Jupiter Ace Graphics

Graphics RAM

The video RAM is located from 2400 hex to 26ff hex. It's a 32 (wide) x 24 (high) array of bytes where the top left byte is at 2400 and the bottom right byte is at 26ff. Each byte in this array contains the ascii code of the character on the screen. The byte comprises; bit 7 = 0 for normal display and bit 7 = 1 for inverted display. Bits [6:0] are the ascii codes.

```
: BOUNDS
 CLS ASCII 1 VIDEO C! ASCII 2
 VIDEO 31 + C! ASCII 3
 VIDEO 32 + C! ASCII 4
 VIDEO 32 + 31 +
 C! ASCII 5 VIDEO 32 21
 * + C! 5
;
```

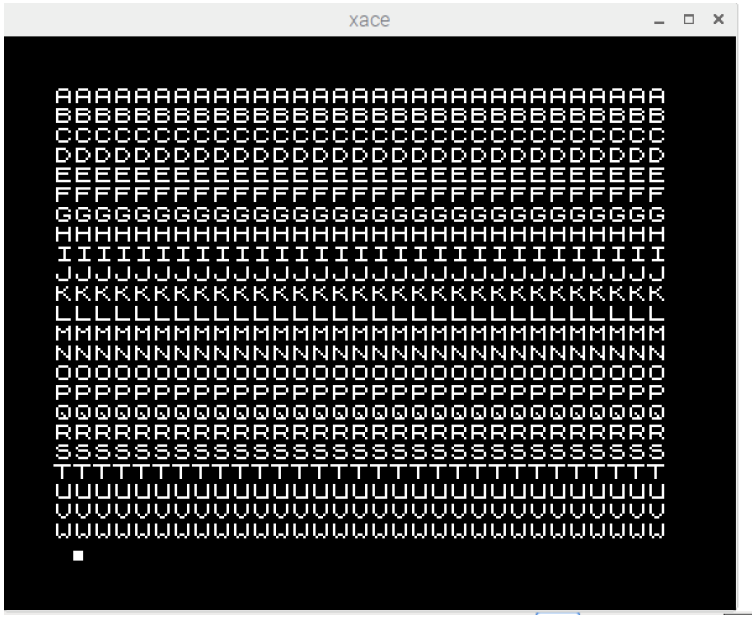
The word BOUNDS writes some characters close to the boundary of the screen**. Be careful not to write to the bottom line as this can corrupt the input PAD array.** What we write on the bottom line <may> be interpreted as a command. [This might allow us to write lambdas…. Code which writes its own code….. but I'd stay clear of that one in Forth.]

Note: The video RAM is also mapped from 2000 to 23FF. On a real ACE, the lower addresses are prioritised for software and the higher addresses are prioritised for hardware.

The AT Word

AT takes two numbers of the stack to define the position in which to start printing. (line column -) where the lines are enumerated from the top of the screen.

```
list cells
: CELLS
 23 0
 DO
  31 0
  DO
   J I AT ASCII A J
   + EMIT
  LOOP
 LOOP
;
  OK
```



The word CELLS uses AT to write characters A to W on each line/row.

The Character Set RAM

The Jupiter Ace manual contains some errors in describing overlapping memory space in hex. The character set RAM is from 11264 **decimal** and 12287 **decimal**. Each group of 8 bytes contains the bit-map of an ascii character. By writing to these bit fields, we can generate custom characters, invent new fonts, fake high resolution plotting or achieve smooth graphical motion. (Similar to the Sinclair Spectrum.)

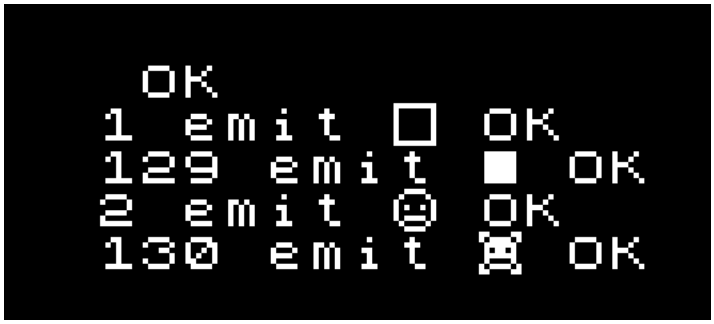
The Character Set RAM is write-only, but can be restored by block-copying the original bit fields from ROM.

```
 OK
list gr
: GR
 ( 8 row numbers , ascii code -
)
 8 * 11263 + DUP
 8 +
 DO
  I C! -1
 +LOOP
;
  OK
```

The word GR comes straight from the Ace manual. It expects 8 rows of bit map and the character code on the stack.

```
   OK
list box
: BOX
 255 129 129 129 129
 129 129 255 1 GR
;
   OK
list smile
: SMILE
 60 66 129 165 129
 189 66 60 2 GR
;
   OK
```

The words BOX and SMILE customise characters 1 and 2 to produce a square box and a smiley face respectively.



BOX and SMILE also implicitly cause characters 129 and 130 to display the inverse bit-patterns.

Note: Running the GR work on a character will cause ALL instances of that character on the screen to change. While this might seem annoying, similar hardware gets used to good effect on Sinclair Spectrum machines to create the illusion of fast/parallel graphics processing ….usually by clever Russian and Slovenian guys who have far too much time on their hands and are STILL writing Z80 games in 2019!!!!!

The word CHARS shows all characters from 0 to 127 and their inverses from 128 to 255. We can see our custom BOX and SMILE characters.

The PLOT Word

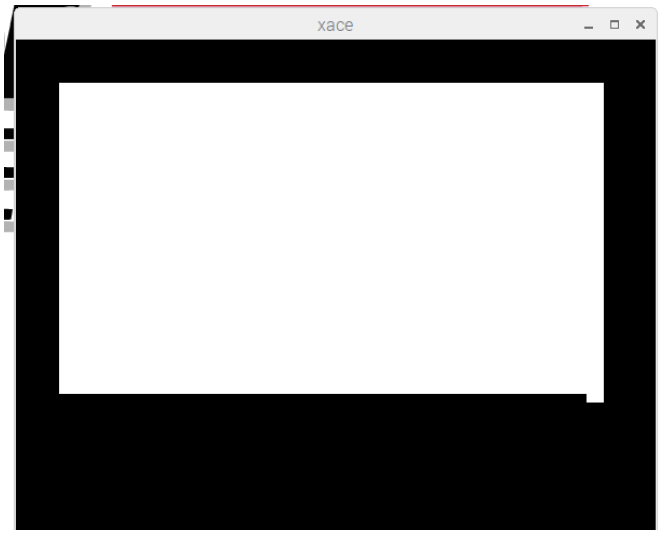
PLOT plots a box on a 32 x 24 grid with its origin in the lower left corner. Avoid plotting on the lower Y values as this can upset the PAD.

Plot expects three numbers on the stack; X , Y and MODE where:-

MODE = 0 set the box black
MODE = 1 set the box white
MODE = 2 do nothing
MODE = 3 toggle the box to the opposite colour

```
 OK
list sweep
:  SWEEP
 47 0
 DO
  63 0
  DO
   I J 3 PLOT
  LOOP
 LOOP
;
  OK
```

The word SWEEP moves from bottom left to top right, toggling each box in the grid.



(Above: screen capture during SWEEP)

VIS and INVIS Words

When plotting or using the AT word, the screen scrolling and/or the OK prompt can interfere with the desired graphics. The word INVIS suppresses these and the word VIS restores scrolling and “OK”

NOTES/OTHER STUFF

Positive

Xace allows us to save and load Forth programs to/from the Linux file system. This setup is very robust and is a huge improvement over the original Ace.

We can run multiple instances of xace on Linux simultaneously in separate X-Windows.

Negative

These simple experiments were performed using xace on a wireless Raspberry Pi Zero. The combination of xace and Raspbian is currently unstable. Xace crashed on one occasion and the Pi Zero’s screen would go dark for a few seconds every minute. Also, xace struggles to keep up with the input buffer on USB, so typing must be slow enough to avoid dropping key strokes. (Not a problem on PI3 or PI4).

Cheap and nasty Jupiter Ace substitute. Having a simple ACE/Fig/Forth79/Forth83 sitting on top of ForthWin is a better long-term substitute.