Proposed Video Game System Vocabulary

This is a description of the VGS vocabularies. The words are presented in ASCII order. The first line of each entry shows a symbolic description of the action of the word: Symbols indicating which parameters are to be placed on the stack before executing the word, 3 dashes (---) indicating execution, then any parameters left on the stack by the word. In this notation, the top of the stack is to the right. If the place of the word in the input string is not completely obvious, it is shown explicitly. If no dashes are shown the word does not affect the stack. Symbols are used as follows:

b	Block number
c	7-bit ASCII character code
f	Flag: 0=False, non-zero=True. All words which return a flag return 0 or 1
m n p q r	16-bit integers
nnnn pppp	The name of a word
ssss	A string of characters
x y z	16 bit integers

Immediately following the name of a word, certain characters may appear within paraenthesees. These denote some special action or characteristics:

C	The word may be used only within a colon-definition. A following digit (C0 or C2) indicates the number of memory cells used when the word is compiled if other than one. A following + or - sign indicates that the word either pushes or pops a value on the stack during compilation. (This action is not related to its action during execution.)
E	The word may not normally be compiled within a colon-definition.
P	The word has its immediate bit set; it is executed directly, even when encountered during compile mode.
U	The word applies to a user variable (in a multi-user system each user would have his own copy.)

Unless stared otherwise, all references to numbers apply 16-bit integers, with the most significant bit as the sign bit and the negative in two's complement form. Similarly, all arithmetic will be assumed to be 16-bit signed integer arithmetic with error and overflow indication unspecified.

Standard Definitions

SWAB m --- n
Swap high and low bytes in integer m

ROTN m --- n
Rotate integer m to the right 4 bits, bit 0 going to bit 12

VECTOR m n o --- p
Do the following n times: Do the the following o times, then increment m: Take value at address m, add 2 to m, and add value to value at address m

Same as doing the following equation: a=a+(b+(c+(... for o number of terms.

Typical use would be vector-address 2 1 VECTOR, which would vector in x and y using x and y velocities.

LIMIT m n o --- p
Compare value at m against n (upper bound) and o (lower bound). Returns -1,0 or +1 for <= lower bound, in limits, and >= upper bound

NEG! m ---
Negate value at address m

RELABS x y s --- p q
Take x,y in scale s and convert to magic screen address (q) and shift amount (p).

Write Routines

SHOW x y s m p r --- f

x	x coordinate
y	y coordinate
s	scale factor on coordinates
m	expand value in upper byte, magic in lower byte
p	pattern address
r	rotation number

Writes pattern to screen. Returns a true/false flag indicating intercept.

0SHOW 1SHOW 2SHOW 3SHOW 4SHOW 5SHOW 6SHOW 7SHOW
Equivalent to Rotation-Number SHOW

0SHOWI x y s m p w r 0SHOWI s
Inputs same as 0SHOW, except w which is an address of a window table. 0SHOWI does exhastive intercept checking: Returns s indicating whether an intercept occured on color 0 1 2 and/or 3 in bits 0 1 2 3.
A window table consists of 4 bytes:
    starting x byte
    starting y line

width in bytes
length in lines

String Display Verbs

POST x y m s l f —
Writes string at address s of length l using font-address f to screen at location (x,y) with magic/expand value of m

1POST x y m c f —
Write character c using font f to screen at location (x,y) with magic/expand value m

#POST x y m n l f —
Display number n using font f and field width l at (x,y) with magic/expand of m

#TOA n o l r —
Convert binary word n to ascii in radix r. Puts converted string in locations o+1,o+2.. with length in location o. Buffer at o must be a minimum of 7 bytes, maximum of l bytes. If l is 0 number is left justified, otherwise, it is right justified, zero suppressed in a field of l spaces long

CLOCK x y m n f —
Converts and displays time n in binary to screen at (x,y) using font f and magic/expand value m. Displays time in minutes and seconds with minutes zero suppressed and seconds zero filled

ERASE —
Erases the whole screen to 0 filled

1DOT x y m — s
2DOT
Show a 1 pixel or 2x2 pixel dot at (x,y) with magic m. Returns s indicating complete intercept status in bits 3 2 1 0 for writing over pixel values 3 2 1 or 0

DRAW x y m n — f
Draw a line between (x,y) and (m,n) using magic/expand/fill of m. Returns f a true/false flag indicating intercept.

BOX x y m n m — f
Show a rectangle with a diagonal from (x,y) to (m,n) with magic/expand m. Returns f a true/false value indicating intercept.

ELLIPSE x y m n m — f
Show ellipse centered at (x,y) with x-radius m and y-radius n using magic/expand/fill of m

SCROLL x y m n l —
Scroll window with diagonal (x,y) (m,n) by l lines. l may be positive or negitive.

SHADE x y p —
Fills in shape bounded by pixel value p with pixel value p.

Special Input/Output Verbs

COLOR m —
Outputs color-list at address m to color ports.

FLOOD m —
Sets all color ports to value m

VERTICAL l —
Sets vertical blanking to start at line number l.

HORIZONTAL m n —
Sets horizontal color boundary to byte m and outside frame pixel color to n.

RANDOM m — n
Takes seed from address m, generates a new random value n, stores the new value n into address m as new seed

SOW m n o —