```
mi0  17.10.1971  14,36,15
 update monitor
 version 26.7.1971
edit begin.
VOL1newmon
edit end.
edit begin.
edit end.
edit begin.
edit end.
edit begin.
edit end.
edit begin.
edit end.
edit begin.
edit end.
edit begin.
edit end.
edit begin.
edit end.
 assembly follows
```

MONITOR

System 2

```
      0
      0 b.m.
      0                 monitor text 1
      0
      0  ; rc 4000 system tape
      0  ; per brinch hansen, leif svalgaard
      0  ;  this tape contains version 2.5 of the rc 4000 multiprogramming
      0  ; system. it is written in the slang 3 language and consists of
      0  ; 10 segments surrounded by a global block:
      0  ;
      0  ; global block, definitions:
      0  ;      a names define system constants;
      0  ;      b names define entries in the monitor table;
      0  ; segment 1; start monitor segment 10:
      0  ;      contains a jump to segment 10;
      0  ; segment 2, monitor:
      0  ;      contains interrupt response code and monitor procedures;
      0  ; segment 3, external processes:
      0  ;      contains send message and code for input/output;
      0  ; segment 4, process descriptions:
      0  ;      contains name table, process descriptions, and message buffers:
      0  ; segment 5, initialize monitor:
      0  ;      executed and removed immediately after loading:
      0  ; segment 6, process functions:
      0  ;      contains code for catalog administration and the
      0  ;      creation and removal of processes:
      0  ; segment 7, initialize process functions:
      0  ;      executed and removed immediately after loading:
      0  ; segment 8, operating system s:
      0  ;      contains code which allows the operators to
      0  ;      create and control new process from consoles:
      0  ; segment 9, initialize catalog
      0  ;      starts the multiprogramming system and is
      0  ;      itself immediately executed as a part of the
      0  ;      process s; it can initialize the backing store
      0  ;      with catalog entries and binary programs
      0  ;      input from paper tape or magnetic tape;
      0  ; segment 10: move monitor:
      0  ;      allocates segment 2 - 9 after autoloading
      0
      0
      0  ; global block, definitions
      0
      0     b128, a128
      0
      0  ; size options:
      0  ; a1 = no of area processes
      0  ; a3 = no of internal processes
```

```
0  ; a5 = no of message buffers
0  ; a9 = no of storage bytes
0  ; a87 = inspection interval
0  ; monitor version = 2.5
0  t.
[illegible]
0
0  ; monitor size options
0
0     a1=72          ; area processes
0     a3=20          ; internal processes
0     a5=142         ; message buffers
0     a9=64<11       ; core (kwords) (does not make simul possible)
0     a87=10000      ; clock inspection interval in 0.1 msec
0  n. ; include monitor size options
0
0  ; a2 = size of area process description
0  ; a4 = size of internal process description
0  ; a6 = size of message buffer
0
0     a2=22, a4=74, a6=24
0
0  ; a88 = size of catalog entry
0  ; a89 = standard interrupt mask
0  ; a85 = max time slice in 0.1 msec
0
0     a88=34, a89=8.4777 7777, a85=256
0
0  ; process options:
0  ; process options determine whether code is included
0  ; for a given kind of external process. they are defined
0  ; by bits in the identifier a91 as follows:
0  ; ✓ rc 315  typewriter:                always included
0  ;           olivetti terminal:         a91=a91 o. 1<23
0  ; ✓ rc 2000 paper tape reader:         a91=a91 o. 1<22
0  ; ✓ rc 150  paper tape punch:          a91=a91 o. 1<21
0  ; ✓ rc 610  line printer:              a91=a91 o. 1<20
0  ; ✓ rc 749  magnetic tape:             a91=a91 o. 1<18
0  ;           interrupt key:             a91=a91 o. 1<16
0  ;   dpc 405 alphanumeric display:      a91=a91 o. 1<12
0  ;   rc 4195 graphic display:           a91=a91 o. 1<11
0  ;   rc 4124 www transmission line      a91=a91 o. 1<8
0  ;   rc 4194 kingmatic plotter          a91=a91 o. 1<7
0  ;   rc 3200 transmission terminal      a91=a91 o. 1<6
0  ;           telex                      a91=a91 o. 1<5
0  t.
[illegible]
0
0  ; include code for external process drivers
0
0     a91= 2.1111 0100 0000 0001 1010 0000
0  n. ; include process code
0
0  ; format of internal process description:
0
0  a10 =  0              ; <kind>
0  a11 =  2              ; <name>
0  a12 = 10, a13 = 11    ; <stop count><state>
0  a14 = 12              ; <identification bit>
0  a15 = 14              ; <next event>
0                        ; <last event>
0  a16 = 18              ; <next process>
0                        ; <last process>
0  a17 = 22              ; <first address>
0  a18 = 24              ; <top address>
0  a19 = 26, a20 = 27    ; <buffer claim><area claim>
0  a21 = 28, a22 = 29    ; <internal claim><function mask>
0  a23 = 30              ; <catalog mask>
0  a24 = 32, a25 = 33    ; <protection register><protection key>
0  a26 = 34              ; <interrupt mask>
```

```
0   a27 = 36              ; <interrupt address>
0   a28 = 38              ; <working register 0>
0   a29 = 40              ; <working register 1>
0   a30 = 42              ; <working register 2>
0   a31 = 44              ; <working register 3>
0   a32 = 46              ; <exception register>
0   a33 = 48              ; <instruction counter>
0   a34 = 50              ; <parent description address>
0   a35 = 52              ; <quantum>
0   a36 = 54              ; <run time>
0   a38 = 58              ; <start run>
0   a39 = 62              ; <start wait>
0   a40 = 66              ; <wait address>
0   ~~a41 = 68~~              ; <creation no>
0   a42 = 70              ; [redacted] bs base
0   a43 = 72              ; <selection mask>
0
0   ; internal process states:
0
0   ; actual bitpatterns are relevant to process functions only
0   a95 = 2.01001000 ; running
0   a96 = 2.00001000 ; running after error
0   a97 = 2.10110000 ; waiting for stop by parent
0   a98 = 2.10100000 ; waiting for stop by ancestor
0   a99 = 2.10111000 ; waiting for start by parent
0   a100= 2.10101000 ; waiting for start by ancestor
0   a101= 2.11001100 ; waiting for process function
0   a102= 2.10001101 ; waiting for message
0   a103= 2.10001110 ; waiting for answer
0   a104= 2.10001111 ; waiting for event
0
0
0   ; bit patterns used to test or change the above states:
0   a105 = 2.00100000; waiting for stop or start
0   a106 = 2.00001000; waiting for start
0
0   ; format of area process description:
0
0   a10 =  0              ; <kind>
0   a11 =  2              ; <name>
0   a50 = 10, a51 = 11    ; <device number * 2><catalog key>
0   a52 = 12              ; <reserved>
0   a53 = 14              ; <users>
0   a60 = 16              ; <first segment number>
0   a61 = 18              ; <number of segments>
0   a62 = 20              ; <creator>
0
0   ; format of peripheral process description:
0
0   a10 = 0  ; <kind>
0   a11 = 2  ; <name>
0   a50 = 10 ; <device number*64>
0   a52 = 12 ; <reserved>
0   a53 = 14 ; <users>
0   a54 = 16 ; <next message>
0   a55 = 18 ; <last message>
0   a56 = 20 ; <interrupt address>
0
0   ; optional parameters for peripheral devices:
0   a70 = 22 ; <parameter 0>
0   a71 = 24 ; <parameter 1>
0   a72 = 26 ; <parameter 2>
0   a73 = 28 ; <parameter 3>
0   a74 = 30 ; <parameter 4>
0   a75 = 32 ; <parameter 5>
0   a76 = 34 ; <parameter 6>
0   a77 = 36 ; <parameter 7>
0   a78 = 38 ; <parameter 8>
0
0   ; format of message buffer:
0
0   ;      relative address:      message:
 0  ;                 0              <next buffer>
 0  ;                 2              <last buffer>
 0  ;                 4              <receiver>
 0  ;                 6              <sender>
 0  ;
 0  ;                 8-22           <message>
 0  ;
 0  ; standard i/o message and answer:
 0  ;                 8       <operation><mode>           <status word>
 0  ;                10       <first storage address>     <number of bytes>
 0  ;                12       <last storage address>      <number of characters>
 0  ;                14       <first segment no>
 0
 0  ; message buffer states:
 0
 0  ; the possible states of a message buffer are defined by the
 0  ; values of the sender and receiver parameters:
 0  ;
 0  ; sender param;  receiver param:  state:
 0  ;       0               0         buffer available
 0  ; sender descr   receiver descr   message pending from existing sender
 0  ; sender descr  -receiver descr   message received from existing sender
 0  ;-parent descr   receiver descr   message pending from removed sender
 0  ;-parent descr  -receiver descr   message received from removed sender
 0  ; sender descr          1         normal answer pending
 0  ; sender descr          2         dummy answer pending (rejected)
 0  ; sender descr          3         dummy answer pending (unintelligible)
 0  ; sender descr          4         dummy answer pending (malfunction)
 0  ; sender descr          5         dummy answer pending (does not exist)
 0
 0
 0  ; segment 1
 0  ;
 0  ; start segment 10 in its last word
 0
 0
 0  s. i2
 0  w.
 0
 0  i0:               i2.     ;   length of segment 1
 2                     0      ;   init cat switch: writetext
 4  i1:                0      ;   init cat switch: medium
 6
 6  ; entry from autoloader:
 6      al. w3      i0.       ;   calculate top address of
 8      wa  w3  x3+0,r.10     ;   segment 10:
28      al. w2      i2.       ;   insert start address of segment 2;
30      dl. w1      i1.       ;   get init cat switches
32      jl        x3-2        ;   jump to segment 10
34  i2:                       ;   first word of segment 2
34
34  ; exit with:
34  ;   w0, w1 = init cat switches
34  ;   w2     = start address of segment 2
34
34  e.   ;  end segment 1
34
34
34  b. f14, e39, d75, c51
34
34  ; segment 2: monitor
34
34  s. k = 8, j36
 8  w.b127=k, j29, k=k-2
 8  ; segment structure:
 8  ;      monitor table          (b names)
 8  ;      interrupt response     (c names)
 8  ;      utility procedures     (d names)
 8  ;      monitor procedures     (e names)
 8  ;      name table             (f names)
 8  ;      process descriptions   (f names)
 8  ;      buffers                (f names)
```

```
   8   ;
   8   ;      (g and h and i names are used locally)
   8
   8   ; monitor table:
   8
   8   w.      0      ; <interrupt number>
  10           0      ; <system start address>
  12           0      ; <interrupt response>
  14           0      ; <start key response>
  16      b0:         ; <interrupt 0-24>
  16           0, r.25
  66      b1:  0      ; <current process>
  68      b2:  0      ; <next running process>
  70           0      ; <last running process>
  72      b3:  0      ; <name table start>
  74      b4:  0      ; <first device in name table>
  76      b5:  0      ; <first area in name table>
  78      b6:  0      ; <first internal in name table>
  80      b7:  0      ; <name table end>
  82      b8:  0      ; <next message buffer>
  84           0      ; <last message buffer>
  86           0      ; <message pool start>
  88           0      ; <message pool end>
  90           0      ; <message buffer size>
  92      b9:  0,0    ; <time base>
  96           ~~0,0~~    ; <not used yet>
 100           0      ; <log mode>
 102      b10:  0     ; <maximum time slice>
 104      b11:  0     ; <time slice>
 106      b12:  0     ; <microseconds>
 108      b13:  0     ; <time>
 110            0     ;
 112      b14:  0     ; <clock value>
 114      b15:  0     ; <clock device no * 64>
 116            0     ; <no of storage bytes>
 118      b16 = k     ; <first monitor procedure>
 118            0, r.38
 194      b17 = k-2 ; <last monitor procedure>
 194      b18:  0     ; <current buffer address>
 196      b19:  0     ; <current receiver>
 198      b20:  0     ; <interrupt return address>
 200      b21:  0     ; <process link in dummy internal process>
 ~~202~~
 202  b. g24
 202
 202  ; comment: after loading and initialization, the system is
 202  ; started by: jl (10) which enters the monitor here:
 202
 202  w.c25:io   w2 (b15)        ; system start:
 204        rs   w2  b14         ;   clock:= sense(timer);
 206        jl   w3  j8          ;   select internal;
 208        jl       g0          ;   goto interrupt return;
 210
 210  ; comment:  after reset-start the monitor simulates an interrupt 24
 210
 210    c26:rs   w3   8          ; start button:
 212        al   w3  48          ;   word (8):= 2*24;
 214        rx   w3   8          ;   comment: all registers unchanged;
 216
 216                             ; interrupt response:
 216    c27:am (b1)              ;   save w0(cur):= w0;
 218        ds   w1  a29         ;   save w1(cur):= w1;
 220        rl   w1  b1          ;   save w2(cur):= w2;
 222        ds   w3  x1+a31      ;   save w3(cur):= w3;
 224        xs       x1+a32+1    ;   save ex(cur):= ex;
 226        dl   w3  10          ;   save ic(cur):= word(10);
 228        rs   w3  x1+a33      ;
 230        jl      (x2+b0)      ;   goto case word (8) of
 232                             ;   (0: interrupt 0
 232                             ;        - - -
 232                             ;   48: interrupt 24);
 232  ; w1 = cur, w2 = interrupt no, w3 = save ic
```

```
232
232   b.h24, c24=k                ; unwanted interrupts:
232   w.g0: rl   w1   b1          ; interrupt return:
234         rl   w3   x1+a33      ;   word(10) := save ic(cur);
236         rs   w3   10          ;   ~~if protected (word(10))~~
238         ~~se        x3+0~~        ;   ~~and protection key(cur) <> 0~~
240         ~~jl        h1~~          ;   ~~then goto program error;~~
242     h0: xl        x1+a32+1    ;   ex:= save ex(cur);
244         dl   w3   x1+a31      ;   w3:= save w3(cur);
246         dl   w1   x1+a29      ;   w2:= save w2(cur);
248         ic        (h2)        ;   w1:= save w1(cur);
250         je        (10)        ;   w0:= save w0(cur);
252     ~~h1: bz   w0   x1+a25~~      ;   ir(1:2):= 0;
254         ~~sn   w0   0~~           ;   enabled goto word(word(10));
256         ~~jl        h0~~          ;
258
258     g1: al   w0   a96         ; program error:
260         jl   w3   j9          ;   remove internal(running after error, irrelev
262         jl        g0          ;   goto interrupt return;
264     h2: 2.11<21               ;
266   e.
```

phil!
remove
these

```
266
266   b.h24
266   w.c0:                       ; interrupt 0:
266         rl   w2   x3-2        ;   function:= word(save ic(cur) - 2)
268         ws   w2   h0          ;                 -jd 1<11;
270         sl   w2   0           ;   if function < 0
272         sl   w2   b17-b16+2   ;   or function > max function
274         jl        j28         ;   then goto internal 0;
276         al   w3   g0          ;
278         rs   w3   b20         ;   return:= interrupt return;
280         jl        (x2+b16)    ;   goto monitor call(function);
282   ;     w1 = cur, w2 = function
282
282     c28:am (b6)               ; internal 0:
284         se   w1   (0)         ;   if cur = name table(first internal)
286         jl         h2         ;
288         sn   w2   (h1)        ;   and function = aw  w3
290         jl         g2         ;   then goto from process functions;
292     h2: am         -2         ;   cause:= 0
294     c1: am        -2          ; interrupt 1: or 2
296     c2: am        -2          ; interrupt 2: or 4
298     c29:al   w0   6           ; internal 3:  or 6;
300         rl   w1   b1          ;
302     g3:                       ; reset:
302         rl   w2   x1+a27      ;   ia:= interrupt address(cur);
304         sn   w2   0           ;   if ia = 0
306         jl        g1          ;   then goto program error;
308         rl   w3   x1+a33      ;   word(ia+12):= cause;
310         ds   w0   x2+12       ;   word(ia+10):= save ic(cur);
312         rl   w0   x1+a32      ;
314         rs   w0   x2+8        ;   word(ia+8):= save ex(cur);
316         dl   w0   x1+a31      ;   word(ia+6):= save w3(cur);
318         ds   w0   x2+6        ;   word(ia+4):= save w2(cur);
320         dl   w0   x1+a29      ;   word(ia+2):= save w1(cur);
322         ds   w0   x2+2        ;   word(ia):= save w0(cur);
324         al   w2   x2+14       ;
326         rs   w2   x1+a33      ;   save ic(cur):= ia + 14;
328         jl        g0          ;   goto interrupt return;
330     h0: jd 1<11
332     h1: aw w3
334   e.
334
334   ; external single interrupt:
334   ; comment: a single interrupt source connected to one interrupt bit.
334   ; the interrupt response causes a jump to a single instruction
334   ; placed on top of a process description, for example:
334   ;            c5: jl   w1   c30
334   ;          c5+2: <process description>
334
334     c30:rs   w1   b19         ;
336         rl   w2   x1+a54      ;   proc:=link;
```

```
338        rs   w2   b18         ;   buf:=next mess(proc);
340        ~~al   w3   g0~~          ;
342        ~~rs   w3   b20         ;   return:=interrupt return;~~
344        jl      (x1+a56)      ;   goto interrupt addr(proc);
346  ;     w1 = proc  w2 = buf
346
346  ; external multiple interrupt:
346  ; comment: multiple interrupt sources connected to one interrupt bit.
346  ; the interrupt response cause a jump to a single instruction
346  ; placed on top of a table defining the digital register and
346  ; the interrupt sources, for example:
346  ;              c9: jl  w1  c31
346  ;            c9+2: <device number * 64>
346  ;            c9+4: <process description address>
346  ;            c9+6: <process description address>
346  ;            etc.
346
346  b.h24
346  w.c31:al   w3   h2          ;   return:= multi return;
348        rs   w3   b20         ;   table:= link;
350        io   w2  (x1+0)       ;   digital:= sense(word(table));
352        al   w3   0           ;   digital:= (digital con 0) shift -2;
354        ld   w3   -2          ; continue:
356    h1: nd   w3   0           ;   normalize(digital,exp);
358        so   w0  (h5)         ;   if exp(1)=0 then
360        jl        g0          ;     goto interrupt return;
362        bs   w1   0           ;   table:= table-exp;
364        bs   w1   0           ;   table:= table-exp;
366        ws   w2   h5          ;   digital(1):= 0;
368        ds   w2   h3
370        rs   w3   h4
372        rl   w1   x1+0
374        rs   w1   b19         ;   proc:= word(table);
376        rl   w2   x1+a54
378        rs   w2   b18         ;   buf:= next mess(proc);
380        jl      (x1+a56)      ;   goto interrupt addr(proc);
382  ;     w1=proc  w2=buf
382    h2: dl   w2   h3          ; multi return:
384        rl   w3   h4
386        jl        h1          ;   goto continue;
388        0                     ;
390    h3: 0                     ;
392    h4: 0                     ;
394    h5: 1<22                  ; bit 1
396  e.
396
396  ; wait interrupt:
396  ; comment: saves an interrupt address for an external process.
396  ;      call:
396  ; w0
396  ; w1   proc
396  ; w2
396  ; w3   interrupt address
396
396    c33:al   w3   c33       ; dummy interrupt: return:= wait intrpt:
398
398    c32:rs   w3   x1+a56    ; wait intrpt: interrupt addr(proc):=link;
400        jl      (b20)       ;   goto return;
402
402  ; from process functions:
402  ; comment: process functions call the monitor by executing
402  ; the instruction  jd  w3  1<11+0 ;
402
402  b.h24
402  w.g2: al   w0   a102       ;
404        rl   w2   x1+a15     ;   proc:=next(event q(cur));
406        sn   w2  (x1+a15+2); if proc=last(event q(cur))
408        jl   w3   j9         ;   then remove internal(wait mess,x);
410        jl   w3   j5         ;   remove(proc);
412        al   w1   x2-a16     ;
414        jl   w3   j10        ;   link internal(proc);
416        jl      ~~(b20)~~ g0    ;   goto return;
```

```
418   e.
418                    jl w3 j10  ;  link internal (cur); ?
418   b.h24                         ;                              should be
418   w.c51:rl   w2   b2            ; interrupt 24:                 included.
420         sn   w2   b2            ;   if next (timer q) <> timer q
422         jl        h0            ;   then
424         bz   w0   x1+a13        ;   remove internal (running,irr);
426         jl   w3   j9            ;
428     h0: rl   w1   b6            ;   proc:= second internal;
430         rl   w1   x1+2   2 4    ;   comment: supposed to be an opsys;
432         jl   w3   j10           ;   link internal (proc);
434         al   w0    8            ;   cause:= 8;
436         jl        g3            ;   goto reset;
438   e.                            ; end;                   third..
438   e.    ; end of interrupt response
438
438   ; monitor utility procedures
438
438   ; procedure print w0
438   ; procedure print w1
438   ; procedure print w2
438   ; procedure print w3
438   ; comment: prints the contents of a working register as a signed
438   ; integer on typewriter 2 in disabled mode. only used for testoutput.
438   ; before the call w2 and w3 must be saved in double-word d0.
438   ; after return all registers are restored.
438   ;          call:     return
438   ; w0                 unchanged
438   ; w1                 unchanged
438   ; w2       saved     restored
438   ; w3       link      restored
438
438   b.g24                         ; begin
438   w.g2: 0                       ; return,
440         0, g3: 0, 0             ; save 0, save 1, save 2,
446     d0: 0                       ; save 3;
448     d1: am   2                  ; print w3: number:= save 3
450     d2: am   d0-4               ; print w2:        or save 2
452     d3: am   2                  ; print w1:        or w1
454     d4: rl   w2   0             ; print w0:        or w0;
456         ds   w1   g3            ;   save 1:= w1;
458         rs   w3   g2            ;   save 0:= w0;
460         dl   w1   g3            ;   w0:= save 0; w1:= save 1;
462         dl   w3   d0            ;   w2:= save 2; w3:= save 3;
464         jl       (g2)           ;
466   e.                            ; end
466
466   ; procedure remove(elem)
466   ; comment: removes a given element from a queue.
466   ;        call:     return:
466   ; w0               unchanged
466   ; w1               unchanged
466   ; w2     elem      elem
466   ; w3     link      next(elem)
466
466   b.g24                         ; begin
466   w.d5: rs   w3   g0            ;
468         rl   w3   x2+2          ;
470         am       (x2+0)         ;
472         rs   w3   2             ;   last(next(elem)):= last(elem);
474         rl   w3   x2+0          ;
476         rs   w3  (x2+2)         ;   next(last(elem)):= next(elem);
478         rs   w2   x2+0          ;
480         rs   w2   x2+2          ;   next(elem):= last(elem):= elem:
482         jl       (g0)           ;
484     g0: 0                       ; comment: link,
486   e.                            ; end
486
486   ; procedure link(head, elem)
486   ; comment: links a given element to the end of a queue
486   ;      call:    return:
486   ; w0            unchanged
```

```
486  ; w1  head     head
486  ; w2  elem     elem
486  ; w3  link     old last(head)
486
486  b.g24                      ; begin
486  w.d6: rs  w3  g0           ;
488        rl  w3  x1+2         ;   old last:= last(head);
490        rs  w2  x1+2         ;   last(head):= elem;
492        rs  w2  x3+0         ;   next(old last):= elem;
494        rs  w1  x2+0         ;   next(elem):= head;
496        rs  w3  x2+2         ;   last(elem):= old last;
498        jl      (g0)         ;
500    g0: 0                    ; comment: link;
502  e.                         ; end
502
502  ; procedure time(slice, usec)
502  ; comment: senses the timer and updates the programmed timers
502  ; microseconds and current time slice.
502  ;      call:     return:
502  ; w0             unchanged
502  ; w1             unchanged
502  ; w2             slice
502  ; w3  link       usec
502
502  b.g24                      ; begin
502  w.d7: rs  w3  g0           ;
504        io  w2 (b15)         ;   new value:= sense(timer);
506        al  w3  x2+0         ;
508        ws  w3  b14          ;   increase:= new value - clock;
510        sh  w3  -1           ;   if increase < 0 then
512        wa  w3  g1           ;   increase:= increase + 16384;
514        rs  w2  b14          ;   clock:= new value;
516        al  w2  x3+0         ;
518        wa  w2  b11          ;   slice:= slice + increase;
520        wa  w3  b12          ;   usec:= usec + increase;
522        ds  w3  b12          ;
524        jl      (g0)         ;         burdi wdde
526    g0: 0, g1: 16384         ;
530  e.                         ; end       Trait current pc
530
530  ; procedure select internal
530  ; comment: selects a new current internal process from the
530  ; timer queue.
530  ;      call:     return:
530  ; w0             unchanged
530  ; w1             unchanged
530  ; w2             unchanged
530  ; w3  link       unchanged
530
530  b.g24                      ; begin
530  w.d8: ds  w1  g1           ;
532        rl  w1  b2           ;
534        sn  w1  b2           ;
536        rl  w1  b21          ;   cur:= (if next(timer q) <> timer q
538        al  w1  x1-a16       ;          then next(timer q)
540        rs  w1  b1           ;          else dummy proc) - a16;
542        rl  w0  x1+a35       ;
544        rs  w0  b11          ;   slice:= quantum(cur);
546 remove ~~pl      x1+a24       ;   pr:= save pr(cur);~~
548        ml      x1+a26       ;   im:= save im(cur);
550        ~~bz  w0  x1+a25~~       ;
552        ks  w0  0            ;   protection key(0):=
554        ks  w0  2            ;   protection key(2):=
556        ks  w0  4            ;   protection key(4):=
558        ks  w0  6            ;   protection key(6):= save pk(cur);
560        dl  w1  g1           ;
562        jl      x3+0         ;
564        0, g1: 0             ;
568  e.                         ; end
568
568  ; procedure remove internal(proc state, proc addr)
568  ; comment: removes current internal process from the timer queue
```

phil! remove these

```
568  ; and sets its state and wait address. after this, a new current
568  ; process is selected.
568  ;      call:       return:
568  ; w0   proc state  proc state
568  ; w1               unchanged
568  ; w2   proc addr   proc addr
568  ; w3   link        link
568
568  b.g24                        ; begin
568  w.d9: ds   w3   g0           ;
570        rl   w3   b1           ;
572        hs   w0   x3+a13       ;   state(cur):= proc state;
574        rs   w2   x3+a40       ;   wait addr(cur):= proc addr:
576        jl   w3   d7           ;   time(slice, usec);
578        rl   w3   b1           ;
580        rs   w2   x3+a35       ;   quantum(cur):=slice;
582        dl   w3   b13+2        ;
584        am       (b1)          ;
586        ds   w3   a39+2        ;   start wait(cur):=time;
588        rl   w3   b1           ;
590        al   w2   x3+a16       ;
592        jl   w3   d5           ;   remove(cur + a16);
594        jl   w3   d8           ;   select internal;
596        dl   w3   g0           ;
598        jl        x3+0         ;
600        0, g0: 0               ;
604  e.                           ; end
604
604  ; procedure link internal (proc)
604  ; comment: links an internal process to the timer queue.
604  ; it is linked as the first proces, if its time quantum is
604  ; less than the maximum time slize; otherwise it is linked
604  ; as the last process.
604  ;      call:     return:
604  ; w0             unchanged
604  ; w1   proc      proc
604  ; w2             unchanged
604  ; w3   link      link
604
604  b.g24
604  w.d10:ds   w1   g5—          ; begin
606        ds   w3   g6—          ;
608        jl   w3   d7           ;   time(slice,usec);
610        rl   w3   b1           ;
612        rs   w2   x3+a35       ;   quantum(cur):= slice;
614        al   w0   a95          ;
616        hs   w0   x1+a13       ;   state(proc):= running;
618        rl   w0   x1+a35       ;
620        al   w2   x1+a16       ;
622        sl   w0  (~~b10~~)     ;   if quantum(proc) < ~~max slice~~ then
624        jl        g1  x1+a24   ;
626        rl   w1   b2           ;   begin
628   g0:  jl   w3   d6           ;   link(next(timer q), proc + a16);
630        jl   w3   d8           ;   select internal;
632        dl   w1   g5           ;   end
634        dl   w3   g6           ;
636        jl        x3+0         ;
638   g1:  al   w3   0            ;   else
640        wd   w0   ~~b10~~ x1+a24 ;   begin
642        rx   w3   x1+a35       ;   new quantum:=
644        ws   w3   x1+a35       ;   quantum(proc) mod ~~max slice;~~
646        ad   w0   -24          ;
648        aa   w0   x1+a36+2     ;   run time(proc):=run time(proc)+
650        ds   w0   x1+a36+2     ;   quantum(proc)-new quantum;
652        al   w1   b2           ;   quantum(proc):=new quantum;
654  [redacted]                   ;   link(timer q, proc);
656        jl        g0           ;   select internal;
658        0  0                   ;   end;
660   g5:  0  1                   ;
662        0  2                   ;
664   g6:  0  3                   ;
666  e.                           ; end
```

link after priority

→ max slice(proc)

```
666
666  ; procedure search name(name, entry)
666  ; comment: searches the name table for a given name and delivers its
666  ; entry in the name table. if the name is undefined the entry
666  ; is name table end.
666  ;      call:    return:
666  ; w0            unchanged
666  ; w1            unchanged
666  ; w2  name      name
666  ; w3  link      entry
666
666  b.g24                      ; begin
666  w.d11:ds  w1  g2           ;
668       rs   w3  g3           ;
670
670  ; if the word following the name points to an entry within
670  ; the name table we at first try that entry.
670
670       rl   w3  x2+8         ; try entry:
672       sl   w3 (b3)          ;   entry:= name+8;
674       sl   w3 (b7)          ;   if entry is outside name table
676       jl       g4           ;   then goto search;
678       rl   w3  x3           ;
680       dl   w1  x2+6         ;   if name in call <>
682       sn   w0 (x3+6)        ;      name in monitor
684       se   w1 (x3+8)        ;   then goto search;
686       jl       g4           ;
688       dl   w1  x2+2         ; found: if name in call = 0
690       sn   w0 (x3+2)        ;         then entry:= name table end:
692       se   w1 (x3+4)        ;   return;
694       jl       g4           ;
696       rl   w3  x2+8         ;
698       jl       g5           ;
700    g4: al  w1  x2-2         ; search:
702       rs   w1 (b7)          ;   nametable(nametable end):= name - 2;
704       rl   w3  b3           ;   entry:= name table start:
706       jl       g1           ;   goto exam;
708    g0: al  w3  x3+2         ; next: entry:= entry + 2;
710    g1: am     (x3+0)        ; exam:
712       dl   w1  8            ;
714       sn   w0 (x2+4)        ;
716       se   w1 (x2+6)        ;
718       jl       g0           ;
720       am      (x3+0)        ;
722       dl   w1  4            ;
724       sn   w0 (x2+0)        ;   if name in call <>
726       se   w1 (x2+2)        ;   name in monitor
728       jl       g0           ;   then goto next;
730    g5: sn  w0  0            ;   if name in call = 0
732       rl   w3  b7           ;   then entry:= name table end;
734       dl   w1  g2           ;
736       jl      (g3)          ;
738       0                     ; comment: w0,
740    g2: 0                    ; w1,
742    g3: 0                    ; link;
744  e.                         ; end
744
744  ; procedure check buf(pool, addr, sorry)
744  ; comment: checks whether an address is a buffer address
744  ; within a given buffer pool.
744  ;      call:    return:
744  ; w0            unchanged
744  ; w1  pool      pool
744  ; w2  addr      addr
744  ; w3  link      link
744
744  b.g24                      ; begin
744  w.d12:ds  w2  g1           ;
746       sl   w2 (x1+4)        ;   if addr < start addr(pool)
748       sl   w2 (x1+6)        ;   or addr >= end addr(pool)
750       jl       x3+0         ;
752       ws   w2  x1+4         ;
```

```
754         al   w1   0            ;
756         am        (g0)         ;   or (addr - start addr(pool))
758         wd   w2   8            ;   mod buf size(pool) <> 0
760         se   w1   0            ;
762         jl        x3+0         ;   then goto sorry;
764         dl   w2   g1           ;
766         jl        x3+2         ;
768     g0: 0                      ;
770     g1: 0                      ;
772    e.                          ; end
772
772    ; procedure release buf(pool, buf)
772    ; comment: links a given buffer to a given pool.
772    ;       call:     return:
772    ; w0              unchanged
772    ; w1   pool       pool
772    ; w2   buf        buf                     !
772    ; w3   link       link               ; d13-2
772
772    b.g24              al w1 b8     ; begin
772    w.d13:ds   w0   g0              ;
774         al   w0   0                ;
776         rs   w0   x2+4             ;   receiver(buf):=
778         rs   w0   x2+6             ;   sender(buf):= 0;
780         jl   w3   d6               ;
782         dl   w0   g0               ;   link(pool, buf);
784         jl        x3+0             ;
786         0,  g0: 0                  ;
790    e.                              ; end
790
790    ; procedure move mess(from, to)
790    ; comment: moves 8 message or answer words from a
790    ; given storage address to another.
790    ;       call:     return:
790    ; w0              unchanged
790    ; w1   from       from
790    ; w2   to         to
790    ; w3   link       link
790
790    b.g24                           ; begin
790    w.d14:ds   w0   g0              ;
792         dl   w0   x1+2             ;
794         ds   w0   x2+2             ;
796         dl   w0   x1+6             ;
798         ds   w0   x2+6             ;
800         dl   w0   x1+10            ;   move (8) words
802         ds   w0   x2+10            ;   from (from)
804         dl   w0   x1+14            ;   to (to);
806         ds   w0   x2+14            ;
808         dl   w0   g0               ;
810         jl        x3+0             ;
812         0,  g0: 0                  ;
816    e.                              ; end
816
816    ; procedure deliver answer(buf)
816    ; comment: delivers an answer from a receiver and starts
816    ; the sender if it is waiting for the answer. if the sender
816    ; has been removed, the buffer is returned to the pool, and
816    ; the buffer claim of the parent is increased by one.
816    ;       call:     return:
816    ; w0              unchanged
816    ; w1              unchanged
816    ; w2   buf        destroyed
816    ; w3   link       destroyed
816
816    b.g24                           ; begin
816    w.d15:ds   w1   g4              ;
818         rs   w2   b18              ;
820         rs   w3   g5               ;
822         [redacted]
824         jl   w3   d5               ;   remove(buf);
826         rl   w3   x2+6             ;
```

```
828        rs   w3   g6         ;   internal:= sender(buf);
830        sh   w3   -1         ;   if internal < 0
832        jl        g3         ;   then goto parent;
834        bz   w0   x3+a13     ;
836        sn   w0   a103       ;   if state(internal) <> wait answer
838        se   w2  (x3+a40)    ;   or wait address(internal) <> buf
840        jl        g2         ;   then goto event;
842        rl   w0   x2+4       ; answer:
844        rs   w0   x3+a28     ;   save w0(internal):= receiver(buf);
846        al   w1   x2+8       ;   from:= buf + 8;
848        rl   w2   x3+a29     ;   answer:= save w1(internal):
850        jl   w3   d14        ;   move mess(from, answer);
852        rl   w1   g6         ;
854        jl   w3   d10        ;   link internal(internal);
856        rl   w2   b18        ;
858   ~~g0: al   w1   b8~~      ; release buf:
860   g0:  jl   w3   d13-2      ;   release buf(mess pool, buf);
862        rl   w1   g6         ;
864        bz   w2   x1+a19     ;
866        al   w2   x2+1       ;   buf claim(internal):=
868        hs   w2   x1+a19     ;   buf claim(internal) + 1;
870   g1:  dl   w1   g4         ;
872        jl       (g5)        ;   goto exit;
874   g2:  al   w1   x3+a15     ; event:
876        jl   w3   d6         ;   link(event q(internal), buf);
878        se   w0   a104       ;   if state(internal) = wait event
880        jl        g1         ;   then
882        rl   w1   g6         ;   begin
884        al   w0   1          ;
886        rs   w0   x1+a28     ;   save w0(internal):= 1;
888        rs   w2   x1+a30     ;   save w2(internal):= buf;
890        jl   w3   d10        ;   link internal(internal);
892        jl        g1         ;   end;
894                             ;   goto exit;
894                             ; parent:
894   g3:  ac   w3   x3+0       ;   internal:= -internal:
896        rs   w3   g6         ;   goto release buf;
898        jl        g0         ; exit:
900        0, g4: 0             ;
904   g5:  0, g6: 0             ;
908   e.                        ; end
~~908~~
908   ; procedure deliver message(buf)
908   ; comment: delivers a message to an internal process and
908   ; starts it if it is waiting for a message.
908   ;      call:     return:
908   ; w0             unchanged
908   ; w1             unchanged
908   ; w2   buf       destroyed
908   ; w3   link      destroyed
908
908   b.g24                     ; begin
908   w.d16:ds   w1   g3        ;
910         rs   w3   g4        ;
912         rl   w3   x2+4      ;
914         rs   w3   g5        ;   internal:= receiver(buf);
916         bz   w0   x3+a13    ;
918         se   w0   a102      ;   if state(internal) <> wait message
920         jl        g2        ;   then goto event;
922         rl   w0   x2+6      ; message:
924         rs   w0   x3+a28    ;   save w0(internal):= sender(buf);
926         rs   w2   x3+a30    ;   save w2(internal):= buf;
928         ac   w1   x3+0      ;
930         rs   w1   x2+4      ;   receiver(buf):= -internal;
932         al   w1   x2+8      ;   from:= buf + 8;
934         rl   w2   x3+a29    ;   message:= save w1(internal);
936         jl   w3   d14       ;   move mess(from, message);
938         rl   w3   g5        ;
940         rl   w2   x3+a28    ;
942         rl   w3   x3+a31    ;   name:= save w3(internal);
944         dl   w1   x2+4      ;
946         ds   w1   x3+2      ;   move(4) words
```

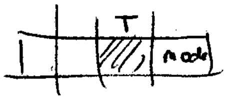
```
        dl   w1   x2+8        ;   from (sender(buf) + 2)
        ds   w1   x3+6        ;   to(name);
        rl   w1   g5          ;
   g0:  jl   w3   d10         ;   link internal(internal);
   g1:  dl   w1   g3          ;
        jl       (g4)         ;   goto exit;
   g2:  al   w1   x3+a15      ; event:
        jl   w3   d6          ;   link(event q(internal), buf)
        se   w0   a104        ;   if state(internal) = wait event
        jl        g1          ;   then
        rl   w1   g5          ;   begin
        al   w0   0           ;
        rs   w0   x1+a28      ;   save w0(internal):= 0;
        rs   w2   x1+a30      ;   save w2(internal):= buf;
        jl        g0          ;   link internal (internal);
        0, g3: 0              ;   end;
   g4:  0, g5: 0              ; exit:
e.                            ; end

; procedure check name area
; comment: checks whether a name area is within the
; current internal process.
;       call:       return:
; w0                unchanged
; w1                cur
; w2                name
; w3    link        destroyed

b.g24                         ; begin   jl g1
w.d17:rs   w3   g0            ;
~~      rl   w1   b1~~  ok.   ;
        rl   w2   x1+a31      ;   name:= save w3(cur);
        al   w3   x2+6        ;
g1:     ~~sl   w2   (x1+a17)~~    ;   if name < first addr(cur)
        ~~sl   w3   (x1+a18)~~    ;   or name + 6 >= top addr(cur)
        ~~jl        c29~~         ;   then goto internal 3;
        ~~jl       (g0)~~         ;
   ~~g0: 0~~                  ;
~~e.~~                        ; end

; procedure check mess area
; comment: checks whether a message or answer area is
; within the current internal process.
;       call:       return:
; w0                unchanged
; w1                cur
; w2                mess (or answer)
; w3    link        destroyed

~~b.g24~~                     ; begin
w.d18:rs   w3   g0            ;
        rl   w1   b1          ;
        rl   w2   x1+a29      ;   mess:= save w1(cur);
        al   w3   x2+14       ;
g1:     ~~sl   w2   (x1+a17)~~    ;   if mess < first addr(cur)
        ~~sl   w3   (x1+a18)~~    ;   or mess + 14 >= top addr(cur)
        ~~jl        c29~~    ok   ;   then goto internal 3;
        ~~jl       (g0)~~         ;
   ~~g0: 0~~                  ;
e.                            ; end
                                                        begin here:
; procedure check event(proc, addr, sorry)
; comment: checks whether an address is a buffer address
; in the event queue of a given internal process.
;       call:       return:
; w0                unchanged
; w1    proc        proc
; w2    addr        addr
; w3    link        link

b.g24                         ; begin
w.d19:rs   w2   g1            ;
```

```
1024         al   w2   x1+a15     ;    buf:= event q(proc);
1026     g0: rl   w2   x2+0       ; next: buf:= next (buf);
1028         sn   w2   x1+a15     ;    if buf = event q (proc)
1030         jl        x3+0       ;    then goto sorry;
1032         se   w2  (g1)        ;    if buf <> addr
1034         jl        g0         ;    then goto next;
1036         jl        x3+2       ;
1038     g1:  0                   ;
1040   e.                         ; end
~~1040~~
1040   ; ~~procedure log buf (buf);~~
1040   ; comment: if logmode is on, the event is output as one single
1040   ; 60 character block onto the monitor log tape (9 track). there
1040   ; is no check on transfer errors except that end of tape or
1040   ; intervention will stop the logging.
1040   ;           call:        return:
1040   ;  w0                    destroyed
1040   ;  w1                    unchanged
1040   ;  w2        buf         buf
1040   ;  w3        link        destroyed
1040
1040   b.g24                      ; begin
1040   w.g0 = b9 + 8              ; log mode address
1040     g1 = k  , 0, r.20        ; event block
1080     g2 = k-2, 0, r.4         ; zero name
1088     g3 = k  , 1<23+1<18      ; intervention+end_of_tape
1090     j36= k                   ; intersegment ref
1090
1090     d75:rl   w0   g0         ;    if log mode = 0
1092         sn   w0    0         ;    then return;
1094         jl        x3         ;    tape:= log mode;
1096
1096     g4: io   w0  (g0)        ; wait:  busy:= sense (tape,status);
1098         sx         1         ;    if busy then goto wait;
1100         jl        g4         ;    if end_of_tape or local
1102         sz   w0  (g3)        ;    then return;
1104         jl        x3         ;
1106
1106         ds   w3   g1+2       ; build event block:
1108         jl   w3   d7         ;    block(0):= buf;
1110         al   w2    0         ;    time(irr,usec);
1112         aa   w3   b13+2      ;    block(2):= return address;
1114         ds   w3   g1+6       ;    block(4:6):= clock+usec;
1116         rl   w2   g1         ;
1118         rl   w2   x2+6       ; copy names:
1120         sh   w2    0         ;    block(8:14):= if sender(buf) > 0
1122         al   w2   g2         ;       then name(sender)
1124         dl   w0   x2+4       ;       else 0;
1126         ds   w0   g1+10      ;
1128         dl   w0   x2+8       ;
1130         ds   w0   g1+14      ;
1132         rl   w2   g1         ;    res:= abs(receiver(buf));
1134         rl   w2   x2+4       ;
1136         sh   w2   -1         ;    block(16:22):= if res > 8
1138         ac   w2   x2         ;       then name (receiver)
1140         al   w3   x2         ;       else res;
1142         sz   w3   -8         ;
1144         dl   w0   x2+4       ;    comment: if res <= 8
1146         ds   w0   g1+18      ;             then block(18:22)
1148         dl   w0   x2+8       ;             is undefined;
1150         ds   w0   g1+22      ;
1152         rl   w2   g1         ; copy event:
1154         dl   w0   x2+10      ;
1156         ds   w0   g1+26      ;    move 8 last words from event buf
1158         dl   w0   x2+14      ;          to the event output block;
1160         ds   w0   g1+30      ;
1162         dl   w0   x2+18      ;
1164         ds   w0   g1+34      ;
1166         dl   w0   x2+22      ;
1168         ds   w0   g1+38      ;
1170
1170         rl   w3   g0         ;    transfer last (tape);
```

```
       al  w0  g2         ;   transfer first(tape);
       io  w0  x3+5       ;
       al  w0  g1         ;   return:
       io  w0  x3+17      ;   comment: transfer is odd parity;
       rl  w2  g1         ;
       jl      (g1+2)     ;
e.                        ; end;

; comment: the following utility procedures are used by external
; processes during input/output;

b. g69

w.g3: am       1          ; result 5: result:= 5
  g4: am       1          ; result 4:     or    4
  g5: am       1          ; result 3:     or    3
  g6: am       1          ; result 2:     or    2
  g7: al  w0   1          ; result 1:     or    1;
      rl  w2   b18        ;
      rs  w0   x2+4       ;   receiver(buf):= result;
      ~~jl  w3   d75        ;   log buf;~~
      jl  w3   d5         ;   remove(buf);
      rl  w1   b1         ;
      al  w1   x1+a15     ;
      jl  w3   d6         ;   link(event q(cur), buf);
      jl       (b20)      ;   goto return;


; procedure check user
; comment: checks whether an external process is used
; by the current internal process. if the external is reserved
; it is also checked whether it is reserved by the current
; internal process.
;       call:     return:
; w0              destroyed
; w1   cur        cur
; w2   buf        buf
; w3   link       link

b.i24                    ; begin
w.g14:am       (b19)     ;
      rl  w0   a52       ;
      se  w0   0         ;   mask:=if reserved(proc)<>0
      jl       i0        ;   then reserved(proc)

; procedure check user only
; comment: works as check user but ignores a possible reservation.
; note: entry point is g14+8 and must not be changed

      am       (b19)     ;   else user(proc);
      rl  w0   a53       ;   bit:=identification(cur);
  i0: so  w0 (x1+a14)    ;   if mask(bit)=0
      jl       g6        ;   then goto result 2;
      jl       x3+0      ;
                         ; end

; procedure check reservation
; comment: checks whether an external process is reserved
; by the current internal process.
;        call:     return:
; w0               reserved
; w1    cur        cur
; w2    buf        buf
; w3    link       link

                         ; begin
w.g15:am       (b19)     ;
      rl  w0   a52       ;   mask:=reserved(proc);
jl i0 ~~so  w0 (x1+a14)~~    ;   bit:=identification(cur);
      jl       g6        ;   if mask(bit)=0
      jl       x3+0      ;   then goto result 2;
```

```
1238  e.                         ; end
1238
1238  ; procedure check operation(oper mask, mode mask)
1238  ; comment: checks whether the operation and mode are
1238  ; within the repertoire of the receiver. the legal values are
1238  ; defined by two bitpatterns in which bit i=1 indicates
1238  ; that operation (or mode) number i is allowed. if the
1238  ; operation is odd, it is checked whether the input/output
 238  ; area is within the internal process.
1238  ;      call:        return:
1238  ; w0   oper mask    destroyed
1238  ; w1   mode mask    destroyed
1238  ; w2   buf          buf
1238  ; w3   link         destroyed
1238
1238  b.i24                      ; begin
1238  w.g16:rs   w3   i0         ;
1240        bz   w3   x2+9       ;
1242        ls   w1   x3+0       ;
1244        bz   w3   x2+8       ;
1246        ls   w0   x3+0       ;
1248        sh   w0   -1         ;    if mode mask(mode(buf))=0
1250        sl   w1    0         ;    or oper mask (operation(buf))=0
1252        jl        g5         ;    then goto result 3;
1254        so   w3   1          ;
1256        jl       (i0)        ;
1258        rl   w1   b1         ;
 260        dl   w0   x2+12      ;    if odd(operation(buf))
1262        sl   w3  (x1+a17)    ;    and (first addr(buf)<first addr(cur)
1264        sl   w0  (x1+a18)    ;    or last addr(buf)>=top addr(cur)
1266        jl        g5         ;
1268        sh   w0   x3-2       ;    or first addr(buf)>last addr(buf))
1270        jl        g5         ;    then goto result 3;
1272        la   w3   g50        ;    make first and
1274        la   w0   g50        ;    last address in
1276        ds   w0   x2+12      ;    message even;
1278        jl       (i0)        ;
1280  ~~i0: 0~~                  ;
1282  ~~e.~~                     ; end
1282
1282  ; procedure link operation
1282  ; comment: links a message to the receiver and
1282  ; returns to the receiver if it is the only message in
1282  ; the queue. otherwise it returns to the sender.
1282  ;      call:     return:
1282  ; w0             operation
 282  ; w1             proc
1282  ; w2   buf       buf
1282  ; w3   link      link
1282
1282  ~~b.i24~~                    ; begin
1282  w.g17:rs   w3   i0          ;
1284        am       (b19)        ;
1286        al   w1   a54         ;
1288        jl   w3   d6          ;    link(mess q(proc),buf);
1290        se   w3   x1+0        ;    if old last=mess q(proc)
1292        jl       (b20)        ;    then goto return;
1294        bz   w0   x2+8        ;
1296        al   w1   x1-a54      ;
1298        jl       (i0)         ;
1300    i0: 0                     ;
1302  e.                          ; end
1302
1302  ; procedure deliver result(result)
1302  ; comment: stores the result and answer of an input/output
 302  ; operation in a buffer, removes the buffer from the
1302  ; message queue and delivers it as an answer to the
1302  ; sender.
1302  ;        call:     return:
1302  ; w0     result    destroyed
1302  ; w1               destroyed
1302  ; w2               destroyed
```

w3:



~~al w3 = 63~~
~~la w3~~

mode

la. w3 i1. ; select mode only

i1: 63 ; to select mode bits only

```
1302  ; w3    link     destroyed
1302
1302  b.i24                      ; begin
1302  w.g18:al  w0  1            ; comment: result=1;
1304    g19:rl  w2  b18          ; comment: result=w0;
1306        rs  w3  i0           ;
1308        rs  w0  x2+4         ;   receiver(buf):=result;
1310        rl  w0  g20          ;
 312        la  w0  g51          ;
1314        rs  w0  x2+8         ;   word(buf+8):=status(0:11);
1316        dl  w1  g22          ;   word(buf+10):=bytes;
1318        ds  w1  x2+12        ;   word(buf+12):=characters;
1320        dl  w1  g24          ;   word(buf+14):=file;
1322        ds  w1  x2+16        ;   word(buf+16):=block;
1324        jl  w3  d15          ;   deliver answer(buf);
1326        jl     (i0)          ;
1328    i0: 0                    ;
1330  e.                         ; end
1330
1330  ; input/output answer:
1330  w.g20: 0  ; status
1332    g21: 0  ; bytes
1334    g22: 0  ; characters
1336    g23: 0  ; file count
1338    g24: 0  ; block count
1340    b27= k  ; simstatus pointer (not used)
1340
 340
1340  ; procedure next operation
1340  ; comment: examines the message queue of the receiver and
1340  ; returns to the receiver if there is a message from a
1340  ; not-stopped sender. otherwise it returns to the current
1340  ; internal process.
1340  ;      call:    return:
1340  ; w0            oper
1340  ; w1            proc
1340  ; w2            buf
1340  ; w3   link     sender
1340
1340  b.i24                      ; begin
1340  w.g25:rs  w3  i2           ;
1342        jl  w3  g64          ;   examine queue(
1344        jl      c33          ;     dummy interrupt);
1346        jl     (i2)          ;
1348    i2: 0                    ;
1350  e.                         ; end
.350
1350  ; procedure examine queue(queue empty)
1350  ;      call:    return:
1350  ; w0            operation
1350  ; w1            proc
1350  ; w2            buf
1350  ; w3   link     sender
1350
1350  b.i24                      ; begin
1350  w.g64:rs  w3  i2           ;
1352    i0: rl  w1  b19          ; exam q:proc:=current receiver;
1354        rl  w2  x1+a54       ;   buf:=next(mess q(proc));
1356        sn  w2  x1+a54       ;   if buf=mess q(proc)
1358        jl     (i2)          ;   then goto queue empty;
1360        rs  w2  b18          ;
1362        rl  w3  x2+6         ;   internal:=sender(buf);
1364        sh  w3  -1           ;
1366        jl      i1           ;   if internal<0
1368        bz  w0  x3+a13       ;   or state(internal)=stopped
.370        sz  w0  a105         ;   then
1372        jl      i4           ;   begin
1374        bz  w0  x2+8         ;   if operation (buf) is even
1376    i3: am     (i2)          ;   then return;
1378        jl      2            ;
1380    i4: bz  w0  x2+8         ;   no operation;
1382        so  w0  1            ;   goto exam q;
```

```
1384          jl       i3          ;
1386    i1:   jl   w3  g26         ;    end;
1388          jl       i0          ;
1390    i2:   0                    ;
1392   e.                          ; end
1392            am     3           ;
1392   ; procedure no operation
1392   ;      call:    return:
 392   ; w0            destroyed
1392   ; w1            destroyed
1392   ; w2            destroyed
1392   ; w3  link      destroyed
1392
1392   b.i24                       ; begin
1392   w.g26:al   w0   1           ;
1394     g27:al   w1   0           ;
1396         rs   w1   g20         ;    status:=
1398     g28:rs   w1   g21         ;    bytes:=
1400         rs   w1   g22         ;    character:=0;
1402         jl        g19         ;    deliver result(1);
1404   e.                          ; end
1404
1404   ; procedure disconnected device
1404   ;      call:    return:
1404   ; w0            destroyed
1404   ; w1            destroyed
1404   ; w2            destroyed
 404   ; w3  link      destroyed
1404
1404   ~~b.i24~~                   ; begin
1404   ~~w.g29:al   w0   4~~   ok  ;    status:=bytes:=characters:=0;
1406   ~~      jl        g27~~     ;    deliver result(4);
1408   ~~e.~~                      ; end
1408
1408   ; procedure sense device
1408   ;      call:  return:
1408   ; w0          destroyed
1408   ; w1          destroyed
1408   ; w2          destroyed
1408   ; w3  link    destroyed
1408
1408   b.i24                       ; begin
1408   w.g30:am       (b19)        ;
1410         io   w0  (a50)        ;    status:=sense(device(proc));
1412         sx        2.11        ;    if ex<>0 then
1414         jl        g29         ;    disconnected device else
 416         rs   w0   g20         ;
1418         al   w1   0           ;
1420         rs   w1   g21         ;    bytes:=
1422         rs   w1   g22         ;    characters:=0;
1424         jl        g18         ;    deliver result(1);
1426                               ;    end;
1426   e.                          ; end
1426
1426   ; procedure increase stop count
1426   ; comment: increases the stop count of the sender by 1.
1426   ;      call:    return:
1426   ; w0            unchanged
1426   ; w1            unchanged
1426   ; w2  buf       buf
1426   ; w3  link      destroyed
1426   ok
1426   ~~b.i24~~                   ; begin
1426   w.g31:rs   w3   i0 3        ;
1428         am       (x2+6)       ;
 430         bz   w3   a12         ;
1432         al   w3   x3+1        ;    stop count(sender(buf)):=
1434         am       (x2+6)       ;    stop count(sender(buf))+1;
1436         hs   w3   a12         ;
1438         jl       (i0) 3       ;
1440     ~~[illegible]~~           ;
1442                               ; end
```

```
TO LEIF
#PRINT 1440

 1440
  1440.    3425698   836 JL          (   1442)
  1442.       2994     0 AW            -1102
  1444.    6227424  1520 RS   W3          ~~1504~~ i3
  1446.    5374146  1312 RL   W2          ~~194~~ b18
  1448.    4988936  1218 BZ   W0   X2+8
  1450.    5316614  1298 RL   W1   X2+6
  1452.   -4980735  2880 SZ   W0          1
  1454.   -6221825  2576 SH   W1         -1
  1456.    3425760   836 JL          (   ~~1504~~) i3
  1458.    6161886  1504 RS   W2          ~~1502~~ i2
  1460. i4:498484[illegible] [illegible] BZ   W0   X1+~~10~~ a12
  1462.    4489217  1096 BS.  W0          1   ~~1462~~



  1464.    6819850  1665 HS   W0   X1+~~10~~ a12
  1466.    5181451  1265 BZ   W3   X1+~~11~~ a13
  1468.   -5505024  2752 SN   W0          0
  1470.   -5046240  2864 SO   W3          ~~32~~ a105
  1472.    3409368   832 JL               ~~1496~~ i1
  1474.    3092488   755 AL   W3   X3+~~0~~ a106
  1476.    7016459  1713 HS   W3   X1+~~11~~ a13
  1478.    5378114  1313 RL   W2   X1+~~66~~ a40
  1480.   -5570376  2736 SE   W3          ~~194~~ a99
  1482.    3409364   832 JL               ~~1492~~ i0
  1484.    2883585   704 AL   W0          1
  1486.    6037508  1474 RS   W0   X2+4
  1488.    3605298   880 JL   W3          ~~815~~ d15
  1490.    6033474  1473 RS   W0   X1+~~66~~ a40
  1492. i0:5312562 [illegible] RL   W1   X1+~~50~~ a34
  1494.    3409332   832 JL               ~~1460~~ i4
  1496. i1:5375[illegible] [illegible] [illegible]L   W2   i2 ~~1502~~
  1498.    6160578  1504 RS   W2          ~~194~~ b18
  1500.    3425760   836 JL          (   ~~1504~~) i3
  1502. i2[illegible]9786     2 AW   X2+1594
  1504. i3: 3064     0 AW            -1032
  1506.    5308416  1296 RL   W1          0
  1508.    1115740   272 LA   W1          1628
  1510
ATT S

FROM LEIF
.    6096180 1488 RS   W1       1332
  1512.     524289   128 BL   W0          1
  1514.    2375874   580 AM          (    194)
  1516.
ATT
```

Type as is!

```
1442
1442  ; procedure decrease stop count
1442  ; comment: the stop count of the sender is decreased by 1
1442  ; if the operation is odd. if stop count becomes zero and the
1442  ; sender is waiting to be stopped, the sender is stopped
1442  ; and the stop count of its parent is decreased by 1.
1442  ; if the parent has stopped its child, an answer is sent to
1442  ; the parent in the buffer defined by the wait address of
 442  ; the child.
1442  ;      call:     return:
1442  ; w0             destroyed
1442  ; w1             destroyed
1442  ; w2             destroyed
1442  ; w3  link       destroyed
1442
1442                                 ; begin
1442  w.g32:rs   w3   i3             ;
1444        rl   w2   b18            ;
1446        bz   w0   x2+8           ;
1448        rl   w3 1 x2+6           ;   internal:=sender(buf);
1450        sz   w0   1              ;   if odd(operation(buf))
1452        sh   w3 1 -1             ;   and internal>=0 then
1454        jl      (i3)             ;   begin
1456        rs   w2   i2 1           ;   save buf:=buf;
1458        bz   w0   x3+a12         ; ~~loop stop~~:
1460        bs.  w0   1  1           ;   stop count(internal):=
1462        hs   w0   x3+a12         ;   stop count(internal)-1;
~~464    i0: se   w0   0              ; exam stop:~~
~~1466       jl        i1 1           ;   if stop count(internal)=0~~
1468        bz   w1 3 x3+a13         ;   ~~and~~ if state(internal)=wait stop
1470        so   w1 3 a105           ;   then   and ~~stop count(internal) = 0~~
1472        jl        i1 3           ;   begin
1474        al   w1 3 x1+a106        ;   child state:=
1476        hs   w1 3 x3+a13         ;   state(internal):=wait start;
1478        rl   w2   x3+a40         ;   buf:=wait address(internal);
1480        rl   w3 1 x3+a34         ;   internal:=parent(internal);
~~1482      bz   w0   x3+a12         ;~~
~~1484      bs.  w0   1              ;   stop count(internal):=~~
~~1486      hs   w0   x3+a12         ;   stop count(internal)-1;~~
1488        se   w1 3 a99            ;   if child state<>wait start parent
1490        jl        ~~i0~~           ;   then goto ~~exam stop;~~ ~~loop stop;~~
1492        al   w0   1              ; child stopped:
1494        rs   w0   x2+4           ;   receiver(buf):=1;
1496        jl   w3   d15            ;   deliver answer(buf);
1498        jl   i0                  ;   end;  ~~goto loop stop;~~ ~~goto loop stop;~~
1498    i1: rl   w2   i2             ;
 500        rs   w2   b18            ;   buf:=save buf;
1502        jl      (i3)             ;   end;
1504    i2: 0                        ;
1506    i3: 0                        ;
1508  e.                             ; end
1508
1508  ; procedure prepare answer(status,count,addr)
1508  ; comment: computes the number of bytes and characters
1508  ; transferred and stores it together with the status bits
1508  ; in the answer. the address points to the last word in
1508  ; which 0,1,2 or 3 characters (as defined by the count)
1508  ; have been transferred.
1508  ;      call:            return:
1508  ; w0   status+count     count
1508  ; w1                    bytes
1508  ; w2   addr             characters
1508  ; w3   link             link
1508
1508  b.i24                          ; begin
 508  w.g33:rl   w1   0
1510        la   w1   g51
1512        rs   w1   g20            ;   status:=status(0:11);
1514        bl   w0   1              ;
1516        am      (b18)            ;
1518        ws   w2   10             ;   diff:=addr-first addr(buf);
1520        al   w1   x2+0           ;
```

```
1522        ls   w2   -1          ;
1524        wa   w2   2           ;   characters:=
1526        wa   w2   0           ;   diff/2*3+count;
1528        sl   w0   1           ;
1530        al   w1   x1+2        ;   bytes:=
1532        ds   w2   g22         ;   if count<1 then diff else diff+2;
1534        jl        x3+0        ;
1536  e.                          ; end
 536
1536  ; procedure exam sender(sender stopped)
1536  ;     call:   return:
1536  ; w0            unchanged
1536  ; w1            unchanged
1536  ; w2            unchanged
1536  ; w3  link      link
1536
1536  b.i24                       ; begin
1536  w.g34:rs   w3   i0          ;
1538        am       (b18)        ;
1540        rl   w3   6           ;   internal:=sender(buf);
1542        sh   w3   -1          ;
1544        jl       (i0)         ;   if internal<0
1546        bz   w3   x3+a13      ;
1548        sz   w3   a105        ;   or state(internal)=stopped
1550        jl       (i0)         ;   then goto sender stopped;
1552        rl   w3   i0          ;
1554        jl        x3+2        ;
 556    ~~i0: 0~~                 ;
1558  [redacted]                  ; end
1558
1558  ; procedure init buffered
1558  ; comment: used in connection with lowspeed devices with an
1558  ; external buffer to make device parameters absolutely
1558  ; addressable before the transfer is initiated.
1558
1558  ;     call:   return:
1558  ; w0            unchanged
1558  ; w1  proc      unchanged
1558  ; w2  buf       buf
1558  ; w3  link      destroyed
1558
1558  [redacted]                  ; begin
1558  w.g35:rs   w3   i0          ;
1560        al   w3   x2+6        ;
1562        rs   w3   g41         ;   sender addr:=buf+6;
1564        rl   w3   x2+6        ;
 566        al   w3   x3+a13      ;
1568        rs   w3   g40         ;   state addr:=sender(buf)+a13;
1570        rl   w3   x1+a50      ;
1572        rs   w3   g42         ;   sense addr:=device(proc);
1574        jl       (i0)         ;
1576    i0: 0                     ;
1578  [redacted]                  ; end
1578
1578  ; procedure wait buffered
1578  ; comment: used in connection with lowspeed devices with an
1578  ; external buffer to save working registers and device parameters
1578  ; before waiting for an interrupt. the procedure must be
1578  ; called as follows:
1578  ;     am       (b19)
1578  ;     ds   w3   a77
1578  ;     jl   w3   g36
1578
1578  [redacted]                  ; begin
1578  w.g36:rl   w2   b19         ;   param 4(proc):=w0;
 580        ds   w1   x2+a75      ;   param 5(proc):=w1;
1582        dl   w1   g44         ;   param 0(proc):=device param 0;
1584        ds   w1   x2+a71      ;   param 1(proc):=device param 1;
1586        dl   w1   g46         ;   param 2(proc):=device param 2;
1588        ds   w1   x2+a73      ;   param 3(proc):=device param 3;
1590        rs   w3   x2+a56      ;   interrupt addr(proc):=link;
1592        jl       (b20)        ;   goto return;
```

```
1594 [redacted]                     ; end
1594
1594  ; procedure continue buffered
1594  ; comment: used in connection with lowspeed devices with an
1594  ; external buffer to restore working registers and device
1594  ; parameters after an interrupt.
1594  ;      call:    return:
1594  ; w0            save w0
 594  ; w1   proc     save w1
1594  ; w2   buf      save w2
1594  ; w3   link     save w3
1594
1594 [redacted]                     ; begin
1594  w.g37:rs   w3   i0            ;
1596        jl   w3   g35           ;   init buffered(proc,buf);
1598        dl   w3   x1+a71        ;   device param 0:=param 0(proc);
1600        ds   w3   g44           ;   device param 1:=param 1(proc);
1602        dl   w3   x1+a73        ;   device param 2:=param 2(proc);
1604        ds   w3   g46           ;   device param 3:=param 3(proc);
1606        dl   w3   x1+a77        ;   w3:=param 7(proc);
1608        dl   w1   x1+a75        ;   w2:=param 6(proc);
1610        jl      (i0)            ;   w1:=param 5(proc);
1612   [redacted]                   ;   w0:=param 4(proc);
1614  e.                            ; end
1614
1614  ; directly addressable parameters for low-speed devices
1614  ; with external buffers :
 614
1614  w.g40:  0  ; address of sender state
1616    g41:  0  ; address of sender in buf
1618    g42:  0  ; device number*64
1620    g43:  0  ; device parameter 0
1622    g44:  0  ; device parameter 1
1624    g45:  0  ; device parameter 2
1626    g46:  0  ; device parameter 3
1628
1628  ; bitpatterns:
1628
1628    g50: 8.7777 7776 ; first 23 bits
1630    g51: 8.7777 0000 ; first 12 bits
1632    g52: 8.0000 7777 ; last 12 bits
1634    g53: 8.0000 0377 ; last 8 bits
1636    g54: 8.0000 0177 ; last 7 bits
1638    g55: 8.0000 0077 ; last 6 bits
1640    g56: 8.3600 0000 ; bits 1-4
1642    g57: 8.3700 0000 ; bits 1-5
 644    g58: 1<22        ; bit 1
1646    g59: 1<21        ; bit 2
1648    g60: 1<20        ; bit 3
1650    g61: 1<19        ; bit 4
1652    g62: 1<18        ; bit 5
1654    g63: 1           ; bit 23
1656
1656  d20= g3, d21= g4, d22= g5, d23= g6, d24= g7, d25=g14, d26=g15
1656  d27=g16, d28=g17, d29=g18, d30=g19, d31=g20, d32=g21, d33=g22
1656  d34=g23, d35=g24, d36=g25, d37=g26, d38=g27, d39=g28, d40=g29
1656  d41=g30, d42=g31, d43=g32, d44=g33, d45=g34, d46=g35, d47=g36
1656  d48=g37, d49=g40, d50=g41, d51=g42, d52=g43, d53=g44, d54=g45
1656  d55=g46, d56=g50, d57=g51, d58=g52, d59=g53, d60=g54, d61=g55
1656  d62=g56, d63=g57, d64=g58, d65=g59, d66=g60, d67=g61, d68=g62
1656  d69=g63, d70=g64
1656
1656  e.
1656
1656  j0=d0, j3=d3, j5=d5, j8=d8, j9=d9, j10=d10, j28=c28, j33=c33
 656
1656  ; procedure set interrupt(address, mask)
1656  ;            call:        return:
1656  ; save w0    mask         mask
1656  ; save w1                 unchanged
1656  ; save w2                 unchanged
1656  ; save w3    address      address
```

```
1656
1656   b.g24                      ; begin
1656   w.e0: rl  w2  x1+a31       ;    address:= save w3(cur);
1658         sn  w2  0            ;    if address <> 0 then
1660         jl      g2           ;
1662         al  w3  x2+14        ;
1664         sl  w2 (x1+a17)      ;    if address < first addr(cur)
1666         sl  w3 (x1+a18)      ;    or address + 14 >= top addr(cur)
 668         jl      c29          ;    then goto internal 3;
1670    g2:  rs  w2  x1+a27       ;    interrupt addr(cur):= address;
1672         rl  w0  x1+a28       ;    new:= save w0(cur);
1674         la  w0  g0           ;
1676         rl  w2  x1+a26       ;    old:= save im(cur);
1678         la  w2  g1           ;
1680         lo  w0  4            ;    im:=
1682         rs  w0  x1+a26       ;    save im(cur):=
1684         ml      0            ;    new(0:2) + old(3:23);
1686         jl     (b20)         ;    goto return;
1688    g0:  8.70000000           ;
1690    g1:  8.07777777           ;
1692   e.                         ; end
1692
1692   ; procedure reset device (device);
1692   ;            call:           return:
1692   ; save w0                    result
1692   ; save w1  device            device
1692   ; save w2                    unchanged
 692   ; save w3                    unchanged
1692
1692   b.g24
1692   w.e1: bz  w0  x1+a22       ; begin
1694         so  w0  1<4          ;    if funct mask(7).cur = 0
1696         jl      c29          ;    then goto internal 3;
1698         al  w0   0           ;    result(cur):= 0;
1700         rs  w0  x1+a28       ;
1702         rl  w2  x1+a29       ;
1704         sz  w2  -256         ;    name table entry:=
1706         jl      g4           ;    first device + 2*device;
1708         wa  w2   4           ;
1710         wa  w2  b4           ;    if entry outside devices
1712         sl  w2 (b5)          ;    then goto result 4;
1714         jl      g4           ;
1716         rl  w1  x2           ;    proc:= word (entry);
1718         jl  w0  c30          ;    goto external single interrupt:
1720    g4:  al  w0   4           ; result 4:
1722         rs  w0  x1+a28       ;    save w0(cur):= 4;
 724         jl     (b20)         ;    goto interrupt return;
1726   e.                         ; end;
1726
1726
1726   ; procedure process description(name, result)
1726   ;            call:       return:
1726   ; save w0                result
1726   ; save w1                unchanged
1726   ; save w2                unchanged
1726   ; save w3  name          name
1726
1726   b.g24                      ; begin
1726   w.e2: jl  w3  d17          ;    check name area;
1728         jl  w3  d11          ;    search name(name, entry)
1730         se  w3 (b7)          ;
1732         rl  w3  x3+0         ;    save w0(cur):=
1734         sn  w3 (b7)          ;    if entry <> name table end
1736         al  w3  0            ;    then name table (entry)
 738         rs  w3  x1+a28       ;    else 0;
 740         jl     (b20)         ;    goto return;
1742   e.                         ; end
1742
1742   ; procedure initialize process(name, result)
1742   ; procedure reserve process(name, result)
1742   ;            call:       return:
1742   ; save w0                result
```

```
1742   ; save w1               unchanged
1742   ; save w2               unchanged
1742   ; save w3  name         name
1742
1742   b.h32, g24                    ; begin
1742   w.e3: am  -1                  ; initialize:
1744     e4: al   w0   1             ; reserve:
1746         rs   w0   g7            ;
 748         jl   w3   d17           ;    check name area;
1750         jl   w3   d11           ;    search name(name, entry);
1752         sn   w3  (b7)           ;    if entry = name table end
1754         jl        g6            ;    then goto result 3;
1756         rl   w3   x3+0          ;    proc:= name table(entry);
1758
1758         am       (x3+0)         ;
1760         jl       (k+2)          ;    goto case kind (proc) of
1762         h3   :   (0: internal process,
1764         h4   :    2: interval clock,
1766         h5   :    4: backing store area,
1768         h6   :    6: rc 4320 drum,
1770         h7   :    8: rc 315 typewriter,
1772         h8   :   10: rc 2000 paper tape reader,
1774         h9   :   12: rc 150 paper tape punch,
1776         h10  :   14: rc 610 line printer,
1778         h11  :   16: rc 405 punched card reader,
1780         h12  :   18: rc 747 magnetic tape,
1782         h13  :   20: dst 401 sense register,
 784         h14  :   22: ixp 401 interrupt register,
1786         h15  :   24: ixp 401 interrupt counter,
1788         h16  :   26: dot 401 static digital output,
1790         h17  :   28: aic 401 analog input,
1792         h18  :   30: dpc 405 alphanumeric display,
1794         h14  :   32: interrupt key,
1796         h12  :   34: rc 749 magnetic tape,
1798         h7   :   36: teletypewriter,
1800         h22  :   38: operator,
1802         h19  :   40: rc 4195 graphic display,
1804         h17  :   42: aic 402 analog input,
1806         h20  :   44: spt 401 set-point terminal,              phil:
1808         h7   :   46: olivetti terminal,                       remove.
1810         h21  :   48: dst 2000,
1812         h16  :   50: dot 402 pulsed digital output,
1814         h23  :   52: www transmission line (rc4124),
1816         h24  :   54: rc 4194 kingmatic plotter,
1818         h25  :   56: ~~rc 3200 transmission terminal,~~ remote
1820         h26  :   58: telex);
 822
1822     h5:; backing store area:
1822         rl   w0   g7            ;
1824         se   w0   1             ;    if reserve then
1826         jl        g0            ;    begin
1828         rl   w0   x1+a23        ;    mask:= catalog mask(cur);
1830         bz   w2   x3+a51        ;    key:= catalog key(proc);
1832         ls   w0   x2+0          ;    if mask(key) = 0
1834         sl   w0   0             ;    then goto result 2;
1836         jl        g5            ;    end;
1838     g0: rl   w2   x3+a53        ;    mask:= user(proc);
1840         sz   w2  (x1+a14)       ;    if mask(identification) = 0 then
1842         jl        g1            ;    begin
1844         rl   w0   x3+a62        ;
1846         se   w0    0            ;    if creator <> 0
1848         ws   w0   x1+a41        ;       and creator <> cur
1850         se   w0    0            ;       then goto result 2;
1852         jl        g5            ;
1854         bz   w0   x1+a20        ;
 856         sn   w0   0             ;    if area claim(cur) = 0
1858         jl        g5            ;    then goto result 2;
1860         bs.  w0   1             ;    area claim(cur):=
1862         hs   w0   x1+a20        ;    area claim(cur) - 1;
1864         lo   w2   x1+a14        ;    user(proc):=
1866         rs   w2   x3+a53        ;    user(proc) or identification(cur);
1868     g1: rl   w0   g7            ;    end;
```

```
         se   w0   1          ;    if -, reserve
         jl        g3         ;    then goto result 0
         jl        g2         ;    else goto reserve;

~~h21: ; [illegible] 2000~~

   h8:  ; rc 2000 paper tape reader:
   h9:  ; rc 150 paper tape punch:
~~h23: ; rc 4134 [illegible]~~
~~h25: ; rc 3200~~
~~h26: ; telex~~
         rl   w2   x3+a52     ;
         lo   w2   x1+a14     ;
         ws   w2   x1+a14     ;
         se   w2   0          ;
         jl        g8         ;
         rs   w2   x3+a70     ;
         rs   w2   x3+a71     ;    if res(proc)=0 or res(proc)=ident(cur) then
         rs   w2   x3+a75     ;    state(proc):=mode(proc):=word(proc):=0;

   h10:; rc 610 line printer:
   h11:; rc 405 punched card reader:
   h12:; rc 747 magnetic tape:
       ; rc 749 magnetic tape:
~~h14:; ixp 401 interrupt register:~~  } ok
~~    ; interrupt key:~~
~~h15:; ixp 401 interrupt counter:~~
   h16:; dot 401 static digital output:
       ; dot 402 pulsed digital output:
   h18:; dpc 405 alphanumeric display:
~~h19:; rc 4195 graphic display:~~
~~h20:; spt 401 set-point terminal:~~
~~h24:; rc 4194 kingmatic plotter~~
   g8: rl   w2   x3+a53     ;    mask:= user(proc);
       so   w2 (x1+a14)     ;    if mask(identification(cur)) = 0
       jl        g5         ;    then goto result 2;

   g2:                      ; reserve:
       rl   w2   x3+a52     ;    mask:= reserved(proc);
       sz   w2 (x1+a14)     ;    if mask(identification(cur)) = 1
       jl        g3         ;    then goto result 0;
       se   w2   0          ;    if mask <> 0
       jl        g4         ;    then goto result 1;
       lo   w2   x1+a14     ;    reserved(proc):=
       rs   w2   x3+a52     ;    reserved(proc) or identification(cur);
       jl        g3         ;    goto result 0;

   h22: ; operator:

   h3:  ; internal process:
   h4:  ; interval clock:
   h6:  ; rc 4320 drum:
   h13: ; dst 401 sense register:
   h17: ; aic 401 analog input:
        ; aic 402 analog input:

~~       rl   w0   g7         ;    if reserve then~~
~~       sn   w0   1          ;    goto result 2;~~
~~       jl        g5~~

   g3: am        -1         ; result 0: result:= 0
   g4: am        -1         ; result 1:       or  1
   g5: am        -1         ; result 2:       or  2
   g6: al   w0   3          ; result 3:       or  3;
       rs   w0   x1+a28     ;   save w0(cur):= result;
       jl       (b20)       ;   goto return;
   g7: 0                    ;

   h7:  ; rc 315 typewriter
        ; olivetti terminal

       rl   w0   g7         ;    if initialize then
```

```
1936         se   w0    1         ;   goto result 0;
1938         jl         g3        ;
1940
1940    g9:  bz   w0   x1+a22     ; test function mask:
1942         sz   w0   1<3        ;   if funct mask (cur,bit 8) = 1
1944         jl        g2         ;   then goto reserve
1946         jl        g5         ;   else goto result 2:
1948    e.                        ; end
 948
1948    ; procedure release process(name)
1948    ;           call:       return:
1948    ; save w0               unchanged
1948    ; save w1               unchanged
1948    ; save w2               unchanged
1948    ; save w3  name         name
1948
1948    b.g24                     ; begin
1948    w.e5: jl   w3   d17       ;   check name area;
1950         jl   w3   d11        ;   search name(name, entry);
1952         sn   w3  (b7)        ;   if entry = name table end
1954         jl       (b20)       ;   then goto return;
1956         rl   w3   x3+0       ;   proc:= name table(entry);
1958         rl   w2   x3+0       ;
1960         sn   w2   0          ;   if kind(proc) <> 0 then
1962         jl       (b20)       ;   begin
1964         rl   w2   x1+a14     ;   mask:= reserved(proc);
1966         lo   w2   x3+a52     ;   mask(identification(cur)):= 0;
 968         lx   w2   x1+a14     ;   reserved(proc):= mask;
1970         rs   w2   x3+a52     ;   end;
1972         jl       (b20)       ;   goto return;
1974    e.                        ; end
1974
1974    ; procedure include user(name, device, result)
1974    ; procedure exclude user(name, device, result)
1974    ;           call:      return:
1974    ; save w0              result
1974    ; save w1  device      device
1974    ; save w2              unchanged
1974    ; save w3  name        name
1974
1974    b.g24                     ; begin
1974    w.e6: am        -1        ; include:
1976      e7: al   w0   1         ; exclude:
1978         rs.  w0   g4.        ;
1980         jl   w3   d17        ;   check name area;
1982         jl   w3   d11        ;   search name(name, entry);
 984         sl   w3  (b6)        ;   if entry < first internal
1986         sn   w3  (b7)        ;   or entry = name table end
1988         jl.       g2.        ;   then goto result 3;
1990         rl   w3   x3+0       ;   child:= name table(entry);
1992         se   w1  (x3+a34)    ;   if cur <> parent(child)
1994         jl.       g2.        ;   then goto result 3;
1996         rl   w2   x1+a29     ;   device:= save w1(cur);
1998         ls   w2   1          ;
2000         wa   w2   b4         ;   entry:= 2 * device + first device;
2002         sl   w2  (b4)        ;   if entry < first device
2004         sl   w2  (b5)        ;   or entry >= first area
2006         jl.       g3.        ;   then goto result 4;
2008         rl   w2   x2+0       ;   proc:= name table(entry);
2010         rl   w0   x2+a50     ;
2012         sh   w0   -1         ;   if device no(proc) < 0
2014         jl.       g3.        ;   then goto result 4;
2016         rl   w0   x2+a53     ;   mask:= user(proc);
2018         so   w0  (x1+a14)    ;   if mask(identification(cur)) = 0
2020         jl.       g1.        ;   then goto result 2;
2022         lo   w0   x3+a14     ;   mask(identification(child)):=
2024         rl.  w1   g4.        ;   if include
2026         se   w1   0          ;   then 1
2028         lx   w0   x3+a14     ;   else 0;
2030         rs   w0   x2+a53     ;   user(proc):= mask;
2032         la   w0   x2+a52     ;   reserved(proc):=
2034         rs   w0   x2+a52     ;   reserved(proc) and user(proc);
```

```
2036    g0: am        -2           ; result 0: result:= 0
2038    g1: am        -1           ; result 2:      or   2
2040    g2: am        -1           ; result 3:      or   3
2042    g3: al   w0   4            ; result 4:      or   4;
2044        rl   w1   b1           ;
2046        rs   w0   x1+a28       ;   save w0(cur):= result;
2048        jl       (b20)         ;   goto return;
2050    g4: 0
 050[redacted]
             ;

2052   e.                          ; end
2052
2052   ; procedure wait answer(buf, answer, result)
2052   ;            call:      return:
2052   ; save w0               result
2052   ; save w1  answer       answer
2052   ; save w2  buf          buf
2052   ; save w3               unchanged
2052
2052   b.g24                       ; begin
2052   w.e9: jl  w3  d18           ;   check mess area;
2054         rl  w2  x1+a30        ;   buf:= save w2(cur);
2056         ~~al  w1  b8~~            ;
2058         jl  w3  d12-2         ;   check buf(mess pool, buf,
2060         jl      c29           ;                    internal 3);
2062         rl  w3  b1            ;
2064         se  w3 (x2+6)         ;   if sender(buf) <> cur
 066         jl      c29           ;   then goto internal 3;
2068         rl  w0  x2+4          ;
2070         sl  w0  6             ;
2072         jl.     g0.           ;   if receiver(buf)>5
2074         sl  w0  1             ;   or receiver(buf)<1
2076         jl.     g1.           ;   then
2078    g0:  al  w0  a103          ;   remove internal(wait answer, buf);
2080         jl  w3  d9            ;
2082         jl     (b20)          ;   else
2084    g1:  rs  w0  x3+a28        ;   begin
2086         bz  w1  x3+a19        ;   save w0(cur):= receiver(buf);
2088         al  w1  x1+1          ;   buf claim(cur):=
2090         hs  w1  x3+a19        ;   buf claim(cur) + 1;
2092         al  w1  x2+8          ;
2094         rl  w2  x3+a29        ;
2096         jl  w3  d14           ;   move mess(buf + 8, answer);
2098         al  w2  x1-8          ;
2100         jl  w3  d5            ;   remove(buf);
2102         ~~al  w1  b8~~            ;
 104         jl  w3  d13-2         ;   release buf(mess pool, buf);
2106         jl     (b20)          ;   end;
2108   e.                          ; end                          Leif
2108
2108   ; procedure wait message(name, mess, buf, result)
2108   ;            call:      return:
2108   ; save w0               result
2108   ; save w1  mess         mess
2108   ; save w2               buf
2108   ; save w3  name         name
2108
2108   b.g24                       ; begin
2108   w.e10:jl  w3  d17           ;   check name area;
2110         jl  w3  d18           ;   check mess area;
2112         al  w2  x1+a15        ;   buf:= event q(cur);
2114    g0:  rl  w2  x2+0          ; next: buf:= next(buf);
2116         rs  w2  b18           ;
2118         sn  w1 (x2+4)         ;   if receiver(buf) = cur
2120         jl.     g1.           ;   then goto found;
 122         se  w2  x1+a15        ;   if buf <> event q(cur)
2124         jl.     g0.           ;   then goto next;
2126         al  w0  a102          ;
2128         jl  w3  d9            ;   remove internal(wait mess, irrelevant);
2130         jl     (b20)          ;   goto return;
2132    g1:  rl  w3  x2+6          ; found:
2134         rs  w3  x1+a28        ;   save w0(cur):= sender(buf);
```

```
2136         rs   w2   x1+a30      ;    save w2(cur):= buf;
2138         sh   w3   0           ;
2140         jl.       g2.         ;
2142         rl   w2   x1+a31      ;    if sender(buf) > 0
2144         dl   w1   x3+4        ;    then
2146         ds   w1   x2+2        ;    move (4) words
2148         dl   w1   x3+8        ;    from(sender(buf) + 2)
2150         ds   w1   x2+6        ;    to (name);
```

ac w1 (x2+4)
rs w1 x2+4

```
2152         rl   w2   b18         ;
2154    g2:  al   w1   x2+8        ;
2156         rl   w2   b1          ;
2158         rl   w2   x2+a29      ;
2160         jl   w3   d14         ;    move mess(buf + 8, mess);
2162         al   w2   x1-8        ;
2164         jl   w3   d5          ;    remove(buf);
2166         ~~rl   w1   x2+4~~        ;
2168         ~~ac   w1   x1+0~~        ;
2170         ~~rs   w1   x2+4~~        ;    receiver(buf):= -receiver(buf):
2172         jl        (b20)       ;    goto return;
2174    e.                         ; end
2174
2174    ; procedure send answer(buf, answer, result)
2174    ;            call:      return:
2174    ; save w0   result      result
2174    ; save w1   answer      answer
2174    ; save w2   buf         buf
2174    ; save w3               unchanged
2174
2174    b.g24                      ; begin
2174    w.e11:jl   w3   d18        ;    check mess area;
2176         rl   w2   x1+a30      ;    buf:= save w2(cur);
2178         ~~[illegible]   w1   e8~~      ;
2180         jl   w3   d12         ;    check buf(mess pool, buf,
2182         jl        c29         ;              internal 3);
2184         ac   w3   (x2+4)      ; check state:
2186         rl   w1   b1          ;
2188         se   w3   x1+0        ;    if receiver(buf) <> -cur
2190         jl        c29         ;    then goto internal 3;
2192         rl   w0   x1+a28      ;    result:= save w0(cur)
2194         sl   w0   1           ;    if result < 1
2196         sl   w0   6           ;    or result > 5
2198         jl        c29         ;    then goto internal 3;
2200         rs   w0   x2+4        ;    receiver(buf):= result;
2202         rl   w1   x1+a29      ;
2204         al   w2   x2+8        ;
2206         jl   w3   d14         ;    move mess(answer, buf + 8);
2208         al   w2   x2-8        ;
2210         jl   w3   d15         ;    deliver answer(buf);
2212         jl        (b20)       ;    goto return;
2214    e.
2214
2214    ; procedure wait event(last buf, next buf, result)
2214    ;            call:      return:
2214    ; save w0               result
2214    ; save w1
2214    ; save w2   last buf    next buf
2214    ; save w3
2214
2214    b.g24                      ; begin
2214    w.e12:rl   w2   x1+a30     ;    last buf:= save w2(cur);
2216         se   w2   0           ;    if last buf = 0
2218         jl.       g1.         ;    then last buf:= event q(cur)
2220         al   w2   x1+a15      ;    else check event(cur, last buf, internal 3);
2222    g0:  rl   w2   x2+0        ;    next buf:= next(last buf);
2224         sn   w2   x1+a15      ;    if next buf = event q(cur)
2226         jl.       g2.         ;    then remove internal(wait event, irrelevant)
2228         rs   w2   x1+a30      ;    else
2230         rl   w0   x2+4        ;    begin
2232                               ;          save w2(cur):= next buf;
2232         sz   w0   -8          ;          save w0(cur):=
2234         am        -1          ;          if receiver(buf)>=0
2236         al   w0   1           ;          and receiver(buf)<8
```

```
2238          rs   w0   x1+a28      ;         then 1 else 0;
2240          jl       (b20)        ;    end;
2242    g1:   jl   w3   d19         ;
2244          jl        c29         ;
2246          jl.       g0.         ;
2248    g2:   al   w0   a104        ;
2250          jl   w3   d9          ;
2252          jl       (b20)        ;
`254  e.                            ; end
¿254
2254  ; procedure get event(buf)
2254  ;              call:     return:
2254  ; save w0                unchanged
2254  ; save w1                unchanged
2254  ; save w2   buf          buf
2254  ; save w3                unchanged
2254
2254  b.g24                         ; begin
2254  w.e13:rl   w2   x1+a30        ;    buf:= save w2(cur);
2256          jl   w3   d19         ;
2258          jl        c29         ;    check event(cur, buf, internal 3);
2260          jl   w3   d5          ;    remove(buf);
2262          rl   w3   x2+4        ;    if receiver(buf)>=0
2264          sz   w3   -8          ;    and receiver(buf)<8 then
2266          jl.       g0.         ;    begin
2268          bz   w3   x1+a19      ;
2270          al   w3   x3+1        ;    buf claim(cur):=
`272          hs   w3   x1+a19      ;    buf claim(cur) + 1;
¿274          ~~al   w1   b8~~          ;
2276          jl   w3   d13 2       ;    release buf(mess pool, buf);
2278          jl       (b20)        ;    end
2280    ~~g0: rl   w3   x2+4~~        ;
2282          ac   w3   x3+0        ;    else
2284          rs   w3   x2+4        ;    receiver(buf):= -receiver(buf);
2286          jl       (b20)        ;
2288  e.                            ; end
2288
2288  ; procedure type w0
2288  ; procedure type w1
2288  ; procedure type w2
2288  ; procedure type w3
2288  ; comment: prints the contents of a working register as a
2288  ; signed integer preceded by the letter w, x, y, or z,
2288  ; respectively. only used during testing of the monitor.
2288
2288  b.g24                         ; begin
 288  w.e14:am        -1            ;    register no:= 0
¿290    e15:am        -1            ;                  or  1
2292    e16:am        -1            ;                  or  2
2294    e17:al   w1   3             ;                  or  3;
2296          jl      c29           ;
2298  e.                            ; end
2298
2298  ; procedure get clock(time)
2298  ;            call:   return:
2298  ; save w0            time
2298  ; save w1            time
2298  ; save w2            unchanged
2298  ; save w3            unchanged
2298
2298  b.g24                         ; begin
2298  w.e18:jl   w3   d7            ;    time(slice, usec);
2300          ad   w0   -24         ;    new time:=time + usec;
2302          aa   w0   b13+2       ;    save w0(cur):=new time(0:23);
2304          ds   w0   x1+a29      ;    save w1(cur):=new time(24:47);
 306          jl      (b20)         ;    goto return;
¿308  e.                            ; end
2308
2308  ; procedure set clock(time)
2308  ;            call:   return:
2308  ; save w0 time       time
2308  ; save w1 time       time
```

g0: ac w3 (x2+4)
rs w3 x2+4

```
2308  ; save w2          unchanged
2308  ; save w3          unchanged
2308
2308  b.g24                      ; begin
2308  w.e19:bz   w0   x1+a22     ;   mask:=func mask(cur);
2310        so   w0   1<4        ;   if mask(7)=0
2312        jl        c29        ;   then goto internal 3;
2314        al   w0   0          ;
2316        rs   w0   b12        ;
2318        dl   w0   x1+a29     ;   usec:=0;
2320        ds   w0   b9+2       ;   time base:= save w0 w1;
2322        ds   w0   b13+2      ;   time(0:23):=save w0(cur);
2324        jl      (b20)        ;   time(24:47):=save w1(cur);
2326                             ;   goto return;
2326  e.                         ; end
2326
2326  ; procedure modify backing store(name, device mask, result):
2326  ;            call:          return:
2326  ; save w0                   result
2326  ; save w1 device mask device mask
2326  ; save w2                   unchanged
2326  ; save w3 name              name
2326
2326  b.g24                      ; begin
2326  w.e36:jl   w3   d17        ;   check name area;
2328        jl   w3   d11        ;   search name(name, entry);
2330        sl   w3  (b6)        ;   if entry < first internal
2332        sn   w3  (b7)        ;   or entry=name table end
2334        jl.       g1.        ;   then goto result 3;
2336        rl   w3   x3+0       ;   child:= name table(entry);
2338        se   w1  (x3+a34)    ;   if cur<>parent(child) then
2340        jl.       g1.        ;   goto result 3;
2342        bz   w0   x3+a13     ;   if state(child)<>waiting for start by parent
2344        se   w0   a99        ;   then goto result 2
2346        jl.       g0.
2348        rl   w0   x1+a42     ;   if (device mask(cur) or save w1(cur))
2350        so   w0  (x1+a29)    ;   <> device mask(cur)
2352        jl.       g0.        ;   then goto result 2;
2354        rl   w0   x1+a29
2356        rs   w0   x3+a42     ;   device mask(child):= save w1(cur);
2358        rs   w0   x3+a43     ;   selection mask(child):= save w1(cur);
2360    g2: am        -2         ; result 0: result:= 0
2362    g0: am        -1         ; result 2:         or 2
2364    g1: al   w0   3          ; result 3:         or 3;
2366        rs   w0   x1+a28     ;   save w0(cur):= result;
2368        jl      (b20)        ;   goto return
 370                             ; end;
2370
2370  ; procedure select backing store(selection mask, result);
2370  ;            call:                return:
2370  ; save w0                         result
2370  ; save w1 selection mask          selection mask
2370  ; save w2                         unchanged
2370  ; save w3                         unchanged
2370                            a29
2370  ~~b.g24~~                   ; begin
2370  w.e37:~~rl   w0   x1+a42~~ ;   if (device mask(cur) or save w1(cur))
2372        ~~so   w0  (x1+a29)~~  ;   <> device mask(cur)
2374        ~~jl.       g0.~~      ;   then goto result 2;
2376        ~~rl   w0   x1+a29~~
2378        rs   w0   x1+a43     ;   selection mask(cur):= save w1(cur);
2380        ~~am        -2~~       ; result 0:  result:= 0
2382    ~~g0: al   w0   2~~        ; result 2:          or 2;
2384        ~~rs   w0   x1+a28~~   ;   save w0(cur):= result;
2386        jl.     ~~(b20)~~ g2.  ;   goto return
 388  e.                         ; end;
2388
2388  ; call of process functions:
2388  ; comment: checks whether parameters are within the current
2388  ; internal process and links it to the process function queue.
2388  ; the process function is activated if it is waiting for a
2388  ; call;
```

```
2388
2388  b.g24
2388  w.e20: ; create entry:
2388    e21: ; look up entry:
2388    e22: ; change entry:
2388         am  a88-22          ;   constant:= catalog entry size - 16
2390    e23: ; rename entry:
2390         am     -4           ;       or    6
2392    e28: ; create internal:
2392
2392    e31: ; modify internal:          or   10
2392         al  w3  10          ;
2394         rs. w2  g5.         ;
2396         rl  w2  x1+a29      ;   first param:= save w1(cur):
2398         wa  w3  4           ;   last param:= first param + constant;
2400         sl  w2 (x1+a17)     ;   if first param < first addr(cur)
2402         sl  w3 (x1+a18)     ;   or last param >= top addr(cur)
2404         jl      c29         ;   then goto internal 3;
2406         jl.     g3.         ;   goto check name;
2408
2408    e30: ; stop internal process:
2408         jl  w3  d17         ;   check name area;
2410         jl  w3  d11         ;   search name(name, entry);
2412         sl  w3 (b6)         ;   if entry < first internal
2414         sn  w3 (b7)         ;   or entry = name table end
2416         jl.     g0.         ;   then goto result 3;
2418         rl  w3  x3+0        ;   child:= name table(entry);
2420         sn  w1 (x3+a34)     ;   if cur <> parent(child) then
2422         jl.     g1.         ;   begin
2424    g0:  am      1           ; result 3: result:= 3
2426  ~~  g6:  al  w0  2           ; result 2:          or 2;~~
2428         rs  w0  x1+a28      ;   save w0(cur):=result;
2430         jl     (b20)        ;   goto return;
2432  ~~  g1:  bz  w3  x1+a19      ;   end;~~
2434         se  w3  0           ;   if buf claim(cur) = 0 then
2436         jl.     g2.         ;   begin
2438         rs  w3  x1+a30      ;     save w2(cur):= 0;
2440         jl     (b20)        ;     goto return;
2442    g2:  al  w3  x3-1        ;   end;
2444         hs  w3  x1+a19      ;   buf claim(cur):= buf(claim) - 1;
2446         rl  w2  b8          ;   buf:= next(mess pool);
2448         jl  w3  d5          ;   remove(buf);
2450         rl  w3 (b6)         ;   proc:= name table(first internal);
2452         ac  w0  x3+0        ;   receiver(buf):= -proc;
2454         ds  w1  x2+6        ;   sender(buf):= cur;
2456         rs  w2  x1+a30      ;   save w2(cur):= buf;
2458         jl.     g4.         ;   goto link call;
2460
2460    e24: ; remove entry:
2460    e25: ; permanent entry:
2460    e26: ; create area:
2460    e27: ; create peripheral:
2460    e29: ; start internal:
2460    e32: ; remove process:
2460    e34: ; generate name:
2460         rs. w2  g5.         ; check name:
2462    g3:  jl  w3  d17         ;   check name area;
2464         rl. w0  g5.         ;   if function=modify internal
2466         se  w0  62          ;   then
2468         jl.     g4.         ;   begin
2470         jl  w3  d11         ;   search name(name,entry);
2472         sl  w3 (b6)         ;   if entry < first internal
2474         sn  w3 (b7)         ;   or entry=name table end
2476         jl.     g4.         ;   then goto link call;
2478         rl  w3  x3+0        ;   child:=name table(entry);
2480         se  w1 (x3+a34)     ;   if cur<>parent(child)
2482         jl.     g4.         ;   then goto link call;
2484         rl  w2  x1+a29      ;   child ic:=word(last param);
2486         rl  w2  x2+10       ;   if child ic<first addr(child)
2488         sl  w2 (x3+a17)     ;   or child ic>=top addr(child)
2490         sl  w2 (x3+a18)     ;   then goto internal 3;
2492         jl      c29         ;   end;
```

OK keep original !!!

```
2494    g4: al   w0   a101       ; link call:
2496        jl   w3   d9         ; . remove internal(wait proc func, irrelevant);
2498        al   w2   x1+a16     ;   elem:= process q(cur);
2500        rl   w1  (b6)        ;   proc:= name table(first internal);
2502        al   w1   x1+a15     ;   head:= event q(proc);
2504        jl   w3   d6         ;   link(head, elem);
2506        al   w1   x1-a15     ;
2508        bz   w0   x1+a13     ;
2510        sn   w0   a102       ;   if state(proc) = wait mess
2512        jl   w3   d10        ;   then link internal(proc);
2514        jl      (b20)        ;   goto return;
2516    g5: 0
2518    e35: ; copy:
2518        dl   w0   x1+a18
2520        sh   w3  (x1+a29)    ;   if first addr(cur)>save w1(cur)
2522        sh   w0  (x1+a31)    ;   or top addr(cur)<=save w3(cur)
2524        jl        c29        ;   then goto internal 3;
2526        rl   w2   x1+a30     ;   buf:= save w2;
2528        ac   w3   x1         ;
2530        se   w3  (x2+4)      ;   if receiver(buf) <> -cur
2532        sn   w1  (x2+4)      ;   and receiver(buf) <> cur
2534        rl   w3   x2+6
2536        sh   w3   -1         ;   or sender(buf)<0 then
2538        jl.       g0.        ;   goto result 3;
2540        bz   w0   x3+a13
2542        sz   w0   a105       ;   if state(sender)=stopped
2544        jl.       g6.        ;   then goto result 2;
2546        dl   w0   x3+a18
2548        sh   w3  (x2+10)     ;   if first addr(sender)>first addr(buf)
2550        sh   w0  (x2+12)     ;   or top addr(sender)<=last addr(buf)
2552        jl.       g0.        ;   then goto result 3;
2554        jl.       g4.        ;   goto link call;
2556
2556    e33=g4  ; monitor log: goto link call;
2556  e.                         ;
2556
2556  b.i0                       ; begin
2556  w.i0: al.  w2   i0.        ; make room:
2558        jl        x3+0       ;   autoloader(end monitor procedures);
2560        jl.       i0.        ; after loading:
2562    j29=k - b127 + 2
2562    k = i0                   ;   goto make room;
2556  e.                         ; end
2556
2556
2556  e.    ; end of monitor segment
```

Leif typed to here.

```
2556
2556
2556  ; segment 3: external processes
2556
2556  s. k = k, h32, g70
2556  w.b127=k, g70, k=k-2
2556
2556  g3 =d20, g4 =d21, g5 =d22, g6 =d23, g7 =d24, g14=d25, g15=d26, g16=d27
2556  g17=d28, g18=d29, g19=d30, g20=d31, g21=d32, g22=d33, g23=d34, g24=d35
2556  g25=d36, g26=d37, g27=d38, g28=d39, g29=d40, g30=d41, g31=d42, g32=d43
2556  g33=d44, g34=d45, g35=d46, g36=d47, g37=d48, g40=d49, g41=d50, g42=d51
2556  g43=d52, g44=d53, g45=d54, g46=d55, g50=d56, g51=d57, g52=d58, g53=d59
2556  g54=d60, g55=d61, g56=d62, g57=d63, g58=d64, g59=d65, g60=d66, g61=d67
2556  g62=d68, g63=d69, g64=d70
2556
2556  ; procedure send message(name, mess, buf)
2556  ;            call:      return:
2556  ; save w0               unchanged
2556  ; save w1  mess         mess
2556  ; save w2               buf
2556  ; save w3  name         name
2556                         ; begin
2556  w.e8: rl   w2   x1+a31     ;   name:= save w3(cur);
2558        al   w3   x2+8       ;
2560        sl   w2  (x1+a17)    ;   if name < first addr(cur)
2562        sl   w3  (x1+a18)    ;   or name + 8 >= top addr(cur)
```

```
         jl       c29         ;   then goto internal 3;

   g1:   jl   w3  d11         ; search: search name(name,entry);
         sn   w3  (b7)        ;   if entry = name table end
         jl.      g2.         ;   then goto unknown;
         rs   w3  x2+8        ;   word (name+8):= entry;
         rl   w3  x3+0        ;   proc:= name table (entry);
   g0:   rs   w3  b19         ; found:
         rl   w1  b1          ;
         rl   w3  x1+a29      ;   mess:= save w1(cur);
         al   w0  x3+14       ;
         sl   w3  (x1+a17)    ;   if mess < first addr(cur)
         sl   w0  (x1+a18)    ;   or mess + 14 >= top addr(cur)
         jl       c29         ;   then goto internal 3;
         bz   w2  x1+a19      ;
         sn   w2  0           ;   if buf claim(cur) = 0
         jl.      g8.         ;   then goto no buffer;
         al   w2  x2-1        ;
         hs   w2  x1+a19      ;   buf claim(cur):= buf claim(cur) - 1;
         rl   w2  b8          ;
         rs   w2  b18         ;   buf:= next(mess pool);
         rs   w2  x1+a30      ;   save w2(cur):= buf;
         dl   w1  x3+2        ;
         ds   w1  x2+10       ;
         dl   w1  x3+6        ;   move 8 message
         ds   w1  x2+14       ;   words to buffer;
         dl   w1  x3+10       ;
         ds   w1  x2+18       ;
         dl   w1  x3+14       ;
         ds   w1  x2+22       ;
         jl   w3  d5          ;   remove(buf);
         rl   w1  b1          ;
         rs   w1  x2+6        ;   sender(buf):= cur;
         rl   w3  b19         ;
         rs   w3  x2+4        ;   receiver(buf):= proc;
         ~~jl   w3  [illegible]~~ ;
         ~~rl   w3  b19~~     ;   ~~[illegible]~~
         am       (x3+0)      ;
         jl.      (2)         ;   goto case kind(proc) of
;        w1 = cur, w2 = buf, w3 = proc
         h3                   ;  (0: internal process,
         h4                   ;   2: interval clock,
         h5                   ;   4: backing store area,
         g3                   ;   6: rc 4320 drum,
         h7                   ;   8: rc 315 typewriter,
         h8                   ;  10: rc 2000 paper tape reader,
         h9                   ;  12: rc 150 paper tape punch,
         ~~h10~~ g3           ;  14: rc 610 line printer,
         ~~h11~~ g3           ;  16: rc 405 punched card reader,
         h12                  ;  18: rc 747 magnetic tape,
         ~~h13~~ g3           ;  20: dst 401 sense register,
         ~~h14~~ g3   g3, ~~[illegible]~~ r.7 ;  22: ixp 401 interrupt register,
         ~~h15~~ g3           ;  24: ixp 401 pulse count,
         ~~h16~~ g3           ;  26: dot 401 static digital output,
         ~~h17~~ g3           ;  28: aic 401 analog input,
         ~~h18~~ g3           ;  30: dpc 405 display,
         ~~h19~~ g3           ;  32: interrupt key,
         h12                  ;  34: rc 749 magnetic tape,
         h7                   ;  36: teletypewriter,
         h22                  ;  38: operator,
Remove
         ~~h1[illegible]~~    ;  ~~40: rc 4195 graphic display,~~
         ~~h17~~              ;  ~~42: aic 402 analog input,~~
         ~~h20~~              ;  ~~44: spt 401 set-point terminal,~~
         ~~h7~~               ;  ~~46: olivetti terminal,~~
         ~~h21~~              ;  ~~48: dct 2000,~~
         ~~h16~~              ;  ~~50: dot 402 pulsed digital output,~~
         ~~h23~~              ;  ~~52: rc 4124 www transmission line,~~
         ~~h24~~              ;  ~~54: rc 4194 kingmatic plotter,~~
         ~~h25~~              ;  ~~56: rc 3200 transmission terminal,~~
         ~~h26~~              ;  ~~58: telex (via rc 4124 (etc));~~

   g2:   rl   w1  b1          ; unknown:
```

```
2702        bz   w2   x1+a19     ;
2704        sn   w2   0          ;    if buf claim(cur) = 0
2706        jl.       g8.        ;    then goto no buffer;
2708        al   w2   x2-1       ;
2710        hs   w2   x1+a19     ;    buf claim(cur):= buf claim(cur) - 1;
2712        rl   w2   b8         ;
2714        rs   w2   b18        ;    buf:= next(mess pool);
2716        rs   w2   x1+a30     ;    save w2(cur):= buf;
2718        rs   w1   x2+6       ;    sender(buf):= cur;
2720   [redacted];
2722        jl        g3         ;     goto result 5;
2724
2724                             ; no buffer:
2724   g8:  rs   w2   x1+a30     ;    save w2(cur):= 0
2726        jl       (b20)       ;    goto return;
2728
2728  ; internal process:
2728
2728  b.i24                      ; begin
2728  w.h3: jl   w3   d16        ;    deliver message(buf);
2730        jl       (b20)       ;    goto return;
2732  e.                         ; end of internal process;
2732
2732  ; interval clock:
2732
2732  b.i24,a0=1<23              ; begin
2732  w.      a0>0
 734    i0: a0>0+a0>2
2736    h4: dl.  w1   i0.
2738        jl   w3   g16        ;    check operation(0,0,2);
2740        dl   w0   x2+12      ;    delay:=doubleword(buf+12);
2742        bz   w1   x2+9
2744        se   w1   0          ;    if mode(buf)<>0 then
2746        jl.       i8.        ;    goto check delay;
2748        al   w0   x3+0
2750        wm.  w0   i9.        ;    delay:=word(buf+10)*10000;
2752    i8:                      ; check delay:
2752        sl   w3   0          ;    if delay < 0
2754        sl   w3   52         ;    or delay > 87 241 523 1
2756        jl        g5         ;    then goto result 3;
2758        sn   w0   0          ;    comment:24 hours + 841.5231 secs:
2760        se   w3   0          ;    if delay = 0
2762        jl.       4
2764        jl        g7         ;    then goto result 1:
2766        rl   w2   b19
2768        al   w1   x2+a54     ;    elem:= mess q(proc);
 770                             ; compare:
2770    i1: rl   w1   x1+0       ;    elem:= next(elem);
2772        sn   w1   x2+a54     ;    if elem = mess q(proc)
2774        jl.       i2.        ;    then goto link;
2776        ss   w0   x1+12      ;    delay:=delay-doubleword(elem+12);
2778        sl   w3   0          ;    if delay>=0 then
2780        jl.       i1.        ;    goto compare;
2782        aa   w0   x1+12      ;    delay:=delay+doubleword(elem+12);
2784        rx   w3   x1+10
2786        rx   w0   x1+12
2788        ss   w0   x1+12      ;    doubleword(elem+12):=doubleword(elem+12)-del
2790        rx   w3   x1+10
2792        rx   w0   x1+12
2794    i2: rl   w2   b18        ; link:
2796        ds   w0   x2+12      ;    doubleword(buf + 12) := delay;
2798        jl   w3   d6         ;    link(elem, buf);
2800    i3: rl   w1   b19        ; wait:
2802        jl   w3   c32        ;    wait interruot(proc);
2804
 804                             ; clock interrupt:
2804  w.c35:rl   w1   b2         ;    if next(timer q) <> timer q then
2806        sn   w1   b2         ;    begin
2808        jl.       i4.        ;    internal:= cur;
2810        rl   w1   b1         ;    remove internal(irrelevant, irrelevant);
2812        jl   w3   d9         ;    link internal(internal);
2814        jl   w3   d10<--     ;    end;
```

```
2816    i4: jl   w3   d7         ;    time(slice, usec);
2818        sh.  w3  (i6.)       ;    if usec <inspection interval
2820        jl.       i3.        ;    then goto wait;
2822        al   w0    0         ;
2824        ac.  w1  (i7.)       ;    time:=time+inspection interval:
2826        aa   w1   b13+2      ;
2828        ds   w1   b13+2      ;    usec:=usec-inspection interval:
2830        wa.  w3   i7.        ;
2832        rs   w3   b12        ;
2834        rl   w1   b19        ;
2836        rl.  w0   i7.
2838        al   w3   -1         ;    delay:=-inspection interval;
2840    i5: al   w2   x1+a54     ; next:
2842        rl   w2   x2+0       ;    elem:= next(mess q(proc));
2844        sn   w2   x1+a54     ;    if elem = mess q(proc)
2846        jl.       i3.        ;    then goto wait;
2848        aa   w0   x2+12      ;    delay:= doubleword(elem + 12):=
2850        sl   w1  (x2+6)      ;    if sender(buf) is removed
2852        ld   w0   -65        ;    then 0 else
2854        ds   w0   x2+12      ;    delay + doubleword(elem + 12);
2856        sn   w3   0          ;    if delay > 0 then
2858        sn   w0   0
2860        sl   w3   1
2862        jl.       i3.        ;    goto wait;
2864        al   w3   1
2866        rs   w3   x2+4       ;    word(elem+4):=1;
2868        ld   w0   -65
2870        ds   w0   x2+10      ;    word(elem+8):=
2872        rx   w0   x2+12      ;    doubleword(elem+12):=0;
2874        jl   w3   d15        ;    deliver answer(elem);
2876        bl   w3   0
2878        bl   w3   6
2880        jl.       i5.        ;    goto next;
2882    i6: a87-1                ;
2884    i7: -a87
2886    i9: 10000
2888    e.                       ; end of interval clock;
```

Leif Typed to here

```
2888
2888    ; backing store area:
2888    ; comment: the backing store can consist of one or more
2888    ; drums and/or disks. from a logical point of view, the backing
2888    ; store can be regarded as one collection of named data areas.
2888    ; each data area occupies a consecutive number of segments on
2888    ; a single backing store device. the segment length is 256 words.
2888    ; a process description of a backing store area has the
2888    ; following format:
 888    ;
2888    ; a10: <kind=4>
2888    ; a11: <name>
2888    ; a50: <device number * 2>,  a51: <catalog key>
2888    ; a52: <reserved>
2888    ; a53: <users>
2888    ; a60: <first segment number>
2888    ; a61: <number of segments>
2888    ;
2888    ; the process description is used to check the validity of
2888    ; a message to the backing store area. if the message is
2888    ; accepted, the device number is used to find a process
2888    ; description of the drum or disk on which the area is
2888    ; stored, and the message is linked to the queue of
2888    ; this device;
2888
2888    b.i24, a0=1<23           ; begin
2888    w.      a0>0+a0>3+a0>5   ;
2890      i0:   a0>0             ;
 892      h5: bz   w0   x2+8     ;
2894          sn   w0   5        ;    if operation(buf)=5
2896          am    g15-g14      ;    then check reservation
2898          jl   w3   g14      ;    else check user;
2900          dl.  w1   i0.      ;
2902          jl   w3   g16      ;    check operation(0.3.5, 0);
2904          rl   w1   b19      ;
```

```
          bz   w0   x2+8         ;
          sn   w0   0            ;   if operation(buf)<>0 then
          jl.       i1.          ;   begin
          rl   w3   x2+12        ;
          al   w3   x3+2         ;   core segments:=
          ws   w3   x2+10        ;   (last addr(buf)+2-first addr(buf))
          as   w3    -9          ;   /512;
          rl   w0   x1+a61       ;   area segments:=
          ws   w0   x2+14        ;   number of segs(proc)-first seg(buf);
          sh   w0  (x1+a61)      ;   if area segments>number of segs(proc)
          sh   w0   0            ;   or area segments<=0
          jl.       i2.          ;   then goto outside area;
          sh   w0   x3-1         ;   number of segs(buf):=
          rl   w3   0            ;   if area segments>=core segments
          rl   w0   x2+14        ;   then core segments else area segments;
          wa   w0   x1+a60       ;   first seg(buf):=
          ds   w0   x2+14        ;   first seg(buf)+first seg(proc);
                                 ;   end;
     i1:  bz   w1   x1+a50       ;
          wa   w1   b4           ;
          rl   w1   x1+0         ;   proc:=
          rs   w1   b19          ;   name table(first device+device no(proc));
          rs   w1   x2+4         ;   receiver(buf):=proc;
          jl.       i5.          ;   goto rc 4320 drum;
   ;      w1=proc       w2=buf       w3=kind
   ; the message have been transformed as follows:
   ;      buf+8 :    <operation> <zero>
   ;      buf+10:    <first storage address>
   ;      buf+12:    <number of segments>
   ;      buf+14:    <first segment no>

     i2:  rl   w1   g62          ; outside area:
          rs   w1   x2+8         ;   word(buf+8):=bit5;
          al   w1   0            ;
          rs   w1   x2+10        ;   word(buf+10):=
          rs   w1   x2+12        ;   word(buf+12):=0;
          jl        g7           ;   goto result 1;

   ; rc 4320 drum used as backing store:
   ; rc 433 disc used as backing store:
   ; process description format:
   ;
   ; a10: <kind=6>
   ; a11: <name=0>
   ; a50: <device number*64>
   ; a52: <reserved=0>
   ; a53: <users=0>
   ; a54: <next message>
   ; a55: <last message>
   ; a56: <interrupt address=c33>
   ; a70: <tries>
~~   ; a71: <operations>~~
~~   ; a72: <errors>~~

   b.j24                        ; begin
   w.i5: jl   w3   g17          ;   link operation;
     j0: sn   w0   0            ; start: if operation(buf)=0
         jl.       j4.          ;   then goto/sense;
         jl   w3   g31          ;   increase/stop count;
         al   w3   0            ;   tries:=0;
     j1: rs   w3   x1+a70       ; repeat:
         rl   w1   x1+a50       ;   device:=device no(proc);
         rl   w3   x2+14        ;
         io   w3   x1+5         ;   transfer(device,first seg(buf));
~~         sx        2.11~~         ;
~~         jl.       j5.~~          ;   if ex<>0 then goto disconnect;
         rl   w3   x2+12        ;
         io   w3   x1+9         ;   transfer(device,number of segs(buf));
~~         sx        2.11~~         ;
~~         jl.       j5.~~          ;   if ex<>0 then goto disconnect;
         rl   w3   x2+10        ;
         sn   w0   5            ;   if operation(buf)=5_
```

```
2998        am      4          ;    then output(device,first addr(buf))
3000        io  w3  x1+13      ;    else input(device,first addr(buf));
3002        ~~ex      2,11~~       ;
3004        ~~jl.     j5.~~        ;    if ex<>0 then goto disconnect;
3006        rl  w1  b19        ;
3008        jl  w3  c32        ;    wait interruot(proc);
3010        ~~rl  w3  x1+a71~~     ;
3012        ~~al  w3  x3+1~~       ;    operations.proc:= op.proc + 1;
~014        ~~rs  w3  x1+a71~~     ;
_016        io  w3  (x1+a50)   ;    status:=sense(device no(proc));
3018        ~~ex      2,11~~       ;
3020        ~~jl.     j5.~~        ;    if ex<>0 then goto disconnect;
3022        rs  w3  g20        ;
3024        se  w3  0          ;    if status<>0
3026        jl.     j3.        ;    then goto ~~error;~~
3028        rl  w1  x2+12      ;
3030        as  w1  8          ;    words:=number of segs(buf)*256;
3032        al  w0  x1+0       ;
3034        as  w0  1          ;    bytes:=words*2;
3036        wa  w1  0          ;    characters:=words*3;
3038        ds  w1  g22        ;
3040        jl  w3  g32        ;    decrease stoo count;
3042        jl  w3  g18        ;    deliver result(1);
3044    j2: jl  w3  g25        ; done: next operation;
3046        jl.     j0.        ;    goto start;
3048
3048
~048
_048    j3: bz  w0  x2+8       ; error:
3050        ~~rl  w3  x1+a72~~     ;    errors:= errors+1;
3052        ~~al  w3  x3+1~~       ;
3054        ~~rs  w3  x1+a72~~     ;
3056        rl  w3  x1+a70     ;
3058        al  w3  x3+1       ;    tries:=tries+1;
3060        sh  w3  2          ;
3062        jl.     j1.        ;    if tries<3 then goto repeat;
3064        jl  w3  g32        ;    decrease stoo count;
3066    j4: jl  w3  g30        ; sense: sense device;
3068        jl.     j2.        ;    goto done;
3070
3070    ~~j5: rl  w3  g32~~        ; disconnect: decrease stop count;
3072        ~~jl  w3  g29~~        ;    disconnected device;
3074        ~~jl.     j2.~~        ;    goto done;
3076
3076  e.     ; end of rc 4320 drum
3076  e.     ; end of backing store area
 076
_076  m.
3076                   monitor text 1 included
3076
3076  m.
3076                   monitor text 2
3076  m.
3076
3076
3076  ; rc 315 typewriter:
3076  ; olivetti terminal:
3076  ;
3076  ; process description format:
3076  ;
3076  ; a10: <kind=if teletype then 36 else if terminal then 46 else 8>
3076  ; a11: <name>
3076  ; a50: <device number*64>
3076  ; a52: <reserved>
3076  ; a53: <users>
 076  ; a54: <next message>
3076  ; a55: <last message>
3076  ; a56: <interrupt address=c33>
3076  ; a70: <state>
3076  ; a71: <operator key> ~~<not used>~~
3076  ; a72: <timer count> <max count>
3076  ; a73: <address>
```

a71+1:
current mode. { 0 normal
8 raw data.

```
3076 ; a74: <last address>
3076 ; a75: <word>
3076 ; a76: <character shift>
3076 ; a77: <link>
3076 ; a78: <user name>
3076 ; a78+8: <repeat>
3076 ; a78+10: <cancel>
3076 ; a78+12: <end medium>
3076 ; a78+14: <substitute>
3076 ;
3076 ;         state: 0  available
3076 ;                1  output of text
3076 ;                2  output from internal process
3076 ;                3  input to internal process
3076 ;                4  input to console buffer
3076 ;                5  input of name
3076 ;                6  tape input to internal process even parity
3076 ;                7  tape input to internal process no parity
3076 b.i59,a0=1<23                ; begin
3076 w.      a0>0+a0>3+a0>5       ;
3078    i0: a0>0+a0>2+a0>4
3080    h7: jl   w3   g14+8       ;    check user only;
3082        dl.  w1   i0.         ;
3084        jl   w3   g16         ;    check operation(0.3.5,0.2.4);
3086        jl   w3   g17         ;    link operation;
3088        rl   w3   x1+a70      ;
3090        se   w3   0           ;    if state(proc)<>0 then
3092        jl        (b20)       ;    goto return;
3094    i1:                       ; start:
3094        sn   w0   0           ;    if operation(buf)=0 then
3096        jl.       i6.         ;    goto sense;
3098 ~~       rl   w3   x1+a10      ;    if kind(proc) <> 8~~
3100 ~~       sn   w3    8          ;    and~~
3102 ~~       jl.       i54.        ;~~
3104 ~~       bz   w3   x2+9~~
3106 ~~       se   w0   5           ;    if operation(buf)=5~~
3108 ~~       sn   w3   0           ;    or mode(buf)=0 then~~
3110 ~~       jl.       4~~
3112 ~~       jl.       i54.        ;    begin~~
3114 ~~       al.  w1   i50.~~
3116 ~~       al   w2   x1+2        ;      typeout(1,first dummy addr,~~
3118 ~~       jl.  w3   i31.        ;                last dummy addr,~~
3120 ~~       [illegible]           ;                disconnect);~~
3122 ~~       rl   w2   x18~~
3124 ~~  i54:                       ;    end;~~
3124        rl   w2   x2+6        ;    internal:=sender(buf);
3126        sh   w2   -1          ;    if internal<0 then
3128        jl.       [illegible] ;    goto ~~disconnect;~~ sense
                      i6.
3130        dl   w0   x2+4        ;
3132        sn   w3 (x1+a78)      ;
3134        se   w0 (x1+a78+2)    ;
3136        jl.       i2.         ;
3138        dl   w0   x2+8        ;
3140        sn   w3 (x1+a78+4)    ;    if user name(proc)<>
3142        se   w0 (x1+a78+6)    ;    name(internal)
3144        jl.       i3.         ;    then
3146        jl.       i4.         ;    begin
3148    i2: ds   w0   x1+a78+2    ;
3150        dl   w0   x2+8        ;
3152    i3: ds   w0   x1+a78+6    ;    user name(proc):=name(internal);
3154
3154        am        (b18)       ;
3156        bz   w0   8           ;
3158        al.  w1   i21.        ;    text:=if operation(buf)=3
3160        se   w0   3           ;    then <:<10>to<32>:>
3162        al.  w1   i22.        ;    else <:<10>from<32>:>;
3164        al   w2   x1+2        ;    typeout(1,text,
3166        jl.  w3   i31.        ;           text+2,
3168        [illegible]           ;           disconnect);
3170        al   w1   x1+a78      ;
3172        al   w2   x1+6        ;    typeout(1,proc+username,
3174        jl.  w3   i31.        ;           proc+username+6,
```

Remove.

```
3176     [redacted]              ;              disconnect);
3178          al. w1   i27.      ;
3180          al. w2   i27.      ;    typeout(1,new line,
3182          jl. w3   i31.      ;              new line,
3184     [redacted]              ;              disconnect);
3186                             ;    end;
3186     i4:  rl  w2   b18       ;    if operation(buf)=5 then
3188          bz  w3   x2+9
3190          bz  w0   x2+8      ;     typeout(2,first addr(buf),
3192          dl  w2   x2+12     ;               last addr(buf),
3194          se  w0   5         ;               disconnect)
3196          jl.      x3+4
3198          am       i36       ;    else begin
3200          am       i41       ;          i:=mode(buf)shift(-1)+1;
3202          am       i40       ;          typein(case i of(3,6,7),first addr(buf
3204          jl. w3   i39.      ;                     last addr(buf),
3206     [redacted]              ;                     disconnect)
3208                             ;         end;
3208          bz  w3   x1+a71    ;    if operator key(proc)=1 then
3210          ls  w3   16        ;    status(7):= 1;
3212          wa  w0   6
3214          jl  w3   g33       ;    prepare answer(status,count,addr);
3216          jl  w3   g18       ;    deliver result(1);
3218     i5:  rl  w1   b19       ; done:
3220          al  w0   0         ;
3222          rs  w0   x1+a70    ;    state(proc):=0;
3224          bz  w3   x1+a71    ;
3226          se  w3   0         ;    if operator key(proc)<>0
3228          jl.      i12.      ;    then goto operator request:
3230          jl  w3   g25       ;    next operation;
3232          jl.      i1.       ;    goto start;
3234
3234     i6:  jl  w3   g30       ; sense: sense device;
3236          jl.      i5.       ;    goto done;
3238
3238     [redacted]              ; disconnect: disconnected device;
3240                             ;    goto done;
3242
3242     c36:rl   w1   x1+a56+2  ; key interrupt:
3244          rs  w1   b19       ;    proc:=word(proc+a56+2);
3246          rl  w0   x1+a70    ;
3248          sn  w0   0         ;    if state(proc)<>0 then
3250          jl.      i12.      ;    begin
3252          al  w0   1         ;    operator key(proc):=1;
3254          hs  w0   x1+a71    ;    goto return;
3256          jl      (b20)      ;    end;
 258
3258     i12:                    ; operator request:
3258          hs  w0   x1+a71    ;    operator key(proc):= 0:
3260          rl  w0   x1+a52    ; test reserver:
3262          sn  w0    0        ;    if reserved(proc) = 0
3264          jl.      i10.      ;    then goto attention;
3266          ns. w0    3        ;
3268          ac  w1    0        ;    sender:= reserver(proc);
3270          wa  w1    2        ;    comment: never proc func;
3272          wa  w1   b6        ;
3274          rl  w0   x1+2      ;    goto found;
3276          jl.      i11.      ;
3278     i10:al.  w1   i28.      ; attention:   text:= <:<10>att<32>:>;
3280          al. w2   i29.      ;    typeout(1,text,
3282          jl. w3   i31.      ;              end text,
3284     [redacted]              ;              done);
3286          al  w1   x1+a78
3288          al  w2   x1+6      ;    typein(5,proc+user name,
3290          jl. w3   i35.      ;             proc+user name+6,
 292     [redacted]              ;             done);
3294          al  w2   x1+a78    ;    search name(proc+user name,
3296          jl  w3   d11       ;                  entry);
3298          sl  w3  (b6)       ;    if entry < first internal in name tb
3300          sn  w3  (b7)       ;    or entry=name table end then
3302          jl.      i13.      ;    goto unknown;
3304          rl  w0   x3+0      ;    sender:= name table(entry):
   i11:jl. w3  i16.        ; found: operator answer(sender,
       jl.     i14.        ;                        no sender):
       jl.     i5.         ;   goto done;
   i13:al. w1  i25.        ; unknown:
       jl.     i15.        ;   text:= <:unknown<10>:>
   i14:al. w1  i23.        ; no sender:
       am      i38         ;      or <:wait<10>:>;
   i15:al. w2  i26.        ;   typeout(1,text,
       jl. w3  i31.        ;           end text,
   [redacted]              ;           done);
       jl.     i5.         ;   goto done;

; procedure operator answer(sender,sorry);
;        call:              exit:
; w0     sender             sender
; w1                        destroyed
; w2                        destroyed
; w3     link               destroyed;
   i16:   rs.w3 i18.        ; begin
          rl w1(b3)         ;   operator:= name table(first entry):
          sz w1 0           ;
   i20:   jl w3 d15         ;
          al w2 x1+a54      ;   buf:= event q(operator);
   i17:   rl w2 x2+0        ; next: buf:= next(buf);
          sn w2 x1+a54      ;   if buf=event q(operator) then
          jl.  (i18.)       ;     goto sorry;
          rl w3 x2+6        ;   if sender(buf) < 0 then
          sh w3 -1          ;   deliver answer (buf);
          jl.   i20.        ;
          se w0(x2+6)       ;   if sender(buf)<>sender then
          jl.   i17.        ;     goto next;
          rl w3 x3+a14      ;   if sender is not user
          la w3 x1+a53      ;   of device
          sn w3  0          ;   then goto sorry;
          jl.  (i18.)       ;
          rs w2 b18
          al w3 0           ;   answer(0):= 0;
          rl w0 b19         ;   answer(2):= proc;
          ds w0 g21
          jl w3 g18         ;   deliver result(1);
          rl.w3 i18.        ; exit:
          jl    x3+2        ; end;
   i18:   0 ; saved return

   i21: <:<10>to<32>:>
   i22: <:<10>from<32>:>
   i23: <:wait<10>:>  w. i24=k-2
   i25: <:unknown<10>:>  w. i26=k-2
   i27: <:<10>:>
   i28: <:<10>att<32>:>  w.  i29=k-2
[redacted]

; procedure typeout(state, first addr, last addr, disconnect)
; comment: outputs the characters from first to last address on a
; typewriter. the output is terminated in the following situations:
;      1.   after an operator key interrupt
;      2.   when the sending process is stopped or removed
;      3.   when the storage area is empty
;      4.   after a timer error
;      5.   when the device is disconnected
; upon return, the address points to the last word from which
; 0, 1, 2, or 3 characters (as defined by count) were output.
;      call:          return:
; w0   state          status and count
; w1   first addr     proc
; w2   last addr      addr
; w3   link           link

b.j24                       ; begin
w.i31:am        -1          ;   state:=1
   i32:al   w0   2          ;     or 2;
       am      (b19)        ;   addr(proc):=first addr;
```

```
         ds   w2   a74        ;   last addr(proc):=last addr;
         rl   w1   b19        ;
         rs   w0   x1+a70     ;   state(proc):=state;
         rs   w3   x1+a77     ;   link(proc):=link;
         rl   w3   x1+a73     ;
;        w0=char or status    w1= proc    w2=shift    w3=addr
    j0:  rl   w3   x3+0       ; next word:
         rs   w3   x1+a75     ;   word(proc):=word(addr(proc));
         al   w2   -16        ;   shift(proc):=-16;
    j1:  rs   w2   x1+a76     ; next char:
         rl   w3   x1+a75     ;
         ls   w3   x2+0       ;   char:=word(proc) shift shift(proc);
OK     ~~la   w3   b54        ;   char:=char(17:23);~~
         sn   w3   0          ;   if char<>0 then
         jl.       j2.        ;   begin
         al   w0   x3
       ~~se   w0   10         ;   if char=10 then~~
       ~~jl.       j6.        ;   begin~~
       ~~al   w3   3          ;   rep:=3;~~
  ~~j7:  al   w3   x3-1       ; repeat:~~
       ~~rs   w3   x1+a78+8   ;   rep:= repeat(proc):=rep-1;~~
       ~~so   w3   2.1        ;   char:=~~
       ~~am        3          ;   if rep(23)=1 then 10~~
       ~~al   w0   10         ;   else 13;~~
         am        (x1+a50)   ;   end;
         io   w0   3          ;   write(device,char);
       ~~sx        2.11~~
       ~~jl        (x1+a72)~~ ;   if ex<>0 then goto disconnect;
         jl   w3   c32        ;   wait interrupt(proc);
ok     ~~io   w0   (x1+a50)   ;   status:=sense(device);~~
       ~~sx        2.11~~
       ~~jl        (x1+a72)~~ ;   if ex<>0 then goto disconnect;
       ~~bz   w0   (a59)~~    ;   if status(2)=1 then goto error;
       ~~jl.       j4.~~
       ~~rl   w3   x1+a78+8   ;   if rep<>0 then~~
       ~~se   w3   0          ;   goto repeat;~~
       ~~jl.       j7.        ;   end;~~
    j2:  rl   w2   x1+a76     ;
         al   w2   x2+8       ;   shift(proc):=shift(proc)+8;
         sh   w2   0          ;   if shift(proc)<=0
         jl.       j1.        ;   then goto next char;

         rl   w3   x1+a70     ; end word:
         se   w3   2          ;
         jl.       j3.        ;   if state(proc)=2 then
         jl   w3   g34        ;   begin
         jl.       j5.        ;   exam sender(done);
         bz   w3   x1+a71     ;   if operator key(proc)=1
         sn   w3   1          ;   then goto done;
         jl.       j5.        ;   end;
    j3:  rl   w3   x1+a73     ;
         sn   w3   (x1+a74)   ;   if addr(proc)=last addr(proc)
         jl.       j5.        ;   then goto done;
         al   w3   x3+2       ;
         rs   w3   x1+a73     ;   addr(proc):=addr(proc)+2;
         jl.       j0.        ;   goto next word;

    j4:  rl   w2   x1+a76     ; error:
    j5:  al   w2   x2+16      ; done:
         ls   w2   -3         ;   count:=
         hl   w0   5          ;     (shift(proc)+16)/8;
         rl   w2   x1+a73     ;   addr:=addr(proc);
       ~~al   w3   x1+a77~~   ;
         jl        ~~x3+2~~   ;
  e.               (x1+a77)   ; end

; procedure typein(state, first addr, last addr, disconnect);
; comment: inputs characters from first to last address from
; a typewriter. the input is terminated in the following
; situations:
;      1.  after an operator key interrupt
;      2.  when the sending process is stopped or removed
```

```
3534  ;      3.  when the storage area is full
3534  ;      4.  after a maximum number of timer errors
3534  ;      5.  when the device is disconnected
3534  ;      6.  after input of a new line character
3534  ; input of a name (state=5) is terminated as follows:
3534  ;      1.  a new line character is not included in the name
3534  ;      2.  if the name is less than four words the remaining
3534  ;          words are filled with null characters.
 534  ;      3.  if the input consists solely of a new line character
 534  ;          the name is unchanged.
3534  ; upon return, the address points to the last word to which
3534  ; 0, 1, 2, or 3 characters (as defined by count) were input.
3534  ;      call:         return:
3534  ; w0   state         status and count
3534  ; w1   first addr    proc
3534  ; w2   last addr     addr
3534  ; w3   link          link
3534
3534  b.j24                        ; begin
3534  w.i33:am       -1            ;   state:=3
3536    i34:am       -1            ;      or 4
3538    i35:am       -1            ;      or 5
3540    i37:am       -1            ;      or 6
3542    i39:al   w0   7            ;      or 7;
3544        am       (b19)         ;   addr(proc):=first addr;
3546        ds   w2   a74          ;   last addr(proc):=last addr;
3548        rl   w1   b19          ;
 550        rs   w0   x1+a70       ;   state(proc):=state;
 552        rs   w3   x1+a77       ;   link(proc):=link;
3554        al   w3   0            ;
3556        hs   w3   x1+a72       ;   timer count(proc):=0;
3558  ;     w0=status or char        w1=proc   w2=shift   w3=addr
3558                               ; next word:
3558    j0: al   w2   16           ;   shift(proc):=16;
3560    j1: rs   w2   x1+a76       ; next char:
3562    j2: am       (x1+a50)      ; repeat:
3564        io        2            ;   read(device);
3566      [redacted]
3568                                   if ex<>0 then goto disconnect;
3570        jl   w3   c32          ;   wait interrupt(proc);
3572        io   w0  (x1+a50)      ;   status:=sense(device);
3574      [redacted]               ;
3576                               ;   if ex<>0 then goto disconnect;
3578        rl   w2   x1+a76       ;
3580      [redacted]               ;   if status(2)=1 then
3582                               ;   status:=status and 1<21;
 584        rs.  w0   j24.         ;   saved status:=status;
 586
3586        rl   w3   x1+a70       ;
3588        so   w3   2.010        ;   if state(proc)=6
3590        jl.       j3.          ;   or state(proc)=7
3592        jl   w3   g34          ;   or state(proc)=3 then
3594        jl.       j6.          ;   exam sender(done);
3596    j3: bz   w3   x1+a71       ;
3598        sn   w3   1            ;   if operator key(proc)=1
3600        jl.       j6.          ;   then goto done;
3602        rl   w3   x1+a70
3604
3606        ~~jl.       j4.          ;   then goto timer;~~
3608
3610        ~~sz   w0  (g58)         ;   or status(1)=1 then~~
3612
3614        la   w0   g54          ;   char:=status(17:23);
3616      [redacted]
3618
 620        sn   w0   10           ;   if char=10
 622        jl.       j5.          ;   then goto end line;
3624      [redacted]
3626
3628      [redacted]
3630                               ;   if char=7 then
3632                               ;   operator key(proc):= 1;
```

} BEL should not be att.

```
3634  ~~      hs   w0   x1+a74      ;   if char=7 then~~
3636  ~~      jl.       j2.         ;~~
3638          sn   w0  (x1+a78+10);   if char=cancel(proc)
3640          al   w0   24         ;   then char:=<can>
3642  ~~      sn   w0  (x1+a78+12);~~  else if char=end medium(proc)
3644  ~~      al   w0   25         ;~~       then char:=<em>
3646  ~~      sn   w0  (x1+a78+14);~~       else if char=substitute(proc)
3648  ~~      al   w0   26         ;~~            then char:=<sub>;
3650  ~~j14:  sn   w0   25~~       ;   if char=<em> then
3652  ~~      jl.       j5.~~      ;   goto end line;
3654    j11:  se   w0   24         ;   if char=<can> then
3656          jl.       ~~j16.~~ 16 ;   goto again;
3658  ~~      se   w0   26         ;~~  if char=<sub> then
3660  ~~      jl.       j15.       ;~~  begin
3662  ~~      al   w2   x2+8       ;~~    shift(proc):=shift(proc)+8;
3664  ~~      sh   w2   16         ;~~    if shift(proc)<=16 then
3666  ~~      jl.       j17.       ;~~    goto zero char;
3668  ~~      al   w0   -2~~
3670  ~~      wa   w0   x1+a73     ;~~    ~~addr(proc):=addr(proc)-2;~~
3672    j16:rs   w0   x1+a73     ; again:
3674          rl   w0   x1+a70
3676          sn   w0   5
3678          jl.       j18.       ;
3680          am        (b18)      ;      if state(proc)=5 then
3682          rl   w0   10         ;      first addr:=user name(proc)
3684          jl.       j19.       ;      else
3686    j18:al   w0   x1+a78     ;      first addr:=first addr(buf);
3688    j19:rl   w3   x1+a73
3690          ws   w3   0          ;      if char=<can>
3692          sh   w3   -1         ;       or addr(proc)<first addr
3694          jl.       j20.       ;      then goto insert first;
3696          al   w2   0          ;      shift(proc):=0;
3698    j17:al   w3   -256       ; zero char:
3700          ls   w3   x2
3702          la   w3   (x1+a73)   ;      word(addr(proc)):=-256 shift shift(proc)
3704          rs   w3   (x1+a73)   ;        and word(addr(proc));
3706          jl.       j1.        ;      goto next char;
3708    j20:rs   w0   x1+a73     ; insert first: addr(proc):=first addr;
3710          jl.       j0.        ;      goto next word;
3712  ~~j16:                       ;    end;~~
3712  ~~      sh   w0   93         ; convert char:    if char<=93~~
3714  ~~      sh   w0   64         ;    and char>64 then~~
3716  ~~      jl.       j10.       ;    begin~~
3718  ~~      rl   w3   x1+a10     ;    if kind=36 then~~
3720  ~~      se   w3   36         ;    char:= char+32~~
3722  ~~      wa   w0   j11.       ;    end;~~
3724    j10:ls   w0   x2+0       ; insert char: char:=char shift shift(proc);
3726          se   w2   16         ;    if shift(proc)<>16 then
3728          lo   w0   (x1+a73)   ;    char:=char or word(addr(proc));
3730          rs   w0   (x1+a73)   ;    word(addr(proc)):=char;
3732          al   w2   x2-8       ;    shift(proc):=shift(proc)-8;
3734          sl   w2   0          ;    if shift(proc)>=0
3736          jl.       j1.        ;    then goto next char;
3738          rl   w3   x1+a73     ; end word:
3740          sl   w3   (x1+a74)   ;    if addr(proc)>=last addr(proc)
3742          jl.       j23.       ;    then goto done1;
3744          al   w3   x3+2       ;
3746          rs   w3   x1+a73     ;    addr(proc):=addr(proc)+2;
3748          jl.       j0.        ;    goto next word;
3750
3750  ~~j4:   so   w3   2.110      ; timer:~~
3752  ~~      jl.       4          ;    if state(proc)=6 or state(proc)=7~~
3754  ~~      jl.       j6.        ;    then goto done;~~
3756  ~~      bz   w3   x1+a72     ;~~
3758  ~~      al   w3   x3+1            timer count(proc):=~~
3760  ~~      hs   w3   x1+a72     ;    timer count(proc)+1;~~
3762  ~~      bs   w3   x1+a72+1   ;    if timer count(proc)<~~
3764  ~~      sh   w3   -1         ;    max count(proc)~~
3766  ~~      jl.       j2.        ;    then goto repeat;~~
3768  ~~      jl.       j6.        ;    goto done;~~
3770
3770  ~~j21:se   w3   7          ; parity:~~
```

ok

ok

Remove

```
~~     al  w0  128+26     ;   if state(proc)<>7 then status:=128+26;~~
~~     al  w3  1~~
~~ j22:sz  w0  x3         ;   for i:=23 step -1 until 17,1 do~~
~~     ba. w0  5          ;   if status(i)=1~~
~~     ls  w3  1          ;   then status:=status+128;~~
~~     sn  w3  128;used~~
~~     ls  w3  15~~
~~     sl  w3  0~~
~~     jl.     j22.~~
~~     la  w0  g53        ;   char:=status(16:23);~~
~~     jl.     j10.       ;   goto insert char;~~

 j5: rl  w3  x1+a70     ; end line:
     sn  w3  5          ;   if state(proc)<>5 then
     jl.     j23.       ;   begin
     ls  w0  x2+0       ;   char:=char shift shift(proc);
     se  w2  16         ;   if shift(proc)<>16 then
     lo  w0  (x1+a73)   ;   char:=char or word(addr(proc)):
     rs  w0  (x1+a73)   ;   word(addr(proc)):=char;
     al  w2  x2-8       ;   shift(proc):=shift(proc)-8:
                        ;   end;
 j23:rl. w0  j24.       ; done1: status:=saved status;
 j6: al  w2  x2-16      ; done:
     as  w2  -3         ;   count:=
     ac  w3  x2+0       ;    -(shift(proc)-16)/8;
     hl  w0  7
     rl  w3  x1+a70
     se  w3  5          ;   if state(proc)<>5
     jl.     j9.        ;   then goto no name;
     sz  w0  (g51)      ;   if status<>0
     jl.     i14.       ;   then goto no receiver;
     rl  w3  x1+a73
     sz  w0  (g52)      ;   if count<>0
     jl.     j8.        ;   then goto nametail;
     sn  w3  x1+a78     ;   if addr(proc)=proc+user name
     jl.     j9.        ;   then goto no name;
 j7: rs  w0  x3+0       ;   word(addr(proc)):= 0;
 j8:                    ; nametail:
     al  w0  0          ;   for addr(proc):= addr(proc)+2
     al  w3  x3+2       ;       while addr(proc)<=
     sh  w3  x1+a78+6   ;       proc+user name+6 do
     jl.     j7.        ;     word(addr(proc)):= 0;
 j9: rl  w2  x1+a73     ; no name:
     ~~rl  w3  x1+a77~~     ;
     jl      ~~x3+2~~       ;
             (x1+a77
 j24: 0
e.                      ; end

 i36=i32-i33
 i41=i33-i37
 i40=i37-i39
 i38=i24-i26

e.     ; end of rc 315 typewriter;
       ; end of teletypewriter;
       ; end of olivetti terminal;

; rc 2000 paper tape reader:
;
; process description format:
;
; a10: <kind=10>
; a11: <name>
; a50: <device number*64>
; a52: <reserved>
; a53: <users>
; a54: <next message>
; a55: <last message>
; a56: <interrupt address=c33>
; a70: <flexowriter case>
; a71: <flexowriter state>
```

```
3860  ; a72: <last address>
3860  ; a73: <read command address>
3860  ; a74: <status and count>
3860  ; a75: <word>
3860  ; a76: <address and convert>
3860  ; a77: <link>
3860  ;
3860  ;      flexowriter case:                 flexowriter state:
"860  ;      0        lower case                0    normal
.860  ;      1<4      upper case             -133 after bar
3860  ;                                      -195 after underline
3860  ;
3860  ;      reserve and initialize process set case, word, and state=0
3860
3860  [redacted]                   ; if include rc 2000 paper tape reader then
3860  b.i24,a0=1<23                ; begin
3860  w.     a0>0+a0>3             ;
3862    i0: a0>0+a0>2+a0>4[redacted]; Remove flexo
3864    h8: jl   w3  g15           ;   check reservation;
3866        dl.  w1  i0.           ;
3868        jl   w3  g16           ;   check operation(0,3, 0.2.4.6);
3870        jl   w3  g17           ;   link operation;
3872    i1: jl   w3  g35           ; start: init buffered;
3874        sn   w0  0             ;   if operation(buf)=0
3876        jl.      i12.          ;   then goto sense;
3878        ld   w0  -65           ;   case:= 0;
3880        ds   w0  g44           ;   state:= 0;
 882        bz   w3  x2+9          ;
3884        ~~se   w3  6.~~            ;   ~~if mode(buf)=6 then~~
3886        ~~jl.      i13~~           ;   ~~begin~~
3888        ~~dl   w0  x1+a71~~        ;   ~~case:= flexowriter case(proc);~~
3890        ~~ds   w0  g44~~           ;   ~~state:= flexowriter state(proc);~~
3892        ~~bz   w3  x2+9~~          ;
3894   i13: dl   w2  x2+12         ;   last addr:=last addr(buf);
3896        [redacted]             ;   if mode(buf)=6
3898                               ;   then first addr(23):=1;
3900        jl.      x3+2          ;   read addr:=sense addr+
3902     0  am       -4    0       ;   case mode(buf) of
3904     2  am       -4    2       ;   (0: read odd,
3906        ~~am       0~~     4       ;    2: read even,
3908     4  al   w3  10    6       ;    4: read general,
3910        wa   w3  g42           ;    6: read general);
3912        ds   w3  g46           ;
3914        al   w2  x1+0          ;   addr:=first addr(buf);
3916        io       (g46)         ; read first:
3918        [redacted]             ;   read(device);
 920                               ;   if ex<>0 then exception(read first);
3922        am       (b19)         ;
3924        rl   w1  a75           ;   word:=word(proc);
3926        sn   w1  0             ;   if word=0 then
3928        jl.      i4.           ;   goto sense 1;
3930        sz.  w1  (i21.)        ;   if word(8:23)=0 then
3932        jl.      4             ;
3934        jl.      i6.           ;   goto sense 2;
3936        sl   w2  (g45)         ;   if addr>=last addr
3938        jl.      i9.           ;   then goto sense last;
3940
3940  ;       w0=char   w1=word   w2=addr and convert   w3=link
3940  ;       in mode 6 (flexowriter conversion)  w2(23)=1
3940
3940    i2: io   w0  (g42) -10, -4 sense 3:
3942        [redacted]  -8'        ;   status:=sense(device);
3944                    -6         ;   if ex<>0 then exception(sense 3);
3946        sz   w0  -256 -4       ;   if status(0:15)<>0
3948   v    jl.  w3  i16.          ;   then status 2(status,sense 3);
 950    i3: io       (g46)         ; read 1:
3952        [redacted]             ;   read(device);
3954                               ;   if ex<>0 then exception(read 1);
3956                               ;   if addr(23)=1
3958                               ;   then convert(status,sense 3);
3960        wa   w1  0             ;   word:=word+char;
3962        rs   w1  x2+0          ;   word(addr):=word;
```

```
3964        al   w2   x2+2        ;   addr:=addr+2;
3966   i4:  al   w1   0           ; sense 1:
3968        io   w0  (g42) -2 ✓   ;   word:=0;
3970        [redacted]      -4 -6 ;   status:=sense(device);
3972        [redacted]       -4   ;   if ex<>0 then exception(sense 1);
3974        sz   w0   -256 -4     ;   if status(0:15)<>0
3976        jl.  w3   i18.        ;   then status 0(status, sense 1);
3978   i5:  io       (g46)        ; read 2:
3980        [redacted]            ;   read(device);
3982        [redacted]            ;   if ex<>0 then exception(read 2);
3984        [redacted]            ;   if addr(23)=1
3986        [redacted]            ;   then convert(status,sense1);
3988        ld   w1   -8          ;   word:=status shift 16;
3990   i6:  io   w0  (g42) -10    ; sense 2:
3992        [redacted]       -8   ;   status:=sense(device);
3994        [redacted]       -6   ;   if ex<>0 then exception(sense 2);
3996        sz   w0   -256 -4     ;   if status(0:15)<>0
3998        jl.  w3   i17. -2     ;   then status 1(status,sense2);
4000   i7:  io       (g46)        ; read 3:
4002        [redacted]            ;   read (device);
4004        [redacted]            ;   if ex<>0 then exception(read3);
4006        [redacted]            ;   if addr(23)=1
4008        [redacted]            ;   then convert(status,sense2);
4010        ls   w0   8           ;
4012        wa   w1   0           ;   word:=word+status shift 8;
4014        sl   w2  (g45)        ;   if addr>=last addr
4016        jl.       i9.         ;   then goto sense last;
4018        jl.       i2.         ;   goto sense 3;
4020
4020   i8:  io       (g46)        ; read last:
4022        [redacted]            ;   read(device);
4024        [redacted]            ;   if ex<>0 then exception(read last);
4026   i9:  io   w0  (g42)        ; sense last:
4028        [redacted]            ;   status:=sense(device);
4030        [redacted]            ;   if ex<>0 then exception(sense last);
4032        sz   w0   -256        ;   if status(0:15)<>0
4034  ->    jl.  w3   i16.        ;   then status 2(status,sense last);
4036        [redacted]            ;   if addr(23)=1
4038        [redacted]            ;   then convert(status,read last):
4040        wa   w1   0           ;   word:=word+status;
4042        al   w0   3           ;   count:=3;
4044   i10:rs    w1   x2+0        ; done 0:word(addr):=word;
4046        al   w1   0           ;   word:=0;
4048   i22:am        (b19)        ; done 2:
4050        rs   w1   a75         ;   word(proc):=word;
4052        la   w2   g50         ;    addr(23):=0;
4054        jl   w3   g33         ;   prepare answer(status,count,addr);
4056        jl   w3   g18         ;   deliver result(1);
4058        dl   w0   g44         ;   flexowriter case(proc):= case;
4060        am       (b19)
4062        ds   w0   a71         ;   flexowriter state(proc):= state;
4064   i11:jl    w3   g25         ; done 1: next operation;
4066        jl.       i1.         ;   goto start;
4068
4068   i12:jl    w3   g30         ; sense: sense device;
4070        jl.       i11.        ;   goto done 1;
4072
4072   i21:8.0017 7777
4074
4074  ; procedure exception(repeat);
4074  ; comment: examines the exception register and returns
4074  ; to the address repeat=link-6 if the devide is busy.
4074  ;     call:    return:
4074  ; w0           unchanged
4074  ; w1           unchanged
4074  ; w2           unchanged
4074  ; w3  link     link
4074
4074  ~~b.i24~~                  ; ~~begin~~
4074  ~~w.i15:sx       2.01~~      ;   ~~if ex(23)=1~~
4076        ~~jl       x3-6~~      ;   ~~then goto repeat;~~
4078        ~~jl  w3   g29~~       ;   ~~disconnected device;~~
```

c

Leif typed to here.

```
4080  ~~jl.      i11.          ;   goto done 1~~
4082  ~~e.                      ; end~~
4082
4082  ; procedure status(status,repeat)
4082  ; comment: called in the following situations:
4082  ;   end buffer: saves working registers and device parameters
4082  ; and waits for an interrupt. if the sender is stopped or removed
4082  ; after the interrupt the action for end tape is performed,
 082  ; otherwise a read operation is initiated and followed by a
 082  ; return to the address repeat=link-10.
4082  ;   parity error: in mode 4 (no parity) the parity bit is
4082  ; removed and the operation continued, but in modes 0, 2,
4082  ; and 6 the action for end tape is performed.
4082  ;   end tape: the last characters input are stored and
4082  ; the operation is terminated.
4082  ;       call:      return:
4082  ; w0  status     status and count
4082  ; w1  word       word
4082  ; w2  addr       addr
4082  ; w3  link       link
4082
4082  b.j24                        ; begin
4082  w.i16:am      1              ; status 2: count:=2
4084    i17:am      1              ; status 1:      or 1
4086    i18:xl.     j3.            ; status 0:      or 0;
4088        so  w0 (g58)           ;
4090        jl.     j0.            ;   if status(1)=1 then
 092        rs. w3  j2.            ; parity:
4094        am     (b18)           ;   begin
4096        bz  w3  9              ;
4098 se w34 ~~so  w3  6~~              ;   ~~if mode(buf)=6~~
4100        ~~so  w0  255~~            ;   ~~and char<>all holes~~
4102        ~~sn  w3  4~~              ;   ~~or mode(buf)<4~~
4104        ~~jl.     4~~
4106        jl.     j5.            ;   then goto insert sub;
4108        la  w0  g53            ;   status(0:15):=0;
4110        jl.    (j2.)           ;   end else
4112    j0: xs      1              ;   if status(0)<>1 then
4114        sl  w0  0              ; end tape: goto done 0;
4116        jl.     i10.           ; end buffer:
4118        al  w0  0              ;   count:=0;
4120        am     (b19)           ;   param6(proc):=addr;
4122        ds  w3  a77            ;   param7(proc):=link;
4124        jl  w3  g34            ;   exam sender(done 2);
4126        jl.     i22.           ;
4128        jl  w3  g36            ;   wait buffered;
 130        jl  w3  g37            ;   continue buffered;
4132        ~~rs. w3  j2.~~            ;
4134 io (g46) ~~[illegible]  (g46)~~  ok  ; more:
4136        ~~[illegible]~~            ;   read(device);
4138        ~~jl. w3  [illegible]~~    ;   if ex<>0 then exception(more);
4140        ~~jl. w3  j2.~~            ;
4142        jl      x3-~~10~~ 6        ;   goto repeat;
4144    j5: xs      1              ; insert sub:
4146        rs  w1  x2+0           ;   word(addr):=word;
4148        ba. w0  1              ;   count:=count+1;
4150        al  w3  3
4152        bs  w3  1              ;
4154        al  w1  26
4156        ls  w3  3
4158        ls  w1  x3
4160        wa  w1  x2+0           ;   word:=word(addr)+sub shift 8*(3-count);
4162        jl.     i10.           ;   goto done 0;
4164    j2: 0                      ;
4166  h.j3: 0, 1, 2, 0             ;
 170  w.
4170  e.                           ; end
4170
4170  ; procedure convert(char,skip);
4170  ; comment: converts a flexowriter character to the iso 7 bit
4170  ; code. in upper case, the flexowriter parity bit is inverted before
4170  ; the conversion. in lower case the flexowriter character is
```

if mode ≠ 4
then goto insert sub

0
2
4

```
; used directly as index in a byte table of iso characters.
; flexowriter character greater than 128 (=carret) cause
; a return to the skip address=link-20.
;   special characters are handled in accordance with
; the following state table after the conversion:
;                         0       -133          -195
;                      normal     after bar    after underline
; character value
; 127 delete      : 0 skip   -133 skip     -195 skip
; 128 caseshift   : 0 shift  -133 shift    -195 shift
; 133 bar         :-133 skip -133 skip       0 substitute
; 195 underline   :-195 skip  0 substitute -195 skip
;  32 space       : 0 return  0 exclamation 0 underline
;<127 other       : 0 return  0 substitute  0 substitute
;
;     call:                 return:
; w0  flexowriter char iso char
; w1                    unchanged
; w2                    unchanged
; w3  link              link

b.j24                          ; begin
w.j19:sl   w0   129            ;   if char>128
      jl.       j2.            ;   then goto delete;
      se   w0   128            ;   if char <> 128 then
      lx   w0   g43            ;   char(19):=char(19) exor case;
      am       (0)             ;
      bz.  w0   j20.           ;   char:=conversion table(char);
      am       (g44)           ;
      sh   w0   126            ;   if char<127+state
      jl        x3+0           ;   then goto exit;
      sn   w0   127            ;   if char=127
      jl.       j2.            ;   then goto delete;
      sn   w0   128            ;   if char=128
      jl.       j1.            ;   then goto caseshift;
      sh   w0   132            ;   if char<133
      jl.       j3.            ;   then goto not special;
      ac   w0  (0)             ; bar or underline:
      rx   w0   g44            ;   old state:= state; state:= -char;
      se   w0   0              ;   if old state=0
      sn   w0  (g44)           ;   or old state=state
      jl.       j2.            ;   then goto delete;
   j0: al  w0   0              ; substitute:
      rs   w0   g44            ;   state:= 0;
      al   w0   26             ;   char:= 26;
      jl        x3+0           ;   goto exit;

   j1: al  w0   1<4            ; caseshift:
      lx   w0   g43            ;
      rs   w0   g43            ;   case:=case exor bit 19;
   j2:                         ; delete:
      jl        x3-20          ;   goto skip;

   j3: se  w0   32             ; not special: if char<>32
      jl.       j0.            ;   then goto substitute;
      al   w0   0              ;
      rx   w0   g44            ;   char:=-state-100;
      ac   w0  (0)             ;
      ws.  w0   j5.            ;   state:=0;
      jl        x3+0           ; exit:
   j5: 100                     ;
e.                             ; end

; flexowriter conversion table
; iso code:                      flexowriter characters:
h.j20:
    32,  49,  50,  47,  52   ; space  1      2      /      4
    59,  91,  55,  56,  41   ; ;      uc 6   7      8      )
   127,  12,  25, 125, 195   ; uc 10  stop   end    lc 13  -
   127,  32,  33,  42,  51   ; uc 15  space  uc 1   *      3
    61,  53,  54,  93,  40   ; =      5      6      uc 7   (
    57, 127,  12,  25,  93   ; 9      lc 10  stop   end    uc 13
```

```
4272    133, 127,  48,  62,  83  ; !      lc 15  0      >      uc s
4277    116,  85, 118, 119,  88  ; t      uc u   v      w      uc x
4282     89, 122, 127,  39, 127  ; uc y   z      lc 26  ten    clear
4287    127,   9, 127,  38,  60  ; red    tab    p off  uc 16  <
4292    115,  84, 117,  86,  87  ; s      uc t   u      uc v   uc w
4297    120, 121,  90, 127,  44  ; x      y      uc z   uc 26  ,
4302    127, 127,   9, 127,  45  ; clear  red    tab    p off  -
4307     74,  75, 108,  77, 110  ; uc j   uc k   l      uc m   n
 312    111,  80,  81, 114, 127  ; o      uc p   uc q   r      lc 42
 317     92, 127, 127, 127, 127  ; uc ø   p on   uc 45  uc 46  lc 47
4322     43, 106, 107,  76, 109  ; +      j      k      uc l   m
4327     78,  79, 112, 113,  82  ; ue n   uc o   p      q      uc r
4332    127, 124, 127, 127, 127  ; uc 42  ø      p on   lc 45  lc 46
4337    127,  91,  97,  98,  67  ; lc 47  uc æ   a      b      uc c
4342    100,  69,  70, 103, 104  ; d      uc e   uc f   g      h
4347     73, 128,  46, 127, 127  ; uc i   lower  .      upper  sum
4352    127, 127, 123,  65,  66  ; black  feed   æ      uc a   uc b
4357     99,  68, 101, 102,  71  ; c      uc d   e      f      uc g
4362     72, 105, 127,  58, 128  ; uc h   i      lower  :      upper
4367    127, 127, 127,  10, 127  ; sum    black  feed   carret fill
4372  w.
4372
4372  e.    ; end of rc 2000 paper tape reader;
4372  ~~;       h8     g3~~            ;   goto result 5;
4372
4372
4372  ; rc 150 paper tape punch:
 372  ;
4372  ; process description format:
4372  ;
4372  ; a10: <kind=12>
4372  ; a11: <name>
4372  ; a50: <device number*64>
4372  ; a52: <reserved>
4372  ; a53: <users>
4372  ; a54: <next message>
4372  ; a55: <last message>
4372  ; a56: <interrupt address=c33>
4372  ; a70: <state>
4372  ; a71: <mode>
4372  ; a72: <write command address>
4372  ; a73: <address>
4372  ; a74: <last address>
4372  ; a75: <word>
4372  ; a76: <character shift>
4372  ; a77: <repeat>
 372  ;
4372  ;        state:  0  lower case
4372  ;                2  upper case
4372  ;                4  after underline
4372  ;                6  after bar
4372  ;
4372  ;        reserve and initialize set state, word  and mode = 0
4372
4372  ~~e.(+a91>21a.1;) 1              ; if~~ include rc 150 paper tape punch then
4372  b.i24, a0=1<23              ; begin
4372  w.      a0>0+a0>5           ;        but include this!
4374    i0: a0>0+a0>2+a0>4[redacted]+a0>8
4376    h9: jl   w3   g15         ;    check reservation;
4378        dl.  w1   i0.         ;
4380        jl   w3   g16         ;    check operation(0.5, 0.2.4.6.8);
4382        jl   w3   g17         ;    link operation;
4384    i1: sn   w0   0           ; start: if operation(buf)=0
4386        jl.       i8.         ;    then goto sense;
4388        bz   w3   x2+9        ;
 390        rs   w3   x1+a71      ;    mode(proc):=mode(buf);
4392        jl.       x3+2        ;
4394        am        -4          ;    write addr(proc):=sense addr(proc)+
4396        am        -4          ;    case mode(buf) of
4398        am        0           ;         (0: write odd,
4400        am        4           ;          2: write even,
4402        al   w3   7           ;          4: write general,
4404        wa  w3  x1+a50     ;          6: write general,
4406        rs  w3  x1+a72     ;          8: write even);
4408        dl  w0  x2+12      ;   addr(proc):=first addr(buf);
4410        ds  w0  x1+a74     ;   last addr(proc):=last addr(buf);
4412  ;     w0=char or status   w1=proc   w2=shift   w3=addr
4412   i2:  rl  w0  x3+0       ; next word:
4414        rs  w0  x1+a75     ;   word(proc):=word(addr(proc));
4416        al  w2  -16        ;   shift(proc):=-16;
 418   i3:  rs  w2  x1+a76     ; next char:
+420        rl  w0  x1+a75     ;
4422        ls  w0  x2+0       ;   char:=word(proc) shift shift(proc);
4424        la  w0  g53        ;   char:=char(16:23);
4426        rl  w3  x1+a71     ;
4428        [redacted]de(proc)=6
4430                           ;   then goto convert;
4432        sn  w3  8          ;   if mode(proc)<>8
4434        se  w0  10         ;   or char<>10
4436        jl.     i4.        ;   then goto writechar;
4438        al  w3  3          ;   rep:= 3;
4440   i10:al  w3  x3-1        ; repeat:
4442        rs  w3  x1+a77     ;   rep:=repeat(proc):=rep-1;
4444        so  w3  2.1        ;   char:=
4446        am      3          ;   if rep(23)=1 then 10
4448        al  w0  10         ;   else 13;
4450   i4:  io  w0  (x1+a72)   ; writechar:
4452        [redacted]         ;   write(device,char);
4454                           ;   if ex<>0 then goto disconnect;
 456        jl  w3  c32        ;   wait interrupt(proc);
4458        io  w0  (x1+a50)   ;   status:=sense(device);
4460        [redacted]         ;
4462                           ;   if ex<>0 then goto disconnect;
4464        dl  w3  x1+a77     ;   rep:=repeat(proc);
4466        se  w3  0          ;   if rep<>0 then
4468        jl.     i10.       ;   goto repeat;
4470        sz  w0  (g59)      ;   if status(2)=1
4472        jl.     i6.        ;   then goto done 0;
4474                           ; skipchar:
4474   i5:  al  w2  x2+8       ;   shift(proc):=shift(proc)+8;
4476        sh  w2  0          ;   if shift(proc)<=0
4478        jl.     i3.        ;   then goto next char;
4480
4480        jl  w3  g34        ; end word:
4482        jl.     i6.        ;   exam sender(done 0);
4484        rl  w3  x1+a73     ;   if addr(proc)<last addr(proc)
4486        sl  w3  (x1+a74)   ;   then
4488        jl.     i6.        ;   begin
 490        al  w3  x3+2       ;   addr(proc):=addr(proc)+2;
4492        rs  w3  x1+a73     ;   goto next word;
4494        jl.     i2.        ;   end;
4496   i6:  al  w2  x2+16      ; done 0:
4498        ls  w2  -3         ;   count:=
4500        hl  w0  5          ;   (shift(proc)+16)/8;
4502        rl  w2  x1+a73     ;
4504        jl  w3  g33        ;   prepare answer(status,count,addr);
4506        jl  w3  g18        ;   deliver result(1);
4508   i7:  jl  w3  g25        ; done 1: next operation;
4510        jl.     i1.        ;   goto start;
4512   i8:  jl  w3  g30        ; sense: sense device;
4514        jl.     i7.        ;   goto done 1;
4516
4516   [redacted]              ; disconnect: disconnected device;
4518                           ;   goto done 1;
4520
4520  ; convert:
4520  ; comment: converts an iso 7 bit character to the flexowriter
 520  ; code. the iso character is used directly as index in a byte table
4520  ; of flexowriter characters with the following format:
4520  ;     lower case:   flexowriter code<2+2.01
4520  ;     upper case:   flexowriter code<2+2.10
4520  ;     case free:    flexowriter code<2+2.11
4520  ; characters>=127 are skipped.
4520  ; a space is output after the characters bar and underline.
```

```
4520  ;    entry:      exit:
4520  ; w0  iso char    flexowriter char
4520  ; w1  proc        proc
4520  ; w2  shift       shift
4520  ; w3              destroyed
4520
4520    i15:sl   w0   128            ;    if char>127
4522        jl.       i5.            ;    then goto skipchar;
 524        am        (0)            ;
 526        bz.  w0   i23.           ;    char:=conversion table(char);
4528        rl   w3   x1+a70         ;
4530        se   w3   0              ;    if state(proc)<>0
4532        jl.       i17.           ;    then goto not lower case;
4534        so   w0   1              ;    if char(23)=0
4536        jl.       i18.           ;    then goto caseshift;
4538                                 ; unpack char:
4538    i16:ls   w0   -2             ;    char:=char shift -2;
4540        sn   w0   127            ;    if char=127
4542        jl.       i5.            ;    then goto skipchar;
4544        se   w0   14             ;
4546        jl.       i4.            ;
4548        al   w3   x3+4           ;    if char=14 then
4550        rs   w3   x1+a70         ;    state(proc):=state(proc)+4:
4552        jl.       i4.            ;    goto writechar;
4554                                 ; not lower case:
4554    i17:se   w3   2              ;    if state(proc)<>2
4556        jl.       i20.           ;    then goto after char 14;
 558        sz   w0   2.10           ;    if char(22)=1
 560        jl.       i16.           ;    then goto unpack char;
4562    i18:lx.  w3   i21.           ; caseshift:
4564        rs   w3   x1+a70         ;    state(proc):=state(proc) exor bit 22:
4566        al   w0   x3+122         ;    char:=122+state(proc);
4568    i19:al   w2   x2-8           ; decrease shift:
4570        rs   w2   x1+a76         ;    shift(proc):=shift(proc)-8:
4572        jl.       i4.            ;    goto writechar;
4574    i20:al   w3   x3-4           ; after char 14:
4576        rs   w3   x1+a70         ;    state(proc):=state(proc)-4:
4578        al   w0   16             ;    char:=space:
4580        jl.       i19.           ;    goto decrease shift;
4582    i21:2.10                     ;
4584  ; flexowriter conversion table:
4584  ; flexowriter code<2+case         iso character
4584  h.i23:
4584    511, 511, 511, 511, 511   ;   nul     soh     stx     etx     eot
4589    511, 511, 511, 511, 251   ;   enq     ack     bel     bs      ht
4594    515, 511,  47, 511, 511   ;   nl      vt      ff      cr      so
 599    511, 511, 511, 511, 511   ;   si      dle     dc1     dc2     dc3
4604    511, 511, 511, 511, 511   ;   dc4     nak     syn     etb     can
4609    115, 511, 511, 511, 511   ;   em      sub     esc     fs      gs
4614    511, 511,  67,  58, 511   ;   rs      us      sp      !       "
4619    511, 511, 511, 130, 238   ;   35      36      %       &       '
4624     34, 102,  10, 258, 237   ;   (       )       *       +       ,
4629    257, 429,  78, 129,   5   ;   -       .       /       0       1
4634      9,  77,  17,  85,  89   ;   2       3       4       5       6
4639     29,  33, 101, 430,  86   ;   7       8       9       :       ;
4644    197,  18, 198, 511, 511   ;   <       =       >       ?       64
4649    390, 394, 462, 402, 470   ;   uc a    uc b    uc c    uc d    uc e
4654    474, 414, 418, 486, 326   ;   uc f    uc g    uc h    uc i    uc j
4659    330, 270, 338, 278, 282   ;   uc k    uc l    uc m    uc n    uc o
4664    350, 354, 294, 202, 142   ;   uc p    uc q    uc r    uc s    uc t
4669    210, 150, 154, 222, 226   ;   uc u    uc v    uc w    uc x    uc y
4674    166, 450, 366,  54, 511   ;   uc z    uc æ    uc ø    uc å    94
4679     57, 511, 389, 393, 461   ;   _       96      a       b       c
4684    401, 469, 473, 413, 417   ;   d       e       f       g       h
4689    485, 325, 329, 269, 337   ;   i       j       k       l       m
 694    277, 281, 349, 353, 293   ;   n       o       p       q       r
4699    201, 141, 209, 149, 153   ;   s       t       u       v       w
4704    221, 225, 165, 449, 365   ;   x       y       z       æ       ø
4709     53, 511, 511             ;   å       126     del
4712
4712
4712  e.     ; end of rc 150 paper tape punch;
```

```
4712  [redacted]
4712
4712  ; rc 610 line printer:
4712  ;
4712  ; process description format:
4712  ;
4712  ; a10: <kind=14>
4712  ; a11: <name>
4712  ; a50: <device number*64>
4712  ; a52: <reserved>
4712  ; a53: <users>
4712  ; a54: <next message>
4712  ; a55: <last message>
4712  ; a56: <interrupt address=c33>
4712  ; a70: <not used>
4712  ; a71: <not used>
4712  ; a72: <last address>
4712  ; a73: <write command address>
4712  ; a74: <count>
4712  ; a75: <word>
4712  ; a76: <address>
4712  ; a77: <link>
4712
4712  c.(:a91>20a.1:)-1        ; if include rc 610 line printer then
4712  b.i24,a0=1<23            ; begin
4712  w.     a0>0+a0>5         ;
4714    i0: a0>0               ;
4716   h10:jl   w3   g15       ;   check reservation;
4718       dl.  w1   i0.       ;
4720       jl   w3   g16       ;   check operation(0.5, 0);
4722       jl   w3   g17       ;   link operation;
4724    i1: jl   w3   g35      ; start: init buffered;
4726       sn   w0   0         ;   if operation(buf)=0
4728       jl.       i10.      ;   then goto sense;
4730       dl   w3   x2+12     ;   addr:=first addr(buf);
4732       rs   w3   g45       ;   last addr:=last addr(buf);
4734       rl   w3   g42       ;
4736       al   w3   x3+3      ;
4738       rs   w3   g46       ;   write addr:=sense addr+3;
4740
4740  ;     w0=char or status   w1=word   w2=addr   w3=link
4740                           ; next word:
4740    i3: rl   w1   x2+0     ;   word:=word(addr);
4742       ld   w1   8         ;   char:=word(1:7);
4744       la   w0   g54       ;   word:=word shift 8;
4746
4746    i4: io   w0  (g46)     ; write 1:
4748       sx        2.11      ;   write(device,char);
4750       jl.  w3   i15.      ;   if ex<>0 then exception(write 1);
4752       sh   w0   13        ;   if char<=13
4754       jl.  w3   i18.      ;   then end line 0;
4756       ld   w1   8         ;   char:=word(1:7);
4758       la   w0   g54       ;   word:=word shift 8;
4760    i5: io   w0  (g46)     ; write 2:
4762       sx        2.11      ;   write(device,char);
4764       jl.  w3   i15.      ;   if ex<>0 then exception(write2);
4766       sh   w0   13        ;   if char<=13
4768       jl.  w3   i17.      ;   then end line 1;
4770       ld   w1   8         ;   char:=word(1:7);
4772       la   w0   g54       ;   word:=word shift 8;
4774    i6: io   w0  (g46)     ; write 3:
4776       sx        2.11      ;   write(device,char);
4778       jl.  w3   i15.      ;   if ex<>0 then exception(write 3);
4780       sh   w0   13        ;   if char<=13
4782       jl.  w3   i16.      ;   then end line 2;
4784       al   w2   x2+2      ;   addr:=addr+2;
4786       jl   w3   g34       ;
4788       jl.       i7.       ;   exam sender(last word);
4790       sh   w2  (g45)      ;   if addr<=last addr
4792       jl.       i3.       ;   then goto next word;
4794    i7: io   w0  (g42)     ; last word:
4796       sx        2.11      ;   status:=sense(device);
```

```
4798        jl. w3  i15.        ;   if ex<>0 then exception(last word);
4800        la  w0  g51         ;   count:=0;
4802    i8: jl  w3  g33         ; done 0: prepare answer(status,count,addr);
4804        jl  w3  g18         ;   deliver result(1);
4806    i9: jl  w3  g25         ; done 1: next operation:
4808        jl.     i1.         ;   goto start;
4810
4810   i10: jl  w3  g30         ; sense: sense device;
 812        jl.     i9.         ;   goto done 1;
4814
4814  ; procedure exception(repeat);
4814  ; comment: examines the exception register and returns
4814  ; to the address repeat=link-6 if the device is busy
4814  ;       call:    return:
4814  ; w0            unchanged
4814  ; w1            unchanged
4814  ; w2            unchanged
4814  ; w3  link      link
4814
4814  b.j24                     ; begin
4814  w.i15:sx      2.01        ;   if ex(23)=1
4816        jl      x3-6        ;   then goto repeat;
4818        jl  w3  g29         ;   disconnected device;
4820        jl.     i9.         ;   goto done 1;
4822  e.                        ; end
4822
4822
 822  ; procedure end line
4822  ; comment: called after output of the following characters:
4822  ;       10 new line
4822  ;       11 vertical tabulation
4822  ;       12 form feed
4822  ;       13 carriage return
4822  ; working registers and device parameters are saved. after
4822  ; the interrupt the output is continued unless the status
4822  ; indicates parity or timer errors.
4822  ;       call:       return:
4822  ; w0    status      status and count
4822  ; w1    word        word
4822  ; w2    addr        addr
4822  ; w3    link        destroyed
4822
4822  b.j24                     ; begin
4822  w.i16:am      1           ; end line 2: ccunt:=2
4824    i17:am      1           ; end line 1:     or 1
4826    i18:xl.     j1.         ; end line 0:     or 0;
 828        sh  w0  9           ;   if char>9 then
4830        jl      x3+0        ;   begin
4832        xs      1           ;
4834        am      (b19)       ;   param 6(proc):=addr;
4836        ds  w3  a77         ;   param 7(proc):=link;
4838        jl  w3  g36         ;   wait buffered;
4840        jl  w3  g37         ;   continue buffered;
4842        hs. w0  j2.         ;
4844        rs. w3  j0.         ;
4846        io  w0  (g42)       ; more:
4848        sx      2.11        ;   status:=sense(device);          discon
4850  →     jl. w3  i15.+4      ;   if ex<>0 then exception(~~more~~);
4852        hl. w0  j2.
4854        sz  w0  (g62)       ;   if status(5)=1 then
4856        al  w2  x2-2        ;   addr:= addr-2;
4858                            ;   if status(1)=1
4858        sz. w0  (j3.)       ;   or status(2)=1
4860                            ;   or status(5)=1
4860        jl.     i8.         ;   then goto done 0;
 862        jl.     (j0.)       ;   end;
4864    j0: 0                   ;
4866  h.j1: 0, 1, 2             ;
4869    j2: 0                   ;
4870  w.j3: 2.011001<18         ;
4872  e.                        ; end
4872
```

```
~~4872  e.      ; end of rc 610 line printer;~~
4872  z.      h10 = g3              ;   goto result 5;
4872          h11 = g3              ;
4872
4872  ; rc 4124 mstc (simple)
4872  ;
4872  ; process description format:
4872  ;
 872  ; a10:   <kind=52>
-872  ; a11:   <name>
4872  ; a50:   <device number*64>
4872  ; a52:   <reserved>
4872  ; a53:   <users>
4872  ; a54:   <next message>
4872  ; a55:   <last message>
4872  ; a56:   <interrupt address=c33>
4872  ; a70:   <mode>
4872  ; a71:   <not used>
4872  ; a72:   <last address>
4872  ; a73:   <data command>
4872  ; a74:   <status+count>
4872  ; a75:   <word>
4872  ; a76:   <address>
4872  ; a77:   <link>
4872
4872  c.(:a91>8a.1:) -1        ; if include rc 4124 then
4872  b.i24,a0=1<23            ; begin
 872  w.     a0>0+a0>3+a0>5+a0>6
-4874   i0: a0>0               ;
4876
4876    h23:jl   w3  g15       ;    check reservation;
4878        dl.  w1  i0.       ;    check operation (0.3.5.6, 0);
4880        jl   w3  g16       ;    link operation;
4882        jl   w3  g17       ;
4884
4884    i1: jl   w3  g35       ; start: init buffered;
4886        so   w0   1        ;    if operation(buf) is even
4888        jl.       i14.     ;    then goto sense it;
4890        dl   w3  x2+12     ; set addr:
4892        rs   w3  g45       ;    addr:= first addr(buf);
4894        rl   w3  g42       ;    last addr:= last addr(buf);
4896        se   w0   3        ;
4898        am        1        ;    data command:= sense command
4900        al   w3  x3+2      ;        + if read then 2 else 3:
4902        rs   w3  g46       ;
4904        rs   w0  g43       ;    direct mode:= operation;
 906        se   w0   3        ;    if operation <> 3
4908        jl.       i15.     ;    then goto start write;
4910
4910    i2: sn   w0 (x1+a70)   ; start read:
4912        jl.       i3.      ;    if mode <> 3 then
4914        io   w3 (g46)      ;    begin read (device,irr);
4916        sx        2.11     ;            if ex<>0 then exception;
4918        jl.  w3  i8.       ;            goto first status;
4920        jl.       i4.      ;    end  else  goto read1;
4922
4922    i8: sx        2.01     ; exception:
4924        jl        x3-6     ;    if busy then goto repeat it;
4926        jl   w3  g29       ;    disconnected device;
4928        jl.       i7.      ;    goto finis;
4930
4930    i3: al   w1   0        ; read1:   word:= 0;
4932        io   w0  (g42)     ;    char:= sense0(device);
4934        sx        2.11     ;    if ex<>0 then exception;
4936        jl.  w3  i8.       ;    if char (0:15) <> 0
 938        sz   w0  -256      ;    then
4940    i4: jl.  w3  i9.       ; first status:  goto status0;
4942        ld   w1  -8        ;    word:= char shift 16;
4944        io   w0  (g42)     ;    char:= sense0 (device);
4946        sx        2.11     ;    if ex<>0 then exception;
4948        jl.  w3  i8.       ;    if char (0:15) <> 0
4950        sz   w0  -256      ;    then goto status1:
```

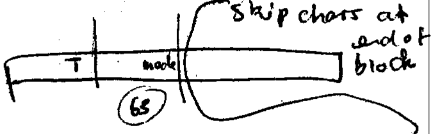
Do not forget!!

```
4952         jl. w3  i10.      ;
4954         ls  w0   8        ;    word:= word + char shift 8;
4956         wa  w1   0        ;
4958         io  w0 (g42)      ;    char:= sense0 (device);
4960         sx       2.11     ;    if ex<>0 then exception;
4962         jl. w3  i8.       ;    if char(0:15) <> 0
4964         sz  w0  -256      ;    then goto status2;
4966         jl  w3  i11.      ;
4968         wa  w1   0        ;    word:= word + char;
4970         rs  w1  x2        ;    word (addr):= word;
4972         al  w2  x2+2      ;    if addr <= last addr
4974         sh  w2 (g45)      ;    then goto read1;
4976
4976     i5: al  w0   0        ; done0:
4978     i6: jl  w3  g33       ; done:   prepare answer(s,c,addr);
4980         jl  w3  g18       ;         deliver result (1);
4982     i7: al  w3  g33       ; finis:
4984         am     (b19)      ;   interrupt addr(proc):= dummy;
4986         rs  w3  a56       ;   next operation;
4988         jl  w3  g25       ;   goto start;
4990         jl.     i9.       ;
4992
4992     i11:am       1        ; status2: count:= 2, or
4994     i10:am       1        ; status1: count:= 1, or
4996     i9: hl. w0  i12.      ; status0: count:= 0;
4998         rs  w1  x2        ;   word (addr):= word;
5000         sl  w0   0        ;   if not buffer end
5002         jl.     i6.       ;   then goto done;
5004         am     (b19)      ;
5006         ds  w3  a77       ;   wait buffered;
5008         jl  w3  g36       ;   continue buffered;
5010         jl  w3  g37       ;
5012         rs. w3  i13.      ;   examine sender(done);
5014         jl  w3  g34       ;
5016         jl.     i6.       ;   return to link - 10;
5018         rl. w3  i13.      ;
5020         jl      x3-10     ;
5022   h.i12: 0,1,2,3          ; counts
5026   w.i13: 0                ; saved link
5028
5028     i14:al. w3  i7.       ; sense it:
5030         sn  w0   0        ;   if operation <> 0 then
5032         jl  w0  g26       ;   begin
5034         al  w0   0        ;      mode(proc):= 0;
5036         rs  w0  x1+a70    ;      sense1 (proc);
5038         rl  w2  x1+a50    ;      if ex<>0 then exception;
5040         io  w3  x2+4      ;   end;
5042         sx       2.11     ;
5044         jl. w3  i8.       ;   no operation;
5046         jl.     i14.      ;   goto finis;
5048
5048     i15:rl  w1  x2        ; start write:
5050         io  w0 (g42)      ;   word:= word (addr);
5052         sx       2.11     ;   status:= sense0(device);
5054         jl. w3  i8.       ;   if ex<> 0 then exception;
5056         sh  w0  -1        ;   if buf full then wait;
5058         jl. w3  i16.      ;
5060         ld  w1   8        ;   char:= word(0:7);
5062         io  w0 (g46)      ;   word:= word shift 8;
5064         sx       2.11     ;   write(device,char);
5066         jl. w3  i8.       ;   if ex<>0 then exception;
5068         io  w0 (g42)      ;   status:= sense0 (device);
5070         sx       2.11     ;   if ex<>0 then exception;
5072         jl. w3  i8.       ;   if buf full then wait;
5074         sh  w0  -1        ;
5076         jl. w3  i16.      ;   char:= word(0:7);
5078         ld  w1   8        ;   word:= word shift 8;
5080         io  w0 (g46)      ;   write(device,char);
5082         sx       2.11     ;   if ex<>0 then exception;
5084         jl. w3  i8.       ;   status:= sense0(device);
5086         io  w0 (g42)      ;   if ex<>0 then exception;
5088         sx      2.11      ;   if buf full then wait;
```

```
5090          jl. w3  i8.       ;
5092          sh  w0  -1        ;   char:= word(0:7);
5094          jl. w3  i16.      ;
5096          ld  w1   8        ;   write(device,char);
5098          io  w0 (g46)      ;   if ex<>0 then exception;
5100          sx       2.11     ;
5102          jl. w3  i8.       ;   addr:= addr + 2;
5104          al  w2  x2+2      ;   examine sender(done0);
 106          jl  w3  g34       ;
 108          jl.     i5.       ;   if addr<= last addr
5110          sh  w2 (g45)      ;   then goto start write;
5112          jl.     i5.       ;   goto done0;
5114          jl.     i5.       ;
5116
5116     i16:am      (b19)      ; wait:
5118          ds  w3  a77       ;   wait buffered;
5120          jl  w3  g36       ;   continue buffered;
5122          jl  w3  g37       ;   return to link-10;
5124          jl      x3-10     ;
5126
5126  e.      ; end for rc 4124 mstc (simple)
5126  z.      ~~h26=g3~~   <- do not forget.
5126
5126
5126
5126
5126  ; rc 4194 drafting machine
 126
5126  ; process description format:
5126
5126  ; a10:  <kind=54>
5126  ; a11:  <name>
5126  ; a50:  <device number*64>
5126  ; a52:  <reserved>
5126  ; a53:  <users>
5126  ; a54:  <next message>
5126  ; a55:  <last message>
5126  ; a56:  <interrupt addr=c33>
5126  ; a70:  <max bytes-2>
5126
5126  c.(:a91>7a.1:)-1              ; if include rc 4194 drafting machine
5126  b.i24, a0=1<23                ; then
5126  w.      a0>0  +a0>5  +a0>10   ; begin
5128    i0:   a0>0                  ;
5130    h24:  jl  w3  g15           ;   check reservation;
5132          dl. w1  i0.           ;   check operation(0.5.10,0);
 134          jl  w3  g16           ;   link operation;
5136          jl  w3  g17           ;
5138
5138    i1:   se  w0   5            ; start:
5140          jl.     i4.           ;   if op <> 5 then goto not out;
5142          jl  w3  g31           ; output: increase stopcount;
5144          dl  w0  x2+12         ;
5146          wa  w3  x1+a70        ;   max:= first addr(buf) + max bytes-2;
5148          sl  w0  x3            ;   if last addr(buf) >= max then
5150          al  w0  x3            ;   last addr:= max;
5152          rl  w3  x1+a50        ;
5154          io  w0  x3+5          ;   transfer(device,last addr);
5156          sx      2.11          ;   if ex <> 0 then goto disconnect;
5158          jl.     i14.          ;
5160          rl  w0  x2+10         ;   transfer(device,first addr);
5162          io  w0  x3+17         ;   if ex <> 0 then goto disconnect;
5164          sx      2.11          ;
5166          jl.     i14.          ;   wait interrupt (proc);
5168          jl  w3  c32           ;
 170
5170          am     (x1+a50)       ; block completed:
5172          io  w0   4            ;   top addr:= sense1 (device);
5174          sx      2.11          ;   if ex <> 0 then goto disconnect;
5176          jl.     i14.          ;
5178          ws  w0  x2+10         ;   bytes:= top addr - first addr(buf);
5180          jl. w3  i8.           ;   sense it(status,bytes);
```

```
5182          jl   w3   g32        ;   decrease stopcount;
5184     i2:  jl   w3   g18        ; done: deliver result (1);
5186     i3:  jl   w3   g25        ; next: next operation(dummy interrupt);
5188          jl.       i1.        ;   goto start;
5190
5190     i4:  al.  w3   i2.        ; not out:
5192          se   w0   10         ;   if op=10 then
5194          jl.       i7.        ;   begin  transfer(device,unload);
5196          am        (x1+a50)   ;          sense it (status,0);
5198          io        9          ;          result:= 1; goto deliver;
5200          al.  w3   i12.       ;   end else
5202     i7:  al   w0   0          ;   sense it (status,0);
5204
5204   ; procedure sense it (status,bytes);
5204   ; senses the device and computes characters transferred.
5204   ;         call:      return:
5204   ; w0      bytes      status
5204   ; w1      proc       proc
5204   ; w2      buf        buf
5204   ; w3      link       link
5204
5204     i8:  rs   w0   g21        ; sense it:
5206          as   w0   -1         ; begin
5208          wa   w0   g21        ;   bytes(io answer):= bytes;
5210          rs   w0   g22        ;   chars(io answer):= 3*bytes/2;
5212     i10: io   w0   (x1+a50)   ;   status:= sense0 (device);
5214          sx        2.11       ;   if ex <> 0 then goto disconnect;
5216          jl.       i14.       ;
5218          rs   w0   g20        ;   status(io answer):= status;
5220          so.  w0   (i15.)     ;   if status(7)=0 then
5222          jl        x3         ;   return;
5224          jl   w3   g32        ;   decrease stopcount;
5226
5226     i11: am        4          ; unknown:  result:= 5;
5228     i12: al   w0   1          ; deliver:
5230          jl   w3   g19        ;   deliver result (result);
5232          rl   w1   b19        ;   buf:= next(mess q(proc));
5234          rl   w2   x1+a54     ;   if buf <> mess q(proc)
5236          rs   w2   b18        ;   then goto unknown;
5238          se   w2   x1+a54     ;   comment: reject all messages
5240          jl.       i11.       ;            still in queue;
5242
5242     i13: al   w0   0          ; remove process:
5244          rs   w0   x1+a11     ;   name(proc):= 0;
5246          rs   w0   x1+a52     ;   reserved(proc):= 0;
5248          am        (x1+a50)   ; clear status:
5250          io        5          ;   transfer(device,last addr=0);
5252          jl        c33        ;   goto dummy interrupt;
5254
5254     i14: jl   w3   g32        ; disconnect: decrease stopcount;
5256          jl   w3   g29        ;   disconnected device;
5258          jl.       i3.        ;   goto next;
5260
5260     i15: a0>7                 ; end sense it;
5262
5262   e.   ~~; end of rc 4194 drafting machine driver~~
5262   z.       ~~m2+5g3~~         ;   else goto result5;
5262
5262   ; rc 747 magnetic tape:
5262   ; rc 749 magnetic tape:
5262   ;
5262   ; process description format:
5262   ;
5262   ; a10: <kind=if rc 749 then 34 else 18>
5262   ; a11: <name>
5262   ; a50: <device number*64>
5262   ; a52: <reserved>
5262   ; a53: <users>
5262   ; a54: <next message>
5262   ; a55: <last message>
5262   ; a56: <interrupt address=c37>
5262   ; a70: <state>
```

do not forget.



```
5262  ; a71: <file count>
5262  ; a72: <block count>
5262~~; a73: <read op>~~
5262  ; a74: <w~~rite~~ op>
5262  ; a75: <er~~ase~~ op>    82
5262  ; a76: <r~~ead~~ par>    34
5262  ; a77: <~~write~~ par>   36
5262  ; a78: ~~<erase p~~ar>   38
 262  ;
 262  ;     state: 0  named tape in remote state
5262  ;            1  unknown tape in local state
5262  ;            2  unknown tape in remote state
5262
5262~~[illegible]~~                       ; if include rc 747 magnetic tape
5262                               ; or include rc 749 magnetic tape then
5262  b.i40,a0=1<23                ; begin
5262  w.      a0>0+a0>3+a0>5+a0>6+a0>8+a0>10
5264     i0: a0>0+a0>2             ;
5266    h12:jl   w3   g15          ;    check reservation;
5268        dl.  w1   i0.          ;
5270        jl   w3   g16          ;    check operation(0.3.5.6.8.10, 0.2);
5272        jl   w3   g17          ;    link operation;
5274     i1: am       (0)          ; start:
5276        jl.       (2)          ;    goto case operation(buf) of
5278        i8                     ;    (0: sense,
5280        i2                     ;     3: input output,
5282        i2                     ;     5: input output,
 284        i7                     ;     6: erase,
5286        i5                     ;     8: move,
5288        i6                     ;    10: output mark);
5290    i2: jl   w3   g31          ; input output:
5292        jl.  w3   i20.         ;    increase stop count;
5294        rl   w1   x1+a50       ;    sense magtape(status);
5296        rl   w3   x2+12        ;
5298        io   w3   x1+5         ;    transfer(device,last address(buf));
5300        ~~sx      2.11~~        ;
5302        ~~jl.     i10.~~        ;    if ex<>0 then goto disconnect;
5304        rl   w3   x2+10        ;
5306        bz   w0   x2+8         ;
5308        sn   w0   5            ;    if operation(buf)=5
5310        al   w1   x1+4         ;    then output(device,
5312        bz   w0   x2+9         ;         first addr(buf),mode(buf));
5314  sz    sn   w0   2            ;    else input(device,
5316        am        32           ;           first addr(buf),mode(buf));
5318
5318        io   w3   x1+13        ;
 320        ~~sx      2.11~~        ;
5322        ~~jl.     i10.~~        ;    if ex<>0 then goto disconnect;
5324        rl   w1   b19          ;
5326        jl   w3   c32          ;    wait interrupt(proc);
5328        ~~bz   w3   x2+8~~     ;
5330        ~~al   w2   1~~        ;    if operation(buf)=input
5332        ~~am        x3-3~~     ;    then read op(proc):=
5334        ~~wa   w2   x1+a73~~   ;    read op(proc)+1
5336        ~~am        x3-3~~     ;    else write op(proc):=
5338        ~~rs   w2   x1+a73~~   ;    write op(proc)+1;
5340        jl.  w3   i20.         ;    sense magtape(status);
5342        ~~rl   w2   b18~~      ;
5344        ~~bz   w3   x2+8~~     ;    if status(1)=1 and status(7)=0 then
5346        ~~al   w2   1~~        ;    begin
5348        ~~am        x3-3~~     ;    if operation(buf)=input
5350        ~~al   w3   x1+a76~~   ;    then read par(proc):= read par(proc)+1
5352        ~~wa   w2   x3+0~~     ;    else write par(proc):=
5354        ~~sz.  w0   (i27.)~~
5356        ~~jl.       6~~
 358        ~~sz   w0   (a58)~~    ;    write par(proc)+1
5360        ~~rs   w2   x3+0~~     ;    end;
5362        jl.  w3   i21.         ;    next block;
5364        am        (x1+a50)     ;
5366        io   w0   4            ;    characters:=sense size(device);
5368        ~~sx      2.11~~        ;
5370        ~~jl.     i10.~~        ;    if ex<>0 then goto disconnect;
```

```
bz  w0   x2+9  ;
~~bz  w0~~
ls  w0   -6    ;
io  w0   x1+25 ;
```

```
          rs  w0  g22       ;
       ~~[illegible]~~      ;
          al  w3  0         ;
       ~~[illegible]~~      ;   words:=characters/
       ~~[illegible]~~      ;   (if kind(proc)=34 then 3
          wd. w0  i12.      ;        else 4);
          se  w3  0         ;   if remainder<>0 then
          ba. w0  1         ;   words:=words+1;
          ls  w0  1         ;
          rs  w0  g21       ;   bytes:=words*2;
     i3:  jl  w3  g32       ; done 0:  decrease stop count:
          jl  w3  g18       ;   deliver result(1);
     i4:                    ; done 1:
          jl  w3  g64       ;   examine queue(
          jl.     i16.      ;     idle tape);
          jl.     i1.       ;   goto start;

     i5:  jl. w3  i20.      ; move:
          rl  w3  x2+10     ;   sense magtape(status);
          sl  w3  0         ;   if move operation(buf)<0
          sl  w3  6         ;   or move operation(buf)>5
          jl.     i8.       ;   then goto sense;
          am      (x1+a50)  ;
          io  w3  9         ;   move(device,move operation(buf));
       ~~[illegible]~~      ;
       ~~[illegible]~~      ;   if ex<>0 then goto disconnect;
          jl  w3  c32       ;   wait interrupt(proc);
          jl. w3  i20.      ;   sense magtape(status);
          rl  w2  x2+10     ;
          jl.     x2+2      ;   case move operation(buf) of
                            ;   (0: next block,
0 1       am      i29       ;    1: next block,
                            ;    2: last block,
2 3       am      i31       ;    3: last block,
                            ;    4: load point,
4 5       jl. w3  i25.      ;    5: load point);
          jl.     i9.       ;   goto size zero;

     i6:  jl. w3  i20.      ; output mark:
          rl  w2  x1+a10    ;   sense magtape(status);
     ~~     se  w2  34        ;   if kind=34 then~~
     ~~     jl.     i18.      ;   begin~~
          am      (x1+a50)  ;   write(device);
          io      3         ;   goto after mark
     ~~     jl.     i19.      ;   end;~~
     ~~i18: al. w3  i17.~~
     ~~     am      (x1+a50)  ;~~
     ~~     io  w3  5         ;   transfer(device,tape mark):~~
     ~~     sx      2.11      ;~~
     ~~     jl.     i10.      ;   if ex<>0 then goto disconnect;~~
     ~~     am      (x1+a50)  ;~~
     ~~     io  w3  49        ;   output(device,tape mark,even);~~
     ~~i19: sx      2.11      ; after mark:~~
     ~~     jl      i10.      ;   if ex<>0 then goto disconnect;~~
          jl  w3  c32       ;   wait interrupt(proc);
          jl. w3  i20.      ;   sense magtape(status);
          jl. w3  i21.      ;   next block;
          jl.     i9.       ;   goto size zero;            al. w0 i2.

     i7:  jl. w3  i20.      ; erase:
          am      (x1+a50)  ;   sense magtape(status);
          io      21        ;   erase tape(device);
       ~~[illegible]~~      ;
       ~~[illegible]~~      ;   if ex<>0 then goto disconnect;
          jl  w3  c32       ;   wait interrupt;
     ~~     al  w2  1~~       ;                              jl. w3 i21. next block
     ~~     wa  w2  x1+a75~~  ;   erase op(proc):=
     ~~     rs  w2  x1+a75~~  ;   erase op(proc)+1;
     i8:  jl. w3  i20.      ; sense: sense magtape(status):
     i30: jl. w3  i25.      ;   load point;
     i9:  al  w3  0         ; size zero:
          rs  w3  g21       ;   bytes:=
```

backspace

```
        rs   w3   g22          ;   characters:=0;
        jl.       i3.          ;   goto done 0;

    i10:                       ; disconnect:
        jl   w3   g32          ;   decrease stop count;
        jl   w3   g29          ;   disconnected device;
        jl.       i4.          ;   goto done 1;
[redacted]                     ;
    i12: 3                     ;

    i14:                       ; intervention:

        jl   w3   g32          ;   decrease stop count;
    i13:al   w0   5            ; reject message:
        jl   w3   g19          ;   deliver result(5);
        rl   w1   b19          ;
        rl   w2   x1+a54       ;
        rs   w2   b18          ;   buf:=next(mess q(proc));
        se   w2   x1+a54       ;   if buf<>mess q(proc)
        jl.       i13.         ;   then goto reject message;
        al   w0   1            ;
        rs   w0   x1+a70       ;   state(proc):=1;
    i15:al   w0   0            ; remove name:
        rs   w0   x1+a11       ;   name(proc):=0;
        rs   w0   x1+a52       ;   reserved(proc):= 0;
        al   w0   -1           ;
        rs   w0   x1+a71       ;   file(proc):= -1;
        am       (x1+a50)      ; remove intervention:
        io        5            ;   transfer(device,irrelevant);

    i16:jl   w3   c32          ; idle tape:
    c37:al   w0   2            ;   wait interrupt(proc);
        rs   w0   x1+a70       ;   state(proc):=2;
        jl.       i15.         ;   goto remove name;

[redacted]                     ; tape mark:

; procedure sense magtape(status)
;       call:     return:
; w0              status
; w1    proc      proc
; w2              unchanged
; w3    link      link

b.j24                          ; begin
w.i20:rs. w3   j0.
        io   w0 (x1+a50)       ;   status:=sense(device);
[redacted]                     ;
                               ;   if ex<>0 then goto disconnect;
        rs   w0   g20          ;
        sh   w0   -1           ;   if status(0)=1
        jl.       i14.         ;   then goto intervention;
        am       (b18)         ;   if (operation(buf)=5
        bz   w3   8            ;   or operation(buf)=6
        so   w3   2.0100       ;   or operation(buf)=10)
        sn   w3   10           ;   and status(8)=0
        sz.  w0 (i28.)         ;   then begin
        jl.      (j0.)         ;
        am       (x1+a50)      ;        transfer(device,irrelevant);
        io        5            ;        load point;
        io   w0 (x1+a50)       ;
        rs   w0   g20          ;
        jl.       i30.         ;        goto size zero
j0: 0                          ;      end;
e.                             ; end

; procedure next block
; procedure next file
;      call:     return:
; w0   status    status
; w1   proc      proc
; w2             unchanged
```

```
5598 ; w3  link      link
5598
5598 b.j24                    ; begin next block:
5598 w.i21:sz. w0 (i27.)      ;   if status(7)=1
5600       jl.      i22.      ;   then next file else
5602       rx   w3  x1+a72    ;   begin
5604       so   w0 (g59)      ;   if status(2)=0 then
5606       al   w3  x3+1      ;   block(proc):=block(proc)+1:
5608                          ;   load point;
5608       jl.      i40.      ;   end;
5610 e.                       ; end
5610
5610 b.j24                    ; begin next file:
5610 w.i22:rx   w3  x1+a71    ;
5612       al   w3  x3+1      ;
5614       rx   w3  x1+a71    ;   file(proc):= file(proc) + 1;
5616       rx   w3  x1+a72    ;
5618       al   w3  0         ;
5620                          ;   block(proc):= 0;
5620       jl.      i40.      ;   load point;
5622 e.                       ; end
5622
5622
5622 ; procedure last block
5622 ; procedure last file
5622 ;     call:    return:
5622 ; w0  status   status
5622 ; w1  proc     proc
5622 ; w2           unchanged
5622 ; w3  link     link
5622
5622 b.j24                    ; begin last block:
5622 w.i23:sz. w0 (i27.)      ;   if status(7)=1
5624       jl.      i24.      ;   then last file else
5626       rx   w3  x1+a72    ;   begin
5628       al   w3  x3-1      ;   block(proc):=block(proc)+1:
5630                          ;   load point;
5630       jl.      i40.      ;   end;
5632 e.                       ; end
5632
5632 b.j24                    ; begin last file:
5632 w.i24:rx   w3  x1+a71    ;
5634       al   w3  x3-1      ;
5636       rx   w3  x1+a71    ;   file(proc):=file(proc)-1;
5638       rx   w3  x1+a72    ;
5640       al   w3  -1        ;
5642   i40:rx   w3  x1+a72    ;   block(proc):=-1;
5644                          ;   load point;
5644 e.                       ; end
5644
5644 ; procedure load point
5644 ;     call:    return:
5644 ; w0  status
5644 ; w1  proc     proc                     Ld w3 -65
5644 ; w2           destroyed
5644 ; w3  link     destroyed
5644
5644 b.j24                    ; begin
5644 w.i25:rs.  w3  j1.       ;
5646       so.  w0 (i26.)     ;   if status(6)=1
5648       jl.      j0.       ;   then
5650       ~~al   w2  0~~         ;   file(proc):=
5652       ~~al   w3  0~~         ;   block(proc):=0;
5654       ds   w3  x1+a72    ;
5656   j0: dl   w3  x1+a72    ;   file count:=file(proc);
5658       ds   w3  g24       ;   block count:=block(proc);
5660       jl.     (j1.)      ;
5662   j1: 0                  ;
5664 e.                       ; end
5664
5664   i26: 1<23>6
5666   i27: 1<23>7
```

```
   i28: 1<23>8
   i29=i21-i23
   i31=i23-i25

e.     ; end of rc 747 magnetic tape;
       ; end of rc 749 magnetic tape;
[redacted]                   ;   goto result 5;
                             ; goto result 5;

; interrupt key;

; process description format:

; a10: <kind=if interrupt key then 32 else 22>
; a11: <name>
; a50: <device number*64>
; a52: <reserved>
; a53: <users>
; a54: <next message>
; a55: <last message>
; a56: <interrupt address=c40>
; a70: <interrupts>

c.(:a91>16a.1:)-1          ; if include ixp 401 interrupt register
                           ; or include interrupt key then
b.i24     ,a0=1<23         ; begin
w.        a0>0
  i0:     a0>0
  h14:    jl w3 g15        ;   check reservation;
          dl.w1 i0.
          jl w3 g16        ;   check operation(0,0);
          jl w3 g17        ;   link operation;
  i4:     rl w0 x1+a10     ; start:
          sh w0 22         ;   if kind(proc)=22 then
          jl.   i5.        ;   goto sense;
          rl w3 x1+a70
          se w3 0          ;   if interrupts(proc)<>0 then
          jl.   i2.        ;   goto key return;
  i1:                      ; wait:
          jl w3 c32        ;   wait interrupt(proc);
  c40:    rl w0 x1+a10
          se w0 32         ;   if kind(proc)<>32 then
          jl.   i5.        ;   goto sense;
          al w3 1          ;   status:=1;
          jl.   i2.        ;     goto key return;
  i5:     al w3 0          ; sense:
          io w3(x1+a50)    ;   status:= sense(device);
          sx    2.11       ;   if ex<>0
          am    3          ;   then begin status:=0;result:=4 end
  i2:     al w0 1          ;   else key return:result:=1;
          lo w3 x1+a70
          rs w3 x1+a70     ;   status:=intpts(proc):=intpts(proc) or status:
          rl w2 x1+a54
          se w0 4          ;   if (result=1
          se w3 0          ;     and status=0)
          sn w2 x1+a54     ;     or next(mess q(proc))=mess q(proc) then
          jl.   i1.        ;     goto wait:
                           ;   register value:= status;
          al w2 0          ;   status:= 0;
          rs w2 x1+a70     ;   interrupts(proc):=0;
          ds w3 g21        ;   deliver result(result);
          jl w3 g19
          jl w3 g64        ;   examine queue(
          jl.   i1.        ;     wait);
          jl.   i4.        ;   goto start;
e.                         ; end of interruot key;
z.  [redacted]             ; goto result 5;

                           ; goto result 5;
                           ; goto result 5;
                           ; goto result 5;

```

```
5670  ; dpc 405 alphanumeric display:
5670
5670  ; process description format:
5670
5670  ; a10: <kind=30>
5670  ; a11: <name>
5670  ; a50: <device number*64>
5670  ; a52: <reserved>
 670  ; a53: <users>
5670  ; a54: <next message>
5670  ; a55: <last message>
5670
5670  c.(:a91>12a.1:)-1       ; if include dpc 405 alphanumeric display then
5670  b.i24     ,a0=1<23      ; begin
5670  w.        a0>5
5670     i0:    a0>0
5670     h18:   jl w3 g15     ;   check reservation;
5670            dl.w1 i0.
5670            jl w3 g16     ;   check operation(5,0,2);
5670            rl w1 b19     ;   proc:=current receiver;
5670     i1:    rl w3 x1+a50  ; start:
5670            al w0 x3+3    ;   write addr:= sense addr(proc)+3;
5670            rs.w0 i6.
5670            al w3 17
5670            io.w3(i6.)    ;     write device(17);
5670            sx    2.11    ;   if ex<>0 then
5670            jl.   i4.     ;     goto disconnect;
 670            rl w1 x2+10   ;   addr:= first addr(buf);
5670            al w0 x1+88
5670            sh w0(x2+12)  ;   if last addr(buf)>=addr+88 then
5670            rs w0 x2+12   ;     last addr(buf):= addr+88;
5670     i2:                  ; next word:
5670            al w3 -16     ;   char shift:= -16;
5670     i3:                  ; next char:
5670            rl w0 x1+0    ;   word:= word(addr);
5670            ls w0 x3+0    ;   word:= word shift char shift;
5670     i5:    io.w0(i6.)    ; write:write device(word);
5670            sx    2.10    ;   if ex=2 then
5670            jl.   i4.     ;     goto disconnect;
5670            sx    2.01    ;   if ex=1 then
5670            jl.   i5.     ;   goto write;
5670            al w3 x3+8    ;   char shift:= char shift+8;
5670            sh w3 0       ;   if char shift<=0 then
5670            jl.   i3.     ;     goto next char;
5670            al w1 x1+2    ;   addr:= addr+2;
5670            sh w1(x2+12)  ;   if addr<=last addr(buf) then
 670            jl.   i2.     ;     goto next word;
5670            al w0 18      ;   word:=18;
5670            sh w3 8       ;   if char shift <=8 then
5670            jl.   i5.     ;   goto write;
5670            al w0 0
5670            al w2 x1-2
5670            jl w3 g33     ;   prepare answer(0,0,addr);
5670            am    g18-g29 ;   deliver result(1);
5670                          ;   goto done1;
5670     i4:    jl w3 g29     ; disconnect:
5670                          ;   disconnected device;
5670                          ; done1:
5670            jl   (b20)    ;   goto return;
5670     i6:    0 ; write addr
5670  e.                      ; end of dpc 405 alphanumeric display;
5670  z.  [redacted]          ; goto result 5;
5670
5670  ; operator process;
5670
 670  ; process description format:
5670
5670  ; a10: <kind=38>
5670  ; a11: <name>
5670  ; a50: <device number*64>
5670  ; a52: <reserved>
5670  ; a53: <users>
```

```
; a54: <next message>
; a55: <last message>

b.i24                    ; begin
w.h22:  al w1 x3+a54
        jl w3 d6         ;   link(event q(proc),buf);
        rs.w1 i1.        ;   buf:= event q(proc);
        al w0 0          ; next:
  i0:   rl w1 x1         ;   buf:= next(buf);
  i2:   sn.w1(i1.)       ; test buf:
        jl    (b20)      ;   if buf=event q(proc)
        sh w0(x1+6)      ;   then goto return;
        jl.   i0.        ;   if sender(buf) >= 0
        al w2 x1         ;   then goto next;
        rl w1 x1         ;   oldbuf:=buf: buf:=next(buf);
        jl w3 d15        ;   deliver answer(oldbuf);
        jl.   i2.        ;   goto test buf;
  i1:   0                ; end operator process:
e.
[redacted]

; rc 4195 graphic display:

; process description format:

; a10: <kind=40>
; a11: <name>
; a50: <device number*64>
; a52: <reserved>
; a53: <users>
; a54: <next message>
; a55: <last message>
; a56: <interrupt addr=c33>
; a70: <max bytes>
; a71: <timer count>

c.(:a91>11a.1:)-1        ; if include rc 4195 graphic display then
b.i24   ,a0=1<23         ; begin
w.      a0>0+a0>3+a0>5   ;
  i0:   a0>0+a0>2        ;
  h19:  jl w3 g15        ;   check reservation;
        dl.w1 i0.        ;
        jl w3 g16        ;   check operation(0.3.5,0.2);
        jl w3 g17        ;   link operation;
  i1:                    ; start:
        se w0 0          ;   if operation(buf)<>0 then
        jl.   i3.        ;     goto input output;
        rl w0 x2+12      ;   timer count(proc):=
        rs w0 x1+a71     ;       max timer count(buf);
  i2:   am    (b18)      ; enable:
        rl w0 10         ;
        am    (x1+a50)   ;
        io w0 3<2+1      ;   control3(word(buf+10)):
        sx    2.11       ;   if ex<>0 then
        jl.   i11.       ;     goto disconnect;
        jl w3 c32        ;   wait interrupt(proc);
        am    (x1+a50)   ;
        io w0 3<2+0      ;   status(buf):= sense3(sense addr);
        sx    2.11       ;   if ex<>0 then
        jl.   i11.       ;     goto disconnect;
        rs w0 x2+8       ;
        al w3 -1         ;
        wa w3 x1+a71     ;   timer count:= timer count-1;
        rs w3 x1+a71     ;
        so w0(g59)       ;   if status(2)=0 then
        jl.   i12.       ;     goto sense;
        sl w3 0          ;   if timer count>=0 then
        jl.   i2.        ;     goto enable;
        ld w0 -65        ;   lpx(buf):= lpy(buf):= 0;
        jl.   i10.       ;   goto done;
  i12:  rl w3 x1+a50     ; sense:
        io w0 x3+2<2+0   ;   lpy:= sense2(sense addr);
```

```
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698             io w3 x3+1<2+0      ;     lpx:= sense1(sense addr);
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698             jl.   i10.          ;     goto done;
5698       i3:   rl w3 0             ;   input output:
5698             ls w3 1             ;     i:= operation(buf)*2
5698             ba w3 x2+9          ;        +mode(buf);
5698             jl.   x3-4          ;     repeat addr:= case i of(
5698             am    i15           ;        6:  read,
5698             am    i14           ;        8:  read point,
5698             am    i17           ;       10:  write,
5698             al.w3 i6.           ;       12:  write point);
5698             rs.w3 i22.          ;
5698             ba.w0 1             ;
5698             ls w0 -1            ;     if operation(buf)=3 then
5698             wa w0 x1+a50        ;       read addr:= sense addr(proc)+2
5698             al w3 1             ;     else write addr:= sense addr(proc)+3;
5698             wa w3 x1+a50        ;
5698             ds.w0 i21.          ;     control addr:= sense addr(proc)+1;
5698             rl w0 x2+14         ;
5698             io.w0(i20.)         ;     control(control addr,word(buf+14);
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698             rl w3 x2+10         ;     max addr:= first addr(buf)
5698             al w0 x3-2          ;         -2+max bytes(proc);
5698             wa w0 x1+a70        ;     if max addr<last addr(buf) then
5698             sl w0(x2+12)        ;       last addr(buf):= max addr;
5698             rl w0 x2+12         ;     last addr(buf):=
5698             bs w0 x2+9          ;         last addr(buf)-mode(buf);
5698             rs w0 x2+12         ;     addr:= first addr(buf);
5698             jl.   i9.           ;     goto check addr;
5698       i4:   rl w0 x3+0          ;   read point:
5698             io.w0(i20.)         ;     control(control addr, word(addr));
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698             al w3 x3+2          ;     addr:= addr+2;
5698       i5:                       ;   read:
5698             io.   (i21.)        ;     read(read addr);
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698             io w0(x1+a50)       ;     word:= sense(sense addr);
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698             rs w0 x3+0          ;     word(addr):= word;
5698             jl.   i8.           ;     goto increase addr;
5698       i6:   rl w0 x3+0          ;   write point:
5698             io.w0(i20.)         ;     control(control addr, word(addr));
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698             al w3 x3+2          ;     addr:= addr+2;
5698       i7:   rl w0 x3+0          ;   write:
5698             io.w0(i21.)         ;     write(write addr, word(addr));
5698             sx    2.11          ;     if ex<>0 then
5698             jl.   i11.          ;       goto disconnect;
5698       i8:                       ;   increase addr:
5698             al w3 x3+2          ;     addr:= addr+2;
5698       i9:                       ;   check addr:
5698             sh w3(x2+12)        ;     if addr<=last addr(buf) then
5698             jl.  (i22.)         ;       goto repeat addr;
5698             ws w3 x2+10         ;     status(buf):= 0;
5698             al w0 0             ;     bytes(buf):= addr-first addr(buf);
5698             rs w0 x2+8          ;     characters(buf):= 0;
5698       i10:  ds w0 x2+12         ;   done:
5698             al w0 1             ;
5698             rs w0 x2+4          ;     receiver(buf):= 1;
5698             am    d15-g29       ;     deliver answer(buf);
5698                                 ;     goto done 1;
5698       i11:                      ;   disconnect:
5698             jl w3 g29           ;     disconnected device;
5698                                 ;   done 1:
```

```
5698          jl  w3  g25          ;    next operation;
5698          jl.     i1.          ;    goto start;
5698
5698     i15=i5-i4
5698     i14=i4-i7
5698     i17=i7-i6
5698     i20:   0          ; control addr
5698     i21:   0          ; read addr, write addr
 698     i22:   0          ; repeat addr
5698   e.                  ; end of rc 4195 graphic display
5698   [redacted]
5698          [redacted]           ; goto result 5;
5698          [redacted]           ; goto result 5;
5698
5698   m.
5698                       monitor text 2 included
5698
5698
```

```
5698
5698  m.
5698                       monitor text 3
5698  m.
5698
 698
5698
5698
5698  ; rc 4124 telex line
5698' ;
5698  ; process description format:
5698  ;
5698  ; a10: <kind=58>
5698  ; a11: <name>
5698  ; a50: <device number*64>
5698  ; a52: <reserved>
5698  ; a53: <users>
5698  ; a54: <next message>
5698  ; a55: <last message>
5698  ; a56: <interrupt address=c33>
5698  ; a70: <state=0 (not used)>
5698  ; a71: <start count><telex case>
5698  ; a72: <timer count><max count>
5698  ; a73: <address>
 698  ; a74: <last address>
5698  ; a75: <word>
5698  ; a76: <character shift>
5698  ; a77: <link>
5698  ;
5698  ;         telex case = 0:  letter case
5698  ;                     =32:  figure case
5698
5698  c.(:a91>5a.1:)-1          ; if include rc 4124 telex then
5698  b.i24,a0=1<23             ; begin
5698  w.     a0>0+a0>3+a0>5     ;
5700    i0: a0>0                ;
5702    h26:jl   w3   g15       ;   check reservation;
5704        dl.  w1   i0.       ;
5706        jl   w3   g16       ;   check operation (0.3.5,0);
5708        jl   w3   g17       ;   link operation;
5710
5710    i1: dl   w2   x2+12     ; start:
5712        am        (0)       ;   addresses:= buf (10:12);
 714        jl.  w3   i3.       ;   goto select;
5716
5716    i2: jl   w3   g25       ; next:  next operation;
5718        jl.       i1.       ;        goto start;
5720
5720    i3: jl   w0   g30       ; select: case operation of
5722        am        i7        ;    (0:  sense device,
5724        jl.  w3   i6.       ;     3:  input,
5726        jl   w3   g33       ;     5:  output);
5728        am     g18-g29      ;         goto next;
5730
5730    i4: jl   w3   g29       ; disconnect: disconnected device;
5732        jl.       i2.       ;   goto next;
5734
5734  ; procedure input (first addr,last addr, status+count):
5734  ; comment: inputs characters from first to last address from telex
5734  ; lines. the input is terminated in the following situations:
5734  ;      1.   when the sending process is stopped or removed
5734  ;      2.   when the storage area is full
 734  ;      3.   after a maximum number of timer errors
5734  ;      4.   when the device is disconnected
5734  ;      5.   after input of a linefeed character
5734  ;      6.   after input of 8 consecutive start signals
5734  ; upon return, the address points to the last word to which 0,1,
5734  ; 2, or 3 characters were input (as defined by count). the charac-
5734  ; ters are converted from ccitt alphabet no.2 to ISO according to
```

```
5734  ; wmo conventions except that small letters are used instead of
5734  ; capitals.
5734  ;
5734  ; w0                                      status+count
5734  ; w1          first address               proc
5734  ; w2          last addr                   addr
5734  ; w3          link                        destroyed
5734
 734  b.j24                           ; begin
5734  w.i5: am       (b19)            ; input:
5736        ds   w2   a74             ;   addr (proc):= first addr;
5738        rl   w1   b19             ;   last addr(proc):= last addr:
5740        rs   w3   x1+a77          ;   link(proc):= link;
5742        al   w3    0              ;   timer count(proc):= 0;
5744        hs   w3   x1+a72          ;   start count(proc):= 0;
5746        hs   w3   x1+a71          ;
5748
5748    j0: al   w2   16              ; next word:  shift(proc):= 16;
5750    j1: rs   w2   x1+a76          ; next char:
5752    j2: am       (x1+a50)         ; repeat:
5754        io         2              ;   read (device);
5756        sx         3              ;   if ex <> 0 then
5758        jl.        i4.            ;   goto disconnect;
5760        jl   w3   c32             ;   wait interrupt(proc);
5762        io   w0  (x1+a50)         ;
5764        sx         3              ;   status:= sense (device);
5766        jl.        i4.            ;   if ex<>0 then goto disconnect:
 768        rl   w2   x1+a76          ;
5770        jl   w3   g34             ;   exam sender (done);
5772        jl.        j7.            ;
5774        sz   w0  (g59)            ;   if status(2)=1 then goto timer;
5776        jl.        j6.            ;
5778
5778    j3: al   w3   31              ; convert:
5780        la   w3    0              ;   char:= status(19:23);
5782        ba   w3   x1+a71+1        ;   char:= conversion(case+char);
5784        bz.  w0   x3+j20.         ;   terminate:= false;
5786        sl   w0   128             ;   if char >= 128 then goto special;
5788        jl.        j8.            ;
5790    j4: al   w3    0              ; pack:
5792        hs   w3   x1+a71          ;   start count(proc):= 0;
5794        ls   w0   x2              ;   char:= char shift (shift(proc)):
5796        se   w2   16              ;   if shift(proc) <> 16 then
5798        lo   w0  (x1+a73)         ;   char:= char or word (addr(proc));
5800        rs   w0  (x1+a73)         ;   word(addr(proc)):= char;
5802        al   w2   x2-8            ;   shift(proc):= shift(proc) -8;
 804        sx         1              ;   if terminate then goto done:
5806        jl.        j7.            ;
5808        sl   w2    0              ;   if shift(proc) >= 0
5810        jl.        j1.            ;   then goto next char;
5812
5812    j5: rl   w3   x1+a73          ; end word:
5814        sl   w3  (x1+a74)         ;   if addr(proc) >= last addr(proc)
5816        jl.        j7.            ;   then goto done;
5818        al   w3   x3+2            ;   addr(proc):= addr(proc)+2;
5820        rs   w3   x1+a73          ;   goto next word;
5822        jl.        j0.            ;
5824
5824  ; garbage interrupt
5824
5824   c50:rl   w1   x1+a56+2         ; garbage:
5826       rl   w0   x1+a54           ;   set buf and proc;
5828       ds   w1   b19              ;   if event q(proc) is empty
5830       sn   w0   x1+a54           ;   then goto interrupt return;
5832       jl       (b20)             ;
 834       jl       (b20)             ; anyhow: goto interrupt return:
5836
5836    j6: bz   w3   x1+a72          ; timer:
5838        al   w3   x3+1            ;   timer count(proc):=
5840        hs   w3   x1+a72          ;   timer count(proc) +1;
5842        bs   w3   x1+a72+1        ;   if timer count(proc) <
5844        sh   w3   -1              ;       max count(proc)
```

```
5846         jl.       j2.      ;    then goto repeat;
5848         am       (g59)     ;    status:= bit 2;
5850
5850    j7: al   w0    0        ; done:
5852        al   w2   x2-16     ;    count:= - (shift(proc)-16)/8;
5854    i8: as   w2   -3        ;    comment:
5856        ac   w2   x2        ;       w0(0:11) = status (0:11).
5858        hl   w0    5        ;       w0(12:23)= count;
5860        rl   w2   x1+a73    ;    addr:= addr(proc);
5862        jl       (x1+a77)   ;    goto link(proc);
5864
5864    j8: so   w0   512       ; special:
5866        jl.       j11.      ;    if char = caseshift then
5868        al   w3   32        ;    begin case:=32;
5870        sz   w0    1        ;       if char=letter shift then
5872        jl.       j9.       ;       begin case:= 0;
5874        bz   w3   x1+a71    ;          start(proc):= start(proc)+1;
5876        al   w3   x3+1      ;          if start(proc) < 8
5878        sz   w3   -8        ;          then goto repeat;
5880        jl.       j10.      ;       end else goto repeat;
5882        ls   w3   12        ;       char:= 127;
5884    j9: rs   w3   x1+a71+1  ;    end
5886        jl.       j2.       ;    else char:= 10;
5888    j10:am        117       ;
5890    j11:al   w0   10        ;    terminate:= true;
5892        xl.        1        ;    goto pack;
5894        jl.       j4.       ;
5896
5896  ; conversion table:  telex to ISO.
5896  ; contains one byte for each of the 32 values of ccitt alphabet no.2
5896  ; in either case. if some special action must be taken the converted
5896  ; value is greater than 127.
5896  ; letter shift (start signal)  converts to        512+0
5896  ; figure shift                      -             512+1
5896  ; linefeed                          -             256+10
5896  ; shift signals are normally not stored. the one exception to that
5896  ; is a sequence of 8 start signals which are stored as one ISO DEL
5896  ; character and at the same time works as a terminating symbol.
5896                  d             j
5896  h.j20:   0,101,266, 97  ;  nul  e   lf   a                j = 9
5900          32,115,105,117  ;  sp   s   i    u                d = 11
5904          13,100,114,106  ;  cr   j   r    d
5908         110,102, 99,107  ;   n   f   c    k
5912         116,122,108,119  ;   t   z   l    w
5916         104,121,112,113  ;   h   y   p    q
5920         111, 98,103,513  ;   o   b   g  figure shift
5924         109,120,118,512  ;   m   x   v  letter shift
5928
5928           0, 51,266, 45  ;  nul  3   lf   -
5932          32, 39, 56, 55  ;  sp   '   8    7
5936          13,  7, 52,266  ;  cr  bel  4   lf
5940          44,123, 59, 40  ;   ,   æ   :    (
5944          53, 43, 41, 50  ;   5   +   )    2
5948         125, 54, 48, 49  ;   å   6   0    1
5952          57, 63,124,513  ;   9   ?   ø  figure shift
5956          46, 47, 61,512  ;   .   /   =  letter shift
5960  w.
5960  e.                      ; end input;
5960
5960
5960  ; procedure output (first addr,last addr,status+count):
5960  ; comment: outputs the characters from first to last address on a
5960  ; telex line. the output is terminated in the following situations:
5960  ;      1.   when the sending process is stopped ore removed
5960  ;      2.   when the storage area is empty
5960  ;      3.   when the device is disconnected.
5960  ; upon return, the address points to the last word from which 0,1,
5960  ; 2, or 3 characters were output. the characters are converted to
5960  ; ccitt alphabet no.2 according to WMO recommandation except that
5960  ; small letters are allowed in the output.
5960  ;
5960  ; w0                                    status + count
```

```
6070        66, 0, 0,72, 0    ;  nl  vt  ff  cr  so
6075         0, 0, 0, 0, 0    ;  si  dle dc1 dc2 dc3
6080         0, 0,66, 0, 0    ;  dc4 nak syn etb can
6085         0, 0, 0, 0, 0    ;  em  suo esc fs  gs
6090         0, 0,68, 0, 0    ;  rs  us  sp   !   "
6095         0, 0, 0, 0,37    ;  35  36   %   &   '
6100        47,50, 0,49,44    ;   (   )   *   +   ,
6105        35,60,61,54,55    ;   -   .   /   0   1
 110        51,33,42,48,53    ;   2   3   4   5   6
6115        39,38,56,46, 0    ;   7   8   9   :   ;
6120         0,62, 0,57, 0    ;   <   =   >   ?
6125         3,25,14,11, 1    ;   A   B   C   D   E
6130        13,26,20, 6, 9    ;   F   G   H   I   J
6135        15,18,28,12,24    ;   K   L   M   N   O
6140        22,23,10, 5,16    ;   P   Q   R   S   T
6145         7,30,20,29,21    ;   U   V   W   X   Y
6150        17,45,58,52, 0    ;   Z   Æ   Ø   Å  94
6155         0, 0, 3,25,14    ;   _  96   a   b   c
6160        11, 1,13,26,20    ;   d   e   f   g   h
6165         6, 9,15,18,28    ;   i   j   k   l   m
6170        12,24,22,23,10    ;   n   o   p   q   r
6175         5,16, 7,30,19    ;   s   t   u   v   w
6180        29,21,17,45,~~58~~52;   x   y   z   æ   ø
6185        ~~52~~56, 0,31          ;   å 126 del
6188  w.                      ;
6188  e.   ;
6188  e.   ; end of rc 4124 telex driver
 188  ~~z.      h26 = g3~~        ;
6188
6188  ; rc 4124 www transmission line
6188  ~~        h23 = g4~~
6188
6188  ; rc 3200
6188  ~~        h25 = g5~~
6188
6188
6188
6188  b.i0                     ; begin
6188  w.i0: al. w2  i0.        ; make room:
6190        jl      x3+0       ;   autoloader(end external processes):
6192        jl.     i0.        ; after loading:
6194    g70= k-b127 + 2
6194    k = i0                 ;   goto make room;
6188  e.                       ; end
6188  i.
6188  e.    ; end of external process segment
 188
6188
```

8 4 2 1

9: 1001

11: 1101

Do not type as is! consult me! L.S.

```
6188
6188
6188  ; segment 4: process descriptions
6188
6188  s. k = k, h25, g65, e20, j20
  88  w.b127=k, h25, k=k-2
6188
6188  ; name table:
6188  ; the table has one entry for each process description. an entry
6188  ; contains the address of the corresponding process description.
6188
6188  w.f0: ; name table start:
6188         h22                  ; operator process
6190
6190  t.
6190* type

6190
6190  ; processes in name table before first device
6190
6190  n.m.
6190                     monitor external process list in name table included
6190    f1: ; first device in name table:
6190  t.
 190* type

6190
6190  ; device list in name table
6190
6190    g0 ,g1 ,g2 , g3 ,g4 ,g5 ,g6 ,g7 ,g8 ,g9
6210    g10,g11,g12,g13,g14,g15,g16,g17,g18,g19
6230    g20,g21,g22,g23,g24,g25
6242  n.m.
6242                     monitor device list in name table included
6242
6242    f2: ; first area in name table:
6242         h7, r.a1
6386    f3: ; first internal in name table:
6386         h8, r.a3
6426    f4: ; name table end:
6426         0
6428
6428
 428  ; dummy internal process:
6428         f5=k-a24
6428         f6=f5+a16
6428         1<19
6430         a89,h1
6434         0,r.5,h0,0,r.4
6454    h0: je.     h0.
6456        je.     h0.
6458
6458    h4: 0,0, <:unknown:>,0,r.5,c33
6480    h1: 0,r.7
6494        jd.     h0.
6496    h22:38,<:operator:>,0
6506        0,0,-1,k,k-2,c33
6518
6518  ; external processes
6518  t.
6518* type

 518
6518  ; descriptions of external processes
6518
6518  n.m.
6518                     monitor external process descriptions included
6518
6518  ; console keys:
```

```
6518
6518       h5: al   w1   x1-a56
6520           jl        c30
6522
6522  ; peripheral processes:
6522  t.
6522* type

 522
6522  ; descriptions of peripheral processes
6522
6522  w.c3:   jl.w1 h5.
6524          c36,g2
6528
6528
6528    c17: jl.w1 h5.
6530          c36,g9
6534
6534    c5:  jl.w1 h5.
6536          c36,g10
6540
6540  b.j32w.
6540    j18: c50,g18
6544    j19: c50,g19
6548    j20: c50,g20
6552    j21: c50,g21
6556    j22: c50,g22
 560    j23: c50,g23
6564    j24: c50,g24
6568    j25: c50,g25
6572
6572    c18: jl w1 c31, 17<6
6576    j18-a56
6578    j19-a56
6580    j20-a56
6582    j21-a56
6584    j22-a56
6586    j23-a56
6588    j24-a56
6590    j25-a56
6592    h4, r.16
6624  e.
6624
6624    c12: jl w1 c30
6626    g0:  10, <:reader:>,0,0
6636         0<6,  0,1<22,  k,k-2
6646         c33,  0,r.8
6664
6664    c15: jl w1 c30
6666    g1:  12, <:punch:>,0,0
6676         1<6,  0,1<22,  k,k-2
6686         c33,  0,r.8
6704
6704    c6:  jl w1 c30
6706    g2:  8,  <:console1:>,0
6716         2<6,  0,1<22,  k,k-2
6726         c33,  0,0,24
6734         0,r.10,    37,25, 8
6760
6760    c14: jl w1 c30
6762    g3:  2,  <:clock:>,0,0
6772         3<6,  0,0,     k,k-2
6782         c35
6784
6784    c11: jl w1 c30
 786    g4:  6,  <:drum:>,0,0
6796         4<6,  0,0,     k,k-2
6806         c33,  0,r.3
6814
6814    c13: jl w1 c30
6816    g5:  14, <:printer:>,0
6826         5<6,  0,1<22,  k,k-2
```

always standard.

```
6836          c33,   0,r.8
6854
6854     c16: jl w1 c30
6856     g6:  6,  <:disc:>,0,0
6866          6<6,   0,0,      k,k-2
6876          c33,   0,r.3
6884
6884     c9:  jl w1 c30
 886     g7:  34, <:tapeunit7:>,0
6896          7<6,   0,1<22,  k,k-2
6906          c37,   1,-1,-1
6914          0   , r.6
6926
6926     c10: jl w1 c30
6928     g8:  34, <:tapeunit8:>,0
6938          8<6,   0,1<22,  k,k-2
6948          c37,   1,-1,-1
6956          0   , r.6
6968
6968     c7:  jl w1 c30
6970     g9:  8,  <:console3:>,0
6980          9<6,   0,1<22,  k,k-2
6990          c33,   0,0,128
6998          0,r.10,   37,25, 8
                        45        127
7024
7024     c8:  jl w1 c30
7026     g10: 8,  <:console2:>,0
 036          10<6,   0,1<22,  k,k-2
7046          c33,    0,0,128
7054          0,r.10,    37,25,8
7080
7080     c19: jl w1 c30
7082     g11: 54, <:plotter1:>,0
7092          11<6,   0,1<22,  k,k-2
7102          c33,    510
7106     c20: jl w1 c30
7108     g12: 54, <:plotter2:>,0
7118          12<6,   0,1<22,  k,k-2
7128          c33,    510
7132
7132     g13=h4
7132     g14=h4
7132     g15=h4
7132     g16=h4 ; telecom controller base reg 0
7132     g17=h4 ; telecom controller base reg 1
7132
 132     g18: 58, <:txp1:>,0,0
7142          18<6, 0,1<22,  k,k-2
7152          c33,  0,32,32
7160          0,r.5
7170
7170     g19: 58, <:txp2:>,0,0
7180          19<6, 0,1<22,  k,k-2
7190          c33,  0,32,32
7198          0,r.5
7208
7208     g20: 58, <:tgp3:>,0,0
7218          20<6, 0,1<22,  k,k-2
7228          c33,  0,32,32
7236          0,r.5
7246
7246     g21: 58, <:telex4:>,0,0
7256          21<6, 0,0<22,  k,k-2
7266          c33,  0,32,32
7274          0,r.5
 284
7284     g22: 58,<:tgp201:>,0,0
7294          22<6, 0,1<22,  k,k-2
7304          c33,  0,32,32
7312          0,r.5
7322
7322     g23: 58, <:txp6:>,0,0
```

```
7332          23<6, 0,1<22,  k,k-2
7342          c33,  0,32,32
7350          0,r.5
7360
7360    g24: 58, <:tgp7:>,0,0
7370          24<6, 0,1<22,  k,k-2
7380          c33,  0,32,32
7388          0,r.5
 598
7398    g25: 58, <:tgp101:>,0,0
7408          25<6, 0,1<22,  k,k-2
7418          c33,  0,32,32
7426          0,r.5
7436
7436    c4:  jl w1 c31 , 16<6
7440          g18       ; connector 1
7442          g19       ; connector 2
7444          g20       ; connector 3
7446          g21       ; connector 4
7448          g22       ; connector 5
7450          g23       ; connector 6
7452          g24       ; connector 7
7454          g25       ; connector 8
7456          h4, r.16  ; not used
7488
7488
7488  n.m.
 488                         monitor peripheral process descriptions included
7488
7488  ; area processes:
7488    f7 = k, h7=f7
7488  ; internal processes:
7488    f8 = f7 + a1 * a2, h8=f8
7488  ; message buffers:
7488    f9 = f8 + a3 * a4, f10 = f9 + a5 * a6 - 2
7488
7488  ; monitor entries used globally:
7488    b29 = f8  ; first internal process
7488    ~~b30 = d4~~  ; print w0
7488    ~~b31 = d3~~  ; print w1
7488    ~~b32 = d2~~  ; print w2
7488    ~~b33 = d1~~  ; print w3
7488    ~~b34 = d0~~  ; save w3
7488    b35 = d5  ; remove
7488    b36 = d6  ; link
7488    b37 = d11 ; search name
 488    b38 = d13 ; release buffer
7488    b39 = d15 ; deliver answer
7488
7488  b.i1                    ; begin
7488  w.i0: rl. w2  i1.       ; make room:
7490        jl      x3+0      ;   autoloader(message pool end + 2):
7492    i1: f10+2             ; after loading:
7494        jl.     i0.       ;   goto make room;
7496  e.                      ; end
7496  i.
7496    h25=k - b127 + 2
7496  e.    ; end of process description segment
7496
7496    k = b29 + a3 * a4 + a5 * a6
13960  ; comment: k = absolute top address of monitor.
13960  i.
13960
13960
```

Continue typing here:

```
13960
13960
13960  ; segment 5: initialize monitor
13960  ; this segment initializes monitor table, process descriptions,
13960  ; and buffers within the monitor as follows:
13960  ; monitor table:
13960  ;       initial monitor table as defined below
13960  ; area process descriptions:
13960  ;       description address is placed in name table
13960  ;       description is initialized to zero
13960  ;       kind is set to 4
13960  ; internal process descriptions:
13960  ;       description address is placed in name table
13960  ;       description is initialized to zero
13960  ;       identification bit is set to 1<n
13960  ;       next and last event are set to next event
13960  ; message buffers:
13960  ;       buffer is initialized to zero
13960  ;       buffer is linked to pool
13960  ; after return to the autoloader, the segment is removed.
13960
13960  s.k=k, g30                   ; begin
13960  w.b127=k, g30, k=k-2
13960  w.g1: rs. w3  g8.            ; start:
13962        al. w1  g10.           ; monitor table:
13964        al  w2  8              ;   for i:=0 step 2
13966    g2: rl  w0  x1+0           ;   until no more do
13968        rs  w0  x2+0           ;   word(8+i):=
13970        al  w1  x1+2           ;   word(initial monitor table+i);
13972        al  w2  x2+2           ;
13974        sh  w2  b21            ;
13976        jl.     g2.            ;
13978        al  w0  0              ; name table:
13980        rl  w2  (b5)           ;
13982    g3: rs  w0  x2+0           ;
13984        al  w2  x2+2           ;   for addr:= name table(first area)
13986        sh  w2  (b8+6)         ;   step 2 until message pool end
13988        jl.     g3.            ;   do word(addr):= 0;
13990        rl  w2  b5             ;   entry:= first area;
13992        rl  w3  x2+0           ;   proc:= name table(entry);
13994        al  w0  4              ; area process:
13996    g4: rs  w3  x2+0           ;   name table(entry):= proc;
13998        rs  w0  x3+0           ;   kind(proc):= 4;
14000        al  w2  x2+2           ;   entry:= entry + 2;
14002        al  w3  x3+a2          ;   proc:= proc + area proc size;
14004        se  w2  (b6)           ;   if entry <> first internal
14006        jl.     g4.            ;   then goto area process;
14008        rl. w0  g9.            ;   id bit:= 1 shift 23;
14010        rl  w3  x2+0           ;   proc:= name table(entry);
14012                               ; internal process:
14012    g5: rs  w3  x2+0           ;   name table(entry):= proc;
14014        rs  w0  x3+a14         ;   identification(proc):= id bit;
14016        al  w1  x3+a15         ;   next(event q(proc)):=
14018        rs  w1  x3+a15         ;   last(event q(proc)):=
14020        rs  w1  x3+a15+2       ;   event q(proc);
14022        ls  w0  -1             ;   id bit:= id bit shift (-1);
14024        al  w2  x2+2           ;   entry:= entry + 2;
14026        al  w3  x3+a4          ;   proc:= proc + internal proc size;
14028        se  w2  (b7)           ;   if entry <> name table end
14030        jl.     g5.            ;   then goto internal process;
14032        al  w1  b8             ;
14034        rl  w2  b8+4           ;
14036    g6: jl  w3  b36            ;   for buf:= first buf(mess pool)
14038        wa  w2  b8+8           ;   step buf size(mess pool)
14040        sh  w2  (b8+6)         ;   until last buf(mess pool)
14042        jl.     g6.            ;   do link (mess pool, buf);
14044        al. w2  g1.            ;
14046        jl.     (g8.)          ;   autoloader(start);
14048
```

```
14048  g8:   0
14050  g9:   1<23
14052
14052                   ; initial monitor table:
14052  g10: 0           ; <interrupt number>
14054       c25         ; <system start address>
14056       c27         ; <interrupt response>
14058       c26         ; <start key response>
  060                   ; <interrupt 0-24>
14060  t.
~~14060*  type~~
~~14060~~
~~14060  ; interrupt list~~
~~14060~~
~~14060     c0 ,c1 ,c2 ,c3 ,c4 ,c5 ,c6 ,c7 ,c8 ,c9~~
~~14080     c10,c11,c12,c13,c14,c15,c16,c17,c18,c19~~
~~14100     c20,c24,c24,c24,c51~~
~~14110~~
~~14110  n.m.~~
14110                      monitor interrupt list included
14110        0          ; <current process>
14112        b2         ; <next running process>
14114        b2         ; <last running process>
14116        f0         ; <name table start>
14118        f1         ; <first device in name table>
14120        f2         ; <first area in name table>
  122        f3         ; <first internal in name table>
14124        f4         ; <name table end>
14126        b8         ; <next message buffer>
14128        b8         ; <last message buffer>
14130        f9         ; <message pool start>
14132        f10        ; <message pool end>
14134        a6         ; <message buffer size>
14136        ~~0,r.5     ; <not used>~~        0,r.2   ; <time base>
                                                c4      ; <timer start>
                                                0,r.2   ; <not used>
14146        a85        ; <maximum time slice>
14148        0          ; <time slice>
14150        0          ; <microseconds>
14152        0          ; <time>
14154        0          ;
14156        0          ; <clock value>
14158        3<6        ; <clock device no * 64>
14160        a9         ; <no of storage bytes>
14162                   ; <monitor procedures>
14162        e0 , e1 , e2 , e3 , e4 , e5 , e6 , e7 , e8 , e9
14182        e10, e11, e12, e13, e14, e15, e16, e17, e18, e19
  202        e20, e21, e22, e23, e24, e25, e26, e27, e28, e29
14222        e30, e31, e32, e33, e34, e35, e36, e37
14238        0          ; <current buffer address>
14240        0          ; <current receiver>
14242        0          ; <interrupt return address>
14244        f6         ; <process link in dummy internal process>
14246
14246        jl. g1. ; after loading: goto start;
14248     g30=k-b127 + 2
14248  k=g1
13960  ;comment: k = absolute first address of initialize monitor
13960  i.
13960  e.   ; end of initialize monitor segment
13960
13960  e.   ; end of monitor block with c, d, e,and f names
13960
13960  m.
13960                  monitor text 3 included
13960
13960  m.
13960                  monitor text 4
13960
13960  b50 = a9 - 20        ;
13960  b51 = a9 - 18        ;
13960  b52 = a9 - 16        ;
13960  b53 = a9 - 14        ;
```

```
13960  b54 = a9 - 12         ;
13960  b55 = a9 - 10         ;
13960  b56 = a9 -  8         ;
13960  b57 = a9 -  6         ;
13960  b58 = a9 -  4         ;
13960  b59 = a9 -  2         ;   last core
13960
13960  ; segment 6:  process functions.
  960  ;   leif svalgaard / jørn jensen
13960  ;   catalog administration; creation, removal, and
13960  ;   start and stop of processes.
13960
13960
13960  s.   c10, d24, e31, f100, i40, j21         ; proc func segment start:
13960  w.b127=k, j21, k=k-2
13960
13960  ; use of slang names:
13960  ;   a:   monitor constants, declared and defined before proc func
13960  ;   b:   monitor absolute entry addresses,  -       -      -   -
13960  ;   c:   global full word constants
13960  ;   d:   global variables, start addresses for records
13960  ;   e:   procedures, mostly called with w3 as return register
13960  ;   f:   constant names, relative addresses in records
13960  ;   g:   local labels in procedures and actions
13960  ;   i:   process functions (actions)
13960  ;   j:   global points in central administration, error exits
13960  ;
  960
13960  ; definition of catalog parameters:
13960  f0  =   a88          ; size of one catalog entry:      34  bytes
13960  f9  =  512 - 2       ; catalog buffer size - 2  :     510  bytes
13960  f10 =   f9/f0        ; number of entries per segment:  15
13960
13960  ; the catalog itself and the backing storage configuration is defined
13960  ;   via the bit table, backing device table, and the catalog parameters
13960  ;   all arranged with the following layout:
13960  ;   (                )
13960  ;   (                )     bit table
13960  ;   (                )
13960  ;   ================
13960  ;   ( device no<13 ) 0    first backing device
13960  ;   ( first segment) 2
13960  ;   (   segments   ) 4
13960  ;   ================
13960  ;   (   --------   ) 6    next backing device
13960  ;   (      +++     ) 8
  960  ;   (      +++     )
13960  ;   ================
13960  ;   (     0<13     )     dummy device terminating
13960  ;   (  top segment )     the device table
13960  ;   (      0       )
13960  ;   ================
13960  ; b50:  number of segments in catalog
13960  ; b51:  last working name:  <:wrk000000:>,0
13960  ; b55:  device number for catalog device
13960  ; b56:  start address of device table
13960  ; b57:  start address of bit table
13960  ; b58:  free entries in the catalog
13960  ; b59:  free segments in backing storage
13960  ; the last working name, free entries, and free segments are updated whe
13960  ;   changed by proc func.
13960
13960  ; record sender.
13960  ;   the absolute address of the description of the calling process is st
13960  ;   in d2.  parameters to and from the sender are found in the register (
  960  ;   as follows:
13960  w. b60 = k                         ; first address.proc func:
13960  f20 =  a31                         ; save w3: name address
13960  f21 =  a29                         ; save w1: tail address
13960  f22 =  a29                         ;      or: new name address
13960  f23 =  a29                         ;      or: catalog key
13960  f24 =  a29                         ;      or: general parameter pointer
```

```
13960  f25 =  a28                         ; save w0: result
13960  f26 =  a40                         ; save wait address
13960
13960  ; address pointers to the catalog parameters:
13960
13960  c0:   b50                          ; addr of (number of catalog segments)
13962  c9:   b51                          ; addr of (last working name: <:wrk
13964        b52                          ;         6 octal
  966        b53                          ;         digits
13968        b54                          ;         3 nulls:>
13970
13970  d18:  b56                          ; addr of (start addr of device table)
13972  d9:   b57                          ; addr of (start addr of bit table)
13974
13974  d4:   b58                          ; addr of (free entries)
13976  d5:   b59                          ; addr of (free segments)
13978
13978  ; definition of proc func communications parameters
13978
13978  f14 =   3              ; operation  read
13978  f15 =   5              ; operation write
13978  f16 =  48              ; minimum value of digit in identifier
13978  f17 =  57              ; maximum   -   -   -   -    -
13978  f18 =  97              ; minimum   -   -  letter -   -
13978  f19 = 12~~5~~2             ; maximum   -   -   -   -    -   (do not include danish letters: æøå)
13978  f37 =   0              ; kind:  internal process
13978  f38 =   4              ; kind:  area process
  978
13978  ; definition of bits and values of process states
13978
13978  f40 =  1<2             ; repeat bit,  in proc state
13978  f41 =  1<3             ; no stop bit, in proc state
13978  f42 =  1<4             ; parent bit,  in proc state
13978  f43 =  1<5             ; stopped bit, in proc state
13978  f44 =  1<6             ; out of q bit,in proc state
13978  f45 =  1<7             ; waiting bit, in proc state
13978  ; process state values
13978
13978  f46 = a95              ; running
13978  f47 = a99              ; waiting start by parent
13978  f48 = a97              ; waiting stop by parent
13978  f49 = a100             ; waiting start by ancestor
13978  f50 = a98              ; waiting stop by ancestor
13978
13978  ; note: the above a-names are defined before proc func loading.
13978  ;      running:                      out of q, no stop
  978  ;      waiting start by parent       stopped, parent, no stop
13978  ;      waiting stop by parent        stopped, parent
13978  ;      waiting start by ancestor     stopped, no stop
13978  ;      waiting stop by ancestor      stopped
13978  ;      waiting events                repeat
13978  ;      waiting for proc func         repeat, out of q
13978  ;
13978  ; record work.    This record holds the current catalog entry, and defin
13978  ;   also the general format of an entry as it appears in cat buf:
13978  ;
13978  ;         f1:  <namekey> , f2:  <catkey>
13978  ;         f3:  <creation number>
13978  ;         f4:  <first segment>  only relevant for areas
13978  ;         f5:  <name>           4 words
13978  ;         f6:  <tail> :         f7: <size>
13978  ;              fill up entry to total of f0 words
13978  ; The first word of the tail contains the size (in number of segments) o
13978  ;   area if the entry describes an area, otherwise the <size> is irrel-
13978  ;   evant for proc func.
  978  ;         size  > 0            =>     size of backing area
13978  ;         size =< 0            =>     not backing area
13978  w.                     ;
13978  d1:                    ; record work:
13978  f1  = k-d1             ;   namekey
13978  f2  = f1 + 1  ,  0     ;   catkey
13980  f3  = k - d1  ,  0     ;   creation number
```

```
13982  f4  = k - d1  ,  0      ;   first segment
13984  f5  = k - d1  ,  0 ,r.4;   name(0:6)
13992  f6  = k - d1            ;   tail(0:tailsize-2)
13992  f7  = k - d1  ,  0      ;   size
13994  f8  = f0 - f6           ;   tailsize even bytes
13994  r. f8/2                 ;   fill up tail and entry
14012  d2:   0                 ; absolute address of calling process
14014
  014  d3:   d0                ; cur entry, points to an entry in cat buf
14016  d6:   0                 ; hole start bit
14018        0    ; d6 + 2     ; hole start word
14020
14020  ; description of record cat
14020  d7:  <:catalog:>    ; cat:  name
14026         0  ,  0      ;   name table entry (monitor concept)
14030  ; description of record cat message
14030  d8:                 ; cat message:
14030  f30 = k - d8, f14<12;   cat seg operation, initially read
14032  f32 = k - d8,  d0   ;   first address of cat buf
14034  f34 = k - d8,  d20  ;   last address of cat buf
14036  f36 = k - d8, -1    ;   current cat seg, initially not existing
14038
14038  ; global constants
14038  c1:  d19             ; max entry = absolute address of last word of
14040                       ;   last entry in the cat buffer
14040  c2:  12              ; used by the bit table administration
14042  c3:  24              ;  -    -    -    -    -       -
   44  c4:  3<22 + 3<10     ; used to test for claims > 1023
14046  c5:  a89             ; initial im for created process
14048  c6:  4095            ; used in central adm. (bytes: 0,-1)
14050  c7:  -1<12 + f41     ; used by stop internal process
14052
14052  ; central process function administration:
14052  ; when entered at waiting point this code will
14052  ;   call the monitor function wait proc func message (jd w3 1<11+0);
14052  ;   this has the effects: proc func is removed from the timer queue.
14052  ;   when a process calls a process function, proc func is
14052  ;   reactivated. The central administration fetches via the
14052  ;   first message in proc func s message queue the address
14052  ;   of the sender description. The administration now checks
14052  ;   that the requesting process is allowed to call the process
14052  ;   function, and jumps to the proper proc func action or to
14052  ;   error 1 if not allowed.
14052  ;   the proc func action will after having done its job return
14052  ;   to one of the error exits or to ex ok, which in turn will set
14052  ;   save w0:= result, write cur cat seg back on the drum if it
  052  ;   has been changed, and end up at waiting point.
14052
14052  j7:                      ; error 7:  w0:= 6;  goto set result;
14052  j6:  am       1          ; error 6:  w0:= 6;  goto set result;
14054  j5:  am       1          ; error 5:  w0:= 5;  goto set result;
14056  j4:  am       1          ; error 4:  w0:= 4;  goto set result;
14058  j3:  am       1          ; error 3:  w0:= 3;  goto set result;
14060  j2:  am       1          ; error 2:  w0:= 2;  goto set result;
14062  j1:  am       1          ; error 1:  w0:= 1;  goto set result;
14064  j0:  al  w0   0          ; ex ok:    w0:= 0;
14066       rl. w1   d2.        ; set result:
14068       rs  w0 x1+f25       ;   result.sender:= w0;
14070
14070       bz. w0   d8.+f30    ; return proc result:
14072       sn  w0   f15        ;   if cat operation = write then
14074       jl. w3   e5.        ;   write cat seg;
14076
14076  j10: jd  w3   1<11 + 0   ; waiting point:
14078       rl  w1   b1         ;   wait proc func message;
14080       rl  w1 x1+a15       ;   sender:= next(messq(proc func))-a16;
14082       al  w1 x1-a16       ;
14084  ; the entry <next message> in proc funcs process description
14084  ;   points to <next process> in the sender (a16).
14084       rs. w1   d2.        ;   save sender;
14086       rl  w3 x1+a33       ;   N:= word (saveIC.sender - 2)
14088       rl  w3 x3-2         ;      - <:jd   1<11+40:> ;
```

```
14090        ws. w3   j12.      ;
14092        la. w3  -3         ;   remove last bit of N;
14094        hs. w3   j13.      ;
14096
14096        bz  w0 x1+a22      ; check call:  if function mask  or
14098        lo. w0 x3+j11.     ;   function key <> all ones
14100        so. w0  (c6.)      ;   then  not allowed:
14102        jl.       j1.      ;      goto error 1;
  104
14104        rl  w2 x1+f20      ; ok:  w2:= name address.sender;
14106        rl  w0 x2+0        ;   working name created:= name(0).sender;
14108        rs. w0   d17.      ;
14110        sh  w3   25        ;   if N < 26 then
14112        ~~jl. w3   e1.      ;   set work name;~~   sn  w3   36
                                                       jl. w3   e1.
14114   j13=k+1, al w3 ;  N     ;
14116        bz. w3 x3+j11.     ;   action:= N;
14118        jl.    x3+j11.     ;   goto process function (action);
14120
14120   j12: jd     1<11 + 40   ;   <:jd   1<11 + 40:>
14122
14122   ; after switch to action:
14122   ;    w0 = namekey.work
14122   ;    w1 = sender
14122   ;    w2 = name address
14122   ;    name key.work defined
14122
14122   ; table of actions and allowed bits used by central administration:
  122
14122   h.                                ; halfword mode: action,check bits
14122   j11: i20         , 4095 - 1<11    ; create entry
14124        i21         , 4095           ; look up entry
14126        i22         , 4095 - 1<10    ; change entry
14128        i23         , 4095 - 1<9     ; rename entry
14130        i24         , 4095 - 1<10    ; remove entry
14132        i25         , 4095 - 1<8     ; permanent entry
14134        i26         , 4095           ; create area process
14136        i27         , 4095           ; create peripheral process
14138        i28         , 4095           ; create internal process
14140        i29         , 4095           ; start internal process
14142        i30         , 4095           ; stop internal process
14144        i31         , 4095           ; modify internal process
14146     74 i32         , 4095           ; remove process
14148     26 ~~[illegible]~~ i33  , 0000   ; monitor log
14150     28 i34         , 4095 - 1<5     ; generate name
14152     70 i35         , 4095           ; copy
14154   w. 72 0            0              ; ~~not used~~
  . 154   74 0
          76 i36         , 4095           ; ~~not~~ lookup ~~[illegible]~~ head and tail) .
14154   ; if action is remove process then the check is postponed until
14154   ;   the action is called, because the function mask only controls
14154   ;   removal of peripheral processes and not internal and area processes.
14154   ; the same holds for create peripheral process because, at MI,
14154   ;   creation of peripheral processes for magnetic tape stations
14154   ;   does not require function bit 4 as it does for other devices.
14154
14154   ; procedure set work name (name address) result:(namekey);
14154   ;   copies the name, consisting of 4 words, at name address.sender
14154   ;   to name.work, and sets namekey.work:= name key function(name.work);
14154   ; call:       w2 = name address
14154   ;             jl. w3   e1.
14154   ; return:     w0 = namekey.work
14154   ;             w1,w2,w3 unchanged
14154   b.    g0                 ; begin
14154   w.                       ; set work name:
14154   e1:  rs. w1    g0.       ;   save w1;
14156        dl  w1 x2+2         ;
  158        ds. w1    d1.+f5+2  ;
14160        dl  w1 x2+6         ;   move 4 words
14162        ds. w1    d1.+f5+6  ;   from sender to work;
14164        aa. w1    d1.+f5+2  ; compute namekey function:
14166        wa  w1     0        ;   w0w1:=
14168        ba  w1     2        ;      long add (name(4).work,name(6).work)
14170        al  w0     0        ;           and:(name(2).work,name(0).work);
```

```
14172        wd. w1  (c0.)      ;    w1:= w0 + w1; w1:= w1 + first byte(w1);
14174        hs. w0   d1.+f1    ;    w0:= namekey.work:= w1 mod cat segments;
14176        rl. w1   g0.       ;    restore w1;
14178        jl      x3         ; return:
14180   g0:  0                  ; save for w1:
14182   e.                      ; end set work name;
14182
14182   ; The namekey is a number ranging from 0 to the number of segments
  182   ; in the catalog. The namekey is computed from the name following the
14182   ; above algorithm and is used to speed up the search for the name in
14182   ; the catalog. When an entry is created it is placed in the first free
14182   ; entry in the catalog on the segment <namekey> or the following seg-
14182   ; ments. The search for an entry then starts from the segment <namekey>
14182   ; and towards higher segment numbers.
14182
14182   ; procedure get key seg ( result: cat buf start);
14182   ;    ensures that the current segment in the catalog buffer is the one gi
14182   ;    by namekey.work; if not then it is read, mayby after a write operati
14182   ;    if cur segment has been changed.
14182   ; call:      jl. w3   e4.
14182   ; return:    as from get next seg, see below
14182   ; subentry: get next seg, write cat
14182
14182   ; procedure write cat; subentry  to get key seg;
14182   ;    if cat operation = write then
14182   ;    cur seg in the catalog buffer is output;
14182   ;    cat seg operation:= read;
  182   ; call:      jl. w3   e5.
14182   ; return:    as from get next seg, see below
14182
14182   b. g5                          ; begin
14182   w.                             ;
14182        0 ; save w0               ;
14184   g0:  0 ; save w1               ;
14186   e5:  rl. w2   d8.+f36          ; write cat: w2:= cur seg:
14188        jl.      g2.              ;    goto get cat seg;
14190   e4:  bz. w2   d1.+f1           ; get key seg:
14192        se. w2  (d8.+f36)         ;    if namekey.work <> cur seg
14194        jl.      g2.              ;    then goto get cat seg;
14196        rl. w2   d8.+f32          ;    w2:= first buf.cat message;
14198        jl      x3                ; return;
14200
14200   ; return:  w2 = cat buf start = absolute address of catalog buffer
14200   ;            w3   changed
14200   ;            cur cat seg defined
14200
  200   ; procedure get next seg ( result: cat buf start ):
14200   ;   inputs the cyclically next segment of the catalog to the catalog
14200   ;   buffer and sets cur cat seg accordingly;
14200   ;   if cat seg operation = write then the current segment in the
14200   ;   catalog buffer is first output;
14200   ; call:       jl.  w3   e6.
14200   ; return:     w2 = cat buf start = absolute address of catalog buffer start
14200   ;             w3 changed, cur cat seg defined
14200
14200   e6:  rl. w2   d8.+f36          ; get next seg:
14202        al  w2 x2+1               ;    w2:= cur cat seg + 1;
14204        am.      (c0.)            ;    if w2 = cat segments then
14206        sn  w2   (0)              ;    w2 := 0;
14208        al  w2    0               ;
14210
14210   g2:  ds. w1   g0.              ; get cat seg:  save(w0,w1);
14212        ds. w3   d12.             ;    save(w2,return);
14214        bz. w0   d8.+f30          ;
14216        se  w0   f15              ;    if cat seg operation = write
  218        jl.      g4.              ;    then begin
14220
14220   g5:  al. w3   d7.              ; try to reserve:
14222        jd       1<11 + 8         ;    reserve process (<:catalog:>,result);
14224   g3:  al. w1   d8.              ; cat transport:
14226        al. w3   d7.              ;    send message
14228        jd       1<11 + 16        ;    (<:catalog:>,cat message);
```

```
14230        bz. w0   d8.+f30          ; comment: no test for buf claim exceeded;
14232        se  w0   f14              ;   if cat seg operation = write
14234        jd       1<11 + 10        ;   then release process (<:catalog:>);
14236
14236        al. w1   d16.             ; comment: w2 unchanged from send message;
14238        jd       1<11 + 18        ;   wait answer(cat answer buffer):
14240        al  w3   f14              ;   cat seg operation:= read;
14242        hs. w3   d8.+f30          ;
  244        sn  w0    1               ;   if answer <> 1 or status.answer <> 0
14246        sh  w0  (x1)              ;   then goto error 2; note the test: sh:
14248        jl.       j2.             ;   end cat transport;
14250
14250   g4:  dl. w3   d12.             ;   unsave(w2,return);
14252        rx. w2   d8.+f36          ;   if w2<>cur cat seg then begin
14254        se. w2  (d8.+f36)         ;   cur cat seg:= w2;
14256        jl.       g3.             ;      goto cat transport end;
14258
14258        rl. w2   d8.+f32          ;   w2:= first buf.cat message:
14260        dl. w1   g0.              ;   unsave(w0,w1);
14262        jl        x3              ; return:
14264
14264   e.                             ; end  cat segment transfer;
14264
14264   ; procedure test format;
14264   ;   tests whether the format of the 4 words at name.work corresponds to
14264   ;   an identifier.
14264   ; call:     jl. w3   e3.
  264   ; return:  not identifier
14264   ; return+2:     ok
14264   ;             w0, w1, w2, w3 changed
14264
14264   b.  g5                        ; begin
14264   w.                            ; test format:
14264   e3:  rs. w3   d12.            ;   save return;
14266        al. w2   d1.+f5          ;   name pointer:= addr(name.work);
14268        al  w1   -11             ;   count:= -11;
14270
14270   g1:  rl  w0 x2                ; next word: cur word:= word(name pointer);
14272   g2:  al  w1 x1+11             ; next char: count:= count + 11:
14274        al  w3    0              ;   cur char:= next char from(cur word);
14276        ld  w0    8              ;
14278        sn  w1    0              ;   if count <> 0 then
14280        jl.       g3.            ;      begin
14282        sn  w3    0              ;      if cur char = 0 then goto finis;
14284        jl.       g4.            ;      if cur char =< max digit then
14286        sh  w3   f17             ;      cur char:= cur char-min digit+min lett
  288        al  w3 x3+f18-f16        ;      end;
14290   g3:  sl  w3   f18             ;   if cur char < min letter or
14292        sl  w3   f19+1           ;      cur char > max letter then return;
14294        jl.      (d12.)          ;
14296        so  w1   1<4             ;   if bit(19,count) = 0 then
14298        jl.       g2.            ;      goto next char;
14300        sn  w1   121             ;   if count = 121 then
14302        jl.      (d12.)          ;      return;
14304
14304        al  w2 x2+2              ;   namepointer:= name pointer + 2;
14306        jl.       g1.            ;   goto next word;
14308
14308   g4:  rl. w3   d12.            ; finis:  restore w3;
14310   g5:  se  w0    0              ; rep finis:
14312        jl     x3                ;   if rest of cur word <> 0 then return;
14314        sl  w1   99              ;   if count >= 99 then
14316        jl       x3+2            ;   return2;
14318        al  w1 x1+33             ;   count:= count + 33;
14320        al  w2 x2+2              ;   name pointer:= name pointer + 2;
  322        rl  w0 x2                ;   cur word:= word(namepointer);
14324        jl.       g5.            ;   goto rep finis;
14326   e.                            ; end test format;
14326
14326   ; a legal name is a small letter followed by not more than 10
14326   ;   small letters or digits and terminated by a NULL character,
14326   ;   and with NULL fill in the rest of the 4 name words.
```

```
14326
14326  ; procedure search (result: cur entry);
14326  ;   searches in the catalog for an entry with the name given by name.
14326  ;   work; the search starts at segment namekey.work of the catalog,
14326  ;   which must be the correct key for name.work;
14326  ;   if the entry is found then the procedure returns with the cur
14326  ;   entry holding the absolute address in the catalog buffer of the
14326  ;   the entry.
  326  ; call:       jl. w3   e2.
14326  ; return:     entry not found, name is not identifier
14326  ; return+2:   entry not found, name ok
14326  ; return+4;   entry found
14326  ;             w2 = cur entry = abs addr of entry in cat buf
14326  ;             cur entry defined
14326  ;             w0, w1, w3 changed
14326  ; uses:       get key seg, get next seg, test format
14326  ;   The last word in cat buf contains the number of entries on the curre
14326  ;   catalog segment (with the namekey given by the segment number).
14326
14326  b.  g8                        ; begin
14326  w.                            ;
14326
14326  g0:  0  ; return              ;
14328       0  ; save key            ;
14330
14330  e2:  bz. w0   d1.+f1          ; search:  savekey:= namekey.work;
14332       ds. w0   g0.+2           ;   save return;
  334       jl. w3   e4.             ;   get key seg (cat buf);
14336       rl  w1 x2+f9             ;   entry count:= key entries.cat buf;
14338       al  w2 x2-f0             ;   entry:= cat buf - entry size;
14340
14340  g1:  rl. w0   d1.+f5          ; next1:  w0:= name (0).work;
14342  g2:  sn  w1     0             ; next2:  if entry count = 0 then
14344       jl.        g4.           ;         goto not found;
14346
14346  g3:  al  w2 x2+f0             ; next3:  entry:= entry + entry size;
14348       sl. w2  (c1.)            ;   if entry >= max entry then
14350       jl.        g5.           ;   get and test next seg;
14352  g6:  bz  w3 x2+f1             ;   if namekey.entry <> save key then
14354       se. w3  (g0.+2)          ;   goto next3;
14356       jl.        g3.           ;
14358       al  w1  x1-1             ; count key:  entry count:= entry count - 1;
14360       se  w0  (x2+f5)          ; test name agreement:
14362       jl.        g2.           ;   if name(0).work <> name(0).entry
14364       dl. w0   d1.+f5+4        ;   then goto next2;
14366       sn  w3  (x2+f5+2)        ;   if name(2).work <> name(2).entry
  368       se  w0  (x2+f5+4)        ;   or name(4).work <> name(4).entry
14370       jl.        g1.           ;   or name(6).work <> name(6).entry
14372       rl. w0   d1.+f5+6        ;   then goto next1;
14374       se  w0  (x2+f5+6)        ;
14376       jl.        g1.           ;
14378
14378       rs. w2   d3.             ; entry found:
14380       rl. w3   g0.             ;   cur entry:= entry;
14382       jl      x3+4             ;   return+4;
14384
14384  ; procedure get and test next seg;
14384  w.                            ; begin
14384  g5:  jl. w3   e6.             ;   get next seg (entry);
14386       rl. w3   d8.+f36         ;   if cur cat seg = save key
14388       se. w3  (g0.+2)          ;   then goto not found;
14390       jl.        g6.           ; end;
14392
14392  g4:  rl. w3   g0.             ; not found:
14394       jl.        e3.           ;   goto test format;
  396
14396  ; search returns via test format if the entry is not found.
14396  e.                            ; end search;
14396
14396  ; procedure set cur entry (result: cur entry, work);
14396  ;   moves the entry in work to the entry given by current entry;
14396  ; call:      jl. w3   e7.
```

```
14396  ; return:   all registers changed
14396
14396  ; procedure move (from address, to address, number of even bytes);
14396  ;    moves a number of words from one place in core to another;
14396  ; call:     w2 = from address
14396  ;           w1 = to address
14396  ;           w0 = n = number of words in even bytes.
14396  ;           jl. w3   e8.
  396  ; return:   all registers changed
14396
14396  b.  g2                          ; begin
14396  w.                              ;
14396  e7:  al   w0   f15              ; set cur entry:
14398       hs. w0   d8.+f30           ;   comment: entry is changed;
14400       al. w2   d1.               ;   cat seg operation:= write;
14402       rl. w1   d3.               ;   move (work, cur entry, entry size);
14404       al   w0   f0               ;   return;
14406
14406  e8:  rs. w3   d12.              ; move:  save return;
14408  g1:  sh   w0    0               ;   for n:= number of even bytes
14410       jl.       (d12.)           ;   step -2 until 0 do begin
14412       rl   w3 x2                 ;   word(to addr):= word(from addr);
14414       rs   w3 x1                 ;   to addr:= to addr + 2;
14416       al   w1 x1+2               ;   from addr:= from addr + 2;
14418       al   w2 x2+2               ; end move;
14420  g2= k-1                         ;
14420       bs. w0   g2.               ;
  422       jl.       g1.              ;
14424  e.                              ; end;
14424
14424  ; procedure find first hole;
14424  ; initialize find holes to search from first device;
14424
14424  ; procedure find holes;
14424  ;   initialize next hole to search from the start of the bit table
14424  ;   and to search for a hole, and finally to return to after the call
14424  ;   of find holes. It returns via next hole with the description of
14424  ;   the first hole encountered in the bit table.
14424  ; call:      jl. w3   e9.
14424  ; return:    as from next hole
14424
14424  ; procedure next hole (result: hole size);
14424  ;   finds the next hole in the bittable; the procedure must be initializ
14424  ;   via a call of find holes and will always return to after the call of
14424  ;   find holes with hole size:= size of first hole found and the global
14424  ;   variables hole start word and hole start bit defined.
  424  ; call:      jl.      e10. ; note:  no link.
14424  ;            w0, w2, and w3 must not be changed between successive calls
14424  ; return:    w1 = hole size = size (in number of segments) of next hole;
14424  ;            the hole corresponds to a free area starting at
14424  ;            first segment:= (hole start word - start of bit table) * 12
14424  ;                                     - hole start bit;
14424
14424  b.  g13              -1            ; begin
14424  w.                                 ;
14424
14424  g0:  0   ; return
14426  g8:  0   ; last bit word addr: init by find holes.
~~[illegible]  g9:  0   ; [illegible] mask~~ all disks selected
14430  ~~g10: 0   ; [illegible]~~
14432  ~~e9:  am.       (d2.)~~           ; find first hole:
14434       ~~rl  w0    a43~~
14436       ~~rs. w0    g9.~~          ;   selection mask:= selection mask.sender;
14438  ~~e9:~~ rs. w3   g0.          Ok.  ;   save return;
  440       ~~al  w0    -6~~
 442        ~~rs. w3    g10.~~         ;   device:= -6;
14444  e9:                             ; find holes:
14444  ~~g12: al  w3    g10~~          ; next device:
14446       ~~al  w3    x3+6~~
14448       ~~rs. w3    g10.~~         ;   device:= device+6;
14450       ~~wa. w3    (d18.)~~       ;   device addr:= start device table + device
14452       ~~bz  w0    x3+0~~
```

```
       ~~al  w1   -1~~            ;   hole size:= -1;
       ~~sn  w0    0~~                        leave as is
       ~~jl.     (g0.)~~
       ~~rl. w0   g9.~~           ;   if selection mask=0
       ~~sn  w0    0~~            ;   or device no <= 0 then
       ~~jl.     (g0.)~~          ;   return;
       ~~ls  w0    1~~            ;   old mask:= selection mask;
       ~~rx. w0   g9.~~           ;   selection mask:= selection mask shift 1;
       ~~sl  w0    0~~            ;   if old mask(0)=0 then
       ~~jl.     g12.~~ e9.       ;   goto next device;
       ~~al  w0    0~~
       rl  w1 x3+8   rl w3 (d18.)
       wd. w1   c2.
       wa. w1  (d9.)
       al  w1 x1-2               ;   last bit word addr:=
       rs. w1   g8.              ;   first segment(next device)/12-2;
       al  w0    0
       rl  w1 x3+2
       wd. w1   c2.
       wa. w1  (d9.)             ;   bit word addr:=
       al  w3 x1-2               ;   first segment(device)/12-2;
       al  w1   -1                      x3 (bwa)

g1:    sl. w3  (g8.)             ; next word skip used:
       jl.     (g0.)             ;   if bwa >= start device table then  return;
       al  w3 x3+2               ;   bit word addr:= bit word addr + 2;
       al  w2    0               ;   bit shift:= 0;
       rl  w0 x3                 ;   bit word:= bit table (bit word addr);
e10:                             ; next hole;
~~g2:~~    al  w1    0               ; skip used:  hole size:= 0;
       sn  w0    0               ;   if bit word = 0 then
       jl.      g1.              ;   goto next word skip used;
       sh  w0    0               ;   if bit (0,bit word) = 1 then
       jl.      g4.              ;   maybe:  goto start hole;
       ns. w0   g3.              ;
g3=k + 1,am      0               ;   bit shift:=
       al  w2 x2-1               ;       bit shift + normalize (bit word) -1;
       ls  w0    1               ;   bit word:= bit word shift 1;

g4:    ds. w3   d6.+2            ; start hole: hole start word:= bit word addr;
                                 ;   hole start bit:= bit shift;
g5:    al  w1 x1+24              ; count hole size:
       sn  w0   -1               ;   hole size:= hole size + 24;
       jl.      g7.              ;   if bit word <> all ones then
       ns. w0   g6.              ;   begin
g6=k+1, am       0               ;   n:= normalize (bit word);
       al  w2 x2-1               ;   bit shift:= bit shift + n -1;
       ls  w0    1               ;   bit word:= bit word shift 1;
       bs. w1   g6.              ;   hole size:= hole size - n -23;
       al  w1 x1-23              ;   if bit shift > -24 then return;
       sh  w2  -24               ;   end;

g7:    sl. w3  (g8.)             ; next word count hole:
       jl.     (g0.)             ;   if bwa >= start device table then return;
       al  w3 x3+2               ;   bit word addr := bit word addr + 2;
       al  w2    0               ;   bit shift:= 0;
       rl  w0 x3                 ;   bit word:= bit table(bit word addr);
       sl  w0    0               ;   if bit (0,bit word) = 0 then
       jl.     (g0.)             ;   return;
       jl.      g5.              ;   goto count hole size;
e.                               ; end find holes and next hole;

; procedure change bits (size,first seg);
;    changes size bits in the bit table, starting with the bit correspond
;    to first segment, from 0 to 1 or vice versa;
; call:       w2 = size
;             w1 = first segment
;             jl. w3   e11.
; return:     via change them  with hole start word and hole start bit def


; procedure change them (neg size);
```

```
s~~a~~ w1   22     ;   if cat key.cur entry
jl.         g5.    ;   =22 then begin
bz  w0      x3+a22 ;   if function mask's
so  w0      1      ;   last bit =0 then
                   ;
jl.         j4.    ;   goto error4 end;
```

e:

~~bz w0 x2+a22~~

rl. w3

s. w. rl. w3 , rl w0 x3+30) bz w0 ~~x~~ x3+29

rl. w3 d2.
rl w0 x3+a23

14564

```
14564  ;   changes -neg size bits in the bit table starting with the bit descri
14564  ;   by hole start word and hole start bit.
14564  ; call:      w1 = neg size
14564  ;            jl. w3   e12.
14564  ; return:    bit table, all registers changed
14564
14564  b.  g2                          ; begin
14564  w.                              ;
  564
14564  e11: al  w0    0                ; change bits:
14566       wd. w1   c3.               ;
14568       ac  w0  (0)                ;   hole start word:= (first seg//24) * 2;
14570       ls  w1    1                ;   hole start bit:= -(first seg mod 24);
14572       wa. w1  (d9.)              ;   hole start word:= hole start word+
14574       ds. w1   d6.+2             ;                     start addr of bit tabl
14576       ac  w1 x2                  ;   neg size:= - size;
14578
14578  ; change them:
14578
14578  e12: rs. w3   d12.              ; change them:  save return;
14580       dl. w3   d6.+2             ;    bit word addr:= hole start word;
14582                                  ;    x:= hole start bit;
14582
14582  g1:  hs. w2   g2.               ; loop:  mask shift := x;
14584       ac  w2 x2+24               ;   y:= -(x+24);
14586       sl  w1 x2                  ;   if neg size >= y then
14588       al  w2 x1                  ;   y:= neg size;
  590       ws  w1    4                ;   neg size:= neg size - y;
14592       al  w0   -1                ;   mask:= all ones shift (24 + y);
14594       ls  w0 x2+24               ; comment:  0  or left shift;
14596  g2=k+1,  ls w0 0                ;   mask:= mask shift maskshift;
14598       lx  w0 x3                  ; comment: first time maybe right shift late
14600       rs  w0 x3                  ;   bit table (bit word addr):=
14602       sn  w1    0                ;   bit table (bit word addr) exor mask;
14604       jl.     (d12.)             ;   if neg size = 0 then return;
14606       al  w2    0                ;   x:= 0;
14608       al  w3 x3+2                ;   bit word addr:= bit word addr + 2;
14610       jl.      g1.               ;   goto loop;
14612
14612  e.                              ; end changes of bit table;
14612
14612  ; procedure set new entry (result: cur entry, work);
14612  ;   reserves a new entry in the catalog with the name given by
14612  ;   namekey.work and moves the whole entry to the reserved entry.
14612  ; call:       jl. w3   e13.
14612  ; return:     free entries decreased and key entries(namekey.work)
  612  ;             increased by 1; cur entry defined and work moved to
14612  ;             cur entry.
14612  ;             w1 = cur entry; w2 = work; w0, w3 changed.
14612
14612  b.  g2                            ; begin
14612  w.                                ;
14612  g0:  0      ; return              ;
14614
14614  e13: rs. w3   g0.                 ; set new entry:  save return;
14616       jl. w3   e4.                 ;   get key seg;
14618       al  w0    1                  ;   key entries.cat buf :=
14620       wa  w0 x2+f9                 ;   key entries.cat buf + 1;
14622       rs  w0 x2+f9                 ;
14624       al  w1   f15                 ;
14626       hs. w1   d8.+f30             ;
14628       al  w0   -1                  ;   cat seg operation:= write;
14630       jl.      g2.                 ;   goto start loop;
14632
14632  g1:  al  w2 x2+f0                 ; loop:
  634       sl. w2  (c1.)                ;   entry:= entry + entry size;
14636       jl. w3   e6.                 ;   if entry >= max entry then get next se
14638  g2:  se  w0  (x2)                 ; start loop: if word(0).entry <> -1 then
14640       jl.      g1.                 ;   goto loop;
14642       wa. w0  (d4.)                ; free entry found:
14644       rs. w0  (d4.)                ;   free entries:= free entries - 1;
14646       rs. w2   d3.                 ;   cur entry:= entry;
```

```
14648       rl. w3    g0.             ;   restore return;
14650       jl.       e7.             ;   goto set cur entry;
14652
14652  e.                             ; end set new entry;
14652
14652  ; procedure remove cur entry;
14652  ;   removes the entry given by cur entry and namekey.work, which must
14652  ;   agree, by setting:  word(cur entry):= -1;
  652  ; call:       jl. w3   e14.
14652  ; return:     key entries on key segment is reduced and free entries
14652  ;             increased by one.
14652  ;             all registers used.
14652
14652  b.  g0                         ; begin
14652  w.                             ;
14652  g0:  0  ; return               ;
14654
14654  e14: rs. w3   g0.               ; remove cur entry: save return;
14656       al  w1   -1               ;
14658       rs. w1   (d3.)            ;   word (cur entry):= -1;
14660       al  w0    f15             ;
14662       hs. w0   d8.+f30          ;   cat seg operation:= write;
14664       jl. w3   e4.              ;   get key seg;
14666       hs. w0   d8.+f30          ;   cat seg operation:= write;
14668       wa  w1 x2+f9              ;   key entries.cat buf:=
14670       rs  w1 x2+f9              ;   key entries.cat buf - 1;
14672       al  w0    1               ;
  674       wa. w0  (d4.)             ;   free entries:=
14676       rs. w0  (d4.)             ;   free entries + 1;
14678       jl.     (g0.)             ;   return;
14680  e.                             ; end remove cur entry;
14680
14680  ; procedure test entry;
14680  ;   searches name.work; tests whether the found entry is locked for
14680  ;   the sending process; if an area entry is found then it looks for
14680  ;   an area process, and returns disabled if found
14680  ; call:      jl. w3   e15.
14680  ; return:    ok, not area entry, w3<>0
14680  ; return+2:  ok, area entry, area process, w3=0
14680  ; return+4:  ok, area entry, no area process, w3<>0
14680  ;                  w1 = proc addr
14680  ;                  cur entry, proc addr defined, all registers used
14680
14680  b.  (g4)  g24                  ; begin
14680  w.                             ; test entry:
14680  g0:  0 ; return                ;
  682
14682  e15: rs. w3    g0.             ;   save return;
14684       jl. w3    e2.             ;   search (name.work);
14686       jl.       j6.             ;   if not found then
14688       jl.       j3.             ;   goto error 3;
14690
14690       am.      (d2.)            ;   if bit (cat key.cur entry) of:
14692       rl  w0    a23             ;   (cat mask.sender) = 0
14694       bz  w1 x2 + f2            ;   then goto error 4;
14696       ls  w0 x1                 ;
14698       sl  w0     0              ;
14700       jl.       j4.             ;
14702
14702  95:  rl  w0  x2+f3             ;   if creation number.cur entry = 0
14704       sn  w0     0              ;   then goto test area;
14706       jl.       g1.             ;
14708       rl  w3    b6              ;   for intproc:= first internal
14710  g2:  rl  w1    x3              ;   step intlength  do
14712       sh  w1 (x1+a11)           ;   if intproc is existing then begin
  714       se  w0 (x1+a41)           ;      if creation number.cur entry =
14716       jl.       g4.             ;         creation number.intproc  and
14718       se. w1   (d2.)            ;         intproc <> sender
14720       jl.       j4.             ;         then goto error 4;
14722  g4:  al  w3   x3+2             ;   end;
14724       se  w3   (b7)             ;
14726       jl.       g2.             ;
```

see opposite page

test selection mask

```
14728  g1:  rl   w0   x2+f7          ; test area: if size.cur entry =< 0
14730        sh   w0     0           ;   then return;
14732        jl.      (g0.)          ;
14734
14734        al   w2  x2+f5          ; disable:
14736        jd   w3    b37          ;   search name (name.cur entry,
14738        rl   w1  x3             ;                 proc addr);
14740
  740        rl   w0  x1+a10         ;   if not found  or
14742        rl.  w2    g0.          ;   kind.proc <> backing area
14744        rs.  w1    d14.         ;
14746        sn   w0    f38          ;   then enabled return4;
14748        sn   w3   (b7)          ;
14750        je        x2+4          ;
14752        rl.  w3    d2.          ;   if area process is reserved
14754        rl   w0   x3+a14        ;   by another process
14756        so   w0  (x1+a52)       ;   then goto
14758        je.        j5.          ;   enabled error 5;
14760        al   w3    0            ;   w3:=0;
14762        jd         x2+2         ;   disabled return2;
14764
14764  e.                            ; end test entry;
14764  ; procedure clear area proc (proc addr);
14764  ;    the process is not reserved by another process, so that removal
14764  ;    may take place.  the area process is thus removed and the area
14764  ;    claim of all users is increased by one
14764  ; call:          w1=area proc descr addr.
  764  ;                    jd. w3   e16.
14764  ; return:        all registers changed.
14764  ; note the procedure is called in disabled mode and returns
14764  ;                in enabled mode.
14764
14764  b.   g24                      ; begin
14764  w.                            ; clear area process:
14764
14764  e16: rs.  w3   d12.           ;   name(0).proc:= 0;
14766       rl   w2  x1+a53          ;   save users;
14768       ld   w0   -65            ;   users:= 0
14770       ds   w0  x1+a53          ;   reservers:= 0;
14772       rs   w0  x1+a11          ;   for all internal procs
14774       rl   w3   b6             ;
14776  g2:  rl   w1  x3+0            ;   do
14778       bz   w0  x1+a20          ;   if user of area proc
14780       ba.  w0   1              ;   then
14782       sz   w2 (x1+a14)         ;   area claim.int proc:=
14784       hs   w0  x1+a20          ;   area claim.int proc +1;
  786       al   w3  x3+2            ;
14788       se   w3  (b7)            ;
14790       jl.       g2.            ;   enabled return;
14792       je.      (d12.)          ;
14794  e.                            ; end clear area proc;
14794
14794  ; some global variables used by the following procedures
14794
14794  d10: 0, r.4          ; save name (0:6), save proc
14802  d13: 0;  d13+0       ; children bits
14804  d14: 0;  d13+2       ; addr of proc description
14806  d15: 0;  d13+4       ; end chain
14808  d12: 0;              ; common return address (lowest level)
14810  d11=d12-2            ; size differens, save cat seg
14810
14810
14810
14810  ; procedure first proc (proc addr,new state);
14810  ;   finds the process given by name.work and checks that it is a child
  810  ;   of the sender.
14810  ;   initializes  end chain  and  children bits  and returns disabled
14810  ;   with w3 = proc addr  and new state = wait stop by parent.
14810  ; call:                jl. w3   e17.
14810  ; return:              disabled with
14810  ;                      w2 = new state
14810  ;                      w3 = proc addr
```

see opposite page

```
14810  ;                         w0,w1 changed
14810  ; error:                  not child: error 3;
14810
14810  e17: rs. w3   d12.           ; first proc: save return;
14812       al. w2   d1.+f5         ;   disable;
14814       jd  w3   b37            ;   search name (name.work,proc addr);
14816       sn  w3  (b7)            ;   if not found
14818       je.      e26.           ;   or
  820       rl  w3  x3              ;   kind proc addr <> internal process
14822       rl  w0  x3+a10          ;   then
14824       se  w0   f37            ;   enabled goto test found;
14826       je.      e26.           ;
14828
14828       rl. w0   d2.            ;   if parent.proc addr <> sender
14830       se  w0 (x3+a34)         ;   then enabled goto error 3;
14832       je.      j3.            ;
14834
14834       al  w2   0              ;   end chain:= children bits:= 0;
14836       rs. w2   d15.           ;   w3:= proc addr;
14838       ds. w3   d14.           ;
14840       al. w2   f48            ;   w2:= new state:= wait stop by parent;
14842       jd.     (d12.)          ; disabled return;
14844
14844  e26: je. w3   e3.            ; test found: test format:
14846       jl.      j6.            ;   goto if illegal name then error 6
14848       jl.      j3.            ;   else error 3;
14850
  850  ; procedure chain and add children:
14850  ;   connects proc addr to the the chain through wait addresses which
14850  ;   ends in end chain and exits via add children
14850  ; call:         jl. w3   e18.
14850  ; return:       all registers changed
14850
14850  b.   g0                               ; begin
14850  w.                                    ;
14850  e18: dl. w2   d13.+4                  ; chain and add children:
14852       rs  w2  x1+f26                   ;   wait addr.proc addr:= end chain;
14854       rs. w1   d13.+4                  ;   end chain:= proc addr;
14856
14856  ; procedure add children;
14856  ;   searches through all internal processes and adds to children bits
14856  ;   the identification bit for all processes with parent = proc addr;
14856  ; call:         jl. w3   e19.
14856  ; return:       all registers changed
14856
14856  e19: rs. w3   d12.                    ; add children:  save return;
  858       dl. w1   d13.+2                  ;
14860       rl  w3   b6                      ;
14862
14862  g0:  rl  w2  x3                       ;   for w3:= first internal in name tab
14864       sn  w1 (x2+a34)                  ;   step 2 until last proc do
14866       lo  w0  x2+a14                   ;   if parent.name table(w3) =
14868       al  w3  x3+2                     ;   proc addr then
14870       se  w3  (b7)                     ;   children bits:= children bits
14872       jl.      g0.                     ;   or  ident bit.name table(w3);
14874       rs. w0   d13.                    ;
14876       jl.     (d12.)                   ;   return;
14878
14878  e.                                    ; end chain/add children;
14878
14878  ; procedure next proc (result: proc addr, new state);
14878  ;   finds proc addr corresponding to one of the bits in children bits,
14878  ;   removes the corresponding bit in children bits, and returns disabled
14878  ;   with new state = wait stop by ancestor and proc addr defined.
14878  ; call:            jl. w3   e20.
  878  ; return:          w2 = new state
14878  ;                  w3 = proc addr
14878  ;                  w0,w1 changed.
14878  ; return 2:        no more children
14878
14878  b.   g0                            ; begin
14878  w.                                 ; next proc:
```

```
14878
14878  e20: rl. w1   d13.              ;
14880       sn  w1     0               ;   if children bits = 0 then
14882       jd      x3+2               ;  .return 2;
14884
14884       rs. w3   d12.              ; more children:  save return:
14886       rl  w3   b6                ;   w3:= first internal proc in name tb:
14888
  888  g0:  rl  w2  x3                 ; loop:  w2:= name table(w3);
14890       al  w3  x3+2               ;   w3:= w3+2:
14892       so  w1 (x2+a14)            ;   if children bits and ident bit.2 = (
14894       jl.       g0.              ;   then goto loop;
14896
14896       ws  w1  x2+a14             ;   proc addr:= w2;
14898       ds. w2   d13.+2            ;   children bits:=
14900       al  w3  x2                 ;   children bits - ident bit.w2;
14902       al  w2   f50               ;   new state:= wait stop by ancestor:
14904       jd.     (d12.)             ;
14906
14906  e.                              ; end next proc;
14906
14906  ; procedure set pk (proc, pk value);
14906  ;   sets the protection key given by pk value on the whole process area
14906  ;   of the internal process proc.
14906  ; call:                w3 = proc addr
14906  ;                      w0 = pk value
14906  ; note:                jl. w2   e22.
  906  ; return:              w1 = top core.proc addr
14906
14906  b.   g24                   ; begin
14906  w.                         ; set pk:
14906
14906  e22: rl  w1  x3+~~a17~~a18  ;   i:= first core.proc: goto test;
14908       jl.      g2.          ;
14910
14910  g0:  ks  w0  x1-20         ; set on ten:
14912       ks  w0  x1-18         ;
14914       ks  w0  x1-16         ;   for j:= -20 step 2 until -4 do
14916       ks  w0  x1-14         ;   pk (i+j):= pk value:
14918       ks  w0  x1-12         ;
14920       ks  w0  x1-10         ;
14922       ks  w0  x1 -8         ;
14924       ks  w0  x1 -6         ;
14926       ks  w0  x1 -4         ;
14928  g1:  ks  w0  x1 -2         ; set on one:  pk (i-2):= pk value;
14930
  930  g2:  al  w1  x1+20         ; test:   i:= i+20;
14932       sh  w1 (x3+a18)       ;   if i =< top core.proc then
14934       jl.      g0.          ;   goto set on ten:
14936       al  w1  x1-18         ;   i:= i-18;
14938       sh  w1 (x3+a18)       ;   if i =< top core.proc
14940       jl.      g1.          ;   then goto set on one;
14942       jl       x2           ;   return;
14944
14944  e.                         ; end set pk;
14944
14944  ; procedure test name;
14944  ;   supplies if word(d17) = 0  a unique working name and stores it in
14944  ;   name area of the sender, with the format: wrk<6 octal digits>.
14944  ;   otherwise it is tested that the name corresponds to a legal identifi
14944  ; call:                  jl. w3   e23.
14944  ; return:                name ok, name.work defined
14944  ;                        all registers used.
14944
14944  b.   g24                     ; begin
  944  w.                           ; test name:
14944  e23: rs. w3   g0.            ;
14946       rl. w2   d17.           ;   if not working name created then
14948       se  w2     0            ;   goto format;
14950       jl.      g11.           ;
14952
14952  g10: dl. w1  (c9.+4)         ; create:
```

```
14954        aa. w1   g3.              ;   digits:= digits + decimalcount;
14956        lo. w0   g2.              ;   digits(0):= digits(0)
14958        la. w0   g4.              ;      or digitmask and octalmask;
14960        lo. w1   g2.              ;   digits(1):= digits(1)
14962        la. w1   g4.              ;      or digitmask and octalmask;
14964        rl. w2   d2.              ;
14966        rl  w2   x2+f20           ; move name:
14968        ds. w1   (c9.+4)          ;
  970        dl. w1   (c9.+2)          ;   name(0):= <:wrk:>;
14972        ds  w1   x2+2             ;   name(2,4):= digits;
14974        dl. w1   (c9.+6)          ;   name(6):= 0;
14976        ds  w1   x2+6             ;
14978
14978        jl. w3   e1.              ;   set work name;
14980        jl.      (g0.)            ;   return;
14982
14982   g11: jl. w3   e3.              ; format: test format;
14984        jl.      j6.              ;   if illegal then goto error 6;
14986        jl.      (g0.)            ;
14988
14988   g2:  <:000:> , g4: <:777:>     ; masks
14992        8.6214 4310               ;
14994   g3:  8.6214 4311               ; decimalcount
14996   g0:  0             ; return    ;
14998   e.                             ; end test name;
14998   d16:  0, r.8                   ; answer buffer
15014
  014   d17: -1                        ; boolean: working name created.
15016
15016
15016   ; procedure created name;
15016   ;   if a working name was created then this procedure returns (x3+0)
15016   ;   else exit to error 3 takes place.
15016   ; call:               al. w3   <return addr.>
15016   ;                     jl.      e24.
15016   ; return:             if work name created then x3
15016   ;                     else enabled error 3.
15016
15016   e24: so. w3  (d17.)            ;   if working name created
15018        je.      j3.              ;   then return
15020        jl       x3               ;   else enabled goto error 3;
15022
15022   ; functions:
15022
15022
15022
  022   ; create entry.
15022   ;   ensures that name.work is a proper identifier and that it does not
15022   ;   already exist in the catalog, creates maybe a working name,
15022   ;   ensures that there is room for the new entry,
15022   ;   moves the tail specified by the tail address.sender to tail.work,
15022   ;   reserves an area if required (first word tail > 0),
15022   ;   resets the catalog key to zero for the entry,
15022   ;   sets the creation number of the calling process in the new entry,
15022   ;   which finally is put into the catalog.
15022   ; note: the strategy for finding a free area is simple-minded, it just
15022   ;       takes the first one which is big enough.
15022
15022   b.   g3                    ; begin
15022   w.                         ; create entry:
                         ? maybe leave it in for system use?
15022   i0:  i20 = i0 - j11        ;
~~15022        bl  w2   x1+a22     ;    if function mask (11).sender = 0~~
~~15024        sz  w2    1         ;    then begin~~
~~15026        jl.      g3.        ;          if free entries < cat segments~~
~~  028        rl. w2 (d4.)        ;          then goto error 5;~~
~~15030        rl. w3 (c0.)        ;       end test of emptying cat;~~
~~15032        sh  w2   x3-1       ;~~
~~15034        jl.      j5         ;~~
15036   g3:  jl. w3   e23.         ;    test name;
15038        jl. w3   e2.          ;    search (name.work);
15040        jl.      j6.          ;    if name format illegal then goto error 6;
```

```
15042        jl.        g0.          ;   if already in then
15044        al. w3     i0.          ;   if created name then goto
15046        jl.        e24.         ;   create entry  else goto error3;
15048  g0:   rl. w0    (d4.)         ; test room:
15050        sh  w0      0           ;   if free entries < 1
15052        jl.        j4.          ;   then goto error 4;
15054
15054        rl. w2     d2.          ;   creation number.entry:=
  056        rl  w1   x2+a41         ;   creation number.sender;
15058        rs. w1    d1.+f3        ;
15060
15060        rl  w2   x2+f21         ; move tail:
15062        al. w1    d1.+f6        ;   move
15064        al  w0     f8           ;   (tail address.sender,
15066        jl. w3     e8.          ;    tail.work,tail size);
15068        hs. w0    d1.+f2        ;   cat key.work:= 0;
15070
15070        rl. w0    d1.+f7        ; test for area:
15072        sh  w0      0           ;   if size.work > 0 then
15074        jl.        g2.          ; reserve area:
15076        jl. w3   ~~e31.~~ d11.  ;   find first hole(hole size);
15078        sl. w1   (d1.+f7)       ;   if hole size>=size.work then
15080        jl.        g1.          ;   goto hole found;
15082        sl  w1     1            ;   if hole size>=1 then
15084        jl.        e10.         ;   goto next hole;
15086        sn  w1     0            ;   if hole size=0 then
15088        jl.        e9.          ;   goto find holes;
  090        jl.        j5.          ;   goto error 5;
15092
15092  g1:   ac. w1   (d1.+f7)       ; hole found:
15094        jl. w3     e12.         ;   change them (-size.work):
15096        rl. w0    (d5.)         ;
15098        ws. w0    d1.+f7        ;   free segments := free segments - size.work;
15100        rs. w0    (d5.)         ;
15102        ~~rl. w1    d6.+2~~     ;   ~~first seg.work:=~~
15104        ~~ws. w1   (d9.)~~      ;   ~~(hole start word - start of bit table) * 1~~
15106        ~~wm. w1    g2.~~       ;   ~~+ hole start bit;~~
15108        ~~ws. w1    d6.~~       ;
15110        ~~rs. w1    d1.+f4~~    ;   end;                      moved.
15112
15112  g2:   jl. w3     e13.         ;   set new entry;
15114  i13:  jl.        j0.          ;   goto ex ok;
15116  e.                           ; end create entry;
15116  i33= i13-j11
15116  ; look up entry.
15116  ;   looks up the entry in the catalog, and
  116  ;   moves the tail of the entry to tail address.sender.
15116
15116  b.    g0                      ; begin
15116  w.                            ; look up:
15116
15116  i1:   i21 = i1 - j11          ;
15116        jl. w3     e2.          ;   search (name.work);
15118        jl.        j6.          ;   if not found then
15120        jl.        j3.          ;   goto error 3;
15122
15122  g0:   rl. w2     d3.          ; deliver tail:
15124        al  w2   x2+f6          ;
15126        rl. w1     d2.          ;   move
15128        rl  w1   x1+f21         ;   (tail.cur entry,
15130        al  w0     f8           ;    tail address.sender,
15132        jl. w3     e8.          ;    tail size);
15134
15134        jl.        j0.          ;   goto ex ok;
  136
  136  e.                            ; end look up;
15136
15136  ;  change entry.
15136  ;   tests that the sender may change the entry given by name.work,
15136  ;   if the change involves a change in area size then it tests that
15136  ;   the sender may change a possible area process,
15136  ;   performs the changes, including a possible release of area.
```

```
15136
15136  b.    g2                       ; begin
15136  w.                             ; change entry:
15136
15136  i2:   i22 = i2 - j11           ;
15136        rl. w2    d2.            ; move tail:
15138        rl  w2   x2+f21          ;   move
15140        al. w1    d1.+f6         ;   (tail address.sender,
  142        al  w0    f8             ;    tail.work,
15144        jl. w3    e8.            ;    tail size);
15146
15146        jl. w3    e15.           ;   test entry:
15148        jl.       g1.            ;
15150
15150        am         0             ;   if area then begin
15152        rl. w2    d3.            ;   comment area proc == w3 = 0;
15154        rl  w0   x2+f4           ;   w2:= cur entry;
15156        rs. w0    d1.+f4         ;   first seg.work:= first seg.cur entry;
15158        rl  w0   x2+f7           ;
15160        ws. w0    d1.+f7         ;   size diff:= size.cur entry - size.work;
15162        rs. w0    d11.           ;
15164
15164        sn  w0     0             ;   if size diff <> 0 then
15166        jl.       g2.            ;   begin
15168        sl  w0     0             ;   if size diff < 0  or
15170        sl  w0   (x2+f7)         ;      size diff >= size.cur entry then
15172        je.       j6.            ;      goto enabled error 6;
  174        rl. w0    d1.+f7         ;   if area process found
15176        sn  w3     0             ;   then
15178        rs  w0   x1+a61          ;   no of segs.proc:= size.work;
15180
15180  g0:   rl. w0   (d5.)           ;  enable:
15182        wa. w0    d11.           ;   free segments:= free segments + size diff;
15184        rs. w0   (d5.)           ;
15186
15186        rl. w1    d1.+f7         ;   change bits (size diff,
15188        wa. w1    d1.+f4         ;      first seg.work + size.work);
15190        rl. w2    d11.           ;   end;
15192        je. w3    e11.           ;   end
15194        jl.       g2.            ;
15196
15196  g1:   rl. w0    d1.+f7         ;   else if size.work > 0
15198        sl  w0     1             ;   then goto error 6;
15200        jl.       j6.            ;
15202  g2:   rl. w3    d3.            ;   keys.work:= keys.cur entry;
15204        dl  w1   x3+2            ;   creation number.work:=
  206        ds. w1    d1.+f3         ;   creation number.cur entry:
15208
15208        jl. w3    e7.            ;   set cur entry;
15210        jl.       j0.            ;   goto ex ok;
15212
15212  e.                             ; end change entry;
```

Leif typed to here.

```
15212
15212  ; rename entry.
15212  ;   tests that the name given by new name address.sender is a proper
15212  ;   identifier and that is does not already exist,
15212  ;   tests that sender may change the entry given by name address.sender,
15212  ;   tests whether a possible area process with the same name may be re-
15212  ;   moved, and removes it,
15212  ;   saves the tail of an entry and removes the entry,
15212  ;   sets the new name in name.work,
15212  ;   and puts the entry back into the catalog.
15212
15212  b.    g1                       ; begin
15212  w.                             ; rename entry:
  212
15212  i3:   i23 = i3 - j11           ;
15212        am.      (d2.)           ;
15214        rl  w2    f22            ;
15216        jl. w3    e1.            ;    set work name (new name address.sender);
15218
15218        dl. w0    d1.+f5+2       ; save new name:
```

```
15220       ds. w0   d10.+2       ;   save name:= name.work;
15222       dl. w0   d1.+f5+6     ;
15224       ds. w0   d10.+6       ;
15226
15226       jl. w3   e2.          ;   search (new name);
15228       jl.      j6.          ;   if not identifier then goto error 6;
15230       jl.      g0.          ;
15232       jl.      j3.          ;   if found then goto error 3;
  234
15234  g0:  am.     (d2.)         ; not in:
15236       rl  w2   f20          ;
15238       jl. w3   e1.          ;   set work name (name address.sender);
15240       jl. w3   e15.         ;   test entry;
15242       jl.      g1.          ;   if area process found
15244       jd. w3   e16.         ;   then clear area proc;
15246
15246  g1:  rl. w2   d3.          ; save entry:
15248       al. w1   d1.          ;
15250       al  w0   f0           ;   move (cur entry,work,entry size);
15252       jl. w3   e8.          ;
15254       jl. w3   e14.         ;   remove cur entry;
15256       al. w2   d10.         ;
15258       jl. w3   e1.          ;   set work name (save name);
15260       jl. w3   e13.         ;   set new entry;
15262       jl.      j0.          ;   goto ex ok;
15264
15264  e.                         ; end rename;
  264
15264  ; remove entry.
15264  ;   tests whether sender may change the entry given by name.work,
15264  ;   tests whether a possible area process with the same name may
15264  ;   be removed and removes it.
15264  ;   releases a possible reserved area,
15264  ;   removes the entry.
15264
15264  b.   g1                         ; begin
15264  w.                              ; remove entry:
15264
15264  i4:  i24 = i4 - j11             ;
15264       jl. w3   e15.              ;   test entry;
15266       jl.      g1.               ;   if area then begin
15268       jd. w3   e16.              ;   if area proc then disabl clear area pro
15270
15270  g0:  rl. w2   d3.               ;   free segments:= free segments +
15272       rl. w0  (d5.)              ;   size.cur entry;
15274       wa  w0   x2+f7             ;
  276       rs. w0  (d5.)              ;
15278       rl  w1   x2+f4             ;   change bits (size.cur entry,
15280       rl  w2   x2+f7             ;                first seg.cur entry);
15282       jl. w3   e11.              ;   end;
15284
15284  g1:  jl. w3   e14.              ;   remove cur entry;
15286       jl.      j0.               ;   goto ex ok;
15288
15288  e.                              ; end remove entry;
15288
15288  ; permanent entry.
15288  ;   sets the creation number to zero and the cat key to
15288  ;   a specified value in an entry in the catalog.
15288
15288  b.   g0                    ; begin
15288  w.                         ; permanent entry:
15288
15288  i5:  i25 = i5 - j11        ;
15288       jl. w3   e15.         ;   test entry;
  290       jl.      g0.          ;   if not found then goto error 3;
15292       jd.      g0.          ;   if locked then goto error 4;
15294  g0:  rl. w2   d2.          ;   if new cat key illegal or
15296       rl  w0   x2+a23       ;   new cat key protects against
15298       rl  w2   x2+f23       ;   calling process
15300       ls  w0   x2           ;   then enabled goto error 4;
15302       sl  w0    0           ;
```

```
15304         je.       j4.       ;    if area proc found
15306         sn   w3     0       ;    then disabled:
15308         hs   w2   x1+a51    ;    cat key.proc:= new cat key;
15310         sn   w3     0       ;    and creator.proc:= 0;
15312         rs   w3   x1+a62    ;
15314         rl.  w1    d3.      ;
15316         al   w0     0       ;    creation number.cur entry:= 0;
15318         hs   w2   x1+f2     ;    cat key.cur entry:= new cat key;
  320         rs   w0   x1+f3     ;    cat seg operation:= write;
15322         al   w0    f15      ;    comment: causes the current segment
15324         hs.  w0    d8.+f30  ;    to be rewritten to the backing store;
15326         je.       j0.[redacted] ;    enabled goto ex ok;
15328
15328    e.                       ; end permanent entry;
15328
15328    ; create area process (name).
15328    ;   checks that the area claim.sender is not exceeded. searches the name
15328    ;   table for an area process with the name given by name.work. If an are
15328    ;   process is not found, the catalog is searched for an area entry.
15328    ;   the area process is described with the sender as a user.
15328    ;   logically the backing store is a linear array of segments. Physically
15328    ;   the segments may be on a number of different devices. Create area
15328    ;   process allocates the actual device number to the area process.
15328
15328    b.   g5                       ; begin
15328    w.                            ; create area process:
15328
  328    i6:  i26 = i6 - j11           ;
15328
              jd   w3   b37            ; disable: search name (name.work);
15330         se   w3   (b7)           ;   if found then goto
15332         jl.        g0.           ;   area defined;
15334         bl   w0   x1+a20         ;   if area claim.sender < 1
15336         sh   w0     0            ;   then enabled goto error 1;
15338         je.        j1., j1=k-2   ;
15340
15340         je.  w3    e2.           ; enable: search (name.work);
15342         jl.        j6., j6=k-2   ;   if name format illegal then goto error 6;
15344         jl.        j3., j3=k-2   ;   if not found then goto error 3;
15346
15346         al   w0    0             ; disable:
15348         rl   w3   (b5)           ;   for proc:= first area entry,
15350    g1:  al   w3   x3+a2          ;   proc+area descr lenth while
15352         se   w0 (x3-a2+2)        ;   name(0).proc <> 0 do;
15354         jd.        g1.           ;
15356         al   w3   x3-a2          ;
  358
15358    ;    w2 = cur entry
15358    ;    w3 = free description
15358    ;   transfers the information from the current catalog entry to the found
15358    ;   free area process description. The kind is unchanged.
15358
15358         ~~rl   w1   x2+f3~~        ;   if creation number.cur entry = 0
15360         ~~sn   w1    0~~           ;   or creator.cur entry = sender
15362         ~~jl.        g2.~~   Out in our system ;   then
15364         ~~am.       (d2.)~~        ;   begin
15366         ~~se   w1  (a41)~~         ;      if size.cur entry <= 0
15368         ~~je.        j5.~~         ;      then enabled goto error 4:
15370    g2:  rl   w0   x2+f7          ;   end
15372         sh   w0    0             ;   else goto error 5;
15374         je.        j4., j4=k-2   ;   size.area proc:= size.cur entry;
15376         ds   w1   x3+a62         ;   creator.area proc:= creator.cur entry;
15378         dl   w1   x2+f5+2        ; copy name:
15380         ds   w1   x3+a11+2       ;   name.area proc:= name.cur entry;
15382         dl   w1   x2+f5+6        ;   comment: the existence of the area proce:
  384         ds   w1   x3+a11+6       ;   is now established;
15386
15386    ; the following piece of code calculates the device from the segments.
15386
15386         rl   w0   x2+f4          ;   device:= first backing device;
15388         rl.  w1   (d18.)         ;   while first seg.cur entry >
15390    g5:  al   w1   x1+6           ;           first seg.next device in table
```

```
15392          sl   w0 (x1+2)          ;   do device:= next device in table;
15394          jl.       g5.           ;   first seg.area proc:=
15396          ws   w0   x1-4          ;   first seg.cur entry
15398          rs   w0   x3+a60        ;   -first seg.device;
15400          rl   w1   x1-6          ;   device no.area proc:= device no.device;
15402
15402          hl   w1   x2+f2        ;   set (device number, catalog key);
15404          rs   w1   x3+a50       ;
  406          ld   w1    -65         ;   reserved:=
15408          ds   w1   x3+a53       ;   users:= 0;
15410          ~~jl.       4~~ se w1 x3 ;   comment: skip rl w3 x3;
15412
15412   g0:    rl   w3   x3           ; area defined:
15414          rl   w0   x3+0         ;   if kind.proc <> area kind
15416          se   w0    f38         ;   then goto enabled error 4;
15418          je.        j4.         ;
15420          rl.  w2   d2.          ;
15422          rl   w1   x3+a53       ;   if already user
15424          sz   w1  (x2+a14)      ;   then goto enabled ex ok;
15426          je.        j0.         ;
15428          ~~rl.  w0   x3+a62       ;    if creator.proc <> 0~~        dette kan
15430          ~~sn   w0    0           ;    and creator.proc <> sender~~  give name
15432          ~~rl   w0   x2+a41       ;    then goto error 5;~~          uden users.
15434          ~~se   w0  (x2+a41)      ;~~
15436          ~~je.        j5.         ;~~
15438          lo   w1   x2+a14       ;   if area claim.sender < 1
15440          bz   w0   x2+a20       ;   then enabled goto error 1;
  442          sh   w0     0          ;
15444          je.        j1.         ;   users:= users.proc or id bit.sender;
15446          bs.  w0     1          ;   area claim.sender:=
15448          rs   w1   x3+a53       ;   area claim.sender - 1;
15450          hs   w0   x2+a20       ;
15452          je.        j0.         ;   enabled goto ex ok;
15454   j0=k-2
15454   e.                           ; end create area process;
15454
15454   ; create peripheral process (name,device number);
15454   ;    tests the name format, searches in the name table for a process desc
15454   ;    tion holding the name. if none is found the descriptions for periphe
15454   ;    devices are searched for the device number. If the device number is
15454   ;    found the name given by name.work is assigned to the peripheral proc
15454   ;    it is required that the sender process is a user of the device,
15454   ;    and that no other process has reserved it.
15454
15454   b.    g5                          ; begin
15454   w.                                ; create peripheral process:
  454
15454   i7:   j27 = i7 - j11               ;
15454          jl.  w3    e23.     ok     ;    test name;
15456          al.  w2    d1.+f5          ; disable:
15458          jd   w3    b37             ;    search name (name.work);
15460          sn   w3   (b7)             ;    if not found then
15462          jl.        g1.             ;    goto look for device:
15464          al.  w3    i7.             ;    goto if created name then
15466          jl.        e24.            ;    create p.process else error 3;
15468   g1:    rl.  w2    d2.             ; look for device:
15470          rl   w0   x2+f24           ;    for descr:= first device,
15472          ls   w0     6              ;    descr + 2 while
15474          rl   w1    b4              ;
15476   g0:    sl   w1   (b5)             ;    device number.descr <>
15478          je.        j4.             ;    device number.sender do;
15480          rl   w3   x1+0             ;
15482          al   w1   x1+2             ;    if device not found then
15484          se   w0   (x3+a50)         ;    goto enabled error 4:
15486          jd.        g0.             ;
  488
15488          rl   w0   x3+a53           ; device found:
15490          so   w0   (x2+a14)         ;   if sender is not user then
15492          je.        j2., j2=k-2     ;   goto enabled error 2:
15494          rl   w0   x2+a14           ;   if device reserved by other user
15496          so   w0   (x3+a52)         ;   then goto enabled error 5:
15498          je.        j5.             ;
```

```
       rl   w0   x3+a10      ;    if process kind <> mag tape
       [illegible]           ;    then
       ~~sn   w0   18~~ ok   ;    begin
       jl.       g2.         ;       if sender has not
       bz   w0   x2+a22      ;       function(4) = 1
       so   w0   1<7         ;       then goto error 1;
       je.       j1.         ;    end;

    g2:dl. w1    d1.+f5+2        ; copy name:
       ds   w1  x3+a11+2         ;    name.device:=
       dl.  w1   d1.+f5+6        ;    name. work;
       ds   w1  x3+a11+6         ;
       al   w0   0               ;    if kind.proc=18
       rl   w1  x3+a10           ;    or kind.proc=~~34~~
       ~~se   w1   18~~          ;    then param1.proc:= 0;
       sn   w1   ~~34~~ 18       ;
       rs   w0  x3+a70           ; enable:

       je.       j0.             ; goto ex ok;

 e.                              ; end create peripheral process;

 ; create internal process (name,parameters);
 ;   creates a description of an internal process with a given name and
 ;   a given set of parameters. the name and the parameters are checked
 ;   for legality.
 b.    g7                        ; begin
 w.                              ; create internal process:
 i8:   i28 = i8 - j11            ;
       bz   w0 x1+a21            ;    if internal claim.sender = 0
       sn   w0     0             ;    then goto error 1;
       jl.         j1.           ;
 g3:   jl. w3    e23.            ;    test name(name.work):
       al. w2   d1.+f5    ok     ;    disable;
       jd   w3   b37             ;    search name (name.work);
       sn   w3   (b7)            ;    if found then goto
       je.       g1.             ;    if created work name then
       al. w3    g3.             ;    create internal process else
       je.       e24.            ;    enabled error 3;

 g1:   al   w0     0             ;    enable;
       rl   w1    b6             ; find free:
 g0:   al   w1   x1+2            ;    for name table entry:= first internal
       rl   w2   x1              ;    step 2 until
       se   w0 (x2+2)            ;    name(0).proc(nametableentry) = 0
       je.       g0.             ;    do proc:= process(name table entry);
       rs.  w2   d14.            ;
       al   w1   x2+a17          ;    move (parameters) from:
       rl.  w2   d2.             ;         (sender) to:
       rl   w2   x2+f24          ;         (proc descr):
       dl   w0   x2+2            ;         (12 bytes);
       la.  w0    g6.            ;
       la.  w3    g6.            ;    first addr.sender(23):= 0;
       ds   w0   x2+2            ;    top addr.sender(23):= 0;
       al   w0   12              ;
       jl.  w3    e8.            ;

       rl.  w2    d14.           ;    w2=proc
       rl.  w1    d2.            ;    w1=sender
       rl.  w3    d24.           ;    creation number:=
       al   w3   x3+1            ;       creation number + 1;
       rs.  w3    d24.           ;    creation number.proc:=
       rs   w3   x2+a41          ;       creation number;
       ~~rl   w0   x1+a43~~      ;
       ~~rs   w0   x2+a42~~  OK. ;    ~~device mask.proc:= selection mask.sende~~
       ~~rs   w0   x2+a43~~      ;    ~~select mask.proc:= select mask.sender;~~

       ld   w0   -65             ;    quantum.proc:= 0;
       rs   w0   x2+a35          ;    run time.proc:= 0;
       ds   w0   x2+a36+2        ;    start wait.proc:= 0;
       ds   w0   x2+a39+2        ;
       rl.  w3    c5.            ;    im.proc:= standard interrupt mask;
```

```
15614        ds   w0  x2+a27           :   interrupt address.proc:= 0;
15616        dl   w0  x1+a23           : test masks:
15618        so   w0 (x2+a23)          :   if catalog mask.proc is not a subset
15620        je.       j1.             :   of catalog mask.sender   or:
15622        bz   w0  x2+a22           :   if function mask.proc is not a subset
15624        so   w3   (0)             :   of function mask.sender
15626        je.       j1.             :   then goto error 1;
15628        rl   w0  x2+a24           : test protection:
  630        sz.  w0  (g5.)            :   if pk.proc > 7  or  pr.proc > 255
15632        je.       j1.             :   then goto error 1;
15634        la.  w0   g7.             :
15636        bz   w3  x1+a25           :   if pk.proc = 0
15638        sz   w0   2.111           :
15640        jl.       6               :   and pk.sender <>0
15642        se   w3   0               :
15644        je.       j1.             :   then goto error 1;
15646        bz   w3   1               :   if bit (pr,pk) <>0 and pk <>0
15648        ls   w0  x3+4             :   then goto error 1;
11500 15650 ~~1150~~ al   w3   7      :   if pr.proc is not a subset
15652        lo   w3  x2+a24           :   of pr.sender then
15654        sl   w0   0               :   goto error 1;
15656        so   w3 (x1+a24)          :
15658        je.       j1.             :
15660  g4:   dl   w0  x2+a18           : test process size:
15662        sl   w3 (x1+a17)          :   if first addr.proc < first addr.sender
15664        sh   w0  x3               :   or   top addr.proc<= first addr.proc
15666        je.       j1.             :   or   top addr.proc >   top addr.sender
  668        sh   w0 (x1+a18)          :   then goto error 1;
15670        jd.       g2.             :   ic.proc:= first addr.proc;
15672        je.       j1.             :   disable;
15674
15674  g2:   rs   w3  x2+a33           : test claims:
15676        rl   w0  x1+a19           :   if buf claim.proc > buf claim.sender
15678        rl   w3  x2+a19           :   or area claim.proc > area claim.sender
15680        sz.  w3  (c4.)            :   then enabled goto error 1;
15682        je.       j1.             :   if internal claim.proc >
15684        ws   w0   6               :      internal claim.sender - 1
15686        bl   w3  x2+a21           :   then enabled goto error 1;
15688        sh   w3  -1               : ok decrease claims:
15690        je.       j1.             :   buf claim.sender:= buf claim.sender
15692        ac   w3  x3+1             :       - buf claim.proc;
15694        ba   w3  x1+a21           :   area claim.sender:= area claim.sender
15696        sl   w3   0               :       - area claim.proc;
15698        sz.  w0  (c4.)            :   int claim.sender:= int claim.sender
15700        je.       j1.             :       - internal claim.proc - 1;
15702        hs   w3  x1+a21           :
  704        rs   w0  x1+a19           :
15706
15706        dl.  w0   d1.+f5+2        : move name:
15708        ds   w0  x2+a11+2         :   comment:  still disabled:
15710        dl.  w0   d1.+f5+6        :   name.proc:= name.work;
15712        ds   w0  x2+a11+6         :
15714        dl   w0   b13+2           : process created:
15716        ds   w0  x2+a38+2         :   start run.proc:= time;
15718        rs   w1  x2+a34           :   parent.proc:= sender;
15720        al   w0   f47             :   stop count.proc:= 0;
15722        rs   w0  x2+a13           :   state.proc:= waiting start by parent;
15724        je.       j0.             :   goto enabled ex ok;
15726
15726  ~~g5:  8.7400 7770~~            : pr,pk mask
15728  g6:   8.7777 7776               : first 23 bits
15730  ~~g7:  8.7577 7777~~            : -1-1<19
15732  c8:   8.0001 0000               : to count in left byte
15734
15734  e.                              : end create internal process;
  734
15734  ; start internal process(name);
15734  ;   follows the process tree starting with the process given by name.worl
15734  ;   which must be a child of the sender (otherwise: error 3): if the stai
15734  ;   of the child is not waiting for start by parent error exit 2 is taker
15734  ;   if ok then the child and all the decendants with state = waiting for
15734  ;   start by ancestor found by following the tree are treated as follows'
```

```
15734  ;    the protection key is set on the whole process area, the description
15734  ;    address of the processes are chained together via wait address with (
15734  ;    chain holding the address of the last process.
15734  ;    when the tree is exhausted then the chain is followed in disabled mo(
15734  ;    and each process is entered in the timer queue, its state is set to (
15734  ;    ning and stop count for its parent is increased by one.
15734                   bz  w0  x3+a13  ;   if state.sender <> wait start by parent
15734  b.   g10         se  w0  f47     ;   then goto error2;  ; begin
  734  w.               je.     j2.     ;   ; start internal process:
15734                                  include the above corrections
15734  i9:   i29 = i9 - j11
15734        jl.  w3   e17.              first proc (proc addr, new state);
15736
15736  g0:   bz   w0   x3+a13          ; treat next:  disable;
15738        se   w0   x2+f41          ;   if state.proc addr = new state+no stop b(
15740        jl.       g1.             ;   then begin.enable;
15742
15742  ; to avoid unnescessary setting of protection keys for processes which
15742  ; shares the same core area and the same key (this is for instance the
15742  ; normal situation for most swap-operating systems) protection keys are
15742  ; not sat if all the following conditions are satisfied:
15742  ;
15742  ; 1* there is another process occupying exactly the same storage area,
15742  ; 2* this process does already have the required key,
15742  ; 3* the two processes have the same parent, and
15742  ; 4* the two processes have precisely one zero-bit in protection reg.
15742                                                       Remove
  742        dl   w1   x3+a18          ;
15744        rl   w3   b6              ;   descr:= first internal;
15746  g5:   rl   w2   x3              ; test storage:
15748  7     sn   w0   (x2+a17)        ;   if storage.descr <> storage.proc addr
15750        se   w1   (x2+a18)        ;   then goto take next;
15752        je.       g7.
15754        sl   w2   (x2+a11)        ; test existence:
15756        je.       g7.             ;   if name.descr=unknown
15758        rl.  w2   d14.            ;   then goto take next;
15760        rl   w0   x2+a34
15762        se   w0   (x1+a34)        ; test parent and key:                are
15764        je.       g6.             ;   if parent.descr <> perant.proc addr
15766        rl   w0   x2+a25    AHAH! ;   or key.descr <> key.proc addr
15768        se   w0   (x1+a25)        ;   or pr.descr <> pr.proc addr
15770        je.       g6.   w2        ;   then goto take next;
15772
15772        bz   w1   1               ; test pr bit:
15774        ac   w2   x2              ;   if pr+128 shift (-pk) <> 255
15776        al   w0   128             ;   then goto take next;
  778        ls   w0   x2              ; ok:
15780        ba   w0   x1+a24          ;   goto chain it;
15782        sn   w0   255             ;
15784        jl.       g8.             ;
15786                                           C
15786  g6:   dl   w1   x1+a18          ; take next:
15788  g7:   al   w3   x3+2            ;   desr:= next internal in name table;
15790        se   w3   (b7)            ;   if -, exhausted then goto test storage;
15792        je.       g5              ;
15794        rl.  w3   d14.            ;   set pk (proc addr,pk.proc addr);
15796        ~~rl w0   x3+a25~~        ; chain it:
15798        [redacted]                ;   chain and add children;
15800        jl.  w3   e18.            ;   end;
15802                  OK                 je. w2 e22.
15802  g1:   je.  w3   e20.            ; next process;
15804        jd.       g0.             ;   if more then goto treat next;
15806
15806  g4:   rl.  w3   d13.+4          ; tree exhausted now start process:
15808        jd.       g3.             ;   proc:= end chain;
  810                                  ;   disable; goto test more;
15810
15810  g2:   al   w0   a95             ; start one process:
15812        hs   w0   x3+a13          ;   state.proc:= running;
15814        rl   w2   x3+a34          ;
15816        bz   w1   x2+a12          ;   stop count(father).proc:=
15818        al   w1   x1+1            ;   stop count(father).proc + 1;
```

```
15820       hs   w1   x2+a12          ;
15822                                                   start of
15822       al   w2   x3+a16          ;   link(head.proc,timer  q);
15824  rL   al   w1    b2             ;
15826       jd   w3    b36            ;
15828       rl   w3   x2+a40-a16      ;   proc:= wait address.proc;
15830
15830  g3:  se   w3     0             ; test more: if proc <> 0 then
  832       jl.         g2.           ;   goto start one process;
15834       je.         j0.           ;   enabled goto ex ok;
15836
15836  e.                             ; end start internal process;
15836
15836  ; stop internal process (name,buf,result);
15836  ;   follows the process tree, starting with the process given by name.
15836  ;   work, of those processes which are not waiting for stop or already
15836  ;   stopped.
15836  ;      each of these processes is treated, in disabled mode, as follows:
15836  ;   if it is in a queue then it is removed from that queue,
15836  ;   if it is in a waiting state then the instruction counter is decrease
15836  ;   by 2 so that the original monitor call will be repeated when it is
15836  ;   restarted.
15836  ;   if stop count is zero then the state is set to:  if the process is
15836  ;   the direct child of the sender, i.e. the first process treated, then
15836  ;   wait start by parent, else wait start by ancestor; and stop count
15836  ;   for the parent is decreased by one, possibly stopping the parent.
15836  ;   if stop count is not zero then state is set to wait stop by parent
  836  ;   or wait stop by ancestor as above.
15836  ;   when the states of all the processess involved are set, the stop cour
15836  ;   of the process given by name.work is inspected. if the count is zero,
15836  ;   indicating that the processes are effectively stopped, then a normal
15836  ;   answer (result = 1) is send to the calling process.
15836
15836  b.   g5                       ; begin
15836  w.                            ; stop internal process:
15836
15836  i10: i30 = i10 - j11          ;
15836       jl. w3   e17.            ;   first proc (proc addr, new state);
15838       am.     (d2.)            ;   wait addr.proc:=
15840       rl  w0   a30             ;   stop buf.sender;
15842       rs  w0  x3+a40           ;
15844       rs. w3   d16.            ;   save proc;
15846
15846  g0:  bz  w0  x3+a13           ; treat next:  disable;
15848       sz  w0   f43             ;   state.w0:= state.proc;
15850       jl.      g3.             ;   if -, stopped bit (state.w0) then
  852
15852       hs  w2  x3+a13           ;   begin
15854       rl  w2  x3+a33           ;   state.proc:= new state;
15856       al  w2  x2-2             ;   if repeat bit (state.w0) then
15858       sz  w0   f40             ;   ic.proc:= ic.proc - 2;
15860       rs  w2  x3+a33           ;
15862
15862       al  w2  x3+a16           ;
15864       sz  w0   f44             ;   if out of queue bit (state.w0)
15866       jd  w3   b35             ;   then remove (proc);
15868       al  w3  x2-a16           ;
15870
15870  g1:  rl  w2  x3+a12           ; loop stop:
15872       sz. w2  (c7.)            ;   if stop count.proc = 0 and
15874       jl.      g2.             ;   -, no stop bit (state.proc) then
15876
15876       al  w2  x2+f41           ;   begin
15878       hs  w2  x3+a13           ;   state.proc:= state.proc + no stop bit;
15880       rl  w3  x3+a34           ;   proc:= parent.proc;
  882       bz  w2  x3+a12           ;   stop count.proc:=
15884       al  w2  x2-1             ;   stop count.proc - 1;
15886       hs  w2  x3+a12           ;   goto loop stop;
15888       jl.      g1.             ;   end;
15890
15890  g2:  jl. w3   e19.            ;   add children;
15892       sn  w0   0               ;   if children bits=0
```

```
     jl.      g4.          ;   then goto no more;
                           ;  end;

g3:  je. w3   e20.         ;   enable:  next proc (proc, newstate);
     jd.      g0.          ;   if more then goto treat next:
~~g4:  rl. w3   d16.         ; no more:unsave proc;~~
     al. w1   d16.         ;   if stop count.proc = 0 then
     rl  w2  x3+a40        ;   send answer (answ addr,
     bz  w0  x3+a12        ;   wait addr.proc,1);
     ba. w0    3           ;   comment:  if error exit
     sn  w0    1 ; used    ;   then no buffer at all
     jd      1<11+22       ;   was ever selected;
     je.       j0.         ;   goto enabled  ex ok;

e.                         ; end stop internal process;

; modify internal process (name,registers);
;   finds the process given by name.work and checks that it is a child o
;   the sender.  if the process is waiting for start by parent then the
;   given values of the registers and the instruction counter are set in
;   the process description.

b.   g5                              ; begin
w.                                   ; modify internal process:

i11: i31 = i11 - j11                 ;
     jl. w3   e17.                   ;    first proc (proc addr,new state);
     bz  w0  x3+a13                  ;    disable;
     se  w0   f47                    ;    if state.proc <> wait start by parent
     je.      j2.                    ;    then goto enabled error2;

g0:  rl. w2   d2.                    ;
     rl  w2  x2+f24                  ;    move (registers.sender,
     al  w1  x3+a28                  ;          save registers.child, 12 bytes):
     al  w0   12                     ;    enable;
     jd. w3   e8.                    ;
     je.      j0.                    ; goto ex ok;
e.                                   ; end modify internal process:

; procedure remove area (intproc,areaproc);
;   intproc is removed as user and as reserver of area proc.
; call:   w1= intproc,   w3=area proc
;         disabled call with link in w2

b.   g24                            ; begin
w.                                  ; remove area:
e25: rl  w0  x3+a53                 ;   if intproc not user
     so  w0 (x1+a14)                ;   of area proc
     je      x2+0                   ;   then enabled return;
     ws  w0  x1+a14                 ;   users.area:=users.area
     rs  w0  x3+a53                 ;   - id bit.intproc;
     sn  w0   0                     ;   if users.area = 0
     rs  w0  x3+a11                 ;   then name(0).area:=0:
     la  w0  x3+a52                 ;   if intproc is reserver
     rs  w0  x3+a52                 ;   then reservers.area:=0;
     al  w0    1                    ;   area claim.intproc:=
     ba  w0  x1+a20                 ;   area claim.intproc + 1;
     hs  w0  x1+a20                 ;   enabled return;
     je      x2+0                   ;
e.                                  ; end remove area;

; remove process (name);
;   area process: the sender is removed as user and reserver of the
;   process, possibly removing the area process (see procedure clear
;   area proc).
;   peripheral process: if the sender is allowed to call the function
;   the peripheral process is removed if it is not reserved by another
;   process.
;   internal process: if the process is a child of the sender and is
;   waiting for start by parent then
;   1*   the protection key is reset in the process area,
;   2*   the process is removed,
15962  ;    3*   the process is removed from all external processes,
15962  ;    4*   all message buffers involving the removed process are cleaned
15962  ;         up, so that the buffers may return to the pool,
15962  ;    2* to 4*  is applied to all descendants of the child in  a recursive
15962  ;    way.
15962
15962  b.   g24                          ; begin
15962  w.                                ; remove process:
  962  i12: i32 = i12 - j11              ; disable:
15962       jd  w3   b37                 ;   search name (name.work):
15964       sn  w3  (b7)                 ;   if not found then enabled
15966       je.      e26.                ;   goto test found:
15968       rl  w3  x3                   ;
15970       al. w2   j0.                 ;   return to ex ok:
15972
15972       rl  w0  x3+a10               ; get and examine kind:
15974       sn  w0   f38                 ;   if kind.proc = area kind then
15976       jd. w0   e25. ; w2 link      ;   remove area (sender,proc):
15978       sn  w0   f37                 ;   if kind.proc = internal kind then
15980       je.      g1.                 ;   enabled goto internal:
15982
15982       bz  w0  x1+a22               ; peripheral:
15984       so       1<5                 ;   if function not allowed then
15986       je.      j1.                 ;   enabled goto error 1;
15988       rl  w0  x1+a14               ;   if sender not user of process
15990       so  w0 (x3+a53)              ;   then enabled goto error 2;
15992       je.      j2.                 ;
  994       lo  w0  x3+a52               ;   if reserved by other then
15996       se  w0 (x1+a14)              ;   enabled goto error5:
15998       je.      j5.                 ;
16000       al  w0   0                   ;   name(0).proc:= 0;
16002       rs  w0  x3+a11               ;   comment: now removed;
16004       rs  w0  x3+a52               ;   reserved.proc:= 0;
16006       je.      j0.                 ;   enabled goto ex ok;
16008
16008  g1:                               ; internal:
16008       jl. w3   e17.                ;   first proc (proc addr,--);
16010       bz  w0  x3+a13               ;   if not child then goto error 3;
16012       se  w0   f47                 ;   if state.proc <> wait start by parent
16014       je.      j2.                 ;   then goto error 2;
16016  [illegible]                       ;
16018  [illegible]                       ;
16020       ~~je. w2   e22. ; w2 link      ;   set ok (proc,ok,parent);~~
16022
16022  g5:  jd. w3   e18.                ; link: chain and add children;
16024       je. w3   e20.                ;   next proc (proc addr,--);
  026       jd.      g5.                 ;   if more then disabled goto link;
16028
16028       rl. w3   d15.                ; tree exhausted: proc:= end chain;
16030
16030  g6:  al  w0   0   ; used          ; remove one process:
16032       rs  w0  x3+a11               ;   name(0).proc:= 0;
16034       ac  w2  x3+0                 ;   childrenbits:= -proc;
16036       ds. w3   d14.                ;   proc addr:= proc;
16038
16038       rl  w3   b4                  ;   extproc:= first periphl in name table:
16040  g2:  rs. w3   d12.                ; examine extproc:
16042       rl. w1   d14.                ;
16044       rl  w3  x3+0                 ;   if kind.extproc = area kind
16046       rl  w0  x3+a10               ;   then disable:
16048       sn  w0   f38                 ;   remove area (proc,extproc);
16050       jd. w2   e25. ; w2 link      ;   enable:
16052       rl  w2  x1+a14               ;   users(id bit.proc).extproc:= 0;
16054       ac  w0  x2+1                 ;   res(id bit,proc).extproc:= 0;
  056       la  w0  x3+a53               ;
  058       rs  w0  x3+a53               ;   comment: proc is removed as
16060       la  w0  x3+a52               ;   user and as reserver of
16062       rs  w0  x3+a52               ;   extproc;
16064
16064  g4:  rl. w3   d12.                ;   extproc:= next proc in name table:
16066       al  w3  x3+2                 ;   if extproc <> first intproc
16068       se  w3  (b6)                 ;   then goto
```

This is d2.

include original code.

```
16070        je.      g2.               ;   examine extproc;
16072
16072        rl  w1   b8+4              ; examine message buffers:
16074  g10:  jd.      2                 ;   disable;
16076        dl  w3  x1+6               ;   for buf:= first mess buf
16078        se. w2  (d13.)             ;   step buf size
16080        sn. w2  (d13.+2)           ;   until last mess buf do
16082        jd.      g12.              ;   begin
16084  g14:  se. w3  (d13.)             ;   if proc = abs (receiver.buf)
16086        sn. w3  (d13.+2)           ;   then clean to (buf): else
16088        jd.      g13.              ;   if proc = abs (sender.buf)  Vah.
16090  g11:  wa  w1   b8+8              ;   then clean from (buf);
16092        sh  w1  (b8+6)             ;   end;
16094        je.      g10.              ;
16096        rl. w3   d14.              ;
16098        dl  w1  x3+a21             ;
16100        al  w2    0                ;
16102        rx  w2  x3+a34             ;   claims.parent.proc:=
16104        hl. w1   g6.+1 ; note      ;   claims.parent.proc + claim.proc;
16106        aa  w1  x2+a21             ;   add one to int claim.parent.proc;
16108        wa. w1   c8.               ;   parent.proc:= 0;
16110        ds  w1  x2+a21             ;   proc:= wait addr.proc;
16112        rl  w3  x3+f26             ;   if proc <> 0 then enabled
16114        se  w3    0                ;   goto remove one process else
16116        je.      g6.               ;   enabled goto ex ok;
16118        je.      j0.               ;
16120                                   ; end remove process;
16120
16120  ; procedure clean to (buf);
16120  ;   delivers a dummy answer <receiver does not exist> in the queue of
16120  ;   the sending process (the buffer administration takes care if the
16120  ;   sender is removed).
16120  g12:  ac  w2  (b1)               ;   receiver.buf:= -procfunc;
16122        rs  w2  x1+4               ;
16124        al  w2  x1                 ;
16126        al. w1   d16.              ;
16128        al  w0   5                 ;
16130        jd      1<11+22            ;   send answer(5,answer addr,buf);
16132        al  w1  x2                 ;
16134        je.      g10.              ;            g10 = 2 60 } which one!!
16136
16136  ; procedure clean from (buf);
16136  ;   releases pending buffers and prepares the return of buffer claims to
16136  ;   the parents of removed processes.
16136
16136  g13:  rl. w3   d14.              ;   sender.buf:= -parent.proc;
16138        ac  w0 (x3+a34)            ;
16140        rs  w0  x1+6               ;
16142        sz  w2    -8               ;   if answer pending
16144        je.      g11.              ;   then
16146        al  w2  x1                 ;
16148        jd  w3   b39               ;   deliver answer (buf);
16150        je.      g11.              ;
16152  e.
16152
16152  ; monitor log (on or off);
16152  ;   turns the monitor log facility on or off. the turning off includes
16152  ;   writing a tape mark on the log tape.
16152
16152  ; call:  w0 = 0  turn log off,    w0 = 1  turn log on,
16152  ;        w1 = log tape station number (must be 9 track magtape).
16152  ; return:
16152  ;        w0 = 0  ok,  w0 = 2  call error,  w0 = 4  no device.
16152
16152  b.    g5                 ; begin
16152  w.                       ; monitor log:
16152
16152  i13:  i33=i13-j11        ;
16152        dl  w3  x1+a29     ;   if turn param <> 0 and turn param <> 1
16154        sz  w2  -2         ;   then goto error 2;
16156        jl.     j2.        ;
16158        wa  w3   6         ;   entry:= first device in name table;
```

```
16160        wa   w3   b4          ;    entry:= entry + 2*device no;
16162        sl   w3  (b4)         ;
16164        sl   w3  (b5)         ;    if entry is outside devices
16166        jl.       j4.         ;    in name table then goto error 4;
16168        rl   w3   x3          ;
16170        rl   w0   x3+a10      ;    proc:= name table(entry);
16172        se   w0   34          ;
16174        jl.       j4.         ;    if kind.proc <> 34
  176        rl   w1   x3+a50      ;    then goto error 4;
16178        sz   w2    1          ;    if turn param is odd
16180        jl.       g1.         ;    then goto set mode;
16182
16182   g0:  io   w0   x1          ; off:  busy:= sense(device,status);
16184        sx         1          ;    if busy then goto off;
16186        jl.       g0.         ;    write (device,tape mark);
16188        io   w0   x1+3        ;
16190        al   w1    0          ;    device:= 0:
16192
16192   g1:  rs   w1   b9+8        ; set mode: log mode:= 64*device:
16194        jl.       j0.         ;    goto ex ok:
16196   e.
16196
16196
16196   ; generate name (name area);
16196   ;    creates a unique working name (not allready existing in the catalog
16196   ;    nor in the name table) and moves it to the name area specified by the
16196   ;    the calling process.
  196
16196   b.   g24                        ; begin
16196   w.                              ; generate name:
16196   i14: i34 = i14 - j11            ;    working name created:= true;
16196        al   w0    0               ;
16198        rs.  w0   d17.             ;    test name;
16200   g0:  jl.  w3   e23.             ;
16202        jl.  w3   e2.              ;    search (name.work);
16204        jl.       j6.              ;    if found then
16206        jl.       g1.              ;    goto generate name;
16208        jl.       g0.              ;
16210
16210   g1:  rl.  w2   j20.             ; disable:
16212        jd   w3   b37              ;    search name (name.work);
16214        sn   w3  (b7)              ; enable:
16216        je.       j0.              ;    if found then goto generate name:
16218        je.       g0.              ;    goto ex ok;
16220   e.                              ; end generate name;
16220
  220   j20: d1+f5                      ; pointer to name.work
16222
16222   ; copy(buf,first,last,result);
16222   ;    copies a core area from one process to another. buf
16222   ;    must be a message buffer in the queue of the calling
16222   ;    process, defining input from the calling process or output
16222   ;    to it.
16222   b.   g24                         ; begin
16222   w.
16222   i15: i35 = i15 - j11             ; copy:
16222        dl   w0   x1+a31            ;    buf:= save w2.proc;
16224        ws   w0   x1+a29            ;    diff:= save w3.proc-save w1.proc;
16226        ~~rl   w2   x3+12~~
16228 OK.    ws   w2   x3+10             ;    diff1:= last addr.buf-first addr.buf;
16230        sl   w0   x2+0              ;    if diff1<diff then
16232        ~~al   w0   x2+0~~              ;    diff:= diff1;
16234        ds.  w0   g6.
16236        rl   w2   x1+a29            ;    to:= save w1.proc;
16238        ~~bl   w1   x3+9~~
  240        ~~se   w1   0                 ;    if mode.buf<>0 then~~        Dumm
16242        ~~jl.       j3.               ;    goto result 3;~~
16244        rl   w1   x3+10             ;    from:= first addr.buf;
16246        bl   w3   x3+8
16248        sn   w3   5                 ;    if operation.buf=5 then
16250        jl.       g0.               ;    goto move;
16252        se   w3   3                 ;    if operation.buf<>3 then
```

```
16254        jl.       j3.          ;    goto result 3;
16256        rx   w2   2            ;    exchange(from,to);
16258  g0:   wa   w0   2            ;  move:
16260        ba.  w0   -1
16262        rs.  w0   g5.          ;    top:= from+diff+2;
16264        jl.       g2.          ;    goto test sixteen;
16266  g1:   al   w2   x2+32        ;  move sixteen:
16268        dl   w0   x1-30        ;    to:= to+32;
  270        ds   w0   x2-30
16272        dl   w0   x1-26        ;    for i:= -32 step 2 until -2 do
16274        ds   w0   x2-26        ;    word(to+i):= word(from+i);
16276        dl   w0   x1-22
16278        ds   w0   x2-22
16280        dl   w0   x1-18
16282        ds   w0   x2-18
16284        dl   w0   x1-14
16286        ds   w0   x2-14
16288        dl   w0   x1-10
16290        ds   w0   x2-10
16292        dl   w0   x1-6
16294        ds   w0   x2-6
16296        dl   w0   x1-2
16298        ds   w0   x2-2          ;  test sixteen:
16300  g2:   al   w1   x1+32        ;    from:= from+32;
16302        sh.  w1   (g5.)        ;    if from<=top then
16304        jl.       g1.          ;    goto move sixteen;
16306        al   w1   x1-32        ;    from:= from-32;
  308  g3:   sl.  w1   (g5.)        ;  test one:
16310        jl.       g4.          ;    if from>=top then
16312        rl   w0   x1+0         ;    goto exit;
16314        rs   w0   x2+0         ;    word(to):= word(from);
16316        al   w1   x1+2         ;    from:= from+2;
16318        al   w2   x2+2         ;    to:= to+2;
16320        jl.       g3.          ;    goto test one;
16322  g4:   dl.  w2   g6.          ;  exit:
16324        al   w2   x2+2
16326        al   w3   x2+0
16328        ls   w3   -1           ;    save w1.proc:= diff+2;
16330        wa   w3   4
16332        am        -2048
16334        rl.  w1   d2.+2048     ;    save w3.proc:=(diff+2)/2*3;
16336        rs   w2   x1+a29       ;
16338        rs   w3   x1+a31       ;
16340        jl.       j0.          ;    goto ex ok;
16342
16342  g5:   0   ; top
  344        0   ; saved buf
16346  g6:   0   ; saved diff
16348  e.
16348
16348  ; creation number.  each time an internal process is created then this
16348  ;    number is increased by one (modulo 2 ** 24) and set in the process
16348  ;    description of the new process.
16348
16348  d24: 1                         ; initially 1
16350
16350  d0:  0, r.f9>1                 ; cat buf (0:size-2);
16860  d20: 0                         ;    last word of cat buf.
16862  d19 = d0 - 2 + f10*f0          ;    abs addr of last word of last entry
16862                                 ;       in cat buf.
16862
16862  ; interrupt address.
16862  ; in case of errors send a message to the second internal process
16862  ; (assumed to be an operating system to take action).
16862
  862  o.    g24                    ;begin
16862  w.                           ; interrupt
16862  g0:   0<12+3<5               ; error message:
16864     <:monitor error:>         ;    monitor error <cause> <ic>
16874        0,0                    ;
16878        L:51, 0, r.4
16878  g1:   ~~0, r.3~~             ; opsys name
```

```
16888
16888  e30: 0, r.7                  ; register dump
16902       ~~rl  w3  66~~          ;
16904       ~~al  w3  x3+a4~~       ; break:
16906       ~~dl  w1  x3+4~~        ;   move the name of the next
16908       ~~ds. w1  g1.+2~~       ;   internal process to proc func;
16910       ~~dl  w1  x3+8~~        ;
16912       ~~ds. w1  g1.+6~~       ;   move cause and ic
  914  →    al. w1  g0.             ;   to err message;
16916       rl. w0  e30.+10         ;
16918       rl. w3  e30.+12         ;   send message (opsys,err,buf);
16920       ds  w0  x1+14           ;
16922       al. w3  g1.             ;   wait answer (buf,irr,irr);
16924       jd      1<11+16         ;
16926       al. w1   d16.           ;   goto waiting point;
16928       jd       1<11+18        ;
16930       jl.       (g2.)         ;
16932  g2:  j10                     ;
16934  e.                           ; end;
16934
16934
16934  ; define the last b-names:
16934
16934  b61 = k             ;   top address.proc func
16934  ~~b62 = e30         ;   interrupt address.proc func~~
16934  b63 = j10+2         ;   waiting point
16934  i.                  ; id list of process functions
  934
16934  ; after loading:
16934  b.   g0                          ; begin
16934  w.g0:al. w2   g0.                ; define last:
16936       jl       x3                 ;   autoload(next segment,top proc func);
16938
16938       jd.      g0.                ; after loading: goto define last;
16940  e.                               ; end.  the load code is removed;
16940    j21=k - b127 + 2
16940
16940  k = b61             ; top proc func
16934  e.                  ; end proc func segment
16934
16934
```

Leif
to here

```
16934
16934
16934  ; segment 7:  Initialize  process  functions
16934  ;   this segment initializes the process descriptions for first
16934  ;   process (proc func). it is executed and then removed
  934  ;   immediately after loading.
16934
16934  s.   g6                        ; begin  init proc func:
16934  w.b127=k, g6, k=k-2
16934
16934  ; process description for process functions:
16934
16934  w.                             ;
16934  g0:  0,0,<:proc func:>         ;   kind and pseudo name;
16944  h.   0,      a102              ;   stop count, state = waiting for message;
16946  w.   1<23                      ;   ident bit
16948       b29+a15, r.2              ;   next, last event
16952       b2%+a16, r.2              ;   next, last process
16956       b60     , b61             ;   first, top address
16960  h.   1,1,0,-1                  ;   claims and function mask;
16964  w.   -1                        ;   catalog mask
16966  h.   2.1000 0000, 0            ;   pr, pk
16968  w.   a89 , ~~b62~~0            ;   im, ia
16972       0, r.5                    ;   w0-w3, ex
  982       b63                       ;   IC = waiting point
16984       0,r.12                    ;   rest of process description cleared
17008
17008  ; init code:
17008
17008  g4:  al. w2   g0.              ; init:  w2:= descr(proc func);
17010       rl  w1   b6               ;        w1:= descr(first internal);
17012       rl  w1 x1+0               ;
17014
17014  g2:  rl  w0 x2                 ; proc func process:
17016       rs  w0 x1                 ;
17018       al  w1 x1+2               ;   move(descr(proc func) to:
17020       al  w2 x2+2               ;       (descr(first internal)));
17022       se. w2   g4.              ;
17024       jl.      g2.              ;
17026
17026  g5:  al. w2   g0.              ; autoloader(load addr defined);
17028       jl     x3                 ; comment: remove init code;
17030
  030       jl.      g4.              ; after loading: goto init;
17032  g6= k - b127 + 2
17032
17032  k = b61                        ; k= first after proc func;
16934  e.                             ; end init proc func
16934
16934  m.
16934                 monitor text 4 included
16934
16934
16934  m.
16934                 monitor text 5
16934
16934  ; segment 8: operating system s
16934  ; per brinch hansen
16934
16934  s. k=k, h14,g85,f29,e69,d49,c70
16934  w.b127=k, c70, k = k-2
 16934
  934  ; segment structure:
16934  ;     definitions          (c names)
16934  ;     utility procedures   (d names)
16934  ;     variables            (e names)
16934  ;     textstrings          (f names)
16934  ;     command actions      (g names)
16934  ;     tables               (h names)
```

check

```
16934  ;
16934  ;      (i and j names are used locally)          38
16934
16934  ; size options;
16934    c0=k    ; first addr of s
16934    c1=36   ; size of console description
16934    c2=86   ; size of work area
16934    c6=1          ; standard keys                                    50
  934    c7=7          ; standard buf                                        150 chars.
16934    c8=6          ; standard area
16934    c9=0          ; standard internal
16934    c10=8.7440    ; standard function
16934    c11=1<23      ; standard catalog
16934    c12=~~40000~~     ; standard size
16934    c23=-1<17     ; standard systemoptions
16934  t.
16934* type                50200

16934
16934  ; opsys s  size options
16934
16934
16934    c3=3                          ; number of work areas
16934    c6=1                          ; standard keys
16934    c7=7                          ; standard buf
16934    c8=6                          ; standard area
16934    c9=0                          ; standard internal
  934    c10=8.7440                    ; standard function mask
16934    c11=1<23                      ; standard catalog mask
16934    c12=12800                     ; standard size (=6400 words)
16934    c23=-1                        ; any system option included
16934  n.m.                              all
16934                          s size options included
16934    c4=c3  ; no of own buffers
16934    c5=1   ; no of own area processes
16934
16934  ; systemoptions:
16934  ; systemoptions determine whether code is included for certain
16934  ; commands. they are defined by bits in the identifier c23
16934  ; as follows:
16934  ;
16934  ;     break:                 c23=c23 o. 1<22
16934  ;     include/exclude:       c23=c23 o. 1<21
16934  ;     call:                  c23=c23 o. 1<20
16934  ;     list:                  c23=c23 o. 1<19
16934  ;     max:                   c23=c23 o. 1<18
  934  ;     date:                  c23=c23 o. 1<17
16934  ;     replace:               c23=c23 o. 1<16
16934
16934      c24 = 0
16934
16934  ; format of console description:
16934
16934    c25=0              ; <device no>                                28
16934    c26=2 , c27=3      ; <keys><command mask>                        30
16934    c28=4              ; <console children>                          32
16934    c29=6              ; <process name>                              34
16934    c30=14             ; <first address>                             36
16934    c31=16             ; <top address>
16934    c32=18, c33=19     ; <buf claim><area claim>
16934    c34=20, c35=21     ; <internal claim><function mask>
16934    c36=22             ; <catalog mask>
16934    c37=24, c38=25     ; ~~<protection register><protection key>~~   bs
16934    c39=26             ; <size>
 6934    c40=28             ; <program name>
  934    c41=36             ; <device mask>
16934  ; meaning of command mask:
16934  ; bit 0: (not used)
16934  ; bit 1: (not used)
16934  ; bit 2: list, max, date
16934  ; bit 3: call
16934  ; bit 4: include, exclude
```

```
16934  ; bit 5: function, catalog, or, ok
16934  ; bit 6: addr, size, buf, area, internal, key
16934  ; bit 7: new, proc, prog, create, init
16934  ;        run, load, start, stop, break, remove
16934  ; bit 8: privileged console
16934  ; bit 9: absolute protection
16934  ; bit 10: absolute address
16934  ; bit 11: console served
  934
16934  ; format of work area:
16934
16934    c50=0             ; <state>
16934    c51=2             ; <interrupt addr>
16934    c52=4             ; <console>
16934    c53=6             ; <console buf> or <last addr>
16934    c54=8             ; <char shift>
16934    c55=10            ; <char addr>
16934    c56=12            ; <child>
16934    c57=14            ; <core addr>
16934    c65=16            ; <textline=30 or 75 characters>
16934    c66=c65+20        ; <input buffer start>
16934    c67=c2-2          ; <last addr.input buffer>
16934  ; the input buffer may be overwritten by output in certain cases
16934
16934  ; meaning of work area state:
16934  ; state=0            available
16934  ; state=buf addr     waiting for answer
  934
16934  ; interrupt address: first event.
16934    ~~d0: 0, r.7, jl. g0,~~
16950
16950
16950  ; procedure next char(char,type)
16950  ; comment: unpacks and classifies the next character from
16950  ; the console buffer:
16950  ;      character type:
16950  ;      0   <small letter>
16950  ;      1   <digit>
16950  ;      2   <radix point>
16950  ;      3   <space>
16950  ;      4   <separator>
16950  ;      5   <end line>
16950  ;      6   <other graphic>
16950  ;      7   <blind>
16950  ;      call:       return:
16950  ; w0               char
  950  ; w1               type
16950  ; w2               destroyed
16950  ; w3  link         link
16950
16950  b.i24                        ; begin
16950  w.d1: dl. w2  e28.           ;
16952        sh  w1  0              ;   if charshift>0 then
16954        jl.     i0.            ;   begin
16956        al  w0  10             ;   char:=10;
16958        sn. w2 (e26.)
16960        jl.     i1.            ;   charshift:=-16;
16962        al  w1 -16             ;   charaddr:=charaddr+2;
16964        al  w2  x2+2           ;   end;
16966    i0: rl  w0  x2+0           ;
16968        ls  w0  x1+0           ;   char:=word(charaddr) shift charshift:
16970        la. w0  i3.            ;   char:=char(17:23);
16972        al  w1  x1+8           ;   charshift:=charshift+8:
16974    i1: ds. w2  e28.           ; classify char:
  976        rl  w1  0              ;
  978        ls  w1  -2             ;
16980        wa. w1  e5.            ;
16982        bz  w1  x1+0           ;   entry:=byte(chartable+char/4);
16984        so  w0  2.10           ;   type:=
16986        ls  w1  -6             ;   if char mod 4=0 then entry(0:2) else
16988        so  w0  2.01           ;   if char mod 4=1 then entry(3:5) else
16990        ls  w1  -3             ;   if char mod 4=2 then entry(6:8) else
```

```
16992         la. w1  i4.       ;                       entry(9:11);
16994         se  w1  5         ;   if type=5 then
16996         jl.     i2.       ;   begin
16998         rl. w2  e26.     .;
17000         rs. w2  e28.      ;   charaddr:= last addr;
17002         al  w2  8         ;   charshift:=8;
17004         rs. w2  e27.      ;   end;
17006    i2:                    ;
 006          jl      x3+0      ;   end;
17008    i3: 8.177              ;
17010    i4: 8.7                ;
17012   e.                      ; end
17012
17012   ; procedure next param(type)
17012   ; comment: converts and classifies the next parameter from
17012   ; the console buffer.
17012   ;       parameter type:
17012   ;       0    <empty>
17012   ;       1    <name>
17012   ;       2    <integer>
17012   ;       3    <unknown>
17012   ;       call:       return:
17012   ; w0                type
17012   ; w1                unchanged
17012   ; w2                unchanged
17012   ; w3    link        link
17012
 012    b.i24                   ; begin
17012   w.d2: rs. w3  e60.      ;
17014         ds. w2  e59.      ;       ld w1 -65
17016         ~~al  w0  0~~     ;
17018         ~~al  w1  0~~     ;   param type:=0;
17020         ds. w1  e19.      ;   integer:=0;
17022         ds. w1  e21.      ;
17024         ds. w1  e23.      ;   name:=0;
17026         al  w0  10        ;
17028         rl. w1  e6.       ;   radix:=10;
17030         ds. w1  e57.      ;   state:=param table;
17032
17032    d3: jl. w3  d1.        ; continue:
17034         wa. w1  e57.      ;   next char(char,type);
17036         bz  w1  x1+0      ;   entry:=byte(state+type);
17038         ld  w2  -2        ;   action:=entry(0:9);
17040         ls  w2  -19       ;
17042         wa. w2  e6.       ;   state:=
17044         rs. w2  e57.      ;   param table+8*entry(10:11);
 046          jl.     x1+d2.    ;   goto action;
17048
17048    d4: rl. w3  e19.       ; letter:
17050         sl  w3  11        ;   if integer>=10
17052         jl.     d7.       ;   then goto unknown;
17054         al  w2  0         ;
17056         wd. w3  i0.       ;
17058         ls  w2  3         ;   char:=char shift
17060         ac  w2  x2-16     ;   (16-integer mod 3 * 8);
17062         ls  w0  x2+0      ;
17064         ls  w3  1         ;   addr:=name+integer/3*2;
17066         lo. w0  x3+e20.   ;
17068         rs. w0  x3+e20.   ;   word(addr):=word(addr) or char;
17070         rl. w3  e19.      ;
17072         al  w3  x3+1      ;
17074         al  w2  1         ;   integer:=integer+1;
17076         ds. w3  e19.      ;   param type:=1;
17078         jl.     d3.       ;   goto continue;
17080
 080     d5: al  w3  0          ; radix:
17082         rx. w3  e19.      ;   radix:=integer;
17084         rs. w3  e56.      ;   integer:=0;
17086         jl.     d3.       ;   goto continue;
17088
17088    d6: rl. w3  e19.       ; digit:
17090         wm. w3  e56.      ;
```

```
17092        al   w3   x3-48      ;    integer:=
17094        wa   w3   0          ;    integer*radix-48+char;
17096        al   w2   2          ;    param type:=2;
17098        ds.  w3   e19.       ;
17100        jl.       d3.        ;    goto continue;
17102
17102    d7: al   w2   3          ; unknown:
17104        rs.  w2   e18.       ;    param type:=3;
17106    d8: rl.  w0   e18.       ; delimiter:
17108        dl.  w2   e59.       ;
17110        jl.      (e60.)      ;
17112    i0: 3                    ;
17114  e.                         ; end
17114
17114  ; procedure next name
17114  ; comment: checks that the next parameter from the console
17114  ; buffer is a name:
17114  ;       call:      return:
17114  ; w0               type
17114  ; w1               unchanged
17114  ; w2               unchanged
17114  ; w3    link       link
17114
17114  b.i24                      ; begin
17114  w.d15:rs.  w3   i0.        ;
17116        jl.  w3   d2.        ;    next param(type);
17118        se   w0   1          ;    if type<>1
17120        jl.       g2.        ;    then goto end line;
17122        jl.      (i0.)       ;
17124    i0: 0                    ; end
17126
17126
17126  ; procedure next integer(integer)
17126  ; comment: checks that the next parameter from the console
17126  ; buffer is an integer.
17126  ;       call:      return:
17126  ; w0               integer
17126  ; w1               unchanged
17126  ; w2               unchanged
17126  ; w3    link       link
17126
17126  w.d16:rs.  w3   i0.        ; begin
17128        jl.  w3   d2.        ;    next param(type);
17130        se   w0   2          ;    if type<>2
17132        jl.       g2.        ;    then goto end line;
17134        rl.  w0   e19.       ;
17136        jl.      (i0.)       ;
17138  e.                         ; end
17138
17138  ; procedure init read(addr)
17138  ; comment: prepares the reading of characters from a given
17138  ; storage address.
17138  ;      call:      return:
17138  ; w0              unchanged
17138  ; w1   addr       destroyed
17138  ; w2              unchanged
17138  ; w3   link       link
17138
17138  b.i24                      ; begin
17138  w.d17:rs.  w1   e57.       ;    read addr:=addr;
17140        al   w1   -16        ;
17142        rs.  w1   e56.       ;    read shift:=-16;
17144        jl        x3+0       ;
17146  e.                         ; end
17146
17146
17146  ; procedure read char(char)
17146  ; comment: unpacks the next character from a storage address
17146  ; initialized by init read.
17146  ;      call:      return:
17146  ; w0              char
17146  ; w1              unchanged
```

```
17146  ; w2                unchanged
17146  ; w3  link          link
17146
17146  b.i24                          ; begin
17146  w.d18:rx. w1  e56.             ;
17148        rx. w2  e57.             ;
17150        sh  w1  0                ;    if readshift>0 then
17152        jl.     i0.              ;    begin
  154        al  w1  -16              ;    readshift:=-16;
17156        al  w2  x2+2             ;    read addr:=read addr+2;
17158    i0: rl  w0  x2+0             ;    end;
17160        ls  w0  x1+0             ;    char:=word(read addr) shift readshift
17162        la. w0  i1.              ;    char:=char(17:23);
17164        al  w1  x1+8             ;    readshift:=readshift+8;
17166        rx. w1  e56.             ;
17168        rx. w2  e57.             ;
17170        jl      x3+0             ;
17172    i1: 8.177                    ;
17174  e.                             ; end
17174
17174  ; procedure init write
17174  ; comment: prepares the writing of characters in the line buffer
17174  ; within the current work area.
17174  ;      call:     return:
17174  ; w0             unchanged
17174  ; w1             unchanged
17174  ; w2             unchanged
  174  ; w3  link       link
17174
17174  b.i24                          ; begin
17174  w.d19:rs. w3  e55.             ;
17176        rl. w3  e24.             ;
17178        al  w3  x3+c65           ;
17180        rs. w3  e45.             ;   line addr:=work+linebuf;
17182        rs. w3  e46.             ;   writeaddr:=lineaddr;
17184        al  w3  16               ;   writeshift:=16;
17186        rx. w3  e55.             ;
17188        jl      x3+0             ;
17190  e.                             ; end
17190
17190
17190  ; procedure writechar(char)
17190  ; comment: packs the next character in the storage address
17190  ; initialized by initwrite.
17190  ;      call:     return:
17190  ; w0   char      destroyed     w.  am       22   ; space
  190  ; w1             unchanged         al  w0  10   ; new line.
17190  ; w2             unchanged     d20: rx. w1 e55.
17190  ; w3  link       link
17190
17190  b.i24                          ; begin
17190  ~~w.d20:rx. w1  e55.~~             ;   if writeshift<0
17192        rx. w2  e46.             ;   then
17194        sl  w1  0                ;   begin
17196        jl.     i0.              ;   writeshift:=16;
17198        al  w1  16               ;   writeaddr:=writeaddr+2;
17200        al  w2  x2+2             ;   end;
17202    i0: ls  w0  x1+0             ;   char:=char shift writeshift;
17204        se  w1  16               ;   if writeshift<>16 then
17206        lo  w0  x2+0             ;   char:=char or word(writeaddr);
17208        rs  w0  x2+0             ;   word(writeaddr):=char;
17210        al  w1  x1-8             ;   writeshift:=writeshift-8;
17212        rx. w1  e55.             ;
17214        rx. w2  e46.             ;
17216        jl      x3+0             ;
  218  e.                             ; end
17218
17218  ; procedure writetext(addr)
17218  ; comment: moves a textstring terminated by a null to the
17218  ; storage address initialized by initwrite.
17218  ;      call:     return:
17218  ; w0             destroyed
```

```
17218  ; w1  addr       destroyed
17218  ; w2             unchanged
17218  ; w3  link       link
17218
17218  b.i24                      ; begin
17218  w.d21:ds. w3  e60.         ;
17220        jl. w3  d17.         ;   initread(addr);
17222    i0: jl. w3  d18.         ;   readchar(char);
  224        sn  w0  0            ;   while char<>0 do
17226        jl.     i1.          ;   begin
17228        jl. w3  d20.         ;   writechar(char);
17230        jl.     i0.          ;   readchar(char);
17232    ~~i1: al  w0  32~~           ;   end;
17234    i1: jl. w3  d20.-2       ;   writechar(32);
17236    i6: dl. w3  e60.         ;
17238        jl      x3+0         ; end
17240
17240  ; procedure writeinteger(integer)
17240  ; comment converts a positive integer to a textstring which
17240  ; is moved to the storage address initialized by initwrite.
17240  ;      call:     return:
17240  ; w0             destroyed
17240  ; w1  integer    destroyed
17240  ; w2             unchanged
17240  ; w3  link       link
17240
17240   d22:ds. w3  e60.         ; begin
  242        al. w2  e54.         ;   addr:=conversion area;
17244    i2: al  w0  0            ;   repeat
17246        wd. w1  i4.          ;   byte(addr):=
17248        wa. w0  i5.          ;   integer mod 10+48;
17250        hs  w0  x2+0         ;   integer:=integer/10;
17252        al  w2  x2-1         ;   addr:=addr-1;
17254        se  w1  0            ;   until integer=0;
17256        jl.     i2.          ;
17258    i3: al  w2  x2+1         ;
17260        bz  w0  x2+0         ;   repeat
17262        jl. w3  d20.         ;   addr:=addr+1;
17264        se. w2  e54.         ;   writechar(byte(addr));
17266        jl.     i3.          ;   until addr=conversion area;
17268        jl.     ~~i6.~~ i1       ;   writechar(32);
17270    i4: 10                   ;
17272    i5: 48                   ;
17274  e.                         ; end
17274
17274  ; procedure typeline(buf)
  274  ; comment: starts the output on the current console of the line buffer
17274  ; within the current work area.
17274  ;      call:     return:
17274  ; w0             destroyed
17274  ; w1             destroyed
17274  ; w2             buf
17274  ; w3  link       destroyed
17274
17274  ~~b.i24~~                      ; begin
17274  w.d23:rs. w3  e60.         ;
17276        rl. w2  e25.         ;
17278        rl  w2  x2+c25       ;
17280        ls  w2  1            ;   proc:=
17282        wa  w2  b4           ;   name table(first device+
17284        rl  w2  x2+0         ;        2*device no(console));
17286        dl  w1  x2+4         ;
17288        ds. w1  e41.         ;
17290        dl  w1  x2+8         ;
17292        ds. w1  e43.         ;   receiver:=name(proc);
  294        al. w1  e44.         ;
17296        al. w3  e40.         ;
17298        jd  1<11+16          ;   send mess(receiver,typemess,buf);
17300        jl.    (e60.)        ;
17302  e.                         ; end
17302
17302  ; procedure find console(device no, console, sorry)
```

```
17302  ; comment: searches a console with a given device number.
17302  ;      call:      return:
17302  ; w0  device no device no
17302  ; w1             console
17302  ; w2             unchanged
17302  ; w3  link       link
17302
17302  b.i24                       ; begin
17302  w.d24:rl. w1  e9.           ;   for console:=first console
17304    i0: sn  w0 (x1+0)         ;   step console size
17306        jl      x3+2          ;   until last console do
17308        sn. w1 (e10.)         ;   if device(console)=device no
17310        jl      x3+0          ;   then goto found;
17312        al  w1  x1+c1         ;   goto sorry;
17314        jl.     i0.           ; found:
17316  e.                          ; end
17316
17316
17316  ; procedure find parent(child,console,sorry)
17316  ; comment: searches the parent console of a given child.
17316  ;      call:      return:
17316  ; w0             destroyed
17316  ; w1             console
17316  ; w2  child      child
17316  ; w3  link       link
17316
17316  b.i24                       ; begin
17316  w.d25:rl. w1  e9.           ;   for console:=first console
17318    i0: rl  w0  x1+c28        ;   step console size
17320        sz  w0 (x2+a14)       ;   until last console do
17322        jl      x3+2          ;   begin mask:=children(console);
17324        sn. w1 (e10.)         ;   if mask(id bit(child))=1
17326        jl      x3+0          ;   then goto found;
17328        al  w1  x1+c1         ;   end;
17330        jl.     i0.           ;   goto sorry;
17332  e.                          ; found:
17332                              ; end
17332
17332
17332  ; procedure next hole(hole addr,hole size,entry)
17332  ; comment: defines the start address and the size of the available
17332  ; storage area between two successive children in the core
17332  ; table. the core table address must be defined before next hole
17332  ; is called. upon return, the core table entry is loaded in w2
17332  ; and the core table address is increased by 2.
17332  ;      call:      return:
17332  ; w0             hole addr
17332  ; w1             hole size
17332  ; w2             entry
17332  ; w3  link       link
17332
17332  b.i24                       ; begin
17332  w.d26:rx. w3  e30.          ;
17334        rl. w0  e16.          ;   entry:=word(core addr-2);
17336        rl  w2  x3-2          ;   hole addr:=if core addr=core table
17338        se. w3 (e15.)         ;   then first core
17340        rl  w0  x2+a18        ;   else top addr(entry);
17342        rl. w1 (e17.)         ;   entry:=word(core addr);
17344        rl  w2  x3+0          ;   hole top:=if entry=0
17346        se  w2  0             ;   then top core
17348        rl  w1  x2+a17        ;   else first addr(entry);
17350        ws  w1  0             ;   hole size:=hole top-hole addr;
17352        al  w3  x3+2          ;   core addr:=core addr+2;
17354        rx. w3  e30.          ;
17356        jl      x3+0          ;
17358  e.                          ; end
17358
17358  ; procedure find size(start,size,sorry)
17358  ; comment: searches the core table for the first hole not less than
17358  ; a given size and delivers its start address.
17358  ;      call:      return:
17358  ; w0             first
```

```
17358  ; w1  size       size
17358  ; w2             destroyed
17358  ; w3  link       link
17358
17358  b.i24                         ; begin
17358  w.d27:rl. w2  e15.            ;
17360        rs. w2  e30.            ;   core addr:=core table;
17362    i0: rs. w1  e57.            ;
  364        rs. w3  e60.            ;   repeat
17366        jl. w3  d26.            ;   next hole(start,hole,entry);
17368        rx. w1  e57.            ;   if hole>=size
17370        rl. w3  e60.            ;   then goto found;
17372        sh. w1 (e57.)           ;   until entry=0;
17374        jl       x3+2           ;   goto sorry;
17376        se  w2  0               ;
17378        jl.      i0.            ; found:
17380        jl       x3+0           ;
17382  e.                            ; end
17382
17382  ; procedure find addr(start,size,sorry)
17382  ; comment: searches the core table for a hole with a given
17382  ; size and start address.
17382  ;      call:      return:
17382  ; w0  first      first
17382  ; w1  size       size
17382  ; w2             destroyed
17382  ; w3  link       link
  382
17382  b.i24                         ; begin
17382  w.d28:ds. w1  e57.            ;
17384        rl. w2  e15.            ;
17386        rs. w2  e30.            ;   core addr:=core table;
17388    i0: rs. w3  e60.            ;
17390        jl. w3  d26.            ;   repeat
17392        rl. w3  e60.            ;   next hole(first,hole,entry);
17394        wa  w1  0               ;   top:=first+hole;
17396        ws. w0  e56.            ;
17398        ws. w1  e56.            ;
17400        sl. w1 (e57.)           ;   if first<=start
17402        sl  w0  1               ;   and top-start>=size
17404        jl.      i1.            ;   then goto found;
17406        dl. w1  e57.            ;   until entry=0;
17408        jl       x3+2           ;   goto sorry;
17410    i1: se  w2  0               ;
17412        jl.      i0.            ;
17414        jl       x3+0           ; found:
  416  e.                            ; end
17416
17416  ; procedure find max(size)
17416  ; comment: searches the core table for the size of the biggest hole.
17416  ;      call:      return:
17416  ; w0             unchanged
17416  ; w1             size
17416  ; w2             destroyed
17416  ; w3  link       destroyed
17416
17416  [redacted]                    ; if max option then
17416  b.i24                         ; begin
17416  w.d29:rs. w3  e60.            ;
17418        al  w1  0               ;
17420        ds. w1  e57.            ;   size:=0;
17422        rl. w2  e15.            ;
17424        rs. w2  e30.            ;   core addr:=core table;
17426    i0: jl. w3  d26.            ;
17428        sl. w1 (e57.)           ;   repeat
  430        rs. w1  e57.            ;   next hole(first,hole,entry)
17432        se  w2  0               ;   if hole>size then size:=hole;
17434        jl.      i0.            ;   until entry=0;
17436        dl. w1  e57.            ;
17438        jl.     (e60.)          ;
17440  e.                            ;
17440  [redacted]                    ; end
```

```
17440
17440  ; procedure reserve core(child)
17440  ; comment: inserts a child in the core table entry given
17440  ; by core table address - 2 and moves the old and the following
17440  ; entries one position upwards.
17440  ;        call:       return:
17440  ; w0                 unchanged
17440  ; w1                 destroyed
  440  ; w2   child        destroyed
17440  ; w3   link         link
17440
17440  b.i24                        ; begin
17440  w.d30:rl. w1  e30.           ;   addr:=core addr;
17442    i0: rx   w2  x1-2          ;   repeat
17444        al   w1  x1+2          ;   exchange(child,word(addr-2));
17446        se   w2  0             ;   addr:=addr+2;
17448        jl.      i0.           ;   until child=0;
17450        jl       x3+0          ;
17452  e.                           ; end;
17452
17452  ; procedure release core(child)
17452  ; comment: removes a child from the core table and moves
17452  ; the following entries one position downwards.
17452  ;        call:       return:
17452  ; w0                 unchanged
17452  ; w1                 destroyed
17452  ; w2   child        destroyed
  452  ; w3   link         link
17452
17452  b.i24                        ; begin
17452  w.d31:rl. w1  e15.           ;   addr:=core table;
17454    i0: al   w1  x1+2          ;
17456        se   w2 (x1-2)         ;   repeat addr:=addr+2
17458        jl.      i0.           ;   until word(addr)=child;
17460    i1: rl   w2  x1+0          ;
17462        rs   w2  x1-2          ;   repeat
17464        sn   w2  0             ;   addr:=addr+2;
17466        jl       x3+0          ;   word(addr-2):=word(addr);
17468        al   w1  x1+2          ;   until word(addr)=0;
17470        jl.      i1.           ;
17472  e.                           ; end
17472
17472  ; procedure find keys(keys,pr,pk,sorry)
17472  ; comment: examines all children and creates a possible
17472  ; protection register with zeroes in all available protection
17472  ; bits. from this possible register, a protection register pr
  472  ; with a given number of keys is selected from left to right.
17472  ; the protection key pk is set equal to the right-most assigned
17472  ; key. upon return, keys is diminished by the number of assigned
17472  ; keys.
17472  ;        call:       return:
17472  ; w0                 pr
17472  ; w1                 pk
17472  ; w2   keys         keys
17472  ; w3   link         link
17472
17472  b.i24                        ; begin
17472  w.d32:ds. w3  e60.           ;
17474        rl   w1  b1            ;
17476        bz   w0  x1+a24        ;   possible:=pr(s);
17478        rl.  w2  e15.          ;   addr:=core table;
17480    i0: rl   w3  x2+0          ;   while word(addr)<>0 do
17482        sn   w3  0             ;   begin
17484        jl.      i2.           ;   child:=word(addr);
17486        bz   w3  x3+a24        ;   possible:=possible or
  488        lx.  w3  i1.           ;   (pr(child) exor last 7);
17490        lo   w0  6             ;   addr:=addr+2;
17492        al   w2  x2+2          ;   end;
17494        jl.      i0.           ;
17496    i1: 8.177                  ;
17498    i2: rl.  w2  e59.          ;   pr:=possible;
17500    i3: ls   w0  1             ;   bit:=16;
```

```
17502          al   w3   x3-1          ;    repeat
17504          sz   w0   1<7           ;    bit:=bit+1;
17506          jl.       i4.           ;    if pr(bit)=0 then
17508          al   w2   x2-1          ;    begin
17510          sn   w2   0             ;    keys:=keys-1;
17512          jl.       i5.           ;    if keys=0 then goto found;
17514     i4:  se   w3   -7            ;    end;
17516          jl.       i3.           ;    until bit=24;
  518          jl.      (e60.)         ;    goto sorry;
17520     i5:  lo.  w0   i1.           ;  found: pk:=bit;
17522          ls   w0   x3+0          ;    while bit<>24 do
17524          ac   w1   x3+0          ;    begin
17526          rl.  w3   e60.          ;    pr(bit):=1; bit:=bit+1;
17528          jl        x3+2          ;    end;
17530   e.                             ;  end
17530
17530   ; procedure child name
17530   ; comment: moves child name to receiver name.
17530   ;      call:       return:
17530   ; w0               destroyed
17530   ; w1               destroyed
17530   ; w2               child
17530   ; w3   link        link
17530
17530   b.i24                          ; begin
17530   w.d33:rl.  w2   e29.           ;
17532          dl   w1   x2+4          ;
  534          ds.  w1   e41.          ;
17536          dl   w1   x2+8          ;    receiver:=name(child);
17538          ds.  w1   e43.          ;
17540          jl        x3+0          ;
17542   e.                             ; end
17542
17542   ; procedure check child
17542   ; comment: checks that the process name in the console
17542   ; description refers to a child of s. the console must
17542   ; either be a privileged console or the parent of the
17542   ; child.
17542   ;      call:       return:
17542   ; w0               destroyed
17542   ; w1               console
17542   ; w2               child
17542   ; w3   link        destroyed                 rx w2 0
17542
17542   b.i24                          ; begin
17542   w.d34:rs.  w3   i0.            ;
  544          rl.  w1   e25.          ;
17546          al   w3   x1+c29        ;   process description(
17548          jd   1<11+4             ;     process name(console),result);
17550          rs.  w0   e29.          ;   child:=result;
17552          rl   w2   0             ;
17554          rl   w0   x2+0          ;
17556          se   w2   0             ;   if child=0
17558          se   w0   0             ;   or kind(child)<>0
17560          jl.       g9.           ;   then goto end line;
17562          jl.  w3   d25.          ;
17564          jl.       g3.           ;   find parent(child,parent,end line);   ok
17566          sn.  w1  (e25.)         ;
17568          jl.      (i0.)          ;   if console<>parent
17570          rl.  w1   e25.          ;
17572          bz   w0   x1+c27        ;   and not privileged(console)
17574          so   w0   1<3           ;
17576          jl.       g3.           ;   then goto end line;
17578          jl.      (i0.)          ;
17580     i0:  0
  582   e.                             ; end                 ch w1 e25.
17582
17582   ; procedure create child
17582   ; comment: allocates resources and creates a child process in
17582   ; accordance with the console parameters. the child is included as
17582   ; user of all devices in the device table. finally, the identification
17582   ; bit of the child is set in the description of the console.
```

```
rl w0 x1+8        ; always
lo.w0 i3.         ; select
rs w0 x1+8        ; key 0
```

```
17582  ;       call:      return:
17582  ; w0               destroyed
17582  ; w1               destroyed
17582  ; w2               destroyed
17582  ; w3   link        destroyed
17582
17582  b.i24                          ; begin
17582  w.d35:rs.  w3   i2.           ; find core:
  584        rl.  w2   e25.           ;
17586        rl   w0   x2+c30         ;   start:=first addr(console):
17588        rl   w1   x2+c39         ;   size:=size(console);
17590  ~~[illegible]~~
17592  ~~[illegible]~~                ; ~~if abs addr(console)~~
17594  ~~[illegible]~~                ; ~~then find addr(start,size,end line)~~
17596        jl.  w3   d27.           ;   else find size(start,size,end line);
17598        jl.       g4.            ;
17600        rl.  w2   e25.           ;
17602        rs   w0   x2+c30.        ;   first addr(console):=start:
17604        wa   w0   x2+c39         ;   top addr(console):=
17606        rs   w0   x2+c31         ;   start+size(console);
17608        ~~bz   w3   x2+c27~~      ; ~~find protection:~~
17610        ~~sz   w3   1<2~~         ;   ~~if not abs protection(console) then~~
17612        ~~jl.       i0.~~         ;   ~~begin~~
17614        ~~bz   w2   x2+c26~~
17616
17616        ~~jl.  w3   d32.~~        ;   ~~find keys(keys(console),~~
17618        ~~jl.       g8.~~         ;      ~~new pr,new pk, end line);~~
  620        ~~rl.  w2   e25.~~        ;   ~~pr(console):=new pr;~~
17622        ~~hs   w0   x2+c27~~      ;   ~~pk(console):=new pk;~~
17624        ~~hs   w1   x2+c32~~      ;   ~~end;~~
17626    ~~i0: bl   w0   x2+c37~~      ;
17628        ~~sz   w0   -1<8~~        ;   ~~if pr(console)(0:3)<>0 then~~
17630        ~~jl.       g8.~~         ;   ~~goto end line;~~
17632        rl   w3   b1             ; check claims:
17634        bz   w0   x2+c32         ;
17636        bz   w1   x3+a19         ;
17638        ws.  w1   e2.            ;   if buf claim(console)>
17640        sl   w0   x1+1           ;   buf claim(s)-own buf
17642        jl.       g5.            ;   then goto end line;
17644        bz   w0   x2+c33         ;
17646        bz   w1   x3+a20         ;   if area claim(console)>
17648        ws.  w1   e3.            ;
17650        sl   w0   x1+1           ;   area claim(s)-own area
17652        jl.       g6.            ;   then goto end line;
17654        bz   w0   x2+c34         ;
17656        bz   w1   x3+a21         ;   if internal claim(console)>
  658        sl   w0   x1+0           ;   internal claim(s)-1
17660        jl.       g7.            ;   then goto end line;
17662        al   w1   x2+c30         ;   create internal process(
17664        al   w3   x2+c29         ;    process name(console),
17666        jd   1<11+56             ;    first addr(console),result);
17668        sn   w0   1              ;
17670        jl.       g8.            ;
17672        se   w0   0              ;   if result<>0
17674        jl.       g10.           ;   then goto end line;
17676        jd   1<11+4              ;   process description(
17678        rs.  w0   e29. ←         ;     process name(console),result);
17680        am       (0)             ;   child:=result;
17682        rl   w1   a14            ;
17684        lo   w1   x2+c28         ;   children(console):=
17686        rs   w1   x2+c28         ;   children(console) or id bit(child);
17688        rl.  w2   e11.           ;   addr:=first device;
17690    i1: bz   w1   x2+0           ;   repeat
17692        jd   1<11+12             ;   include user(process name(console),
17694        al   w2   x2+1           ;                 byte(addr));
  696        sh.  w2  (e12.)          ;   addr:=addr+1;
17698        jl.       i1.            ;   until addr>last device;
17700        rl.  w2   e29.           ;
17702        jl.  w3   d30.           ;   reserve core(child);
17704        jl.      (i2.)           ;
17706    ~~i2: 0~~
17708  e.   ~~[illegible]~~           ; end
```

← Do not forget: i3:1<23

```
17708
17708  ; procedure modify child(addr)
17708  ; comment: modifies the registers of the current child as follows:
17708  ;      child w0 = 0 or process description of parent console
17708  ;      child w1 = process description of s
17708  ;      child w2 = process description of parent console
17708  ;      child w3 = process description of child
17708  ;      child ex = 0
  708  ;      child ic = addr
17708  ;      call:      return:
17708  ; w0   addr       destroyed
17708  ; w1              destroyed
17708  ; w2              destroyed        wa wl 2
17708  ; w3   link       destroyed
17708
17708  b.i24                     ; begin
17708  w.d36:rs. w3  i0.         ;
17710        rs. w0  e66.        ;   child ic:=addr;
17712        rl  w0  b1          ;
17714        rs. w0  e62.        ;   child w1:=s;
17716        jl. w3  d33.        ;   child name;
17718        jl. w3  d25.        ;   find parent(child,console,
17720        am      0           ;                irrelevant);
17722        rl  w1  x1+d25      ;
17724        ~~ls  w1  1~~           ;   child w2:=
17726        we  w1  b4          ;   name table(first device+
17728        rl  w1  x1+0        ;          2*device no(console));
  730        rs. w1  e61.        ;   child w0:= child w2;
17732        ds. w2  e64.        ;   child w3:=child;
17734        al. w1  e61.        ;
17736        al. w3  e40.        ;   modify internal process(
17738        jd  1<11+62         ;      receiver, child w0);
17740        jl.     (i0.)       ;
17742    i0: 0                   ;
17744  e.                        ; end
17744
17744  ; procedure load child
17744  ; comment: loads a program from backing store into
17744  ; a child process in accordance with the console parameters.
17744  ; the program must be described as follows in the catalog:
17744  ;             <size of area>
17744  ;             <6 irrelevant words>
17744  ;             <first segment to load>
17744  ;             <content=3><instruction counter>
17744  ;             <bytes to load>
17744  ;      call:      return:
  744  ; w0              destroyed
17744  ; w1              destroyed
17744  ; w2              destroyed
17744  ; w3   link       destroyed
17744
17744  b.i24                     ; begin
17744  w.d37:rs. w3  i20.        ; create and look up:
17746        rl. w2  e25.        ;
17748        dl  w1  x2+c40+2    ;
17750        ds. w1  e41.        ;
17752        dl  w1  x2+c40+6    ;
17754        ds. w1  e43.        ;   receiver:=prog(console);
17756        al. w3  e40.        ;
17758        jd  1<11+52         ;   create area process(receiver,result);
17760        sn  w0  2           ;   if result=2
17762        jl.     g11.        ;
17764        sn  w0  3           ;   or result=3
17766        jl.     g12.        ;
 7768        sn  w0  4           ;   or result=4
  770        jl.     g12.        ;   then goto end line;
17772
17772        ~~jd  1<11+8          ;   reserve process(receiver,result);~~
17774        ~~sn  w0  1           ;   if result=1~~
17776        ~~jl.     i0.         ;   then goto give up 0;~~
17778        al. w1  e51.        ;   look up entry(
17780        jd  1<11+42         ;     receiver,tail,result);
```

```
17782         sn   w0   2          ;    if result=2
17784         jl.       i1.        ;    then goto give up 0;
17786         rl. w2   e29.        ;  check description:
17788         bz. w0   e59.        ;
17790         se   w0   3          ;    if content(tail)<>3
17792         jl.       i2.        ;    then goto give up 0;
17794         rl   w0   x2+a17     ;    first addr(area mess):=
17796         rl. w1   e60.        ;    first addr(child);
  798         al   w1   x1+511     ;
17800         as   w1   -9         ;    load size:=
17802         as   w1   9          ;    (bytes(tail)+511)/512*512;
17804         wa   w1   0          ;    last addr(area mess):=
17806         al   w1   x1-2       ;    first addr(child)+load size-2;
17808         sl   w1  (x2+a18)    ;    if last addr(area mess)>=
17810         jl.       i3.        ;      top addr(child)
17812         ds. w1   e49.        ;      then goto give up 0;
17814         rl. w1   e58.        ;    segment(area mess):=
17816         rs. w1   e50.        ;    segment(tail);
17818         bz. w1   e67.        ;
17820         wa   w1   0          ;    child ic:=
17822         rs. w1   e66.        ;    first addr(child)+ic(tail):
17824         sl   w1  (x2+a18)    ;    if child ic>=top addr(child)
17826         jl.       i4.        ;    then goto give up 0;
17828         al. w1   e47.        ;  load program:
17830         al. w3   e40.        ;
17832         jd   1<11+16         ;    send mess(receiver,area mess,buf);
17834         al. w1   e51.        ;
  836         jd   1<11+18         ;    wait answer(buf,answer,result):
17838
17838         rl. w1   e51.        ;
17840         sn   w0   1          ;    if result<>1
17842         se   w1   0          ;    or status(answer)<>0
17844         jl.       i5.        ;    then goto give up 0;
17846         ~~al. w3   e40.~~    ;
17848         jd   1<11+64         ;    remove process(receiver,result);
17850         rl. w0   e66.        ;
17852         jl. w3   d36.        ;    modify child(child ic);
17854         jl.      (i20.)      ;    goto exit;
17856     i0: am        4          ;
17858     i1: am       -2          ;
17860     i2: am       -4          ;
17862     i3:
17862     i4: am       -2          ;
17864     i5: al. w3   g15.        ;
17866         ~~jl. w3   i20.~~    ;  give up 0:
17868         al. w3   e40.        ;
  870         jd   1<11+64         ;    remove process(receiver,result);
17872         ~~jl.      (i20.)~~  ;    goto end line;
17874     i20:0                    ;  exit:
17876                              ;  end;
17876
17876  ; procedure start child
17876  ; comment: starts a child process.
17876  ;      call:      return:
17876  ; w0              destroyed
17876  ; w1              destroyed
17876  ; w2              destroyed
17876  ; w3   link       destroyed
17876
17876  b.i24                       ; begin
17876  w.d38:rs. w3   i0.          ;
17878        jl. w3   d33.         ;    child name;
17880        al. w3   e40.         ;
17882        jd   1<11+58          ;    start internal process(receiver,result);
17884        jl.      (i0.)        ;
  886      ~~i0: 0~~               ;
17888                              ; end
17888
17888
17888  ; procedure stop child
17888  ; comment: stoos a child process.
17888  ;      call:      return:
```

everything as original

jl x2

2

```
17888  ; w0             destroyed
17888  ; w1             destroyed
17888  ; w2             destroyed
17888  ; w3  link       destroyed
17888
17888                                 ; begin
17888  w.d39:rs. w3  i20.             ;
17890        jl. w3  d33.             ;   child name;
  892        al. w3  e40.             ;
17894        jd  1<11+60              ;   stop internal process(receiver,buf,result);
17896        al. w1  e51.             ;
17898        jd  1<11+18              ;   wait answer(buf,answer,result);
17900        jl.     (i20.)           ;
17902  ~~i0: 0~~                      ;
17904  ~~e.~~                         ; end
17904
17904  ; procedure remove child
17904  ; comment: excludes a child as a user of all devices and
17904  ; removes it.
17904  ;      call:      return:
17904  ; w0             destroyed
17904  ; w1             destroyed
17904  ; w2             destroyed
17904  ; w3  link       destroyed
17904
17904  ~~b.i24~~       i20           ; begin
17904  w.d40:rs. w3  ~~i4.~~          ;
 /906        jl. w3  d33.             ;   child name;
17908        jl. w3  d25.             ;   find parent(child,console,
17910        am      0                ;                irrelevant);
17912        rl  w0  x1+c28           ;
17914        lx  w0  x2+a14           ;   children(console):=
17916        rs  w0  x1+c28           ;   children(console) exor id bit(child)
17918        jl. w3  d31.             ;   release core(child);
~~17920        al. w3  e40.             ;~~
~~17922        al  w1  0                ;   device:=0;~~
~~17924  i0: jd  1<11+14              ;   repeat~~
~~17926        al  w1  x1+1             ;   exclude user(receiver,device);~~
~~17928        sh. w1  (e4.)            ;   device:=device+1;~~
~~17930        jl.     i0. i21          ;   until device>max device~~
17932        jd  1<11+64              ;   remove process(receiver,result);
17934        jl.     (i20.)           ;
17936  ~~i4: 0~~                      ;
17938  e.                             ; end
17938
17938  ; procedure find work(state,work)
 /938  ; comment: searches a work area in a given state.
17938  ;      call:      return:
17938  ; w0             unchanged
17938  ; w1             work
17938  ; w2  state      state
17938  ; w3  link       link
17938
17938  b.i24                          ; begin
17938  w.d41:rl. w1  e13.             ;   for work:=first work
17940    i0: sn  w2  (x1+c50)         ;   step worksize
17942        jl      x3+0             ;   until forever do
17944        al  w1  x1+c2            ;   if state(work)=state
17946        jl.     i0.              ;   then goto found;
17948  e.                             ; found:
17948                                 ; end;
17948
17948
17948  ; procedure save work(state)
 17948 ; comment: saves a state and a number of variables in the
  948  ; current work area and proceeds to examine the event queue.
17948  ;      call:      return:
17948  ; w0             destroyed
17948  ; w1             work
17948  ; w2  state      destroyed
17948  ; w3  link       link
17948
```

} done by remove process anyway !!

```
17948  b.i24                        ; begin
17948  w.d42:rl. w1  e24.         ;   state(work):=state;
17950        ds  w3  x1+c51       ;   interrupt addr(work):=link:
17952        al. w2  e25.         ;
17954    i0: rl  w0  x2+0         ;
17956        rs  w0  x1+c52       ;   save(console)
17958        al  w1  x1+2         ;   to(core addr)
17960        al  w2  x2+2         ;   in(work):
  962        sh. w2  e30.         ;
17964        jl.     i0.          ;
17966        rl. w3  e2.
17968        al  w3  x3-1         ;   own buf:= own buf-1
17970        rs. w3  e2.
17972        jl.     g30.         ;   goto exam first;
17974  e.                         ; end
17974
17974  ; procedure restore work(work, state)
17974  ; comment: restores a number of variables from a work area
17974  ; and jumps to the interrupt address.
17974  ;      call:     return:
17974  ; w0             destroyed
17974  ; w1   work      work
17974  ; w2             state
17974  ; w3   link
17974
17974  b.i24                        ; begin
17974  w.d43:rs. w1  e24.         ;
  976        al. w2  e25.         ;
17978    i0: rl  w0  x1+c52       ;
17980        rs  w0  x2+0         ;   restore(console)
17982        al  w1  x1+2         ;   to(core addr)
17984        al  w2  x2+2         ;   from(work);
17986        sh. w2  e30.         ;
17988        jl.     i0.          ;
17990        rl. w1  e24.         ;   state:=state(work);
17992        al  w2  0            ;   state(work):=0;
17994        rx  w2  x1+c50       ;
17996        rl. w3  e2.          ;
17998        al  w3  x3+1         ;   own buf:= own buf+1
18000        rs. w3  e2.
18002        jl      (x1+c51)     ;   goto interrupt addr(work);
18004  e.                         ; end
18004
18004  ; procedure next bitnumbers(bits, type)
18004  ; comment: converts a sequence of integers from the console buffer
18004  ; and sets the corresponding bits in a word equal to one.
  004  ;      call:     return:
18004  ; w0             type
18004  ; w1             unchanged
18004  ; w2             bits
18004  ; w3   link      link
18004
18004  b.i24                        ; begin
18004  w.d45:rs. w3  i1.          ;
18006        al  w2  0            ;   bits:=0;
18008    i0: jl. w3  d2.          ; next bit;
18010        se  w0  2            ;   next param(type);
18012        jl.    (i1.)         ;   if type=2 then
18014        ac. w3 (e19.)        ;   begin
18016        al  w0  1            ;
18018        ls  w0  x3+23        ;   bits(23-integer):=1;
18020        lo  w2  0            ;   goto next bit;
18022        jl.     i0.          ;   end;
18024    i1: 0                    ;
18026  e.                         ; end
  026
18026  w.e0: c0     ; <first addr>
18028    e2: c4     ; <own buf>
18030    e3: c5     ; <own area>
18032    e4: 0      ; <max device>
18034    e5: h0     ; <char table>
18036    e6: h1     ; <param table>
```

```
18038    e7: h2       ; <first command>
18040    e8: h3       ; <last command>
18042    e9: h4       ; <first console>
18044    e10:h5       ; <last console>
18046    e11:h6       ; <first device>
18048    e12:h7       ; <last device>
18050    e13:h8       ; <first work>
18052    e14:h9       ; <last work>
  054    e15:h10      ; <core table>
18056    e16:h11      ; <first core>
18058    e17:b57      ; <top core>
18060    e18:0        ; <param type>
18062    e19:0        ; <integer>
18064    e20:0        ; <name>
18066    e21:0        ;
18068    e22:0        ;
18070    e23:0        ;
18072       0
18074    e24:0        ; <work>
18076    e25:0        ; <console>
18078    e26:0        ; <console buf> or <last addr>
18080    e27:0        ; <char shift>
18082    e28:0        ; <char addr>
18084    e29:0        ; <child>
18086    e30:0        ; <core addr>
18088    e31:0        ; <opbuf>
18090    e32:0,r.8    ; <message>
  106
18106    e39:0        ; <event>
18108    e40:0        ; <receiver>
18110    e41:0        ;
18112    e42:0        ;
18114    e43:0,0      ;
18118    e55:0        ; <write shift>
18120    e44:5<12     ; <type mess>
18122    e45:0        ; <line addr>
18124    e46:0        ; <write addr>
18126       0
18128    e47:3<12     ; <area mess> or <input mess>
18130    e48:0        ; <first addr>
18132    e49:0        ; <last addr>
18134    e50:0        ; <segment>
18136    e51:0        ; <entry tail> or <answer> or <message>
18138    e52:0        ;
18140    e53:0        ;
18142    e54:0        ; <convert area>
  144       0
18146    e56:0        ; <read shift> or <radix> or <start>
18148    e57:0        ; <read addr> or <state> or <size>
18150    e58:0        ; <save w1> or <first segment>
18152    e59:0        ; <save w2> or <content> or <keys> or <result>
18154    e60:0        ; <link> or <bytes to load>
18156    e61:0        ; <child w0>
18158    e62:0        ; <child w1>
18160    e63:0        ; <child w2>
18162    e64:0        ; <child w3>
18164    e65:0        ; <child ex>
18166    e66:0        ; <child ic>
18168    e67=e59+1    ; <ic in entry>
18168
18168    f0: <:system break<0>:>
18178    f1: <:ready<0>:>
18182    f2: <:syntax error<0>:>
18192    f3: <:not allowed<0>:>
18200    f4: <:no core<0>:>
  206    f5: <:no buffers<0>:>
18214    f6: <:no areas<0>:>
18220    f7: <:no internals<0>:>
18230    [redacted]
18238    f9: <:process unknown<0>:>
18250    f10:<:process exists<0>:>
18260    f11:<:catalog error<0>:>
```

```
18270     f12:<:area unknown<0>:>
18280     f13:<:area reserved<0>:>
18290     f14:<:program too big<0>:>
18302     f15:<:area error<0>:>
18310     f16:<:device unknown<0>:>
18320     f17:<:device reserved<0>:>
18332     f18:<:not inplemented<0>:>
18344     f23:<:operator:>,0,0
  354     f24:<:message<0>:>
18360     f25:<:pause<0>:>
18364     f26:<:max<0>:>
18368     ~~f27:<:date<0>:>~~
18372
18372                               ; end line:
18372     g0: am      f0-f1     ;
18374     ~~g1:~~ am      f1-f2      ;   text:=<:ready:>
18376     g2: am      f2-f3      ;      or <:syntax error:>
18378     g3: am      f3-f4      ;      or <:not allowed:>
18380     g4: am      f4-f5      ;      or <:no core:>
18382     g5: am      f5-f6      ;      or <:no buffers:>
18384     g6: am      f6-f7      ;      or <:no areas:>
18386     g7: am      f7-f8      ;      or <:no internals:>                    keep as is
18388     ~~g8: am      f8-f9      ;      or <:key trouble:>~~
18390     g9: am      f9-f10     ;      or <:process unknown:>
18392     g10:am      f10-f11    ;      or <:process exists:>
18394     g11:am      f11-f12    ;      or <:catalog error:>
18396     g12:am      f12-f13    ;      or <:area unknown:>
  398     g13:am      f13-f14    ;      or <:area reserved:>
18400     g14:am      f14-f15    ;      or <:program too big:>
18402     g15:am      f15-f16    ;      or <:area error:>
18404     g16:am      f16-f17    ;      or <:device unknown:>
18406     g17:am      f17-f18    ;      or <:device reserved:>
18408     g18:al. w1  f18.       ;      or <:not implemented:>;
18410         jl. w3  d19.       ;   init write;
18412     g25:jl. w3  d21.       ;   writetext(text);
18414         ~~al  w0  10~~         ;
18416         jl. w3  d20.-2     ;   writechar(10);
18418         jl. w3  d23.       ;   typeline(buf);
18420         jl. w3  d42.       ;   save work(buf);
18422     g1: rl. w1  e25.       ; end line ready:
18424         al  w2  -2
18426         la  w2  x1+c27
18428         rs  w2  x1+c27     ;   served(console):=false;
18430
18430     g30:al  w2  0          ; exam first:
18432         jl.     g32.       ;   event:=0;
 434      g31:rl. w2  e39.       ; exam next:
18436     g32:jd  1<11+24        ;   wait event(event,next,result);
18438         rs. w2  e39.       ;   event:=next;
18440         rl  w1  x2+6       ;   sender:=word(event+6);
18442         sn  w0  0          ;   if result=0 then
18444         jl.     g50.       ;   goto message;
18446         sn. w2  (e31.)     ; answer:
18448         jl.     g34.       ;   if event=opbuf then
18450         al. w1  e51.       ;   goto operator answer;
18452         jd  1<11+18        ;   wait answer(event,answer,
18454         rs. w0  e59.       ;                    result);
18456         jl. w3  d41.       ;   find work(event,old work);
18458         rs. w1  e24.       ;   work:= old work;
18460         jl. w3  d43.       ;   restore work(work,event);
18462
18462     g33:rl. w2  e39.       ; reject message:
18464         jd  1<11+26        ;   get event(event);
18466         al  w0  2          ;
 8468         al. w1  e51.       ;
 470          jd  1<11+22        ;   send answer(event,answer,2);
18472         jl.     g30.       ;   goto exam first;
18474
18474     g34:rl. w0  e2.        ; operator answer:
18476         sh  w0  0          ;   if own buf<=0 then
18478         jl.     g31.       ;   goto exam next;
18480         rl  w1  x2+10      ;   proc:= word(buf+10);
```

```
18482        dl   w0   x1+4
18484        ds.  w0   e41.       ;   receiver:= name(proc);
18486        dl   w0   x1+8
18488        ds.  w0   e43.
18490        rl   w0   x1+a50
18492        ls   w0   -6         ;   device:= device no(proc)/64;
18494        jl.  w3   d24.       ;   find console(device,new console,
18496        jl.       g33.       ;                 reject message);
  498        bz   w3   x1+c27
18500        sz   w3   2.1        ;   if served(new console) then
18502        jl.       g31.       ;   goto exam next;
18504        rs.  w1   e25.       ;   console:= new console;
18506        al   w3   x3+1
18508        hs   w3   x1+c27     ;   served(console):= true;
18510        jd   1<11+26         ;   get event(buf);
18512        al.  w1   e51.
18514        al.  w3   f23.
18516        jd   1<11+16         ;   send mess(<:operator:>,buf);
18518        rs.  w2   e31.       ;   opbuf:= buf;
18520        al   w2   0
18522        jl.  w3   d41.       ;   find work(0,new work);
18524        rs.  w1   e24.       ;   work:= new work;
18526        al   w2   x1+c66     ;   first addr:= work+linebuf;
18528        al   w3   x1+c67     ;   last addr:= work+outputlinebuf-2;
18530        ds.  w3   e49.
18532        rs.  w2   e28.       ;   char addr:= first addr;
18534        al.  w3   e40.
  536        al.  w1   e47.
18538        jd   1<11+16         ;   send mess(receiver,buf,input mess);
18540        jl.  w3   d42.       ;   save work(buf);
18542        ~~al   w2   x1+c66-2~~
18544        ~~wa.  w2   e52.       ;   last addr:= char addr+bytes-2;~~
18546        al   w3   -16
18548        ds.  w3   e27.       ;   char shift:= -16;
18550                             ; next command:
18550    g35:jl.  w3   d2.        ;   next param(type);
18552    g36:sn   w0   0          ; exam command:
18554        jl.       g1.        ;   if type=0
18556        se   w0   1          ;   or type<>1
18558        jl.       g2.        ;   then goto end line;
18560        dl.  w2   e21.       ;
18562        rl.  w3   e7.        ;   addr:=first command;
18564    g37:sn   w1   (x3+0)     ;   repeat
18566        jl.       g39.       ;   if word(addr)=name(0:23)
18568    g38:sn.  w3   (e8.)      ;   and word(addr+2)=name(24:47)
18570        jl.       g2.  g18   ;   then goto found;
 572        al   w3   x3+6       ;   addr:=addr+6;
18574        jl.       g37.       ;   until addr=last command;
18576    g39:se   w2   (x3+2)     ;   goto end line;
18578        jl.       g38.       ;
18580        rl.  w1   e25.       ; found:
18582        ~~bz   w0   x1+c27~~     ;
18584        ~~bz   w0   e43~~        ;   if not privileged(console)
18586        ~~jl.       g40.~~
18588        ~~bz   w2   x3+4       ;   and not allowed(console)~~
18590        ~~so   w0   x2+0       ;~~
18592        ~~jl.       g3.        ;   then goto end line;~~
18594    g40:bz   w3   x3+5       ;
18596    g45:jl.       x3+0       ;
18598        ; w0=command mask(console)   w1=console
~~18598~~
18598    g50:rl.  w0   e2.        ; message:
18600        sh   w0   0          ;   if own buf<=0 then
18602        jl.       g31.       ;   goto exam next;
18604        sh   w1   -1         ;   if sender<0 then
 606        jl.       g33.       ;   goto reject message;
18608        dl   w0   x2+10
18610        ~~ds.  w0   e32.+2     ;   move message from buffer to~~ <message>;
18612        dl   w0   x2+14
18614        ds.  w0   e32.+6
18616        dl   w0   x2+18
18618        ds.  w0   e32.+10
```

if bytes = 0 then
begin last addr:= last addr +2;
word (last addr):=
10 shift 16
end,

dl w0 6
sn w0 6
jl. g33.

if status message then goto reject message

```
18620          dl  w0  x2+22
18622          ds. w0  e32.+14
18624          al  w2  x1+0
18626          jl. w3  d25.       ;   find parent(sender,parent,
18628          jl.     g33.       ;                      reject message);
18630          rs. w1  e25.       ;   console:= parent;
18632          rs. w2  e29.       ;   child:= sender;
18634     g41:al  w2  0
  636          jl. w3  d41.       ;   find work(0,new work);
18638          rs. w1  e24.       ;   work:= new work;
18640          jl. w3  d19.       ;   init write;
18642          rl. w3  e32.       ;   if message(0)(23)=1 then
18644          so  w3  2.1        ;     begin stop child;
18646          am      d33-d39    ;       writetext(<:pause:>)
18648          jl. w3  d39.       ;     end
18650          se. w3  0          ;   else
18652          am      f25-f24    ;     begin child name;
18654          al. w1  f24.       ;       writetext(<:message:>)
18656          jl. w3  d21.       ;     end;
18658          rl. w2  e39.
18660          jd  1<11+26        ;   get event(event);
18662          al  w0  1
18664          al. w1  e32.
18666          jd  1<11+22        ;   send answer(event,message,1);
18668          al. w1  e40.
18670          jl. w3  d21.       ;   writetext(receiver);
18672          al. w2  e32.+2     ;   index:= 2;
  674     g43:rl  w1  x2+0        ; next word:
18676          bl. w3  e32.+1     ;   word:= message(index);
18678          ls  w3  1          ;   bits:= message(1);
18680          hs. w3  e32.+1     ;   message(1):= bits shift 1;
18682          sh  w3  -1         ;   if bits(0)=1 then
18684          jl.     g44.       ;   goto number;
18686          sn  w1  0          ;   if word=0 then
18688          jl.     g42.       ;   goto test more;
18690          al  w0  0          ;   char:= word(0:7);
18692          ld  w1  8          ;   word:= word shift 8;
18694          jl. w3  d20.       ;   writechar(char);
18696          al  w0  0          ;   char:= word(0:7);
18698          ld  w1  8          ;   word:= word shift 8;
18700          jl. w3  d20.       ;   writechar(char);
18702          al  w0  0          ;   char:= word(0:7);
18704          ld  w1  8          ;   word:= word shift 8;
18706          am      d20-d22    ;   writechar(char);
18708                             ;   goto test more;
18708                             ; number:
  708                             ;   writeinteger(word);
18708     g44:jl. w3  d22.        ; test more:
18710     g42:al  w2  x2+2        ;   index:= index+2;
18712          sh. w2  e32.+14    ;   if index<=14 then
18714          jl.     g43.       ;   goto next word;
18716          ~~al  w0  10~~
18718          jl. w3  d20.-2     ;   writechar(10);
18720          jl. w3  d23.       ;   typeline(buf);
18722          jl. w3  d42.       ;   save work(buf);
18724          jl.     g30.       ;   goto exam first;
18726
18726    b.i24                    ; new:
18726    w.g51:la. w0  i0.        ;   abs addr(console):=
18728          wa. w0  i1.        ;   abs protection(console):=false:
18730          rs  w0  x1+c26     ;   keys(console):=standard keys:
18732          ~~hs  w0  x1+c37~~     ;   ~~pr(console):=illegal pr;~~
18734          dl. w3  i2.        ;   buf claim(console):=standard buf:
18736          ds  w3  x1+c34     ;   area claim(console):=standard area:
18738          dl. w3  i3.        ;   internal claim(console):=standard int:
  740          rs  w2  x1+c36     ;   func mask(console):=standard func:
18742     i10:rs  w3  x1+c39      ; set: cat mask(console):=standard cat:
18744          dl. w3  i4.        ;   size(console):=standard size:
18746          ds  w3  x1+c40+2   ;
18748          dl. w3  i5.        ;
18750          ds  w3  x1+c40+6   ;   prog(console):=standard prog:
18752          rl  w2  x1+c26     ;
```

⊗ new stuff.
if operation.mess
=1 then stop
remove
goto reject
mess
0

remove
if op=2

BL WO X1
SN WO 2
JL. w3 d40.

```
18754        sz  w2  1<2        ;
18756        jl.     g35.       ;
18758        jl.     g52.       ;   goto process;
18760    i0: 8.47771            ;
18762    i1: c6<12              ; standard keys:
18764        c7<12+c8           ; standard buf and area:
18766    i2: c9<12+c10          ; standard int and func:
18768        c11                ; standard cat:
  770    i3: c12                ; standard size:
18772    i4=k+2, i5=k+6         ; standard prog:
18772        <:fp:>,0,0,0       ;
18780
18780  w.g81:la. w0  i0.        ; ~~claim:~~ job
18782        oa. w0  i0.        ;   abs addr(console):= false;
18784        rs  w0  x1+c26     ;   abs protection (console):= true;
18786        jl. w3  d15.       ;
18788        al. w3  e20.       ;   next name;
18790        al. w1  e51.       ;   lookup entry (name,tail,result);
18792        jd    1<11+42      ;
18794        sn  w0    0        ;   if result <> 0
18796        se  w0 (x1)        ;   or tail (0) <> 0
18798        jl.     g9.        ;   then goto end line (proc unknown);
18800        rl. w1  e25.       ;
18802        dl. w3  e51.+4     ;
18804        ds  w3  x1+c29+2   ;   proc name (console):= tail (2:8);
18806        dl. w3  e51.+8     ;   set the rest of the console params
18808        ds  w3  x1+c29+6   ;   from the entry tail;
  810        dl. w3  e51.+14    ;   size:  tail+10;
18812        ds  w3  x1+c34     ;   param for create internal: tail 12:18;
18814        dl. w3  e51.+18    ;
18816        ds  w3  x1+c37     ;   goto next command;
18818        rl. w3  e51.+10    ;   comment: via  set in new command;
18820        jl.     i10.       ;
18822  e.
18822
18822    g52:am      c29-c40    ; process:
18824    g53:al  w1  x1+c40     ; program:
18826        jl. w3  d15.       ;   next name;
18828        dl. w3  e21.       ;
18830        ds  w3  x1+2       ;
18832        dl. w3  e23.       ;
18834        ds  w3  x1+6       ;   name(console):=name;
18836        jl.     g35.       ;   goto next command;
18838
18838  [illegible]
18838  ~~w.g54:la. w0~~ [illegible] ~~; address:~~
  840        ~~hs  w0  x1+c27~~     ;   abs addr(console):=true;
18842        ~~am      c30-c39~~    ;
18844  W.g56:al  w2  x1+c39     ; size:
18846        jl. w3  d16.       ;   next integer(integer);
18848        sz  w0  2.1
18850        bs. w0  1          ;   integer(23):= 0;
18852        rs  w0  x2+0       ;   word param(console):=integer;
18854        jl.     g35.       ;   goto next command;
18856    ~~i0: 1<1~~
18858  ~~e.~~
18858
18858  ~~b.i24~~                    ; catalog:
18858  w.g55:jl. w3  d45.       ;   next bitnumbers(bits, type);
18860        ~~la. w2  i0.        ;   bits(23):=0;~~
18862        rs  w2  x1+c36     ;   catalog(console):=bits;
18864        jl.     g36.       ;   goto exam command;
18866    ~~i0: 8.7777 7776~~
18868  [illegible]
18868
 .868  b.i24
18868  w.g57:al  w2  x1+c26     ; key claim:
18870        la. w0  i2.        ;   abs protection(console):=false;
18872        jl.     i0.        ;   goto set param;
18874    g59:al  w2  x1+c38     ; pk:
18876        lo. w0  i3.        ;   abs protection(console):=true;
18878    i0: hs  w0  x1+c27     ; set param:
```

```
~~[illegible]~~
   g60:am        g32-c33    ; buffer claim:
   g61:am        c33-c34    ; area claim:
   g62:al   w2   x1+c34     ; internal claim:
        jl. w3   d16.       ;   next integer(integer);
        hs  w0   x2+0       ;   byte param(console):=integer;
        jl.      g35.       ;   goto next command;
~~   i2: 8.[illegible]~~
~~   i3: 1[illegible]2~~
~~e.~~

~~b.i24                      ; pr:~~
~~w.g58:jl. w3   d45.       ;   next bitnumbers(bits, type);~~
~~        ls  w2   -16        ;   bits:=bits shift -16;~~
~~        lx. w2   i0.        ;   bits:=bits exor 8.377;~~
~~        lo. w2   i1.        ;   bits(16):=1;~~
~~        hs  w2   x1+c37     ;   pr(console):=bits(12:23);~~
~~        jl.      g36.       ;   goto exam command;~~
~~   i0: 8.377~~
~~   i1: 1<7~~
~~e.~~

                            ; function mask:
   g63:jl. w3   d45.        ;   next bitnumbers(bits, type);
        ls  w2   -12         ;
        hs  w2   x1+c35      ;   function mask(console):=bits(0:11);
        jl.      g36.        ;   goto exam command;

~~   g64:                      ; create:~~
~~w.      jl. w3   d35.       ;~~
~~        rl. w2   e29.       ;   create child;~~
~~        rl  w0   x2+a17     ;~~
~~        jl. w3   d36.       ;   modify child(first addr(child));~~
~~        jl.      g35.       ;   goto next command;~~
~~~~
~~                            ; init:~~
~~   g65:jl. w3   d35.        ;   create child;~~
~~        jl. w3   d37.       ;   load child;~~
~~        jl.      g35.       ;   goto next command;~~

                            ; run:
   g66:jl. w3   d35.        ;   create child;
        jl. w3   d37.       ;   load child;
   g67:jl. w3   d38.        ;   start child;
        jl.      g35.       ;   goto next command;

~~                            ; load:~~
~~   g67:jl. w3   d34.        ;   check child;~~
~~        jl. w3   d37.       ;   load child;~~
~~        jl.      g35.       ;   goto next command;~~

                            ; start:
   g68:jl. w3   d34.        ;   check child;
~~        jl. w3   d38.~~       ;   start child;
        jl.      ~~g35.~~ g67.  ;   goto next command;

                            ; stop:
   g69:jl. w3   d34.        ;   check child;
        jl. w3   d39.       ;   stop child;
        jl.      g35.       ;   goto next command;

   ~~[illegible]~~                  ; break:
~~[illegible]~~               ; if break option then
w.g70:jl. w3   d34.         ; begin check child;
        jl. w3   d39.       ;   stop child;
        rl. w2   e29.       ;
        rl  w3   x2+a27     ;   addr:=interrupt addr(child);
        sn  w3   0          ;   if addr<>0 then
        jl.      g35.       ;   begin
        dl  w1   x2+a29     ;   word(addr):=save w0(child);
        ds  w1   x3+2       ;   word(addr+2):=save w1(child);
        dl  w1   x2+a31     ;   word(addr+4):=save w2(child);
```

18880

```
; ~~slice command~~ slice command
b. i24
w.g77:  jl. w3 d34.   ; check child
        jl. w3 d2.    ; next param (type);
        se  w0 2      ; if type<>2
        jl.    g36.   ; then goto exam command
        wm. w0 i0.    ; max slice child :=
        ~~rs w0~~     ; 10000 * param;
        rl. w3 e29.   ;
        rs  w0 x3+a24 ;
        jl.    g35.   ; goto next command;
i0:     10000         ;
e.
```

```
18982         ds  w1   x3+6      ;   word(addr+6):=save w3(child);
18984         dl  w1   x2+a33    ;   word(addr+8):=save ex(child);
18986         ds  w1   x3+10     ;   word(addr+10):=save ic(child);
18988         al  w1   8         ;   word(addr+12):=8;
18990         rs  w1   x3+12     ;
18992         al  w0   x3+14     ;   modify child(addr+14);
18994         jl. w3   d36.      ;   start child;
18996         jl. w3   d38.      ;   end;
  998         jl.      g35.      ;   goto next command;
19000    ~~[illegible]~~         ; end else goto end line;
19002
19002                            ; remove:
19002      g71:jl. w3   d34.      ;   check child;
19004         jl. w3   d39.      ;   stop child;
19006         jl. w3   d40.      ;   remove child;
19008         jl.      g35.      ;   goto next command;
19010
19010      g82:am       2         ; reset:
19012      g72:am       2         ; include:
19014      ~~[illegible]~~        ; exclude:
19014    ~~[illegible]~~         ; if include/exclude option then
19014    b.i24                    ; begin
19014    w.g73:rl. w3   i2.       ;
19016         rs. w3   i1.       ;
19018         se. w3  (i3.)      ;   if not reset then
19020         jl. w3   d34.      ;   check child;
19022      i0: jl. w3   d2.       ; more:
  024         se  w0   2         ;   next param(type);
19026         jl.      g36.      ;   if type<>2
19028         rl. w1   e25.      ;   then goto exam command;
19030         al  w3   x1+c29    ;
19032         rl. w1   e19.      ;   include/exclude(name(console),
19034      i1: am       0         ;       integer,result);
19036         se  w0   0         ;   if result=0
19038         jl.      g16.      ;   then goto more
19040         jl.      i0.       ;   else goto end line;
19042      i2: jd   1<11+14      ;
19044         jd   1<11+12      ;
19046      i3: jd     1<11+2     ;
19048    e. [illegible] ~~[illegible]~~  ; end else goto end line;
19050
19050      ~~[illegible]~~        ; call:
19050    ~~[illegible]~~         ; if call option then
19050    ~~[illegible]~~         ; begin
19050    w.~~[illegible]~~ jl. w3   d2.   ; more: next param(type);
19052         se  w0   2         ;   if type<>2
  054    ~~[illegible]~~ jl.      g36.      ;   then goto exam command;
19056         rl. w1   e19.      ;   device:=integer;
19058         jl. w3   d15.      ;   next name;
19060  g74:   al. w3   e20.      ;   create peripheral process(
19062         jd   1<11+54      ;   name,device,result);
19064         sn  w0   3         ;   if result=3
19066         jl.      g10.      ;
19068         sn  w0   4         ;   or result=4
19070         jl.      g16.      ;
19072         sn  w0   5         ;   or result=5
19074         jl.      g17.      ;   then goto end line
19076         jl.      [illegible] g74.  ;   else goto more;
19078    [illegible] ~~[illegible]~~   ; end else goto end line;
19080
19080      ~~[illegible]~~        ; list:
19080    ~~[illegible]~~         ; if list option then
19080    b.i24                    ; begin
19080    w.g75:rl. w3   e15.      ;   core addr:=core table;
19082      i0: rl  w2   x3+0      ;   while word(core addr)<>0 do
  084         rs. w2   e29.      ;   begin
19086         sn  w2   0         ;   child:=word(core addr);
19088         jl.      g35.      ;
19090         al  w3   x3+2      ;
19092         rs. w3   e30.      ;   core addr:=core addr+2;
19094         jl. w3   d33.      ;   child name;
19096         jl. w3   d19.      ;   init write;
```

```
19098          al. w1   e40.        ;
19100          jl. w3   d21.        ;   writetext(receiver);
19102          rl  w1   x2+a17      ;
19104          jl. w3   d22.        ;   writeinteger(first addr(child));
19106          ~~al  w0   32~~        ;
19108          ~~jl. w3   d20.~~-4    ;   writechar(32);
19110          rl  w1   x2+a18      ;
19112          ws  w1   x2+a17      ;   writeinteger(top addr(child)
  ,14          jl. w3   d22.        ;         -first addr(child)):
19116          ~~al  w0   32~~        ;
19118          ~~jl. w3   d20.~~-4    ;   writechar(32);
19120          ~~rl~~  w1   x2+a~~[illegible]~~-41 ;
19122          jl. w3   d22.        ;   writeinteger(~~bk~~creation number(child));
19124          ~~al  w0   10~~        ;
19126          ~~jl. w3   d20.~~-2 ok ;   writechar(10);
19128 rl       jl. w3   d23.        ;   typeline(buf);
19130          jl. w3   d42.        ;   save work(buf);
19132          rl. w3   e30.        ;   end;
19134          jl.      i0.         ;   goto next command;
19136    e.    ~~jl.      [illegible]~~ ; end else goto end line;
19138
19138    ~~z7~~                     ; max:
19138    ~~c.(:c23>16a.1:)-1~~      ; if max option then
19138    b.i24                      ; begin
19138    w.g76:jl. w3   d19.        ;   initwrite;
19140          al. w1   f26.        ;
19142          jl. w3   d21.        ;   writetext(<:max:>);
  144          jl. w3   d29.        ;   find max(size);
19146          jl. w3   d22.        ;   writeinteger(size);
19148          ~~al  w0   32~~        ;
19150          ~~jl. w3   d20.~~-4    ;   writechar(32);
19152          rl  w2   b1          ;
19154          bz  w1   x2+a19      ;
19156          ws. w1   e2.         ;   writeinteger(buf claim(s)
19158          jl. w3   d22.        ;              -own buf);
19160          ~~al  w0   32~~        ;
19162          ~~jl. w3   d20.~~-4    ;   writechar(32);
19164          bz  w1   x2+a20      ;
19166          ws. w1   e3.         ;   writeinteger(area claim(s)
19168          jl. w3   d22.        ;              -own area);
19170          ~~al  w0   32~~        ;
19172          ~~jl. w3   d20.~~-4    ;   writechar(32);
19174          bz  w1   x2+a21      ;
19176          jl. w3   d22.        ;   writeinteger(internal claim(s));
19178          ~~al  w0   32~~        ;
19180          jl. w3   d20.-4      ;   writechar(32);
  182          al  w2   8           ;   keys:=8;
19184          jl. w3   d32.        ;   find keys(keys,pr,pk,
19186          jl.      i0.         ;           typekeys);
19188          am       0           ;
19190     i0:  ac  w1   x2-8        ; typekeys:
5000 192       jl. w3   d22.        ;   writeinteger(8-keys);
19194          ~~al  w0   10~~        ;
19196          jl. w3   d20.-2      ;   writechar(10);
19198          jl. w3   d23.        ;   typeline(buf);
19200          jl. w3   d42.        ;   save work(buf);
19202          jl.      g35.        ;   goto next command;
19204    e.    ~~jl.      g18.~~      ; end else goto end line;
19206    ~~z7~~
19206
19206    c.(:c23>17a.1:)-1          ; if date option then
19206    b.i34                      ; begin
19206    w.    jd  1<11+36          ; date:
19208          wd. w1   i20.        ;   get clock(clock);
19210          al  w3   0           ;   fourmin:=clock/2400000;
  212          wd. w0   i19.        ;   clock:=clock mod 2400000;
19214          hs. w0   i29.        ;   min:=clock/600000;
19216          al  w2   0           ;   clock:=clock mod 600000;
19218          wd. w3   i18.        ;
19220          hs. w3   i30.        ;   sec:=clock/10000;
19222          al  w0   0           ;   days:=fourmin/360;
19224          wd. w1   i15.        ;   fourmin:=fourmin mod 360;
```

```
19226          al   w3   0          ;
19228          wd.  w0   i12.       ;    hour:=fourmin/15;
19230          hs.  w0   i28.       ;    fourmin:=fourmin mod 15;
19232          as   w3   2          ;
19234          ba.  w3   i29.       ;
19236          hs.  w3   i29.       ;    min:=fourmin*4+min;
19238          al   w0   0          ;
19240          wd.  w1   i17.       ;
  242          as   w1   2          ;    year:=days/1461*4+68;
19244          al   w1   x1+68      ;    days:=days mod 1461;
19246          se   w0   59         ;    if days=59 then
19248          jl.       i0.        ;    begin
19250          al   w2   2          ;    month:=2;
19252          al   w3   29         ;    day:=29;
19254          jl.       i2.        ;    end else
19256     i0:  sl   w0   60         ;    begin
19258          bs.  w0   1          ;    if days>59 then days:=days-1;
19260          al   w3   0          ;
19262          wd.  w0   i16.       ;    year:=year+days/365;
19264          wa   w1   0          ;    days:=days mod 365;
19266          al   w2   13         ;    month:=13;
19268     i1:  al   w2   x2-1       ;    repeat
19270          bz.  w0   x2+i10.    ;    month:=month-1
19272          am        (0)        ;    until
19274          sh   w3   -1         ;    month table(month)<=days;
19276          jl.       i1.        ;
19278          ws   w3   0          ;    day:=days-month table(month)+1;
  280          al   w3   x3+1       ;    end;
19282     i2:  hs.  w1   i27.       ;
19284          hs.  w2   i26.       ;
19286          hs.  w3   i25.       ;
19288          am        -500       ;    comment: byte value on d19;
19290          jl.  w3   d19.+500   ;    initwrite;
19292          al.  w1   f27.       ;
19294          am        -500       ;
19296          jl.  w3   d21.+500   ;    writetext(<:date:>);
19298          al.  w2   i25.       ;    addr:=day;
19300     i3:  bz   w1   x2+0       ;    repeat
19302          am        -500       ;
19304          jl.  w3   d22.+500   ;    writeinteger(byte(addr));
19306          bz   w0   x2+1       ;
19308          am        -500       ;
19310          jl.  w3   d20.+500   ;    writechar(byte(addr+1));
19312          al   w2   x2+2       ;    addr:=addr+2;
19314          sh.  w2   i30.       ;    until addr>sec;
19316          jl.       i3.        ;
  318          jl.  w3   d23.       ;    typeline(buf);
19320          jl.  w3   d42.       ;    save work(buf);
19322          jl.       g35.       ;    goto next command;
19324     z.
19324     ~~g78.~~
19324     ~~e,(e23>47a.(.)+1~~
19324
19324   g78:  al.  w2   i25.       ;  newdate:
19326     i6:  am        d16-e0     ;    addr:= day;
19328          jl.  w3   e0.
19330          hs   w0   x2+0       ;    repeat
19332          al   w2   x2+2       ;    next integer(integer);
19334          sh.  w2   i30.       ;    byte(addr):=integer;
19336          jl.       i6.        ;    addr:=addr+2;
19338          bz.  w1   i27.       ;    until addr>sec;
19340          bz.  w2   i26.       ;    if month<3 then
19342          sl   w2   3          ;    begin
19344          jl.       i7.        ;    year:=year-1;
19346          al   w1   x1-1       ;    month:=month+12;
  348          al   w2   x2+12      ;    end;
19350     i7:  al   w1   x1-68      ;
19352          wm.  w1   i17.       ;
19354          as   w1   -2         ;    days:=(year-68)*1461/4+
19356          ba.  w1   x2+i10.    ;          month table(month)+
19358          ba.  w1   i25.       ;          day;
19360          wm.  w1   i13.       ;
```

✓

sl w0 1900
bs. w0 -1

```
19362         ba. w1  i28.        ;    hours:=days*24+hour;
19364         wm. w1  i14.        ;
19366         al  w2  0           ;
19368         bz. w3  i29.        ;
19370         aa  w1  6           ;    min:=hours*60+min;
19372         wd. w1  i11.        ;    fourmin:=min/4;
19374         wm. w0  i14.        ;    min:=min mod 4;
19376         ba. w0  i30.        ;
  378         wm. w0  i18.        ;    msec:=(min*60+sec)*10000;
19380         al  w2  0           ;
19382         rl  w3  0           ;
19384         wm. w1  i20.        ;
19386         ea  w1  6           ;    clock:=fourmin*2400000+msec;
19388         jd  1<11+38         ;    set clock(clock);
19390         jl.     g35.        ;    goto next command;
19392
19392   ; month table:
19392   ; comment. contains one byte for each month of the year defining
19392   ; the number of days elapsed from january 1st until the first of
19392   ; the present month;
19392   h.i10=k-1, 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334, 365, 3
19406
19406   ; constants:
19406   w. i11: 4
19408      [redacted]
19410      i13: 24
19412      i14: 60
  414      [redacted]
19416
19418      i17: 1461
19420      i18: 10000
19422      [redacted]
19424      i20: 2400000
19426
19426   ; working locations:
19426      i25: 46    ; day:
19428      i26: 46    ; month:
19430      i27: 32    ; year
19432      i28: 44    ; hour
19434      i29: 44    ; min:
19436      i30: 10    ; sec:
19438
19438   e.[redacted]                 ; end else goto end line;
19440
19440   ~~b.i24                      ; dump:~~
19440   w.g79:am      d15-e0
  442         jl. w3  e0.         ;    next name;
19444         jl. w3  d34.        ;    check child;
19446         al. w3  e20.
19448         jd  1<11+52         ;    create area process(name,result);
19450         sn  w0  2           ;    if result=2 then
19452         jl.     g11.        ;    goto end line;
19454         sl  w0  2           ;    if result>=2 then
19456         jl.     g12.        ;    goto end line;
19458         jd  1<11+8          ;    reserve process(name,result);
19460         se  w0  0           ;    if result<>0 then
19462         jl.     i0.         ;    goto give up;
19464         jl. w3  d39.        ;    stop child;
19466         rl. w2  e29.
19468         dl  w3  x2+a18
19470         al  w3  x3-2        ;    line addr:= first addr(child);
19472         ds. w3  e46.        ;    write addr:= top addr(child)-2:
19474         al. w3  e20.
19476         al. w1  e44.
 9478         jd  1<11+16         ;    send mess(name,output,buf):
  480         al. w1  e51.
19482         jd  1<11+18         ;    wait answer(buf,answer,result):
19484         rl. w1  e51.
19486         sn  w0  1           ;    if result<>1
19488         se  w1  0           ;    or status(answer)<>0 then
19490                             ;    goto give up;
19490         am      g15-g35     ;    remove process(name,result);
```

31 28 31 30 31

30 31 31

?

396 check and see what it should be!

365
+ 31
396

```
19492         am      g35-g13    :   goto next command;
19494    i0: al. w2   g13.       : give up:
19496         jd  1<11+64        :   remove process(name,result);
19498         jl        x2+0     :   goto end line;
19500  e.
19500  ~~g10:~~                  ~~: replace:~~
19500  ~~c.(:c23>16a.1:)-1~~     ~~: if replace option then~~
19500  b.i24                     : begin
  500  a.     rl.w0(e15.)        :
19502         se w0 0            :   if word(core table)<>0 then
19504         jl.    g10.        :     goto end line;
19506         am     d15-e0      :
19508         jl.w3 e0.          :   next name;
19510         al.w1 e51.         :
19512         rl w3 66           : next buffer:
19514    i0:  al w2 0            :   buf:=0;
19516         jd     1<11+24     :   wait event(buf);
19518         jd     1<11+26     :   get event(buf);
19520         ba.w0 1            :   result:=result+1;
19522         sn w0 1            :   if result=1 then
19524         jd     1<11+22     :     send answer(buf,answer,result);
19526         rl w0 x3+a15       :   next:=word(event q(proc));
19528         se w0 x3+a15       :   if next<>event q(proc) then
19530         jl.    i0.         :     goto next buffer;
19532         al.w3 e20.         :
19534         jd     1<11+42     :   lookup entry(name,tail,result):
19536         se w0 0            :   if result<>0 then
  538         jl.    i4.         :     goto give up;
19540         bz.w0 e59.         :
19542         se w0 8            :   if content<>8 then
19544         jl.    i4.         :     goto give up;
19546                                ;
19546         rl w0 x1           :   if modekind >= 0
19548         sl w0 0            :   then goto create;
19550         jl.    i2.         :
19552         dl w0 x1+4         :   name:= tail (2:8);
19554         ds. w0 e20.+2      :
19556         dl w0 x1+8         :
19558         ds.w0 e20.+6       :   lookup entry (name,tail,result);
19560         al.w3 e20.         :   if result <> 0 then
19562         jd  1<11+42        :   goto give up;
19564         se w0 0            :
19566         jl.    i4.         :
19568    i2:  jd  1<11+52        : create: create area proc(name,result);
19570         se w0 0            :   if result <> 0 then goto give up:
19572         jl.    i4.         :
  574
19574         rl w2 66           :   proc:= word(66);
19576         dl w0 x1+4         :
19578         se w3 0            :   if tail(2) <> 0 then
19580         ds w0 x2+a11+2     :   name.proc:= tail(2:8);
19582         dl w0 x1+8         :
19584         ds w0 x2+a11+6     :
19586         rl.w3(e17.)        :
19588         rs w3 x2+a18       :   top addr(proc):= top core;
19590         rl.w1 e60.         :
19592         al w1 x1+511       :
19594         ls w1 -9           :   load size:=
19596         ls w1 9            :       (bytes(tail)+511)/512*512;
19598         wa.w1 e0.          :   last addr(area mess):=
19600         al w1 x1-2         :       first addr+load size-2;
19602         rl.w0 e0.          :
19604         ds.w1 e49.         :   first addr(area mess):= first addr;
19606         rl.w1 e58.         :   segment(area mess):=
19608         rs.w1 e50.         :       segment(tail);
  610         bz.w1 e67.         :
19612         wa w1 0            :
19614         rs.w1 i20.         :   entry:= first addr+entry(tail);
19616         sl w1(6)           :   if entry>=top core then
19618         jl.    i5.         :     goto give up;
19620         al w1 x3+0         :
19622         ws.w1 e16.         :   to:= top core;
```

```
19624           rs.w1  i21.        ;    length:= top core-first core;
19626           wa.w1  e0.         ;    last:= first addr+length;
19628           sh.w1(e49.)        ;    if last<=last addr(area mess) then
19630           jl.    i6.         ;      goto give up;
19632           rl.w2  e16.        ;    from:= first core;
19634    i10:   al w3  x3-2        ; move:
19636           al w2  x2-2        ;    to:= to-2;
19638           rl w0  x2+0        ;    from:= from-2;
  640           rs w0  x3+0        ;    word(to):= word(from);
19642           sl w3  x1+0        ;    if to>=last then
19644           jl.    i10.        ;      goto move;
19646           rl.w1  i21.        ;
19648           jl.    x1+2        ;    jump to moved code;
19650           al.w1  e47.        ;
19652           al.w3  e20.        ;
19654           jd     1<11+16     ;    send mess(name,area mess,buf);
19656           al.w1  e51.        ;
19658           jd     1<11+18     ;    wait answer(buf,answer,result);
19660           rl.w1  e51.        ;
19662           sn w0  1           ;    if result=1
19664           se w1  0           ;        and status(answer)=0 then
19666           jl.    i11.        ;      return:= ok
19668           jd     1<11+64     ;    remove process(name,result);
19670           rl.w0  i22.        ;
19672           rs.w0  g30.        ;
19674           jl.    g1.         ;
19676    i11:   rl.w0  i23.        ;    else return:= sorry;
  678           rs.w0  g30.        ;
19680           jl.    g15.        ;
19682    i12:   rl.w1  e24.        ; ok:
19684           rl w2  x1+c50      ;    buf:= state(work);
19686           jd     1<11+18     ;    wait answer(buf,work,result);
19688           ld w1  -65         ;    w0:= w1:= 0;
19690           rl.w2  e25.        ;    w2:= console;
19692           rl w3  66          ;    w3:= current process;
19694           xl.    0           ;    ex:= 0;
19696           jl.   (i20.)       ;    goto entry;
19698
19698    i13:                      ; sorry:
19698           jd.    0           ;    wait forever in disabled mode;
19700
19700    i4:    am     -4
19702    i5:
19702    i6:    al.w2  g14.        ; give up:
19704           al.w3  e20.        ;
19706           jd     1<11+64     ;    remove process(name,result);
. 708           jl.    x2+0        ;    goto end line;
19710    i20:   0                  ; entry
19712    i21:   0                  ; length
19714    i22:   jl.    i12-g30     ; return to ok
19716    i23:   jl.    i13-g30     ; return to sorry
19718  ~~e.z.   jl.    g18.~~      ; end else goto end line;
19720
19720  ; character table:
19720  ; contains an entry of 3 bits defining the type of each
19720  ; character in the iso 7 bit character set.
19720                3
19720  w.h0: 8.7777 7777           ; nul soh stx etx eot enq ack bel
19722        8.7057 7777           ; bs  ht  nl  vt  ff  cr  so  si
19724        8.7777 7777           ; dle dc1 dc2 dc3 dc4 nak syn etb
19726        8.7567 7777     5     ; can em  sub esc fs  gs  rs  us
19728        8.3666 6666           ; sp
19730        8.6666 4644           ; (   )   *   +   ,   -   .   /
19732        8.1111 1111           ; 0   1   2   3   4   5   6   7
19734        8.1125 6466           ; 8   9   :   ;   <   =   >
. .736       8.6666 6566           ;     a   b   c   d   e   f   g
19738        8.6666 6666           ; h   i   j   k   l   m   n   o
19740        8.6666 6666           ; p   q   r   s   t   u   v   w
19742        8.6646 6667           ; x   y   z   æ   ø           _
19744        8.6000 0000           ;     a   b   c   d   e   f   g
19746        8.0000 0000           ; h   i   j   k   l   m   n   o
19748        8.0000 0000           ; p   q   r   s   t   u   v   w
```

```
19750        8.0000 0067          ; x    y    z    æ    ø           del
                6 bb
19752
19752  ; parameter table:
19752  ; contains a byte for each character type in the follwoing states:
19752  ;      0   initial state
19752  ;      1   after letter
19752  ;      2   after digit
19752  ; each entry defines the address of an action (relative to the
  752  ; procedure next param) and a new state:
19752  ;      entry=action<2 + new state
19752
19752  b.i24
19752    i0=(:d3-d2:)<2+0, i1=i0+1,  i2=i0+2
19752    i3=(:d4-d2:)<2+1, i4=(:d5-d2:)<2+2,  i5=(:d6-d2:)<2+2
19752    i6=(:d7-d2:)<2+0, i7=(:d8-d2:)<2+0
19752
19752                               ; initial state:
19752  h.h1: i3, i5, i6, i0     ;   letter 1, digit 2, unknown 0, continue 0
19756        i6, i7, i6, i0     ;   unknown 0, delimit 0, unknown 0, continue 0
19760                               ; after letter:
19760        i3, i3, i6, i7     ;   letter 1, letter 1, unknown 0, delimit 0
19764        i7, i7, i6, i1     ;   delimit 0, delimit 0, unknown 0, continue 1
19768                               ; after digit:
19768        i6, i5, i4, i7     ;   unknown 0, digit 2, radix 2, delimit 0
19772        i7, i7, i6, i2     ;   delimit 0, delimit 0, unknown 0, continue 2
19776  e.
19776
  776  ; command table:
19776  ; each entry consists of two words defining the name of the
19776  ; command, a byte defining a bit to test in the console mask,
19776  ; and a byte defining the address of the command action
19776  ; relative to g45.
19776
19776  w.h2:         ; first command:                g61 ok as is !!!!!!
19776        [redacted]
19782        [redacted]                                   ok ⇒
19788        <:break:>     , 1<16+g70-g45
19794        <:buf<0>:>    , 1<17+g60-g45          <:job<0>:>
19800        <:call:>      , 1<20+g74-g45
19806        <:catalo:>    , 1<18+g55-g45
19812        [redacted]    , 1<15+g81-g45
19818        [redacted]
19824        [redacted]
19830        [redacted]
19836        <:exclud:>    , 1<19+g73-g45
19842        <:functi:>    , 1<18+g63-g45
  848        <:includ:>    , 1<19+g72-g45
19854        [redacted]
19860        <:intern:>    , 1<17+g62-g45
19866        [redacted]                          <:key<0>:>, 1<18+g55-g45
19872        <:kill:>      , 1<16+g71-g45
19878        <:list:>      , 1<21+g75-g45
19884        [redacted]
19890        <:max<0>:>    , 1<21+g76-g45
19896        <:name:>      , 1<20+g74-g45
19902        <:new<0>:>    , 1<16+g51-g45
19908        <:newdat:>    , 1<15+g78-g45
19914        [redacted]
19920        [redacted]
19926        <:proc:>      , 1<16+g52-g45
19932        <:prog:>      , 1<16+g53-g45
19938        <:remove:>    , 1<16+g71-g45
19944        [redacted]
19950        <:reset:>     , 1<20+g82-g45
19956        <:run<0>:>    , 1<16+g66-g45
  962        <:size:>      , 1<17+g56-g45
19968        <:start:>     , 1<16+g68-g45
19974        ~~<:stop:>      , 1<16+g69-g45~~
19980  h3=k-6        ; last command:
19980
19980  ; console table:
19980
```

<0><0>
<:go:>, 1<16+g68-g45

```
19980    h4:          ; first console:
19980  t.
19980* ~~type~~

19980
19980  ~~; console table within s~~
19980
19980  ~~w.  2, 8.7776  h.0, r.c1-4~~
20016  ~~w.  9, 8.7760  h.0, r.c1-4~~
20052  ~~w. 10, 8.7760  h.0, r.c1-4~~
20088  n.m.
20088                         s console table included
20088    h5=k-c1      ; last console;
20088
20088  ; device table:
20088
20088  h. h6:         ; first device:
20088  t.
20088* ~~type~~

20088
20088  ~~; the following devices are included by s~~
20088
20088   ~~0,1,2,5,7,8,9,10,11,12,18,19,20,21,22,23,24,25~~
20106  n.m.
20106                         s device table included
20106     h7=k-1      ; last device:
20106  w.
20106
20106  ; work table:
20106
20106  h. h8:         ; first work:
20106        0,r.c2*c3
20364     h9=k-c2     ; last work:
20364
20364  ; core table:
20364  ; contains an entry for each storage area allocated to a child.
20364  ; an entry defines the address of a child description within the
20364  ; monitor. the entries are arranged in the same order as the
20364  ; storage areas from low towards high addresses. the table is
20364  ; terminated by a zero.
20364
20364  w.h10: 0,r.a3-1
20402    h11:am       e51-h0    ; first core:
20404        al. w1   h0.
20406        al  w3   x1-e51+f23
20408        jd  1<11+16       ;   send mess(<:operator:>,buf);
20410        rs  w2   x1-e51+e31;   opbuf:= buf;
20412        jl       x1-e51+g30;   goto exam first;
20414        jl.      h11.
20416    h12:
20416
20416  b.i24
20416  w.i0=b29+a4                ; process description address:
20416    i1: i0                   ;
20418    i2: i0+a16               ; process description:
20420    i3: 0                    ; <kind>
20422        <:s:>,0,0,0          ; <name>
20430        a95                  ; <stop count><state>
20432        1<22                 ; <identification bit>
20434        i0+a15, i0+a15       ; <event queue>
20438        i0+a16, i0+a16       ; <process queue>
20442        c0                   ; <first address>
20444        a9                   ; <top address>
20446  h.    a5-1, a1-1           ; <buffer claim><area claim>
20448        a3-2, 8.7777         ; <internal claim><function mask>
20450  w.    8.7777 7777          ; <catalog mask>
20452        1<19                 ; <protection register><protection key>
20454        a89                  ; <interrupt mask>
20456        d0                   ; <interrupt address>
20458        0, r.5               ; <working registers>
20468        h12                  ; <instruction counter>
```

```
20470        0, r.9              ; <parent etc.>
20488        1                   ; <creation no>
20490        ~~-1~~ 0                 ; ~~<device mask>~~   bs base
20492        -1                  ; <selection mask>
20494
20494    i10:rs. w3  i12.        ; initialize segment:
20496        rl  w1  b5          ;
20498        ws  w1  b4          ;
  500        al  w1  x1-2        ;   max device:=
20502        ls  w1  -1          ;   (first area-first device-2)/2;
20504        am      e4-h0       ;
20506        rs. w1  h0.         ;
20508        al. w1  i3.         ;   from:=process description;
20510        rl. w2  i1.         ;   to:=name table(first internal);
20512    i11:rl  w0  x1+0        ;   repeat
20514        rs  w0  x2+0        ;   word(to):=word(from);
20516        al  w1  x1+2        ;   from:=from+2;
20518        al  w2  x2+2        ;   to:=to+2;
20520        se. w1  i10.        ;   until from=end description;
20522        jl.     i11.        ;
20524        al  w1  b2          ;   link(timer q,
20526        rl. w2  i2.         ;        process q(process description));
20528        jl  w3  b36         ;
20530        al. w2  i1.         ;
20532        jl.    (i12.)       ;   autoloader(first core);
20534    i12:0                   ; after loading:
20536        jl.     i10.        ;   goto initialize segment;
  538    c70= k-b127 + 2
20538    k=i1                    ;
20416  e.                        ;
20416  i.
20416
20416  e.     ; end of operating system s
20416
20416
20416                ~~monitor text 5 included~~                  leif
20416
20416
20416                ~~monitor text 6~~
20416
20416  ; segment 9: initialize catalog on backing store
20416  ; per brinch hansen
20416  s.k=k, h9,g54,f19,e25,d39,c25
20416  w.b127=k, c25, k=k-2
20416
20416  ; segment structure:
  416  ;     definitions            (c names)
20416  ;     variables              (d names)
20416  ;     textstrings            (e names)
20416  ;     utility procedures     (f names)
20416  ;     command actions        (g names)
20416  ;     tables and buffers     (h names)
20416  ;
20416  ;     (i and j names are used locally)
20416
20416    d0=k-2                  ; start s:
20416
20416  w.    jl.   g0.           ; first instruction: goto init catalog:
20418
20418  ; definition of backing store:
20418      c0=0                  ; free segments
20418      c1=0                  ; first segment
20418  ; each backing storage device must be defined below by the
20418  ; following statements:
20418  ; monitor 2:
  418  ;     device<13,c1,number of segments on device
20418  ;     c0=c0+number of segments on device
20418  ;     c1=c1+number of segments on device/24*24+24
20418
20418    d1=k                    ; first device:
20418  t.
20~~418~~  ~~type~~
```

```
20418
20418  ; definition of backing store configuration
20418
20418      4<13 , c1, 512*4          ; drum
20424      c0=c0+512*4               ;
20424      c1=c1+512*4/24*24+24      ;
20424
20424      6<13 , c1, 203*10*4       ; disc
20430      c0=c0+203*10*4            ;
20430      c1=c1+203*10*4/24*24+24   ;
20430
20430  n.m.
20430                      init catalog definition of backing store included
20430
20430          0<13,c1               ; dummy device:
20434          0
20436      d2=k                      ; last device:
20436          c2=c1/12              ; size of bit table
20436          c10=1                 ; alternative main console
20436
20436  ; definition of catalog
20436  t.
20436* type

20436
20436  ; definition of catalog
20436
20436      c3=4                          ; device number of cat device
20436      c4=0                          ; first segment on that device (must be zero)
20436      c5=80                         ; catalog size (segments)          ?!
20436      c10=10                        ; alternative main console
20436  n.m.
20436                      init catalog definition of catalog included
20436
20436                                ; catalog area process:
20436      d3: 4                     ; <kind>
20438          <:catalog:>, 0        ; <name>
20446          c3<13+23              ; <device*2><catalog key>
20448          0                     ; <reserved>
20450          2.11<22               ; <users>
20452          c4                    ; <first segment>
20454          c5                    ; <no of segments>
20456      d4: 0                     ; <creator>
20458
20458          c6=2871455            ; hash sum of <:catalog:>
20458          c6=c6-c6/c5*c5        ; hash sum modulo catalog size
20458
20458                                ; catalog entry:
20458      d5: c6<12+23              ; <name key><cat key>
20460          0                     ; <creator>
20462      d6: 0                     ; <first segment>
20464          <:catalog:>,0         ; <name>
20472      d7: c5                    ; <catalog size>
20474          <:wrk000000:>, 0      ; <last work name>
20482      d8: c3                    ; <catalog device>
20484      d9: b50-d2+d1             ; <device table address>
20486      d10:b50-d2+d1-c2          ; <bit table address>
20488      d11:0                     ; <free catalog entries>
20490      d12:0                     ; <free area segments>
20492
20492      d13:b50                   ; <catalog tail address>
20494
20494          510/a88*c5            ; <all catalog entries>
20496      d14:c0                    ; <all area segments>
20498
20498      d15:0,r.8                 ; answer:
20514      d16:0,0                   ; input answer:
20518      d17:0                     ; characters:
20520      d18:-1,0,r.4              ; cur char:
20530      d19:h0                    ; action table:
20532      d20:h1                    ; action end:
```

~~1<12, 0 ; catlength~~

~~c5 ; cat.size~~

```
20534    d21:0                  ; cur action:
20536    d22:0                  ; input buf:
20538    d24:h4                 ; command buf:
20540    d25:h5                 ; command end:
20542    d26:0                  ; cur command:
20544    d27:0                  ; top command:
20546    d28:h6                 ; load buf:
20548    d29:h7                 ; load end:
  550    d30:5<12,h6,h7, 0      ; area output mess:
20558    d31:3<12,h6,h7, 0      ; area input mess:
20566    d33:0                  ; input seg:
20568    d34:0                  ; max seg:
20570    d35:0                  ; checksum:
20572    d36:0                  ;   init cat switch writetext
20574    d37:0                  ;   init cat switch medium
20576    d38:3<12,0,0,0,0       ;   input message 1
20586    d39:3<12,0,0,0,0       ;   input message 2
20596
20596    e0: <:console1:>,0,0
20606    e1: <:inputname:>,0,0
20616    e2: <:catalog:>,0,0
20626    e3: <:<10>initialize catalog?:>, e4=k-2
20640    e5: <:result:>, e6=k-2
20644    e7: <:status:>, e8=k-2
20648    e9: <:input sumerror:>, e10=k-2
20658    e11:<:input sizeerror:>, e12=k-2
20668    e13:<:syntaxerror:>, e14=k-2
  676    e15:<:catalogerror:>, e16=k-2
20684    e17:<:<10>monitor loaded ok<10>:>, e18=k-2
20698    e19:<:<10>initialize date : :>, e20=k-2
20712    e21:27<16 + 54<8 + 49      ;    <:oldcat:>
20714        36<16 + 24<8 + 54      ;    <:end:>
20716         5<16 + 52<8 + 25      ;    as 6-bits
20718        22<16 + 57<8 + 36      ;    characters
20720       121<16                  ;    checksum
20722    e22:<:<10>old catalog<0>:>,e23=k-2
20724    e24:<:<10>new catalog<0>:>,e25=k-2
20742
20742  ; procedure typechar(char)                        Leif to
20742  ; comment: outputs a character on the console.
20742  ;       call:       return:                           here
20742  ; w0   char        char
20742  ; w1               unchanged
20742  ; w2               unchanged
20742  ; w3   link        link
20742
  742  b.i24                    ; begin
20742  w.f0: ds. w0   i1.       ;
20744        ds. w2   i2.       ;
20746        al. w1   i0.       ;
20748        al. w3   e0.       ;
20750        jd   1<11+16       ;   send mess(<:console1:>,char,buf);
20752 31470 - al. w1   d15.     ;
20754        jd   1<11+18       ;   wait answer(buf,answer,result);
20756        dl. w0   i1.       ;
20758        dl. w1   i2.       ;
20760        jl       x3+0      ;
20762    i0: 5<12, i1, i1       ;
20768        0, i1: 0           ;
20772        0, i2: 0           ;
20776  e.                       ; end
20776
20776  ; procedure typeout(first,last)
20776  ; comment: outputs a text on the console.
20776  ;       call:       return:
  776  ; w0               unchanged
20776  ; w1   first       first
20776  ; w2   last        last
20776  ; w3   link        link
20776
20776  b.i24                    ; begin
20776  w.f1: ds. w0   i0.       ;   first addr(mess):=first;
```

```
20778         ds. w2   i2.          ;    last addr(mess):=last;
20780         al. w1   i1.          ;
20782         al. w3   e0.          ;
20784         jd   1<11+16          ;    send mess(<:console1:>,mess,buf);
20786         al. w1   d15.         ;
20788         jd   1<11+18          ;    wait answer(ouf,answer,result);
20790         al   w0   32          ;
20792         jl. w3   f0.          ;    typechar(32);
20794         dl. w0   i0.          ;
20796         dl. w2   i2.          ;
20798         jl        x3+0        ;
20800         0, i0: 0              ;
20804    i1: 5<12, 0, i2: 0         ;
20810    e.                         ; end
20810
20810    ; procedure typein(first,last)
20810    ; comment: inputs a text from the console.
20810    ;      call:      return:
20810    ; w0               unchanged
20810    ; w1   first      first
20810    ; w2   last       last
20810    ; w3   link       link
20810
20810    b.i24                      ; begin
20810    w.f2: ds. w0   i0.         ;    first addr(mess):=first;
20812          ds. w2   i2.         ;    last addr(mess):=last;
20814      i3: al. w1   i1.         ; repeat:
20816          al. w3   e0.         ;
20818          jd   1<11+16         ;    send mess(<:console1:>,mess,buf);
20820          al. w1   d15.        ;
20822          jd   1<11+18         ;    wait answer(buf,answer,result);
20824          rl   w1   x1+0
20826          sn   w0   1          ;    if result<>1
20828          se   w1   0          ;    or status<>0 then
20830          jl.       i3.        ;    goto repeat;
20832          dl. w0   i0.         ;
20834          dl. w2   i2.         ;
20836          jl        x3+0       ;
20838          0, i0: 0             ;
20842     i1: 3<12, 0, i2: 0        ;
20848    e.                         ; end
20848
20848    ; procedure typepause
20848    ;
20848    ; comment: the procedure outputs <:initialize catalog:> and
20848    ;   reads a text from the console.
20848    ;   If neither <:yes:> nor <:no:> is typed, the output
20848    ;   message will be repeated.
20848    ;
20848    ;   The return value ,answer, corresponds to the values
20848    ;   received from the autoloader, describing the initialization
20848    ;   ie.
20848    ;      2 = no initialization
20848    ;      4 =    initialization
20848    ;
20848    ;   registers    call    return
20848    ;     w0          -      unchanged
20848    ;     w1          -      unchanged
20848    ;     w2          -      answer
20848    ;     w3         link    destroyed
20848
20848    b. i1, j7
20848    w.
20848
20848    f3:  ds. w1     j1.      ;   save w0, w1;
20850         rs. w3     j3.      ;   save return;
20852    i0:  al. w1     e3.      ; start:
20854         al. w2     e4.      ;   writetext(initialize catalog);
20856         jl. w3     f1.      ;
20858         al. w1     j6.      ;
20860         al. w2     j7.      ;   read answer;
20862         jl. w3     f2.      ;
```

```
20864         rl. w0     j6.      ;   test answer;
20866         sn. w0    (j4.)     ;   if answer = <:no:> then          ↓
20868         jl.        i1.      ;   goto no;
20870         se. w0    (j5.)     ;   if answer <> <:yes:> then
20872         jl.        i0.      ;   goto start;
20874         am          2       ; yes: answer := 4  or
20876   i1:   al  w2      2       ; no:  answer := 2;
20878         dl. w1     j1.      ;   restore w0, w1;                  ja!
20880         jl.       (j3.)     ;   return;
20882
20882   j0:               0       ;   save w0
20884   j1:               0       ;   save w1
20886   j3:               0       ;   save return
20888   j4:   <:no<10>:>          ;
20890   j5:   <:~~yes~~:>         ;
20892   j6:            0, 0       ;   input buffer = 2 words
20896   j7 = k-1
20896
20896   e.
20896
20896   ; procedure typecommand
20896   ; comment: outputs the command name on the console
20896   ;      call:      return:
20896   ; w0              unchanged
20896   ; w1              unchanged
20896   ; w2   link       link
20896   ; w3              unchanged
20896
20896   b.i24                      ; begin
20896   w.f4: ds. w1  i0.          ;
20898         ds. w3  i1.          ;
20900         rl. w1  d26.         ;
20902         al  w2  x1+2         ;
20904         jl. w3  f1.          ;   typeout(name(cur command));
20906         dl. w1  i0.          ;
20908         dl. w3  i1.          ;
20910         jl      x2+0         ;
20912         0, i0: 0             ;
20916         0, i1: 0             ;
20920   e.                         ; end
20920
20920   ; procedure typeresult(name,result)
20920   ; comment: outputs a name and result on the console.
20920   ;      call:      return:
20920   ; w0   result     result
20920   ; w1              unchanged
20920   ; w2   link       link
20920   ; w3   name       name
20920
20920   b.i24                      ; begin
20920   w.f5: ds. w1  i2.          ;
20922         ds. w3  i3.          ;
20924         al  w1  x3+0         ;
20926         al  w2  x1+6         ;
20928         jl. w3  f1.          ;   typeout(name);
20930         al. w1  e5.          ;
20932         al. w2  e6.          ;
20934         jl. w3  f1.          ;   typeout(<:result:>);
20936     i0: wa. w0  i1.          ;
20938         jl. w3  f0.          ;   typechar(result+48);
20940         dl. w1  i2.          ;
20942         dl. w3  i3.          ;
20944         jl      x2+0         ;
20946     i1: 48                   ;
20948         0, i2: 0             ;
20952         0, i3: 0             ; end
20956
20956   ; procedure typestatus(name,status)
20956   ; comment: outputs a name and the number of the
20956   ; leftmost status bit.
20956   ;      call:      return:
20956   ; w0   status     status
```

```
20956  ; w1              unchanged
20956  ; w2  link       link
20956  ; w3  name       name
20956
20956                                ; begin
20956  w.f6: ds. w1  i2.            ;
20958        ds. w3  i3.            ;
20960        al  w1  x3+0           ;
  962        al  w2  x1+6           ;
20964        jl. w3  f1.            ;   typeout(name);
20966        al. w1  e7.            ;
20968        al. w2  e8.            ;
20970        jl. w3  f1.            ;   typeout(<:status:>);
20972        ls  w0  -1
20974        ns  w0  1              ;
20976        bl  w1  1              ;   typechar(leftmost bit+48);
20978        ac  w0  x1             ;
20980        jl.     i0.            ;
20982  e.                           ; end
20982
20982  ; procedure inchar(char, trouble)
20982  ; comment: inputs the next character from the <input>
20982  ;      call:     return:
20982  ; w0             char
20982  ; w1             unchanged
20982  ; w2             unchanged
20982  ; w3  link       link
  982
20982  b.i24                        ; begin
20982  w.f7: ds. w2  i8.            ;
20984        rs. w3  i9.   72       ;
20986        rl. w2  d18.  74       ;
20988        al  w2  x2+1  376      ;   cur char:=cur char+1;
20990    i0: rs. w2  d18.  0        ;   while cur char=characters do
20992        se. w2 (d17.) 2        ;   begin
20994        jl.     i3.   4        ;
20996        jl. w3   f9.  6        ;   inblock
20998        al  w2  0     10       ;   end;
21000        jl.     i0.   12       ;   cur char:=0;
21002    i3: al  w1  0     14       ;   end;
21004        wd. w2  i6.   16       ;
21006        ls  w1  3     20       ;   pos:=(cur char mod 3)*8-16;
21008        ls  w2  1     22       ;
21010        wa. w2  d22.  24       ;   addr:=input buf+cur char/3*2;
21012        rl  w0  x2+0           ;
21014        ls  w0  x1-16          ;   char:=word(addr) shift pos;
  016        la. w0  i7.            ;   char:=char(17:23);
21018        dl. w2  i8.            ;
21020        rl. w3  i9.            ;
21022        jl      x3+2           ;
21024    i6: 3                      ;
21026    i7: 8.177                  ;
21028        0, i8: 0               ;
21032    i9: 0                      ;
21034  e.                           ; end
21034
21034  ; procedure inword(word, trouble, endseg)
21034  ; comment: inputs a binary word from the <input>. at the
21034  ; end of an input segment the checksum is checked.
21034  ;      call:     return:
21034  ; w0             word
21034  ; w1             unchanged
21034  ; w2             unchanged
21034  ; w3  link       link
21034
  034  b.i24                        ; begin
21034  w.f8: ds. w2  i7.            ;
21036        rs. w3  i8.            ;
21038        al  w0  0              ;   word:=0;
21040        al  w1  18             ;   pos:=18;
21042        rl. w2  d35.           ;
21044    i0: rs. w0  i6.            ;   repeat
```

```
 21046          jl. w3  f7.        ;   incher(char, trouble);
 21048          jl.    (i8.)       ;
 21050          sl  w0  64         ;   if char>63
 21052          jl.     i1.        ;   then goto checksum;
 21054          wa  w2  0          ;   sum:=sum+char;
 21056          ls  w0  x1+0       ;
 21058          lo. w0  i6.        ;   word:=word or char shift pos;
 21060          al  w1  x1-6       ;   pos:=pos-6;
   062          sl  w1  0          ;   until pos<0;
 21064          jl.     i0.        ;
 21066          rs. w2  d35.       ;
 21068          dl. w2  i7.        ;
 21070          rl. w3  i8.        ;
 21072          jl      x3+4       ;   goto exit;
 21074     i1:  se  w1  18         ; checksum:
 21076          jl.     i2.        ;   if pos<>18
 21078          la. w0  i4.        ;
 21080          la. w2  i4.        ;   or char(18:23)<>sum(18:23)
 21082          sn  w0  x2+0       ;
 21084          jl.     i3.        ;   then
 21086     i2:  al. w1  e9.        ;   begin
 21088          al. w2  e10.       ;   typetext(<:input sumerror:>);
 21090          jl. w3  f1.        ;   goto trouble;
 21092          jl.    (i8.)       ;   end;
 21094     i3:  al  w0  0          ;
 21096          rs. w0  d35.       ;   sum:=0;
 21098          dl. w2  i7.        ;
   100          rl. w3  i8.        ;
 21102          jl      x3+2       ;   goto endseg;
 21104     i4:  8.77               ;
 21106     i5:  0, i6: 0           ;
 21110          0, i7: 0           ;
 21114     i8:  0                  ; exit:
 21116  e.                         ; end
 21116
 21116  ; procedure inoutseg(name, mess, trouble, endarea)
 21116  ; comment: inputs or outputs the load buffer from or to the backing store
 21116  ;      call:          return:
 21116  ; w0                  status
 21116  ; w1                  mess
 21116  ; w2   link           link
 21116  ; w3   name           name
 21116
 21116  b.i24                      ; begin
 21116  w.f10:ds. w3  i5.          ;
 21118         rs. w1  i6.         ;
   120         jd  1<11+16         ;   send mess(name,area mess,buf);
 21122         al. w1  d15.        ;   wait answer(buf,answer,result);
 21124         jd  1<11+18         ;   if result<>1 then
 21126         sn  w0  1           ;   begin
 21128         jl.     i1.         ;   typeresult(result,name);
 21130         jl. w2  f5.         ;   goto trouble;
 21132         jl.    (i4.)        ;   end;
 21134    i1:  rl. w0  d15.        ;   status:=word(answer);
 21136         sn  w0  0           ;   if status<>0 then
 21138         jl.     i2.         ;   begin
 21140         rl. w2  i4.         ;
 21142         sz. w0 (i3.)        ;   if status(5)=1
 21144         jl      x2+2        ;   then goto end area;
 21146         jl. w2  f6.         ;   typestatus(status,name);
 21148         jl.    (i4.)        ;   goto trouble;
 21150    i2:  rl. w1  i6.         ;   end;
 21152         rl  w2  x1+6        ;
 21154         al  w2  x2+1        ;
 21156         rs  w2  x1+6        ;   cur seg:=cur seg+1;
   158         dl. w3  i5.         ;
 21160         jl      x2+4        ;
 21162    i3:  1<18                ;
 21164    i4:  0, i5: 0            ;
 21168    i6:  0                   ;
 21170  e.                         ; end
 21170
```

```
21170  ; procedure clear(first,last)
21170  ; comment: initializes a storage area with -1.
21170  ;      call:      return:
21170  ; w0              -1
21170  ; w1   last       last
21170  ; w2   first      last+2
21170  ; w3   link       link
21170
21170  b.i24                     ; begin
21170  w.f11:al   w0   -1        ;
21172    i0: rs   w0   x2+0      ;   repeat
21174        al   w2   x2+2      ;   word(first):=-1;
21176        sh   w2   x1+0      ;   first:=first+2;
21178        jl.       i0.       ;   until first=last+2;
21180        jl        x3+0      ;
21182  e.                        ; end
21182
21182  ; procedure move(first,last,to)
21182  ; comment: moves words from one storage area into another.
21182  ;      call:      return:
21182  ; w0   last       destroyed
21182  ; w1   first      last+2
21182  ; w2   to         to+last-first+2
21182  ; w3   link       link
21182
21182  b.i24                     ; begin
21182  w.f12:rs.  w0   i1.       ;
21184    i0: rl   w0   x1+0      ;   repeat
21186        rs   w0   x2+0      ;   word(to):=word(first);
21188        al   w1   x1+2      ;   first:=first+2;
21190        al   w2   x2+2      ;   to:=to+2;
21192        sh.  w1  (i1.)      ;   until first=last+2;
21194        jl.       i0.       ;
21196        jl        x3+0      ;
21198    i1: 0                   ;
21200  e.                        ; end
21200
21200  ; procedure init bittable
21200  ; comment: initializes the bittable at the top of the store
21200  ;      call:      return:
21200  ; w0              destroyed
21200  ; w1              destroyed
21200  ; w2              destroyed
21200  ; w3   link       destroyed
21200
21200  b.i24                     ; begin
21200  w.f13:rs.  w3   i2.       ;
21202        dl.  w2   d10.      ;
21204        al   w1   x1-2      ;   clear(bittable,
21206        jl.  w3   f11.      ;          bittable+bittable size-2);
21208        al.  w1   d1.       ;   addr:= first device;
21210    i1: rl   w2   x1+2      ;   repeat
21212        wa   w2   x1+4      ;   first seg:= word(addr+2)+word(addr+4);
21214        rl   w0   x1+8      ;   number:= word(addr+8)-first seg;
21216        ws   w0   4         ;
21218        jl.  w3   f15.      ;   reserve seg(first seg,number,
21220        jl        -1        ;         goto -1);
21222        al   w1   x1+6      ;   addr:= addr+6;
21224        sh.  w1   d2.-8     ;   until addr>last device-8;
21226        jl.       i1.       ;
21228        dl.  w1   d14.      ;   free entries:=all catalog entries;
21230        ds.  w1   d12.      ;   free segments:=all area segments;
21232        jl.      (i2.)      ;
21234    i2: 0                   ;
21236  e.                        ; end
21236
21236  ; procedure init devicetable
21236  ; comment: initializes the device table at the top of the store
21236  ;      call:      return:
21236  ; w0              destroyed
21236  ; w1              destroyed
21236  ; w2              destroyed
```

```
21236 ; w3  link        destroyed
21236
21236 b.i24                          ; begin
21236 w.f14:rs. w3  i1.              ;
21238       al. w2  d1.              ;   from:=first device;
21240       rl. w3  d9.              ;   to:=device table;
21242   i0:
21242       rl  w0  x2+4             ;   repeat
21244       rs  w0  x3+4             ;   word(to+4):= word(from+4);
21246       dl  w1  x2+2
21248       ds  w1  x3+2             ;   word(to):=word(from);
21250       ~~ls  w0  -13~~              ;   word(to+2):=word(from+2);
21252       ~~sn. w0 (d8.)~~             ;   ~~if word(from) shift -13 = catalog device~~
21254       ~~rs. w1  d6.~~              ;   ~~then first catseg:=word(from+2);~~
21256       al  w2  x2+6             ;   from:= from+6;
21258       al  w3  x3+6             ;   to:= to+6;
21260       se. w2  d2.              ;   until from=last device;
21262       jl.     i0.              ;
21264       jl.    (i1.)             ;
21266   i1: 0                        ;
21268 e.                             ; end
21268
21268 ; procedure reserve seg(first, number, trouble)
21268 ; comment:sets a string of bits in the bittable equal to zero
21268 ;     call:       return:
21268 ; w0  number      number
21268 ; w1              unchanged
21268 ; w2  first       first
21268 ; w3  link        link
21268
21268 b.i24                          ; begin
21268 w.f15:ds. w1  i7.              ;
21270       ds. w3  i8.              ;
21272       al  w1  x2+0             ;
21274       wa  w1  0                ;   top:=first+number;
21276       sh. w1 (d2.-4)           ;   if top > max top then
21278       jl.     i1.              ; conflict:
21280   i0: al. w1  e15.             ;   begin
21282       al. w2  e16.             ;   typeout(<:catalogerror:>);
21284       jl. w3  f1.              ;   goto trouble;
21286       jl.    (i8.)             ;   end;
21288   i1: al  w1  0                ;
21290       wd. w2  i5.              ;   pos:=number mod 12;
21292       wa. w2  d10.             ;   addr:=bittable+number/12;
21294   i2: bz  w3  x2+0             ; next byte:
21296       ls  w3  x1+0             ;   bit:=byte(addr);
21298   i3: so. w3 (i6.)             ; next bit:
21300       jl.     i0.              ;   if bit(pos)=0 then goto conflict;
21302      _lx. w3  i6.              ;   bit(pos):=0;
21304       bs. w0  1                ;   number:=number-1;
21306       sn  w0  0                ;   if number<>0 then
21308       jl.     i4.              ;   begin
21310       ls  w3  1                ;   pos:=pos+1;
21312       al  w1  x1+1             ;
21314       se  w1  12               ;   if pos<12 then goto next bit;
21316       jl.     i3.              ;
21318   i4: ac  w1  x1+0             ;
21320       ls  w3  x1+0             ;
21322       hs  w3  x2+0             ;   byte(addr):=bit;
21324       al  w1  0                ;   pos:=0;
21326       al  w2  x2+1             ;   addr:=addr+1;
21328       se  w0  0                ;   goto next byte;
21330       jl.     i2.              ;   end;
21332       dl. w1  i7.              ;
21334       dl. w3  i8.              ;   byte(addr):=bit;
21336       jl      x3+2             ;
21338   i5: 12                       ;
21340   i6: 1<11                     ;
21342       0, i7: 0                 ;
21346       0, i8: 0                 ;
21350 e.                             ; end
21350
```

```
21350   ; read block
21350   ; comment delivers one block from input;
21350   ; in case of a hard error, return is made to
21350   ; initialization with the boolean writetext
21350   ; set to true;
21350   ;               call      return
21350   ;      w0        -        destroyed
21350   ;      w1        -        destroyed
21350   ;      w2        -        destroyed
21350   ;      w3       link      destroyed
21350   ; on return d17 is initialized
21350
21350   b. i8, j6
21350   w.
21350
21350   f9:  rx. w3      j3.     ;   save return; get mess. addr.;
21352        rl  w2  x3+6        ;   get buffer address;
21354   i0:  al. w1     d16.     ; wait: get answer address;
21356        jd     1<11+18      ;   wait transfer;
21358        se  w0       1      ;   if result <> 1 then
21360        jl.         i1.     ;   goto result error;
21362        rl  w0  x1+0        ;   test status;
21364        sz. w0    (j0.)     ;   if any error then
21366        jl.         i2.     ;   goto read error;
21368        al  w0       5      ;
21370        rs. w0      j4.     ;   error count := 5;
21372   i6:  rl  w0  x3+2        ; continue:
21374        rs. w0     d22.     ;   save buffer start;
21376        rl  w2  x1+2        ;   no of characters :=
21378        ls  w2      -1      ;   no of bytes +
21380        wa  w2  x1+2        ;   no of no of bytes//2;
21382        rs  w2  x1+4        ;
21384        rl  w1  x3+8        ;   get new mess;age address
21386   i5;  al. w3      e1.     ; read:  get name address;
21388        jd     1<11+16      ;   start transfer;
21390        rs  w2  x1+6        ;   save buffer address;
21392        rx. w1      j3.     ;   save message address;
21394        jl      x1+0        ;   return;
21396
21396   ; result error
21396   i1:  al, w1      f5.     ;
21398        jl.         i4.     ;   out error(type result);
21400
21400   ; read error
21400   i2:  rl. w1     d37.     ;   test init cat medium;
21402        sn  w1       0      ;   if medium = reader then
21404        jl.         i7.     ;   goto test end;
21406        so. w0    (j1.)     ;   if not parity error then
21408        jl.         i3.     ;   goto hard error;
21410        rl. w1      j4.     ;
21412        al  w1  x1-1        ;   decrease error count;
21414        rs. w1      j4.     ;
21416        sh  w1       0      ;   if error count = 0 then
21418        jl.         i3.     ;   goto hard error;
21420        rs. w3      j2.     ;   save message address;
21422        al. w1      j5.     ;   insert move message address;
21424        al. w3      e1.     ;   insert name address;
21426        jd     1<11+16      ;
21428        al. w1     d16.     ;   insert answer address;
21430        jd     1<11+18      ;   wait move;
21432        rl. w1      j2.     ;   restore message address;
21434        jd     1<11+16      ;   start new input;
21436        rl  w3       2      ;   w3 := message address;
21438        jl.         i0.     ;   goto wait;
21440
21440   ; hard error:
21440   i3:  al, w1      f6.     ;   out error( type status);
21442
21442   ; out error:
21442   i4:  al. w3      e1.     ;   get name address;
21444        jl  w2  x1+0        ;   type error;
21446        al  w2       0      ;   no pending answer
```

```
21448         rs. w2      j3.     ;    := true;
21450         jl.        g10.     ;    goto initerror;
21452
21452   ; test end of tape
21452   i7:   sz. w0     (j6.)    ;    if end of tape then
21454         jl.         i6.     ;    goto continue;
21456         jl.         i3.     ;    goto hard error;
21458
21458
21458   ; procedure start transfer
21458   ; comment initializes reading from input
21458   ;           call      return
21458   ;      w0    -        destroyed
21458   ;      w1    -        destroyed
21458   ;      w2    -        destroyed
21458   ;      w3   link      destroyed
21458
21458   f16:  rs. w3      j3.     ;    save return;
21460         al. w1    d38.      ;
21462         al. w2    d39.      ;    get message addresses;
21464         rs. w1   x2+8       ;    establish chain;
21466         rs  w2   x1+8       ;
21468         ~~rl. w0    d37.      ;    blocklength :=~~                   bl:= 64;
21470         ~~se  w0       0      ;    if medium = reader then~~
21472         ~~am    512-64        ;    64 else 512;~~
21474         al  w0      62      ;
21476         al. w3      h7.     ;
21478         al  w3   x3+2       ;    insert buffer addresses;
21480         rs  w3   x1+2       ;
21482         wa  w3       0      ;
21484         rs  w3   x1+4       ;
21486         al  w3   x3+2       ;
21488         rs  w3   x2+2       ;
21490         wa  w3       0      ;
21492         rs  w3   x2+4       ;
21494         jl.         i5.     ;    goto read;
21496
21496
21496   ; procedure end transfer
21496   ; comment the last answer is checked.
21496   ;
21496   ;    registers     call       return
21496   ;       w0           -       destroyed
21496   ;       w1           -       destroyed
21496   ;       w2           -       destroyed
21496   ;       w3          link     destroyed
21496
21496   f17:  rx. w3      j3.     ;    save return;
21498         sn  w3       0      ;    if no pending answer then
21500         jl.         i8.     ;    goto exit;
21502         rl  w2   x3+6       ;    get buffer address
21504         al. w1    d16.      ;    insert answer address;
21506         jd     1<11+18      ;    wait answer;
21508   i8:   al  w2       0      ;  exit:
21510         rx. w2      j3.     ;    change(0, return);
21512         jl       x2+0       ;    return;
21514
21514   j0:   8.77 20 00 00       ;    error bits
21516   j1:   8.20 00 00 00       ;    parity error bit
21518   j2:                0      ;    saved message address
21520   j3:                0      ;    saved return or message address
21522   j4:                5      ;    error count
21524   j5:        8<12,   3      ;    backspace message
21528   j6:   8.01 20 00 00       ;    end of tape bit
21530
21530   e.
21530
21530
21530   ; procedure initialize date
21530   ; comment initializing of date from
21530   ;           the main console;
21530   ;            call      return
```

```
21530 ;     w0     -      destroyed
21530 ;     w1     -      destroyed
21530 ;     w2     -      destroyed
21530 ;     w3    link    destroyed
21530
21530 f18: jl   x3
21532
21532
  532
21532 ; error in initialization
21532 ;
21532 ; The catalog may now be initialized
21532 ; from paper tape
21532
21532
21532 g10: al   w0      0       ; initerror:
21534      rs.  w0    d36.      ;   init cat medium := 0;
21536      rs.  w0    d37.      ;   init cat writetext := 0;
21538      jl.  w3    f17.      ;   end transfer;
21540
21540
21540 ; start initialize catalog
21540 ;
21540 ; get document names of console and input
21540 ; from device table
21540
21540
  540 g0:  am         (b4)      ; start: device:= 2;
21542      rl   w2      +4      ; move:  move name of
21544      dl   w1   x2+4       ;   device 2
21546      ds.  w1      e0.+2   ;   (console);
21548      dl   w1   x2+8       ;
21550      ds.  w1      e0.+6   ;
21552      ~~al   w3   x2~~         ;   sense (device);
21554      ~~am         (b4)~~      ;   if malfunction then
21556      ~~rl   w2    c10+2~~     ;   begin device:= c10;
21558      ~~io   w0  (x3)+a30~~    ;        goto move;
21560      ~~sx           2~~       ;   end;
21562      ~~jl.        2+g0.~~     ;
21564      al.  w3      e1.     ;   insert input name address;
21566      ~~rl.  w1     d37.~~     ;
21568      ~~ls   w1       1~~      ;
21570      ~~wa   w1      b4~~      ;   move input name
21572      rl   w2   ~~x1+0~~ b4    ;   from process descr.
21574      dl   w1   x2+4       ;   to input name area;
21576      ds   w1   x3+2       ;
  578      dl   w1   x2+8       ;
21580      ds   w1   x3+6       ;
21582      ~~rl.  w1     d37.~~     ;   insert device number;
21584      ~~jd      1<11+[illegible]~~   ;   create input;
21586      jd      1<11+8       ;   reserve (input);
21588      ~~jl.  w3    f18.      ;   initialize date;~~
21590      rl.  w2    d36.      ;   if init cat writetext
21592      sn   w2      0       ;   =0 then type pause;
21594      jl.  w3      f3.     ;
21596 ; now w2 contains 2 or 4 which
21596 ; means no initialization and initialization resp.
21596      jl.      x2+0        ;
21598      jl.        g11.      ;   goto no initialization;
21600
21600 ; initialize input
21600      al   w0      0       ;
21602      al   w1     -1       ;   characters := 0;
21604      ds.  w1    d18.      ;   cur char := -1;
 1606      rs.  w0    d35.      ;   sum := 0;
  608      jl.  w3    f16.      ;   start transfer;
21610      al.  w1    e24.      ;   inittext :=
21612      al.  w2    e25.      ;   <:new catalog:>;
21614      jl.        g12.      ;   goto type init;
21616
21616 ; no initialization
21616 g11: al   w0     13       ;
```

Do something here to select another console if error.

```
21618        al   w1      -1       ;   characters := 13;
21620        ds.  w1     d18.      ;   cur char := -1;
21622        al   w0       0       ;
21624        rs.  w0     d35.      ;   character sum := 0;
21626        al.  w0     e21.      ;   buffer :=
21628        rs.  w0     d22.      ;   <:oldcatend:>;
21630        al.  w1     e22.      ;   inittext :=
21632        al.  w2     e23.      ;   <:old catalog:>;
  634   g12: jl.  w3      f1.      ; type init: typeout(inittext):
21636
21636     g1: rl.  w1   d24.       ; input commands:
21638         rs.  w1   d26.       ;   cur command:=
21640     g2: jl.  w3   f8.        ;   top command:=command buf;
21642         jl.       g10.       ;
21644         jl.       g4.        ;   repeat
21646         sh.  w1  (d25.)      ;   input word(input, initerror,next command);
21648         jl.       g3.        ;   if top command>command end then
21650         al.  w1   e11.       ;   begin
21652         al.  w2   e12.       ;   typetext(<:input sizeerror:>);
21654         jl.  w3   f1.        ;   goto initerror;
21656         jl.       g10.       ;   end;
21658     g3: rs   w0   x1+0       ;   word(command top):=input;
21660         al   w1   x1+2       ;   command top:=command top+2:
21662         jl.       g2.        ;   until no limit;
21664     g4: rs.  w1   d27.       ;
21666     g5: rl.  w1   d26.       ; next command:
21668         sl.  w1  (d27.)      ;   if cur command>=command end
  670         jl.       g1.        ;   then goto inout commands;
21672         rl   w0   x1+0       ;   cur action:=action table;
21674     g6: rl.  w2   d19.       ;   repeat
21676     g7: sn   w0  (x2+0)      ;   if word(cur action)=word(cur command)
21678         jl.       g8.        ;   then goto before command;
21680         al   w2   x2+6       ;   cur action:=cur action+6;
21682         sh.  w2  (d20.)      ;
21684         jl.       g7.        ;   until cur action>action end;
21686         jl.  w2   f4.        ;   typecommand;
21688         al.  w1   e13.       ;
21690         al.  w2   e14.       ;
21692         jl.  w3   f1.        ;   typeout(<:syntaxerror:>);
21694         jl.       g10.       ;   goto initerror;
21696     g8: rs.  w2   d21.       ; before command:
21698         rl.  w3   d26.       ;
21700         al   w3   x3+4       ;
21702         al   w1   x3+8       ;
21704         jl       (x2+2)      ;   goto word(cur action+2);
21706  ;      w1=cur command+12    w3=cur command+4
  706
21706     g9: rl.  w2   d21.       ; after command:
21708         rl.  w1   d26.       ;
21710         wa   w1   x2+4       ;   cur command:=
21712         rs.  w1   d26.       ;   cur command+word(cur action+4):
21714         jl.       g5.        ;   goto next command;
21716
21716
21716                              ; create:
21716    g20:jd   1<11+40          ;   create entry(name,tail,result):
21718        jl.       g25.        ;   goto test result;
21720
21720                              ; change:
21720    g21:jd   1<11+44          ;   change entry(name,tail,result):
21722        jl.       g25.        ;   goto test result;
21724
21724                              ; rename:
21724    g22:jd   1<11+46          ;   rename entry(name,result);
21726        jl.       g25.        ;   goto test result:
  728
21728                              ; remove:
21728    g23:jd   1<11+48          ;   remove entry(name,tail,result):
21730        jl.       g25.        ;   goto test result:
21732
21732    g24:rl   w1   x1+0        ; perman:
21734        jd   1<11+50          ;   permanent entry(name,key,result):
```

```
21736
21736                                ; test result:
21736     g25:sn  w0   0            ;   if result<>0 then
21738         jl.      g9.          ;   begin
21740         jl. w2   f4.          ;   typecommand;
21742         jl. w2   f5.          ;   typeresult(result, name);
21744         jl.      g10.         ;   goto initerror;
21746                                ;   end;
21746                                ;   ~~goto after command;~~
21746
21746     g30:al  w0   0            ; load:
21748         rl  w1   x1+0         ;   input seg:=0;
21750         ds. w1   d34.         ;   max seg:mand param;
21752         sh  w1   0            ;   if max seg<=0
21754         jl.      g9.          ;   then goto after command;
21756         rs. w0   d30.+6       ;   cur seg:=0;
21758         jd   1<11+52          ;   create area process(name,result);
21760         se  w0   0            ;   if result<>0
21762         jl.      g25.         ;   then goto test result;
21764         jd   1<11+8           ;   reserve process(name,result);
21766     g31:rl. w1   d28.         ; next buf: addr:=load buf;
21768     g32:jl. w3   f8.          ; next word:
21770         jl.      g35.         ;
21772         jl.      g33.         ;   inword(binword,after trouble,next segment;
21774         rs  w0   x1+0         ;   word(addr):=bin word;
21776         al  w1   x1+2         ;   addr:=addr+2;
21778         sh. w1  (d29.)        ;   if addr<=load end
21780         jl.      g32.         ;   then goto next word;
21782         al. w1   d30.         ;
21784         rl. w3   d26.         ;
21786         al  w3   x3+4         ;
21788         jl. w2   f10.         ;   inputseg(name, area output,
21790         jl.      g35.         ;            after trouble,
21792         jl.      g36.         ;            area exceeded);
21794         jl.      g31.         ;   goto next buf;
21796     g33:rl. w3   d33.         ; next segment:
21798         al  w3   x3+1         ;
21800         rs. w3   d33.         ;   input seg:=input seg+1;
21802         se. w3  (d34.)        ;   if input seg<>max seg
21804         jl.      g32.         ;   then goto next word;
21806         sn. w1  (d28.)        ;
21808         jl.      g34.         ;   if addr<>load buf then
21810         al. w1   d30.         ;
21812         rl. w3   d26.         ;
21814         al  w3   x3+4         ;
21816         jl. w2   f10.         ;   inputseg(name, area output,
21818         jl.      g35.         ;            after trouble,
21820         jl.      g36.         ;            area exceeded);
21822     g34:rl. w3   d26.         ; after load:
21824         al  w3   x3+4         ;
21826         jd   1<11+64          ;   remove process(name,result);
21828         jl.      g9.          ;   goto after command;
21830     g36:jl. w2   f6.          ; area exceeded:
21832                                ;   typestatus(status, name);
21832
21832     g35:rl. w3   d26.         ; after trouble:
21834         al  w3   x3+4         ;
21836         jd   1<11+64          ;   remove process(name,result);
21838         jl.      g10.         ;   goto initerror;
21840
21840    ; clear backing storage catalog
21840    ; initializes the following description of the backing store at the
21840    ; top of the core store:
21840    ;
21840    ;      <bit table>
21840    ;      <device table>
21840    ; b50:<number of segments in catalog>
21840    ; b51:<last work name>
21840    ; b52:        -
21840    ; b53:        -
21840    ; b54:        -
21840    ; b55:<catalog device number>
```

```
21840   ; b56:<device table address>
21840   ; b57:<bit table address>
21840   ; b58:<free catalog entries>
21840   ; b59:<free area segments>
21840   ;     (top of core store)
21840   ;
21840   ;     the bit table contains one bit for each segment on the
21840   ; backing store. all bits are initialized to 1 (free segment).
21840   ; in order to prevent areas from extending over several devices,
21840   ; each device in the bit table is terminated by some extra bits
21840   ; which are initialized to 0 (reserved segment). bits corresponding
21840   ; to the catalog area are also set to 0.
21840   ;
21840   ;     the device table contains three words for each backing store
21840   ; device:
21840   ;     <device number>*8192
21840   ;     <first segment on device>
21840   ;     <number of segments on the device>
21840   ;
21840   ;     the first segment is the number of the first bit in the
21840   ; bit table which corresponds to the backing store device.
21840   ;
21840   ;     after the initialization of the backing store description,
21840   ; all catalog segments are initialized as follows:
21840   ;     entry0=-1
21840   ;     entry1=-1
21840   ;     ----
21840   ;     entry14=-1
21840   ;     last word=0
21840   ;
21840   ;     finally, the catalog segment on which the catalog itself is
21840   ; described is initialized as follows:
21840   ;     entry0=catalog entry
21840   ;     entry1=-1
21840   ;     ----
21840   ;     entry14=-1
21840   ;     last word=1
21840
21840   b.i24                     ; new catalog:
21840   w.g50:jl. w3  f13.        ;    init bittable;
21842         jl. w3  f14.        ;    init devicetable;
21844         rl. w1  d29.        ;
21846         rl. w2  d28.        ;
21848         jl. w3  f11.        ;    clear(load buf,load end);
21850         al  w0  0           ;
21852         rs  w0  x1+0        ;    word(load end):=0;
21854         rs. w0  d30.+6      ;    cur seg:=0;
21856         al. w3  e2.         ;
21858         jd  1<11+8          ;    reserve process(<:catalog:>,result);
21860     i0: al. w1  d30.        ; write catalog:
21862         al. w3  e2.         ;
21864         jl. w2  f10.        ;    inoutseg(<:catalog:>, area output,
21866         jl.     i5.         ;             cattrouble,
21868         jl.     i1.         ;             write catentry);
21870         jl.     i0.         ;    goto write catalog;
21872     i1: dl. w2  d12.        ; write catentry:
21874         al  w1  x1-1        ;    free entries:=free entries-1;
21876         ws. w2  d7.         ;    free segments:=free segments-catalog size;
21878         ds. w2  d12.        ;
21880         rl. w0  d7.         ;
21882         rl. w2  d6.         ;    reserve seg(first catseg,
21884         jl. w3  f15.        ;                catalog size,
21886         jl.     i5.         ;                cattrouble);
21888         al. w0  d12.        ;
21890         al. w1  d7.         ;
21892         rl. w2  d13.        ;    move(catalog size,free segments,
21894         jl. w3  f12.        ;         catalog tail);
21896         al. w0  d12.        ;
21898         al. w1  d5.         ;
21900         rl. w2  d28.        ;    move(catalog entry,free segments,
21902         jl. w3  f12.        ;         load buf);
21904         al  w0  1           ;
```

```
21906          rs. w0 (d29.)       ;   word(load end):=1;
21908          bz. w0  d5.         ;
21910          rs. w0  d30.+6      ;   cur seg:=catalog namekey;
21912          al. w1  d30.        ;
21914          al. w3  e2.         ;
21916          jl. w2  f10.        ;   inoutseg(<:catalog:>, area outout,
21918          jl.     i5.         ;           cattrouble,
21920          jl.     i5.         ;           cattrouble);
21922          al. w3  e2.         ;
21924          jd  1<11+10         ;   release process(<:catalog:>);
21926          jl.     g9.         ;   goto after command;
21928
21928    i5: al. w3  e2.           ; cattrouble:
21930          jd  1<11+10         ;   release process(<:catalog:>);
21932          jl.     g10.        ;   goto initerror;
21934  e.                          ;
21934
21934  b.i24                       ; old catalog:
21934  w.g51:jl. w3  f13.          ;   init bittable;
21936          jl. w3  f14.        ;   init devicetable;
21938          al. w0  0           ;
21940          rs. w0  d31.+6      ;   cur seg:=0;
21942    i0: al. w1  d31.          ; read catalog:
21944          al. w3  e2.         ;
21946          jl. w2  f10.        ;   inoutseg(<:catalog:>, area input,
21948          jl.     g10.        ;           initerror,
21950          jl.     i4.         ;           move cattail);
21952          rl. w1  d28.        ;   entry:=load buf;
21954    i1: rl  w0  x1+0          ;   repeat
21956          sh  w0  -1          ;   if namekey(entry)>=0 then
21958          jl.     i3.         ;   begin
21960          rl  w0  x1+14       ;   if size(entry)>0 then
21962          sh  w0  0           ;   begin
21964          jl.     i2.         ;
21966          rl  w2  x1+4        ;
21968          jl. w3  f15.        ;   reserve seg(first(entry),size(entry),
21970          jl.     g10.        ;              initerror);
21972          rl. w3  d12.        ;   free segments:=
21974          ws  w3  0           ;   free segments-size(entry);
21976          rs. w3  d12.        ;   end;
21978    i2: rl. w3  d11.          ;   free entries:=free entries-1;
21980          al  w3  x3-1        ;
21982          rs. w3  d11.        ;   end;
21984    i3: al  w1  x1+a88        ;   entry:=entry+entry size;
21986          se. w1 (d29.)       ;   until entry=load end;
21988          jl.     i1.         ;
21990          jl.     i0.         ;   goto read catalog;
21992    i4: al. w0  d12.          ; move cattail:
21994          al. w1  d7.         ;
21996          rl. w2  d13.        ;   move(catalog size,free segments,
21998          jl. w3  f12.        ;        catalog tail);
22000          jl.     g9.         ;   goto after command;
22002  e.                          ;
22002  b. i0, j0
22002  w.
22002
22002    g54:                      ; end initcat:
22002       jl. w3    f17.         ;   end transfer;
22004       al. w3     e1.         ;
~~22006       rl. w0    d37.         ;   if medium <>~~
~~22008       sn  w0      0          ;   reader then~~
~~22010       jl.         i0.        ;   begin~~
~~22012       al. w1      j0.        ;     rewind(input);~~
~~22014       jd      1<11+16        ;~~
~~22016       jd      1<11+18        ;     remove(input);~~
~~22018       jd      1<11+64        ;   end;~~
22020    i0: jd  1<11+10           ;   release process(input);
22022       al. w1   e17.          ;
22024       al. w2   e18.          ;
22026       jl. w3   f1.           ;   typeout(<:ready:>);
22028       jl.      d0.           ;   goto start s;
22030  ~~j0: 8<12+4~~                  ;   rewind message
```

al w0 10
jl. w3 f0.

```
22034
22034  e.
22034
22034  ; action table:
22034  ; each command is described by its name, the address of
22034  ; the command action, and the number of command bytes.
22034
22034  w.h0=k
  034       <:cre:>, g20,32    ; <:create:><name><tail>
22040       <:cha:>, g21,32    ; <:change:><name><tail>
22046       <:ren:>, g22,20    ; <:rename:><name><new name>
22052       <:rem:>, g23,12    ; <:remove:><name>
22058       <:per:>, g24,14    ; <:perman:><name><cat key>
22064       <:loa:>, g30,14    ; <:load:><name><segments>
22070       <:new:>, g50,4     ; <:newcat:>
22076       <:old:>, g51,4     ; <:oldcat:>
22082    h1: <:end:>, g54,2    ; <:end:>
22088
22088    h4=k                  ; command buf:
22088    h5=h4+510             ; command end:
22088
22088    h6=h5+2               ; load buf:
22088    h7=h6+510             ; load end:
22088
22088  b.i24                   ; begin
22088  w.i0: rs. w3  i2.       ; initialize segment:
22090       al. w0  d4.        ;
  092       al. w1  d3.        ;   move(catalog area process,
22094       rl  w2 (b5)        ;        no of segments,
22096       jl. w3  f12.       ;        name table(first area));
22098       am.     (i2.)      ;
22100       dl   w1    -2      ;   transfer
22102       ds.  w1   d37.     ;   init cat switches;
22104       al  w1      0      ;
22106    i1: al  w2  x1        ;   sense all devices;
22108       ls  w2   6         ;
22110       io  w0  x2         ;
22112       al  w1  x1+1       ;   comment: to open interrupt lines
22114       sh  w1  [redacted]     ;   from data communications controller;
22116       jl.      i1.       ;
22118       ic       -1        ;   clear interrupts
22120       jl    (10)         ;   goto system start;
22122    i2: 0                 ; after loading:
22124       jl.      i0.       ;   goto initialize segment;
22126    c25=k - b127 + 2
22126  e.                      ; end
22126  i.
22126  e.     ; end of initialize catalog on backing store
22126
22126
```

~~t62~~ 16

no init

W0 = 2 / 4 → init cat

W1 = irrel.

```
22126
22126
22126  ; segment 10
22126  ; bjørn ø-thomsen
22126  ;
22126  ; this segment moves segment 2 - 9 in this way:
22126  ;
22126  ; segment 2 is moved to cell 8 and on, after which
22126  ; control is transferred to the last moved word with the
22126  ; following parameters:
22126  ;     w2 = top load address (= new address of last  moved
22126  ;                                 word + 2)
22126  ;     w3 = link
22126  ;
22126  ; after initializing itself, the program segment returns
22126  ; to this segment with:
22126  ;     w2 = load address of next segment
22126  ;
22126  ; the next segment will then be moved to cell(w2) and on,
22126  ; after which it is entered as described above.
22126  ;
22126  ; when initialize catalog (segment 9) is entered, the values
22126  ; of the two switches (writetext, medium) may be found in
22126  ; the words x3-4 and x3-2.
22126  ;
22126  ; segment 10 is entered from segment 1 in its last word
22126  ;   entry conditions:
22126  ;     w0,w1 = init catalog switches
22126  ;     w2    = start address of segment 2
22126
22126
22126                          ok
22126
22126  s.   i5,  j5
22126  w.
22126                 j3. [redacted]    ;   length of segment 10
22128  j0:       0 →                    ; init cat switch: writetext
22130  j1:                0             ; init cat switch: medium
22132  [redacted]
22132
22132  ; return point from initializing of some segment
22132
22132  i0:  rl. w1      j2.       ; get load address;
22134  i1:  wa  w1  x1+0          ; calculate top address:
22136       rx. w1      j2.       ;   change(old load address, top address);
22138       al  w1  x1+2          ;   skip segment length;
22140
22140  ; now w1, w2 = old, new load address
22140
22140  ; move segment:
22140
22140  i2:  rl  w0  x1+0          ;   move word from old
22142       rs  w0  x2+0          ;     to new address;
22144       al  w1  x1+2          ;   update old addr;
22146       al  w2  x2+2          ;   update new addr;
22148       se. w1    (j2.)       ;   if old addr <> top addr
22150       jl.         i2.       ;   then goto move segment;
22152
22152  ; now the segment has been moved
22152  ; jump to the last moved word
22152
22152       al. w3     i0.        ;   insert return;
22154       jl      x2-2          ;   goto word(top addr - 2);
22156
22156  ; comment:  jump to last loaded word with
22156  ;           w2          = top load address
22156  ;           w3          = link
22156  ;           word(x3-4) = init cat switch, writetext
22156  ;           word(x3-2) = init cat switch, medium
```

```
  22156
  22156
  22156  ; initialize segment 10
  22156
  22156  i3:  ds. w1      j1.     :   save init cat switches
  22158       rs. w2      j2.     :
  22160       al  w2       8      :   new load addr := 8;
  22162       jl.         i0.     :   goto get load address;
    164
  22164  j2:               0      :   top address
  22166       jl.         i3.     :   goto initialize segment 10
  22168  j3:                      : top address of segment 10:
  22168
  22168  e.   ;  end segment 10
  22168  i.
  22168
  22168  m.
  22168                 monitor text 6 included
  22168
  22168  e.
slang ok 10/16272/32
```

```
; rc 4000 monitor options
; Danish Meteorological Institute

(message assembly follows
  set t t                     ; final tape newmon

monitor0=set r r r 0          ; label file
monitor1=set r r r 1          ; text1: disabled part
monitor2=set r r r 2          ; text2: io drivers
monitor3=set r r r 3          ; text3: more drivers
monitor4=set r r r 4          ; text4: enabled part
monitor5=set r r r 5          ; text5: oosys s
monitor6=set r r r 6          ; init catalog
monitor7=set r r r 7          ; update monitor
clear mimonitor               ; remove old entry

mimonitor=set 50              ; binary monitor

mimonitor=slang  monitor1 monitor2 monitor3 ,
                monitor4 monitor5 monitor6 ,
                type.yes list.no           ;


head 1
o c)


; monitor size options

  a1=72         ; area processes
  a3=20         ; internal processes
  a5=142        ; message buffers
  a9=64<11      ; core (kwords) (does not make simul possible)
  a87=10000     ; clock inspection interval in 0.1 msec
n.

; include code for external process drivers

  a91= 2,1111 0100 0000 0001 1010 0000
n.       ø123 456

; processes in name table before first device

n.

; device list in name table

  g0 ,g1 ,g2 , g3 ,g4 ,g5 ,g6 ,g7 ,g8 ,g9
  g10,g11,g12,g13,g14,g15,g16,g17,g18,g19
  g20,g21,g22,g23,g24,g25
n.

; descriptions of external processes

n.

; descriptions of peripheral processes

w.c3:  jl.w1 h5.
       c36,g2


  c17: jl.w1 h5.
       c36,g9

  c5:  jl.w1 h5.
       c36,g10
```

```
b.j32w.
   j18: c50,g18
   j19: c50,g19
   j20: c50,g20
   j21: c50,g21
   j22: c50,g22
   j23: c50,g23
   j24: c50,g24
   j25: c50,g25

   c18: jl w1 c31, 17<6
   j18-a56
   j19-a56
   j20-a56
   j21-a56
   j22-a56
   j23-a56
   j24-a56
   j25-a56
   h4, r.16
e.

   c12: jl w1 c30
   g0:  10, <:reader:>,0,0
        0<6,  0,1<22,  k,k-2
        c33,  0,r.8

     5: jl w1 c30
   g1:  12, <:punch:>,0,0
        1<6,  0,1<22,  k,k-2
        c33,  0,r.8

   c6:  jl w1 c30
   g2:  8,  <:console1:>,0
        2<6,  0,1<22,  k,k-2
        c33,  0,0,24
        0,r.10,   37,25, 8

   c14: jl w1 c30
   g3:  2,  <:clock:>,0,0
        3<6,  0,0,     k,k-2
        c35

   c11: jl w1 c30
   g4:  6,  <:drum:>,0,0
        4<6,  0,0,     k,k-2
        c33,  0,r.3

   c13: jl w1 c30
   g5:  14, <:printer:>,0
        5<6,  0,1<22,  k,k-2
        c33,  0,r.8

   c16: jl w1 c30
   g6:  6,  <:disc:>,0,0
        6<6,  0,0,     k,k-2
        c33,  0,r.3

   c9:  jl w1 c30
   g7:  34, <:tapeunit7:>,0
        7<6,  0,1<22,  k,k-2
        c37,  1,-1,-1
        0   , r.6

   c10: jl w1 c30
   g8:  34, <:tapeunit8:>,0
        8<6,  0,1<22,  k,k-2
        c37,  1,-1,-1
        0   , r.6

   c7:  jl w1 c30
   g9:  8,  <:console3:>,0
```

```
      9<6,  0,1<22,  k,k-2
      c33,  0,0,128
      0,r.10,    37,25, 8

c8:   jl w1 c30
g10:  8,  <:console2:>,0
      10<6,  0,1<22,  k,k-2
      c33,   0,0,128
      0,r.10,   37,25,8

c19:  jl w1 c30
g11:  54, <:plotter1:>,0
      11<6,  0,1<22,  k,k-2
      c33,   510
c20:  jl w1 c30
g12:  54, <:plotter2:>,0
      12<6,  0,1<22,  k,k-2
      c33,   510

g13=h4
g14=h4
g15=h4
g16=h4 ; telecom controller base reg 0
g17=h4 ; telecom controller base reg 1

g18:  58, ~~<:txp1:>,0,~~0   <:txp1quick:>, 0
      18<6, 0,1<22,  k,k-2
      c33,  0,32,32
      0,r.5

g19:  58, ~~<:txp2:>,0,0~~   <:txp2quick:>, 0
      19<6, 0,1<22,  k,k-2
      c33,  0,32,32
      0,r.5

g20:  58, ~~<:tgp3:>,0,0~~   <:tgp3oslo:> , 0
      20<6, 0,1<22,  k,k-2
      c33,  0,32,32
      0,r.5

g21:  58, <:telex4:>,0,0
      21<6, 0,0<22,  k,k-2
      c33,  0,32,32
      0,r.5
                             <:tgp201ring2: >
g22:  58,~~<:tgp201:>,0,0~~
      22<6, 0,1<22,  k,k-2
      c33,  0,32,32
      0,r.5

g23:  58, ~~<:txp6:>,0,0~~   <:txp1utrecht:>
      23<6, 0,1<22,  k,k-2                        bytom !! ?
      c33,  0,32,32
      0,r.5

g24:  58, ~~<:tgp7:>,0,0~~   <:tgp7oslo:>
      24<6, 0,1<22,  k,k-2
      c33,  0,32,32
      0,r.5

g25:  58, ~~<:tgp101:>,0,0~~   <:tgp101ring1:>
      25<6, 0,1<22,  k,k-2
      c33,  0,32,32
      0,r.5

[illegible]:   jl w1 c31 , 16<6
      g18        ; connector 1
      g19        ; connector 2
      g20        ; connector 3
      g21        ; connector 4
      g22        ; connector 5
      g23        ; connector 6
```

```
       g24      ; connector 7
       g25      ; connector 8
       h4, r.16  ; not used


n.

; interrupt list

  c0 ,c1 ,c2 ,c3 ,c4 ,c5 ,c6 ,c7 ,c8 ,c9
  c10,c11,c12,c13,c14,c15,c16,c17,c18,c19
  c20,c24,c24,c24,c51

n.

; opsys s  size options


  c3=3                       ; number of work areas
  c6=1                       ; standard keys
  c7=7                       ; standard buf
  c8=6                       ; standard area
  c9=0                       ; standard internal
  c10=8.7440                 ; standard function mask
  c11=1<23                   ; standard catalog mask
  c12=12800                  ; standard size (=6400 words)
  c23=-1                     ; any system option included
n.

; console table within s

w.  2, 8.7776  h.0, r.c1-4
w.  9, 8.7760  h.0, r.c1-4
w. 10, 8.7760  h.0, r.c1-4
n.

; the following devices are included by s

   0,1,2,5,7,8,9,10,11,12,18,19,20,21,22,23,24,25
n.

; definition of backing store configuration

  4<13 , c1, 512*4          ; drum
  c0=c0+512*4               ;
  c1=c1+512*4/24*24+24      ;

  6<13 , c1, 203*10*4       ; disc
  c0=c0+203*10*4            ;
  c1=c1+203*10*4/24*24+24 ;

n.

; definition of catalog

  c3=4                      ; device number of cat device
  c4=0                      ; first segment on that device (must be zero)
  c5=80                     ; catalog size (segments)
  c10=10                    ; alternative main console
n.


clear monitor0 monitor1 monitor2 monitor3,
      monitor4 monitor5 monitor6 monitor7 r
message binary monitor on mimonitor

end
```