```
 /INTERDATA 70/7477-16 OPERATING SYSTEM
/Department of Computer Science
 /Edinburgh University
 / Editing station version from LEGOS:22

 /Ascii characters
 $define soh=1, stx=2, etx=3, eot=4, bs=8, lf=10,nl=10
$def ff=12, rt=13, can=24, esc=27, del=127

 /Sign-bit
$def s=x'8000'

 /Condition-code values
$def neg=1, a=2, zero=\3, less=8, carry=8
 $def ousv=8, clear=\15, trouble=7

/User status (psw)
 $def usr=x'4100'

/Process kernel components
 $def ps=0, pc=2                 /hardware psw
 $def exec=4                     /ad of terminal process
$def link=6                      /for squeuering
 $save 8                         /register save displacement
                                 /register save area (32 b)
$def incb=b(40), outcb=b(41)     /current io
 $def inout=40
 $def phit=42
$def nplim=44
 $def np=46                      /nest pointer
 /User processes only
$def in0=b(48), out0=b(49)       /stream slots (8+ b)
 $def inout0=48
 $def in1=b(50), out1=b(51)
$def ttin=b(56), ttout=b(57)
 $def ttinout=56
 $def kbin=b(58), monout=b(59)
$def cfparm=60                   /command file parm pointer
 $def insno=b(62), outsno=b(63) /current slotnos
 $def prom=64
$def gram=66
 /   spare 68, 70
 $def save=72, store=80
$def inname=store, outname=store+60, progname=store+120
 $def usize=256                  /total user kernel size

/Terminals (executives) only
 $def attn=32                    /user interrupt (r12)
 $def vtype=48                   /vdu type (sign-bit)
$def userno=b(49)                /filestore user no
 $def uname=50                   /user name (7 b)
 /    ttout=b(57)
$def libname=58                  /library name (7 b)
 $def xstate=b(65)               /number of sub-pros
 /   gram=66
/    spare 68
 /   spare 70
 /   save 72
$def tstore=88
 $def tsize=256

$/svcs
 $def call=svc 0         /push return address to nest and branch to <arg>
 $macro nroutine n       /to declare a nest routine
$def n=svc 0,*
 $end
```

```
$def return=ssvc 1,0    /pop value from nest and branch to it
$def push=ssvc 2        /push specified register contents to nest
$def pop=ssvc 3         /pop value from nest to specified register
$def claim=svc 4
$def release=svc 5
$def service=svc 6
$def selin=svc 7        /select input stream <arg> (0-3)
$macro selout arg       /select output stream <arg> (0-3)
svc 7,arg+x'8000'
$end
$def nsym=svc 9,999     /next symbol to r1
$def rsym=svc 9,0       /read symbol to r1
$def match=svc 9        /NSYM: if r1 = <arg>, RSYM & cc zero
/                               else, cc non-zero
$def testbusy=svc 9,512 /test no data available (not TT)
$def rhsym=svc 9,-1
$def psym=svc 10        /print symbol <arg>
$def prompt=svc 11      /set <arg> as address of prompt sequence
/                        for subsequent terminal input
$macro refin no,name    /define input stream <no> (1-3)
/                        as <name> (string address)
r14=name: r0=no
ssvc 14,4
$end
$macro refout no,name   /define output stream <no> (1-3)
/                        as <name> (string address)
r14=name: r0=no
ssvc 14,5
$end
$def refall=ssvc 14,2   /define input streams 1-3 and output streams 1-3
/                        using parameter strings in process base
$def instream=ssvc 12,12 /return current input stream no. in r0
$def outstream=ssvc 12,13 /return current output stream no. in r0
$def resetin=ssvc 13,12 /reset current input stream
$def resetout=ssvc 13,13 /reset current output stream
$def setin=ssvc 12,14   /set current input file to blockno. r0
$def closin=ssvc 13,14  /close current input stream
$def closout=ssvc 13,15 /close current output stream
$def fclaim=ssvc 14,10  /claim filestore links [r0 preserved]
$def prolog=ssvc 13,0   /print symbol in r0; then puserno
$def puserno=ssvc 14,8  /print filestore user-number enclosed in commas
$def response=ssvc 14,12 /read filestore response: no. to r0, rest ignored
$def frelease=ssvc 14,11 /release filestore links [r0 preserved]
$def ferr=ssvc 14,9     /print filestore error message
$def fcomm=ssvc 13,1    /perform complete filestore interchange
/                        command letter in r0 (+256 to suppress err fail);
/                        string address(es) in r14(,r15);
/                        response returned in r0
$def stop=ssvc 14,15    /close files and terminate current program
$def fail=ssvc 14,13    /ignore pending input and STOP
$def abandon=ssvc 14,14 /mark disk output files as transient and FAIL
$def ignore=ssvc 15,0   /ignore pending input
/                        (all if current input is not from terminal)
$def skip=ssvc 15,1     /skip spaces, leaving next symbol in r1
$def rhex=ssvc 15,2     /read hexadecimal number to r0
/                        (0 if next non-space symbol not hex digit)
$def phex=ssvc 15,3     /print r0 as hexadecimal number (four digits)
$def phex2=ssvc 13,7    /print r0 as hex number (two digits)
$def chname=ssvc 15,5
$def rdec=ssvc 15,6     /read unsigned decimal number to r0
/                       (0 if next non-space symbol not digit)
$def pdec=ssvc 15,7     /print r0 as unsigned decimal number (1-5 digits)
$def rstring=ssvc 14,1  /read string to string variable addressed by r14
/                        terminator <= r15
$def pstring=ssvc 15,9  /print string addressed by r14
$def rname=ssvc 15,8    /read name to string variable
/                        addressed by r14 (max 19 chars)
/                        name terminated by: sp  ,  /  ;  nl
$def ptext=ssvc 15,10   /print text sequence following call [r0 preserved]
/                        (0 as terminator)
$def attend=ssvc 15,11  /return user interrupt character in r0 (0 if none)
```

```
 $def pdec=ssvc 15,7      /print r0 as unsigned decimal number (1-5 digits)
 $def rstring=ssvc 14,1   /read string to string variable addressed by r14
 /                         terminator <= r15
 $def pstring=ssvc 15,9   /print string addressed by r14
 $def rname=ssvc 15,8     /read name to string variable
 /                         addressed by r14 (max 19 chars)
 /                         name terminated by: sp  ,  /  ;  nl
 $def ptext=ssvc 15,10    /print text sequence following call [r0 preserved]
 /                         (0 as terminator)
 $def attend=ssvc 15,11   /return user interrupt character in r0 (0 if none)
 $def newline=ssvc 15,12  /PSYM nl   [r0,r1 preserved]
 $def space=ssvc 15,13    /PSYM ' '  [r0,r1 preserved]
 $def testdig=ssvc 15,14
 $def testlet=ssvc 15,15
 $def tod=ssvc 15,4       /read time-of-day from filestore
 /                         to string addressed by r14
 /                         [r15 destroyed]
 $def bmove=ssvc 13,2
 $def smove=ssvc 13,3
 $def scomp=ssvc 13,4
 $def sappend=ssvc 13,5
 $def sfind=ssvc 13,6
 $def vet=ssvc 12,10
 $def rparm=ssvc 13,8
 $def setcursor=ssvc 13,9 /print characters required to set vdu cursor
 /                         to position defined by r0
 /                         ms byte: line (0-23); ls byte: column (0-79)
 $def clearline=ssvc 13,10 /print characters required to clear
 /                         from current cursor position to end of line
 $macro default def       /set defaults for stream assignments
 /                         as specified by <def> (string address)
 r0=def: ssvc 13,11
 $end
 $def resetcursor=ssvc 12,11 /ac=23<<8; SETCURSOR
 $def envinfo=ssvc 12,8   /set initial register contents
 $macro nsize b           /define nest size
 r0=b; ssvc 12,9
 $end

 $macro routine n,r[ret]
 $def n=bal r,w(*)
 $redef out=w(r)
 $end
 $def out=0

 /Fixed addresses
 $def reslim=x'2C00'            /resident system limit
 $def syslim=reslim+512         /loaded system limit

 /Video screen size
 $def vlines=24, vlast=vlines-1 /numbered 0-23

 $/Assembler signal for core image
 $assloc x'81'
 /Prefixed bootstrap for entry from Autoload
   jump w(*) if \clear          /check status ok after autoload
   r5 = b(x'78')                /bootstrap dev
   r1 = x'40'
   r0 = 1: oc r0,r1             /display (inc)
   r4 = x'D0'                   /load pointer
   cycle
     exbr r5,r4
     wh r0,r5
     cycle
       ss r3,r1
     repeat if \clear
     rd r3,b(r4)
     r4 = r4+1
   repeat if r4 # #zbase+x'D0'
   b x'D0'-* $ x'08'            /executable padding
```

```
 $assloc 1                          /0 & no loader codes

 $label command, despatch, abdn, abdn0, abdn1, kill
 $label ferrfail,svgo, syn0, pt0, pbase
$def pcomm=call #pcomm0

 $/initial entry to system
/Copy down from wherever loaded to location zero
 / unavoidably tortuous code to work even at loc 0
 $def jumpr1=x'0301'
   r4 = jumpr1+syslim-24
   r3 = 8
   r2 = jumpr1
   w(8) = r2
   bal r1,w(r3)
   lm r12,w(r1)
   stm r12,w(r2+x'10'-jumpr1)
   r1 = r1+8
   bxle r2,w(x'10')
   jump w(reslim)

 /Power-fail (x'22')
   0, 0, 0

 /Interrupt psws (x'28')
   0, 0, x'28', #illeg              /fp fault
   0, 0, x'30', #illeg              /illegal inst
   0, 0, x'38', #illeg              /machine error
intpsw:0, 0, 0, #intser             /external int
   0, 0, x'48', #illeg              /divide error


 $def fsrdev=x'08', fstdev=x'0C'

/Isys bootstrap (x'50')
 $begin
 $def blk=r10, lt=r13, lr=r14
   lm blk,regs                      /(blk must be even reg)
   stm r11,w(x'34')                 /will do
   lr = lr>>8                       /rx
   wb lt,blk                        /send request
   cycle
     ss lr,r0
   repeat if \clear
   rd lr,r0                         /n1
   jump w(x'7C')                    /->autoload
   mess:b 'z,,isys80.z',n1
 regs:#mess                         /r10
   #mess+11                         /r11
   x'5A'                            /r12
   r0 = fstdev                      /r13 (& mini-boot)
   b fsrdev,x'80'                   /r14 (at x'78')
   wd r0,r0                         /(mini-boot)
   x'D500'; x'00CF'                 /autoload
 $end


 $def disable=b(x'79')
$def enable=b(*), retrans=b(*+1)
 $def clerr=b(*+2), bit9=b(*+3)
   b x'40', x'20', x'10', x'01'
topstore:0                          /set by init
 bgstart:0                          /"
 fskill:0
   0                                /spare
 proset:x'C0'
 $def tbase0=reslim+512
ubase0:0                            /set by init
 pblim:0                            /"
   0                                /spare
```

```
tobstore:0                      /set by init
 bgstart:0                      /"
 fskill:0
   0                            /spare
 proset:x'C0'
 $def tbase0=reslim+512
ubase0:0                        /set by init
 pblim:0                        /"
   0                            /spare

 /Svc area (x'94')

svcarg:0                        /svc argument
 svcpsw:0, 0                    /svc psw
   0                            /svc status (new)
   #svcall                      /svc 0
   #svret                       /ssvc 1
   #svpush                      /ssvc 2
   #svpop                       /ssvc 3
   #svclaim                     /svc 4
   #svrel                       /svc 5
   #svser                       /svc 6
   #svsel                       /svc 7
   #svnsym                      /svc 8
   #svrsym                      /svc 9
   #svpsym                      /svc 10
   #svprom                      /svc 11
   #svc12                       /svc 12
   #ssvcall                     /ssvc 13
   #ssvcall                     /ssvc 14
   #ssvcall                     /ssvc 15

 waitq:0                        /wait queue
null:0                          /empty string

 /Legos bootstrap (x'C0')
   r10 = #leg; r11 = #leg+14
   lm r12,w(x'74')              /rest of isys boot regs
   jump w(x'54')
leg:b 'Z,,ISYS80.LEGO', nl

 sysident:b 10,'ISYS80.V03'
lp:b 3,'lp.'
   3 $ 0                        /spare
 fsave:0
usave:0, 0, 0, 0

 $/Resource control blocks (x'100')

 $def ltemp=*list
 $list *list&(\1)

 $def user=*                    /ad of using process
 $def devinf=user+2             /(various) (+s for eof)
$def server=user+4              /ad of serving pro (or cb a)
 $def pointer=user+6            /buffer control pointer
 $def bmask=\(s+511)

 /Unbuffered streams (4 bytes)
/Null
   0, x'8000'
   0, x'8000'

 /Devices (* device-code = cb *)
 /user:devinf
$def lkcb0=*-user, fsr=lkcb0, fst=fsr+4
   0, 0                         /fsr
   0, 0                         /fst
$def kcb0=*-user, pcb0=kcb0+4, timer=#devinf(kcb0-pcb0)
   1, 0                         /ttl (kb)
```

```
   1, 0                          /tt1 (tp)
   1, 0                          /ir1
   1, 0                          /it1
   1, 0                          /tt2 (kb)
   1, 0                          /tt2 (tp)
   1, 0                          /ir2
   1, 0                          /it2
   1, 0                          /tt3 (kb)
   1, 0                          /tt3 (tp)
   1, 0                          /ir3
   1, 0                          /it3
   1, 0                          /tt4 (kb)
   1, 0                          /tt4 (tp)
   1, 0                          /ir4
   1, 0                          /it4

/Buffered streams (8 bytes)

 /Terminal input
/user:limit:server:pointer
 $def tcb0=*-user, limit=devinf
   1, 0, 0, 0                    /set by init
   1, 0, 0, 0
   1, 0, 0, 0
   1, 0, 0, 0

 /Disk file control blocks
 /user:comm.alt.xno:server:pointer
/NB buffers must be on 512 byte boundaries
 $def dcb0=*-user
   0, 0, #dbase, reslim        /to cover initial code
   1, 0, #dbase, 0             / set by init
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
   1, 0, #dbase, 0
$def bg=*-user
   0, 0, #dbase, 0
 $def cblim=*-user
   0, 0, #dbase, 0

 /Built-in process kernels

 /Filestore driver
 dbase:
   x'4000', #dkent, 0, reslim+512
   0, 0, *-12, 0, 0, 0, fsrdev, fstdev
   0, 0, 0, 0, 0, 0, 0, 0
   0, s>>9, *+4, *+10, 4 $ 0

 ibase:
   usr, #idle, 0, 0
   0, 0, *-12, pcb0, 16, tcb0-12, x'7FFF', 1
   8 $ 0
   0, s>>8

 $list ltemp

 $begin basic supervisor
 $temp r15
```

```
    usr, #rate, 0, 0
    0, 0, *-12, pcb0, 16, tcb0-12, x'7FFF', 1
    8 $ 0
    0, s>>8

  $list ltemp

  $begin basic supervisor
  $temp r15
 $def pb=r12, sw1=r13, sw2=r14, temp=r15
  $def newpb=r13, cb=r14          /note equivalences
  $def sercb=r3                   /service queue

  /Interrupt service
  intser:
    usave = pb                    /free a register
    pb = pbase                    /current process base
    stm r0,r0(pb)                 /save registers
    pb(pb) = usave
    lm r14,intpsw                 /save intpsw as psw
    stm r14,ps(pb)
    ai cb,temp                    /acknowledge interrupt
    oc cb,disable
    newpb = user(cb)              /user waiting?
    jump w(*) if newpb >= 0       /no <=>
    newpb = newpb-s               /remove wait bit
    user(cb) = newpb
    link(newpb) = pb              /push-down current pro
  svgo:
    sw2 = pbit(newpb)\proset
    temp = 1
    wn temp,pbit(newpb); wh temp,sw2
    pbase = newpb
    lm r0,r0(newpb)               /restore all registers
    x'C200'                       /lpsw
  pbase:0                         /*impure*


  svnout:
    svcpsw[1] = 1                 /cc non-zero
  svout:
    lm r12,usave
    lpsw svcpsw

  /SVC  c l a i m    resource (class)
 /sign bit means no waiting
  /r1 <-resource (neg if unavailable)
  svclaim:
    stm r12,usave
    pb = pbase
    r1 = svcarg[1]
    cycle
      if user(r1) = 0             /resource available
        user(r1) = pb
        jumps svout
      finish
      r1 = r1+8                   /next disk file cb
    repeat if r1 > dcb0 and r1 < cblim
    r1 = svcarg
    jump svout if r1 < 0          /no waiting if neg

  /Add claimant to wait queue
  svq:
    sw1 = #waitq-link             /adjusted pointer
    cycle                         /Find end of queue
      sw2 = sw1
      sw1 = link(sw2)
    repeat if sw1 # 0
    link(sw2) = pb                /append current pro
 /Wait and re-try
  svwt0:
```

```
    svcpsw(2) = svcpsw(2)-4          /back off to repeat svc
  /Relinquish cpu
  svwait:
    newpb = link(pb)                 /next pro on run queue
    jumps svwt1
  svswop:
    link(newpb) = link(pb)           /new pro replaces current
  svwt1:
    link(pb) = 0
  /Save registers etc
  svswitch:
    lm r14,usave(4)                  /*pb=r12,newpb=r13*
    stm r0,r0(pb)
    lm r14,usave
    stm r14,r12(pb)
    pc(pb) = svcpsw(2)
    jump svgo

  /SVC  r e l e a s e   resource
  /OK if resource not owned
  svrel:
    stm r12,usave
    pb = phase
    cb = svcarg
    jump svrel1 if cb >= 0

  /Process termination
    sw2 = exec(pb)
    sw1 = xstate(sw2)-1              /sole sub-pro?
    jump svout if sw1 = 0            /yes =>
    xstate(sw2) = sw1
    proset = proset\pbit(pb)
    exec(pb) = 0                     /process released
    jump svwait

  svrel1:
    if user(cb) = pb                 /owned
      jump svserz if cb >= dcb0 and devinf(cb) # 0
      user(cb) = 0                   /set free
    finish
    newpb = waitq                    /pros on wait queue?
    jump svout if newpb = 0          /no =>
    temp = newpb                     /put all on run queue
    cycle
      sw2 = temp
      temp = link(sw2)
    repeat if temp # 0
    link(sw2) = pb                   /ahead of current pro
    waitq = temp                     /zero
    jump svswitch

  /Close disk file, ignominiously
  svserz:
    temp = x'E000'
  svserd:
    temp = temp!devinf(cb)
    devinf(cb) = temp
  /Call server and repeat original svc
  svser0:
    svcpsw(2) = svcpsw(2)-4
    jumps svser1

  /SVC  s e r v i c e   control-block
  /Sign-bit means no waiting
  svser:
    stm r12,usave
    pb = phase
    cb = svcarg
    jumps svser2 if cb < 0
  svser1:
    user(cb) = pb+s                  /waiting
```

```
 /SVC  s e r v i c e   control-block
 /Sign-bit means no waiting
svser:
   stm r12,usave
   pb = pbase
   cb = svcarg
   jumps svser2 if cb < 0
 svser1:
   user(cb) = pbts                /waiting
 svser2:
   cb = cb&255
   newpb = server(cb)
   if newpb >= #dbase            /not on service q already
     server(cb) = 0
     temp = sercb(newpb)          /head of service q
     if temp = 0                  /server idle
       sercb(newpb) = cb
       jump svswop if user(cb) < 0 /to wait
       link(newpb) = pb
       jump svswitch
     finish
     cycle                        /find end of service q
       sw1 = temp
       temp = server(sw1)
     repeat if temp # 0
     server(sw1) = cb             /append this cb
   finish
   jump svwait if user(cb) < 0
   jump svout

 /SVC  n e x t   s y m b o l  to r1
 /Read if svm = svcarg
/Svcarg=512: test data available (link only)
 svnsym:
   r1 = svcarg                    /* temp for old progs *
   if r1 <= 0
     r1 = 999
     svcarg = r1
   finish
 /SVC  r e a d   s y m b o l  to r1
 svrsym:
   stm r12,usave
   pb = pbase
   svcpsw[1] = 0                  /cc zero
   sw1 = svcarg
   cb = incb(pb)
   if cb < tcb0                   /Device (or null)
     jump sveof if devinf(cb) < 0
     ss cb,temp
     if busy
       jump svout if sw1 = 512 /just testing =>
       jump dev2
     finish
     temp = temp<<15              /bit9 to sign
     if sw1 # 0                   /nsym
       oc cb,retrans
       rd cb,r1
       r1 = r1+temp
       jump svnout if r1 # sw1 /no match  =>
     finish
     rd cb,r1
     r1 = r1+temp
/     devinf(cb) = devinf(cb)+temp
     jump svout
   finish
   if cb < dcb0                   /Terminal input
     if user(cb) # pb
       jump svq if user(cb) # 0
       user(cb) = pb
     finish
```

```
    temp = pointer(cb)
    jump svser0 if temp = limit(cb)
    r1 = b(temp)
    r1 = -1 if r1 = eot
    jump svnout if sw1 # 0 and sw1 # r1
    temp = temp+1
    pointer(cb) = temp
    jump svrel1 if r1 < esc
    jump svout
  finish
  temp = pointer(cb)            /Disk
  if temp < 0                   /data available
    r1 = b(temp-s)
    jump svnout if sw1 # 0 and sw1 # r1 /no match  =>
    sw1 = temp
    jump svpinc
  finish
  jump svser0 if devinf(cb) > 0 /not ended ->
  if devinf(cb) = 0             /repeating after close
    sw1 = cfparm[pb]
    user(sw1) = 0 if sw1 # cb   /release
    sw1 = pointer(sw1)&bmask    /buffer start
    cfparm[pb] = b(sw1+120)     /restore former cfparm
    temp = b(sw1+121)           / and input 0
    sw1 = pb+8
    cycle
      sw1 = sw1-2
      in0(sw1) = temp if in0(sw1) = cb
    repeat if sw1 # pb
    incb(sw1) = temp
    r1 = -1
    jump svrel1
  finish
sveof:
  r1 = -1
  jump svnout if cb # in0(pb) or sw1 # 0
svserc:
  temp = x'6000'
  jump svserd

/Read back symbol
svbsym:
/   cb = outcb(pb)
/   jump sveof if cb < dcb0
/   sw1 = pointer(cb)
/   jump svrs5 if sw1&(\bmask) = 0 /buffer empty ->
/   sw1 = sw1-s+512 if sw1 < 0
/   sw1 = sw1-1
/   pointer(cb) = sw1
/   r1 = b(sw1)
/   jump svout


/SVC  p r i n t   s y m b o l  sym
svpsym:
  stm r12,usave
  pb = pbase
  cb = outcb(pb)
  if cb < dcb0                  /Device
    if cb&8 # 0                 /link (ie not tt)
      ss cb,temp
      jump dev1 if \clear
      temp = svcarg
      oc cb,bit9 if temp < 0
      wd cb,temp
      jump svout
    finish
    if user(cb) # pb            /cb not claimed
      jump svout if cb = 0      /null =>
      jump svq if user(cb) # 0 /not available ->
```

```
      jump dev1 if \clear
      temp = svcarg
      oc cb,bit9 if temp < 0
      wd cb,temp
      jump svout
    finish
    if user(cb) # pb             /cb not claimed
      jump svout if cb = 0       /null =>
      jump svq if user(cb) # 0 /not available ->
      user(cb) = pb              /claim it
    finish
    cycle
      ss cb,temp
      jump dev2 if busy
      sw1 = devinf(cb)           /pending lf?
      exit if sw1 >= 0           /no ->
      sw1 = sw1<<1>>9            /line count
      jump svq if sw1 = 64+vlast /page-mode and last line ->
      sw1 = sw1+1 if sw1&31 # vlast
      devinf[cb] = sw1
      sw1 = lf; wd cb,sw1
    repeat
    temp = svcarg
    jump svout if temp < 0       /dummy to flush
    if temp = ni
      sw1 = sw1+s
      devinf(cb) = sw1
      temp = rt
    finish
    wd cb,temp
    sw1 = sw1+1 if temp >= esc /(will do)
    sw1 = sw1&255
    sw1 = 0 if temp = rt
    sw1 = sw1-1 if temp = bs and sw1 > 0
    devinf[cb+1] = sw1
    jump svout if sw1 # 0
    timer(cb) = sw1              /zero
    jump svrel1
  finish
  sw1 = pointer(cb)
  jump svser0 if sw1 < 0         /buff full ->
  b(sw1) = svcarg                /store sym
svpinc:
  sw1 = sw1+1
  sw1 = sw1+s-512 if sw1&511 = 0
  pointer(cb) = sw1
  jump svout

/Device error or busy
 dev1:
  if trouble
    oc cb,clerr
    r13 = -1
    jump abdn1
  finish
 dev2:        /busy
  oc cb,enable
  user(cb) = pb+s
  jump svwt0

/SVC  s e l e c t / r e s e t
 /stream:0-5  + sign for out
 /             + x'0000' select (old reset)
/              + x'4000' reset
 /(             + x'6000' close)
 svsel:
  stm r12,usave
  pb = phase
  sw2 = svcarg
  /convert arg if pre-Isys80 program
  sw2 = sw2<<15+x'4000' if sw2&(\1)=0 and pbit(pb)&1#0
```

```
    sw2 = sw2<<1
    sw2 = sw2+1 if carry          /old sign to ls bit
  svsel1:
    sw1 = sw2&1+pb                /to map out to in
    jump svres1 if sw2 < 0
    sw2 = sw2&255                 /** temp **
    clh sw2,12
    sw2 = sw2&1 if \less
    insno(sw1) = sw2              /slotno (0-11)
    sw2 = sw2+pb                  /slotad
    incb(sw1) = in0(sw2)
    jump svout
  svres1:
    cb = incb(sw1)                /(NB cb==sw2)
    jump svout if cb = 0 or cb = in0(sw1)
    temp = devinf(cb)
    if cb >= dcb0                 /disk
      temp = temp!x'4000'         /'reset'
      devinf(cb) = temp
      jump svser1                 /get server to do it
    finish
    temp = temp&x'7FFF'           /clear eof
    devinf(cb) = temp
    jump svout


  /SVC  p r o m p t  textad
  svprom:
    stm r12,usave
    pb = pbase
    prom(pb) = svcarg
    jump svout

  /SVC  c a l l  user routine
  svcall:
    stm r12,usave
    pb = pbase
    sw1 = svcarg                  /entry point
    sw2 = svcpsw(2)               /return address
    jump svpp1                    /-> push

  /SSVC  c a l l  system routine
  / E1Cx, E1Dx, E1Ex, F1Fx
  svc12:                          /old select
    stm r12,usave
    pb = pbase
    sw2 = svcarg
    jump svsel1 if pbit(pb)&1 # 0 /old prog ->
    jumps ssvc1
  ssvcall:
    stm r12,usave
    pb = pbase
  ssvc1:
    sw2 = svcpsw(2)-2             /return address
    sw1 = srtab(w(sw2-2)<<1-(x'E1C0'<<1)) /entry point
  svpp1:
    temp = np(pb)
    if temp = nplim(pb)           /nest full (apparently)
      jump nestabdn if temp # pb+usize /really full ->
      if r2 # pb and sw1 # #pt0 /(not ptext)
        w(temp-2) = sw2
        w(temp-4) = r2
        r2 = pb
        temp = temp-4
        bal sw2,svpp2
        pop r2; return
      finish
    finish
  svpp2:
    temp = temp-2
    w(temp) = sw2
```

```
      w(temp-4) = r2
      r2 = pb
      temp = temp-4
      bal sw2,svpp2
      pop r2; return
    finish
  finish
svpp2:
  temp = temp-2
  w(temp) = sw2                 /store value on nest
svpp3:
  np(pb) = temp                 /update np
  svcpsw(2) = sw1
  jump svout

/SSVC  r e t u r n
svret:
  stm r12,usave
  pb = pbase
  temp = np(pb)
  sw1 = w(temp)                 /recover return address
  temp = temp+2                 /np+2
  jump svpp3

/SSVCR  p u s h  reg
svpush:
  stm r12,usave
  pb = pbase
  sw1 = svcpsw(2)-2             /resumption ad
  sw2 = r0                      /value to be nested
  sw2 = svcarg-w(sw1) if w(sw1-2)&15 # 0
  temp = np(pb)
  jump svpp2 if temp # nplim(pb)
nestabdn:
  r13 = 'n'<<8+'e'
  jump abdn0

/SSVCR  p o p  reg
svpop:
  stm r12,usave
  pb = pbase
  temp = np(pb)
  jump svout if temp = pb+usize /empty ->
  inst(2) = temp
  temp = temp+2
  np(pb) = temp
  sw2 = svcpsw(2)-2
  svcpsw(2) = sw2
  inst = w(sw2-2)<<4-(x'F130'<<4)+x'4800' /lr
  lm r12,usave
inst:0; 0                       /r? = nest  *impure*
  lpsw svcpsw

/System Routine entry points
srtab:
$end basic supervisor

  8 $ #abdn
  #env0, #ns0, #dv0, #rcurs0
  #inst0, #outst0, #setin0, #abdn
  #prlog0, #fc0, #bm0, #sm0
  #sc0, #sa0, #sf0, #ph2
  #rp0, #scurs0, #clear0, #defl0
  #rsin0, #rsout0, #clin0, #clout0
  #abdn, #rs0, #refl0, #abdn
  #rin0, #rout0, #oclin0, #oclout0
  #puser0, #ferr0, #fclai0, #frele0
  #respo0, #fai0, #abdn, #stop0
  #ig0, #sk0, #rh0, #ph0
  #tod0, #cbn0, #rd0, #pd0
  #rn0, #ps0, #pt0, #att0
```

```
  #nl0, #sp0, #td0, #tl0

$begin system processes
$def ac=r0, temp=r1, pb=r2, cb=r3
$def work=r4, ret=work, pro=r5

$temp r1

/Idle process - just times out tt hoggers
/ NB must never WAIT
$def timelim=r6, c1=r7, c0=r8
idle:
  timer(cb) = timer(cb)+c1
  if timer(cb) = timelim
    timer(cb) = c0
    if user(cb) >= ubase0
      user(cb) = pb             /to permit release
      release cb
    finish
  finish
  bxle cb,idle
  cb = pcb0
  jump idle

routine seteot
  devinf(cb) = devinf(cb)!s
  jump out

routine siguser
  pro = user(cb)                /user waiting ?
  jump out if pro >= 0          /no =>
  pro = pro-s
  user(cb) = pro                /no longer waiting
routine schedule
  link(pro) = link(pb)          /insert after this pro
  link(pb) = pro
  jump out

$temp

routine stop,temp
  work = server(cb)             /service queue
  server(cb) = pb               /restore server
  cb = work                     /more service requests?
  jump out if cb # 0            /yes  =>

$def newpb=r13                  /*nb*
routine wait,temp
  pc(pb) = temp
  stm r0,r0(pb)                 /save registers
  newpb = link(pb)              /pop-up next pro
  jump svgo
$temp r1


$begin terminal input (keyboard) handlers
/ac=r0, temp=r1, pb=r2, cb=r3
/work=r4, ret=work, pro=r5 (siguser)
$def sym=r5, kcb=r6, altcursor=r7
$def errpos=r8, macpos=r9, pr=r10, ret2=r11
$def attn=r12, putr=r13, start=r14, buffr=r15
$def pcb=kcb+4                  /output device cb
$def ucb=start                  /for state, etc
$def bel=7

/Switch between alternative cursor positions
routine switch
  ac = altcursor
  if ac >= 0
    altcursor = devinf(pcb)     /save current
```

```
$def pcb=kcb+4                  /output device cb
$def ucb=start                  /for state, etc
$def bel=7

/Switch between alternative cursor positions
routine switch
  ac = altcursor
  if ac >= 0
    altcursor = devinf(pcb)     /save current
    setcursor                   /set the other
  finish
  jump out

/Read character from kb device
routine kread
  ss kcb,temp
  if busy
    oc kcb,enable
    user(kcb) = pb+s
    wait
  finish
  rd kcb,sym
  sym = sym&127
  jump out

/Read character from macro defn (self-cancelling)
routine macread
  sym = b(macpos); macpos = macpos+1
  macpos = 0 if b(macpos)&128 # 0
  jump out

/Reset buffer pointers
routine restore
  putr = buffr
  pointer(cb) = buffr
  limit(cb) = buffr
  jump out

routine erase,temp
  psym bs
  psym ' '
  psym bs
  jump out

routine vetsym
  ac = errpos
  jump out if ac # 0 /earlier error (cc g) =>
  jump out if pr&256 = 0 /not vetting (cc,ac z) =>
  ac = pr if putr = start /initial call (else zero)
  temp = sym
  cycle
    vet
  repeat if g /action
  jump out /cc z if ok else neg

/Lookup macro symbol: no return if found
routine maclook
  macpos = #tstore(pb)
  cycle
    temp = b(macpos)
    macpos = macpos+1
    jump out if temp = 128      /not found =>
  repeat if temp # sym+128
  jumps kbnext
/Read line to buffer @ PUTR (incrementing)
/ mode controlled by PR:
/      0  - echoing, no vetting
/      2  - no echoing, no vetting
/    256+ - echoing, vetting
/      1  - after ^D, defining escape sequence
/      s  - echoing then erasing typeahead
```

```
routine readline,ret2
  start = putr
kbnext:
  jump out if putr > pb+tsize-2 /buff full =>
  jump out if pr = 1 and putr > pb+tsize-72
  sym = macpos
  if sym # 0
    if sym&(\255) = 0          /pending symbol
      macpos = 0
    else                       /macro or typeahead
      macread
    finish
  else
    switch
    kread
    switch
    if sym = esc
      kread                    /get next
      maclook                  /no return if found
      macpos = sym             /hold next
      sym = esc
    finish
  finish
  b(putr) = sym                /assume wanted
  jump kbput(kbsw(sym)<<1) if sym < 32 /control =>
$def k=*                       /switch base
kbput:
  vetsym
  if zero                      /no error
    jump lastinc if ac&256 # 0 /end of phrase =>
  else if neg and sym # del
    errpos = putr
    psym '?'; psym bel
  finish
  jump kbcan if sym = del
  putr = putr+1
  if pr # stx                  /echoing
    sym = '~' if sym = esc
    psym sym
    clearline if altcursor >= 0 and putr = start+1
  finish
kbign:
  jump kbnext
kc: /uncommitted ctrl
  maclook                      /no return if found
  macpos = 0
  jump can2
kbbs:
  jump kbcan if putr # start or pr = 1
  psym '!'
  kread
  if sym < ' '
    erase
    psym rt
    jump out
  finish
  psym sym
  attn = sym!32
  jump define if attn = 'd'
  kread
  jump kbidl1 if sym # rt
  jump desp if attn = 'n'
  newline
  jump state if attn = 's'
  jump kbattn if attn = 'e'    /(errpos 0)
  errpos = #abdn
  jump kbattn if attn = 'a'
  errpos = #kill
  jump kbattn if attn = 'k'
  jump kbidl1
```

```
    jump dosb if attn = 'n'
    newline
    jump state if attn = 's'
    jump kbattn if attn = 'e'    /(errpos 0)
    errpos = #abdn
    jump kbattn if attn = 'a'
    errpos = #kill
    jump kbattn if attn = 'k'
    jump kbidll
kbcan:
    while putr # start
      putr = putr-1
      erase if pr # stx
      if putr = errpos
        erase                    /'?'
        errpos = 0
      finish
    repeat if sym # del or errpos # 0 /can,bs
can2:
    jump out if putr = start and pr < 0
rescan:
    if pr&256 # 0 and errpos = 0     /vetting
      ac = pr           /to reset
      work = start
      while work # putr
        work = work+1
        temp = b(work-1)
        cycle
          vet
        repeat if a
      repeat if zero             /(safety)
    finish
    jump kbnext
kbq:                             /quit page-mode
    work = 0
    jumps kbs1
kbs:                             /start page-mode
    work = x'4000'
kbs1:
    devinf(pcb) = devinf(pcb)&x'80FF'+work
    jump can2
kbb:                            /temporary facility
    work = x'80'                 /Bantam
    jumps kbv1
kbv:
    work = x'40'                 /VT52
kbv1:vtype[ph] = work
    jump kbnext
kbrt:kblf:
    jump lastinc if pr < 0     /typeahead ->
    sym = sym\(lf\rt)             /rt->lf(nl), lf->rt
    b(putr) = sym
    vetsym
    jumps kbnl if zero
    psym bel
    macpos = 0
    jump rescan
kbeot:
    psym '$'
kbnl:
    newline if altcursor < 0
lastinc:
    putr = putr+1
kbetx:
    jump out

kbsw:
  b #kbign-k>>1, #kc-k>>1, #kbb-k>>1, #kbetx-k>>1, /nul, soh, stx, etx
    #kbeot-k>>1, #kc-k>>1, #kc-k>>1, #kc-k>>1, /eot, enq, ack, bel
    #kbbs-k>>1, #kc-k>>1, #kblf-k>>1, #kc-k>>1, /bs, ht, lf, vt
    #kbnl-k>>1, #kbrt-k>>1, #kc-k>>1, #kc-k>>1, /ff, rt, so, si
```

```
  #kc-k>>1, #kbq-k>>1, #kc-k>>1, #kbs-k>>1, /dle, dc1, dc2, dc3
  #kc-k>>1, #kc-k>>1, #kc-k>>1, #kc-k>>1, /dc4, nak, syn, etb
  #kbcan-k>>1, #kc-k>>1, #kc-k>>1, #kbout-k>>1, /can, em, sub, esc
  #kc-k>>1, #kc-k>>1, #kc-k>>1, #kc-k>>1 /fs, gs, rs, us

/Main program -- exit sequence
kbidle:
  attn = 0
kbidl1:
  switch /restore cursor
  release pcb                   /(best
  siguser                       / order?)
  user(kcb) = pb+s              /ints ok
  oc kcb,enable
  stop
/Main program -- entry-point
kbent:
  oc kcb,disable
  ac = devinf(pcb)              /tt output in page mode?
  devinf(pcb) = ac&x'C0FF' if ac&x'4000' # 0 /reset line count if so
  errpos = 0; altcursor = -1
  if cb # 0                     /service requested
    work = user(cb)&x'7FFF'
    user(pcb) = pb if user(pcb) = work /grab tp
    if ac&x'2000' # 0 /alt cursor position
      altcursor = ac&x'7FFF'
      resetcursor
    finish
    gram(pb) = gram(work)
    pr = 0
    work = prom(work)
    if work # 0
      sym = b(work) /length
      work = work+1 if sym < 32 /(in case old style)
      cycle
        pr = b(work)
        sym = sym-1
        pr = 0 if sym < 0
        exit if pr < eot
        psym pr
        work = work+1
      repeat
    finish
    pr = b(work+1)+256 if pr = soh /grammar entry
    psym -1                     /flush lf if pending
    if pointer(cb) < putr       /type-ahead present
      macpos = pointer(cb)
      b(putr) = 128
    finish
    restore
    readline
    limit(cb) = putr
    while macpos >= putr        /type-ahead remaining
      macread
      b(putr) = sym
      putr = putr+1
    repeat
  else                          /spontaneous input
    cb = kcb>>1+tcb0-8          /tcb
    server(cb) = 0
    restore if pointer(cb) = putr /all prior input away
    rd kcb,sym
    sym = sym&127
    if sym = lf and ac = x'C000'+(vlast<<8)
      ac = ac-256; devinf(pcb) = ac
      jump kbidle
    finish
    jump kbidle if ac&x'2000' # 0 and sym # bs
    macpos = sym
    pr = s
```

```
    rd kcb,sym
    sym = sym&127
    if sym = lf and ac = x'C000'+(vlast<<8)
      ac = ac-256; devinf(pcb) = ac
      jump kbidle
    finish
    jump kbidle if ac&x'2000' # 0 and sym # bs
    macpos = sym
    pr = s
    readline
    work = putr                 /Erase from screen
    cycle
      work = work-1
      exit if work <= start
      erase
    repeat
  finish
  jump kbidle

$/attention sequences

define:
  kread
  kread if sym = esc
  if sym >= ' '
    ptext; b ' esc',0
    psym sym
  else
    ptext; b ' ctrl', 0
    psym sym+64
  finish
  psym '='
  putr = #tstore(pb)            /search macro buffer
  cycle
    temp = b(putr)
    if temp = sym+128           /old def
      work = putr               /find end
      cycle
        work = work+1
      repeat if b(work)&128 = 0
      cycle                     /copy up rest
        temp = b(work)
        b(putr) = temp
        exit if temp = 128
        work = work+1; putr = putr+1
      repeat
    finish
    putr = putr+1
  repeat if temp # 128
  attn = altcursor; altcursor = sym-128 /save (altc neg)
  ac = 0; pr = 1
  readline
  putr = putr-1 if putr = start+1
  b(start-1) = altcursor      /sym+128 (as byte)
  altcursor = attn
  devinf(pcb) = 0 if altcursor >= 0 /inhibit lf
  b(putr-1) = 128
  buffr = putr
  restore
  jump kbidle

routine findsubpro,ac
  temp = pb
routine findpro,ac
  cycle
    pro = protusize
    pro = ubase0 if work = 0
    work = work+1
    jump out if exec(pro) = temp
  repeat if pro < pplim
  pro = 0
```

```
    jump out

/New command: despatch process
desp:
    erase                           /'n'
    restore
despatch:
    siguser
    work = 0; temp = 0
    findpro
    if pro = 0                      /none available
      ptext; b'No go',nl,0
      jump kbidle
    finish
    exec(pro) = pb
    ttinout(pro) = inout(pb)
    inout0(pro) = ttinout(pro)
    kbin(pro) = kcb                 /kb as device
    monout(pro) = 0
    cfparm(pro) = 0
    schedule
    proset = proset!ppit(pro)
    xstate[pb] = xstate[pb]+1
    jump kbidle

kbattn:
    work = 0
    cycle
      findsubpro
      jump kbidle if pro = 0
      ucb = lkcb0+16
      cycle
        if user(ucb) = protst       /user waiting
          ac = cb; cb = ucb
          oc cb,disable
          seteot
          siguser
          cb = ac
        finish
        ucb = ucb+4
        ucb = ucb+8
      repeat if ucb < tcb0
      if errpos # 0                 /abandon, kill
        if user(fsr)&x'7FFF' = pro
          if user(fst) = 0          /running remote program
            ac = fst; wd ac,attn
          else
            fskill = errpos
          finish
        else
          pc(pro) = errpos
        finish
      finish
    repeat

state:
    work = 0
    cycle
      findsubpro
      if pro = 0
        ptext; b'free',0
      else
        psym 'p'
        ac = work; pdec
        ptext; b '(kb',0
        ac = kbin(pro)>>4; pdec
        psym ')'
      finish
      ucb = lkcb0
      cycle
```

```
    else
      psym 'p'
      ac = work; pdec
      ptext; b '(kb',0
      ac = kbin(pro)>>4; pdec
      psym ')'
    finish
    ucb = lkcb0
    cycle
      if user(ucb)&x'7FFF' = pro
        space
        temp = ucb; cbname
        psym '!' if user(ucb) < 0
      finish
      ucb = ucb+4
      ucb = ucb+4 if ucb > tcb0
    repeat if ucb < cblim
    newline
  repeat if pro # 0
  jump kbidle

$end teletype

$begin filestore servicer process
 $def fxno=r5, fr=r6, ft=r7
 $def alt=r8, buffr=r9, pos=r10, lim=r11
$def usr=r12, comm=r13, work=r14, ptr=r15

 /Commands
/ buff empty/not          'ended'
 / 0/2   read/ wback    8/10   err/wback
 / 1/3   rback/write    9/11   err/write
/   4    reset (->0)     12    reset (->0)
 /  5    reset (->1)     13    reset (->1)
 /  6    close           14    close
/   7    close           15    uclose
 $def fcode=b(*)
   b 'x','i','z','y','u','u','k','k'
   b  0,  0, 'z','y','u','u','k','h'
 tempfile:
   b 4 ',,9E'

 routine resp,work
   response
   ac = 0 if ac = x'801B'       /(*reading past end*)
   jump out if ac >= 0
   ac = ac!x'4000'              /inhibit printing of name
endit:
   ac(usr) = ac
   pc(usr) = #ferrfail
done:
   pointer(cb) = buffr          /reset pointer
   frelease
   lpsw w(*+4)
   0, *+2                       /ints off
   siguser
   stop

 dkent:                         /ints on (but privileged)
   fclaim                       /get links
   lm usr,user(cb)              /user,devinf,x,pointer
   usr = usr&x'7FFF'
restart:
   devinf(cb) = comm&x'1FFF'
   fxno = comm&63
   comm = comm>>6; alt = comm&63 /alt xno
   comm = comm>>6               /fcommand
   buffr = ptr&bmask            /buff start (=ptr if empty)
   comm = comm&1+2 if buffr # ptr and comm&6 = 0
   work = fcode(comm)           /filestore code
```

```
    usr, #rate, 0, 0
    0, 0, *-12, pcb0, 16, tcb0-12, x'7FFF', 1
    8 $ 0
    0, s>>8

  $list ltemp

  $begin basic supervisor
  $temp r15
  $def pb=r12, sw1=r13, sw2=r14, temp=r15
  $def newpb=r13, cb=r14          /note equivalences
  $def sercb=r3                   /service queue

  /Interrupt service
  intser:
    usave = pb                    /free a register
    pb = pbase                    /current process base
    stm r0,r0(pb)                 /save registers
    pb(pb) = usave
    lm r14,intpsw                 /save intpsw as psw
    stm r14,ps(pb)
    ai cb,temp                    /acknowledge interrupt
    oc cb,disable
    newpb = user(cb)              /user waiting?
    jump w(*) if newpb >= 0       /no <=>
    newpb = newpb-s               /remove wait bit
    user(cb) = newpb
    link(newpb) = pb              /push-down current pro
  svgo:
    sw2 = pbit(newpb)\proset
    temp = 1
    wn temp,pbit(newpb); wh temp,sw2
    pbase = newpb
    lm r0,r0(newpb)               /restore all registers
    x'C200'                       /lpsw
  pbase:0                         /*impure*


  svnout:
    svcpsw[1] = 1                 /cc non-zero
  svout:
    lm r12,usave
    lpsw svcpsw

  /SVC   c l a i m    resource (class)
  /sign bit means no waiting
  /r1 <-resource (neg if unavailable)
  svclaim:
    stm r12,usave
    pb = pbase
    r1 = svcarg[1]
    cycle
      if user(r1) = 0             /resource available
        user(r1) = pb
        jumps svout
      finish
      r1 = r1+8                   /next disk file cb
    repeat if r1 > dcb0 and r1 < cblim
    r1 = svcarg
    jump svout if r1 < 0          /no waiting if neg

  /Add claimant to wait queue
  svq:
    sw1 = #waitq-link             /adjusted pointer
    cycle                         /Find end of queue
      sw2 = sw1
      sw1 = link(sw2)
    repeat if sw1 # 0
    link(sw2) = pb                /append current pro
  /Wait and re-try
  svwt0:
```

```
    1, 0                        /tt1 (tp)
    1, 0                        /lr1
    1, 0                        /lt1
    1, 0                        /tt2 (kb)
    1, 0                        /tt2 (tp)
    1, 0                        /lr2
    1, 0                        /lt2
    1, 0                        /tt3 (kb)
    1, 0                        /tt3 (tp)
    1, 0                        /lr3
    1, 0                        /lt3
    1, 0                        /tt4 (kb)
    1, 0                        /tt4 (tp)
    1, 0                        /lr4
    1, 0                        /lt4

/Buffered streams (8 bytes)

  /Terminal input
/user:limit:server:pointer
  $def tcb0=*-user, limit=devinf
    1, 0, 0, 0                  /set by init
    1, 0, 0, 0
    1, 0, 0, 0
    1, 0, 0, 0

  /Disk file control blocks
  /user:comm.alt.xno:server:pointer
/NB buffers must be on 512 byte boundaries
  $def dcb0=*-user
    0, 0, #dbase, reslim        /to cover initial code
    1, 0, #dbase, 0             / set by init
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
    1, 0, #dbase, 0
$$def bg=*-user
    0, 0, #dbase, 0
  $def cblim=*-user
    0, 0, #dbase, 0

  /Built-in process kernels

  /Filestore driver
  dbase:
    x'4000', #dkent, 0, reslim+512
    0, 0, *-12, 0, 0, 0, fsrdev, fstdev
    0, 0, 0, 0, 0, 0, 0, 0
    0, s>>9, *+4, *+10, 4 $ 0

  ibase:
    usr, #idle, 0, 0
    0, 0, *-12, pcb0, 16, tcb0-12, x'7FFF', 1
    8 $ 0
    0, s>>8

  $list ltemp

  $begin basic supervisor
  $temp r15
```

```
tobstore:0                      /set by init
 bgstart:0                      /"
 fskill:0
   0                            /spare
 proset:x'C0'
 $def tbase0=reslim+512
ubase0:0                        /set by init
 pblim:0                        /"
   0                            /spare

 /Svc area (x'94')

svcarg:0                        /svc argument
 svcpsw:0, 0                    /svc psw
   0                            /svc status (new)
   #svcall                      /svc 0
   #svret                       /ssvc 1
   #svpush                      /ssvc 2
   #svpop                       /ssvc 3
   #svclaim                     /svc 4
   #svrel                       /svc 5
   #svser                       /svc 6
   #svsel                       /svc 7
   #svnsym                      /svc 8
   #svrsym                      /svc 9
   #svpsym                      /svc 10
   #svprom                      /svc 11
   #svc12                       /svc 12
   #ssvcall                     /ssvc 13
   #ssvcall                     /ssvc 14
   #ssvcall                     /ssvc 15

 waitq:0                        /wait queue
null:0                          /empty string

 /Legos bootstrap (x'C0')
   r10 = #leg; r11 = #leg+14
   lm r12,w(x'74')              /rest of isys boot regs
   jump w(x'54')
leg:b 'Z,,ISYS80.LEGO', nl

 sysident:b 10,'ISYS80.V03'
lp:b 3,'lp.'
   3 $ 0                        /spare
 fsave:0
usave:0, 0, 0, 0

 $/Resource control blocks (x'100')

 $def ltemp=*list
 $list *list&(\1)

 $def user=*                    /ad of using process
 $def devinf=user+2             /(various) (+s for eof)
$def server=user+4              /ad of serving pro (or cb q)
 $def pointer=user+6            /buffer control pointer
 $def bmask=\(s+511)


 /Unbuffered streams (4 bytes)
/Null
   0, x'8000'
   0, x'8000'

 /Devices (* device-code = cb *)
 /user:devinf
$def lkcb0=*-user, fsr=lkcb0, fst=fsr+4
   0, 0                         /fsr
   0, 0                         /fst
$def kcb0=*-user, pcb0=kcb0+4, timer=#devinf(kcb0-pcb0)
   1, 0                         /ttl (kb)
```

```
  $assloc 1                          /0 & no loader codes

  $label command, despatch, abdn, abdn0, abdn1, kill
  $label ferrfail,svgo, syn0, pt0, pbase
  $def pcomm=call #pcomm0

  $/initial entry to system
  /Copy down from wherever loaded to location zero
  / unavoidably tortuous code to work even at loc 0
  $def jumpr1=x'0301'
    r4 = jumpr1+syslim-24
    r3 = 8
    r2 = jumpr1
    w(8) = r2
    bal r1,w(r3)
    lm r12,w(r1)
    stm r12,w(r2+x'10'-jumpr1)
    r1 = r1+8
    bxle r2,w(x'10')
    jump w(reslim)

  /Power-fail (x'22')
    0, 0, 0

  /Interrupt psws (x'28')
    0, 0, x'28', #illeg          /fp fault
    0, 0, x'30', #illeg          /illegal inst
    0, 0, x'38', #illeg          /machine error
intpsw:0, 0, 0, #intser          /external int
    0, 0, x'48', #illeg          /divide error


  $def fsrdev=x'08', fstdev=x'0C'

  /Isys bootstrap (x'50')
  $begin
  $def blk=r10, lt=r13, lr=r14
    lm blk,regs                  /(blk must be even reg)
    stm r11,w(x'34')             /will do
    lr = lr>>8                   /rx
    wb lt,blk                    /send request
    cycle
      ss lr,r0
    repeat if \clear
    rd lr,r0                     /n1
    jump w(x'7C')                /->autoload
  mess:b 'z,,isys80.z',n1
  regs:#mess                     /r10
    #mess+11                     /r11
    x'5A'                        /r12
    r0 = fstdev                  /r13 (& mini-boot)
    b fsrdev,x'80'               /r14 (at x'78')
    wd r0,r0                     /(mini-boot)
    x'D500'; x'00CF'             /autoload
  $end


  $def disable=b(x'79')
  $def enable=b(*), retrans=b(*+1)
  $def clerr=b(*+2), bit9=b(*+3)
    b x'40', x'20', x'10', x'01'
tonstore:0                       /set by init
  bgstart:0                      /"
  fskill:0
    0                            /spare
  proset:x'C0'
  $def tbase0=reslim+512
ubase0:0                         /set by init
  pblim:0                        /"
    0                            /spare
```

```
    jump out

  /New command: despatch process
  desp:
    erase                          /'n'
    restore
  despatch:
    siguser
    work = 0; temp = 0
    findpro
    if pro = 0                     /none available
      ptext; b'No go',nl,0
      jump kbidle
    finish
    exec(pro) = pb
    ttinout(pro) = inout(pb)
    inout0(pro) = ttinout(pro)
    kbin(pro) = kcb                /kb as device
    monout(pro) = 0
    cfparm(pro) = 0
    schedule
    proset = proset!ppit(pro)
    xstate[pb] = xstate[pb]+1
    jump kbidle

  kbattn:
    work = 0
    cycle
      findsubpro
      jump kbidle if pro = 0
      ucb = lkcb0+16
      cycle
        if user(ucb) = protss      /user waiting
          ac = cb; cb = ucb
          oc cb,disable
          seteot
          siguser
          cb = ac
        finish
        ucb = ucb+4
        ucb = ucb+8
      repeat if ucb < tcb0
      if errpos # 0                /abandon, kill
        if user(fsr)&x'7FFF' = pro
          if user(fst) = 0         /running remote program
            ac = fst; wd ac,attn
          else
            fskill = errpos
          finish
        else
          pc(pro) = errpos
        finish
      finish
    repeat

  state:
    work = 0
    cycle
      findsubpro
      if pro = 0
        ptext; b'free',0
      else
        psym 'p'
        ac = work; pdec
        ptext; b '(kb',0
        ac = kbin(pro)>>4; pdec
        psym ')'
      finish
      ucb = lkcb0
      cycle
```

```
      else
        psym 'p'
        ac = work; pdec
        ptext; b '(kb',0
        ac = kbin(pro)>>4; pdec
        psym ')'
      finish
      ucb = lkcb0
      cycle
        if user(ucb)&x'7FFF' = pro
          space
          temp = ucb; cbname
          psym '!' if user(ucb) < 0
        finish
        ucb = ucb+4
        ucb = ucb+4 if ucb > tcb0
      repeat if ucb < cblim
      newline
    repeat if pro # 0
    jump kbidle


$end teletype

$begin filestore servicer process
 $def fxno=r5, fr=r6, ft=r7
 $def alt=r8, buffr=r9, pos=r10, lim=r11
$def usr=r12, comm=r13, work=r14, ptr=r15

 /Commands
/ buff empty/not          'ended'
 / 0/2  read/ wback    8/10  err/wback
 / 1/3  rback/write    9/11  err/write
/  4    reset (->0)     12   reset (->0)
 /  5    reset (->1)     13   reset (->1)
 /  6    close           14   close
/  7    close           15   uclose
 $def fcode=b(*)
   b 'x','i','z','y','u','u','k','k'
   b  0,  0, 'z','y','u','u','k','h'
 tempfile:
   b 4 ',,9E'

 routine resp,work
   response
   ac = 0 if ac = x'801B'       /(*reading past end*)
   jump out if ac >= 0
   ac = ac!x'4000'              /inhibit printing of name
endit:
   ac(usr) = ac
   pc(usr) = #ferrfail
done:
   pointer(cb) = buffr          /reset pointer
   frelease
   lpsw w(*+4)
   0, *+2                       /ints off
   siguser
   stop

 dkent:                         /ints on (but privileged)
   fclaim                       /get links
   lm usr,user(cb)              /user,devinf,x,pointer
   usr = usr&x'7FFF'
restart:
   devinf(cb) = comm&x'1FFF'
   fxno = comm&63
   comm = comm>>6; alt = comm&63 /alt xno
   comm = comm>>6               /fcommand
   buffr = ptr&bmask            /buff start (=ptr if empty)
   comm = comm&1+2 if buffr # ptr and comm&6 = 0
   work = fcode(comm)           /filestore code
```

```
  ac = work
  jump endit if ac = 0
  jump switch if work = 'z'
  psym work
  psym ','
  ac = fxno; phex2                /transaction no
  if comm = 3                     /write
    psym ','
    ac = ptr&511; ac = 512 if ac = 0
    pos = buffr; lim = pos+ac
    phex
    resp
    cycle
      cycle
        ss ft,temp
      repeat if \clear
      wd ft,b(pos)
      pos = pos+1
    repeat if pos # lim
  else if comm < 2                /read, read back
    resp
    if ac # 0
      lim = buffr+512; pos = lim-ac
      buffr = pos+s               /new pointer value
      cycle
        cycle
          ss fr,temp
        repeat if \clear
        rd fr,b(pos)
        pos = pos+1
      repeat if pos # lim
    else
      jump swback if comm # 0 and alt # 0
    finish
    seteot if ac < 512
  else                            /close, reset
    if comm&2 # 0                 /close
      if alt # 0                  /hidden alt xno
        response                  /from first close
        psym 'k'; psym ','
        ac = alt; phex2
      finish
      devinf(cb) = 0
    else
      ac = ac(usr)
      psym ','
      phex
    finish
    resp
  finish
  jump done
switch:
  if alt = 0                      /first time
    exec(pb) = exec(usr)          /for puserno
    ac = 't'                      /open output
    prolog
    work = #tempfile              /(NB work = s1)
    pstring
    resp
    alt = ac
  finish
swback:
  comm = ((comm-1)<<6+fxno)<<6+alt /0->2->1, 1->0
  jump restart
$end filestore handler
$end server processes

$begin system routines
$temp r1
$def ac=r0, temp=r1, sym=temp, ph=r2
```

```
    comm = ((comm-1)<<6+fxno)<<6+alt /0->2->1, 1->0
    jump restart
$end filestore handler
 $end server processes

$begin system routines
 $temp r1
 $def ac=r0, temp=r1, sym=temp, ph=r2
$def inn1=r12, outn1=r13, s1=r14, s2=r15
 $def not=3, so=\3

routine restore,r3 /reset nest etc
    pb = pbase
    np(pb) = pb+usize
    nplim(pb) = pb+usize-20
    pbit(pb) = pbit(pb)&(\1)
    selout 0
    release fsr; release fst
    jump out

/Filestore operations

 /FCLAIM: claim filestore links
fclai0:
    claim fsr
    claim fst
    fsave = inout(pb)           /save current io
    inout(pb) = fsr<<8+fst      /set to filestore links
    devinf(fsr) = 0             /safety
    return

 /FRELEASE: release filestore links
frele0:
    inout(pb) = fsave
    temp = -fskill              /killed?
    fskill = fskill+temp        /zero
    release fsr; release fst
    temp = temp\x'FFFF'+1       /ie -temp
    jump w(temp) if temp # 0
    return

/PROLOG: print fs command letter and userno
 prlog0:
    temp = ac&255               /fs command code
    psym temp
    jumps puser1 if temp = '1'  /no userno for 'logon'
 /PUSERNO: print fs userno
puser0:
    psym ','
    ac = userno(exec(pb))
    phex2
 puser1:
    psym ','
    return

 /RESPONSE: read fs response no
respo0:
    newline
    match '-'
    if so
      rhex; ac = ac!s
    else
      rhex
    finish
    cycle
      rsym
    repeat if sym # nl
    return
```

```
 $def work=r13 /(saved)
/Print file-name (+ owner if work&512 # 0)
 nroutine pfname
   if work&512 # 0 and s2 # 0 and b(s2) # 0
     rll s1,16                    /swop s1::s2
     pstring
     rll s1,16
     psym '.'
   finish
   pstring
   return

 /FCOMM: filestore communication
/ac: operation code (fs command letter)
 /     + 256 to suppress error failure
 /     + 512 for separate owner-name in s2
/      +1024 for second parameter in s2
 /s1 [and s2]: name pointer[s]
 fc0:
   push work; work = ac
   fclaim
   prolog                         /command letter, user no
   pfname
   if work&1024 # 0               /second operand?
     psym ','
     rll s1,16                    /swop s1::s2
     pstring
     rll s1,16                    /restore
   finish
   response
   frelease
   temp = work; pop work          /restore
   jumps fserr if ac < 0
   return                         /cc \neg

 deverr:                          /from REFIN, REFOUT
   temp = ac
   ac = x'800B'                   /not found
 referr:                          /from REFIN, REFOUT
   pop work
/fserr:                           /from FCOMM
   temp = temp<<7                 /256 to sign
   return if temp < 0             /error suppressed =>
   rll s1,16 if ac = x'800A' or ac = x'8013'
 ferrfail:
   restore
   ferr
   fail

/FERR: print filestore error message
 ferr0:
   if ac&x'4000' = 0
     push ac
     pstring; space
     pop ac
   finish
   temp = ac&255
   jump f0(feswl[temp]) if temp <= x'13'
/f0:
   ptext; b 'fs err ', 0
   phex2
/f00:
   newline
   return
/f4:ptext; b 'incorrect',0
   jump f00
 fb:ptext; b 'not found',0
   jump f00
 fa:ptext; b 'exists',nl,0
   return
```

```
    newline
    return
f4:ptext; b 'incorrect',0
    jump f00
fb:ptext; b 'not found',0
    jump f00
fa:ptext; b 'exists',nl,0
    return
fd:ptext; b 'inaccessible',nl,0
    return
fe:ptext; b 'Quota', 0
fex:ptext; b ' exceeded',nl,0
    return
fef:ptext; b 'Files', 0
    jump fex
f10:ptext; b 'Extents', 0
    jump fex
fesw:
    b 0, 0, 0, 0,
    #f4-#f0, 0, 0, 0,
    0, 0, #fa-#f0, #fb-#f0,
    #fb-#f0, #fd-#f0, #fe-#f0, #fef-#f0,
    #f10-#f0, 0, 0, #fa-#f0


/TOD: time of day
tod0:
    fclaim
    ptext; b 'a',nl,0
    s2 = nl; rstring
    frelease
    return

$/Reference input [output] stream
/ac: 0,1,2,3 (+256 to suppress error)
/              (+512 for separate owner-name)
/       (preserved unless err)
/s1 [s2]: string ad (preserved)
/REFOUT
rout0:
    ac = ac!s
/+REFIN
rin0:
    push work
/   work = ac&(s+3) /(unsatisfactory: rethink)
/   svc 7,work+x'1000' /select
/   svc 7,work+x'6000' /close
    work = b(s1)                    /length
    temp = work
    jump refend if temp = 0
    if b(s1+1) # '.'                /Disk file
      temp = ac&x'F8'               /editor?
      temp = dcb0 if temp < dcb0
      claim temp
      push temp
      fclaim
      work = ac
      ac = ac>>15+'s' /input:'s', output:'t'
      prolog
      pfname
      response
      frelease
      pop temp
      if ac < 0                     /error
        release temp
        temp = work                 /saved ac
        jump referr
      finish
      ac = ac+(1<<12) if work < 0 /disk write
      devinf(temp) = ac             /fcomm, fxno
```

```
    ac = work
  else                              /Device
    if work = 1                     /just '.'
      temp = inout0(pb)
    else
      temp = b(s1+2)&95             /first letter
      if temp = 'N'
        temp = 0
      else if temp = 'T'
        temp = ttinout(pb)
      else
        jump deverr if temp # 'L'
        work = (b(s1+4)-'0')<<4
        jump deverr if work <= 0 or work >= tcb0
        temp = (fsr+work)<<8+fst+work
      finish
    finish
    temp = temp>>8 if ac >= 0 /input
    temp = temp&255
    if temp&8 # 0                   /link - dedicated device
      claim temp if user(temp) # pb
    finish
  finish
  work = ac&3<<1+pb
  work = work+1 if ac < 0
  in0(work) = temp
refend:
  work = ac&(s+3)
  svc 7,work+x'1000'                /select (1000 in case old prog)
  pop work
  ac = ac&(\s)
  return

/REFALL: Reference all streams
refl0:
  s1 = #inname(pb)
  ac = 1
  cycle
    refin ac,s1
    s1 = s1+20
    ac = ac+1
  repeat if ac < 4
  ac = 1
  cycle
    refout ac,s1
    s1 = s1+20
    ac = ac+1
  repeat if ac < 4
  selin 1; selout 1
  return

/INSTREAM
inst0:
  ac = insno(pb)
inst1:
  ac = ac>>1
  return
/OUTSTREAM
outst0:
  ac = outsno(pb)
  jump inst1

$/Close output stream (old version)
oclout0:
  ac = acts

/Close (input) stream (old version)
oclin0:
  temp = ac; svc 7,temp+x'1000' /select stream
  jump clout0 if ac < 0
```

```
 $/Close output stream (old version)
 oclout0:
   ac = acts

 /Close (input) stream (old version)
 oclin0:
   temp = ac; svc 7,temp+x'1000' /select stream
   jump clout0 if ac < 0

 /CLOSIN: close current input stream
 clin0:
   ac = 0
   in0(insno(pb)+pb) = ac
   temp = incb(pb)
   release temp if temp # in0(pb)
   incb(pb) = 0
   return

 /CLOSOUT: close current output stream
 clout0:
   ac = 0
   in0(outsno(pb)+pb) = ac
   temp = outcb(pb)
   if temp >= dcb0               /disk
     ac = pointer(temp)&(\bmask)
     service temp if ac # 0      /write last block
     ac = devinf(temp)
     if ac # 0
       ac = ac!x'6000'           /close
       devinf(temp) = ac
       service temp
     finish
   finish
   if temp < kcb0                /link
     psvm steot
   finish
   release temp
   outcb(pb) = 0
   return

 /RESETIN: reset current input stream
 rsin0:
   ac = 0
 setin0:
   svc 7,x'4000'
   return

 /RESETOUT: reset current output stream
 rsout0:
   ac = 0
   svc 7,x'4000'+s
   return

 $/Process termination
 $def err=r13
 $redef work=r15 /same as Supervisor temp

 /Machine interrupt, usually illegal inst
 illeg:
   epsr temp,temp                /get int address
   ac = w(temp)                  /get users status
   work = w(temp+2)              /and address
   jump w(*) if ac&x'0100' = 0 /system - fatal
   err = 'i'<<8+'i'
   jumps abdn1

 /ABANDON
 abdn:
   err = 'a'<<8+'*'
 abdn0:
```

```
  work = 0
abdn1:
  restore
  resetcursor
  psym '*'
  if err >= 0
    sym = err>>8
    psym sym
    sym = err&255
    psym sym
  else
    r1 = r14                    /*r14 is Supervisor cb
    cbname
  finish
  space
  ac = work; phex
  newline

 /+FAIL
fai0:
  work = #out0(pb+6)            /out slot 3
  cycle
    temp = b(work)
    release temp                /prevent normal closing
    b(work) = 0
    work = work-2
  repeat if work > #out0(pb)
  jumps ign
kill:
  work = 3
  cycle
    selout work
    resetout
    work = work-1
  repeat if work > 0

 ign:
  restore
  resetcursor
  selin 0
  ignore

 /STOP: close streams and terminate
stop0:
  restore
  work = 3                      /close streams
  cycle
    selin work
    closin
    selout work
    closout
    work = work-1
  repeat if work > 0
  release bg
  if cfparm(pb) = 0             /not obeying command file
    work = dcb0                 /release buffers
    cycle
      release work
      work = work+8
    repeat if work <= cblim
    release -1                  /stop
  finish
  jump command

/NSIZE: define nest size
 ns0:
  clh ac,176
  if less
    nplim(pb) = pb+usize-ac
    return
```

```
    finish
    jump command

/NSIZE: define nest size
 ns0:
    clh ac,176
    if less
      nplim(pb) = pb+usize-ac
      return
    finish
    work = ac
    ac = 'n'<<8+'e'
    jump abdn1

 $/Input/output routines

 $def clearcc=x'0511'           /clhr r1,r1
 $temp                          /temp,sym used explicitly

 /IGNORE: skip pending input (not good enough)
 ig0:cycle
    temp = inch(pb)
    if temp = ttin(pb)
      ac = pointer(temp)
      if ac = limit(temp)       /no data available
        ac = server(temp)
        return if ac >= #tbase0 /server not active =>
      finish
    finish
    rsym
    repeat if sym >= 0 and sym # '}'
    return

/SKIP: skip spaces
 sk0:match ' '
    jump sk0 if zero
    return

 /ATTEND: get attention request
att0:temp = exec(pb)
    ac = -attn(temp)
    attn(temp) = attn(temp)+ac  /clear
    temp = -ac
    ac = temp                   /(setting cc)
    return

 $temp r1
 /RESETCURSOR: set video cursor to bottom of screen
rcurs0:
    ac = vlast<<8
 /SETCURSOR: set video cursor
scurs0:
    psym esc
    if vtype(exec(pb)) > 0     /VT52
      psym 'Y'
      sym = ac<<3>>11; psym sym+' '
    else                        /Bantam
      psym 'X'
      sym = ac<<3>>11; psym sym+' '
      psym esc; psym 'Y'
    finish
    sym = ac&255
    sym = 79 if sym > 79
    psym sym+' '
    devinf(ttout(pb)) = ac
    return

 /CLEARLINE: clear rest of video line
 clear0:
    psym esc
    if vtype(exec(pb)) > 0      /VT52
```

```
    work = 0
abdn1:
    restore
    resetcursor
    psym '*'
    if err >= 0
      sym = err>>8
      psym sym
      sym = err&255
      psym sym
    else
      r1 = r14                    /*r14 is Supervisor cb
      cbname
    finish
    space
    ac = work; phex
    newline

  /+FAIL
fai0:
    work = #out0(pb+6)            /out slot 3
    cycle
      temp = b(work)
      release temp                /prevent normal closing
      b(work) = 0
      work = work-2
    repeat if work > #out0(pb)
    jumps ign
kill:
    work = 3
    cycle
      selout work
      resetout
      work = work-1
    repeat if work > 0

  ign:
    restore
    resetcursor
    selin 0
    ignore

  /STOP: close streams and terminate
stop0:
    restore
    work = 3                      /close streams
    cycle
      selin work
      closin
      selout work
      closout
      work = work-1
    repeat if work > 0
    release bg
    if cfparm(pb) = 0             /not obeying command file
      work = dcb0                 /release buffers
      cycle
        release work
        work = work+8
      repeat if work <= cblim
      release -1                  /stop
    finish
    jump command

/NSIZE: define nest size
  ns0:
    clh ac,176
    if less
      nplim(pb) = pb+usize-ac
      return
```

```
   jump command

/NSIZE: define nest size
 ns0:
   clh ac,176
   if less
     nplim(pb) = pb+usize-ac
     return
   finish
   work = ac
   ac = 'n'<<8+'e'
   jump abdn1

 $/Input/output routines

 $def clearcc=x'0511'          /clhr r1,r1
 $temp                         /temp,sym used explicitly

 /IGNORE: skip pending input (not good enough)
 ig0:cycle
   temp = inch(pb)
   if temp = ttin(pb)
     ac = pointer(temp)
     if ac = limit(temp)       /no data available
       ac = server(temp)
       return if ac >= #tbase0 /server not active =>
     finish
   finish
   rsvm
   repeat if sym >= 0 and sym # '}'
   return

/SKIP: skip spaces
 sk0:match ' '
   jump sk0 if zero
   return

 /ATTEND: get attention request
att0:temp = exec(pb)
   ac = -attn(temp)
   attn(temp) = attn(temp)+ac  /clear
   temp = -ac
   ac = temp                   /(setting cc)
   return

 $temp r1
 /RESETCURSOR: set video cursor to bottom of screen
rcurs0:
   ac = vlast<<8
 /SETCURSOR: set video cursor
scurs0:
   psvm esc
   if vtype(exec(pb)) > 0     /VT52
     psvm 'Y'
     sym = ac<<3>>11; psym sym+' '
   else                       /Bantam
     psvm 'X'
     sym = ac<<3>>11; psym sym+' '
     psvm esc; psvm 'Y'
   finish
   sym = ac&255
   sym = 79 if sym > 79
   psvm sym+' '
   devinf(ttout(pb)) = ac
   return

 /CLEARLINE: clear rest of video line
 clear0:
   psvm esc
   if vtype(exec(pb)) > 0       /VT52
```

```
    psym 'x'
  else                              /Bantam
    ptext; b 'I',1,0 /(plus filler)
  finish
  ac = -2 /(esc is counted)
  devinf(ttout(pb)) = devinf(ttout(pb))+ac
  return

$temp

/TESTDIG: test digit
 /Read and map to 0:9 if so
td0:nsym
  if sym >= '0' and sym <= '9' /*cc non-z if false*
    rsym
    sym = sym-'0'
    clearcc
  finish
  return

 /TESTLET: test letter
 /Read and map to 0:25 if so
tl0:nsym
  if sym >= 'A'
    sym = sym&95
    if sym <= 'Z'
      rsym
      sym = sym&95-'A'
      clearcc
    finish
  finish
  return

 /RHEX: read hex
rh0:skip; ac = 0
  cycle
    testdig
    if \zero
      sym = sym&95
      testlet if sym <= 'F'   /*cc non-z if false*
      return if \zero
      sym = sym+10
    finish
    ac = ac<<4+sym
  repeat

/PHEX: print hex
 ph2:ac = ac<<8+x'80'
  sym = 0
  jumps ph1
 ph0:sym = x'8000'              /marker
 ph1:cycle
  rll ac,4                      /rotate next digit to sym
  return if ac = 0              /finished ->
  sym = sym-9
  sym = sym+7 if sym > 0        /'A':'F'
  psym sym+'9'
  sym = 0
  repeat

 /CBNAME: print cb mnemonic
$def cb=r1
 cbn0:
  if cb < tcb0                  /Device
    if cb&8 = 0                 /tt not link
      psym 't'
    else
      psym 'l'
    finish
    if cb&4 = 0                 /in not out
```

```
cbn0:
  if cb < tcb0                    /device
    if cb&8 = 0                   /tt not link
      psym 't'
    else
      psym 'l'
    finish
    if cb&4 = 0                   /in not out
      psym 'r'
    else
      psym 't'
    finish
    ac = cb>>4
  else if cb < dcb0
    ac = (cb-tcb0+8)>>3
    ptext; b'kb',0
  else if cb # bg
    psym 'b'
    ac = (cb-dcb0+8)>>3
  else
    ptext; b 'bg=',0
    ac = topstore-bgstart+1
  finish
  pdec
  return

/RDEC: read decimal
rd0:skip; ac = 0
  cycle
    testdig
    return if \zero
    ac = ac<<1
    sym = sym+ac
    ac = ac<<2+sym                /10*ac+sym
  repeat

/PDEC: print decimal
pd0:sym = ac
  if sym # 0
    ac = 0; dh ac,c10000
    ac = ac<<1+1                  /2*rem + flag
    while sym = 0                 /num < 10000
      sym = ac; ac = 5
      mhu ac,ac                   /complete mult by 10
      dh ac,c10000
      ac = ac<<1
    repeat
  finish.
  cycle
    psym sym+'0'
    sym = ac; ac = 5
    mhu ac,ac
    dh ac,c10000
    ac = ac<<1
  repeat if ac # 0
  return
c10000:10000

/RSTRING: read string
/s1: dest ad [preserved]
/s2: terminator [preserved]
rs0:ac = 0
  cycle
    rsym
    exit if sym <= s2
    s1 = s1+1; ac = ac+1
    b(s1) = sym
  repeat if s1 # pp(pb) /safety
  s1 = s1-ac; b(s1) = ac
  return
```

```
/PSTRING: print string
 /s1: string ad [preserved]
 ps0:ac = 0
   while ac # b(s1)
     ac = ac+1; sym = b(s1+ac)
     psym sym
   repeat
   return

/PTEXT: print text
 pt0:
   temp = pp(pbase)              /Swap nest,r3 (can't use pop,push)
   r3 = r3-w(temp)               /a-b
   w(temp) = w(temp)+r3          /b+a-b = a
   temp = w(temp)-r3             /a-(a-b) = b
   cycle
     r3 = b(temp); temp = temp+1
     exit if r3 = 0
     psym r3
   repeat
   temp = (temp+1)>>1<<1         /even up
   pop r3
   jump w(temp)

/NEWLINE
 nl0:psym nl
   return

 /SPACE
 sp0:psym ' '
   return

 $temp r1

 nroutine crossref
   sym = sym-'1'
   if sym >= 0 and sym < 6
     sym = sym<<2; s1 = s1+sym /+ *4
     sym = sym<<2; s1 = s1+sym /+ *16 = + *20
     return
   finish
   ac = sym+('='<<8+'1')
   jump abdn0

 /RNAME: read name
/s1: dest ad (preserved)
 rn0:skip; ac = 0
   if sym = '%'                  /parm macro
     push s2; s2 = s1
     s1 = cfparm[pb]
     jump syn0 if s1 = 0
     s1 = pointer(s1)&bmask
     rsym
     rsym
     crossref
     match '='                   /redundant
     ac = 0
     while ac # b(s1)
       s2 = s2+1; ac = ac+1
       b(s2) = b(s1+ac)
     repeat
     s1 = s2; pop s2
     jump rn0 if ac = 0
     ac = ac+256
   finish
   cycle
     nsym
     exit if sym <= ' ' or sym = ',' or sym = '/' or sym = ';'
     rsym
```

```
    s1 = s2; pop s2
    jump rn0 if ac = 0
    ac = ac+256
  finish
  cycle
    nsym
    exit if sym <= ' ' or sym = ',' or sym = '/' or sym = ';'
    rsym
    sym = sym!32                    /standardise case
    if ac < 19
      b(s1+1) = sym
      s1 = s1+1; ac = ac+1
    finish
  repeat
  ac = ac&255
  s1 = s1-ac; b(s1) = ac
  return

/RPARM: read parameters
$def count=s2, terminator=r9
rp0:s1 = #inname(pb)
  inn1 = s1
  cycle
    count = 3
    cycle
      rname
      s1 = s1+20
      count = count-1
      exit if count = 0
      match ','
    repeat
    skip
    jump syn0 if sym = ','
    exit if s1 = #progname(pb)
    outn1 = s1
    match '/'
  repeat
  rsym                              /nl or ';'
  terminator = sym
  selout 5                          /monitor output
  psym '{'
  jump rp5

/Print command
pcomm0:
  s1 = #progname(pb)
  pstring; space                    /progname
  s1 = inn1; s2 = s1
  cycle
    if b(s1) # 0
      if s1 >= outn1 and outn1 > s2 /first output
        psym '/'; s2 = outn1
      finish
      while s2 < s1
        psym ','
        s2 = s2+20
      repeat
      pstring
    finish
    s1 = s1+20
  repeat if s1 # #progname(pb)
  return

rp5:
  pcomm
  newline
  selout 0
/+ENVINFO [r0 preserved]
env0:
  r5 = exec(pb)
```

```
    r6 = #uname(r5)
    inn1 = #inname(pb)
    outn1 = #outname(pb)
    s1 = inn1; r11 = 0          /set up parm types
    cycle
      if s1 = outn1
        r10 = r11; r11 = 0
      finish
      s2 = b(s1)                /length
      s2 = 8 if s2 # 0 and b(s1+1) # '.' /file
      r11 = r11<<4+s2
      s1 = s1+2
    repeat if s1 # #progname(pb)
    s1 = inn1; s2 = outn1
    return

syn0:
    selout 0
    ptext; b 'En? ',0
    cycle
      rsym
      psym sym
    repeat if sym >= ' '
    fail

/SMOVE: string move  --  s1 -> s2
 sm0:ac = b(s1)+1               /length+1
 /BMOVE: bulk move --  ac bytes, s1 -> s2; all updated
bm0:
    while ac > 0
      b(s2) = b(s1)
      s1 = s1+1; s2 = s2+1
      ac = ac-1
    repeat
    return

 /SAPPEND: string append
sa0:
 /  ac = b(s1)
 /  temp = b(s2)+ac
 /  b(s2) = temp
 /  s2 = s2+temp-ac+1
 /  s1 = s1+1
 /  jump bm0

 /SCOMP: string compare
sc0:ac = 0
   cycle
     exit if b(s1+ac) # b(s2)   /cc non-zero
     exit if b(s1) = ac         /cc zero
     ac = ac+1; s2 = s2+1
   repeat
   return

 /SFIND: find symbol in string
/s1: string ad [preserved]
 /s2: symbol [preserved]
 /ac <- length of pre-string (-1 if symbol not found)
sf0:ac = b(s1) /length
   cycle
     ac = ac-1
   repeat if ac >= 0 and b(s1+ac+1) # s2
   return                       /cc z iff found

/DEFAULT: apply defaults to parm strings
 $def p=r5, d=r6, dmax=r8 /(xb,unam,cnam)
 $def types=r10

def10:
   envinfo                      /to set inn1 etc
```

```
  return                          /cc z iff found

/DEFAULT: apply defaults to parm strings
$def p=r5, d=r6, dmax=r8 /(xb,unam,cnam)
$def types=r10

def10:
  envinfo                         /to set inn1 etc
  d = ac; dmax = b(d)+d
  p = inn1
  cycle
    types = types<<4
    s1 = 0
    cycle
      ac = 0
      ac = b(d+1) if d # dmax
      exit if ac # '=' and ac # '#'
      s1 = inn1; sym = b(d+2)
      crossref
      d = d+2
      if ac = '#'
        if types < 0              /file
          s2 = p; scomp
          if so
            selout 0
            ptext; b 'Overwrite ',0
            pstring; psym '?'
            newline
            fail
          finish
        finish
        s1 = 0
      finish
    repeat
    s2 = b(p)+p                   /end of parm string
    if s2 = p                     /parm string empty
      if s1 # 0                   /crossref present
        smove; s2 = s2-1
      finish
      ac = 1                      /take default extension
    else
      sym = ac; ac = 0
      if sym = ':'
        s1 = s2
        cycle
          s1 = s1-1
          ac = b(s1+1)-':'
        repeat if ac # 0 and s1 # p
      finish
    finish
    while d # dmax
      d = d+1
      sym = b(d)
      exit if sym = ',' or sym = '/'
      if ac # 0
        s2 = s2+1 if s2 # p+19
        b(s2) = sym
      finish
    repeat
    s2 = s2-p
    b(p) = s2
    p = p+20
    if sym = '/'
      p = outn1; types = r11
    finish
  repeat if p # #progname(pb)
  jump env0

/VEI: data vet
```

```
 $temp                          /sym used explicitly
$def sp=r14, gp=r15             /stack pointer, gram pointer
 $def gstack=store, stsize=6
 $def pcall=9

 dvok:
   sym = sym<<8                 /Test for unambiguous end
   ac = ac+256 if sym = b(gp) and sp = #gstack(pb) /(ouch)
   sym = sym>>8
   gp = gp-gram(pb)             /base relative
   clearcc
 dvout:
   stm sp,save(pb+4)            /preserving ...
   jumps dvend
 dvfail:
   ac = -2
dvend:
   lm r14,save(pb)              /   ... cc
   return

 dv0:
   stm r14,save(pb)
   sp = #gstack(pb)
   gp = ac-256
   lm sp,save(pb+4) if gp < 0   /not initial call
   cycle
     gp = gp+gram(pb)           /absolute
     ac = b(gp)                 /next component
     gp = gp+2                  /assume 2 bytes
     if ac >= pcall
       if ac > pcall            /symbol match
         ac = ac!32 if sym >= 96
         jump dvok if ac = sym /direct match
         if ac&128 # 0          /range
           gp = gp+1            /3 bytes
           if ac <= sym+128     /min<=sym
             ac = b(gp-2)
             ac = ac!32 if sym >= 96
             jump dvok if ac >= sym+128 /sym<=max ->
           finish
         finish                 /no match
       else                     /start of phrase
         jump dvfail if sp = #gstack(pb+stsize)
         ac = gp-gram(pb)       /base relative
         b(sp) = ac; sp = sp+1
       finish
       gp = b(gp-1)             /alternative or phrase-start
       jump dvfail if gp = 255 /no alternative ->
     else if ac # 0             /user action
       gp = gp-1                /1 byte only
       gp = gp-gram(pb)         /base relative (cc g)
       jump dvout
     else                       /end of phrase
       if sp = #gstack(pb)      /stack empty
         ac = -1                /termination
         jump dvend
       finish
       sp = sp-1; gp = b(sp)
     finish
   repeat
 $temp r1



/grammar for streams
 $macro phrase n
 $def n=*-#sgram
$redef p=n
 $end
 $redef p=0
```

```
/grammar for streams
 $macro phrase n
 $def n=*-#sgram
$redef p=n
 $end
 $redef p=0

 sgram:
 /  l = 'A'-'Z'     d = '0'-'9'
/ident ->  (l,d)*
 phrase ident
 b 'A'+128,'Z'+128,p+8,
   '0'+128,'9'+128,p+0,
   11,p+3,
   0
/name -> ("." (l(ld?)?)?, "$" ident, ident ("." ident)?)
 /           (":" ident)*
 phrase name
b ' ',p+4,
   11,p+0,
   '.',p+17,
   'A'+128,'Z'+128,p+15,
   'A'+128,'Z'+128,p+15,
   '0'+128,'9'+128,p+15,
   11,p+29,
   '$',p+23,
   pcall,ident+3,
   11,p+29,
   pcall,ident,
   '.',p+29,
   pcall,ident,
   ':',p+35,
   pcall,ident+3,
   11,p+29,
   0
 phrase group
b pcall,name,
   ',',p+10,
   pcall,name,
   ',',p+10,
   pcall,name,
   0
phrase comm
 b '@',p+2,
   pcall,name,
   ' ',p+12
 phrase parm
 b pcall,group,
   '/',p+6,
   pcall,group,
   ';',p+13,
   ' '+128,126+128,p+13,
   11,p+8,
   nl,255,
   0

 /Match input to format
/ ac: phrase letter
 / s1: dest
 /fm0:ac = ac!25o                /initial call to vet
/   work = s1
 /  cycle
 /    nsym
/     cycle
 /      vet
 /    repeat if g
/     exit if \zero
 /    rsym
```

```
/     if sym #
/       work = work+1
/       work = work-1 if work = np(pb)
/       b(work) = sym
/     finish
/   repeat if ac&256 = 0
/   b(s1) = work-s1          /enter length of string
/   if ac > 0
/     temp = 0
/   else
/     ac = ac+1
/   finish
/   return

 $/resident utilities
 $temp r1
/$def ac=r0, temp=r1, sym=r1, pb=r2
 $def xb=r5, unam=r6
 /$def terminator=r9, intypes=r10, outtypes=r11
$def nerr=256, owner=512, double=1024

 $macro move s[s1],d[s2]
   s1=s; s2=d
   smove
 $end
$macro comp st1[s1],st2[s2]
   s1=st1; s2=st2
   scomp
$end

 /command table
ctab:
 b 3,'set', 6,'delete', 6,'rename', 6,'reperm',
   0, 1,'e', 1,'s', 3,'lib',
   4,'echo', 1,'p', 5,'start', 4,'stop',
   5,'quote', 3,'spy', 3,'who',
   5,'legos', 5,'logon', 6,'logoff',
   1,'l', 5,'empty', 3,'tod', 1,'t'
 cswitch:
   'j', 'd', 'b'+double, 'e'+double
$def fclim=*
   #xfin, #ed, #show, #libry
   #echo, #print, #logon, #logoff
   #quote, #spy, #who
   x'C0', #logon, #logoff
   #show, #empty, #todp, #trans
$def entlim=*, clim=entlim

 $label load


 $begin command processor

 /ac=r0,temp=r1,pb=r2,xb=r5,unam=r6
 /terminator=r9,intypes=r10,outtypes=r11
$def work=r3, count=r4, pos=r7
 $def inout1=inout0+2, inout2=inout1+2, inout3=inout2+2
 $def ok=\neg


 system:b 6,'isys80'

 cprom:b 3,'}',soh,comm

/Obey command file
 cfile:
   nsym                          /provoke input
   ac = in0(pb)+cfparm(pb)       /save current in0, cfparm
   work = in1(pb)
   in0(pb) = work
```

```
cprom:b 3,'}',soh,comm

/Obey command file
cfile:
  nsym                          /provoke input
  ac = in0(pb)+cfparm(pb)       /save current in0, cfparm
  work = in1(pb)
  in0(pb) = work
  if pointer(work)&511 < 122    /not enough space
    claim dcb0
    work = temp
  finish
  s2 = pointer(work)&bmask      /buffer start (dest)
  w(s2+120) = ac
  cfparm(pb) = work
  s1 = inn1; ac = 120           /source, parm size
  bmove

/+ initial entry
command:
  pb = phase
  gram(pb) = #sgram
  inout1(pb) = inout0(pb)
  inout2(pb) = 0
  inout3(pb) = 0
  selin 0
  selout 0
  prompt #cprom
  s1 = #progname(pb)
  match '@'                     /test for 'remote' marker
  if so and topstore > 0
    rname                       /command word
    rparm
    fclaim
    ac = '@'; prolog
    pcomm                       /print command
    response
    if ac < 0
      frelease
      ferr
      fail
    finish
    release fst      /convention: to allow killing
    selout 0                    /relay report output to tt
    cycle
      rsym
      sym = sym&255
      exit if sym = eot
      psym sym
    repeat
    frelease
    stop
  finish
  rname
  s2 = #ctab; pos = #cswitch    /Look up
  cycle
    scomp
    exit if so
    cycle
      s2 = s2+1
    repeat if b(s2) > 32
    pos = pos+2
  repeat if pos # clim
  rparm                         /read parameters
  prompt 0
  ac = w(pos)
  if pos < fclim
    fcomm
    stop
  finish
```

```
  jump w(ac) if pos < entlim

/not resident function
  s1 = #progname(pb)
  s2 = '.'                      /test owner-name present
  sfind
  work = ac
  if work < 0                   /none present
    s2 = #system                /try ISYS80 first
  else
    s2 = 0
  finish
  count = b(s1)                 /length
  b(s1) = count+3               /for extension
  pos = s1+count                /end of name
  b(pos+1) = ':'                /for ':xi', ':ci'
  b(pos+3) = 'i'
  cycle
    b(pos+2) = 'x'              /':xi'
    refin 3+nerr+owner,s1       /try as execute file
    jump load if ok     /found ->
    b(pos+2) = 'c'              /':ci'
    refin 1+nerr+owner,s1       /try as command file
    jump cfile if ok    /found ->
    exit if work >= 0
    s2 = #libname(xb)
    work = 1
  repeat
  b(s1) = count                 /restore
  if s2 = 0                     /explicit owner-name
    refin 3+nerr+owner,s1       /last resort: no extension
    jump load if ok
  finish
  ptext; b 'Command ',0
  ferr
  fail

/ error reports
 exerno:
  ptext; b'No!',nl,0
  fail

 pprom:b 7,' Pass:',stx

 mail:b 4,'mail'
 off:b 3,'off'

 routine readpass,work
  if b(s1) = 0                  /no pass in command
    prompt #pprom
    rname
    match nl
    jump syn0 if not
  finish
  jump out

 logon:
 logoff:
  if userno[xb] # 0             /currently logged on
    jump exerno if xstate[xb] # 1
    s1 = #null
    ac = nerr+'m'; fcomm            /logoff
    ferr if neg
  finish
  b(unam) = 0
  libname[xb] = 0
  userno[xb] = 0
  monout(pb) = 0
  if b(inn1) = 0                /no (new) username
    newline; stop
```

```
    ferr if neg
  finish
  b(unam) = 0
  libname[xb] = 0
  userno[xb] = 0
  monout(pb) = 0
  if b(inn1) = 0                /no (new) username
    newline; stop
  finish
  s1 = inn1+20                  /input name 2
  readpass
  s2 = s1; s1 = inn1
  ac = '1'+double; fcomm        /logon
  userno[xb] = ac
  move s1,unam
  move unam,#libname(xb)
  s1 = #sysident
  refin 1,s1                    /system message
  pstring; space
  s1 = inn1; s2 = -1
  rstring; pstring

  s2 = #mail
  refin 2+nerrtowner,unam
  if ok
    ptext; b 'You have mail', nl, 0
  finish
xfin:
  stop

who:
  s1 = unam
ps1:
  pstring; newline
  stop

todp:
  tod
  jump ps1

libry:
  move s1,#libname(xb)
  stop

echo:
  work = 0
  comp s1,#off
  work = ttout(pb) if not
  monout(pb) = work
  stop

quote:
  readpass
  ac = 'q'; fcomm
  stop

empty:
  fail if b(s1) = 0
  stop

/Spy
$def ad=r8
sprom:b 1,'/'
spy:prompt #sprom
  match nl
  stop if so
  rhex
  ad = ac
  prompt 0
  count = 1
```

```
    cycle
      skip
      match nl
      if not
        rdec
        rsym
        count = ac
        jump spy if count = 0
      finish
      ac = ad; phex
      psym ':'
      work = count
      cycle
        ac = w(ad)
        phex; space
        ad = ad+2; work = work-1
      repeat if work > 0
      match '='
      if so
        rhex
        w(ad-2) = ac
      finish
    repeat

  $end command processing

  $begin loader
  $def ac=r0,pend=r0, sym=r1,item=r1, lstart=r3
  $def loc=r4, format=r6, lend=r7
  $def word=r8
  $def ra=r14, rb=r15

  /Read 4-bit item
  $temp                                 /item=sym=r1
  routine getitem,ra
    item = pend
    if item < 0
      rsym
      jump moan if sym < 0 /premature end-of-file =>
      pend = sym&15
      item = sym>>4                     /item on l-half
    else
      pend = -1
    finish
    jump out
  $temp r1

  /Read word (4 items)
  routine getword,rb
    word = 1                           /marker
    cycle
      getitem
      word = word<<4
      exit if carry                     /last item ->
      word = word+item
    repeat
    word = word+item
    jump out

  routine putword,rb
    w(loc) = word
    clh loc,lend
    if less
      loc = loc+2
      jump out
    finish
    ptext; b 'Too big', nl, 0
    fail

  /Load object file
```

```
    if less
      loc = loc+2
      jump out
    finish
    ptext; b 'Too big', nl, 0
    fail

  /Load object file
load:
    nsym; format = sym              /format byte (will be ignored)
    if format = X'F2'               /mini-program
      claim #dcb0                   /load to disk buffer
      lstart = pointer(r1)
      lend = lstart+511
    else
      claim bg                      /load to background
      lstart = bgstart
      lend = topstore
      nsize 0                       /assume will be r2 corrupter
    finish
    loc = lstart
    pend = -1
  10:
    getitem                         /loader code
    jump 10(lsw(item1)              /switch on it

15:                                 /set loc counter
    getword
    loc = word+lstart
13:jump 10                          /+ toggle (?)
  110:                              /4 bytes abs
    getword
    putword
  18:                               /+ 2 bytes abs
    getword
181:putword
    jump 10
  111:                              /4 bytes rel
    getword
    putword
  19:                               /+ 2 bytes rel
    getword
    word = word+lstart
    jump 181
115:                                /the 'F' of 'F0' etc
    getitem
    if item = 0
      getword                       /skip seqnum
      getword                       /skip checksum
    finish
    jump 10

  12:16:17:
112:113:114:
 moan:ptext; b 'Program corrupt', nl, 0
    fail

  11:closin
    selin 1
14:if format < x'F1'                /old formats
      pbit(pb) = pbit(pb)+1
      r5 = pb; r2 = loc
      r13 = lstart; r14 = lend
      r13 = r13+4 if w(r13+2) = x'A72E' /BCPL module
      bal r15,w(r13)
      abandon
    finish
    envinfo
    jump w(lstart)
```

```
  lsw:b #10-#10, #11-#10, #12-#10, #13-#10,
    #14-#10, #15-#10, #16-#10, #17-#10,
    #18-#10, #19-#10, #110-#10, #111-#10,
    #112-#10, #113-#10, #114-#10, #115-#10

  $end loader

$begin transfer
  $def chars=r3, lines=r7
  $def terminator=r9,intypes=r10,outtypes=r11

  print:
    s2 = #lp                        /output to printer
    outtypes = x'800'               / which is a file
  trans:
    if terminator = nl
      intypes = intypes&x'8FF'
      outtypes = outtypes&intypes
      if intypes = x'800' and outtypes = intypes
        ac = 'o'+double             /filestore copy
        fcomm
        stop
      finish
      ac = x'FFFF'
    else
      rdec; rsym
    finish
    lines = ac; chars = 0
    refout 1,s2
    cycle                           /for each file
      refin 1,inn1
      cycle
        rsym
        exit if sym < 0
        chars = chars+1
        psym sym
        if sym = nl
          lines = lines-1
          stop if lines = 0
        finish
      repeat
      closin
      inn1 = inn1+20
    repeat if inn1 < outn1
    selout 0
    ac = chars
    pdec; newline
    stop

$end

  $begin ecce (screen editing version)

  /Registers
  $temp r1
$def ac=r0, sym=r1,temp=r1, work=r3,ret=r3
  $def line=r5
  $def pend=r4, rcret=r6            /command input
/     p=r4, min=r6, no=r7           /execution
  $def ci=r8, code=r9, text=r10, num=r11
  $def ti=r12, chain=r13, type=r14, item=r15 /command input
/     lbeg=r12, lend=r13, pp=r14, fp=r15 /execution

  $def c=0, t=2, n=4                /code,text,num fields
$def maxlin=90
  $def sp=\3, not=3

$macro var n,b[2]
  $def n=base
  $redef base=w(#base+b)
```

```
$def c=0, t=2, n=4              /code,text,num fields
$def maxlin=90
$def so=\3, not=3

$macro var n,b[2]
 $def n=base
 $redef base=w(#base+b)
$end
 $def base=save(pb)

var lbeg1                       /*r12*
 var lend1
 var pp1
var fp1                         /*r15*
 var killed                     /*r3*
 var limit
var secfp
 var sin
 var outblocks
var ms
 var ml
 var clim
var mode
 var casing
 var error
var sbeg
 var slim                       /*r15*
 $def in2=b(inout0+4)
$def topcb=out1(pb), botcb=in1(pb), seccb=in2(pb)
 $def fend=slim, cbuff=#base, tlim=nplim(pb)
 $def inf=x'7FFF'


 $/command input routines

 stype:
 /     !"#       $%&'       ()*+       ,-./
   x'0333', x'3233', x'AC03', x'B233'
 /     0123       4567       89:;       <=>?
   x'0000', x'0000', x'0031', x'333D'
 /     @abc       defg       hijk       lmno
   x'9299', x'5749', x'9699', x'9799'
 /     pqrs       tuvw       xyz[       \]^_
   x'9296', x'5559', x'222A', x'DC33'

 routine readsym,ac
   sym = pend
   rsym if sym = 0
   pend = 0
   jump out

 routine number                 /accept number
   jump out if type # 0
   jumps rd1
 routine minus                  /accept minus
   jump out if item # '-'
   code = code-5                /e->@, f->a, m->h
 routine rditem
rd1:type = 1
   cycle
     readsym
     jump out if sym < ' '      /control =>
   repeat if sym = ' '
   sym = sym-32 if sym >= 96    /map lower-case to upper
   item = sym
   sym = sym>>1                 /*sets cc*
   type = stype[sym-16]         /*preserves cc*
   type = type>>4 if \8         /sym was even
   type = type&15
```

```
  jump out if type # 0
  num = item-'0'
  num = 0 if num < 0              /'*'
  cycle
    readsym
    pend = sym
    sym = sym-'9'
    jump out if sym > 0           /not digit  =>
    sym = sym+9
    jump out if sym < 0           /not digit  =>
    num = num<<1; sym = sym+num
    num = num<<2; num = num+sym /10*num+sym-'0'
    pend = 0
  repeat

routine unchain
  cycle
    text = chain
    jump out if text = 0
    chain = t(text)
    t(text) = ci
  repeat if w(text) # 'l'&31
  jump out

routine stack
  ci = ci+6
  jump er6 if ci >= ti
  c(ci-6) = code
  t(ci-6) = text
  n(ci-6) = num
  jump out

edgram:
  b 'F',50,
  '-',4,
  '.',35,
  ' '+128,'.'-1+128,30,
  '.',6,
  '*',13,
  '0'+128,'9'+128,18,
  11,13,
  '\',22,
  11,26,
  '?',26,
  11,26,
  ',',0,
  11,0,
  '.'+1+128,126+128,255,
  11,9,
  '/',119,
  ' '+128,'/'-1+128,44,
  '/',37,
  11,11,
  '/'+1+128,126+128,255,
  11,40,0,
  'D',54,
  11,4,
  'I'+128,'V'+128,59,
  11,4,
  'I',83,
  '.',72,
  '.',67,
  11,11,
  ' '+128,126+128,108,
  11,63,
  '/',255,
  '/',78,
  11,11,
  ' '+128,126+128,108,
  11,74,
```

```
    11,11,
    ' '+128,126+128,108,
    11,63,
    '/',255,
    '/',78,
    11,11,
    ' '+128,126+128,108,
    11,74,
    'S',87,
    11,61,
    'E',93,
    '-',11,
    11,11,
    'M',97,
    11,89,
    'B'+128,'W'+128,102,
    11,11,
    '(',110,
    pcall,0,
    ')',114,
    11,11,
    '[',118,
    11,104,
    ']',118,
    11,11,
    0,
    '0'+128,'9'+128,255,
    '0'+128,'9'+128,4,
    11,122
$def edcomm=*-#edgram
    b '0'+128,'9'+128,edcomm+8,
    '0'+128,'9'+128,edcomm+15,
    11,edcomm+3,
    '%',edcomm+18,
    'A'+128,'Z'+128,255,
    pcall,0,
    nl,255,
    0,
    rt,edcomm+21,
    0,
    '.'+128,'@'+128,edcomm+25,
    0,
    'J',edcomm+28,
    0,
    'w',edcomm+31,
    0,
    'o',edcomm+13,
    0
 $def any=*-#edgram
 b nl+128, 127+128, 255,
    0
 prom:b 3,'>',soh,edcomm
 erprom:b 3,'!',soh,edcomm
anyprom:b 2,soh,any
 routine readco,rcret
 c1:
    ci = clim; pend = 0
    gram(pb) = #edgram
    cycle
      temp = error
      prompt #prom(temp)
      error = 0
      rsym
      jump out if sym = rt and ci # 0 /->repeat last command
    repeat if sym <= ' '
    code = -1; pend = sym
    code = '1'&31 if pend = ';'
    code = 'r'&31 if pend = ':'
    code = 'h'&31 if pend = '.' /m-
    code = 'm'&31 if pend = '/'
```

```
  code = 'e'&31 if pend = 'j'
  code = pend&31 if pend = '@' or pend = 'w' or pend = 'o'
  jump out if code >= 0
  rditem                          /read first item
  if item = '%'
    readsym
    code = sym&95
    pend = -1
    jump out if code = 'C' or code = 'D' or code = 'S'
    code = code-'U'
    code = 32 if code # 0
    casing = code
    jump c1
  finish
  if type = 0 and pend = nl and ci # 0
    n(ci-6) = num
    pend = 0
    jump out
  finish
  ci = chuff; ti = tlim-1
  chain = 0
cget:
  code = item&31; num = 1
  text = tsw[type]
  rditem if type >= 4
  temp = text; text = 0
  jump c00(temp)
c00:c20: c30:
  jump er2
c10:
  abandon if sym < 0
  jump c1
c40: /Find
  minus
  num = 0 if type # 0
c50: /+Delete, Traverse, Uncover, (Verify)
  code = code+(num<<5)
  number
  num = 0                         /as indicator
c60: /+Insert, Substitute
/  jump er4 if type # 3
  text = ti
  cycle
    readsym
    if sym = nl
      pend = sym
      sym = item
    finish
    exit if sym = item
    sym = sym-casing if sym >= 96 and num = 0
    b(ti) = sym; ti = ti-1
    jump er6 if ti = ci
  repeat
/  jump er4 if ti=text and num = 0
  b(ti) = 0; ti = ti-1
  num = 1                         /restore default
  rditem
  jumps c90
c70: /Move, Erase
  minus
c90: /Get, Kill, etc
/  jump er1 if type = 3
  jump c121
c100: /Open bracket
  code = '['&31
  jumps c111
c110: /Comma
  code = ','&31
  num = 0
/  rditem if type = 1             /permit line break
```

```
c110: /Comma
   code = '^'&31
   num = 0
/  rditem if type = 1          /permit line break
c111:
   text = chain
   chain = ci
   jumps cput
c120: /Close bracket
   unchain
   jump er5 if text = 0
   code = '1'&31
   n(text) = num
   text = text+6
c121:number
cput:c130:
   stack
   jump caet if type # 1
   cycle
     unchain
   repeat if text # 0
   code = '1'&31; text = cbuff; num = 1
   stack
   clim = ci
   pend = 0
   jump out

/recognition errors
er1:
/  space
/  psym code+64
er2:
/  code = item-64
/  jumps er5
er4:
/  ptext; b' TEXT FOR',0
/  code = code&31
er5:
/  space
/  psym code+64
/  jumps er9
er6:
/  ptext; b'SIZE',0
er9:
/  psym '?'
/  newline
   clim = 0 if ci # cbuff
/  ignore
   error = #erprom-#prom
   jump c1

tsw:b #c00-#c00, #c10-#c00, #c20-#c00, #c30-#c00,
   #c40-#c00, #c50-#c00, #c60-#c00, #c70-#c00,
   0, #c90-#c00, #c100-#c00, #c110-#c00,
   #c120-#c00, #c130-#c00, #c20-#c00, #c20-#c00

$/* ti,chain,type,insym dropped *
$def p=r4, min=r6, no=w(r7)
$def lbeg=r12, lend=r13, pp=r14, fp=r15

/Set file pointer to start of line
routine lstar,ac
   cycle
     jump out if pp = lbeg
     pp = pp-1; fp = fp-1
     b(fp) = b(pp)
```

```
  repeat

/Set file pointer to end of line
routine rstar,ac
  cycle
    jump out if fp = lend
    b(pp) = b(fp)
    pp = pp+1; fp = fp+1
  repeat

/Write disk block
 /p: cb
 /temp: no of bytes
routine put,ac
  temp = 0 if temp = 512
  temp = temp+s                 /data there
  pointer(p) = pointer(p)+temp
  service p
  jump out if devinf(p) > 0 and pointer(p)&(\bmask) = 0
  abandon

 /Expel one block from start of store
/  shift up rest of top info (ie up to pp)
 /  NB lbeg >= sbeg+512 (except when closing)
nroutine puttop
  temp = lbeg-sbeg              /removeable info
  jump edstop if temp <= 0
  temp = 512 if temp > 512      /restricted to one block
  pp = pp-temp; lbeg = lbeg-temp /adjust
  p = topcb
  put if p >= dcb0
  outblocks = outblocks+1
  p = sbeg                      /Shift up
  while p < pp
    w(p) = w(p+512)
    p = p+2
  repeat
  return

 /Bring in next block of file if any
nroutine slave
  if pp > fp-512
    return if lbeg < sbeg+512 /(overlength lines)
    puttop
  finish
  return if botcb < dcb0 or devinf(botcb) <= 0
                                /fall through
 /Shift up bottom info (from fp)
 /  read one block to end of store
/  NB fp-pp >= 512, lend = slim
nroutine getbot
  push lbeg; lbeg = pp          /Shift up to free lower buffer
  ms = 0 if fp # ms
  rstar
  p = botcb /Read block
  service p
  lend = pointer(p)             /neg if data there
  pointer(p) = lend&bmask       /keep pointer at buffer start
  lend = lend-s
  lend = slim if lend < 0
  fp = lend
  lstar /bring back
  ms = fp if ms # 0
  pop lbeg
  return
```

```
/Shift down top part of info
/  read one block back from output file to start of store
/  NB outblocks > 0, fp-pp >= 512, lbeg = sbeg
nroutine gettop
  push lend; lend = fp          /Shift down
  lstar
  p = topch                     /Read back block
  service p
  pointer(p) = lbeg
  outblocks = outblocks-1
  lbeg = lbeg+512; pp = lbeg
  rstar
  pop lend
  return

/Expel one block from end of store to temp file
/  shift down bottom info
/  NB lend < slim-512 (except sometimes for %S)
nroutine puthot
  p = hotch
  temp = 512; put               /write block
  fp = fp+512; lend = lend+512
  p = slim                      /Shift down
  while p > fp
    p = p-2
    w(p) = w(p-512)
  repeat
  return

/Display (rest of) line
routine displine,ac
  cycle
    p = fp if p = pp
    if p = slim
      push ac
      temp = lend-fp; push temp
      lend = p
      slave
      p = lend; pop temp; lend = fp+temp
      pop ac
      jump out if p = slim
    finish
    sym = b(p); p = p+1
    jump out if sym < ' '
    psym sym
  repeat

$def vline=min                  /NB

routine update,ac
  jump out if min = inf         /no change
  if line >= 0 and line < vlast /on screen
    push ac
    min = pp if min > pp
    ac = line<<8+min-lbeg+x'2000'
    setcursor
    p = min; vline = line
    cycle
      displine
      clearline
      vline = vline+1
      exit if vline = vlast
      exit if killed = 0 and p # pp
      newline
    repeat
    pop ac
```

```
      putbot if fp < pp+512
      gettop
    finish
    exit if lbeg # pp and b(lbeg-1) < ' '
    lbeg = lbeg-1
  repeat
mp1:
  lend = fp-1
  lstar
  line = line-1
  jump mset if line = -2
  jump out

routine insert
  if pp = fp                    /store full
    push sym
    if lbeg-sbeg >= 512
      puttop
      min = min-512             /(ok if min=inf)
    else
      putbot
    finish
    pop sym
  finish
  min = pp if pp < min
  b(pp) = sym
  pp = pp+1
  jump out

routine verify,r0
  p = fp; work = text
  if lend # fend
    cycle
      sym = b(p)
      sym = sym-casing if sym >= 96
      exit if sym # b(work)
      p = p+1; work = work-1
    repeat
  finish
  jump out if b(work) # 0       /fail (cc non-zero*)
  ms = fp
  p = p-fp; ml = p
  temp = 0                      /(cc zero*)
  jump out

$/ entry points
$macro mmove s,d
s1=s; s2=d; smove
$end

ed:
  outn1 = inn1 if b(outn1) = 0 /no output specified
show:
/  mmove cnam,inn1 if b(inn1) = 0
/  mmove outn1,cnam if b(outn1) # 0 and b(outn1+1) # '.'
  refin bg+8+1,inn1
  refin 2,inn1+20
  refout bg+1,outn1
  selin 0; selout 0
  r3 = 0; r4 = 0                /killed, limit
  r6 = 0; r7 = 0                /sin, outblocks
  r8 = 0; r9 = 0                /ms, ml
  r10 = 0 ; r11 = x'2000' /clim, mode
  r12 = 32; r13 = 0             /casing, error
  r14 = pointer(bg)             /sbeg (&pp)
  r15 = pointer(bg+8)+512       /slim (&fp)
/INTERDATA 70/74/7-16 OPERATING SYSTEM
/Department of Computer Science
/Edinburgh University
/ Editing station version from LEGOS:22
```

```
    r10 = 0 ; r11 = x'2000' /clim, mode
    r12 = 32; r13 = 0           /casing, error
    r14 = pointer(bq)           /sheq (&pp)
    r15 = pointer(bq+8)+512     /slim (&fp)
    r5 = r15                    /secfp
    stm r3,killed
    attend
    min = inf
    lend = fp-1
    move
new:
    temp = 0
ernew:
    error = temp
    update
    display
    stm lbeg,lbeg1
    readco
    lm lbeg,lbeg1
    r7 = #lno
    min = inf
    if pend # 0
      if pend < 0
        jump edfin if code = 'C'
/       line = 999 if code = 'D'
        if code = 'S'
          switch
          min = pp; killed = vlines
        finish
        jump new
      finish
      num = 1
      jump rep
    finish
    ci = chuff
nxt:
    jump new if ci = clim       /end of command
    code = c(ci)&31
    text = t(ci)
    num = n(ci)
    ci = ci+6
rep:
    jump x0(xsw[code]<<1)

oka:
    min = pp if pp < min
ok:
    num = num-1
    jump nxt if num = 0
    jump rep

m50:
    if c(ci) = '\'&31 or c(ci) = '?'&31
      ci = ci+6
      jump nxt
    finish
    while c(ci) <= 'I'&31
      ci = t(ci) if c(ci) = '['&31
      ci = ci+6
    repeat
    num = n(ci)
    ci = ci+6
lno:
    jump nxt if num <= 0
    jump m50 if ci # clim
/Execution error
/     selout 0
/     ptext: b'FAILURE: ',0
/     if code <= 'A'&31 or code = 'H'&31 /@,a,h
/       psym code+64+5            /e,f,m
```

```
/     finish
/     psym code+64
/     if text # 0
/       psym au
/       cycle
/         sym = b(text)
/         exit if sym = 0
/         psym sym
/         text = text-1
/       repeat
/       psym au
/     finish
/     newline
/     selin 0
/     ignore
xer:
   temp = #erprom-#prom
   jump ernew


edfin: /Close files
   rsym                              /nl
   if topcb >= dcb0
     switch if sin # 0
     while lend # fend
       move
     repeat
     cycle                           /until => edstop
       puttop
     repeat
   finish
edstop:
   resetcursor; clearline
   stop

x0: /Base label for execution switch

xopen: /Open bracket
   n(text) = num
xquery:
   jump nxt

xclose: /Close bracket
   attend; jump new if 3
   num = num-1
   jump xquery if num = 0
   n(ci-6) = num

xcomma: /+Comma
   ci = text
   jump xquery

xinv: /Invert
   jump no

xc: /Case change with right shift
   jump no if fp = lend
   sym = b(fp)&95
   if 'A' <= sym and sym <= 'Z'
     min = pp if pp < min
     sym = b(fp)\32
     jumps xr1
   finish

xr: /Right shift
```

```
    sym = b(fp)\32
    jumps xr1
  finish

xr: /Right shift
  jump no if fp = lend
  sym = b(fp)
xr1:
  b(pp) = sym
  pp = pp+1; fp = fp+1
  jump ok

xl: /Left shift
  jump no if pp = lbeg
  fp = fp-1; pp = pp-1
  b(fp) = b(pp)
  ms = 0
  jump ok

xe: /Erase
  jump no if fp = lend
  fp = fp+1
  jump oka

xeb: /Erase back
  jump no if pp = lbeg
  pp = pp-1
  jump oka

xs: /Substitute
  jump no if fp # ms
  fp = fp+ml
  min = pp if pp < min

xi: /+Insert
  jump no if pp-lbeg+lend-fp > maxlintmaxlin
  cycle
    sym = b(text)
    exit if sym = 0
    insert
    text = text-1
  repeat
  text = t(ci-6)
  jump ok

xw: xo: /Write, Overwrite
  prompt #anyprom
  cycle
    update
    zdisplay
    rsym
    jump nxt if sym < ' '
    if sym # del
      insert
      fp = fp+1 if code = 'o'&31 and fp # lend
    else if pp # lbeg
      pp = pp-1; min = pp
      if code = 'o'&31
        fp = fp-1; b(fp) = b(pp)
      finish
    finish
  repeat

xg: /Get
  lstar
  sym = nl; insert              /Create blank line
  lend = fp-1
  lstar
  killed = killed-1
  update
  zdisplay
```

```
   prompt 0
   cycle
     rsym
     exit if sym < ' '
     insert
   repeat
   min = pp
   if b(lbeg) = ':'
     pp = lbeg
     kill
     jump no
   finish
   min = inf
   move
   jump ok

xb: /Break
   sym = nl
   if num <= 0
     sym = ff; num = 1
   finish
   insert
   killed = killed-1
   update
   lbeg = pp
   line = line+1
   jump ok

xj: /Join
   rstar
   jump no if pp-lbeg > maxlin
   jumps xk1
 xk: /Kill
   pp = lbeg; fp = lend
xk1:
   kill
   jump ok

 xp: /Print
   update
   ac = num-1; vline = 0
   pdisplay
   prompt #prom
   match nl
   jump nxt if so
   jump new

 xm: /Move
   move
   jump ok

 xmb: /Move back
   jump no if sin # 0
   moveback
   jump ok

 xv:  /Verify
   verify
   jump no if not
   jump ok

xd: xt:                          /Delete, Traverse
   limit = c(ci-6)>>5
   fp1 = fp
   jump xf2
```

```
xu: xt:                          /Delete, Traverse
    limit = c(ci-6)>>5
    fp1 = fp
    jump xf2

 xfb: /Find back
    jump no if sin # 0
 xf: xu:                         /+Find, Uncover
    limit = c(ci-6)>>5
xf1:fp1 = fp
    jumps xf3 if fp = ms
 xf2:fp = fp-1
xf3:ac = b(text)
 xf4:fp = fp+1
    jumps xf5 if fp = lend
    jump xf4 if b(fp)\ac&(\32) # 0
    verify
    jump xf3 if not
    jump xf9
 xf5:fp = fp1
    limit = limit-1
    jump no if limit = 0
    if code = 'A'&31             /Find back
      moveback
    else if code = 'U'&31
      kill
    else
      move
    finish
    jump xf1
xf9:
    jump oka if code > 'T'&31    /Uncover
    fp = fptp if code = 'I'&31
    work = fp1
    while work # fp
      b(pp) = b(work)
      pp = pp+1; work = work+1
    repeat
    jump ok if code # 'D'&31
    fp = fptp
    jump oka

/Execution switch vector
 xsw:b #xeb-#x0>>1, #xfb-#x0>>1, #xb-#x0>>1, #xc-#x0>>1,
    #xd-#x0>>1, #xe-#x0>>1, #xf-#x0>>1, #xg-#x0>>1,
    #xmb-#x0>>1, #xi-#x0>>1, #xj-#x0>>1, #xk-#x0>>1,
    #xl-#x0>>1, #xm-#x0>>1, #xw-#x0>>1, #xo-#x0>>1,
    #xp-#x0>>1, #x0-#x0>>1, #xr-#x0>>1, #xs-#x0>>1,
    #xt-#x0>>1, #xu-#x0>>1, #xv-#x0>>1, #xw-#x0>>1,
    #x0-#x0>>1, #x0-#x0>>1, #x0-#x0>>1, #xopen-#x0>>1,
    #xinv-#x0>>1, #xclose-#x0>>1, #xcomma-#x0>>1, #xquery-#x0>>1

 $end editor

$end resident utilities

 $redef ltemp=*1
$list *1&(\1)
 <= 2C00
 b  reslim-* $ 0

 $list ltemp

$begin system initialisation
 $def ac=r0, temp=r1
 $def work=r4, work2=r5, cb=r6, tcb=r7
$def bit=r8, free=r9, pros=r10, dcbs=r11
 $def lr=r12, lt=r13, blk=r14,top=r14
 $temp r1
$def newpb=r13                   /*nb*
   lm r14,w(x'34')               /preserve 34,36
```

```
$/svcs
 $def call=svc 0          /push return address to nest and branch to <arg>
 $macro nroutine n        /to declare a nest routine
$def n=svc 0,*
 $end
   0                      /illegal inst

 reas1:lkcb0,tcb0         /cb, tcb
   s>>10, reslim+512, 1, 2    /bit, free, pros, dcbs
   0, 0, 0, 0
 reas2:fsrdev, fstdev, #mess, #mess+5 /lr, lt, blk
mess:b 'abort',nl

 $macro put char
  temp = char; tput
 $end
 routine tput,work2
  work = cb+4             /output device
  wd work,temp
routine wait,work2
  ac = 1
  cycle
    ss work,temp
    jump out if \busy
    ac = ac+1
  repeat if ac # 0
  jump out

sinit:stm r14,w(x'34')    /restore
  work = 1                /display
  oc work,enable          /incremental mode
  lm cb,reas1
/Find top of store
  work = 0
  cycle
    work = work-2
    w(work) = -1
  repeat if w(work) = 0
  top = work+1
  topstore = top
/Clear store
  cycle
    w(work) = 0
    work = work-2
  repeat if work # #zbase
  w(0) = 0
/General Reconnaissance Of Peripheral Equipment
  cycle
    oc cb,disable
    if \4                 /device available
      user(cb) = 0
      if cb&15 = 0        /kb: create terminal process
        user(tcb) = 0
        pc(free) = #ttinit
        link(free) = free+ttsize
        exec(free) = free
        r2(free) = free    /pb
        r3(free) = tcb     /cb
        r6(free) = cb      /kcb
        r7(free) = -1      /altcursor
        r13(free) = free+tstore+6 /putr
        r15(free) = free+tstore+6 /buffr
        pointer(tcb) = free+tstore+6
        limit(tcb) = free+tstore+6
        incb(free) = tcb
        outcb(free) = cb+4
        ttout(free) = cb+4
        pbit(free) = bit
        np(free) = free+tstore
/         put esc           /Send esc Z to test if VT52
```

```
        ttout(free) = cb+4
        pbit(free) = bit
        np(free) = free+tstore
/        put esc                 /Send esc Z to test if VT52
/        put 'Z'
/        put rt
/        work = cb; wait         /any response?
/        if ac # 0               /yes
/          rd work,temp          /discard esc
/          wait
/          rd work,temp          /discard 'K'
          ac = x'C000'
/        finish
        ac = ac+x'8000'
        vtype(free) = ac
/        put rt
/        put lf
        proset = proset+bit
        tstore(free) = x'8920' /tab
        tstore(free+2) = x'2020'
        tstore(free+4) = x'8000'
        free = free+tsize
        bit = bit>>1
        if top&x'C000' # 0
          dcbs = 4 if dcbs = 2 and top < 0
          pros = pros+1; dcbs = dcbs+3
        finish
      finish
    finish
    tcb = tcb+8 if cb&15 = 0
    cb = cb+4
  repeat if cb < tcb0
  jump w(*) if free = reslim+512
  link(free-tsize) = #ibase
  ubase0 = free
  bit = s
  cycle
    ps(free) = usr
    pc(free) = #command
    pbit(free) = bit
    np(free) = free+usize
    nplim(free) = free+usize-20
    free = free+usize
    bit = bit>>1
    pros = pros-1
  repeat if pros > 0
  pblim = free
  free = (free+511)>>9<<9
  cb = dcb0+8
  cycle
    pointer(cb) = free
    dcbs = dcbs-1
    if dcbs >= 0
      user(cb) = 0
      free = free+512
    finish
    cb = cb+8
  repeat if cb <= cblim
  pointer(bg+8) = free+x'A00'
  bgstart = free

  lm lr,regs2
  wb lt,blk
  cycle
    cycle
      ss lr,r1
    repeat if \clear
    rd lr,r1
  repeat if r1 # n1
```

```
/     fclaim
/     work = 40
/     cycle
/       ptext; b 'H,', 0
/       r0 = work; phex
/       response
/       work = work-1
/     repeat if work > 0
/     work = 40
/     cycle
/       ptext; b 'M,', 0
/       r0 = work; phex
/       response
/       work = work-1
/     repeat if work > 0
/     frelease

    newpb = reslim+512
    jump svgo

 ttinit:
    newline
    r14 = #sysident
    pstring; newline
    jump despatch
zbase:

 $end init
$end of program
```