LINE-A Function Reference

$A000 - Initialize

Return pointers to the **Line-A** variable structures.

EXAMPLE BINDING

```
; Retrieve Line-A variable table address
; and store in A5 for other bindings

             .dc.w            $A000
             .move.l          a0,a5    ; Line-A variables
             .move.l          a1,a6    ; System font headers
```

RETURN VALUE

The initialize function returns the following information:

Register	Contents
D0	Pointer to **Line-A** variable table.
A0	Pointer to **Line-A** variable table.
A1	Pointer to a **NULL** terminated array of pointers to system font headers.
A2	Pointer to a longword array containing sixteen pointers which are addresses of the actual **Line-A** functions in memory. For example, JSR'ing through the pointer in the first array element has the same result as calling the **Initialize** instruction by an exception except that the function must be called from supervisor mode.

COMMENTS

This call is required to return the address of the **Line-A** variable structure needed for all other **Line-A** calls. All processes (including the **VDI**) share this structure so don't expect variables to remain constant between calls.

SEE ALSO

v_opnvwk()

$A001 - Plot Pixel

Plot a single pixel at the specified coordinates.

PARAMETERS

INTIN points to a **WORD** containing the color register of the pixel to plot at the specified coordinates. *PTSIN* points to two **WORD**s which are the X and Y coordinates respectively.

EXAMPLE BINDING

```
; Plot a pixel at ( 10, 10 ) using color 1

             move.l           #intin,8(a5)
             move.l           #ptsin,12(a5)
             .dc.w            $A001

             .data
intin:
             .dc.w            1
ptsin:
```

```
                    .dc.w            10, 10
```

SEE ALSO **v_pmarker()**

$A002 - Get Pixel

Get the color register of the pixel at the specified coordinates.

PARAMETERS *PTSIN* points to two words which are the X and Y coordinates of the pixel to read.

EXAMPLE BINDING

```
; Read the color index of point ( 10, 10 )

             move.l       #ptsin,12(a5)
             .dc.w        $A002

             .data
ptsin:
             .dc.w        10, 10
```

RETURN VALUE The color register of the pixel is returned in D0.

SEE ALSO **v_getpixel()**

$A003 - Arbitrary Line

Draw a line between any two coordinates.

PARAMETERS *COLBIT0-4* are set appropriately to determine the line color. *LSTLIN* is a flag in which a value of 0 specifies to draw the last point in each line or a value of 1 which specifies not to. *LNMASK* specifies the pattern mask to apply to the line. *WRMODE* specifies the write mode of the function (0-3). (*X1*, *Y1*), and (*X2*, *Y2*) give the starting and ending coordinates of the line.

EXAMPLE BINDING

```
;Draw a solid line from ( 0, 0 ) to ( 100, 100 )

       move.w        #1,24(a5)         ; COLBIT 0
       move.w        #1,26(a5)         ; COLBIT 1
       move.w        #1,28(a5)         ; COLBIT 2
       move.w        #1,30(a5)         ; COLBIT 3
       move.w        #0,32(a5)         ; LSTLIN
       move.w        #$FFFF,34(a5)         ; LNMASK
       move.w        #0,36(a5)         ; WRMODE
       move.w        #0,38(a5)         ; X1
       move.w        #0,40(a5)         ; Y1
       move.w        #100,42(a5)       ; X2
       move.w        #100,42(a5)       ; Y2
       .dc.w         $A003
```

CAVEATS *LNMASK* is modified as a result of this call.

SEE ALSO **$A004, v_pline()**

$A004 - Horizontal Line

Draw a horizontal line between the specified coordinates.

PARAMETERS *COLBIT0-3* defines the color of the line and *WRMODE* determines the write mode (0-3). (*X1*, *Y1*) and (*X2*, *Y1*) determine the starting and ending points of the line. *PATMSK* is AND'ed with *Y1* to determine a line index into the pattern pointed to by *PATPTR*. *PATMSK* is normally the number of lines in the pattern (should be an even power of 2) minus one. If *MFILL* is non-zero, *WMODE* is disregarded and the fill is colored from the values in *COLBIT0-3*.

EXAMPLE BINDING

```
;Draw a horizontal dashed line from ( 0, 10 ) to ( 100, 10 )

             move.w          #1,24(a5)        ; COLBIT 0
             move.w          #1,26(a5)        ; COLBIT 1
             move.w          #1,28(a5)        ; COLBIT 2
             move.w          #1,30(a5)        ; COLBIT 3
             move.w          #0,36(a5)        ; WRMODE
             move.w          #0,38(a5)        ; X1
             move.w          #0,40(a5)        ; Y1
             move.w          #100,42(a5)      ; X2
             move.l          #pat,46(a5)      ; PATPTR
             move.w          #0,50(a5)        ; PATMSK
             move.w          #0,52(a5)        ; MFILL
             .dc.w           $A004
```

SEE ALSO **v_pline()**

$A005 - Filled Rectangle

Draw a filled rectangle at the specified coordinates.

PARAMETERS *CLIP* is a flag which when set to 1 enables clipping and when set to 0 disables it. All output of this function is confined to the region bounded by (*XMINCL*, *YMINCL*) and (*XMAXCL*, *YMAXCL*). Other parameters are consistent with the definitions given under **$A004**.

EXAMPLE BINDING

```
; Draw a filled rectangle with its upper
; left corner at ( 0, 0 ) and its lower
; right corner at ( 100, 100 ). Clip the
; rectangle to within ( 10, 10 ) and
; ( 90, 90 )

             move.w          #1,24(a5)        ; COLBIT0
             move.w          #1,26(a5)          ; COLBIT1
             move.w          #1,28(a5)          ; COLBIT2
             move.w          #1,30(a5)          ; COLBIT3
             move.w          #0,36(a5)          ; WRMODE
             move.w          #0,38(a5)          ; X1
             move.w          #0,40(a5)          ; Y1
             move.w          #100,42(a5)        ; X2
             move.w          #100,44(a5)        ; Y2
             move.l          #stipple,46(a5)    ; PATPTR
             move.w          #1,50(a5)          ; PATMSK
             move.w          #0,52(a5)          ; MFILL
             move.w          #1,54(a5)          ; CLIP
             move.w          #10,56(a5)         ; XMINCL
             move.w          #10,58(a5)         ; YMINCL
             move.w          #90,60(a5)         ; XMAXCL
             move.w          #90,62(a5)         ; YMAXCL
             .dc.w           $A005

             .data
stipple:
             .dc.w           $AAAA
             .dc.w           $5555
```

SEE ALSO **v_bar(), vr_recfl()**

$A006 - Filled Polygon

Draw a filled polygon line-by-line.

PARAMETERS *PTSIN* contains the X/Y coordinate pairs of the vertices of the polygon with the last point being equal to the first. *CONTRL[1]* specifies the number of vertices. The rest of the variables are consistent with previous usages.

EXAMPLE BINDING

```
; Draw a filled polygon with vertices at
; ( 0, 0 ), ( 319, 120 ), and ( 25, 199 ).

move.l       #ptsin,12(a5)          ; PTSIN
move.l       #contrl,4(a5)          ; CONTRL
move.w       #1,24(a5)         ; COLBIT0
move.w       #1,26(a5)         ; COLBIT1
move.w       #1,28(a5)         ; COLBIT2
move.w       #1,30(a5)         ; COLBIT3
move.w       #0,36(a5)         ; WRMODE
move.w       #stipple,46(a5)   ; PATPTR
move.w       #1,50(a5)         ; PATLEN
move.w       #0,52(a5)         ; MFILL
move.w       #0,54(a5)         ; CLIP

; loop to draw the polygon
move.w       #0,40(a5)         ; upper Y line
move.w       #199,d4           ; lowest Y line
                               ; - upper Y line
loop:
             .dc.w             $A006
addq.w       #1,40(a5)
```

```
dbra          d4,loop

              .data
ptsin:
              .dc.w           0, 0, 319, 120, 25, 199, 0, 0
contrl:
              .dc.w           0, 3
stipple:
              .dc.w           $AAAA
              .dc.w           $5555
```

CAVEATS Register A0, *X1*, and *X2* are destroyed as a result of this call.

SEE ALSO **v_fillarea()**

$A007 - BitBlt

Perform a bit-block transfer.

PARAMETERS The address of a **BitBlt** parameter block is passed in register A6. That structure is defined with the following members:

Member	Offset/Type	Meaning
B_WD	+0 (**WORD**)	Width of block to blit (in pixels)
B_HT	+2 (**WORD**)	Height of block to blit (in pixels)
PLANE_CT†	+4 (**WORD**)	Number of bit planes to blit.
FG_COL†	+6 (**WORD**)	Bit array used to create index into **OP_TAB**. **FG_COL** contributes its bit #'n' (where 'n' is the plane number) to bit #1 of the index used to select the operation code from **OP_TAB**.
BG_COL†	+8 (**WORD**)	Bit array used to create index into **OP_TAB**. **BG_COL** contributes its bit #'n' (where 'n' is the plane number) to bit #0 of the index used to select the operation code from **OP_TAB**.
OP_TAB	+10 (**LONG**)	**OP_TAB** is a 4 byte array containing four logic operation codes (0 to 16) to be applied to the image. The table is indexed by using the bit in **FG_COL** and **BG_COL** corresponding to the current plane as bit #1 and bit #0 respectively yielding an offset into **OP_TAB** of 0-3.
S_XMIN	+14 (**WORD**)	X pixel offset to source upper left.
S_YMIN	+16 (**WORD**)	Y pixel offset to source upper left.
S_FORM	+18 (**WORD**)	Address of the source form.
S_NXWD	+22 (**LONG**)	Number of bits per pixel.
S_NXLN	+24 (**WORD**)	Byte width of form.
S_NXPL	+26 (**WORD**)	Byte offset between planes (always 2).
D_XMIN	+28 (**WORD**)	X pixel offset to destination upper left.
D_YMIN	+30 (**WORD**)	Y pixel offset to destination upper left.

D_FORM	+32 (**LONG**)	Address of the destination form.
D_NXWD	+36 (**WORD**)	Number of bits per pixel.
D_NXLN	+38 (**WORD**)	Byte width of form.
D_NXPL	+40 (**WORD**)	Byte offset between planes (always 2).
P_ADDR	+42 (**LONG**)	Address of pattern buffer (0 = no pattern).
P_NXLN	+46 (**WORD**)	Bytes of pattern per line (should be even).
P_NXPL	+48 (**WORD**)	Bytes of pattern per plane (if using a single plane fill with a multi-plane destination, this should be 0).
P_MASK	+50 (**WORD**)	**P_MASK** is found by the expression: If **P_NXLN** = 2 ^ n then **P_MASK** = (length in words - 1) << n
SPACE	+52 (**WORD**)	24 bytes of blank space which must be reserved as work area for the function.

†These members may be altered by this function.

EXAMPLE BINDING

```
; Perform a blit using the information located
; at bprmblk

            lea         bprmblk,a6
            .dc.w           $A007
```

SEE ALSO **vro_cpyfm(), vrt_cpyfm()**

$A008 - TextBlt

Blit a single character to the screen.

PARAMETERS When performing this call, the following **Line-A** variables are evaluated:

Variable	Meaning
WMODE	Writing mode (see comments below).
CLIP, XMINCL, YMINCL, XMAXCL, YMAXCL	Standard clipping flags and extents.
XDDA	Scaling accumulator (should be initialized to $8000 prior to each **TextBlt** call when scaling).
DDAINC	This amount specifies the fractional amount to scale the character outputted by. If scaling down, this value may by found by the formula: 0x100 * scaled size / actual size If scaling up, this value may be found with the formula: 0x100 * (scaled size - actual size) / actual size This variable is only evaluated if scaling is active.
SCALDIR	Scaling direction (1 = up, 0 = down).

MONO	If 1 set to monospacing mode, if 0 set to proportional spacing mode.
SOURCEX, SOURCEY	**SOURCEX** is the pixel offset into the font form of the character you wish to render. **SOURCEY** is usually 0 indicating that you wish to render the character from the top.
DESTX, DESTY	**DESTX** and **DESTY** specify the destination screen coordinates of the character.
DELX, DELY	**DELX** and **DELY** specify the width and height of the character to print.
FBASE	Pointer to start of font data.
FWIDTH	Width of font form.
STYLE	**STYLE** is a mask of the following bits indicating special effects: 0x01 = Bold 0x02 = Light 0x04 = Italic 0x08 = Underlined 0x10 = Outlined
LITEMASK	Mask used to lighten text (usually $5555).
SKEWMASK	Mask used to italicize text (usually $5555).
WEIGHT	Width by which to thicken boldface text (should be set from font header).
ROFF	Offset above character baseline when skewing (set from font header).
LOFF	Offset below character baseline when skewing (from font header).
SCALE	Scaling flag (0 = no scaling, 1 = scale text).
CHUP	Character rotation vector (may be 0, 900, 1800, or 2700).
TEXTFG	Text foreground color.
SCRTCHP	Pointer to start of text special effects buffer (should be twice as large as the largest distorted character and is only required when using a special effect).
SCRPT2	Offset of scaling buffer in **SCRTCHP** (midpoint).
TEXTBG	Text background color.

EXAMPLE BINDING

```
; Print a NULL-terminated string with
; no effects or clipping

             move.w          #0,36(a5)          ; WMODE
             move.w          #0,54(a5)          ; CLIP
             move.w          #1,106(a5)         ; TEXTFG
             move.w          #0,114(a5)         ; TEXTBG
             move.w          #100,76(a5)        ; DESTX
             move.w          #100,78(a5)        ; DESTY
             move.w          #4,90(a5)          ; STYLE
             move.w          #0,102(a5)         ; SCALE
             move.w          #1,70(a5)          ; MONO

; Find the 8x8 font
             move.w          4(a6),a6           ; Address of 8x8
                                                ; font
             move.w          76(a6),84(a5)          ; FBASE
             move.w          80(a6),88(a5)          ; FWIDTH
             move.w          82(a6),82(a5)          ; DELY

; Print the string
             lea             string,a2
             move.l          72(a6),a3          ; offset table
             moveq.l         #0,d0
print:
             move.b          (a2)+,d0          ; Get next char
             ble             end
             sub.w           36(a6),d0         ; Fix offset
             lsl.w           #1,d0             ; Double for
                                               ; WORD offset
             move.w          0(a3,d0),72(a5)   ; SOURCEX
             move.w          2(a3,d0),d0       ; x of next char
             sub.w           72(a5),d0         ; get true width
             move.w          d0,80(a5)         ; DELX
             moveq.l         #0,74(a5)         ; SOURCEY
             movem.l         a0-a2,-(sp)       ; Save a0-a2
             .dc.w           $A008
             movem.l         (a7)+,a0-a2       ; Restore regs
             bra         print
end:
             rts

             .data
string:
             .dc.b           "The Atari Compendium",0
```

COMMENTS The value for *WMODE* is a special case with **TextBlt**. Values from 0-3 translate to the standard **VDI** modes. Values from 4-19 translate to the **BitBlt** modes 0-15.

SEE ALSO **v_gtext()**

$A009 - Show Mouse

Show the mouse cursor.

PARAMETERS No parameters required. Optionally, *INTIN* can be made to point to a **WORD** value of 0 to force the mouse cursor to be displayed regardless of the number of times it was hidden.

EXAMPLE BINDING

```
; Show the mouse regardless of the number
; of times it was hidden

             move.l          #intin,8(a5)      ; INTIN
             .dc.w           $A009

             .data
intin:
             .dc.w           0
```

COMMENTS 'Show' and 'Hide' mouse calls are nested, that is, in order to return the mouse cursor to its original state, it must be 'shown' the same number of times it was 'hidden'.

SEE ALSO **v_show_c(), graf_mouse()**

$A00A - Hide Mouse

Hide the mouse cursor.

EXAMPLE BINDING

```
; Remove the mouse from the screen

             .dc.w           $A00A
```

COMMENTS See ‘**Show Mouse**’.

SEE ALSO **v_hide_c(), graf_mouse()**

$A00B - Transform Mouse

Change the mouse’s form.

PARAMETERS On entry *INTIN* should point to a structure containing the new mouse form data. The format of the structure is defined under the entry for **vsc_form()**.

EXAMPLE BINDING

```
; Change the mouse form to the data held in
; the newmouse structure.

             move.b          -339(a5),d0       ; Save old value
             move.b          #0,-339(a5)       ; Disable mouse
                                               ; interrupts
             move.l          #newmouse,8(a5)   ; INTIN
             .dc.w           $A00B
             move.b          d0,-339(a5)       ; Restore
                                               ; MOUSE_FLAG
```

COMMENTS The old data can be saved from the information stored in the **Line-A** variable table at offset -356. To avoid ‘mouse droppings’ you should disable mouse interrupts by setting *MOUSE_FLAG* (offset -339) to 0 and restoring it when done.

SEE ALSO **vsc_form(), graf_mouse()**

$A00C - Undraw Sprite

Undraw a previously drawn sprite.

PARAMETERS Prior to calling this function, A2 should be loaded with a pointer to the ‘sprite save block’ defined when drawing the sprite. For the format of this data, see ‘**Draw Sprite**’

EXAMPLE

```
; ‘Undraw’ sprite previously drawn from data
```

BINDING

```
; stored in savesprite.

             lea          savesprite,a2
             .dc.w            $A00C
```

CAVEATS

Register A6 is destroyed as a result of this call.

COMMENTS

When 'undrawing' sprites, they should be removed in reverse order of drawing to avoid the possibility of creating garbage on screen.

$A00D - Draw Sprite

Draw a 16x16 sprite on the screen.

PARAMETERS

Prior to calling this function, four 68x00 registers must be initialized. D0 and D1 should contain the horizontal and vertical position respectively of the coordinates of the sprite to draw. This is relative to the 'hot spot' of the sprite as defined in the sprite definition block.

A0 should contain a pointer to a sprite definition block defined as follows:

Offset/Type	Meaning
0x0000 (**WORD**)	X offset of 'hot spot'. This value is subtracted from the value given in D0 to yield the actual screen position of the upper-left pixel.
0x0002 (**WORD**)	Y offset of 'hot spot'. This value is subtracted from the value given in D1 to yield the actual screen position of the upper-right pixel.
0x0004 (**WORD**)	Format flag. This value specifies the mode in which the mouse pointer will be drawn. A value of 1 specifies '**VDI** mode' whereas -1 specifies X-OR mode. The default is 1.
0x0006 (**WORD**)	Background color of sprite.
0x0008 (**WORD**)	Foreground color of sprite.
0x000A (32 **WORD**s)	Sprite form data. The bitmap data consists of two 16x16 rasters, one each for the mask and data portion of the form. The data is presented in interleaved format. The first **WORD** of the mask portion is first, followed by the first **WORD** of the data portion, and so on.

Register A2 is a pointer to a buffer which will be used to save the screen area where the sprite is drawn. The size of the buffer can be determined by the following formula:

$$(10 + (\mathit{VPLANES} * 64))$$

EXAMPLE BINDING

```
; Draw a sprite at ( 100, 100 ) whose data
; is stored at spritedef with a valid save
; buffer at savebuf.

             move.w          #100,d0          ; X position
             move.w          #100,d1          ; Y position
             move.l          #spritedef,a0        ; Sprite form
             move.l          #savebuf,a2      ; Save buffer
             .dc.w           $A00D
```

CAVEATS Register A6 is destroyed as a result of this call.

COMMENTS In order to avoid the mouse form running into any sprites you draw, the mouse should be hidden before drawing and restored afterwards. It may also be advisable to call **Vsync()** prior to each call to avoid screen flicker.

$A00E - Copy Raster

Copy a raster form using opaque or transparent mode.

PARAMETERS *INTIN* should point to a **WORD** array whose first entry specifies the write mode of the operation. In transparent mode, this is a **VDI** standard mode (0-3), however in opaque mode the full range of **BitBlt** modes (0-15) are available. In transparent mode, the second and third array entries of *INTIN* contain the foreground and background color of the destination copy respectively.

CONTRL should point to a memory buffer which is filled in with the source and destination **MFDB**'s (Memory Form Definition Block's) at offsets 14 and 18 respectively. The structure of an **MFDB** is discussed under **vro_cpyfm()**.

PTSIN should point to an array of 8 **WORD**'s containing the pixel offsets for the blit in the order SX1, SY1, SX2, SY2, DX1, DY1, DX2, DY2.

COPYTRAN specifies the write mode. A value of 0 indicates an opaque copy while a value of 1 indicates a transparent copy.

The settings for *CLIP*, *XMINCL*, *YMINCL*, *XMAXCL*, and *YMAXCL* are utilitized by this call.

EXAMPLE BINDING

```
; Copy a 32x32 raster form 'myrast' from a
; buffer in memory to the ST medium resolution
; screen at ( 100, 100 ) using transparent mode.

             move.l          #contrl,4(a5)        ; CONTRL
             move.l          #srcmfdb,contrl+14
             move.l          #destmfdb,contrl+18

             move.l          #intin,4(a5)     ; INTIN
             move.l          #ptsin,4(a5)     ; PTSIN
             move.w          #1,116(a5)       ; COPYTRAN
             move.w          #0,54(a5)        ; CLIP

; Fill in some info for MFDB's
```

```
                move.l          #myrast,srcmfdb  ; Source raster
                move.w          #$02,-(sp)       ; Physbase()
                trap            #14
                addq.l          #2,sp
                move.l          d0,destmfdb

                .dc.w           $A00E

                .data
contrl:
                .dc.w           0, 0, 0, 0, 0, 0, 0, 0, 0, 0
intin:
                .dc.w           0, 1, 0
ptsin:
                .dc.w           0, 0, 15, 15, 100, 100, 115, 115
srcmfdb:
                .dc.w           0, 0, 16, 16, 1, 0, 0, 0, 0, 0
destmfdb:
                .dc.w           0, 0, 320, 200, 16, 0, 2, 0, 0, 0
myrast:
                .dc.w           $AAAA,$AAAA,$AAAA,$AAAA
                .dc.w           $5555,$5555,$5555,$5555
                .dc.w           $AAAA,$AAAA,$AAAA,$AAAA
                .dc.w           $5555,$5555,$5555,$5555
                .dc.w           $AAAA,$AAAA,$AAAA,$AAAA
                .dc.w           $5555,$5555,$5555,$5555
                .dc.w           $AAAA,$AAAA,$AAAA,$AAAA
                .dc.w           $5555,$5555,$5555,$5555
```

COMMENTS For a more indepth explanation, refer to the **VDI** calls parallel to these, **vro_cpyfm()** and **vrt_cpyfm()**.

SEE ALSO **vro_cpyfm(), vrt_cpyfm()**

$A00F - Seed Fill

Seed fill an irregularly shaped region.

PARAMETERS *INTIN* points to a word value which specifies the mode of this function. If the value is negative, color mode is used. In color mode, the fill spreads from the initial point until it hits a color other than that of the initial point. If the value is positive, outline mode is used. It then is interpreted as the **VDI** color index value at which to stop the fill.

PTSIN points to an array of two **WORD**s which specify the X and Y coordinates respectively of the inital fill point.

CUR_WORK should point to a **WORD** array of 16 words with the sixteenth **WORD** being the fill color specified as a **VDI** color index.

WMODE specified the **VDI** writing mode of the fill (0-3). *PATPTR* and *PATMSK* define the fill pattern (as defined in '**Horizontal Line**').

SEEDABORT points to a user routine which can abort the fill, if desired, when called. This routine is called once for each line of the fill. It should zero register D0 to continue or place a non-zero value in it to abort.

EXAMPLE BINDING

```
; Seed fill an area starting at ( 100, 100 )
; in color mode with a clip region defined
; as the VDI rectangle ( 50, 50 ), ( 200, 200 ).

             move.l         #intin,8(a5)         ; INTIN
             move.l         #ptsin,12(a5)            ; PTSIN
             move.l         #cur_work,-464(a5)   ; CUR_WORK
             move.l         #seedabort,118(a5)   ; SEEDABORT
             move.w         #0,36(a5)            ; WMODE
             move.l         #stipple,46(a5)      ; PATPTR
             move.w         #0,50(a5)            ; PATMASK
             move.w         #0,52(a5)            ; MFILL
             move.w         #50,56(a5)           ; XMINCL
             move.w         #50,58(a5)           ; YMINCL
             move.w         #200,60(a5)          ; XMAXCL
             move.w         #200,62(a5)          ; YMAXCL
             .dc.w          $A00F

seedabort:
             moveq.l        #0, d0               ; Clear D0
             rts

             .data
intin:
             .dc.w          -1
ptsin:
             .dc.w          100, 100
cur_work:
             .dc.w          0, 0, 0, 0, 0, 0, 0, 0
             .dc.w          0, 0, 0, 0, 0, 0, 0, 1
stipple:
             .dc.w          $AAAA
             .dc.w          $5555
```

COMMENTS

The clipping variables *XMINCL*, *YMINCL*, *XMAXCL*, and *YMAXCL* must always be set as they are interpreted regardless of the clipping flag.

SEE ALSO

v_contourfill()