GIER SYSTEM LIBRARY

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

REGNECENTRALEN

[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;
T = 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
T=e32                    ;
d6: ttype;               ;
d7: tcall;               ;
d8: tsubscripts;         ;
e32=T                    ;
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: pmm[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]);
    pmm(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;
```

[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:arm(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]:= int result; go to REPEAT;
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 INPUT;
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 ≠ 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 to 27(R);
    ck  6                   ;
    mb  15    D             ;    output (bits 30 to 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-6;
    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 formal 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 literal, 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 > ≠  =

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 delete call
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
```

[7.7.67]
[GIER ALGOL 4, pass 6, page 13]

[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
```

[7.7.67]
[GIER ALGOL 4, pass 6, page 14]

```
[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 10110 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.]
```

```
[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 ≠ <=> 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 ≠ 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 ≠ 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 -, comma 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 do
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 KUTZPQR 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 + 2;
     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: if 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 V 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 V (srp bits + indir);
     pt  b     , ud  c30    ;   part 2 instr:= 0; rps bits:= indir:= 0;
     bs  s     , c1  -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  e60    ; 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    ;     nlhead:= 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 ≠ 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
```

```
[                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 ≠ 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 ≠ 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 XVIIMNLKQTZPQ                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]            ;
```

```
[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, x  ;   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 [take 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      ; simpel 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  1      ; 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     , c1  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 ≠ variabel then goto outvar;
    tk  16    , nc (b9)    ;    if blockno ≠ 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 ON BOOLEAN: if top -< R then
c44h:hr s1    , hs   e5      ;    output(tk1); return;
    hv  e3    , qqn  e34     ; STACK: alarmprint(<<stack>);
[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 worktable;
    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(«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 ≠ 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 ≠ 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 ≠ 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
V1
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, blockstack
     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,
x    [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: exchange, >, hop LT
qq c75 t  22b.9                ; 58, >then: >, hop NT
qq c73 t    0.9+ 25b.19        ; 59, ≠then: =, 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, ≠ : ≠, 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, aux 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 minimizing 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 is -.

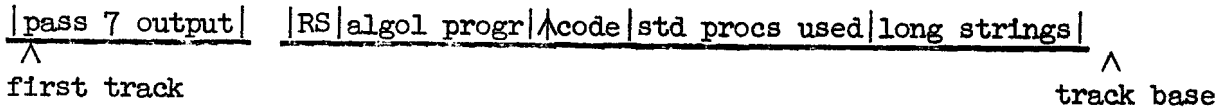
3. Standard procedure tracks described in the tracklist (pass 8 input) are transferred to the final programarea.

State of e4 parameters:

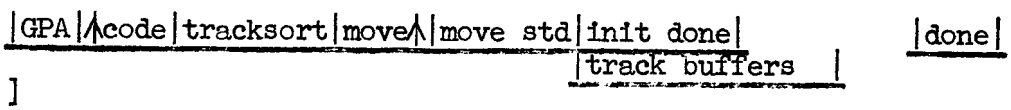
Initially:	Finally:
1e4: used tracks	changed to drummode before task 1
4e4: available	updated after task 1
8e4: inputtrack	updated before task 1
9e4: output track	updated after task 1
10e4: increment	changed to drummode (1) before task 1
13e4: running track	running track - ↑ length

Testprint from 8.1 consists of every tracknumber selected.

Drumpicture after pass 8:



8.1 core picture:



]

```
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         ;                                 else outputtrack) - 1;
[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;
```

```
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
```

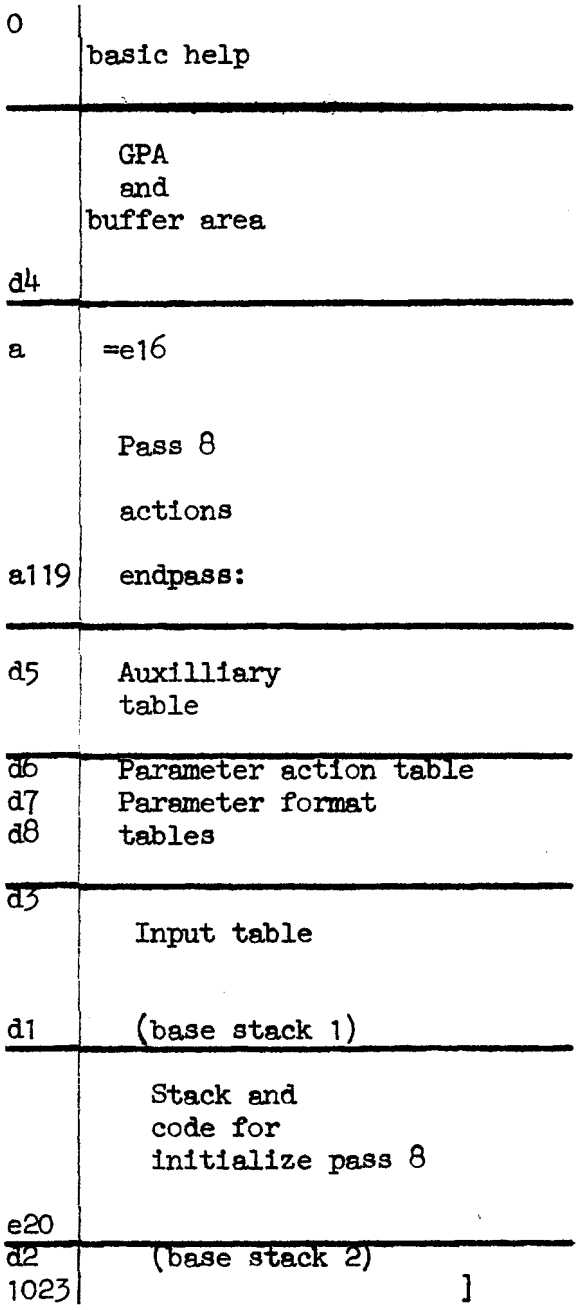
```
bk=e22+e14, i=e16-e47, a131, b50, d70  ;
I=e16                                  ;

  d27=40                                   ; define tracklength;

d0=-d27-d27-d27-d27+3                  ; maxapp:= -(4xtracklength - 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 ≠ 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) ≠ 0 then
     ck  -10   , nc  0      ;    begin comment enddotrack ≠ 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                  ;   stack 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]40, hr  s1      ;   old B:= 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] OV -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 (simulateloop);
     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   ;   paramact:= 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;   PC:= 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

x     [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 value 38
    qq 3.5+28d5.39+13 ft d62          ;                              take integer value 39
    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>      gm 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
x    [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 by 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 a101. 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 initstack;

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                 ;
X1                      ;
<e49-1, <-e49+3,        ;
d e56=0                 ;
V1                      ;
<e56                    ;
i wrong tape
s1                      ;
V1                      ;

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
V1                      ;
```

```
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=1, 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: pmm 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                         ;   specif top:= i; top section 3:= core[i];
     arn d4    D                     ; move proctable to core top:
a6:  pm (b11)  t   -1                ;   for i:= specif top 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;

     pmm(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 tracksx40) 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 drum;
     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.9+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=1, e49=7              ;   tape number := 7;

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

ia   T6, L2
s [end loader code]        ;
```

[Here follows STOPCODE and CLEARCODE]

```
<e49-6, <-e49+8, x  ;   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
>                   ;
```

[7.3.1967

read integer
read real, track 1]

```
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            ; sig: 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;
```

[27.2.1967 read general, track 2
BUFFER VERSION]

<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 constant to get p-mark;
b7:   nc  0     , ca  0     ; comment constant 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                             ;
```

[27.2.1967

read string(A,word,number)
BUFFER VERSION track 1]

```
<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;
```

[27.2.1967 read string(A,word,number)
BUFFER VERSION track 2]

```
     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                           ;
```

```
x                                   ;    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;
```

[27.2.1967

read string(A,word,number)
CORE STORE VERSION track 2]

```
      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 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                          ;
>                               ; end condition 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/3x0.333306+arg;
    mkf  ra9   , arf  c17           ;
    grf  c17   , hh   c5            ;     goto exit std proc
                                    ; big arg:
a1: dkf  c17   , it   a11           ;     base:=-3xpi/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)/4xabs(arg);
    sr   256   D                    ;
    an   c52   , tk   -1            ;
    dk   c17                        ;     UV:=(UV-0.5+abs(arg)/2)/2xUV;
    gr   c17    X                   ;
    mkn  c17   , gr   c51           ;     work2:=UV/2;
    pan  r1                t5       ;
    ar   r0     X          t1       ;     R:=UVxpolynomium;
    mkn  c17   V                LA  ;
    mkn  c51   , hv   r-2           ;
a3: ar   r0    , mt   ra            ;     R:=(base+R)xsign(a);
a4: nkf  1     , grf  c17           ;     UV:=2xR;
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             ;     -3xpi/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)-511xln(2);
   hv   ra3                     NT ;     if R>0 then goto large;
   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:=0.75x(1+bxUV/\2);
   gr   c52   , arn  ra6             ;
   mk   ra8    X                     ;
   mkn  c17   , sc   c52             ;     work2:=work1-0.75x(axUV+cxUV/\3);
   ar   c51    X                     ;
   mln  ra9   , dl   c52             ;     R:=work1/work2/sqrt(2);
a1:nkf  0               t1           ;     RF:=Rx2/\(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                        ;     511xln(2)
a5: 369/ 337/ 869/ 174               ;     log base2(e)/2;
a6: 266/ 172/ 574/ 725               ;     0.75xa=0.519 860 384 44;
a7:  73/ 797/ 660/  37               ;     0.75xb=0.144 099 511 27;
a8:   8/ 524/ 902/ 710               ;     0.75xc=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
```

[25.8.1967

put(A,Area,place)
get(A,Area,place)
CORE STORE 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
```

[25.8.1967

put(A,Area,place)
get(A,Area,place)
DISC VERSION, track 2]

```
<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 used 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: itext:= p; equal:= t;
c15: pmn[iword]Xt   1  IPC          ; get word: iword:=iword+1; M:= 0;
                                    ;   R:= setPC(store[iword]);
c16: hv (rc4)  Dt   1  LC           ; next track: test for last word on track;
     hr  s1    X       NZA          ;   if LC then begin track:= track + 1;
                                    ;   goto select track end;
c17: gr r[iparam]Vt 1  LA           ;   if found then begin swap; return end;
c18=c-1c17                          ;   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]                          ;
```

where, cancel, reserve, 1. track
common part, entries]

```
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 iword = 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
```

[31.8.1967 where, cancel, reserve 4. track
reserve 2]

```
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                        ;
V                             ; 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

x    [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];

x    [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   x 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
xde49=9
iload samba tape
xde49=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                 ;
x                       ;
<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,x                    ;
i wrong version number T1 and L1
>                               ;

<e68-e58+1,                     ; test version number T8 and L4
<e58-e68+1,x                    ;
i wrong version number T8 and L4
>                               ;

<e48,                           ; test version number T2
<e52-e62+1,                     ;
<e62-e52+1,x                    ;
<e48,                           ;
i wrong version number T2
>                               ;

<e48,                           ; test version number T3
<e53-e63+1,                     ;
<e63-e53+1,x                    ;
<e48,                           ;
i wrong version number T3
>                               ;

<e48,                           ; test version number T4
<e54-e64+1,                     ;
<e64-e54+1,x                    ;
<e48,                           ;
i wrong version number T4
>                               ;
```

```
<e48,                          ; test version number T5
<e55-e65+1,                    ;
<e65-e55+1,x                   ;
<e48,                          ;
i wrong version number T5
>                              ;

<e56-e66+1,                    ; test version number T6 and L2
<e66-e56+1,x                   ;
i wrong version number T6 and L2
>                              ;

<e57-e67+1,                    ; test version number T7 and L3
<e67-e57+1,x                   ;
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 places 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    ;   LFA:= 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>0Vone 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     ;   place2:=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]

```
i  T9,L5,M1

e
e
```