.2+    5  t    a15         ;                     <op>      hs             85
    pm         5  D    a15         ;                     <op>      pm D           86
    arn        5  D    a15         ;                     <op>      arn D          87
    ar         5  D    a15         ;                     <op>      ar D           88
    gr         5  t    a15      MA ;                     <op>      gr MA          89
```

[Input−table]

```
    gr         5  t    a15      MB ;                    <op>      gr MB          90
    gm         5  t    a15      M  ;                    <op>      gr M           91
    grn        5  t    a15      M  ;                    <op>      grn M          92
    mln        5  X    a15      IZA;                    <op>      mln X IZA      93
    acn       13  t    a15      MA ;                    <op>      acn MA         94
    mb         5  X    a15         ;                    <op>      mb X           95
<e27 [buffermode]
    qq 1.2+2.5+5 ft    a15+ d25.29 ;                    <op       reserve array  96
×    [coremode]
    ck 1.2+6.5+3  , qq d22+  10.29 ;                    <op       reserve array  96
>   pa     6.5+7  , qq d53+ c30.29 ;                    <op>      var to UA      97
    hv 1.2+6.5+4  , qq d52+  c2.29 ;                    <op>      go to local    98
    arn        9  V    a56      LT ;                    <op>      index upper    99
    sr         9  V    a56      NT ;                    <op>      index lower   100
    hs 1.2+2.5+5  , qq a55+ c28.29 ;                    <op>      move formal   101
    hs 1.2+2.5+5  , qq a55+  c8.29 ;                    <op>      take formal   102
    hs 1.2+2.5+5  , qq a55+ c25.29 ;                    <op>      contr. formal 103
    hs 1.2+2.5+5  , qq a55+ c24.29 ;                    <op>      take assign   104
d12:hs  c3                         ; [used in a11]               internal     [105]
    ck    +6.5+3  , qq a           ;      <address constant>     ck            106
d61:arn (c30)   f , qq1.21+3.24+1.29; [used in 38d3]             internal     [107]
    qq 1.2+2.5+5f t a+22d5.29      ;      <address constant>     outchar const 108
    ps p   6.5+3  , qq a1+    2.29 ;      <no of formals>        ps p          109
    qq 1.2        t    a76         ;                             label declar 110
d33=d32−i [used by action 1101. d32=3d5]
    hs 1.2+2.5+5  , qq a101+ c4.29 ;                             ⍋ real        111
    qq            t    a110        ;                             carret        112
    hs 1.2+2.5+5  , qq a101+ c4.29 ;                             ⍋ integer     113
    qq  1.2       t    a54         ; <words><no of words>        code          114
    qq            t    a9          ;       <do app>              newtrack      115
    qq            t    a119        ;                             endpass       116
d62:arn (c30)   f , qq 1.21+2.24+1.29; [used by 39d3]            internal     [117]
d47:gm         5  t    a15      MA ; [used in a7]               internal     [118]
d37:pmr    6.5+3  t    a13         ; [used in a7]               internal     [119]
d29:hs  c1                         ; [used in a11]               internal     [120]

    [Adding 512 to the following byte values will cause the generated
          instruction to be f−marked]

d40=i−d3+512 [used in action a22]
    arn     4.5+4  t    a15        ;                    <op>      arn            121
    ar      4.5+4  t    a15        ;                    <op>      ar             122
    sr      4.5+4  t    a15        ;                    <op>      sr             123
    gr      4.5+4  t    a15        ;                    <op>      gr             124
    srn     4.5+4  t    a15        ;                    <op>      srn            125
    ann     4.5+4  t    a15        ;                    <op>      ann            126
    gr          6  t    a15     M  ;                    <op>      gr      M      127
    gr          6  V    a15     LA ;                    <op>      gr      VLA    128
```

```
d1=i-1                    ; base stack 1 ↓

a121:gr  d2    t  -1   M  ; initialize stack to zero
     ca (r1)  , hv  a22   ; and goto next;
     hv  a121             ;

     [Initialize Pass 8]  ; Entry to pass 8 is here:

a122:srn(13e4) D          ;
     ck  10    , sc  b8    ;   base:= base + current track;
     sc  d32   , sc  2d32  ;   ↓descr:= ↓descr + current track;
     hs  a120  , ac  d60   ;   sr0:= nextbyte;
     ga  2e4   , arn e4    ;   inf 1:= sr0;
     tk  30    , ga  b37   ; linecount:= CRcount mod 1024;
     ck  10    , ar  d57   ; word[w]:= exitword + inpart 4(linecount);
     gr  d54          MA   ; t:= maxapp; oldhalf:= sets:= false;
     pt  e12   t   -41     ; change GPA to pass 8 mode
     pp  d0    , pi  8     ; oldA:= 1; oldB:= 0;
     hvn a121              ; R:= 0; goto iniststack;

d57: hhn c29   , [lastline]; exitword:

d2=e20                    ; base stack 2 ↓

d66: qq [pass sum]        ;
d i=i-1, e22=k-e14+1, e47=0; set load parameters. Note e22 to first free track.
b k=e23, i=0              ; Load segment word 15
i=15e21                   ;
qq e16.9+1d66.19-e16.19+1.20+a122.39f ;
e                                     ;

e [final end]                         ;

s
```

```
d e49=6    ; tape number := 6;
d i=e4     ;
 qq e14.39 ; first track of system (for merger)

[After i follows STOPCODE, SUMCODE and a sum character]
ia T5
s
```

```
[Here follows STOPCODE and CLEARCODE]

d e56=1             ;   test tape number
<e49−5, <−e49+7     ;   should be 2 or 6
d e56=0             ;
×                   ;
<e49−1, <−e49+3     ;
d e56=0             ;
>                   ;
<e56                ;
i wrong tape
s                   ;
>                   ;

d e56=12            ;   version number

<e56−e50,           ;   if version number T6 and L2 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=9             ;   define version number T2
d e63=10            ;   define version number T3
d e64=4             ;   define version number T4
d e65=11            ;   define version number T5
d e66=e56           ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e56
>                   ;
```

```
b d10                       ;   outermost block
d d1=e14                    ;
<e48, d1=d1+e22>            ;
d i=1e4                     ;
     qq  d1.39              ;   parameter to Merger: first track of library
                            ; BEGIN LOADER CODE:
b i=3e4, a30, b35, c15      ;
a:   qq                     ;   saved M, partial identifier
[1a] qq                     ;   saved M, partial textword
[2a] qq  67.39,             ;   constant used by c9+1
[3a] qq  0.9                ;   FTR of latest std proc
[4a] qq  39.39              ;   constant
[5a] qq  1.9                ;   word size std proc spec
[6a] qq  40.39              ;   constant
[7a] qq  1.19               ;
[8a] 1023/0/1023/1023       ;   mask for output short word
[9a] qq  43.39              ;   constant

d b2=i, b3=41b2             ;   define output buffers
d d=d1, i=41b3              ;   define running std proc track
                            ; ERRORPRINT:
c1:  vy  16    , sy  29     ;   select(typewriter); RED RIBBON;
b1:  qq[textaddr],hs b16    ;   writetext(name of current std proc);
     hsf 2                  ;   return to HELP;
b16: sy  58    , sy  64     ; writetext: writechar(lower case); writecr;
     it (s)    , pa  b20    ;   addr:= part 1 of core[s];
b20: pmn 0     X            ; next word: M:=0; R:=core[addr];
a1:  cl  34    , ck  -4     ; next char: char:=next;
     ga  a2    , ca  15     ;   if char=15 then
     hv (b1)   D   1        ;    begin addr:=addr+1; goto next word end;
     ca  10    , hv  s1     ;   if char=10 then return;
     ca  63    , it  1      ;   if char=63 then char:=char+1;
a2:  sy [char] , cln -6     ;   writechar(char);
     hh  a1                 ;   goto next char;
                            ; OUTIDENT:
c2:  gr  3b2   t   1   MRC  ;   store R in buffer; marks:=RC;
     it (c2)   , bs  40b2   ;   if buffer not full then
     hr  s1                 ;    return;
     vk (b4)   , sk  b2     ;   output buffer to drum;
b4:  vk  d1    t   1        ; [outident track]
     hv (c2)   D   -41      ;
                            ; OUTSPECIF:
c3:  gm  b3-1  t   1   MRC  ;   store R in buffer; marks:=RC;
     it (c3)   , bs  40b3   ;   if buffer not full then
     hr  s1                 ;   return;
     vk  (b5)  , sk  b3     ;   output buffer to drum;
b5:  qq  0                  ;   [outspeciftrack]
     [b13-2 replaces this by
      vk d1+ident tracks t1]
     hv (c3)   D   -41      ;
```

```
c4:  pmn(b6)   Xt   1      ; NEXT TEXTWORD: increase ident address;
a4h: cl  34    , ck  −4    ;   M:= next identword; Radr:= next char;
     ga  a3    , it  b7    ;   execute (inputtable[next char + base]);
a3:  ud  0     , hh (r)    ;   if next char −< ciffer
     bs  p     , hv  c1    ;     ∧ case = upper then goto errorprint;
     gm  1a    , pm  a     ;   if next char −< ciffer ∨ next char −< letter
     hr  s1                ;     then begin save text M; M:= partial ident;
                           ;     Radr:= pass 2 value; return end;
c5:  gm  a     , pm  1a    ; GET PASS 2 CHAR: save M;
a5:  cln −6    , hh  a4    ;   M:= saved text M; goto get next char;

     [inputtable, executed from get pass 2 char]

b7:  qq        , hv  a5    ; space     [0]     blind
     arn 58    D           ; 1
     arn 59    D           ; 2
     arn 60    D           ; 3
     arn 61    D           ; 4
     arn 62    D           ; 5
     arn 63    D           ; 6
     arn 64    D           ; 7
     arn 65    D           ; 8
     arn 66    D           ; 9
     arn a     , hr  c6    ; end text  [10]    R:= partial ident; return to
     qq        , hv  a5    ; stop code         blind               end ident
     qq        , hv  a5    ; end code          blind
     qq        , hv  c1    ; inadm             goto errorprint
     qq        , hv  c1    ; inadm             goto errorprint
     qq        , hv  c4    ; end word          goto next textword
     arn 57    D           ; 0
     qq        , hv  c1    ; inadm             goto errorprint
     arn p19   DV          ; s
     arn p20   DV          ; t
     arn p21   DV          ; u         [20]
     arn p22   DV          ; v
     arn p23   DV          ; w
     arn p24   DV          ; x
     arn p25   DV          ; y
     arn p26   DV          ; z
     qq        , hv  c1    ; inadm             goto errorprint;
     qq        , hv  c1    ; inadm             goto errorprint;
     qq        , hv  c1    ; inadm             goto errorprint;
     qq        , hv  c1    ; inadm             goto errorprint;
     qq        , hv  a5    ; tab       [30]    blind
```

```
qq       , hv  a5    ; punch off [31]    blind
qq       , hv  c1    ; inadm             goto errorprint
arn p10  DV          ; j
arn p11  DV          ; k
arn p12  DV          ; l
arn p13  DV          ; m
arn p14  DV          ; n
arn p15  DV          ; o
arn p16  DV          ; p
arn p17  DV          ; q         [40]
arn p18  DV          ; r
qq       , hv  c1    ; inadm             goto errorprint
arn p28  DV          ; ø
qq       , hv  a5    ; punch on          blind
qq       , hv  c1    ; inadm             goto errorprint
qq       , hv  c1    ; inadm             goto errorprint
qq       , hv  c1    ; inadm             goto errorprint
arn p27  DV          ; æ
arn p1   DV          ; a
arn p2   DV          ; b         [50]
arn p3   DV          ; c
arn p4   DV          ; d
arn p5   DV          ; e
arn p6   DV          ; f
arn p7   DV          ; g
arn p8   DV          ; h
arn p9   DV          ; i
pp  0    , hv  a5    ; LC                case:= lower
qq       , hv  c1    ; inadm             goto errorprint
pp  28   , hv  a5    ; UC        [60]    case:= upper
qq       , hv  c1    ; inadm             goto errorprint
qq       , hv  c1    ; inadm             goto errorprint
qq       , hv  a5    ; CR                blind
```

[Check and transformation of proctable]

```
u i                          ; ENTRY TO STANDARD PROC LOADER:
b11: pm  d4    X             ;   for i:= identbase,
     hv (r-1)  Dt 1    NB    ;     i+1 while core[i] not f-marked do;
     ga  b12                 ;   speciftop:= i; top section 3:= core[i];
     arn d4    D             ; move proctable to core top:
a6:  pm (b11)  t   -1        ;   for i:= speciftop step -1 until identbase do
b6:  gm  1023  t   -1        ; [proctable adr]
     nc (b11)  , hv  a6      ;   proctable[i - identbase]:= core[i];

c7h: it  -1    , it (b6)     ; NEXT IDENTIFIER: prepare errorprint:
     pa  b1    , arn b6      ;   textaddress:= proctable adr;
     ca  1022  , hv  c8      ;   if proctable adr = coretop then goto out to drum;
     ps (b8)   , pp  0       ;   s:= final proctable adr; case:= lower;
     hsn c4              IZC ;   short:= empty:= true; call (next textword);
     ga  r1    , it  56      ;   first char:= pass 2 value; partial ident:= 0;
b9:  bsn 0     , hv  c1      ; [first char] if first char>56 then goto errorprint;

c9:  hs  c5    X         IOB ; CONTINUE IDENT: empty:= false;
     ck  10    , ml  2a      ;   partial ident:= partial ident × 67 + pass 2 value;
     hv  c9    X         LZ  ;   if partial ident<2↑39 then goto continue ident;

b8:  gm  d4    Vt   1    NZA ; [final proctable adr] if -, short then
a7:  cl  19    VX        IZA ;     begin store long word; goto continue ident end;
     hv  c9                  ; store short: store bits 20-39; marks:= long ident;
     ck  -19   , ud  a8      ;   partial ident:= partial ident : 2↑20;
     hv  c9              LA  ;   if not called from end ident then
                             ;     goto continue ident;
c6:  hv  c1              LZB ; END IDENT: if empty then goto errorprint;
     pm   0    DV        LZA ;   M := 0 ;
     pm  512   DVX       NZA ;   if short ∧ part 1 (partial ident)=0
     ca  0     , hv  a8      ;   then goto short ident;
     hv  a7    X         LZA ;   if short then goto store short;
     gm (b8)   t   1         ;   store long word;
     it (b8)   , pt  s1      ;   part 2(final proctable[s+1]):=final proctable adr;
     ac  s2    V             ;   bit 0 (final proctable[s+2]):= 1; skip line;
a8:  gr (b8)   Xt   1  MZA   ; short ident: store short word; marks:= short;
     [ executed from 2a7 ]
     it (b9)   , pa  s1      ;   part 1(final proctable[s+1]):= first char;

     pmn(b6)   Xt  1         ; MOVE SPECIFIKATIONS:
     gr (b8)   Vt  1   NZ    ;   move spec 1 from proctable to final proctable
     hv  c1                  ;   if spec 1 = 0 then goto errorprint;
     cl  30    , cln -10     ;
     ga  a14   , pi  128     ;   rel:= part 3 (spec 1); prepare test of ETR;
     cln -5    , tk  -15     ;
     gt  a9    , cln -5      ;   ETR:= bits 15-19 (spec 1);
     tk  -5    , ga  a9      ;   NTR:= bits 10-14 (spec 1); FTR:= part 1 (spec 1);
     ca  0     , hv  a10     ;   if NTR = 0 then skip specifikation check;
a14: it [rel]  t  472        ;   if rel < 0 ∨ rel > 39 then
     bs  472   , hv  c1      ;     goto errorprint;
a9:  bs[NTR]   Xt [ETR]      ;   if NTR ≤ ETR
     sr  3a              ITA ;     ∨ FTR < latest FTR then
     hv  c1              LTA ;     goto errorprint;
a10: pm (b6)   t   1         ;   move spec 2 from proctable to final proctable;
     gm (b8)   t   1         ;   latest FTR:= FTR;
     ac  3a    , it  2       ;   spec length:= spec length + 2;
     qq  (5a)  , hh  c7      ;   goto next identifier;
```

```
c8:  grn(b8)   t   1   M   ; MOVE SECTION 3:
     pmn b12   X
     sr  d1    D           ;   The program Merger requires 4 parameters:
     cl  10    , mln 6a    ;   word[0], word[1], word[2], word[3];
     cl  −20   , ar  7a    ;   word[2]:= (std proc code tracks×40) pos 19;
     gr  2b2               ;   word[2]:= word[2] + 1 pos 19;
     pmn (b8)  X D 1       ;   word size std proc ident := R :=
     sr  d4    D           ;   final proc table − final proc table base+
     sr  5a    , ck  −10   ;   2−word size std proc spec;
     gr  b2    , ck  −20   ;   word[0]:= word size std proc ident pos 19;
     ar  9a    , gm  3b2   ;   M:=R+39;
     pa  3b2   X   e19     ;   word[3]:= checksum RS−names pos 9;
     dln 6a    , gr  1b2   ;   ident tracks:= M:40;
     ac  2b2   , ck  −10   ;   word[2]:=word[2]+ident tracks;
     gr  b5    , arn 5a    ;   R:=word size std proc spec;
     ck  −10   , gt  1b2   ;   word[1]:=word[1]+ R pos 19;
     ck  −20   , ar  4a    ;   displacement:= (R+39):40;
     pp  (b12) X   −1      ;   p:=top section 3:=top section 3 −1;
     dln 6a    , ac  2b2   ;   word[2]:=word[2]+displacement;
     ck  −10   , ar  b5    ;   displacement:=displacement+ident tracks;
     ga  b13   , ac  b12   ;   form the final vk instruction
     arn b4    , ac  b5    ;   in  b5;

     vk  p     , lk  b3    ;   for p:=p step −1 until base section 3 do
b13: vk  p0    , sk  b3    ;   move track(p) to track: (p+displacement);
     [displacement]
     vk  p−1   , nc  p     ;   wait drump;
     pp  p−1   , hh  b13−1 ;

b10: arn d4    t   1   IRC ; [i]  I:=base proctable;
     hv  c10            LZ ;   if ident=0 then goto END TO DRUM;
     gt  a11               ;   long address:= part 2 of (final proctable[i]);
     mb  8a             NA ;   if long ident then clear bits 10−19;
a11h:hs  c2    , qq  0     ; [long address]  outident; s:=long address;
     hv  a13            LRA ;   if long ident then
     gs  b10   , gs  a12   ;    begin i:=long address;
a12: arn 0              ITA ; [n] for n:=long address step −1 until
     hs  c2                ;     proctable less 0 do
     hv  (a12) D   −1  NTA ;   outident(final proctable[n]);
                           ;   end;
a13: pm  (b10) t   1       ;
     hs  c3                ;   i:=i+1; outspecif(final proctable[i]);
     pm  (b10) t   1       ;   i:=i+1; outspecif(final proctable[i]);
     ps  b10−1 , hv  c3    ;   goto next proc out;

c10: pi  0     , hsn c2    ; END TO DRUM: outident(0); comment checksum;
     hsn c3    X           ;   outspecif(0); comment checksum;
     vk  (b4)  , sk  b2    ;   output last ident buffer;
     vk  (b5)  , sk  b3    ;   output last specif buffer;
     vk  (b5)              ;   wait drum;
```

```
b12:
d6:  arn[last track]D t  1 ;   form the parameter for binout; this looks:
     ck  10    , gr  2e4   ;
     ck  10                ;
     ar  d1    D           ;   qq  first track.0+0.19+last track+1.29;
     ac  a15               ;
a24: grn −1    , hsf 2     ; punch library: hsf2 (image dump);

     hs  1                 ;
     hv  b18               ;
     toutparam;            ;   outparam,
     tbinin;               ;            binin,
     tfree;                ;            free
     qqf,                  ;            <

b18: hs  1                 ;
     hv  b19               ;
     tbinout;              ;   binout,
     qq  50,               ;          b,
     qqf 0                 ;          0,
     qq  37,               ;          n,
a15: qqf[library tracks]   ;          library tracks
     qqf,                  ;          <

b19: hs  1                 ;
     hv  a24               ;
     toutparam;            ;   outparam,
     tt;                   ;             t
     qqf,                  ;             <
e                          ;   goto punch library;
d d4=i, e49=7              ;   tape number := 7;

[After i follows STOP CODE, SUM CODE and a sum character]

ia  T6, L2
s [end loader code]        ;
```

```
[Here follows STOPCODE and CLEARCODE]

<e49−6, <−e49+8, ×  ;   test tape number
i wrong tape
s                   ;
>                   ;

d e57=7             ;   version number

<e57−e50,           ;   if version number T7 and L3 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=2             ;   define version number T2
d e63=3             ;   define version number T3
d e64=4             ;   define version number T4
d e65=5             ;   define version number T5
d e66=6             ;   define version number T6 and L2
d e67=e57           ;   define version number T7 and L3
d e68=8             ;   define version number T8 and L4
d e50=e57
>                   ;
```

```
b e                          ;
b k=d,i=11,a20,b40,d10       ;
                             ;
b:   pin 0     X             ; read integer: M:=0; number:=real:=false;
b1:  arn c54   , tk  31      ; start:
     tk -1     , ga  ra      ;    Raddr:=char; comment R00=R0;
     mb  ra8   , sc  ra      ;    case:=128×casebit of char; Raddr:=char-case;
     gp  ra10  , pa  ra9     ;    save p:=p; sign:=0;
a:   pp [case] , hv  ra4     ;    savep:=p; p:=case; goto first;
                             ; digit:
a1:  hv [read real]    LTB   ;    if real then goto read real;
b2:  ck  10    , ml  c58     ; mult: M:=M×10+ integer(R shift 10);
     dl  c58   V X      NZ   ;    if -,overflow then
b25: hvn ra3           IZA   ; numb: begin number:=true; goto next end;
                             ;    M:=RM:10;
a2:  ps  b32                 ; sign act: s:=2;
     hv (ra1)          LTB   ;    if real then goto read real;
     hv  ra8           LOA   ;    if number then goto terminator act;
b3:  it  p-128 , pa  ra9     ; sign: sign:=p-128; goto next;  plus: sign:=0;
                             ; next:
a3:  ud  c37   , hs  c37     ;    next in;
a4:  ga  c37   , is (c37)    ; first: part 1 of next in:=Raddr;
     bs  s502  t  502        ;    if the char+p>0 ∧ the char+p<10 then
a5:  ps  b31   , hv  ra1     ;     begin s:=1; goto digit end;
     ps  b35   , ca  p16     ;    s:=5; if the char=p+16 then
     ps  b31   , hvn ra1     ;    begin s:=1; R:=0; goto digit end;
     ca  32    , hv  ra2     ;    if the char=32 then goto sign act;
     ca  58    , ppn 0       ;    if the char=58 then p:=R:=0;
     ca  60    , ppn 128     ;    if the char=60 then begin R:=0; p:=128 end;
     hv  ra3             LZ  ;    if R=0 then goto next;
     ca  p59   , ps  b33     ;    if the char=p+59 then s:=3;
     ca  p-101 , ps  b34     ;    if the char=p-101 then s:=4;
     mb  ra6                 ;    if bits(4,9,R)=63 then
a6:  ca  63    , hv  ra3     ;     goto next;
a7:  hv (ra1)          LTB   ;    if real then goto read real;
                             ; terminator act:
a8:  qq  895   V X     LOA   ;    if -, number then
a9:  qq [sign] , hh  rb3     ;     goto plus;
     mt  ra9   , gr  c17     ;    UV:= if sign<0 then -M else M;
b4:  arn(c37)  D             ; store terminator:
     ck  10    , gr  c54     ;    char:= integer((the char+p) shift 10);
a10: pp[save p], hh  c5      ;    p:=save p; goto exit std proc;
                             ; entry read real:
b5:  pmn c53   , grn c17-1   ;    exp:=0;
     xr  1     , hs  c6      ;    rel track(1); factor:=1.0; M:=0;
     ps  s1    , gs  ra1     ;    read real:=trackplace+1;
     it  ra9   , pa  sb23    ;    real:=true;
     pi 1.1+1.3, hv  rb1     ;    number:=after ten:=false; goto start;

d b5=b5-b,e=b5,b2=b2-a9      ;
d b3=b3-a9,b4=b4-a9          ;
d a3=a3-a9,b25=b25-a9        ;
```

```
b6:                          ; read real:  symb:=R;
b11: gr  c17−2 , gs  rb7     ; look:
     gm  c17           LOB ;    if −,after ten then UV:=M;
b7:  arn r0    , tl −2       ;    R:=state table[s];
     ga  rb7   , it  6       ;    action[s]:=part 1 of ((R shift −2)∧ 10 255);
b8:  tl[state.7],mb  rb9     ;    R:= (R shift state×4+4)∧ 10 60;
b9:  nc  60    , hv  rb10    ;    if part 1 of R =60 then
     hvn rb10          LOA ;     begin if number then cleargoto ST;
     arn(rb13)         NOB ;       if after ten then R:= location[addr(10.0)+neg];
     ga (rb10) , pa  rb8     ;       sign:= part 1 of R; state:=0;
     hhn rb11  X       IZB ;       M:=0; after ten:=false; goto look
                             ;     end;
b10: ps [a9]   , ga  rb8     ; ST: state:= part 1 of ((R shift −2)∧ 10 15);
     nc  0     , hv (rb7)    ;    if state≠0 then goto action[s];
                             ; terminator:
d:   hhn sb3   X       NOA ;    if −,number then begin M:=0; goto plus end;
b12: pm  c17   X           ;    R:=UV;
     nkf 39    , dkf c50     ;    RF:= float(R)/factor;
     bt (c17−1)t   −1        ;    for exp:=exp step −1 until 1 do
b13: mkf rb20  , hv  r−1     ;     RF:=RF×location[addr(10.0)+neg];
     ns (s)    , bs  s       ;    if sign<0 then
     grf c17   , srnf c17    ;     RF:= −RF;
     grf c17   , hv  sb4     ;    UV:=RF; goto store terminator;
                             ; digits:
b14: nc  16    , hh  rb15    ;    if state=4 then
     arnf c50  , mkf rb20    ;     factor:=factor×10;
b15: grf c50   , pm  c17     ;    M:=UV;
     arn c17−2               ;    R:=symb;
     hv  sb2           LOB ;    if −,after ten then goto mult;
     pm  c17−1 , ml  c58     ;    exp:=exp×10+R;
     gm  c17−1 , hv  sb25    ;    goto numb;
                             ; signs:
b18: hvn sb3   X       LOB ;    if −,after ten then begin M:=0; goto sig end;
     bs  p384  , it  d7      ;    if p<128 then neg:=1 else
b19: pa  rb13  t   d6        ; ten: neg:=0;
     arn c42           IZB ;    if −,number then UV:=1;
     gr  c17           NOA ;    after ten:=true;
b17: pm  c17   , hv  sa3     ; point: M:=UV; goto next;
                             ;
b20: qq  3     , qq  320     ; comment 10.0;
b21: cm (r−4)  , can r409    ; comment 0.1;

d b23=b10−b6,d1=b14−b7,d2=b18−b7,d3=b17−b7,d4=b19−b7,d5=d−b7        ;
d b31=i−b7,b32=1b31,b33=2b31,b34=3b31,b35=4b31,d6=b20−b13,d7=b21−b13;

        [action]    [0]   [1]   [2]   [3]   [4]   [5]   [6]   [7]   ; state table:
     qq d1[ b14].7+ 2.11+ 2.15+ 2.19+ 4.23+ 4.27+ 7.31+ 7.35+ 7.39  ; digit
     qq d2[ b18].7+ 1.11+ 1.15+ 0.19+15.23+ 0.27+ 6.31+15.35+ 0.39  ; sign
     qq d3[ b17].7+ 3.11+ 3.15+ 3.19+15.23+15.27+15.31+15.35+15.39  ; point
     qq d4[ b19].7+ 5.11+ 5.15+ 5.19+15.23+ 5.27+15.31+15.35+15.39  ; ten
     qq d5[ d  ].7+ 0.11+ 0.15+ 0.19+15.23+ 0.27+15.31+15.35+ 0.39  ; all others

e                                ; end a, b and d names
                                 ;
t read integer;                  ;
qq d−d1.9+1.14+0.19+0.29+1.33+16.39;
qq                               ;
                                 ;
t readreal;                      ;
qq d−d1.9+2.14+0.19+e.29+1.33+17.39;
qq                               ;
e                                ; end e
d d=2d                           ;
s                                ;
```

```
b k=d,i=1,a30,b10,d10      ;
                           ;
d:   qq  3     , qq  320   ; comment 10.0;
                           ; store terminator:
a:   hs [a12]              ;    store char;
d1h: it (c37)  , pt [a14]  ;    term:=p+the char;
     pm  256   D       NTA ;    if full then m:=2↑38;
                           ; comment to force overflow on the next char;
a1:  ud  c37   , hs  c37   ; next: the char:=next in;
                           ; start:
a2:  ga  c37   , is (c37)  ;    part 1 of next in:= Raddr;
     bs  s511  t   520     ; d: if the char+p<1 ∧ the char+p>9 then
     hv  ra4            M ;     begin R40:=R41:=1; goto test point or zero end;
                           ; zero:
a3:  ck  10    , ml  c58   ;    M:=M×10+ integer(R shift 10);
     dl  c58   V        NZ ;    if −,overflow then
     qq 0      V       ITB ;     number:=true else
     hh  ra6   X           ;    begin M:= RM:10; goto overflow act end;
     hv  ra1          NZB ;
     gm  c17   , arnf c50  ;    if after point then
     mkf rd    , grf c50   ;     factor:=factor×10.0;
     pm  c17   , hv  ra1   ;    goto next;
                           ; test point or zero:
a4:  ca  p59           LQB ;    if real then
     hv  ra8           LQB ;     begin if the char=p+59 then goto point act end;
     ca  p16   , hhn ra3   ;    if the char=p+16 then cleargoto zero;
     ca  32    , hh  ra7   ;    if the char=32 then goto sign act;
d2:  nc[blind1], ca[blind2];    if the char+p=blind 1∨the char+p=blind 2 then
     ps  ra1−1 , hv (ra6)  ;     begin set return(next); goto store end;
d3:  nc[exit 1], ca[exit 2];    if the char+p=exit1∨the char+p=exit2 then
     is (ra6)  , hv  sd5   ;     goto exit1 or exit2 term;

     ca  58    , ppn 0     ;    if the char=58 then p:=R:=0;
     ca  60    , ppn 128   ;    if the char=60 then begin R:=0; p:=128 end;
     hv  ra1            LZ ;    if R=0 then goto next;
     mb  ra5   , ps  ra−1  ;    if bits(4,9,R)=63 then
a5:  ca  63    , hv  ra1   ;     goto next; set return(store terminator);
                           ;    goto store;
a6:  hv [a11]  , is (ra6)  ; overflow act:
     hv  sd6           NTA ;    if full then goto full term;
     hs (ra6)              ;    store;
     arn(c37)  D           ;    Raddr:= the char+p; goto d;
                           ; sign act:
a7:  hh  ra2   , hs (ra6)  ;    store;
     hh  ra6           NTA ;    if full then goto overflow act;
d4:  it  p−128 , pa [a18]  ;    sign:=p−128;
     hvn ra1           IZA ;    signed:=true; goto next;
                           ; point act:
a8:  hs (ra6)          LZB ;    if after point then store;
     hh  ra6           NTA ;    if full then goto overflow act;
     hvn ra1           IZB ;    after point:=true; goto next;
```

```
<e27 [BUFFER MODE: ]

a9:  ps  ra20  V       IRB ; exit1 or exit2 term: exit:=true;
                           ;    set return(out); goto store;
a10: ps  ra19              ; full term: set return(filled);
                           ; store:
a11: hvn ra15  X       NTB ;    if number then goto store number;
     hv  ra19          NZC ;    if -,(signed∨after point) then goto ask;
                           ; store char:
a12: arn c42   , ar  c51   ;    R:=n+1;
a13: gr  c51   , gr  c17   ; store chain word: n:=UV:=R;
a14: pt  c17 X V[term] LRA ;    if first then
     gr  c56 X V       IRA ;     begin chain start:=n; first:=false end else
                           ;    begin part 2 of UV:=term;
     arn c52   , ar  c17-1 ;      R:= array descr+chain; comment always negative;
     us  0              LT ;      us(array descr+chain)
     gmn c52   X           ;    end; chain:=R; M:=0;
     sr  c17-2         ITA ;    full:= n=N;
     hvn s1    X       NZC ;    if -,(signed∨after point) then return;
     qq (ra18) t   256 LZC ;    if signed∧after point then sign:=sign+256;
     it (ra18) t   160 LZA ;    term:= if signed then sign+160
     pt  ra14  t   59      ;           else 59;
     hv  ra18              ;    goto prep next;
                           ; store number:
a15: hv  ra16          NQB ;    if -,real then goto store integer;
     nkf 39    , dkf c50   ;    RF:=float(M)/factor;
     grf c17   , ncn(ra18) ;    if sign≠0 then
     srnf c17  , grf c17   ;     RF:=-RF;
     pm  c53               ;    UV:=RF;
     gm  c50   V           ;    factor:=1.0; goto count n;
                           ; store integer:
a16: mt  ra18  , gr  c17   ;    UV:= if sign≠0 then -M else M;
     arn c42   , ar  c51   ; count n:
     gr  c51   , ar  c17-1 ;    n:=n+1;
     us  0     , arn c51   ;    us(array descr+n);

a17: sr  c17-2         ITC ;    number:=full:= n=N;
a18: pan[sign] D X     IOC ; prep next: sign:=M:=0; signed:=after point:=false;
a19: hvn s1            LTA ; ask: if -,full then return;
                           ; filled:
     srn c42           NRC ;    if first∧-,exit then
a20: gr  c56           NRC ;     chain start:= -1;
     pm  c51   , arn c17-4 ; out:
     ga  c33   , gm (c33)  ;    store n;
     ca  c17   , us  0     ;
     hsn ra13          LRA ;    if -,first then store chain word;

     pm (c37)  D X         ;    R:= the char+p;
     ck  10    , gr  c54   ;    char:= integer (R shift 10);
a21: pm  c56   , gm  c17   ; finis: UV:=chain start;
a22: pp [savep], hh  c5    ;    p:=savep; goto exit std proc;
```

```
×    [CORE STORE MODE: ]

a9:  ps  ra20  V       IRB ; exit1 or exit2 term: exit:=true;
                           ;    set return(out); goto store;
a10: ps  ra19              ; full term: set return(filled);
                           ; store:
a11: hvn ra15  X       NTB ;    if number then goto store number;
     hv  ra19          NZC ;    if −,(signed∨after point) then goto ask;
                           ; store char:
a12: arn c42   , ar (c17−4);    R:=n:=n+1; M:=R;
     gr (c17−4)X           ; store chain word:
a13: arn c52   , ar  c17−1 ;    if first then
     ck  −10   , ga  ra14  ;     begin chain start:=n; first:=false end else
     gm (ra14) V       LRA ;    begin A[chain]:=n;
     gm  c56   V       IRA ;      part 2 of A[chain]:=term
a14: pt[A(c)]  t [term]    ;    end;
     gmn c52   X           ;    chain:=M; R:=M; M:=0;
     sr  c17−2         ITA ;    full:= n=N;
     hvn s1    X       NZC ;    if −,(signed∨after point) then return;
     qq (ra18) t   256 LZC ;    if signed∧after point then sign:=sign+256;
     it (ra18) t   160 LZA ;    term:= if signed then sign+160
     pt  ra14  t   59      ;           else 59;
     hv  ra18              ;    goto prep next;
                           ; store number:
a15: hv  ra16          NQB ;    if −,real then goto store integer;
     nkf 39    , dkf c50   ;    RF:=float(M)/factor;
     grf c17   , ncn(ra18) ;    if sign≠0 then
     srnf c17  , grf c17   ;    RF:=−RF; UV:=RF;
     pm  c53   , gm  c50   ;    factor:=1.0;
     pm  c17   V           ;    M:=UV; goto count n;
                           ; store integer:
a16: mt  ra18  X           ;    M:= if sign≠0 then −M else M;
     arn c42   , ar (c17−4); count n:
     gr (c17−4), ar  c17−1 ;    n:=n+1;
     ck −10    , ga  r1    ;    A[n]:=M;
     gm[A(n)]  , arn(c17−4);    R:=n;

a17: sr  c17−2         ITC ;    number:=full:= n=N;
a18: pan[sign] D X     IOC ; prep next: sign:=M:=0; signed:=after point:= false;
a19: hvn s1            LTA ; ask: if −,full then return;
                           ; filled:
     srn c42           NRC ;    if first∧−,exit then
a20: gr  c56           NRC ;     chain start:= −1;
     hsn ra13  X       LRA ; out: if −,first then store chain word;
     pm (c37)  D X         ;    R:= the char+p;
     ck  10    , gr  c54   ;    char:= integer(R shift 10);
a21: pm  c56   , gm  c17   ; finis: UV:=chain start;
a22: pp [savep], hh  c5    ;    p:=savep; goto exit std proc;
     qq                    ;    not used;

>    [END SLIP CONDITION e27]                          ;

d d5=a9−a11,d6=a10−a11,a9=a9−1,a11=a11−a9,a12=a12−a9     ;
d a14=a14−a9,a17=a17−a9,a18=a18−a9,a21=a21−a9,a22=a22−a9 ;
```

```
     gt  rb    , ps (c50)  ; entry read general:
     srn c42   , ar  s1    ;    R:=arrayword-1;
b:   ps (c50)  t [dor] IQB ;    real:=bit(41,arrayword)=1;
     ar  s2    , pm  s3    ;    array descr:=arrayword+const term -1;
     gr  c17-1         MA  ;    set bit(40,array descr);
     gm  c17-2 , pm  c17-3 ;    N:=length of array; M:= skipex;
     qq -1     , hs  c6    ;    rel track(-1); save p:=p;
     gp  sa22  , pp  s     ;    p:=trackplace for track 2;
     qq -1     , hs  c6    ;    rel track(-1); s:=trackplace for track 1;
     it  pa11  , pa  sa6   ;    set address of(store);
     it  pa12  , pa  sa    ;    set address of(store char);
     it  pa14  , pt  sd1   ;    set address of(term);
     it  pa18  , pt  sd4   ;    set address of(sign);
     srn c42   , gr  c56   ;    chain start:= -1;

<e27 [BUFFER MODE:]
     arn c17-4 , ga  c33   ;    take descr of actual n;
     ca  c17   , il  0     ;    if subscr variable then from buffer;
     arn(c33)  , gr  c51   ;    store value of n;

×    [CORE STORE MODE: ]
     arn(c17-4)V           ;
     qq                    ;    not used;
     qq                    ;    not used;
>
     hv  pa21          LT  ;    if n negative then goto finis;
     sr  c17-2         ITC ;    full:=number:=false;
<e27 [BUFFER MODE:]
     gr  c51   V       LZ  ;    if n=N then n:=0 else
×    [CORE STORE MODE]
     gr (c17-4)V       LZ  ;    if n=N then n:=0 else
>
     hv  pa21          NT  ;    if n greater N then goto finis;
     pp  s     , hs  rb5   ; comment this code unpacks the terminator word
     ck -10    , hs  rb4   ;    in M, one byte at a time starting with the
     ac  pd3   , hs  rb5   ;    right most, and generates the two instructions
     ck -10    , hs  rb4   ;    nc  ,ca  in the locations d2 and d3;
     ac  pd2   , grn c56   ;    chain start:=0;
     pm  c53   , gm  c50   ;    factor:=1.0;
     pin 0     X t 252     ;    first:=true; exit:=signed:=after point:=false; M:=0;
     arn c54   , ps  p     ; unpack char:
     tk  31    , tk -1     ;    R:= integer((char shift -10)∧ 10 1023);
     ga  rb3   , mb  rb2   ; comment R00:=R0;
     sc  rb3   , ud  sd4   ;    p:= 128×casebit of char; sign:=0;
b3:  pp [case] , hv  sa2   ;    Raddr:= char-p; goto start;
b4:  ar  rb7   , gr (s1)   ;
b5:  cln-10    , tk  2     ; comment
     ar  rb6           NT  ;    subroutine used by the code
     tk  1     , tk -3     ;    above for unpacking the bytes in M;
b2:  qq  895   , hv  s1    ;
b6:  qq  3.27              ; comment constat to get p-mark;
b7:  nc  0     , ca  0     ; comment constat to get test instructions;

e                               ; end a,b and d-names
t read general;                 ;
qq d-d1.9+3.14+2.19+0.33+33.39  ;
qq    7.9+5.14+10.19            ;
d d=3d                          ;
s                               ;
```

```
<e27                          ;    The following two tracks are
                              ;    only loaded if e27 is positive;
b k=d, i=1, a10, b10          ;
                              ; set case:
b:   ca  60.5  , pp  128      ;    if R= 60 pos 5 then p:=128;
     gr  c17−1 , hv  rb2      ;    case:=R; goto input;
b1:  hh  rb4           NOB    ; next: if case in then goto add case;
[1]  qq        V       NZA    ; term: if exit term then
     arn c58   , hh  rb4      ;    begin R:= 10 pos 39; goto add case end;
b2:  ud  c37   , hs  c37      ; input: next in;
b3:  nc[term 1], ca[term 2]   ; start: if the char+p=term 1 ∨ the char+p=term 2 then
     arn c17−2 , hv  rb9      ;    begin R:=exit terminators; goto shift term end;
     tk  4     , ca  0        ;    R:=R shift 4;
     sr  1.5   D V     LT     ;    if R pos 5 = 64 then R:=R − 1 pos 5
     ca  63.5  , hv  rb2      ;    else if R pos 5 = 63 then goto input;
     nc  58.5  , ca  60.5     ;    if R pos 5 = 58 ∨ R pos 5 = 60 then
     pp  0     , hv  rb       ;    begin p:=0; goto set case end;
     hv  rb2           NRA    ;    if prelude then goto input;
b4h: ck  6     , ar  c17−1    ;    R:=R shift 6;
                              ; add case: R:=R+case; R00:=0;
     ck  0     , nc (rb6)     ;    if old case≠case then
     ga  rb6   , ck  6        ;    begin old case:=case; R:=R shift 6 end;
b5:  bt  6 [i] t   −1  IOB    ;    i:=i−1; case in:=R00=R0;
                              ;    if i≠ −1 then begin
[1]  tl −6     , hv  rb1      ; shift: RM:=RM shift −6; goto next end;
     xr        , ck −3        ; store: swap; R:=R shift −3;
     ar  15.3  D              ;    R:=R + 15 pos 3;
     sr  5.3   D       LOC    ;    if −,case in ∧ exit term then R:=R − 5 pos 3;
     gr  c17   , arn c52      ;    UV:=R;
     ar  c42   , gr  c52      ;    R:=n:= n+1;
     ar  c50   , us  0        ;    us(UV,0,R+base array);
     arn c52   , sr  c51      ;    if (case in ∨ −,exit term) ∧ N≠n then
b6:  qq [old case] V   LOC    ;    begin i:=i+6; swap; M:=0; goto shift end;
     hvn(rb5) D V X 6  LT     ;    comment this instruction counts in b5, but
                              ;    jumps to b5+1 because it is D and V modified;
     pm  c52   , arn c17−4    ;    M:=n; R:=description(number);
     ga  rb6   , gm (rb6)     ;    store n at STORE[Raddr];
     ca  c17   , us  0        ;    if subscr var then us(UV,0,R);
     pm (c37)  D X            ;    R:= (the char+p) pos 9;
     ck  10    , gr  c54      ;    char:= R shift 10;
     grn c17   V       LOC    ;    if exit term ∧ −,case in then UV:=0 else
b7:  srn c42   , gr  c17      ; exit: UV:= −1;
b8:  pp[save p], hh  c5       ;    p:=save p; goto exit std proc;
                              ; shift term:
b9:  qq [first case]   M      ;    R40:=R41:=1;
     gr  rb3   V       NRA    ;    if prelude then terminators:=exit terminators
     pan c17−1 t 58.5  IZA    ;    else begin case:= 58 pos 5; exit term:=true end;
     hv  r1b1          IRA    ;    prelude:=false; goto term;
```

```
     gt  ra    , ps (c50)  ; entry read string:
a:   arn s1    , ps s[dor] ;   base array := array word + constant term − 1;
     ar  s2    , sr  c42   ;   comment done by rel track below;
     pm  s3    , gm  c51   ;   N:= length of A;
     qq  −1    , hs  c6    ;   rel track (−1); s:= trackplace;
     arn c17−4 , ga  ra1   ;   take descr of actual number;
     ca  c17   , il  0     ;   if subscr var then il(UV,0,R);
a1:  arn[n]    , gp  sb8   ;   R:= value n; save p:=p;
     hv  sb7           LT  ;   if R negative then goto exit;
     gr  c52   , sr  c51   ;   n:=R; R:=R−N;
     gr  c52   V       LZ  ;   if R=0 then n:=R else
     hv  sb7           NT  ;   if R not negative then goto exit;
     pp  s             IZB ;   p:= s; OB:= n = N;
     pm  c17−3 , hs  ra4   ;   comment this code unpacks the terminator−
     ck  −10   , hs  ra3   ;   word in M, one byte at a time starting
     ac  c17−2 , hs  ra4   ;   with the right most, and generates the two
     ck  −10   , hs  ra3   ;   instructions: nc[term 1], ca[term 2];
     ac  pb3   , arn c17−3 ;   OA:= exit term:= bits(1, 2, word) = 3
     ab  ra8   X           ;     ∧ bits(1, 2, word) = 3;
                           ;   comment now term 1 = skip 1 and term 2 = skip 2;
     pa  pb5   t    6  IZA ; unpack char:
     ps  p     , arn c54   ;   s:=p; i:=6;
     tk  30    , ga  sb9   ;   R:=bits(30,39,char) pos 9;
     mb  ra5   , sc  sb9   ;   p:= R∧ 10 128;
     pp (sb9)  , ga  c37   ;   R := part 1 of next in := R−p;
     pa  sb6   X  58.5 IRA ;   old case:=58 pos 5; prelude:=true; swap;
     grn c17−1 , arn c17−2 ;   STORE[UV−1]:=0; R:= exit terminators;
     bs  p     , it  60.5  ;   if p≠0 then case:= 60 pos 5 else
     pa  c17−1 t  58.5 M   ;   case:= 58 pos 5; R40:=R41:=1;
     hvn sb3   X       NOC ;   if −, OA ∧ −, OB then begin swap; M:= 0;
                           ;   goto start end;
     gr  sb3           IOA ;   terminators:=exit terminators; exit term:=false;
     pa  sb6           LOB ;   if OB then old case:= 0;
     hvn sb3   X       IRA ;   prelude:= false; swap; M:= 0; goto start;
a3:  ar  ra7   , gr (s1)  ;   comment
a4:  cln −10   , tk  2    ;   THIS SUBROUTINE IS USED
     ar  ra6          NT  ;   BY THE CODE ABOVE FOR
     tk  1     , tk  −3    ;   UNPACKING THE BYTES
a5:  qq  895   , hv  s1    ;   IN M;
a6:  qq  3.27              ;   comment constant to get p−mark;
a7:  nc  0     , ca  0     ;   comment instruction part of terminator test;
a8:  639/639/1023/1023     ;   comment mask used by: ab ra8 X;

e                              ; end a and b names;
t read string;                 ;
qq d−d1.9+2.14+1.19+0.33+33.39 ;
qq    7.9+5.14+10.19           ;
d d=2d                         ;
```

```
×                          ;    The following two tracks are
                           ;    only loaded if e27 is not positive;
b k=d, i=1, a10, b10       ;
                           ; set case:
b:   ca  60.5  , pp  128   ;    if R= 60 pos 5 then p:=128;
     gr  c17−1 , hv  rb2   ;    case:=R; goto input;
b1:  hh  rb4         NOB   ; next: if case in then goto add case;
[1]  qq       V      NZA   ; term: if exit term then
     arn c58   , hh  rb4   ;    begin R:= 10 pos 39; goto add case end;
b2:  ud  c37   , hs  c37   ; input: next in;
b3:  nc[term 1], ca[term 2]; start: if the char+p=term 1 ∨ the char+p=term 2 then
     arn c17−2 , hv  rb9   ;    begin R:= exit terminators; goto shift term end;
     tk  4     , ca  0     ;    R:=R shift 4;
     sr  1.5   D V    LT   ;    if R pos 5 = 64 then R:=R − 1 pos 5
     ca  63.5  , hv  rb2   ;    else if R pos 5 = 63 then goto input;
     nc  58.5  , ca  60.5  ;    if R pos 5 = 58 ∨ R pos 5 = 60 then
     pp  0     , hv  rb    ;    begin p:=0; goto set case end;
     hv  rb2         NRA   ;    if prelude then goto input;
b4h: ck  6     , ar  c17−1 ;    R:=R shift 6;
                           ; add case: R:=R+case; R00:=0;
     ck  0     , nc (rb6)  ;    if old case≠case then
     ga  rb6   , ck  6     ;    begin old case:=case; R:=R shift 6 end;
b5:  bt  6 [i] t   −1  IOB ;    i:=i−1; case in:=R00=R0;
                           ;    if i≠ −1 then begin
[1]  tl −6     , hv  rb1   ; shift: RM:=RM shift −6; goto next end;
     xr        , ck −3     ; store: swap; R:=R shift −3;
     ar  15.3  D           ;    R:=R + 15 pos 3;
     sr  5.3   D      LOC  ;    if −,case in ∧ exit term then R:=R − 5 pos 3;
a2:  gr[A(index)] t 1      ;    index:=index+1; A[index]:=R;
     arn c42   , ar (c17−4);    number:=number+1;
     gr (c17−4), sr  c51   ;    if (case in ∨ −,exit term) ∧ N≠n then
b6:  qq [old case] V   LOC ;    begin i:=i+6; swap; M:=0; goto shift end;
     hvn(rb5) D V X 6  LT  ;    comment this instruction counts in b5, but
                           ;    jumps to b5+1 because it is D and V modified;
     pm (c37)  D X         ;    R:= (the char+p) pos 9;
     ck  10    , gr  c54   ;    char:= R shift 10;
     grn c17   V      LOC  ;    if exit term ∧ −,case in then UV:=0 else
b7:  srn c42   , gr  c17   ; exit: UV:= −1;
b8:  pp[save p], hh  c5    ;    p:=save p; goto exit std proc;
                           ; shift term:
b9:  qq [first case]   M   ;    R40:=R41:=1;
     gr  rb3   V       NRA ;    if prelude then terminators:=exit terminators
     pan c17−1 t 58.5  IZA ;    else begin case:= 58 pos 5; exit term:=true end;
     hv  r1b1          IRA ;    prelude:=false; goto term;
     qq                    ;    not used;
     qq                    ;    not used;
     qq                    ;    not used;
     qq                    ;    not used;
```

```
     gt  ra    , ps (c50)  ; entry read string:
a:   arn s1    , ps s[dor] ;   base array := array word + constant term − 1;
     ar  s2    , sr  c42   ;   comment done by rel track below;
     pm  s3    , gm  c51   ;   N:= length of A;
     qq  −1    , hs  c6    ;   rel track (−1); s:= trackplace;
     arn(c17−4), gp  sb8   ;   save p := p;
     hv  sb7           LT  ;   if number negative then goto exit;
     sr  c51           IZB ;   OB:= N=n;
     gr (c17−4) V      LZ  ;   if number=N then number:=0
     hv  sb7           NT  ;   else if number not greater N then goto exit;
     arn(c17−4), ar  c50   ;   index:= base array+number;
     ck −10    , ga  sa2   ;
     pp  s                 ;   p:=s;
     pm  c17−3 , hs  ra4   ;   comment this code unpacks the terminator−
     ck  −10   , hs  ra3   ;   word in M, one byte at a time starting
     ac  c17−2 , hs  ra4   ;   with the right most, and generates the two
     ck  −10   , hs  ra3   ;   instructions: nc[term 1], ca[term 2];
     ac  pb3   , arn c17−3 ;   OA:= exit term:= bits(1, 2, word) = 3
     ab  ra8   X           ;     ∧ bits(1, 2, word) = 3;
                           ;   comment now term 1 = skip 1 and term 2 = skip 2;
     pa  pb5   t    6  IZA ; unpack char:
     ps  p     , arn c54   ;   s:=p; i:=6;
     tk  30    , ga  sb9   ;   R:=bits(30,39,char) pos 9;
     mb  ra5   , sc  sb9   ;   p:= R∧ 10 128;
     pp (sb9)  , ga  c37   ;   R := part 1 of next in := R−p;
     pa  sb6   X  58.5 IRA ;   old case:=58 pos 5; prelude:=true; swap;
     grn c17−1 , arn c17−2 ;   STORE[UV−1]:=0; R:= exit terminators;
     bs  p     , it  60.5  ;   if p≠0 then case:= 60 pos 5 else
     pa  c17−1 t  58.5 M   ;   case:= 58 pos 5; R40:=R41:=1;
     hvn sb3   X       NOC ;   if −, OA ∧ −, OB then begin swap; M:= 0;
                           ;   goto start end;
     gr  sb3           IOA ;   terminators:=exit terminatos; exit term:=false;
     pa  sb6           LOB ;   if OB then old case:= 0;
     hvn sb3   X       IRA ;   prelude:= false; swap; M:= 0; goto start;
a3:  ar  ra7   , gr (s1)  ;   comment
a4:  cln −10   , tk  2    ;   THIS SUBROUTINE IS USED
     ar  ra6          NT  ;   BY THE CODE ABOVE FOR
     tk  1     , tk  −3    ;   UNPACKING THE BYTES
a5:  qq  895   , hv  s1    ;   IN M;
a6:  qq  3.27              ;   comment constant to get p−mark;
a7:  nc  0     , ca  0     ;   comment instruction part of terminator test;
a8:  639/639/1023/1023     ;   comment mask used by: ab  ra8 X;

e                              ; end a and b names;
t read string;                 ;
qq d−d1.9+2.14+1.19+0.33+33.39 ;
qq    7.9+5.14+10.19           ;
d d=2d                         ;
>                              ; end codition e27 positive;
s
```

```
b k=d, i=11, a5, b2, d2    ;

[bits 0-9 of the first 1  words are used as digit table]
d:  qq  0     , gp  ra5    ;     save p := p;
    qq  0     , pa  rb1    ;     sign:=0;
    qq  0     , tk  14     ; unpack layout:
    qq  0     , mb  ra2    ;     R:= (R shift 14)∧(10 15 11 31)
    qq  0     , gr  c17-1  ;     UV[-1]:=b;
    qq  0     , tk  10     ;
    qq  0     , gr  c17-2  ;     UV[-2]:=d;
    qq  0     , ck  10     ;
    qq  0              IOB ;     OB:= n=1;
    qq  0     , ps  rd-1   ;
    qq  0     , gs  rd2    ;
    qq  0     , gs  ra4    ;
    qq  0     , arn c17-4  ;     R:=N;
                           ;
    pa  rb1   t   32    LT ;     if R<0 then sign:=32;
    ann c17-4 , tl -39     ;     RM:=abs N; comment tl-39 to save overflow;
a:  pp  13    , dl  c58    ;     for i:=1,i+1 while M≠0 do
    ps  s1    X        IZA ;      begin R:=RM:10; M:=RM rem 10;
    ck -10    , ga  s      ;       table[i]:= R shift -10; R:=0
    hhn ra             NZA ;      end;
                           ;     for p:=13 step -1 until 1 do
    arn c50   , tl -39     ;      begin
b:  hv  rb2   , pp  p-1    ;      if p=13 then begin
    bs (ra4)  , hv  ra2    ;      M:=picture; R:=if picture<0 then -1 else 0;
    it (c17-2), bs  p-1    ;      goto first end;
    bs  p-1   , hh  ra3    ;      if table[p]≠0 ∨ p≤d+1 ∨ p=1 then
                           ;       begin
a2: pa  15 D  V   15   LOB ;        if -,OB then
b1: qq [sign] , hs  c39    ;         next out(sign); OB:=true;
    pi  1.1   , it (c17-2) ;        if p≤d then
    bs  p     , hv  ra3    ;         begin
    qq  59    , hs  c39    ;          next out(59);
    pa  c17-2 , tln 1      ;          d:=0; R:=0; RM:=RM shift 1
a3: gp  c17-1 , it (c17-1) ;         end; b:=p
    bs  p     , hh  rb     ;       end;
                           ;      if p≤b then
    tln 1     , ca (ra4)   ;       begin R:=0; RM:=RM shift 1;
    pa (rd2)  t   16   LOB ;        if table[p]=0∧OB then table[p]:=16;
a4: qq (p0)   , hs  c39    ; space:  next out(table[p]);
d2: pa  p0    , bs  p510   ;        table[p]:=0;
a5: pp  0     , hh  c5     ; first: if R≠0 then
                           ;         begin
b2: hh  rb              LZ ;          R:=if p<13 then 0 else RM shift 1;
    bs  p499  , hhn ra4    ;          goto space end
    tln 1     , hh  ra4    ;       end
                           ;     end; goto exit std proc;
e                          ;

t write integer;                   ;
qq d-d1.9+1.14+0.19+0.29+0.33+36.39;
qq    2.9+5.14                     ;
d d=d+1                            ;
```

```
[write(layout,number,...,number):
 writetext(word): ]

b b1                          ;
b k=d, i=11                   ;
b a52                         ;

a0: arn  c35   , ga   ra1     ; write:
a1: ud         , hs   c8      ;     UA:=adr(layout);
    pm  (c30)  , gm  (c35)    ;
a2: ps  (c35)  , ps   s1      ; next param: s:=last used+1;
    gs   ra3   , arn  s       ;     if return information then
    ck   20    , ca   40      ;      goto address expr
    gs   c35   , hvn  c19     ;
a3: ud         , hs   c8      ;     UA:=adr(number);
    qq   1     , hs   c6      ;     call track 2
    gp   ra3   , pp   s1      ;     save p
    qq   1     , hs   c6      ;     call track 3
    it   s1    , pa   pa27    ;     start address
    gp   ra9   , pp   s1      ;     start addr. track 2
    it   ra2   , pa   pa41    ;     return address
    qq   1     , hs   c6      ;     call track 4
    it   s1    , pa   pa42    ;     start address
    qq   −3    , hsn  c6      ;     call this track and clean c50
    ps  (c35)         t1      ;     last used:=last used+1
    grn  c17−3 , pp   256     ;     clean working loc.; minexp:=256;
    pm  (c30)  , arn  s       ;     if floating point number then
    arnf(c30)   XV        LB  ;     begin M:=mantissa; R:=exp2 end else
    arn  28     D             ;      begin M:=number; R:=28 end;
    gr   c17   , gm   c52     ;     exp2; x1
    pan  c33    X     t509    ;     exppart:=false; M:=0;
    arn  s−1   , gr   s       ;
    cl   −20   , ck   20      ; unpack layout:
a6: hv   ra7    X         NO  ;
    qq   0     , hs   c39     ; print spaces before number:
    ck   1     , hv   ra6     ;
a7: gm   c17−1 , ck   −5      ;     store picture
    ga   c30   , tk   10      ;     b
a8: ck   −6    , ga   c17−3   ; expprinting: h
    tk   10    , ck   −8      ;
    gr   c17−4 , tk   10      ;     f1
    ck   −6    , ga   c50     ;     d
    tk   11              ITA  ;     TA:=p=1
    ck   −7    , gr   c51     ;     bE+f2
    pp   0                LZ  ;     if bE=0 then minexp:=0
    ps  (c17)       t10       ;     s:=exp2+10;
a9: hv   0                    ;     goto next track;

d a10=a8−a2,  a4=a3−a2        ;
```

```
a11:ca   1     , pp   10     ;     if bE=1 then minexp:=10
    ca   2     , pp   100    ;     if bE=2 then minexp:=100
    grn  c17-2 , pa   c37    ;     H:=exp10:=0;
a12:ann  c52   , nk   r1     ;     R:=abs(x1); value to be printed =
    ps   s0    , gr   c56    ;      x=x1×2↑exp2
    hv   ra18            LZ ;     if x1=0 then compute b1
    pm   c56   , bs   s1     ; conversion:
    mkn  ra25  , hh   ra13   ;     begin comment
    tk   s1    , gr   c56    ;      by multiplication by
    ps   s7    , sr   ra24   ;      2↑3/10 or 10/2↑4
    mkn  320    DV        LT ;      x is converted to form
a13:hv   ra15  , it   1      ;      x=x2×10↑H where 1>x2≥.1
    qq  (c17-2)       t-1    ;      x2 is stored in c56
a14:ps   s-3   , hh   ra12   ;     end conversion;
a15:arn (c30)   D            ; compute exp10:
    sr   c50   , ga   ra16   ;     a16:=b-d; R:=b-d-h-1;
    sr    1     D            ;
    sr   c17-3 , it  (c17-3) ; L1: if H>h then
    bs  (c17-2), hs   ra20   ;      begin Hh:=true; goto increxp10 end;
    mt   ra14  , it  (c17-2) ;     R:=-R;
                             ; L2: if H<b-d then
a16:bs   0     , hs   ra21   ;      begin Hh:=false; goto decrexp10 end;
a17:arn  ra16  , ar   c50    ;     R:=b-d;
                             ; L3: R:=R+d;
a18:ga   c17   , ps  (c17)   ;     b1:=R
    arn  256    D         NT ; rounding:
a19:bs   s511  , hh   ra22   ;     if b1≥0 then
    xr         , mkn  ra26   ;      R:=.5×.1↑b1;
    ps   s-1   , hv   ra19   ;     goto roundx2
a20:bs  (c51)  , hv   ra21   ; increxp10: if bE>0 then goto decrexp10
    bs  (c33)                ;     if -,exppart then
    srn (c51)   DV    t1     ;      begin R:=-1; bE:=1 end
    itn (c17-2), pa   c17-3  ;     else begin h:=H; R:=0 end;
a21:sc   c37   , bs  (c37)   ; decrexp10: exp10:=exp10-R;
    ac   c17-2 , hh   s-1    ;     if exp10+minexp>0 then
    ac   c37   , arn  c17-2  ;      begin H:=H+R; goto if Hh then L1 else L2 end;
a22:hh   ra17  , ar   c56    ;     exp10:=exp10+R; R:=H; goto L3;
                             ; roundx2:
a23:hv   0      X         NO ;     R:=R+x2; M:=R; goto next track;
    ps   -1    , hh   ra12   ;     if overflow then goto conversion
a24:vy   p51   , mln (204)   ;     .1+epsilon
a25:can  s409  , cm  (r-410) ;     2↑3/10
a26:vy   p51   , mln (s204)  ;     .1-epsilon

d a27=a23-a11                ;
```

```
a28:pp  (c17−4), ps  (ra38)  ;     p:=f1
    bs  (c17)  , hv   ra29   ;     if b1≤0 then
    it  (c50)  , pa   c17    ;      begin b1:=d;
    grn  c52   X             ;        M:=x:=exp10:=0
a34:qq   59    , pa   c37    ;      end;
a29:pa   ra35  , bs  (c17−2) ;    lead zero:=0;
    srn  c17−2 , hh   ra30   ;    if H>0  then begin R:=−H; goto on end;
    bsn (c33)                ;    R:= if −,exppart∧NTA then
    sr  −1     D         NTA ;    1 else 0;
    bs  (c17−3), ga   ra35   ;    if −,exppart then lead zero:=Raddr;
a30:mt   r−2   , ar   c17−3  ; on:R:= −R+h;
    ga   c17−3 V         NTA ;    if NTA then h:=Raddr else
    ga   ra35  , pa   c17−3  ;     begin lead zero:=h; h:=0 end;
    bs   p509  , hv   ra33   ;     if f1<3 then goto print leading spaces
a31:arn (c33)   D     t15    ; print ten and sign:
    hs   s               LT  ;     if exppart then print ten
    pp   p10   , arn  c52    ;     p:=p+10; R:=x;
    arn −480    DV        NT ;     plus
    arn  32     DV           ;     minus
    bs   p500  , ck   10     ;     if p<12 then space for plus
    ca   p−10  , hh   ra32   ;     if p=10∧x≥0 then skip sign
a32:hs   s     , pp   4      ;      else print sign; sign printed:=true;
a33:bt  (c17−3)       t−1    ; print leading spaces:
    hsn  s7    , hv   ra33   ;     count h
    bs   p509  , hv   ra31   ;     print sign if −,sign printed
a35:bt   0             t−1    ; print leading zeros:
a36:hsn  s6    , hv   ra35   ;     count leading zeros
    bt  (c17−2)       t−1    ; print digits before point:
a37:hsn  s4    , hv   r−1    ;     count H
    arn  ra34  , bs  (c50)   ;     if d>0 then
a38:hs   0     , arn  c58    ;      print point
    bt  (c50)         t−1    ; print decimals:
a39:hs   s2    , hh   ra38   ;     count d
a40:ps   0     , arn  c51    ;     s:=adr.on first track; R:=bE+f2
    pa   c17   V      t−2 NZ ;     exp2:=−2;
    pp  (sa4)  , hv   s      ;     if R=0 then restore p, goto next param
    ga   c30   , pm   c37    ;     b:=bE;
    gm   c52   , hh   sa10   ;     x1:=exp10; goto expprinting
    qq                       ;     not used;
    qq                       ;     not used;

d  a41=a40−a28, a42=a38−a28  ;
```

```
b:
a43:bs  (c17)  , hv   ra47   ;
    hvn  ra48                ;     if b1<0 then output space;
a44:it  (c17−2)       t1     ; output decimals: count H
a45:bs   0     , hvn  ra43   ;     if H<0 then output 0
a46:bt  (c17)         t−1    ;     count b1
    mln  c58   , tk   30     ;     next digit to R
a47:ar   16     D         LZ ;     zero instead of space
a48:gs  ra52   , ga   ra51   ;     return:=s;  next:= Raddr;
    bs (ra49)  , mt   ra53   ;     if R0 = bit (0, case) then
    hv  ra50              NT ;      begin
a49:nt  570    , qq   630    ;      case:= 630−case;
    qq (ra49)  , hs   c39    ;       next out(case) end;
a50:bsn p507   , arn c17−1   ;     R:=0;  if p < 5 then R:= picture;
a53:qq  −1     V          NT ;     if R0 = 1 then
    qq   0     , hs  c39     ;       next out (0);
a51:qq [next]  , hs  c39     ;     next out (next);
    ac  c17−1  , ps ra43     ;     picture:= picture+R;  s:= first entry;
a52:hhn  [return]            ;     R:= 0;  go right (return);
e    [block for write]       ;

ba6  [write text]
a1: hv  ra2           LZA ; next word: if drum string then goto drum word;
b1: it (c35)  , pa  r1    ; write text:
[1] ud [param], hs  c8    ;   take formal(text param);
    arn (c30) , pm (c30)  ;   M:= value;
    mb  r1    V       IZA ;   drum string := value ∧ mask = 0;
[1] qqf −1.13+1.19−1.29   ;   if −, drum string then
    hhn ra3   X       NZA ;     begin R:= M;  M:= 0;  goto write it  end;
    hs  c63   X           ;   R:= M; get track (string descr);
a2: arn  c65  , ca  s−40  ; drum word:
    qq   −1   , hs   c6   ;   if last word then get rel track (−1);
    pan (c65) X   512     ;   priority for current track:=  −512;  M:= 0;
    arn  s1   , ps  s1    ;   R:= string word;  s:= s+1;  save s:= s;
a3h:gs  ra6   , cl  34    ; write it:  RM:= RM shift 34;
    ck  −4    , ga  ra4   ;   R:= R shift −4;  out:= Raddr;
    ca  10    , hvn ra5   ;   if out=10 then goto exit;
    ca  15    , hvn ra1   ;   if out=15 then goto next word;
    ca  63    , it  1     ;   if out=63 then out:=64;
a4: xrn[out]  , hs  c39   ;   R:=M; M:=0; next out(out);
a5: qq (c35)  V   1   LZ  ;   if R≠0 then
a6: ps[save s], hh  ra3   ;    begin s:= save s; goto write it;
    hv  c19               ;   lastused:=lastused+1; goto addr expr;
e                         ;   end writetext;
    qq                    ;   not used;

e                         ;   end drumblock;

d b1=b1−b                          ;
twrite;                            ;
qq d−d1.9+4.14+0.19+ 0.29+3.33+44.39;
qq    4.9+5.14                     ;
d d=d+3                            ;
twritetext;                        ;
qq d−d1.9+1.14+0.19+b1.29+2.33+40.39;
qq   11.9                          ;
d d=d+1                            ;
e                                  ;
s                                  ;
```

```
[cos(x), sin(x) − algorithm as described in TRIG−2 −
 sqrt(x) − Newton iteration]

 b b1                          ;
 b k=d, i=11, a12              ;

a: qq   −1    VX         ITA ; cos:cos:=true; skip next;
a1:qq          X         ITA ; sin:cos:=false;
   ga   r1                   ;
   bs           X     t−16   ;     if arg<2↑(−15)∧arg≠0 then
   dkf  ra8   , hv    ra4    ;      begin
a2:arnf c53               LTA ; small: if cos then RF:=1;
a3:grf  c17   , hh    c5     ; out:   UV:=RF; goto exit std proc;
                              ;      end;
a4:hv   ra2                LZ ;     if RF=0 then goto small;
   tkf  10                    ;     R:=modulus(RF);
   ar   128  D             LTA ;     if −,cos then R:=R+0.25;
   tk   2     , gr    c50     ;     t:=4×R;
   pm   c50   , pt    ra6     ;     M:=t; a6T:=0;
   pt   ra6           t−3  NO ;     if overflow then a6T:=−3;
   srn  256  D                ;     R:=−0.5;
   mk   c50   , gr    c51     ;     w:=t↑2−0.5;
   pan  r1            ta9     ;     R:=0;
a5:ar   r0     X     t1      ;     for i:=6 step −1 until 0 do
   arn  c50   V            LA ;      R:=(R+a[i])×w;
   mkn  c51   , hv    ra5     ;
a6:mk   c50   , mt    r       ;      R:=t+r×R;
   nkf  0     , grf   c17     ;      RF:=R;
   hh   c5                    ;      goto exit std proc;
a7:1023/1022/ 205/ 680        ;      a[5]=−0.000 003 431 618;
      0/  79/ 525/ 728        ;      a[4]= 0.000 151 659 755;
   1021/ 781/   4/ 140        ;      a[3]=−0.004 369 728 013;
     37/ 335/ 871/ 857        ;      a[2]= 0.072 906 210 715;
    732/ 319/ 201/ 597        ;      a[1]=−0.569 703 680 308;
    136/ 805/ 921/ 102a       ;      a[0]= 0.267 162 131 329a;
a8:   2/ 402/ 126/ 852        ;      2×pi;
d a9=a7−a5−1
d b=a1−a
a10:grf  c17  , grf  c51      ; sqrt:UV:=x; work:=x;
    hh   c5                LZ ;     if x=0 then goto RS;
    pa   ra12 VX           NT ;     if x>0 then a12:=0
    pt   c49−7, hs   c27      ;      else alarm(↯<sqrt↯);
    tk   −1   , ga   c51      ;     work:=entier(exponent(x)/2);
a11:arnf c17  , dkf  c51      ;     for i:=1 step 1 until 5 do
    arf  c51   X              ;       work:=(UV/work+work)/2;
    sr   1   D X              ;
a12:bt   0           t−128    ;
    grf  c17  , hh   c5       ;     UV:=work; goto RS;
    grf  c51  , hv   ra11     ;
d b1=a10−a0                   ;
e                             ;
                              ;
t cos;                              ;
qq d−d1.9+1.14+0.19+ 0.29+0.33+34.39;
qq    3.9                           ;
t sin;                              ;
qq d−d1.9+1.14+0.19+ b.29+0.33+34.39;
qq    3.9                           ;
t sqrt;                             ;
qq d−d1.9+1.14+0.19+b1.29+0.33+34.39;
qq    3.9                           ;
e                                   ;
d d=1d                              ;
```

```
[arctan(x): the method is that described by Lance in Numerical Methods
page 40, supplemented close to the origin by the two first terms of
a modified Taylor expansion]

b b1                                ;
b k=d, i=11, a11                    ;
                                    ;
b:  hh   ra4                  LZ ;      if arg =0 then goto out
    grf  c17    X                 ;      UV:=arg;
    ga   ra    VX             LT ;      if exponent≥0 then
    srnf c53   , hv   ra1        ;       begin RF:=−1; goto big arg end;
a:  bs   0             t−8       ;      if exponent>−8 then
    hv   ra2                     ;       goto compute
    mkf  c17   , mkf  c17        ;      UV:=−arg↑3×0.333306+arg;
    mkf  ra9   , arf  c17        ;
    grf  c17   , hh   c5         ;      goto exit std proc
                                 ; big arg:
a1: dkf  c17   , it   a11        ;      base:=−3×pi/16;  arg:=−1/arg;
a2: pa   ra3           ta10      ; compute: base:=pi/16;
    ga   ra    , tkf  9          ;      a:=sign(arg);
    gr   c52   , snn  c52        ;      work1:=arg/2;
    pm   128  D X                ;
    mk   ra6    , gr   c17       ;      UV:=0.25+(sqrt(2)−1)/4×abs(arg);
    sr   256  D                  ;
    an   c52   , tk   −1         ;
    dk   c17                     ;      UV:=(UV−0.5+abs(arg)/2)/2×UV;
    gr   c17    X                ;
    mkn  c17   , gr   c51        ;      work2:=UV↑2;
    pan  r1            t5        ;
    ar   r0     X      t1        ;      R:=UV×polynomium;
    mkn  c17   V               LA ;
    mkn  c51   , hv   r−2        ;
a3: ar   r0    , mt   ra         ;      R:=(base+R)×sign(a);
a4: nkf  1     , grf  c17        ;      UV:=2×R;
a5: hh   c5                      ; out: goto exit std proc
       1/   8/ 730/ 199          ;      a[13]= 0.001 969 743 882
    1021/ 305/  10/ 740          ;      a[11]=−0.005 277 613 694
       5/ 189/ 815/ 724          ;      a[9] = 0.010 127 633 264
    1014/ 216/  81/ 547          ;      a[7] =−0.019 119 110 826
      19/1000/ 777/ 190          ;      a[5] = 0.039 018 171 254
     975/ 496/   3/ 827          ;      a[3] =−0.094 757 072 986
     212/  79/ 204/ 983a         ;      a[1] = 0.414 213 562 309
a6:  917/ 984/ 409/ 515          ;      −(sqrt(2)−1)/2=−0.207 106 781 184
a7:  722/ 416/ 896/ 819          ;      −3×pi/16 = −0.589 048 622 548
a8:  100/ 543/ 725/  68          ;      pi/16 = 0.196 349 540 849
a9: f −0.333 306                 ;      modified Taylor coefficient

b1: tk   −29   , gr   c17        ; sign: sign:=if argument<0 then −1
    hh   c5                      ;      else if argument>0 then 1 else 0;
                                 ;      goto exit std proc;

d a10=a8−a3, a11=a7−a3           ;
e                                ;

t arctan;                              ;
qq d−d1.9+1.14+0.19+0.29+0.33+34.39    ;
qq    3.9                              ;
d b1=b1−b                              ;
t sign;                                ;
qq d−d1.9+1.14+0.19+b1.29+0.33+33.39   ;
qq    3.9                              ;
e                                      ;
d d=1d                                 ;
s                                      ;
```

```
b k=d,i=11,a4,e            ;
                           ; entry ln(x):
     tk  9     V       NT  ;   if w≥0 then w:=mantissa(x)/4
     pt  c49-6 , hs  c27   ;   else alarm;
     gr  c50   V X     NZ  ;   if w≠0 then z:=w
     arnf ra2  , hv  ra    ;   else begin ln:= -2↑166; goto exit end;
     ga  ra1   , gm  c51   ;   exp2:=exponent(x)×2↑(-9)+2↑(-10);
     arn ra4   , ac  c50   ;   z:=sqrt(2)/4-w)/
     sr  c51   , dk  c50   ;      sqrt(2)/4+w);
     gr  c50   X           ;
     mkn c50   , gr  c51   ;   z2:=z↑2;
     it  ra4   , pan r1    ;   R:=0;
     ar  0     X t   1     ;   for k:= a[7],a[5],a[3] do
     mkn c50   V       LA  ;
     mkn c51   , hv  r-2   ;   R:=(R+k)×z2;
     ar  ra1   X           ;   ln:= ((R+a[1]×z+exp2)
     mkn ra3               ;        ×ln(2)
     nkf 9                 ;        ×2↑9;
a:   grf c17   , hh  c5    ; exit: goto exit std proc;
a1:  qq [exp2] t   512     ;
a2:  qq  165   t   512     ;
a3:   354/912/766/1001     ; ln(2)          (= 0.6931 4718 056)
a4:   181/ 19/819/254      ; sqrt(2)/4      (= 0.3535 5339 059)
     1023/579/325/663      ; a[7]           (=-0.4342 5975 1292×2↑(-9))
     1023/433/591/509      ; a[5]           (=-0.5765 8334 2056×2↑(-9))
     1023/ 39/118/823      ; a[3]           (=-0.9618 0076 2286×2↑(-9))
     1021/117/369/224    a ; a[1]           (=-2.8853 9007 2738×2↑(-9))

e                          ;

t ln;                      ;
qq d-d1.9+1.14+34.39       ;
qq    3.9                  ;
d d=1d                     ;
s                          ;
```

```
b b1                              ;
b k=d, i=11, a9                   ;

b: grf  c17   V            NZ ; exp: UV:=x;
   arnf c53   , hv   ra2       ;    if x=0 then RF:=1; goto out;
   annf c17   , srf  ra4       ;    R:=abs(x)−511×ln(2);
   hv   ra3                 NT ;
   arnf c17    X               ;    R9M:=x/ln(2);
   ga   ra    , mln  ra5       ;    a1:=entier(R9M);
a: tl   0              t3       ;
   ga   ra1   , tl   10        ;
   cl   −2                     ;    R:=(R9M−a1)/2;
   sr   128   DX               ;    M:=R−0.25;
   gm   c17   , mkn  c17       ;    UV:=M;  R:=UV↑2;
   pm   384   DX               ;    R:=0.75; M:=UV↑2;
   mk   ra7   , gr   c51       ;    work1:=0.75×(1+b×UV↑2);
   gr   c52   , arn  ra6       ;
   mk   ra8    X               ;
   mkn  c17   , sc   c52       ;    work2:=work1−0.75×(a×UV+c×UV↑3);
   ar   c51    X               ;
   mln  ra9   , dl   c52       ;    R:=work1/work2/sqrt(2);
a1:nkf  0              t1       ;    RF:=R×2↑(a2+1);
a2:grf  c17   , hh   c5         ; out:goto exit std proc;
a3:arnf c17                     ; large:
   hvn  r−2    X            LT ;     if arg<0 then RF:=0; goto out;
   pt   c49−8 , hs   c27       ;     if arg>0 then alarm(≠<exp≠);
a4:f 354.1982                   ;     511×ln(2)
a5: 369/ 337/ 869/ 174          ;     log base2(e)/2;
a6: 266/ 172/ 574/ 725          ;     0.75×a=0.519 860 384 44;
a7:  73/ 797/ 660/  37          ;     0.75×b=0.144 099 511 27;
a8:   8/ 524/ 902/ 710          ;     0.75×c=0.016 626 132 08;
a9: 362/  39/ 614/ 509          ;     sqrt(2)/2=0.707 106 781 186;

b1:pmn  1023   X                ; checksum:  M:=0;
   cl   −20   , gr   c17        ;     checksum:=loc(1023) shift −20∧(20−1);
   arn  c50   , ck   20         ;     loc(1023):=saveR shift 20;
   gr   1023  , hh   c5         ;     goto exit std proc;

   qq                           ; not used;
   qq                           ; not used;
   qq                           ; not used;
   qq                           ; not used;
   qq                           ; not used;
   qq                           ; not used;
   qq                           ; not used;
e                               ; end drumblock;
d b1=b1−b                            ;

t exp;                               ;
qq d−d1.9+1.14+0.19+ 0.29+0.33+34.39;
qq    3.9                            ;

t checksum;                          ;
qq d−d1.9+1.14+0.19+b1.29+0.33+33.39;
qq    2.9                            ;

d d=d+1                              ;
e                                    ;
s                                    ;
```

```
[25.8.1967                                                    put(A,Area,place)
                                                              get(A,Area,place)
                                                                 BUFFER VERSION
                                                          DISC VERSION, track 1]

b e2                        ;
d e2=0                      ;
<e18,e2=1>                  ;
<-e18,e2=1>                 ;

<e27, [BUFFER VERSION, DISC VERSION track 1]
b k=d,i=1,a7,b3             ; comment only loaded if e27 is positive;
    pm  rb2   V       IPB ; entry get: get blocks:=true; goto drum;
    pm  rb3           IPB ; entry put: get blocks:=false;
e1: gt  ra    X           ; drum: dor:=Rincr; swap;
    ab  c17-3 , ps  (c50) ;    if (-,get blocks ∧ bit(5,Area)=0)
b2: nc  127   , hv  ra4   ;      ∨ bits(0,2,Area)≠0 then alarm;
a:  arn s1    , ps  s[dor];
    ar  s2    , gr  c17   ;    UV:=arrayword+constant term;
    arn s3    , ud  rb1   ;    M:=length of A -1;
    arn s3    , sr  ra5   ;    if length of A less 40 then
    dln ra5   V X     NT  ;    alarm
    pt  c49-11, hs  c27   ;    else tracks:=M:40;
    ar  c42   , gr  c17-2 ;    remainder:=M mod 40 +1;
    gm  c17-1 , pm  c17-3 ;    M:=Area;
    arn c42   , hs  c63   ;    reserve trackplace(1);
    ps  s1    , gs  c17   ;    part 1 of UV:=trackplace+1;
    gs  ra3   , tln 7     ;    start:=trackplace+1;
    arn c17-4 , ud  rb1   ;    TA:=(place-1) less 0; swap;
    tl -23    , sr  c17-1 ;    TB:=(no of tracks in area-tracks
    sr  c17-4         ITB ;       -place) less 0;
    pt  c17   V  40   NTC ;    if NTC then part 2 of UV:=40
a4: pt  c49-11, hs  c27   ;    else alarm;
    tln 16    , ar  c17-4 ;    M:=first track in area+place-1;
    ps  c17-2 , ud  rb1   ;    s:=address of remainder; comment now UV holds
                          ;    the parameterword for the instructions il/us;
    dln ra6   , ar  ra6   ;    group:=M:960+960;
    ck -10    , ga  ra2   ;    track:=M mod 960;
    cl -10    , ga  ra1   ;    R:=tracks; comment tracks to be transferred-1;
    pm  c17   , arn c17-1 ;    M:=UV; comment parameterword for il/us;
                          ; next track:
a1: is [track], ca  s-960 ;    if track=960 then
    pa  ra1   , it  1     ;     begin track:=0; group:=group+1 end;
a2: vk [group], vk (ra1)  ;    select(group); select(track);
b1: sr  c42   X       ITA ;    R:=R-1; TA:=R less 0; swap;
a3: lk [start]V       NPB ;    if get blocks then from drum(track,start)
    il  0     , sk (ra3)  ;    else begin il(R); to drum(track,start) end;
    vk (ra1)  , us  0     ;    wait drum; us(R);
    ar  s     , ps  ra5   ;    R:=R+cell[s]; s:=address of 40;
    hv (ra1)  X D 1   NTA ;    if -,TA then begin track:=track+1; swap;
                          ;    goto next track end; value:=0;
    grn c17   , hh  c5    ;    UV:=0; goto exit std proc;
a5: qq  40.39             ;
a6: qq 960.39             ;
b3: qqf 111               ;
e                         ;

<-e2+1                    ;   only loaded if not disc
t put;                      ;
qq d-d1.9+ 1.14+1.29+0.33+33.39 ;
qq    2.9+2.14+10.19        ;

t get;                      ;
qq d-d1.9+ 1.14+0.29+0.33+33.39 ;
qq    2.9+2.14+10.19        ;
d d=1d                      ;
>                           ; end BUFFER VERSION
```

```
<−e27+1 [CORE STORE VERSION]
b k=d,i=10,a7,b1,e        ; comment only loaded if e27 is 0 or negative;
    pmn 127   DVX     IZA ; entry get: put tracks:=false; goto test;
    pm  111   D X     IZA ; entry put: put tracks:=true;
    ab  c17−3 , ps (c50)  ; test:
    nc  127   , hv  ra4   ;    if (put tracks ∧ bit(5,Area)=0)
    arn c50   , gt  ra    ;      ∨ bits(0,2,Area)≠0 then alarm; dor:=part 1 save R;
a:  arn s1    , ps  s[dor];    put tracks:= R≠0;
    ar  s2    , ck −10    ;    start:=arrayword+constant term;
    ga  re    , pm  s3    ;    M:=length of A;
    arn s3    , ud  rb1   ;    R:=length of A −1; swap;
e:  ps [start], sr  ra5   ;    if R less 40 then
    dln ra5   V X      NT ;    alarm
    pt  c49−11, hs  c27   ;    else begin R:=M mod 40; M:=M:40 end;
    ar  c42   , ck −10    ;    rem:=R+1;
    ga  ra3   , gm  c17−1 ;    tracks:=M;
    pm  c17−3 , tln 7     ;    M:=Area;
    arn c17−4 , ud  rb1   ;    TA:=(place−1) less 0; swap;
    tl −23    , sr  c17−1 ;    TB:=(no of tracks in area−tracks
    sr  c17−4         ITB ;        −place) less 0;
    tln 16    V       NTC ;    if NTC then R:=first track in area
a4: pt  c49−11, hs  c27   ;    else alarm;
    ar  c17−4 , ud  rb1   ;    M:=R+place−1;
    dln ra6   , ar  ra6   ;    group:=M:960+960;
    ck −10    , ga  ra2   ;
    cl −10    , ga  ra1   ;    track:=M mod 960;
    arn c17−1             ;    R:=tracks; comment no of tracks to be transferred−1;
                          ; next track:
a1: is [track],ca  s−960  ;    if track=960 then
    pa  ra1   , it  1     ;     begin track:=0; group:=group+1 end;
a2: vk [group], vk (ra1)  ;    select(group); select(track);
    sk  s     V       NZA ;    if put tracks then to drum(track,start)
    lk  s                 ;    else from drum(track,start);
a3: ps  s[rem], vk (ra1)  ;    start:=start+rem; wait drum;
b1: sr  c42   X       ITA ;    R:=R−1; TA:=R less 0; swap;
    pa  ra3   X   40      ;    rem:=40; swap;
    hv (ra1)  D   1   NTA ;    if −,TA then begin track:=track+1;
                          ;    goto next track end; value:=0;
    grn c17   , hh  c5    ;    UV:=0; goto exit std proc;
a5: qq  40.39             ;
a6: qq 960.39             ;
    qq                    ;    not used;
    qq                    ;    not used;
    qq                    ;    not used;
e                         ;

t put;                          ;
qq d−d1.9+ 1.14+1.29+0.33+33.39 ;
qq    2.9+ 2.14+10.19           ;

t get;                          ;
qq d−d1.9+ 1.14+0.29+0.33+33.39 ;
qq    2.9+ 2.14+10.19           ;

d d=1d                          ;
>                               ; end CORE STORE VERSION
```

```
<e2, [DISC VERSION track 2]
b k=d,i=10,a9,b2          ; comment only loaded if e18≠0;
    pm  ra3   , hh  ra    ; entry get: get blocks:=true; goto T;
a:  pm  ra8   , gt  ra2   ; entry put: get blocks:=false;
    ps (c50)  X  −124 IPB ; T: dor:=Rincr; swap; save p:=p;
a5: qq [unit] , ab  c17−3 ;   if (−,get blocks ∧ bit(5,Area)=0)
    ca  255   , hh  ra1   ;     ∨ bits(0,2,Area)≠1 then
    xr  −1    , hs  c6    ;   begin swap; rel track(−1);
a1: hv  se1   , srn rb2   ;    goto drum;
a2: pm  s125  , ps  s[dor];   end;
a3: ar  s127  X       ITA ;   parameterword:=
    ar  s126  , gr  c17−1 ;   arrayword + constant term;
    arn s127  , sr  c42   ;   if array length ≥ 40 then
    pa  c17−1 X V e18 NTA ;   part 1 of parameterword := block length
a4: pt  c49−11, hs  c27   ;   else alarm;
    dln rb    , gr  c50   ;   blocks := (array length − 1):block length;
    pm  c17−3 , tl  7     ;   R := bits(7,23,Area)
    tln 16    , sr  c50   ;       − blocks − place;
    sr  c17−4 , ca  0     ;   if R ≥ 0 ∧ place > 0 then
    arn c17−4 , sr  c42   ;   R := place − 1
    hv  ra4           LT  ;   else alarm;  part 2 of parameterword := 0;
    tk  18    , pt  c17−1 ;   parameterword := parameterword
    ac  c17−1 , tln 4     ;   + R pos 21
    ck  −10   , ga  ra5   ;   + bits(28,39,Area) pos 21;
    tln 30    , ac  c17−1 ;   unit := bits(24,27,Area);
    pm  c35   , tl  9     ;   if last used in buffer
    tln 15    , sr  c36   ;     = lower buf limit then alarm array
    hv  2c20          LZ  ;   else R := −array length;  M:=0;
    srn s127  , ps (ra5)  ; next block:
a6: ar  rb    X       ITA ;   R := R + block length; swap;
a7: cln −10           NTA ;   if M ≥ 0 then R := M shift −10;
    mt  ra7   , ar  c17−1 ;   R := parameterword − R;
    il  s     V       NPB ;   if get blocks then il(unit,R)
a8: qqf 111   , us  s16   ;   else us(unit+16,R);
    ar  rb1   , gr  c17−1 ;   parameterword :=
    arn 1c36  , il  s16   ;   R + 1 pos 21 + block length;
    il  0     , can(c17)  ;   UV := statusword;
    hvn ra6   X       LTA ;   if part 1 of statusword = 0 ∧ M < 0 then
    hh  c5                ;    begin R := 0; swap; goto next block end;
                          ;   p:= save p; goto exit std proc;
b:  qq  e18.39            ;   constant used by a4+1;
b1: qq  1.21+e18.39       ;   constant used by a8+1;
b2: qq  40.39             ;   constant use by a1;
e                         ;

d d=d−1                     ;

t put;                      ;
qq d−d1.9+2.14+1.19+1.29+33.39  ;
qq    2.9+2.14+10.19        ;

t get;                      ;
qq d−d1.9+2.14+1.19+0.29+33.39  ;
qq    2.9+2.14+10.19        ;

d d=2d                      ;
>                           ; end DISC VERSION
e                           ;
s                           ;
```

```
[31.8.1967                                          where, cancel, reserve
                                                   dummy reading of search]

b e3                        ; common core block
b k=d, i=10, a59, b11       ; common drum block
b c21                       ; block for search, only read in to define names

c:   qq[free word]          ;
     qq[free word 1]        ;
     qq[area word]          ;
     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
                            ; read track and sum:
c7:  lk [place0], 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: qq1k[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: itest:= 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                  ;   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  e18.2+1.5 , hvrc14;
     [mask for test cancel allowed, used by cancel]

[Define relative addresses in look up]
a1=c2−c1, a2=c3−c1, a3=c4−c1, a4=c8−c1, a5=c14−c1, b1=c4−c1, b2= c5−c1 ;
b3=c6−c1, b4=c7−c1, b5=c8−c1, b6=c9−c1, b7=c15−c1, b8=c19−c1, b9=c21−c1;

e [search]                  ;
```

```
i=i−40                    ; overwrite search
a10: qq40.19+1.39f[see a13]; constants
[1]  qq39.19+1.39         ;
[2]  qq −1.13+1.19−1.29   ;

e1:  hvn ra11             ; cancel: entry:= 0; goto common;
e2:  arn 1     DV         ; reserve: entry:= 1; goto common;
e3:  arn −1    D          ; where: entry:= −1; goto common;
a11: gr (c35)  t   −1     ; common:
     qq  14    , hs  c7   ;   block entry (12) locals block level: (1);
     hv  c11   , hh  c−2  ;
     hh (c38)  , arn c35  ;   i:= 2;                    Usage of stack:
     gr  p−1              ;   w[i] MA:= 0;              w[1]: last used:
     grn(p−1)  Vt  1   MA ;   goto get text;              cancel: qq 0
                          ;                               reserve: value length
a12: pa  c30   Vt  c50 LZA ; next word:                   where: addr of area
                          ;   if drum str then          w[2]: qq,
     ud  p3    , hs  c8   ;     UA:= addr(saveR) else   w[3]: first string word
     arn(c30)  , mb  r2a10 ; get text: format(str descr); ...
     pm (c30)  X       IZA ;   drum str:=value∧mask=0;   w[i]: last string word
     hv  ra14  X       NZA ;   M:= value;                ...
                          ;   if drum str then          w[12]: address of w[i]
     pm  c23              ;   begin R:=M; M:=track;     sr: p: block inf 1
a13: mt  ra10  , hs  c63  ;   marks:= 01; get track(R); p+1:     −    −  2
     arn(c30)  , sr  r1a10 ;  R:= description of        p+2: entry
     ar  ra10          LT  ;     next word of string;   p+3: str descr
     gm  c23   , pm  s1   ;   track:= M; M: string word; p+4:
     qq  0     , hs  c6   ;   get rel track (0);          cancel: return inf
                          ;   comment to set track place  reserve: length descr
a14: gm (p−1)  Xt  1   M  ;   and do save R:= R end;      where: area descr
     tl  −6    , is (p−1) ;   i:= i + 1; w[i] M:= M;    p+5: outside level or
     it  s−510 , bs  p−512 ;  if i < 11 ∧ string          return inf.
     ca  −1    , hv  ra12 ;     continued then goto next word;

     ncn(p2)              ;   if entry ≠ cancel then
     ud  p4    , hs  c8   ;     begin formal(second parameter)
     bs (p2)   , arn(c30) ;     if entry = reserve then R:= value end
     qq  1     , hs  c6   ;   else R:= 0;
     gs  rb10  , it  1    ;   w[1]:= R; get rel track (1);
     qq (p2)   , hs  c6   ;   addr of next track:= s; entry:= entry + 1;
     gr (c35)  , pp  s1   ;   get rel track (entry); p:= addr of the
     arn c42   , hs  c63  ;     called track; get place(1);
b10: hv[next track] t 1   ;   goto next track;
     qq ; fill
     qq ;
     qq ;
     qq ;

d e1=e1−a10, e2=e2−a10    ; define relative
d e3=e3−a10               ;   entries
```

```
a20: gp  rb11  , pp  s1    ; common part continued: set addr of next track;
     arn ra28  , hs  c63   ;   p:= addr of working area;
     vk  960   , can(c64)  ;   get place with rel (2, 5);
     arn ra24  , hv  ra21  ;   if no search track then
     vk (c64)  , lk  s-4   ;     begin Raddr:= 1; goto after search end
     vk (c64)  , gp  sb4   ;   from drum (search track, track place);
     pp (c35)              ;   set address of work area in search;
     pp  p1    , grn s-2   ;   p:= address of string:= last used + 1;
     hs  s                 ;   area word:= 0;  search;

a21: bs (p13)  , hh  ra22  ; after search:
     hv  ra26          NZ  ;   if entry = where then
     pp (c-1)  , arn s-2   ;     begin if not found then goto exit 1;
     hv  r4    X       NZ  ;     p:= sr;  M:= area word;
     srn c31   , tk  16    ;     if M = 0 then
     ar  s-4               ;        M:= free word - used in top of free pos 23
     ar  1.5   DX          ;        + 1 pos 5;
[4]  ud  p4    , hs  c24   ;     store formal ( M ) in: (area); R:= 0
     qq (c35)  , gm (c30)  ;     if area is not reserved in free then
     cln  44               ;       goto exit 1;
     hvn ra26          NO  ;     goto exit;
                           ;     end
a22h:hvn ra27  , it  1     ;   else if entry = cancel then
     bs (p13)  , hh  ra23  ;     begin
     pp  s                 ;   p := addr of search;
[-1] hv  ra26          NZ  ; search end entry:  if R ≠ 0 then goto exit 2;
     ps  r-2           IQB ;   set return(search end entry);  LQB := LB;
     qq        X       NA  ;   if -,area word then swap;
     hv  pb7           NZ  ;   if R ≠ 0 then goto get word;
a23h:                      ;     goto cancel 2
b11: hv [next track],pap13 ;     end;
     hv  ra25          NT  ; reserve 1: entry:= 0;
     arn(p10)  , tl  -6    ;   if -, found ∧ string correct terminated then
     ca  -6    , hv (rb11) ;     goto reserve 2;
a24: srn 1 [see 3a20] D V  ;   Raddr:= if found then 2 else
a25: arn 5     DV      NZ  ;     if caterror then 5 else 1; goto exit;
a26: ar  2     D           ; exit 1: Raddr:= 2;
a27: pp (c-1)  , ps  p5    ; exit: p:= sr;
     bs (p2)   , ps  p4    ;   s:= p + (if entry = cancel then 4 else 5);
     tl  -69   , hh  c21   ;   M:= Raddr;
                           ;   goto exit func with value in M;
a28: qq  5.19+2.39         ;   constant for get place.

     qq ; fill
     qq ;
```

```
a29: arn p−2   , mb  ra30  ; cancel 2: if area word does not say
     nc (pb9)  , hhn ra40  ;     reserved in free then
                           ;       begin R:= 0; goto exit 1 end; LA:= t;
     hs  ra38          LA  ;   while LA do back up;
     hs  ra36          NA  ;   while NA do clear back up;
     hs  ra36          LA  ;   while LA do clear back up;
     hs  ra36          LZ  ;   while R = 0 do clear back up;
     hs  ra33          IZA ;   found:= t; sum and write;
a30: qq 976[see r−7],hs p  ;   search; comment to end of catalog;
     hh  ra40          NT  ;   if cat error then goto exit 1; comment now NA;

a31: hs  ra38          NA  ; skip to area: while NA do backup;
     hs  ra37          LA  ;   while LA do set area;
     arn p−2   , tk  4     ;   if bit (3, area word) = 1 then
     tk  2     V       NT  ;     R:= free word 1
     arn p−3   , hv  ra32  ;   else if bit (5, area word) = 0 then
     hv  ra31          NT  ;     goto skip to area
     ck  −22   , ar  p−2   ;   else R:= areaword + bits(8, 23, areaword);
a32: sc  p−4   , ps  ra40  ;   free word:= free word − R; set return (exit);
     hs  pa1           IZA ; adjust free: found:= t; get free;
     arn p−4   , tk  24    ;   R:= bits(24, 39, free word);
     ck  −8    , ac (pb4)  ;   store[place 0]:= store[place 0] +
     ck  −16   , sc (pb4)  ;     R pos 23 − R;

a33: hs  pa4               ; sum and write: sum track;
     sk (pb4)  , vk (pb5)  ;   to drum (place 0); R:= 0;
     hrn s1                ;   wait track; return;
a34: hs  ra33          IZA ; get previous track: found:= t; sum and write;
     arn(pb1)  Dt  −1  M   ;   track:= track − 1;
     ca  −1    , ac  pb2   ;   if track = −1 then
     pa  pb1   t 959   NB  ;     begin group:= group − 1; track:= 959 end;
a35: nt 2[see a40], qq(pb5);   rel track:= rel track − 1;
     hs  pa3               ;   select track and read it;
     arn pb6   , hh  ra39  ;   iword:= isum; goto get word;

a36: grn(pb7)  V       MQB ; clear back up: store[iword] MQB:= 0;
                           ;   goto back up;
a37: gr  p−2               ; set area back up: area word:= R;

a38: arn pb7   , ca (pb4)  ; back up: if word = place 0 then
a39h:hv  ra34  , ga  pb7   ;     goto get previous track;
     pm (pb7)  Xt  −1      ; get word: iword:= iword − 1; R:= store[iword];
                           ;   return same;
a40h:hr s[not s1], ar ra35 ; exit 1: Raddr:= Raddr + 2;
     pp (c−1)  , ps  p4    ; exit: p:= display[−1]; s:= p+4;
     tl  −69   , hh  c21   ;   M:= Raddr; goto exit func with value in M;

     qq ; fill
```

```
a45: srn(c35)  , pm  s−4   ; reserve 2:
     hvn ra55          NT  ;   if length ≤ 0 then begin R:= 0; goto exit 1 end
     tk  16    , ar (c35)  ;   M:= free word;
     ac  s−4   , tln 7     ;   free word:= free word − length pos 23 + length;
     tln 16    , sr (c35)  ;   if length part (M) < length + used in top of free
a46: qq 3[see a55], sr c31 ;     then begin R:= 0; goto exit 1 end;
     hvn ra55          LT  ;   R:= new area word:=
     arn(c35)  , tl  16    ;     length pos 23 + first block part (M) +
     ar (sb9)  D       IZC ;     reserved in free bits;
     pp  s     , hs  ra48  ;   p:= address of search; store RMA;

a47: hsn ra50  X       IZB ; loop fill: fill:= t; M:= 0; store M MB;
     hv  ra47          NZ  ;   if R ≠ 0 then goto loop fill;
     arn r1    , hs  pa1   ;   Raddr:= not zero; get free;
     arn p−4   , gr (pb4)  ;   store [place 0]:= free word;
     ps  ra55  , hv  ra52  ;   set return (exit); goto sum and write;

a48: gr (pb7)  V       MA  ; store R MA: store[iword] MA:= R; goto testit;
a49: gm (pb7)          MPC ; store M: store[iword] MPC:= M;
     hs ra51               ; testit: test track; last used:= last used + 1;
     arn(c35)  t   1   IPC ;   M:= R:= set PC(store[last used]);
     tl  −6    , pm (c35)  ;   if bits(0, 3, R) ≠ string termination then
     nc  −6    , hv  ra49  ;     goto store M;

a50: gm (pb7)          MB  ; store M MB: store[iword] MB:= M;

a51: pmn(pb7)  DXt 1       ; test track: iword:= iword + 1;
     nc (pb6)  , hr  s1    ;   if iword ≠ isum then return;
     grn(pb6)          MC  ;   store[isum] MC:= 0;
     hv  ra52          LZB ;   if −, fill ∧ number of catalog tracks = rel tr
     arn p−3   , tk  −2    ;     then begin R:= 0; goto exit 2 end;
     ca (pb5)  , hhn ra55  ;
a52: hs  pa4               ; sum and write: sum track;
     sk (pb4)  , pa  pb3   ;   to drum(place 0); mode := 0;
a53: it 1[see a55], qq(pb5);   rel tr:= rel tr + 1;
     hv (pb1)  Dt   1      ;   track:= track + 1; goto select track;


a55: ar  ra53  , ar  ra46  ; exit 1: Raddr:= Raddr + 1; exit 2: Raddr:= Raddr + 3;
     pp (c−1)  , ps  p5    ; exit: p:= sr; s:= p + 5
     tl  −69   , hh  c21   ;   M:= Raddr; goto exit func with value in M;
     qq ; fill
     qq ;
     qq ;
     qq ;
     qq ;

d a59=170                  ;
a59: [check tracks]        ;

e [reserve, cancel, where]
```

[31.8.1967 where, reserve, cancel descriptions]

```
twhere;                          ;
qq d-d1.9+2.14+e3.29+2.33+37.39  ;
qq    7.9+11.14                  ;

treserve;
qq d-d1.9+4.14+e2.29+2.33+37.39  ;
qq    2.9+11.14                  ;

tcancel;                         ;
qq d-d1.9+3.14+e1.29+2.33+37.39  ;
qq   11.9                        ;

d d = 4d ;
e        ;
```

```
<e27                        ; comment the following is only loaded if
                            ; e27 is positive, i.e. in BUFFER MODE;
b k=d,i=10,a8               ; entry us:
     it  1                  ;    to unit:=true; goto start;
     pt  ra4   , gt  ra     ; entry il: to unit:=false;
     arn c17-4 , tl -12     ; start:
     tln 12    , gr  c17-2  ;    first:=bits(28,39,PARAMETER);
     hv  ra1            LZ  ;    if first=0 then alarm;
     psn(c50)  X            ;    M:=0;
     srn c42   , gr  c17    ;    UV:=R:=-1;
a:   ar  s1    , ps  s[dor] ;    PARAMETER:=(integer PARAMETER+R
     ar  s2    , tl  -12    ;    +arrayword+constant term)
     tln 12    , ac  c17-4  ;    ∧28012m;
     arn c17-3 , tl -9      ;    if bits(0,30,FUNCTION)=0 then
     gm  c17-3 V X      LZ  ;    FUNCTION:= short integer FUNCTION
a1:  pt  c49-12, hs  c27    ;    else alarm;
     hv  ra1            LZ  ;    if FUNCTION=0 then alarm;
                            ;    M:=PARAMETER;
     ck  6              IOA ;    if bits(6,9,FUNCTION)≠7
     pm  c17-4 , nc  7.3    ;      ∧bits(6,6,FUNCTION)=0 then
     hvn ra2   X        NOA ;    begin swap; goto magnetic tape end;
     ck -11    , nc  0      ;    if bits(1,4,FUNCTION)≠0
     nc  8     , hv  ra1    ;      ∧bits(1,4,FUNCTION)≠8 then alarm;
     hhn ra6   X        LOA ;    if bits(6,6,FUNCTION)=1 then
                            ;    begin swap; goto disc file end;
     xr        , ck -20     ; carrousel:
     tl -4     , ck  12     ;    R:=bits(10,15,M)+bits(20,27,R);
     tk  23    , hv  ra8    ;    M:=no of blocks pos 4; goto test;
                            ; magnetic tape:
a2:  tl -12    , ck -8      ;    R:=bits(0,6,R)+bits(20,27,R);
     tl -13                 ;    M:=no of words pos 13;
a8:  hv  ra1            NZ  ; test: if R≠0 then alarm;
a3:  tln 13    , gr  c50    ;    R:=RM shift 13;
                            ; test upper bound:
     arn s3    , sr  c50    ;    if length-number of words
     sr  c17-2 , ar  c42    ;     -first+1less 0
     hv  ra1            LT  ;     then alarm;
                            ; transfer:
a4h: arn c17-4 ,bs[to unit] ;    R:=PARAMETER;
     us (c17-3), hh  ra5    ;    if to unit then begin us(FUNCTION); UV:=0 end
a5h: il (c17-3), grn c17    ;    else begin il(FUNCTION); if -,busy then UV:=0 end;
a6h: hh  c5    , cl  10     ;    goto exit std proc;
                            ; disc file:
     tk  12    , ck -4      ;    if bits(22,27,R)≠0
     nc  0     , hv  ra1    ;    then alarm;
     xr        , sr  ra7    ;    if number of words greater blocklength on disc
     hv  ra1            NT  ;     then alarm
     ar  ra7   , hh  ra3    ;     else begin R:=number of words;
a7:  qq  e18.39+1.39        ;           goto test upper bound end;
e                           ;

t il;                       ;
qq d-d1.9+1.14+1.29+0.33+35.39 ;
qq    5.9+2.14+10.19        ;
t us;                       ;
qq d-d1.9+1.14+0.29+0.33+36.39 ;
qq    5.9+2.14+10.19        ;
d d=1d                      ;
>                           ; end condition e27 not positive
s                           ;
```

```
b k=d, i=10, a6            ;

     gt  ra    , ps (c50)  ; entry:
a:   arn s1    , ps  s[dor];

<e27 [BUFFER MODE]         ;
     ar  s2    , gr  c17   ;   UV := arrayword + constant term;
     arn s3    , sr  ra6   ;   if arraylength = 40 then
     pt  c17   V   40  LZ  ;    part 2 of UV := 40

×    [CORE STORE MODE]     ;
     ar  s2    , ck  −10   ;   first := arrayword + constant term;
     ga  ra3   , arn s3    ;   if arraylength = 40 then
     sr  ra6               ;
     qq        V       LZ  ;    skip next
>                          ;
     pt  c49−13, hs  c27   ;   else alarm(error 13);
     arn c42   , hs  c63   ;   get place(1);
     gp  ra2   , ps  s1    ;   save p := p;
     gs  ra1   , arn c42   ;   place 1 := trackplace + 1;
     ar  c42   , hs  c63   ;   get place(2);
     vk  960   , vk (c64)  ;
     lk  s1    , vk  960   ;   to core(search track, place 1);
a1:  it[place1], pa  s15   ;   search place 1 := place 1;
     pp  ra5               ;   p := address of text(system) − 1;
     hs  s5                ;   search;
     hv  ra4           NZ  ;   if R = 0 then
     pm  s3                ;
     tl  23    , tln 16    ;    begin M := areaword;
     hs  s7    X           ;     select(system track);

<e27 [BUFFER MODE]         ;
     lk  s1    , vk  960   ;   to core(system track, trackplace + 1);
a3:  ps  s1    , gs  c17   ; move:
     arn c17   , us  0     ;   A[1:40] := system[1:40];

×    [CORE STORE MODE]     ;
a3:  lk [first], vk  960   ;   to core(system track, first);
>                          ; move:  A[1:40] := system[1:40];
a2:  pp[save p], hh  c5    ;   p := save p;  goto exit std proc;
                           ;   end system found;
a4:                        ; fill zeroes:
<−e27+1,is(ra3), ps  s−1>  ;
     pa  r1    , pm  r1    ;   for i := 1 step 1 until 40 do
     grn s0    X t 1   M   ;
     nc  39    , hh  r−2   ;    system[i] := 0 mark 0;
<e27,hv  ra3   × hv  ra2>  ;   goto move;
d a5=i−1, tsystem;         ;
a6:  qq  40.39             ;
e                          ;

tsystem;                   ;
qq d−d1.9+1.14+36.39       ;
qq   10.9                  ;
d d=1d                     ;
s                          ;
```

```
<-e40+1,<e40+1
×de49=9
iload samba tape
×de49=8

[After i follows STOPCODE,SUMCODE and a sum character]
ia T7,L3
>

s
```

[Here follows STOPCODE and CLEARCODE]

```
d e58=1             ;   test tape number
<e49−5, <−e49+7,    ;   should be 8 or 6
d e58=0             ;
×
<e49−7, <−e49+9,    ;
d e58=0             ;
>                   ;
<e58                ;
i wrong tape
s                   ;
>                   ;

d e58=8             ;   version number

<e58−e50,           ;   if version number T8 and L4 gr max version number
                    ;   then the following definitions are loaded;
d e61=1             ;   define version number T1 and L1
d e62=2             ;   define version number T2
d e63=3             ;   define version number T3
d e64=4             ;   define version number T4
d e65=5             ;   define version number T5
d e66=6             ;   define version number T6 and L2
d e67=7             ;   define version number T7 and L3
d e68=e58           ;   define version number T8 and L4
de50=e58
>                   ;

<e61−e51+1,                 ; test version number T1 and L1
<e51−e61+1,×                ;
i wrong version number T1 and L1
>                           ;

<e68−e58+1,                 ; test version number T8 and L4
<e58−e68+1,×                ;
i wrong version number T8 and L4
>                           ;

<e48,                       ; test version number T2
<e52−e62+1,                 ;
<e62−e52+1,×                ;
<e48,                       ;
i wrong version number T2
>                           ;

<e48,                       ; test version number T3
<e53−e63+1,                 ;
<e63−e53+1,×                ;
<e48,                       ;
i wrong version number T3
>                           ;

<e48,                       ; test version number T4
<e54−e64+1,                 ;
<e64−e54+1,×                ;
<e48,                       ;
i wrong version number T4
>                           ;
```

```
<e48,                          ; test version number T5
<e55−e65+1,                    ;
<e65−e55+1,×                   ;
<e48,                          ;
i wrong version number T5
>                              ;

<e56−e66+1,                    ; test version number T6 and L2
<e66−e56+1,×                   ;
i wrong version number T6 and L2
>                              ;

<e57−e67+1,                    ; test version number T7 and L3
<e67−e57+1,×                   ;
i wrong version number T7 and L3
>                              ;


[end std proc]

t abs;                                ;
qq     0.9+ 1.23+ 0.29+ 5.33+46.39 ;
qq    19.9                            ;

t char;
qq c54−c.9+ 1.23+ 1.29+ 4.33+24.39 ;
qq                                    ;

t entier;                             ;
qq     0.9+ 1.23+ 0.29+ 5.33+45.39 ;
qq    20.9                            ;

t gier;                               ;
qq     0.9+ 1.23+ 0.29+ 5.33+45.39 ;
qq    16.9                            ;

t kbon;                               ;
qq     0.9+ 1.23+ 0.29+ 6.33+ 0.39 ;
qq                                    ;

t lyn;                                ;
qq     0.9+ 1.23+ 0.29+ 7.33+ 0.39 ;
qq                                    ;

t select;                             ;
qq     0.9+ 1.23+ 0.29+ 5.33+45.39 ;
qq    17.9                            ;

t tracks transferred;                 ;
qq c61−c.9+ 1.23+ 1.29+ 4.33+24.39 ;
qq                                    ;

t writechar;                          ;
qq     0.9+ 1.23+ 0.29+ 5.33+47.39 ;
qq    18.9                            ;

t writecr;                            ;
qq     0.9+ 1.23+ 0.29+ 8.33+ 0.39 ;
qq                                    ;

qqf  d                                ;
```

```
b k=e23,i=0     ;   load GPA segment word
d i=e21         ;
  qq e19.9+e10.19−e28.19+e50.29
e               ;


d e49=0         ;   tape number:=0;


[After i follows STOPCODE,SUMCODE and a sum character]
iø T8,L4 − wait − then SLIP or other

e
e
e
```

```
;slip<

[25.8.67                                T9, L5, M1  Gier Algol 4, page 1]



b i=15, a30, b20, c20, e20 ; The program starts in c1

d e4=i                     ; first core location used during translation
iif e4≠15 then set e4
iif any of the following parameters is not set,
idefine i and the parameter
s                                          ;


[Here follows STOPCODE, CLEARCODE]

[e4: qq  first track of system.39          ; set by tape T1, L1
1e4: qq  first track of library.39         ; set by tape L4
2e4: qq  first track of working area.39    ; set by tape L4 to the first free
         track following the library; 27 working tracks are needed]


[the working area must hold ≥ no of full tracks in RS and pass 8. The
following table is generated below and holds descriptions of how the
segments will be moved. The table consists of two parts, each of 19
words. The first part determines the final places and the second part
the actual palces for the segments. The words in the two parts look:

     qq  rel to.7+word size.19+to track.39
     qq  rel from.7+next1.14+next2.19+from track.39

after run of the program the table is intact, and location holds the
number of words in the translator as an integer with unit in pos. 39]

d i=4e4                    ; define i

c13: [part 1 of table]     ;
[ 0] qqf                   ;   GP
[ 1] qq                    ;   pass 1
[ 2] qq                    ;   pass 2
[ 3] qq                    ;   pass 3.1
[ 4] qq       ,            ;   std. proc ident
[ 5] qq                    ;   pass 3.2
[ 6] qq                    ;   pass 4
[ 7] qq                    ;   pass 5.1
[ 8] qq                    ;   pass 5.2
[ 9] qq       ,            ;   std proc descr
[10] qq                    ;   pass 6
[11] qq                    ;   pass 9
[12] qq                    ;   pass 7
[13] qq                    ;   pass 8.1
[14] qq       ,            ;   std proc code
[15] qq                    ;   pass 8.2 A         pass 8.2 and 8.3 are
[16] qq                    ;   pass 8.3 A (RS A)  moved twice;

[17] qq                    ;   pass 8.2 B
[18] qq                    ;   pass 8.3 B (RS B)
```

```
[part 2 of table]          ;

[19] qqf 1.14+ 1.19        ; 0   GP
[20] qq  2.14+ 2.19        ; 1   pass 1
[21] qq  3.14+ 3.19        ; 2   pass 2
[22] qq  5.14+13.19        ; 3   pass 3.1
[23] qq  4.7+9.14+9.19,    ; 4   std proc ident
[24] qq  6.14+ 4.19        ; 5   pass 3.2
[25] qq  7.14+ 5.19        ; 6   pass 4
[26] qq  8.14+ 6.19        ; 7   pass 5.1
[27] qq 10.14+ 7.19        ; 8   pass 5.2
[28] qq 14.14+14.19,       ; 9   std proc descr
[29] qq 11.14+ 8.19        ; 10  pass 6
[30] qq 12.14+10.19        ; 11  pass 9
[31] qq 13.14+11.19        ; 12  pass 7
[32] qq  4.14+12.19        ; 13  pass 8.1
[33] qq 15.14+15.19,       ; 14  std proc code
[34] qq 16.14+16.19        ; 15  pass 8.2 A
[35] qq 20.14+20.19        ; 16  pass 8.3 A (RS A)

[36] qq 18.14+18.19        ; 17  pass 8.2 B
[37] qq  0.14+ 0.19        ; 18  pass 8.3 B (RS A)

     qq[save segm],        ; must be a-marked
a12: qq [sum.39]           ;
a15: qq-1.7+1023.29+1023.39;
[1]  qq 3.9+1023.19        ;
c2:  qq [from track.39]    ;
c3:  qq [from rel.9]       ;
c4:  qq [to track.39]      ;
c5:  qq [to rel.9]         ;
c6:  qq [size.39]          ;
c7:  qq  40.39             ;
c8:  qq   1.39             ;
c9:  qq [max words.39]     ;
c10: qq [max tracks.39]    ;
c11: qq [total size.39]    ;

a10: pp  17    , grn a12   ; next segment:
     pm  pc13  , tln  7    ;   p:= chain; sum:= 0;
     ck  -10   , gr  c5    ;   to rel:= bits(0,7, word[p]);
     tln 12    , gr  c11   ;   total size:= bits(8, 19, word[p]);
     tln 20    , gr  c4    ;   to track:= bits(20, 39, word[p]);
     pm  p19c13, tln 7     ;
     ck  -10   , gr  c3    ;   from rel:= bits(0, 7, word[p+19]);
     tln 7     , ck  -10   ;   chain:= bits(8, 14, word[p+19]);
     ga  a10   , tln 5     ;   LPA:= word is std proc segm word;
     ck  -10           IPC ;   if -,direc then
     ga  a10           NTA ;   chain:= bits(15, 19, word[p+19]);
     tln 20    , gr  c2    ;   from track:= bits(20, 39, word[p+19]);
     ar  c3    , sr  c5    ;   R:= from track + from rel pos 9
     sr  c4    , pt  a8    ;     - to rel pos 9 - to track;
     pt  a8    t   1   LZ  ;   write:= R = 0;
                           ; move next part:
a11: arn c9    , sr  c11   ;   if total size>max words then
     pm  c11   V       NT  ;     begin size:=max words;
     pm  c9    , it  1     ;       more:=true;
     pa  b               IOB ;     end else
     gm  c6                ;     begin size:=total size; more:=false
     arn c3    , ck  10    ;     end;
     ar  c6    , gr  a6    ;   words:= size + from rel pos 39;
     ppn(c3)   X   c   IZA ;   read tracks:= true;
                           ;   from rel:= p:= place 2 + from rel;
     pm  c2    , hs  a     ;   M:= from track; transfer tracks;
```

```
     pm  a12   , arn c6    ;   M:=size; R:=sum;
a7:  sr  c8    X       IZA ;   for i:=1 step 1 until size do
     ar  p     , pp  p1    ;    begin  R:=R+core[p]; p:=p+1;
     ar  2     D       LA  ;      if mark a then R:=R+2pos9;
     ar  1     D       LB  ;      if mark b then R:=R+1pos9;
     hv  a7    X       NZA ;
     gr  a12   , ck  0     ;      sum := R;  R00 := 0;
b:   bs [more] , hv  a8    ;      if more then goto after sum;
     qqn b1    V       NTB ;     if old comp∧−,LPA∧−,no checksum then
     qqn               LPA ;      begin
     sr  a12   V       LZ  ;        if R≠0 then alarm(≮<pass sum≯)
     qq  c12   , hs  c14   ;      end;  core[p]:= core[p] − R;
     ac  p−1           M   ;     if check GP then goto update segment table;
     hv  a24           LPB ;    end;
a8:  pm  c4    , bs [write]; after sum: M:=to track;
     sk (a4)   , hh  b11   ;     if write then begin write last track;
                           ;     goto end transf end;
a22: hsn a             IZC ;    onetrack:=read tracks:=true;
     it (c3)   , pa  a4    ;   transfer tracks;
     pp (c5)   , arn c5    ;   last place:=place2+from rel;
     ck  10    , ar  c6    ;   p:=to rel;
     gr  a6    X       IOC ;   one track:=read tracks:=false;
     dln c7    , cln −10   ;   rel:=(size+to rel) mod 40;
     ga  b5    , srn a6    ;   R:=−size−to rel;
     qq  pe8   , hs  b2    ;   move words;
     ar  c7    , ps  b4    ;   R:=R+40; set return(after write);
     qq (a4)   t   −40 IZB ;   last place:=last place−40; LZB:=R=0;
     hv  a3    X       LZ  ;   if R≤0 then
b4:  hv  a3    X       LT  ;    begin swap; goto TR end;
     hh  b11           LZB ; after write:
                           ;   if LZB then goto end transf;
     ca (b5)               ;   if rel=0 then
     ps  b6    , hv  b1    ;    begin set return(end transf);
                           ;     goto count and transfer end;
     qq (a2)   t   −1  NZ  ;   if R≠0 then track:=track−1;
b5:  pp [rel]  , ps (a4)   ;   p:=rel;
     ps  s39   , gs  b6    ;   save:=last place+39;
     ud  a1                ;   last place:=place1;
     hsn b1            IZC ;   read tracks:=one track:=true;
                           ;   goto count and transfer;
     it  p     , qq (a4)   ;   last place:=last place+p;
     ps (a4)   , it (a10)  ;   if chain > 19 then
     bs  20    , hv  b6    ;
     grn s             M   ;   for i := 1 step 1 until 39 do
     bt  38    t   −1      ;     fill zeroes;
     ps  s1    , hv  r−2   ;
b6:  qq  p0    , hs  b2    ;   move words; write last track
b11: sk (a4)   , bs (b)    ; end transf:
     arn c9    , hh  b9    ;   if −,more then
     bs (a10)  t  19   M   ;    begin if chain>19 then
     qq  c17   , hs  c18   ;     writetext(≮<end merger≯);
b9:  hv  a10   , sc  c11   ;    end;  goto next segment;
     arn c10   , ac  c2    ;   total size:=total size−max words;
a25: ac  c4    , nt  c     ;   to track:=to track+max tracks;
     qq (c3)   , hv  a11   ;   from track:=from track+max tracks;
                           ;   from rel:=from rel−place1;
```

```
b2:  bs  p511  , hv  s1    ; procedure move words;
     pm (s)           IRC ; move: if p=0 then return;
     gm (a4)   t −1   MRC ;   core[lastplace+p]:=core[addr(s)+p];
     pp  p−1   , hv  b2    ;   p:=p−1; goto move;

a:   dln a5    , ar  a5    ; transfer tracks:
     ck  −10   , ga  a3    ;   group:=M:960;
     cl  −10   , ga  a2    ;   track:=Mmod960;
     pa  a4    V   e6  NZB ;   last place:=if only one track then
a1:  pa  a4    t   e9      ;               place1 else place2;
     srn a6    X           ;   M:=−words;
a2:  is [track], can s−960 ; next tracks: if track=960 then
     pa  a2    , it  1     ;   begin track:=0; group:=group+1 end;
a3:  vk [group], vk (a2)   ;   select(group); select(track);
a4:  lk[last pl]VXt40  LZA ; TR: last place:=last place+40; swap;
     sk (a4)   X   40      ;   if read tracks then lk else sk;
     ar  c7    , vk (a2)   ;   R:=R+40; wait drum;
     hvn s1           LZB ;   if R≥0∨one track then
                          ;    begin R:=0;  return end;
     hvn s1           NT  ; count and transfer:
b1:  hv (a2)   DX  1       ;   track:=track+1; swap; goto next tracks;
a5:  qq  960.39            ;
a6:  qq [size+rel]         ;
                           ;
c14: sy  29                ; alarm: RED RIBBON;
c18: sy  64                ; writetext: writecr;
     it (s)    , pa  r1    ;   next core:=addr(core[s]);
[2]  pmn 0     X           ; next textword: M:=0; R:=core[next core];
[3]  cl  34    , ck −4     ;   RM:=RM shift 34;
     ga  c15   , ca  15    ; next char: R:=R shift −4; char:=Raddr;
     hv (2c18) D   1       ;   if char=15 then begin
     ca  10    , hsf 2     ;   next core:=next core+1; goto next textword end;
     ca  63    , it  1     ;   if char=10 then Call Help;
c15: sy −1     , cln−6     ;   writechar(char); R:=0; RM:=RM shift −6;
     hh  3c18              ;   goto next char;

a24: hv  a8           LPA ; update segment table:
     pm  a12−1        IPA ;   if no sumcheck then goto after sum;
     hsn a            IZC ;   no sumcheck:=only one track:=read tracks:=true;
a26: pp [rel]  , arn 4c13  ;   transfer tracks; P:=rel segm table;
     gt  pe13  , arn 9c13  ;   part2 of core[p+4+place1]:=word size std 1;
     gt  pe14  , arn 14c13 ;   part2 of core[p+9+place1]:=word size std 2;
     mb  1a15  , gr  pe15  ;   bits 0−19 of core[p+14+place1]:=
                           ;   word size std 3;
     sk (a4)   , grn a12   ;   updated track back to drum; wait drum;
     vk (a2)   , hh  a25   ;   sum:=0; from rel:=from rel−place2;
                           ;   goto move next part;
c12: tpass sum;            ;
c16: tsystem match;        ;
c17: tend merger;          ;
```

[the following code used for generating the two tables in c13ff and initializing the program is overwritten, i.e. a new run with the program requires, that the tape is reloaded]

```
c1:
u c1                          ; START MERGER:
d c=40c1                      ;   define place 2;
     vy  17    , pm  e4       ;   select(17);
     hsn a             IZC    ;   from drum (first comp, c1);
     arn 1e4           IOB    ;   only one track:=false;
     hs  a     X              ;   from drum(first std, e5);
     pmn e11   X              ;   abs track segm table:=M:=first comp+
     cl  10    , dln c7       ;   part 1 of second word comp:40;
     ar  e4    X              ;   rel := M mod 40;
     ck  −10   , ga  a16      ;   only one track:=true;
     gm  a12−1 , ga  a26      ;   save abs track segm list
                              ;   save rel track segm list
     hsn a             IZB    ;   from drum(M,c1); i:=0;
a16: pp [rel]  , pm  pe5      ;   for p:=rel step 1 until rel+16 do
     tl  −20   , tl  20       ;    begin
a19: gm  c13−1 t   1          ;     word[i]:= segm table[p] ∧ 20 0 20 m;
     is (a18)  , ncn s−2      ;     if i≠14 then
     pa (a19)                 ;      part 1 of word[i]:=0;
a18: bt  16    t  −1          ;    i:=i+1;
     pp  p1    , hh  a16      ;    end;
                              ;
     arn pe10  , nc (e3)      ;   if part 1 of word 3 std proc ≠
     qq  c16   , hs  c14      ;   part 1 of first word segment list
                              ;   then alarm(≮<system match≯);
     srn 4c13          ITB    ;   old comp:= word[4]≠0; sum:=R:=0;
     grn 3e4   , pm  16c13    ;
     tl  −20                  ;   if −,old comp then
     gm  3e4           NTB    ;    sum:= wordsize RS else sum:=0;
[−1] pp  0             IZB    ; next scan: p:=0; first:= R=0;
a13: arn pc13  , mb  1a15     ; next word:
     ck  −20   , ar  3e4      ;   sum:=sum + bits(0,19,word[p]);
     gr  3e4   , pp  p1       ;   p:=p+1;
     bs  p−16  , hv  a27      ;   if p<17 then begin
     nc  p−11  , hv  a17      ;   if p=11 then begin
     gr  c2    V       LZB    ;   if first then save sum:=sum
     sr  c2            ITA    ;   else direc:= sum < save sum end;
a17: pm  3e4   , dln c7       ;   R:= sum:40 +
     tk  8     , cl  −8       ;       (sum mod 40) pos 7 +
     ar  e4    , pm  a15      ;        first comp; comment marks in R := 00;
     pm  pc13          NTB    ;   if −,first then word[p] := word[p] + R
     ac  pc13  V       NZB    ;   else if −,old comp∧word[p] is std proc word then
     pp  p1            LA     ;   p:=p+1;
     ac  p19c13        LZB    ;   if first then word[p+19] := word[p+19] + R;
     hv  a13                  ;   goto next word end;
a27: hv  a14           NZB    ;   if first then
                              ;    begin
     pm  e7    , gm  4c13     ;     word[4]:= word size std proc ident pos 19;
     arn e1    , mb  1a15     ;     word[9]:= word size std proc spec pos 19;
     gr  9c13  , grn 3e4      ;     word[14]:= word size std proc code pos 19;
     arn e2    , mb  1a15     ;     sum:=0;
     gr  14c13 , hv  a13−1    ;     comment R≠0; goto next scan;
                              ;    end; goto modify;
```

```
a21: mb  1a15  , ar  1e4   ; subroutine used below;
a23: ar  e12   t   1       ;
     gr  (s)   , hv  s1    ;
a14: pt  e1    , pa  e2    ; modify:
     pt  e2    , pt  e7    ;   clear bits 10-19 of std proc words descr.
     arn  23c13, hs  a21   ;   set from track and from rel
     arn  28c13, hs  a21   ;   in word[23], word[28]
     arn  33c13, hs  a21   ;   and word[33]
     pa  23c13 t   4.7     ;
     arn e4                ;
     ac  c13   , ac  19c13 ;   word[0]:=word[0]+ first comp;
                           ;   word[19]:=word[19]+first comp;
     hv  a20           LTB ;
     arn c13   , mb  1a15  ;
     tl  -59   , dln c7    ;
     tk  8     , cl  -8    ;   word[37]:= word[37] +
     ar  e4    V           ;     if old comp then word[35] ∧ 8 m 12 0 20 m
a20: arn 35c13 , mb  a15   ;     else word size GP:40 + first comp +
     ac  37c13             ;      (word size GP mod 40) pos 7;
     arn 34c13 , mb  a15   ;
     ac  36c13 , arn 34c13 ;   word[36]:= word[36] + word[34] ∧ 8 m 12 0 20 m;
     mb  1a15  , gr  34c13 ;   clear bits 0-6 and 20-39 of word[34];
     arn 35c13 , mb  1a15  ;
     gr  35c13 , arn 15c13 ;   clear bits 0-6 and 20-39 of word[35];
     gt  17c13 , arn 16c13 ;   part 2 of word[17]:= part 2 of word[15];
     gt  18c13 , arn 2e4   ;   part 2 of word[18]:= part 2 of word[16];
     ac  17c13 , ac  34c13 ;   word[17]:= word[17] + first work
     arn 15c13 , mb  1a15  ;   word[34]:= word[34] + first work;
     ck  -20   , sr  c8    ;
     xr        , dln c7    ;   next:=(word size pass 8-1):40+1;
     ar  c8    , ar  2e4   ;   word[18]:=word[18]+next+first work;
     ac 18c13  , ac 35c13  ;   word[35]:=word[35]+next+first work;

     pmn-c-40  XD          ;
     cl  10    , dln c7    ;   max tracks := free words : 40;
     gr  c10   X           ;
     mln c7    , gm  c9    ;   max words:= max words × 40;
     pa  a10   t  17       ;   chain:= 17;
     pt  a1-1  t  c-40     ;   palce2:=e5;
     pt  a1    t  c1-40    ;
     hv  a10               ;   goto next segment;

d e5=i,e11=1e5, e12=e5+39  ;
d e1=41e5, e2=42e5, e3=43e5;
d e9=e5-40, e7=40e5,e8=c1-1;
d e6=e5, e10=e5-16         ;
d e13=4c1, e14=9c1,e15=14c1;
```

[After i follows STOPCODE, SUMCODE and a sum character]

```
ix T9,L5,M1

e
e
```