```
! New VAX program for the Fred Machine.        GB Dec 1984.

! Uses NXDRIVER at the Vax end.

%include "inc:util.imp"
%include "inc:fs.imp"

%begin

%constinteger vax = 16_72ØØ
%constinteger vax timeout = 7 { secs, before timing out Vax at start-up }
%constinteger delay = 5Ø { ms, between messages to Vax }
%constinteger ta limit = 1<<7-1 { size-1 of type-ahead buffer }
%constinteger in limit = 1<<1Ø-1 { size-1 of input buffer }

@16_3724 %integer control mask
@16_3FA2 %routine print symbol(%byte ch)

%recordformat nx messf  %c
 (%byte term, type,
  ((%integer unit no) %orc
   (%half read func, %byte read flags, read seed, %half max len,
    %half read timeout, %string(127) terminator mask) %orc
   (%byte read done flags, read done seed, %half read status,
    %half read len, read term len, read buff len,
    %bytearray read data(Ø:243)) %orc
   (%half write func, %byte write flags, write seed,
    %half write len, write buff len,
    %bytearray write data(Ø:245)) %orc
   (%byte write done flags, write done seed, %half write status,
    %half lines out, %byte new col, new row) %orc
   (%byte dev class, dev type, %half page width,
    %integer dev depend 1, dev depend 2,
    %byte tx speed, rx speed, cr fill, lf fill, parity) %orc
   (%integer input mask, noninput mask) %orc
   (%byte trigger ch) %orc
   (%half ahead len, %byte ahead ch)))

%constbyte nx start = Ø, nx stop = 1, nx read = 2, nx write = 3,
           nx mode = 4, nx trigger = 5, nx ahead = 6

%constbyte ctrlc = 3, bel = 7, bs = 8, tab = 9,
           lf = 1Ø, vt = 11, ff = 12, cr = 13,
           ctrlo = 15, ctrlp = 16, ctrlr = 18,
           ctrlu= 21, ctrlx = 24, ctrly = 25,
           ctrlz = 26, esc = 27, del = 127

%integer chan
%record(nx messf) mess
%byte read pending = Ø, read seed
%byte skip lf = Ø, blank line = Ø, last char = Ø
%integer lf count = Ø
%byte stop flag = Ø
%byte del flag = Ø
%byte ctrlo flag = Ø
%half x = Ø, width = 8Ø
%byte y = Ø, page = 24
%byte hard flag = Ø, form flag = 1, tab flag = Ø, wrap flag = 1
%byte ta flag = 1, echo flag = 1, trm echo flag = 1, lower flag = 1
%byte eightbit flag = Ø, passall flag = Ø, filter flag = 1, timed flag = Ø
%integer read expire
%integer mode i = 16_18ØØ13AØ, mode 2 = Ø
%half max read
%integer i, j, lport, rport, terminal ch
%string(255) s
```

```
%integer last cpu = 0
%integer input mask = 0, noninput mask = 16_02000008
%record(nx messf) comm
%string(32) terminator mask = ""
%string(255) next prompt = "", terminal prompt = ""

%bytearray ta buffer(0:ta limit)
%integer ta put = 0, ta get = 0

%bytearray input buffer(0:in limit)
%integer input put = 0, input get = 0
%shortarray input x(0:in limit)

%string(2) crlf = tostring(cr).tostring(lf)
%string(3) bsspbs = tostring(bs).tostring(' ').tostring(bs)

%halffn resex half {Bloody IBM} (%half i)
%result = (i&255)((8 ! (i))8)
%end

%integerfn resex long {Sexual perversion} (%integer i)
%result = (i&255)((24 ! (i&255((8)((8 ! (i&255((16))8 ! (i))24)
%end


%routine output char(%byte ch)
x = 0 %if ch = esc  { Assume esc always dangerous, a la VMS }
%if passall flag # 0 %then print symbol(ch) %else %start
   ch = ch&127 %if eightbit flag = 0
   %if ch = lf %start
      blank line = 1 %if last char = cr
      %if skip lf # 0 %then skip lf = 0 %else %start
         print symbol(lf)
         lf count = lf count+1
         y = y+1
         y = 0 %if y >= page
      %finish
   %else %if ch = cr
      print symbol(cr) %unless last char = cr
      x = 0
   %else %if ch = tab
      %if tab flag = 0 %start
         output char(cr) %and output char(lf) %if wrap flag # 0 %and x >= width
         print symbol(tab)
         x = (x+8)&(\7)
         x = width %if x > width
      %else
         output char(' ') %until x&7 = 0
      %finish
   %else %if ch = ff
      %if form flag = 0 %start
         print symbol(ff)
         y = 0
      %else
         output char(lf) %until y = 0
      %finish
   %else %if ch = vt
      output char(lf) %for i = 1, 1, 4
   %else
      skip lf = 0 %unless lf < ch < cr
      %if ' ' <= ch < del %start
         blank line = 0
         output char(cr) %and output char(lf) %if wrap flag # 0 %and x >= width
         x = x+1
      %else %if ch = bs
         x = x-1 %unless x = 0
```

```
      %finish
      print symbol(ch)
   %finish
%finish
last char = ch
%end


%routine mod output char(%byte ch, cr suppress)
%if passall flag # Ø %then print symbol(ch) %else %start
   ch = ch&127 %if eightbit flag = Ø
   %if ch # cr %start
      output char(ch)
   %else %if blank line = Ø %or cr suppress = Ø
      skip lf = Ø
      output char(cr);   output char(lf)
      skip lf = 1
   %finish
%finish
%end


%routine output string(%string(255) s)
%integer i
output char(charno(s, i)) %for i = 1, 1, length(s)
%end


%routine repaint input
%integer i
%if terminal prompt # "" %start
   mod output char(charno(terminal prompt, 1), 1)
   %for i = 2, 1, length(terminal prompt) %cycle
      output char(charno(terminal prompt, i))
   %repeat
%finish
%if echo flag # Ø %start
   i = input get
   %while i # input put %cycle
      input x(i) = x
      mod output char(input buffer(i), Ø)
      i = (i+1)&in limit
   %repeat
%finish
%end


%routine log out(%record(nx messf)%name comm, %integer len)
%integer i
%while cpu time-delay < last cpu %cycle; %repeat
last cpu = cpu time
ether write(lport, comm_term, len)
%end


%routine set read modes
timed flag = Ø
echo flag = 1-mode1 >> 1 {TT$V_NOECHO} &1
lower flag = mode1 >> 7 {TT$V_LOWER} &1
passall flag = mode1 {TT$V_PASSALL} &1
terminal prompt = ""
input get = input put
%end


%routine attempt input
```

```
%byte cn, ocn, store, term
%integer i, j, k, ptr
term = Ø
%while ta get # ta put %cycle
   ch = ta buffer(ta get)
   ta get = (ta get+1)&ta limit
   store = 1
   %if passall flag = Ø %start
      ch = ch&127 %if eightbit flag = Ø
      %if filter flag # Ø %start
         %if ch = del %start
            %if input put # input get %start
               input put = (input put-1)&in limit
               %if echo flag # Ø %start
                  och = input buffer(input put)
                  %if hard flag = Ø %start
                     %if ' ' <= och <= del %start
                        output string(bsspbs)
                     %else %if och = tab
                        output string(bsspbs) %while x ) input x(input put)
                     %finish
                  %else
                     output char('\') %and del flag = 1 %if del flag = Ø
                     output char(och) %if och = tab %or ' ' <= och < del
                     output char('\') %and del flag = Ø %if input put = input get
                  %finish
               %finish
            %finish
            store = Ø
         %else %if ch = ctrlu %or ch = ctrlx
            %if echo flag # Ø %and input put # input get %start
               output string("^U".tostring(cr))
               input put = input get
               repaint input
            %else
               input put = input get
            %finish
            store = Ø
         %else %if ch = ctrlr
            output string("^R".tostring(cr))
            repaint input
            store = Ø
         %finish
      %finish
   %finish
   %if store # Ø %start
      ch = ch-'a'+'A' %if lower flag = Ø %and 'a' <= ch <= 'z'
      input buffer(input put) = ch
      input x(input put) = x
      input put = (input put+1)&in limit
      term = (charno(terminator mask, ch))3+1)))(ch&7))&1
      j = (input put-input get)&in limit
      %if term = Ø %or j = max read %start
         %if echo flag # Ø %start
            output char('\') %and del flag = Ø %if del flag # Ø
            mod output char(ch, Ø)
         %finish
      %else %if trm echo flag # Ø
         output char('\') %and del flag = Ø %if del flag # Ø
         %if ch = ctrlz %start
            output string("^Z".tostring(cr))
         %else
            mod output char(ch, Ø)
         %finish
      %finish
      %exit %if i = max read %or term # Ø
```

```
   %finish
%repeat
j = (input put-input get)&in limit
%if j = max read %or term # Ø %or (timed flag # Ø %and cpu time )= read expire) %
   output char('\') %and del flag = Ø %if echo flag # Ø %and del flag # Ø
   mess_type = nx read
   mess_read done seed = read seed
   %if j = max read %or term # Ø %start
      mess_read status = 1 {SS$_NORMAL}
   %else
      mess_read status = 16_22C {SS$_TIMEOUT}
   %finish
   mess_read status = resex half(mess_read status)
   mess_read len = resex half(j-term)
   mess_read term len = resex half(term)
   %cycle
      %if j <= 244 %start
         i = j
         mess_read done flags = Ø
      %else
         i = 244
         mess_read done flags = 128
      %finish
      mess_read buff len = resex half(i)
      %for k = Ø, 1, i-1 %cycle
         mess_read data(k) = input buffer(input get)
         input get = (input get+1)&in limit
      %repeat
      log out(mess, 12+i)
      j = j-i
   %repeat %until j = Ø
   read pending = Ø
   set read modes
   %if ta put # ta get %start
      mess_type = nx ahead
      mess_ahead len = (ta put-ta get)&ta limit
      mess_ahead ch = ta buffer((ta put-1)&ta limit)
      log out(mess, 5)
   %finish
%finish
%end


%routine handle keyboard
%byte store
%integer i
store = 1
%if read pending = Ø %or passall flag = Ø %start
   %if terminal ch < ' ' %start
      %if (input mask!noninput mask)&(1<<terminal ch) # Ø %start
         store = Ø %if noninput mask&(1<<terminal ch) # Ø
         mess_type = nx trigger
         mess_trigger ch = terminal ch
         log out(mess, 3)
      %finish
      %if terminal ch = ctrlc %or terminal ch = ctrly %or terminal ch = ctrlx %st
         store = Ø %unless input mask&(1<<terminal ch) # Ø
         %if terminal ch = ctrlx %start
            %if echo flag # Ø %and input put # input get %start
               output string("^X".tostring(cr))
               input put = input get
               repaint input
            %finish
         %else
            output string(crlf."^".tostring(terminal ch+'A'-1).tostring(cr))
         %finish
```

```
         ta put = ta get;   input put = input get
      %else %if terminal ch = ctrlo
         store = Ø %unless input mask&(1((ctrlo) # Ø
         %if read pending = Ø %start
            ctrlo flag = 1-ctrlo  flag
            output string("^O".tostring(cr)) %if ctrlo flag # Ø
         %finish
      %finish
   %finish
%finish
%if store # Ø %start
   %if ta flag = Ø %or (ta put-ta get)&ta limit = ta limit %start
      output char(bel)
   %else
      ta buffer(ta put) = terminal ch
      ta put = (ta put+1)&ta limit
      %if read pending ( 2 %start
         %if (ta put-ta get)&ta limit = 1 %start
            mess_type = nx ahead
            mess_ahead len = resex half(1)
            mess_ahead ch = terminal ch
            log out(mess, 5)
         %finish
      %else
         attempt input
      %finish
   %finish
%finish
%end


%routine handle nx(%record(nx messf)%name comm)
%byte refresh flag, temp flag, old blank
%half read fun, write fun
%integer i, j, read timeout, count
%string(*)%name s
%switch sw(nx start:nx ahead)
%if comm_term = rport %and nx start (= comm_type (= nx ahead %start
   -) sw(comm_type)
   sw(nx stop):
      stop flag = 1
      %stop

   sw(nx read):
      set read modes %if read pending = 2
      read pending = 2
      read seed = comm_read seed
      max read = resex half(comm_max len)
      max read = 1Ø24 %if max read ) 1Ø24
      terminator mask = comm_terminator mask
      %while length(terminator mask) ( 32 %cycle
         terminator mask = terminator mask.tostring(Ø)
      %repeat
      read fun = resex half(comm_read func)
      i = read fun&16_3F
      passall flag = 1 %if i = 16_C {IO$_READPBLK} %or  %c
                           i = 16_3A {IO$_TTYREADALL} %or  %c
                           i = 16_3B {IO$_TTYREADPALL}
      echo flag = Ø %if read fun )) 6 {IO$V_NOECHO} &1 # Ø
      lower flag = Ø %if read fun )) 8 {IO$V_CVTLOW} &1 # Ø
      filter flag = 1-read fun )) 9 {IO$V_NOFILTR} &1
      ta put = ta get %if read fun )) 11 {IO$V_PURGE} &1 # Ø
      trm echo flag = 1-read fun )) 12 {IO$V_TRMNOECHO} &1
      timed flag = read fun )) 7 {IO$V_TIMED} &1
      %if timed flag # Ø %start
         read timeout = resex half(comm read timeout)
```

```
      read expire = cpu time + 10000*read timeout
   %finish
   ctrlo flag = 0
   terminal prompt = next prompt
   next prompt = ""
   %if terminal prompt = "" %or cr # charno(terminal prompt, 1) # lf %start
      %if passall flag = 0 %and echo flag # 0 %and last char = cr %start
         terminal prompt = tostring(lf).terminal prompt
      %finish
   %finish
   repaint input
   attempt input
   -> end

sw(nx write):
   comm_write buff len = resex half(comm_write buff len)
   s == string(addr(comm_write buff len)+1)
   mess_lines out = 0
   mess_write status = resex half(1) {SS$_NORMAL}
   %if comm_write flags&16_40 # 0 %start
      ctrlo flag = 0
      %if read pending # 0 %and blank line = 0 %start
         output char(cr);   output char(lf)
      %finish
      set read modes %if read pending = 2
      read pending = 1
      next prompt = s
   %else
      write fun = resex half(comm_write func)
      ctrlo flag = 0 %if write fun >> 6 {IO$V_CANCTRLO} &1 # 0
      %if ctrlo flag # 0 %start
         mess_write status = resex half(16_609) {SS$_CONTROLO}
      %else
         temp flag = passall flag
         passall flag = passall flag!write fun >> 8 {IO$V_NOFORMAT} &1
         refresh flag = write fun >> 13 {IO$V_REFRESH} &1
         count = lf count
         %if passall flag # 0 %or %c
         read pending = 0 %or refresh flag = 0 %or  %c
         (terminal prompt = "" %and  %c
         (echo flag = 0 %or input put = input get)) %start
            old blank = blank line
            output string(s)
            %if refresh flag # 0 %start
               mod output char(cr, 1) %if passall flag = 0 %and old blank # 0
            %finish
         %else
            %if blank line = 0 %and (length(s) < 2 %or substring(s, 1, 2) # cr]
               output char(cr);   output char(lf)
            %finish
            output string(s)
            %if blank line = 0 %and  %c
            (length(terminal prompt) = 0 %or charno(terminal prompt, 1) # cr) %
               output char(cr);   output char(lf)
            %finish
            repaint input
         %finish
         mess_lines out = lf count-count %if passall flag = 0
         passall flag = temp flag
      %finish
   %finish
   mess_type = nx write
   mess_write done flags = comm_write flags&16_80
   mess_write done seed = comm_write seed
   mess_lines out = resex half(mess_lines out)
   mess new col = x;   mess new row = v
```

```
      log out(mess, 10)
      -> end

   sw(nx mode):
      width = resex half(comm_page width)
      mode 1 = resex long(comm_dev depend 1)
      page = mode 1>>24
      ta flag = 1-mode1 >> 2 {TT$V_NOTYPEAHD} &1
      tab flag = 1-mode1 >> 8 {TT$V_MECHTAB} &1
      wrap flag = mode1 >> 9 {TT$V_WRAP} &1
      hard flag = 1-mode1 >> 12 {TT$V_SCOPE} &1
      eightbit flag = mode1 >> 15 {TT$V_EIGHTBIT} &1
      form flag = 1-mode1 >> 19 {TT$V_MECHFORM} &1
      mode 2 = resex long(comm_dev depend 2)
      set read modes %if read pending = 0
      -> end

   sw(nx trigger):
      input mask = resex long(comm_input mask)
      noninput mask = resex long(comm_noninput mask)
      -> end

   sw(*):
   end:
%finish
%end


%on 0 %start
   mod output char(cr, 1)
   %if stop flag = 0 %start
      mess_type = nx stop
      log out(mess, 2)
   %finish
   control mask = 0
   %stop
%finish


s = cli param

lport = 1
lport = 2 %if lport = fs port
ether open(lport, vax+0)
mess_term = 9
ether write(lport, mess_term, 1)
i = cpu time
%cycle
   -> ok %if dtx&(1<<(lport) # 0
%repeat %until cpu time-i >= vax timeout*1000
newline;   print string("No response from Vax");   newline
%stop

ok:
i = ether read(lport, charno(s, 1), 255)
%unless i = 2 %and 1 <= charno(s, 1)-'0' <= 31 %and charno(s, 2) = nl %start
   length(s) = i&255
   print string(s)
   %stop
%finish

rport = charno(s, 1)-'0'
ether open(lport, vax+rport)
prompt("")
set terminal mode(nopage)
control mask = 16 FFF5FFFF
```

```
mess_term = rport
mess_type = nx start
mess_unit no = resex long(lport((8 ! ether station&16_FF)
log out(mess, 6)

mess_type = nx mode
mess_dev type = 16_40  { VT52 }
mess_page width = resex half(80)
mess_dev depend1 = resex long(16_1800013A0)
                   {24 line page, TTSYNC, LOWER, MECHTAB, WRAP, SCOPE}
mess_dev depend2 = 0
mess_tx speed = 16_10;   mess_rx speed = 16_10  { 19.2 }
mess_cr fill = 0;   mess_lf fill = 0;   mess_parity = 0
log out(mess, 19)

%cycle
   terminal ch = test symbol

   %if terminal ch )= 0 %start
      handle keyboard
   %finish

   %if dtx&(1((lport) # 0 %start
      i = ether read(lport, comm_term, 256)
      handle nx(comm)
   %finish

   %if timed flag # 0 %start
      attempt input %if cpu time )= read expire
   %finish
%repeat

%endofprogram
```