VTEmulator#

General Information

Author: Charles Green
Language: ACTION!
Compiler/Interpreter: ACTION!
Published: usenet

Well, I've gotten enough responses that I'd annoy my UUCP neighbor by mailing this file out that many times, so I guess it'll be better to post it.

A couple of things I never got around to doing:

- Adding a character count to the data in each line, maybe in the byte just before the 320-byte area where the characters for each line are "stenciled" in for display. This count could be used to speed up the insert-character and clear-to-end-of-line operations; they'd no longer have to go all the way out to column 80 if there was no data displayed out that far.

- Adding a single graphics-0 display line, either at the top or bottom of the screen, which could hold the current baud rate, parity, etc. (assuming that someone adds the code to change them, in addition to maybe brightness). (P.S. - I've since discovered that luminance values of 0 for brightness, 2 or 4 for cursor, and 6 for the characters works better than the values in this program. You also probably want to put the contrast at a minimum.)

- Smooth scrolling.

Enjoy, Charles Green char...@c3.COM

```
;''''''''''''''''''''''''''''''''''''
; "TVI925.ACT" - A display-list based
; terminal emulator by Charles Green.
;
; Derived from the public domain VT52+
; emulator written by Michael Jenkin.
;,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
MODULE
;A: handler, originally by Jenkin.

DEFINE LDY = "$A0",
 RTS = "$60", JMP = "$4C", PLA="$68",
 GR8 = "$0F", SKIP8 = "$70",
 JVP = "$41", LMS = "$40", RTI="$40",
 ESC = "1", CUP = "2", SGR = "3",
 PHA= "$48", TAX= "$AA", TXA= "$8A",
 TAY="$A8",TYA="$98",
 CMO = "4",
 XITVBV = "$E462"

TYPE line = [BYTE mode
 CARD adrs]
BYTE ARRAY display(8831) ;data+dlist
BYTE ARRAY enhance(8), cursor(4)
line POINTER dlist
BYTE POINTER curln ;current line data
BYTE pmbase = $D407, gractl=$D01D,
 lmargin = $52, rmargin = $53,
 rowcrs = $54, colcrs = $55, brkcnt,
```

```
 gprior = $26F, pcolr0 = $2C0,
 audc1 =$D201, hposm0 = $D004,
 sdmctl = $22F, soundr = $41, saudc1,
; audf1 = $D200,
; sskctl = $232,
 skctl = $D20F,
 state, need, needx, ins_mode, curcnt
CARD
; savmsc = $58,
 sdlist = $230,cursad,keydef = $79,
 vvblkd = $224, brkky = $236
 BYTE ARRAY xlate = [ ; New kbd map
$6C $6A $3B $80 $80 $6B $2B $2A
$6F $80 $70 $75 $0D $69 $2D $3D
$76 $FF $63 $80 $80 $62 $78 $7A
$34 $80 $33 $36 $1B $35 $32 $31
$2C $20 $2E $6E $80 $6D $2F $7E
$72 $80 $65 $79 $09 $74 $77 $71
$39 $80 $30 $37 $08 $38 $3C $3E
$66 $68 $64 $80 $82 $67 $73 $61 ;lowr
$4C $4A $3A $80 $80 $4B $5C $5E
$4F $80 $50 $55 $0D $49 $5F $7C
$56 $FF $43 $80 $80 $42 $58 $5A
$24 $80 $23 $26 $1B $25 $22 $21
$5B $20 $5D $4E $80 $4D $3F $60
$52 $80 $45 $59 $FE $54 $57 $51
$28 $80 $29 $27 $7F $40 $7B $7D
$46 $48 $44 $80 $82 $47 $53 $41 ;uper
$0C $0A $F0 $80 $80 $0B $1C $1E
$F0 $80 $10 $15 $0A $09 $1F $F0
$16 $FF $03 $80 $80 $02 $18 $1A
$F0 $80 $F0 $F0 $1B $F0 $F0 $F0
$1B $00 $1D $0E $80 $0D $7F $F0
$12 $80 $05 $19 $FD $14 $17 $11
$F0 $80 $F0 $F0 $7F $00 $F0 $F0
$06 $08 $04 $80 $82 $07 $13 $01 ;ctrl
 ]

PROC vbi()
 IF curcnt = 0 THEN
  pcolr0 ==! 5 ;invert cursor
  curcnt = 15
 ELSEIF curcnt < 255 THEN
  curcnt ==- 1
 FI
 IF saudc1 > $E0 THEN
  saudc1 ==- 1
  audc1 = saudc1
; SOUND(2,100,14,bel)
 FI
 IF brkcnt THEN
  brkcnt ==- 1
  IF brkcnt = 0 THEN
;  sskctl ==& $7F ;remove force space
;  skctl = sskctl
   skctl = $73
  FI
 FI
 [JMP XITVBV]
```

```
PROC brk() ;break interrupt handler
; [TYA PHA TXA PHA]
 brkcnt = 15;
; sskctl ==% $80 ;force serial space
; skctl = sskctl
 skctl = $F3
; [PLA TAX PLA TAY]
 [PLA RTI]

PROC del_ch(BYTE POINTER adr, BYTE col)
 BYTE POINTER badr,wadr
 BYTE x, y, bcount
 bcount = (80-col) RSH 1
 FOR y = 1 TO 7
 DO
  adr ==+ 40
  wadr = adr + (col RSH 1) - 1
  badr = wadr + 1
  IF col & 1 THEN
   badr ==+ 1
   wadr ==+ 1
   wadr^ =(wadr^ & $F0)%(badr^ RSH 4)
  FI
  x = bcount
  WHILE x > 0
  DO
   badr ==+ 1
   wadr ==+ 1
   wadr^ =(wadr^ LSH 4)%(badr^ RSH 4)
   x ==- 1
  OD
  wadr^ ==& $F0
 OD
 RETURN ; null proc for now

PROC ins_ch(BYTE POINTER adr, BYTE col)
 BYTE POINTER badr,wadr
 BYTE x, y, bcount
 bcount = (79-col) RSH 1
 FOR y = 1 TO 7
 DO
  adr ==+ 40
  badr = adr + 39
  wadr = badr - 1
  x = bcount
  WHILE x > 0
  DO
   badr^ =(wadr^ LSH 4)%(badr^ RSH 4)
   x ==- 1
   badr ==- 1
   wadr ==- 1
  OD
  IF col & 1 THEN
   badr^ ==& $F0
  ELSE
   badr^ ==RSH 4
  FI
 OD
```

```
 RETURN ; null proc for now

PROC Achr(BYTE cx, cy, cc)
 BYTE POINTER base, offset
 BYTE i, chr, c
 BYTE ARRAY chset = [
  $00 $00 $00 $00 $00 $00 $00 ; SP
  $44 $44 $44 $00 $44 $00 $00
  $AA $AA $00 $00 $00 $00 $00
  $AA $EE $AA $EE $AA $00 $00
  $EE $CC $EE $66 $EE $00 $00
  $AA $22 $44 $88 $AA $00 $00
  $CC $44 $CC $EE $EE $00 $00
  $66 $22 $44 $00 $00 $00 $00
  $22 $44 $44 $44 $22 $00 $00
  $88 $44 $44 $44 $88 $00 $00
  $00 $AA $EE $AA $00 $00 $00
  $00 $44 $EE $44 $00 $00 $00
  $00 $00 $00 $66 $22 $44 $00
  $00 $00 $EE $00 $00 $00 $00
  $00 $00 $00 $66 $66 $00 $00
  $22 $22 $44 $44 $88 $88 $00
  $44 $AA $AA $AA $44 $00 $00 ;'0'
  $44 $CC $44 $44 $EE $00 $00
  $CC $22 $44 $88 $EE $00 $00
  $CC $22 $44 $22 $CC $00 $00
  $AA $AA $EE $22 $22 $00 $00
  $EE $88 $CC $22 $CC $00 $00
  $44 $88 $EE $AA $44 $00 $00
  $EE $22 $44 $88 $88 $00 $00
  $44 $AA $44 $AA $44 $00 $00
  $44 $AA $66 $22 $44 $00 $00
  $66 $66 $00 $66 $66 $00 $00
  $66 $66 $00 $66 $22 $44 $00
  $22 $44 $88 $44 $22 $00 $00
  $00 $EE $00 $EE $00 $00 $00
  $88 $44 $22 $44 $88 $00 $00
  $CC $22 $44 $00 $44 $00 $00
  $44 $AA $AA $88 $66 $00 $00 ;'@'
  $44 $AA $EE $AA $AA $00 $00
  $CC $AA $CC $AA $CC $00 $00
  $66 $88 $88 $88 $66 $00 $00
  $CC $AA $AA $AA $CC $00 $00
  $EE $88 $CC $88 $EE $00 $00
  $EE $88 $CC $88 $88 $00 $00
  $66 $88 $AA $AA $66 $00 $00
  $AA $AA $EE $AA $AA $00 $00
  $EE $44 $44 $44 $EE $00 $00
  $66 $22 $22 $AA $EE $00 $00
  $AA $CC $88 $CC $AA $00 $00
  $88 $88 $88 $88 $EE $00 $00
  $AA $EE $EE $AA $AA $00 $00
  $CC $AA $AA $AA $AA $00 $00
  $66 $AA $AA $AA $CC $00 $00
  $CC $AA $CC $88 $88 $00 $00 ;'P'
  $66 $AA $AA $AA $EE $22 $00
  $CC $AA $CC $CC $AA $00 $00
  $66 $88 $CC $22 $EE $00 $00
  $EE $44 $44 $44 $44 $00 $00
```

```
  $AA $AA $AA $AA $66 $00 $00
  $AA $AA $AA $44 $44 $00 $00
  $AA $AA $EE $EE $AA $00 $00
  $AA $AA $44 $AA $AA $00 $00
  $AA $AA $44 $44 $44 $00 $00
  $EE $22 $44 $88 $EE $00 $00
  $66 $44 $44 $44 $66 $00 $00
  $88 $88 $44 $44 $22 $22 $00
  $CC $44 $44 $44 $CC $00 $00
  $44 $AA $00 $00 $00 $00 $00
  $00 $00 $00 $00 $EE $00 $00
  $66 $44 $22 $00 $00 $00 $00
  $00 $CC $66 $AA $66 $00 $00 ;'a'
  $88 $CC $AA $AA $CC $00 $00
  $00 $66 $88 $88 $66 $00 $00
  $22 $66 $AA $AA $66 $00 $00
  $00 $EE $EE $88 $66 $00 $00
  $22 $44 $EE $44 $44 $00 $00
  $00 $66 $AA $AA $66 $22 $CC
  $88 $EE $AA $AA $AA $00 $00
  $44 $00 $CC $44 $EE $00 $00
  $44 $00 $CC $44 $44 $88 $00
  $88 $88 $AA $CC $AA $00 $00
  $CC $44 $44 $44 $EE $00 $00
  $00 $CC $EE $EE $AA $00 $00
  $00 $CC $AA $AA $AA $00 $00
  $00 $66 $AA $AA $CC $00 $00
  $00 $CC $AA $AA $CC $88 $88 ;'p'
  $00 $66 $AA $AA $66 $22 $22
  $00 $EE $88 $88 $88 $00 $00
  $00 $CC $EE $22 $CC $00 $00
  $44 $EE $44 $44 $44 $00 $00
  $00 $AA $AA $AA $66 $00 $00
  $00 $AA $AA $44 $44 $00 $00
  $00 $AA $EE $EE $66 $00 $00
  $00 $AA $44 $AA $AA $00 $00
  $00 $AA $AA $AA $66 $22 $CC
  $00 $EE $44 $88 $EE $00 $00
  $66 $44 $88 $44 $66 $00 $00
  $44 $44 $44 $44 $44 $00 $00
  $CC $44 $22 $44 $CC $00 $00
  $CC $66 $00 $00 $00 $00 $00
  $AA $44 $AA $44 $AA $44 $AA
 ]
;strip high bit (inverse video)
 offset = cc - $20
;display character
 dlist=sdlist + cy * 10 + 3
 base = dlist.adrs + (cx RSH 1) + 40
 IF ins_mode THEN
  ins_ch(dlist.adrs, cx)
 FI
 offset = chset+(offset LSH 3)-offset
 FOR i = 1 TO 7
 DO
  c = offset^
   c = c XOR enhance(i)
  chr = base^
  IF (cx & 1) THEN
```

```
   c ==& $0F
   chr ==& $F0
  ELSE
   c ==& $F0
   chr ==& $0F
  FI
   base^ = chr % c
   base ==+ 40
   offset ==+1
 OD
 RETURN

PROC Acurse(); move cursor
 BYTE old_y ;Where we left cursor
 Zero(cursad+(old_y LSH 2), 4)
 hposm0 = (colcrs LSH 1) + 48
 MoveBlock(cursad+(rowcrs LSH 2),cursor,4)
 old_y = rowcrs ;memory for later clr
 pcolr0=5
 IF curcnt < 255 THEN
  curcnt=254
 FI
 RETURN

PROC scroll(BYTE first, last)
 line POINTER tdlist
 CARD tempadrs
 INT delta
 BYTE i
 dlist=sdlist+3 + first*10
 tempadrs=dlist.adrs
 Zero(tempadrs,320)
 IF first>last THEN
  i=first
  first=last
  last=i
  delta=-1
 ELSE
  delta=1
 FI
 FOR i=first+1 TO last
 DO
  tdlist = dlist+10            *delta
  dlist.adrs = tdlist.adrs
  dlist = tdlist
 OD
 dlist.adrs = tempadrs
 RETURN

PROC Amargin() ; keep between margins
 IF colcrs > rmargin THEN
  colcrs = lmargin
  rowcrs ==+ 1
 FI ;fallthrough to Ascroll()
PROC Ascroll() ; roll screen if req'd
 IF rowcrs > 23 THEN
  scroll(0,23)
  rowcrs = 23
 FI
```

```
 RETURN

PROC curpos(BYTE chr)
 IF need = 2 THEN ; 1st ESC Y
  needx = chr - $20
  need ==- 1
 ELSEIF need = 1 THEN ; 2nd ESC Y
  chr ==- $20
  IF (needx <= 23) AND (chr <= rmargin) THEN
   colcrs = chr
   rowcrs = needx
;   Acurse()
  FI
  state = 0
 FI
 RETURN

PROC cursat(BYTE chr) ;cursor type
 IF chr = '2 THEN ;blinking block?
  curcnt = 254
  SetBlock(cursor,4,3)
 ELSEIF chr = '4 THEN ;steady line?
  curcnt = 255
  SetBlock(cursor,3,0)
  cursor(3) = 3
 FI
 pcolr0=5 ;third-intensity cursor
 state=0
 RETURN

PROC attrib(BYTE chr) ; enhance
 IF chr = '0 THEN
  Zero(enhance, 8)
 ELSEIF chr = '4 THEN ; reverse
  SetBlock(enhance, 8, $FF)
 ELSEIF chr = '8 THEN ; underline
  Zero(enhance, 8)
  enhance(6) = $FF
 ELSEIF chr = '> THEN ; both modes
  SetBlock(enhance, 8, $FF)
  enhance(6) = $00
 FI
 state = 0
 RETURN

PROC Aesc(BYTE chr) ; escape sequence
 BYTE ch
 CARD i
 IF chr = 'E THEN ; insert line
  dlist=sdlist+3+rowcrs*10
  scroll(23,rowcrs)
 ELSEIF chr = '. THEN ; cursor attrib
  state = CMO ; cursor rendition
  RETURN
 ELSEIF chr = 'G THEN ; inverse off
  state = SGR ; select rendition
  RETURN
 ELSEIF chr = 'I THEN ; back tab
  colcrs = (colcrs - 1) & $F8
```

```
 ELSEIF chr = 'j THEN ; reverse lf
  IF rowcrs > 0 THEN
   rowcrs ==- 1
  ELSE
   scroll(23,0)
  FI
;  Acurse()
 ELSEIF chr = '* THEN ; home & clear
  rowcrs = 0
  colcrs = 0
  Aesc('Y)
; erase to EOS ( y?? )
 ELSEIF chr = 'Y OR chr = 'y THEN
  FOR ch = colcrs TO 79
  DO
   Achr(ch,rowcrs,' )
  OD
  IF rowcrs < 23 THEN
   dlist=sdlist+3+10*rowcrs
   FOR i = rowcrs+1 TO 23
   DO
    dlist==+ 10
    Zero(dlist.adrs,320)
   OD
  FI
;  Acurse()
 ; erase to EOL
 ELSEIF chr = 'T OR chr = 't THEN
  FOR ch = colcrs TO 79
  DO
   Achr(ch,rowcrs,' )
  OD
;  Acurse()
 ELSEIF chr = 'R THEN ; delete line
  dlist=sdlist+3+10*rowcrs
  scroll(rowcrs,23)
 ELSEIF chr = '= THEN ; cursor addr
  state = CUP
  need = 2
  RETURN
 ELSEIF chr = 'Q THEN ; insert char
  dlist=sdlist+3+10*rowcrs
  ins_ch(dlist.adrs,colcrs)
 ELSEIF chr = 'W THEN ; delete char
  dlist=sdlist+3+10*rowcrs
  del_ch(dlist.adrs,colcrs)
 ELSEIF chr = 'Z THEN ; multi-insert
  ins_mode = 1;
 ELSEIF chr = 'r THEN ; end multi-ins
  ins_mode = 0;
 ELSEIF chr = 'b THEN ; BlackOnWhite
  SetColor(2,0,15)
  SetColor(1,0,0)
 ELSEIF chr = 'd THEN ; WhiteOnBlack
  SetColor(2,0,0)
  SetColor(1,0,15)
 FI
 state = 0
 RETURN
```

```
PROC Aopen()
 BYTE i
 sdmctl=0 ;No Antic DMA till DL ready
 vvblkd=vbi ;Put our vbi routine in
 brkky=brk ;Put our break routine in
; Examine data array for alignment
 sdlist=(display & $F000) + $1C80
 pmbase=sdlist RSH 8 ;page of plr/mis
 cursad=sdlist+$100
 dlist=sdlist
 FOR i = 0 TO 2 DO
  dlist.mode = SKIP8
  dlist==+1
 OD
 curln=display
 FOR i=0 TO 23 DO
  IF (curln = sdlist) THEN
   curln==+ $180
  FI
  IF (curln & $FFF) > $EC0 THEN
   curln = (curln & $F000) + $1000
  FI
  dlist.mode = GR8 + LMS
  dlist.adrs = curln
  SetBlock(dlist+3,7,GR8)
  curln==+ 320
  dlist ==+ 10
 OD
 dlist.mode = JVP
 dlist.adrs = sdlist
 SetColor(2,0,0) ; was 0,0,0
 SetColor(1,0,15) ; was 1,12,15
 SetColor(4,0,2) ; new - border
 gprior=8 ;cursor behind letters
 gractl=1 ;turn missiles on
 sdmctl=$26 ;enable missile DMA
 Zero(cursad,128) ;clear missile data
 cursad ==+ 16 ;true top of screen
 state = 0
 lmargin = 0
 rmargin = 79
 rowcrs = 0
 colcrs = 0
 Acurse()
 attrib('0) ;normal video
 cursat('4) ;normal cursor
 [LDY 1 RTS]

PROC Aclose()
 [LDY 1 RTS]

PROC Aput(BYTE areg)
 IF state = ESC THEN; escape sequence
  Aesc(areg)
 ELSEIF state = CUP THEN ; cursor pos
  curpos(areg)
 ELSEIF state = CMO THEN ; curs enhnc
  cursat(areg)
```

```
 ELSEIF state = SGR THEN ; enhance
  attrib(areg)
 ELSEIF areg = $1B THEN ; ESC
  state = ESC
 ELSEIF areg = $1A THEN ; ClearScreen
  colcrs = 0
  rowcrs = 0
  Aesc('Y)
 ELSEIF areg = $1E THEN ; home
  colcrs = 0
  rowcrs = 0
;  Acurse()
 ELSEIF areg = $0B THEN ; cursor up
  IF rowcrs > 0 THEN
   rowcrs ==- 1
;   Acurse()
  FI
 ELSEIF areg = $16 THEN ; cursor down
  IF rowcrs < 23 THEN
   rowcrs ==+ 1
;   Acurse()
  FI
 ELSEIF areg = $0C THEN ; cursor right
  IF colcrs < rmargin THEN
   colcrs ==+ 1
;   Acurse()
  FI
 ELSEIF areg = $0D THEN ; CR
  colcrs = lmargin ; was = 0
 ELSEIF areg = $0A THEN ; lf
  rowcrs ==+ 1
  IF colcrs > rmargin THEN
   colcrs = 0 ;altos vi kludge
  FI
  Ascroll()
 ELSEIF areg = $08 THEN ; BS
  IF colcrs > lmargin THEN ;was >0
   colcrs ==- 1
  ELSEIF rowcrs > 0 THEN ;bs wraps
   colcrs = rmargin
   rowcrs==- 1
  FI
 ELSEIF areg = $07 THEN ; bell
  saudc1 = $F0
; bel = 16 ; vbi() will pickup
 ELSEIF areg = $09 THEN ; TAB
  colcrs = (colcrs + 8) & $F8
  Amargin()
 ELSEIF areg > $1F THEN ;printable
  Amargin() ;new here
  Achr(colcrs,rowcrs,areg)
  colcrs ==+ 1
;ELSE unrecognized control-nothing
 FI
 Acurse() ;update cursor
 [LDY 1 RTS]

PROC Anofunc()
 [RTS]
```

```
PROC Adummy()
 [LDY 1 RTS]

PROC Ahandler()
 BYTE ARRAY hatabs = $031A
 BYTE pos, found
 ;do not change the following 3 lines
 CARD ARRAY atab(6)
 BYTE Jump = [JMP]
 CARD init
 ; define device entry points
 atab(0) = Aopen - 1       ;OPEN
 atab(1) = Aclose - 1      ;CLOSE
 atab(2) = Anofunc - 1     ;READ
 atab(3) = Aput - 1        ;WRITE
 atab(4) = Adummy - 1      ;STATUS
 atab(5) = Anofunc - 1     ;SPECIAL
 init = Adummy             ;INIT
 ; find entry in hatabs
 found = 0;
 pos = 0
 WHILE (pos < 34) AND (found = 0)
 DO
  IF hatabs(pos) = 0 THEN
   found = 1
  ELSE
   pos ==+ 3
  FI
 OD
 IF found = 0 THEN
  PrintE("*** A: too many devices")
 ELSE
  hatabs(pos) = 'A
  hatabs(pos + 1) = atab & 255
  hatabs(pos + 2) = atab RSH 8
 FI
 RETURN

;******************************
;*  MAIN PROGRAM
;******************************

MODULE

BYTE
 ch = $02FC,
 bcount = $02EB,
 speed = [3], ;CWG 1=300BPS 3=1200
 wsize = [0],
 sbits = [0],
 lf = [0],
 iparity = [0],
 oparity = [0]

PROC init_R(); set options for R:
 Close(3)
 Open(3,"R:",13,0)
 XIO(3,0,38,lf*64+32+iparity*4+oparity,0,"R1:") ;32=no xlate
```

```
 XIO(3,0,36,speed+7+wsize*16+128*sbits,0,"R1:")
 XIO(3,0,34,192,0,"R1:")
 XIO(3,0,40,0,0,"R1:") ;concurrent IO
 bcount = 0
 RETURN

PROC init_A(); set up A: device
 Ahandler() ; install A: handler
 Close(2)
; Graphics(8+16)
 Open(2,"A:",8,0)
 RETURN

PROC intro()
 soundr=0
 Close(7)
 Open(7,"K:",4,0)
 keydef = xlate ; load key translate
 init_R()
 init_A()
 soundr=0
 RETURN

PROC do_remote(); process remote
 BYTE chr
 BYTE stop = [0]
 BYTE HIWAT = [128]
 XIO(3,0,13,0,0,"R:")
 IF bcount > HIWAT THEN
  PutD(3,$13)     ;XOFF
  stop=1
 FI
 WHILE bcount > 0
 DO
  chr = GetD(3)
  PutD(2,chr & $7F)
  bcount ==- 1
 OD
 IF stop THEN
  PutD(3,$11)     ;XON
  stop=0
 FI
 RETURN

PROC do_local(); process local
 BYTE chr
 IF $FF - ch THEN        ;INPUT
  chr = GetD(7)
  PutD(3,chr)
 FI      ; END IF CHAR
 RETURN

PROC main()
 intro()
 DO
  do_remote()
  do_local()
 OD
 RETURN
```