

VIDEO BASIC-64

**Graphics and Sound
Software Development Package
For The COMMODORE 64**

YOU CAN COUNT ON
Abacus 
Software

=====

VIDEO BASIC-64

Software Development Package for High Resolution,
Multicolor, Sprite and Turtle Graphics, Sound
and Game Features for the Commodore-64

(C) COPYRIGHT 1984 ROY WAINWRIGHT

=====

56735

ABACUS SOFTWARE
P.O. BOX 7211
GRAND RAPIDS, MI 49510

First Printing - August 1984

COPYRIGHT NOTICE

ABACUS Software makes this package available for use on a single computer only. It is unlawful to copy any of the software onto any medium for any purpose other than backup. It is unlawful to give away or resell copies of any part of this package. Any unauthorized distribution of this product deprives the authors of their deserved royalties. For use on multiple computers, please contact ABACUS Software to make such arrangements.

WARRANTY

ABACUS Software makes no warranties, expressed or implied as to the fitness of this software package for a particular purpose. In no event will ABACUS Software be liable for consequential damages. ABACUS Software will replace any copy of the software which is unreadable if returned within 30 days of purchase. Thereafter it will charge a nominal fee for replacement.

If you are not satisfied with our software, you may return it in original condition within 30 days of purchase with your receipt for a refund. We want you to be a happy customer.

PREFACE

VIDEO BASIC-64 is an extremely powerful software package for those of you who want to develop software using the wonderful graphics and sound capabilities of the Commodore 64. Not only does VIDEO BASIC-64 provide the extended graphics and sound commands to standard Commodore BASIC, but you can also sell or give away programs that use these extended commands without having to pay royalties for the use of the VIDEO BASIC-64 runtime software. This makes VIDEO BASIC-64 ideal for software developers. No longer do you have to force the purchaser of your software to buy an extended BASIC cartridge or diskette. You can give away the runtime version of VIDEO BASIC-64 to anyone who purchases your program.

Although VIDEO BASIC-64 may be new to the marketplace, it has proven to be a solid and reliable product for almost a year now. Two commercially available products already are using the runtime version of VIDEO BASIC-64. These two products are TAS-64 and CADPAK-64. We have proven that VIDEO BASIC-64 not only works well, but that it considerably speeds up software development requiring graphics and/or sound.

Roy Wainwright has produced a product that finally gives other software developers features that were previously available only to machine language programmers. We hope that you can take advantage of this powerful software development tool.

Once again we owe a great deal of thanks to Roy Wainwright for producing such a flexible software tool.

ABACUS Software
August 12, 1984
Grand Rapids, MI



TABLE OF CONTENTS

I.	INTRODUCTION (What is VIDEO BASIC-64?).....	1
II.	GETTING STARTED USING VIDEO BASIC-64.....	3
III.	COMMAND FORMATS.....	5
	TABLE 1 - COLOR NUMBERS.....	5
IV.	HIRES/MULTICOLOR GRAPHICS	
	A. GRAPHIC DISPLAY FORMAT.....	7
	FIGURE 1 SCREEN LAYOUT.....	8
	B. DISPLACEMENT OF AXES.....	9
	C. SCREEN SELECTION.....	9
	D. SAVING AND RESTORING GRAPHIC DISPLAYS.....	9
	E. HIRES/MULTICOLOR COMMANDS.....	10
	F. SCREEN CONTROL COMMANDS.....	19
V.	SPRITE GRAPHICS	
	A. INTRODUCTION TO SPRITES.....	21
	B. SPRITE PICTURES.....	22
	C. SPRITE COMMANDS.....	24
	D. SPRITE EXAMPLE.....	27
VI.	TURTLE GRAPHICS.....	28
	FIGURE 2 - TURTLE DIRECTIONS.....	28
VII.	GAME FEATURES	
	A. INPUT FUNCTIONS.....	31
	B. TIMERS.....	33
	C. SPRITE COLLISION FUNCTIONS.....	34
VIII.	SOUND FEATURES.....	35
	TABLE 2 - TONE/NOTE CHART decimal.....	36
	TABLE 3 - TONE/NOTE CHART hexadecimal.....	40
IX.	OTHER FEATURES	
	A. Repeating commands - BRACKET[and BRACKET].....	42
	B. Exit from repeat loops - BRACKETE.....	42
	C. Hardcopy of graphics screen - HARD.....	43
	D. Reset stack - HIRES 99,0	44
	E. Reset BASIC - HIRES 99,1	45
	F. Disable STOP and STOP/RESTORE - HIRES 99,2	45
	G. Transfer memory - XFER.....	46
	H. Copying graphic screen - REGION.....	47
X.	PROGRAMMING NOTES	
	A. GENERAL INFORMATION.....	49
	B. THREE-DIMENSIONAL PLOTTING.....	49
	C. MEMORY ORGANIZATION.....	50

D. COLOR MEMORY PROBLEM.....	51
E. TESTING AND DEVELOPMENT OF APPLICATIONS.....	52
F. MEMORY MANAGEMENT.....	53
G. MAKING A DISTRIBUTION DISKETTE.....	53
XI. ERRORS.....	55
XII. COMMAND SUMMARY.....	56
XIII. APPENDICES	
APPENDIX A-PRINTER INTERFACE SUPPORT.....	58
APPENDIX B-TONE/PITCH TABLE.....	59
APPENDIX C-CONVERTING FROM ULTRABASIC-64 TO VIDEO BASIC.....	60
APPENDIX D-DUMP FORMAT ON DISK.....	62
APPENDIX E-USING XREF-64 WITH VIDEO BASIC-64.....	63
APPENDIX F-A SHORT LESSON IN HEXADECIMAL NUMBERS..	64
APPENDIX G-COLOR NUMBER TABLE.....	66
APPENDIX H-VIDEO BASIC Tutorial 3.....	67

VIDEO BASIC-64 DEVELOPMENT PACKAGE

I. INTRODUCTION (What is VIDEO BASIC-64?)

The Commodore 64 brings you excellent color graphics, sprite animation, sound and music capabilities at a very low price. Unfortunately for most folks, many of these features are not easy to use from BASIC and require much POKKING and trial and error.

VIDEO BASIC-64 helps you create programs using all of the Commodore 64 features EASILY because it adds many commands to Commodore BASIC. Cartridges or other programs which add graphics features cannot be duplicated, making it uneconomical to create applications. Because VIDEO BASIC-64 includes a runtime version which is designed to be duplicated and distributed without royalty payments, it is now practical to create advanced applications on the Commodore 64.

VIDEO BASIC-64 makes it easy to do the following:

- * Plot in either high resolution or multicolor mode. You can plot dots, lines, boxes, circles, ellipses and write text on the graphics screen in several sizes.
- * Define and manipulate sprites in either high resolution or multicolor formats.
- * Create sound effects and music using any or all of the three voices. The sound commands are "tuned" to the normal chromatic scale.
- * Detect the collision of sprites with background drawings or other sprites.
- * Read the joystick, game paddle or lightpen inputs directly.
- * Draw using a very friendly set of TURTLE GRAPHIC commands.
- * Easily manage two screens (BASIC and graphics) at once.
- * Save and restore graphic screens to disk or tape.
- * Produce hardcopy printout of the graphic screen in two sizes to Commodore, Epson compatible, Okidata or C. Itoh Prowriter graphic printer.
- * Access all memory areas in the '64, including the memory under the ROMs.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

Several extensions and new commands are designed to support advanced graphics functions, such as pattern filling of areas, copying graphics screen areas, etc.

Once VIDEO BASIC-64 is loaded into your computer and started, all of the more than 50 VIDEO BASIC-64 commands are added to the standard Commodore 64 BASIC commands, allowing you to use both sets to create really powerful, compact and efficient programs.

VIDEO BASIC-64 is compatible with the Commodore 1541 DOS WEDGE program. Simply load VIDEO BASIC-64 first and then the wedge.

VIDEO BASIC-64 has two versions - a development version and a runtime version.

The development version gives you all of the capabilities of VIDEO BASIC-64 in both the program mode (RUNning) and command (direct keyboard command) modes. This is used to develop and test application programs using VIDEO BASIC-64 commands.

The runtime version provides all of the command features, but the user cannot LIST, SAVE or modify the program in any way. This version would be included with your application program for distribution.

Demonstration programs and tutorials are included that show you how these features are used.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

II. GETTING STARTED WITH VIDEO BASIC-64

In this section we show you how to use VIDEO BASIC-64. You will learn how to load the program and run the demonstration and tutorial programs.

The demonstration program is on the diskette as a runtime (or user) program and named VBDEMO.

- A. To run the demonstration program as a runtime program, insert the diskette into the disk drive #8.

Type `LOAD "VBDEMO",8` and press `<RETURN>`.

The disk drive should run for about 15 seconds.

When it stops and the `READY.` prompt appears, type `RUN` and press the `<RETURN>` key.

VIDEO BASIC-64 will present a short menu to select the printer of your choice. Choose the number which corresponds to the type of printer attached to your computer. If you have no printer attached, choose any number.

The disk drive will run for another 30 seconds and the demonstration program will start automatically.

- B. To run the demo as a development program, type `LOAD "VBDEV",8` and press `<RETURN>`.

The disk drive will run for a few seconds.

When it stops and the `READY.` prompt appears, type `RUN` and press the `<RETURN>` key.

After the copyright message appears, you are asked to select the printer. Key in one of the numbers and press `<RETURN>`. The rest of the program will load and stop, showing `BREAK IN 62999`. The development version has been loaded. Now type `LOAD "VBDEMO",8` to load the demonstration program. Type `RUN` to start.

The difference between these two methods is that you can list and change the demo program with the development version, but not with the runtime version.

There is a three-part tutorial program which gives a step-

VIDEO BASIC-64 DEVELOPMENT PACKAGE

by-step description of most commands in VIDEO BASIC-64. To start it, type **LOAD "VBTUTOR",8**.

The disk drive will run for about 40 seconds and the **READY.** will show on the screen. Type **RUN** and press **<RETURN>** to start the tutorial. The tutorial stops and prints the prompt **PRESS C TO CONTINUE.** Press the **C** key when you are ready for the next screen.

When you have completed the first part of the tutorial, the second will automatically load and execute. You may start the second part of the tutorial directly by typing **LOAD "VBTUTOR2",8** and **RUN** and press **<RETURN>** when the **READY.** prompt appears.

When you have completed the second part of the tutorial, the third will automatically load and execute. You may start the second part of the tutorial directly by typing **LOAD "VBTUTOR3",8** and **RUN** and press **<RETURN>** when the **READY.** prompt appears.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

III. COMMAND FORMATS

VIDEO BASIC-64 commands have a simple command structure. The graphics commands operate in both the hires and multicolor graphic modes. In the two modes, the commands are nearly identical except for the occasional use of extra color parameters for multicolor mode.

The commands usually have one or more variables (parameters). These must be separated by commas. The parameter may be any valid BASIC expression.

For example, each of these is a valid parameter:

```
A
25*A+15
20*SIN(X/180)
FNA(X)+12
```

The expression need not be enclosed in (), although doing so helps make the program easier to read. Spaces are allowed and also help make the program easier to read.

Colors are designated by the number in TABLE 1 (also in APPENDIX G). For convenience, we have used the numbers on the top of the COMMODORE-64 keys 1-8 represent the first 8 colors in TABLE 1.

<u>COLOR</u>	<u>NUMBER</u>	<u>COLOR</u>
1		BLACK
2		WHITE
3		RED
4		CYAN
5		PURPLE
6		GREEN
7		BLUE
8		YELLOW
9		ORANGE
10		BROWN
11		LIGHT RED
12		DARK GRAY
13		MEDIUM GRAY
14		LIGHT GREEN
15		LIGHT BLUE
16		LIGHT GRAY

Note that these are not the same color numbers that you POKE to color memory.

TABLE 1 COLOR NUMBERS

VIDEO BASIC-64 DEVELOPMENT PACKAGE

IF and **REM** commands require a colon (:) before the graphic command for proper operation. For example --- **IF A=5 THEN DOT 2,3,5** will plot the point at 2,3 every time, regardless of the value of A. For proper operation of the **IF**, change the statement to - **IF A=5 THEN : DOT 2,3,5**. This will plot the point only if A=5. The **REM** also needs a colon to bypass any graphic command following the **REM**.

Note that in the following descriptions of the commands, the symbols [and] denote an optional parameter. Any parameters with the [and] need not be entered unless those features are required.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

IV. HIRES/MULTICOLOR GRAPHICS

A. GRAPHICS DISPLAY FORMAT

The VIDEO BASIC-64 display screen is arranged corresponding to normal graphing conventions. The x-axis goes from left to the right and the y-axis goes from bottom of the screen to the top. X values range from 0 on the left to 319 on the right. Y values range from 0 on the bottom to 199 at the top.

Therefore point 0,0 ($x=0$, $y=0$) is at the lower left corner and point 319,199 ($x=319$, $y=199$) is at the upper right corner.

When functions to be displayed have both plus and minus values, it is necessary to move the axes to the center of the screen by adding constant values in the plotting commands (see next section - DISPLACEMENT OF AXES).

There are two different graphics modes on the COMMODORE-64. They are HIGH RESOLUTION and MULTICOLOR. To select the screen background color, border color and the mode use the HIRES or MULTI commands. The plotting color is controlled by a color number in the command.

Each multicolor point is twice as wide on the screen as each hires point. To keep the proportions and mathematics simple, each multicolor point is also twice as high. In order to make the commands easy to use, the coordinates are the same, but in multicolor commands, only the even numbered points are used. In this case, you should use a STEP 2 in any FOR-loops since points 0,2,4,6 are next to each other in multicolor mode.

The graphic display area is made up internally of 25 rows with 40 sections (cells) in each row (FIGURE 1). Each cell consists of 64 points (8X8) in hires mode or 16 points (4X4) points in multicolor mode. In hires mode, there may be only the background color (set by the HIRES or BLOCK commands) and a plot command color. If two different plot commands put different colors into the cell, all points within the cell will take on the color of the most recent command.

In multicolor mode, there is a background color (only one for the entire screen) plus three different plot command colors. You can think of it by imagining three paintbrushes per cell. Each brush may have any color, but only three are allowed. Colors are designated in commands using numbers 1-16 (1-8 correspond to the keys). Paintbrush "A" is used normally. Paintbrush "B" will be used if the color number is 101-116, and paintbrush "C" will be used if the number is 201-216. (To use paintbrush "B", just add 100 to the color number; to use paintbrush "C", just add 200 to the color

VIDEO BASIC-64 DEVELOPMENT PACKAGE

number).

For most work, you can probably forget about the paintbrushes.

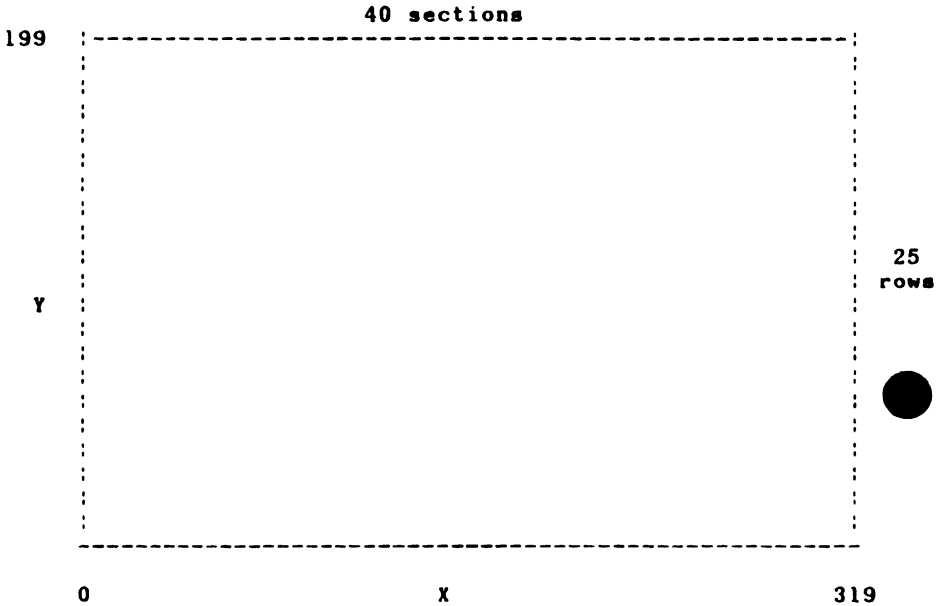


FIGURE 1 GRAPHIC SCREEN LAYOUT

VIDEO BASIC-64 DEVELOPMENT PACKAGE

B. DISPLACEMENT OF AXES

To move the x-axis up to the center of the screen, simply add 100 to the Y-value. For example, use DOTX,Y+100,2. To draw a line as the X-axis, use the command DRAW0,100,319,100,2.

In a similar way, the Y-axis can be placed in the center of the screen. Add 160 to the X-value. For example use DOTX+160,Y+100,2. The new Y-axis can be shown with a command DRAW160,0,160,199,2.

C. SCREEN SELECTION

VIDEO BASIC-64 has two display screens. The normal BASIC text screen is separate from the screen areas used to build the graphic displays. You can switch from the graphic display to the BASIC text screen by pressing Function key 5 (F5). The graphics program is not interrupted by this process. Function key 7 (F7) will switch back to the graphic display.

Your program will automatically switch to the graphic screen when display commands are executed.

D. SAVING AND RESTORING GRAPHIC DISPLAYS

Graphic displays may be saved (dumped) directly to tape or disk. To do this, press the F5 key (function key 5) during a graphic display program (you may have to press the STOP key if the program is running). The BASIC screen will appear. Type DUMP followed by the filename enclosed in "" and press the <RETURN> key. The standard device is the datasette (tape recorder), but you may add a "," and a different device number after the filename (such as disk: ,8). The graphics display will reappear as the dumping takes place.

Example: DUMP "PICTURE",8

Graphic displays can be restored from tape or disk in a similar way using the GREAD command, followed by a filename and optional device number.

Example: GREAD "PICTURE",8

VIDEO BASIC-64 DEVELOPMENT PACKAGE

E. HIRES/MULTICOLOR COMMANDS

HIRES a,b[,c] - setup HIGH RESolution screen

This command clears the graphic screen and sets the screen, border and optionally, the foreground colors for subsequent high-resolution graphics.

- a - this is a value between 1 and 16 which sets the screen (background) color. This number is chosen from TABLE 1.
 - b - this is a value between 1 and 16 which sets the screen border color and is also chosen from TABLE 1.
 - c - this optional parameter is a value between 1 and 16 which is used to set the foreground color in the graphic screen. The default value is 2 (white).
-

MULTI a,b[c,d,e] - setup MULTicolor screen.

This command clears the graphic screen and sets the screen, border and optionally, the foreground colors for subsequent multicolor graphics.

- a - this is a value between 1 and 16 which sets the screen (background) color. This number is chosen from TABLE 1.
- b - this is a value between 1 and 16 which sets the screen border color and is also chosen from TABLE 1.
- c - this optional parameter is a value between 1 and 16 which is used to set the foreground color for paintbrush A in the graphic screen. The default value is 2 (white).
- d - this optional parameter is a value between 1 and 16 which is used to set the foreground color for paintbrush B in the graphic screen. The default value is 3 (red).
- e - this optional parameter is a value between 1 and 16 which is used to set the foreground color for paintbrush C in the graphic screen. The default value is 6 (green).

Because the VIDEO BASIC-64 commands set the foreground color automatically when plotting, these optional parameters (c,d & e) are needed only when VIDEO BASIC-64 plotting routines ARE NOT used to put data onto a graphics screen, such as when loading bit-map only screens or copying areas without color memory copying or doing BLOCK inverse commands.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

TIC a,b,c - TIC mark the screen

This command places tic marks of the specified color along the edges of the display area. These tic marks provide a scale that is useful in measuring distances on the screen.

- a - this value is the tic mark interval in the x-direction (horizontal)
- b - this value is the tic mark interval in the y-direction (vertical)
- c - this value selects the color of the tic marks (see TABLE 1).

For example TIC 10,15,7 will plot blue(7) tic marks at x=0, x=10, x=20, etc. and at y=0, y=15, y=30, etc.

DOT x,y,c - plot a DOT (point)

This command plots a single point of the chosen color at the position given by the x value and the y value.

- x - This value is the x-coordinate (0-319) of the point.
 - y - This value is the y-coordinate (0-199) of the point.
 - c - This is the value which specifies the color of the point (TABLE 1).
-

DRAW x1,y1,x2,y2,c - DRAW a line

This command draws a straight line of the specified color starting at point x1,y1 and going to point x2,y2.

- x1 - This is the x-coordinate of the first point on the line.
 - y1 - This is the y-coordinate of the first point of the line.
 - x2 - This is the x-coordinate of the end point of the line.
 - y2 - This is the y-coordinate of the end point of the line.
 - c - This is the color of the line (see TABLE 1).
-

VIDEO BASIC-64 DEVELOPMENT PACKAGE

BOX x1,y1,x2,y2,c - draw a BOX

This command draws a box (rectangle) of the specified color with its two diagonal corners at the coordinates x1,y1 and x2,y2

x1 - This is the x-coordinate of the first corner of the box.

y1 - This is the y-coordinate of the first corner of the box.

x2 - This is the x-coordinate of the diagonally opposite corner of the box.

y2 - This is the y-coordinate of the diagonally opposite corner of the box.

c - This is the color of the box (see TABLE 1).

CIRCLE x,y,r,c,xf,yf - draw a CIRCLE

This command draws a circle of the specified color with the center at the point specified by x and y. The radius of the circle is given by the r value. The x-factor xf or y-factor yf is optional and is a "squash" factor. It is a number between 0 and 1000. To adjust the width of a circle to 80% of normal, the x-factor is set to 800. To adjust the height of a circle to 70% of normal, the y-factor is set to 700. A factor of 0 is ignored. E.g. To squeeze a circle down in height to 70%, simply add the parameters ,0,700 to a CIRCLE command.

x - The x-coordinate of the center of the circle.

y - The y-coordinate of the center of the circle.

r - The radius of the circle.

c - This is the color of the CIRCLE (see TABLE 1).

xf - The x-factor to multiply times X-dimension to flatten circle (1000 = 100%)

yf - The y-factor to multiply times Y-dimension to flatten circle (1000=100%)

VIDEO BASIC-64 DEVELOPMENT PACKAGE

CHAR g,x,y,o,"string" - display CHARacters.

This command displays the string of characters given in the command. The string may be a value, character string or a string variable. The characters are 8 points high and wide in high-resolution mode and 16 points high and wide in multicolor mode. The characters are displayed in the color specified.

- g - This value selects one of the C-64's character sets:
- 1 - upper case letters, numbers and graphics
 - 2 - reversed upper case letters, numbers and graphics.
 - 3 - upper and lower case letters and numbers.
 - 4 - reversed upper and lower case letters and numbers.

To double the character height on the screen, add 4 to the character set number.

To double the character width on the screen, add 8 to the character set number.

Therefore the table of values and parameters is:

	UPPER CASE			
	.	UPPER CASE	REVERSE	
	.	.	lower case	
	.	.	.	lower case
	.	.	.	reverse

Norm. height,norm. width	1	2	3	4
Doub. height,norm. width	5	6	7	8
Norm. height,doub. width	9	10	11	12
Doub. height,doub. width	13	14	15	16

This parameter may also be used to select a character set other than the standard one in ROM. The new character set must start on a \$n000 boundary (n=1-8). Simply add the value n*16 to the parameter value g.

For example, if the new character set begins at \$5000, you would add 5*16 to the parameter value g. To select the first character set in that area, use the command **CHAR 5*16,.....** You may add +4 and/or +8 to the 5*16 to get double height or width.

NOTE- You will have to manage the BASIC pointers to keep BASIC out of the memory area occupied by these new character masters. (See Memory Management in the PROGRAMMING NOTES Section).

VIDEO BASIC-64 DEVELOPMENT PACKAGE

- x - This is the x-coordinate of the upper left point of the first letter of the string.
- y - This is the y-coordinate of the upper left point of the first letter of the string.
- c - This is the color of the characters (see TABLE 1).

"string" - This is the character string to be displayed. This command functions much the same as the BASIC PRINT command. If a number or numeric variable is given for this parameter, it is converted and displayed. Strings may be concatenated (such as A\$ + "DATA").

Note In order to properly create the string for character sets 3 and 4, you should switch the C-64 screen to upper/lower case mode by pressing the SHIFT and C= keys at the same time. Then, what you see in the string is what will be displayed.

BLOCK x1,y1,x2,y2,c - BLOCK of color

This command fills a rectangular area of the screen in the specified color. The block is defined by two diagonal corners, just as in the BOX command. The block is filled using the 8x8 character cells (see Figure 1), so the edges are automatically adjusted to be multiples of 8 in the x and y directions.

- x1 - This is the x-coordinate of the first corner of the block.
- y1 - This is the y-coordinate of the first corner of the block.
- x2 - This is the x-coordinate of the diagonally opposite corner of the block.
- y2 - This is the y-coordinate of the diagonally opposite corner of the block.
- c - This is the color of the block (see TABLE 1).

In high resolution mode, the block is written in the background color, permitting plotting on top of it. In multicolor mode, the block is written by one of the three "paintbrushes", selected by the range of the color number selected (1-16, 101-116, or 201-216).

The BLOCK command also has an expanded form which performs several additional functions:

VIDEO BASIC-64 DEVELOPMENT PACKAGE

BLOCK x1,y1,x2,y2,c[,m,p1,p2,p3,p4,p5,p6,p7,p8]

x1,y1,x2,y2,c are the same as above

m - mode

0 = normal mode

1 = reverse color memory. In high resolution mode this switches the background and foreground colors of the screen image. In multicolor mode, this switches the colors of paintbrushes A and B of the screen image. The background and paintbrush C are not affected.

2 = write pattern into block in color c. The pattern is specified by p1-p8 (p1 is top row).

p1-p8 - bit patterns (p1 is on top)

If you wanted to fill an area with a brick type pattern for example, you have to first draw it out, compute the values of each row and then use the values in the command.

Each column in the pattern has a place value, starting with 128 at the left and then 64, 32, 16, 8, 4, 2 and 1 as you move towards the right. Simply add the place values corresponding to the points turned on for each row:

	<u>PATTERN</u>	<u>INDIVIDUAL PIXEL VALUE</u>	<u>ROW VALUE</u>
row 1	xxxxxxxx	128+64+32+16+8+4+2+1	= 255
row 2	x.....	128	= 128
row 3	x.....	128	= 128
row 4	x.....	128	= 128
row 5	xxxxxxxx	128+64+32+16+8+4+2+1	= 255
row 6x...	8	= 8
row 7x...	8	= 8
row 8x...	8	= 8

These value are then used as p1-p8 of the BLOCK command.

In HIRES mode an example is:

BLOCK 50,50,100,100,2,2,255,128,128,128,255,8,8,8

In multicolor filling, each pattern should be two bits wide, because 2 bits are used to specify each multicolor pixel on the screen. Further, each of the four two-bit pairs controls the color though paintbrush selection.

If the pair is .x - then paintbrush A color is used;

If the pair is x. - then paintbrush B color is used;

If the pair is xx - then paintbrush C color is used.

MODE a - set MODE of display

This command controls the plotting mode for all commands. There are three modes:

- 0 - (MODE 0) This is the normal mode of operation. It is automatically set by the HIRES or MULTI commands. In this mode, the selected point(s) are turned on.
- 1 - (MODE 1) This is the erase mode. After this command is executed, all plotting turns the specified point(s) off, erasing them from the display. The point is made the same color as the background color.
- 2 - (MODE 2) This is the reversing mode. When this has been set, if a point is initially off, it is turned on. If initially on, its turned off. This mode may be used to erase part of a display by displaying it a second time. It is also useful to draw a pattern which will be visible, even though it crosses other points turned on or off.

Note - These modes stay in effect until another mode command is issued or a HIRES or MULTI command. Therefore, to reset plotting to normal, give a MODE 0 command.

FILL x,y,c,b - FILL an area

This command fills an area with the specified color. The filling starts at the x and y coordinates given and proceeds until the entire area contains the color.

- x - The x coordinate of a point inside the area (really doesn't matter where inside).
- y - The y-coordinate of the point inside the area.
- c - The color to be used to fill the area.
- b - This parameter is NOT needed in HIRES filling (or may be 0). In MULTICOLOR filling, this is the hundreds digit (0,1 or 2) of the color used to draw the edge of the area. In other words, this parameter tells the program which of the three "paintbrushes" was used to draw the edge of the area.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

The STOP key, the SPACE bar or the left arrow (<) key aborts any FILL operation and returns to BASIC without an error indication.

The FILL operation always leaves the graphics MODE set at 0 (write mode).

NOTE - In multicolor mode, you cannot fill the area with the same "paintbrush" as was used to draw the edge.

The FILL command has an expanded format to permit pattern filling:

FILL x,y,c,b[m,p1,p2,p3,p4,p5,p6,p7,p8]

x,y,c[b] are same as above

m - mode

0 = normal mode (solid color filling)

1 = pattern filling

p1-p8 are the bit patterns (p1 is on top)

If you wanted to fill an area with a brick type pattern for example, you have to first draw it out, compute the values of each row and then use the values in the command.

Each column in the pattern has a place value, starting with 128 at the left and then 64, 32, 16, 8, 4, 2 and 1 as you move towards the right. Simply add the place values corresponding to the points turned on for each row:

	<u>PATTERN</u>	<u>INDIVIDUAL PIXEL VALUE</u>	<u>ROW VALUE</u>
row 1	xxxxxxxx	128+64+32+16+8+4+2+1	= 255
row 2	x.....	128	= 128
row 3	x.....	128	= 128
row 4	x.....	128	= 128
row 5	xxxxxxxx	128+64+32+16+8+4+2+1	= 255
row 6x...	8	= 8
row 7x...	8	= 8
row 8x...	8	= 8

These value are then used as p1-p8 of the FILL command.

In HIRES mode an example is:

FILL x,y,2,0,1,255,128,128,128,255,8,8,8 (note b-value must be 0 in HIRES mode).

VIDEO BASIC-64 DEVELOPMENT PACKAGE

In multicolor filling, each pattern should be two bits wide, because 2 bits are used to specify each multicolor pixel on the screen. Further, each of the four two-bit pairs controls the color through paintbrush selection.

If the pair is .x - then paintbrush A color is used;
If the pair is x. - then paintbrush B color is used;
If the pair is xx - then paintbrush C color is used.

NOTE - The memory area \$A000-\$BFFF is used as a work area during the execution of the pattern filling option and therefor will not be preserved.

PIXEL(x,y) - return the PIXEL value

This is not a command. It is a function, like the BASIC functions INT, SIN, PEEK, etc. The PIXEL function returns a value depending on the color at the point designated by the x and y coordinates.

The value is 0 if no color is present (background color).

The value is 1 if the point is on in high-resolution mode.

The value is 1-3 if the point is on in multicolor mode-
1 if the point was written with a color 1-16
2 if the point was written with a color 101-116
3 if the point was written with a color 202-216
.

Examples of uses are:

PRINT PIXEL (x,y)

IF PIXEL (x,y) = 2 THEN

VIDEO BASIC-64 DEVELOPMENT PACKAGE

7. SCREEN CONTROL COMMANDS

DUMP "filename" [,dev,me] - save graphic display to memory device

This command saves the graphic screen, color memory, sprites, etc. to the specified device. disk.

"filename" - this is the name that the display is to be given on tape or disk. The name may be either a literal enclosed in quotes or a string variable (strings may be concatenated such as "0:"+PN\$).

[,dev] - this is the device number. The default is 1 (for tape)

me - memory control option

0 = (or if parameter is omitted) saves everything including sprite patterns

1 = saves only the 8K bit-mapped graphics area (\$E000-\$FFFF).

2 = saves the bit-mapped area plus the video matrix area (\$C000-\$C3FF). This includes hires background and foreground colors or paintbrushes A and B only, in multicolor mode.

3 = saves the bit-mapped area plus video matrix plus the chip registers (\$C400-\$C42F). This will save the border color (and background color in multicolor mode).

4 = saves the color memory save area (\$C430-\$C82F) in addition. This option saves the color information for paintbrush C.

GREAD "filename" [,dev] - restore the graphic display from memory device

This command restores (reads) the entire graphic display from tape or disk.

"filename" - this is the name of the display to be read from tape or disk. The name may be either a literal enclosed in quotes or a string variable (strings may be concatenated such as "0:"+PN\$).

[,dev] - this is the device number for either the tape or disk. If you wish to read the graphic display from tape, it is not necessary to specify ",1" since this is the default. To read the graphic display from disk, specify ",8" which is the device number of the disk drive.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

NORM - switch the screen to NORMAl (BASIC text)

This command places the COMMODORE-64 into the NORMAL text mode. You can switch between graphics and text in programs mode by alternately using the NORM and GRAPH commands.

GRAPH - switch the screen to GRAPHics display

This command places the COMMODORE-64 into graphic display mode. You can switch between graphic and text mode in programs by alternately using the NORM and GRAPH commands. The graph command is implied by most other graph commands.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

V. SPRITE GRAPHICS

A. INTRODUCTION TO SPRITES

In addition to all of the graphic capabilities described above, VIDEO BASIC-64 can help you manage sprites easily.

The sprite is a hook, upon which you "hang" a little picture. The picture is called a sprite pattern and it may be either hires (all one color) or multicolor. Once you have put the picture on the hook, you can move the hook (and the picture hanging on it) easily around the screen. Animation of the picture is easy because you can change the picture on the hook.

Here's how to do it.

The first step is drawing the picture you want to hang onto the sprite. VIDEO BASIC-64 provides four different ways of coding the picture in your BASIC program (more about this later).

The second step is copying the picture into a special area of memory (we call them sprite slots). VIDEO BASIC-64 provides room for 15 different pictures at any one time (although new ones may be copied in over old ones).

The third step is turning on the sprite, attaching the picture and defining size and colors.

The fourth step is moving the sprite around the screen under program control. During this time while the picture is being shown, you may change the picture, change the colors, size, etc. of the picture attached to the sprite.

The last step (optional) is turning the sprite off.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

B. SPRITE PICTURES

A hires sprite picture is 24 dots wide and 21 dots high. Usually you will draw the pattern on a piece of graph paper. In single color mode, each of those dots may be either on or off (1 or 0). In a multicolor sprite picture, it is 12 points wide, each point may be 0,1,2 or 3, which selects whether nothing shows at that point (0) or which of the three colors (1,2 or 3) is to display at that point. (note that these points and colors are completely separate from the colors and points in the other commands).

In the sprite section of the COMMODORE 64 manual they describe an example and go through the difficult work of coding the picture into BASIC DATA statements. If you have sprite patterns already coded that way, we have included the SDATA command which is just like the DATA command for a sprite picture. However, there's a much easier way-

For a hires (single color) sprite picture, code the pattern using the BIT command.

BIT"00000000000000000000000000000000"

A hires sprite picture can be coded (and seen) in BASIC using 21 lines of the BIT command. A " must follow the word BIT, and there must be 24 numbers (0 or 1) before the closing ". There must be 20 more lines of BIT commands following the first. Remarks are allowed after the second " on any line. When these lines are listed on the screen or printer, you can see the pattern quite easily.

For a multicolor sprite picture, code the pattern using the **COLORS** command.

COLORS "012301230123"

The **COLORS** command is very similar to the **BIT** command, except that there are only 12 digits within the " " and the numbers may be 0, 1, 2, or 3.

The color of a hires sprite and the color 1 of a multicolor sprite may differ for each sprite. Colors 2 and 3 in multicolor sprites are shared by all sprites, so they must be chosen carefully. Note that the patterns don't contain the actual color, they only contain a color selection digit. The actual colors are assigned when the sprite is turned on.

The fourth way a sprite picture may be coded is in hexadecimal, using the **HEX** command. The **HEX** statement may be used in place of the **BIT** or **SDATA** statements.

HEX"0F"

VIDEO BASIC-64 DEVELOPMENT PACKAGE

This command contains 21 bytes of data in hexadecimal, or 42 nibbles of digits. The digits allowed are 0-9 and A-F. There must be three lines of HEX statements to completely code a sprite. While HEX is more complicated than BIT or COLORS, it takes up less space in programs. You must convert your BIT values to hexadecimal values and place them into the HEX statements.

Any of these four statements may be used to define a sprite picture and they may be mixed within a program. The only requirement is that only one method may be used per sprite picture.

Here's a comparison of three methods using the Commodore balloon (pg 70 of the COMMODORE 64 User's Guide)

Note-the line numbers used don't matter, as long as the lines are together.

These commands (BIT, HEX, COLORS & SDATA) are treated as REM lines by BASIC and bypassed.

```
901 SDATA 0,127,0,1,255,192,3,255,224,3,231,224
902 SDATA 7,217,240,7,223,240,7,217,240,3,231,224
903 SDATA 3,255,224,3,255,224,2,255,160,1,127,64
904 SDATA 1,62,64,0,156,128,0,156,128,0,73,0,0,73,0
905 SDATA 0,62,0,0,62,0,0,62,0,0,28,0
```

```
901 HEX"007F0001FFC003FFE003E3E007D9F007DFF007D9F0"
902 HEX"03E7E003FFE003FFE002FFA0017F40013E40009C80"
903 HEX"009C80004900004900003E00003E00003E00001C00"
```

```
901 BIT"000000000111111100000000"
902 BIT"00000001111111111000000"
903 BIT"000000011111111111100000"
904 BIT"000000111110001111100000"
905 BIT"000001111101100111110000"
906 BIT"000001111101111111110000"
907 BIT"000001111101100111110000"
908 BIT"000000111110011111100000"
909 BIT"000000111111111111100000"
910 BIT"000000111111111111100000"
911 BIT"0000001011111111110100000"
912 BIT"000000010111111101000000"
913 BIT"0000000100111111001000000"
914 BIT"000000001001110010000000"
915 BIT"000000001001110010000000"
916 BIT"000000000100100100000000"
917 BIT"000000000100100100000000"
918 BIT"000000000011111000000000"
919 BIT"000000000011111000000000"
920 BIT"000000000011111000000000"
921 BIT"000000000001110000000000"
```

VIDEO BASIC-64 DEVELOPMENT PACKAGE

C. SPRITE COMMANDS

After the sprite picture(s) has been coded into your program, you may use the following command to control the sprites.

COPY a,l1l1l1 - COPY sprite image

a - sprite slot number 1-15.

l1l1l1 - line number of the first BIT, COLORS, HEX or SDATA command to be copied. If this line number is missing or there is something wrong with the picture data, an **UNDEFINED LINE NUMBER** error will be returned, pointing to the COPY statement. The copy statements may be anywhere within the program, but normally they are executed before the command which turns the sprite on. The line number in the COPY command must be 32767 or less, but not less than 1.

SPRITE n,s,m,p,x,y,c1,c2,c3 - turn SPRITE on

This command attaches a picture to a specified sprite, sets up several attributes for the sprite picture and finally turns the sprite on.

The variables in the command are:

n - The sprite number being started (1-8). Note that sprite 1 has the highest priority and will show in front of any sprite of lower priority number.

s - slot number where the sprite picture has been COPY'd (the number in the COPY command).

m - multicolor control:

0 = high-resolution (one-color) sprite

1 = multicolor sprite

p - priority control for the sprite vs background:

0 = sprite in front of background.

1 = sprite behind background.

x - x-expand:

0 = sprite picture normal size (24 pixels wide) in x-direction.

1 = sprite picture double width in x-direction (48 pixels).

VIDEO BASIC-64 DEVELOPMENT PACKAGE

- y - y-expand
0 = sprite picture normal size (21 pixels high)
in y-direction.
1 = sprite picture double height (42 pixels).
- c1 - sprite color (hires sprite mode) or color 1 in
multicolor sprite mode. (See TABLE 1)
- c2 - In multicolor sprite mode only, the color 2 color
number. Note that this is shared by all multicolor
sprites. This value is not needed for hires sprites.
(See TABLE 1)
- c3 - color 3 of multicolor sprites. The same notes apply
as with c2. (See TABLE 1)
-

OFF n - turn sprite OFF

This command turns the sprite off.

n - the sprite number

PLACE n,ex,sy - Place a sprite

This command positions a sprite on the screen. The sprite controls in the COMMODORE-64 use different x and y coordinates from normal graphics. This is to allow sprites to be moved out beyond the normal screen area. Therefore you have to compute the x and y positions for sprites as follows:

sprite x = graphic x + 24

sprite y = graphic y + 6

n - sprite number

sx - sprite x position of upper left corner of sprite

sy - sprite y position of upper left corner of sprite

ROTATE d,m,s - Rotate a sprite

This command turns a sprite pattern 90 degrees within its slot. In order to rotate a sprite pattern, it must have the same number of elements in both directions. For a hires pattern, only the left 21 of the 24 bits on each row are used (the other 3 won't be rotated). For a multicolor pattern, all 12 color elements used per row of the

VIDEO BASIC-64 DEVELOPMENT PACKAGE

pattern, but only the first 12 of the 21 rows are be used.

d = direction:

0 = clockwise

non-zero = counter clockwise

m = multicolor:

0 = hires sprite is being rotated

1 = multicolor sprite is being rotated

s = slot number (1-15)

VIDEO BASIC-64 DEVELOPMENT PACKAGE

D. SPRITE EXAMPLE

The easiest way to understand these sprite commands is by trying the following example.

After VIDEO BASIC-64 is initialized, enter the following BASIC program:

Don't key the comments in []

```
10 HIRES 1,7           [setup black screen, blue border]
20 COPY 5,901          [copy the sprite picture starting
                        in line 901 to sprite slot 5]
30 SPRITE 3,5,0,0,1,1,5 [turn on sprite 3, with picture
                        from slot 5, single color,
                        normal priority, expand x & y
                        sizes, in purple (color 5)]
40 FOR Y=50 TO 230 STEP 40 [do display with various
                        y-positions up the screen]
50 FOR X=0 TO 345        [move across screen]
60 PLACE 3,X,Y          [position the sprite]
70 NEXTX
80 NEXTY
```

Now copy any of the three balloon patterns from page 17 into lines starting with 901. Run this program and you should see the balloon move across the bottom of the screen, then make passes at increasing heights.

Now experiment with changing the pattern, x & y expand, and color.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

VI. TURTLE GRAPHICS

VIDEO BASIC-64 includes TURTLE graphics. The turtle is a colorful creature that can move about on the screen. The turtle carries a pen and is capable of leaving a trail as he moves about. TURTLE commands instruct the turtle to turn and move.

The turtle normally starts in the center of the screen, pointing towards the top of the screen (north). Turns are specified in degrees (0-360). A positive number turns the turtle in the clockwise direction. A negative number turns the turtle in the counter-clockwise direction. There is also a command which turns the turtle to a specific direction.

The directions and degrees are as follows:

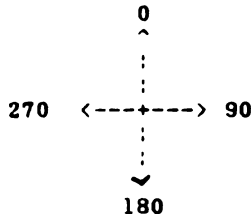


FIGURE 2- TURTLE DIRECTIONS

TURTLE COMMANDS:

TURTLE c[,x,y] - initialize turtle

The turtle is placed in the middle of the screen (x=160, y=100) unless an x and y position is specified.

c = color of the turtle and default line color (1-16)

x = starting x position of the turtle (0-319)

y = starting y position of the turtle (0-199)

NOTE - sprite number 8 and slots 12-15 are used by the turtle! Therefore you may not define and use sprite 8 nor use slots 12-15 while you are using any of the TURTLE commands.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

TCOLOR c - set the turtle line color

This command changes the color of the line with which the turtle draws. The color of the turtle does not change.

c = line color (1-16)

TUP - lift the turtle pen

This command lifts the turtle's pen from the screen. When it moves with the pen up, the turtle moves without leaving a trail

TDOWN - put the turtle pen down

This command puts the turtle's pen down onto the screen. When it moves with the pen down, the turtle leaves a line in its trail.

TURN d - Turn the turtle

This command causes the turtle to turn, clockwise if d is positive or counter-clockwise if d is negative. The turtle on the screen may not appear to move for small changes in direction. However, the lines will be drawn at their correct angle.

d = number of degrees to turn

TURNT0 d - point the turtle in a direction

This command turns the turtle to a specific direction. The direction corresponds to the degrees on a compass with the top of the screen corresponding to 0 degrees (north), the bottom of the screen corresponding to 180 degrees (south), etc.

d = direction (in degrees)

VIDEO BASIC-64 DEVELOPMENT PACKAGE

MOVE a - move turtle

This command causes the turtle to move a given number of points in the direction it is pointing.

a - number of points

BYE - make turtle disappear

This command makes the turtle disappear from the screen (although he can still draw). This command permits the turtle to draw more quickly since drawing can be done faster if the turtle is not visible.

TPOS(v)

This function returns the turtle's position or direction.

The letter inside the parenthesis () determines whether the value returned is the x-coordinate, y-coordinate or the angle of the turtle.

v = A returns the angle the turtle is pointing

v = X returns the x position

v = Y returns the y position

The value may be used to set a variable (W1=TPOS(X)) or in a test command (IF TPOS(Y)>100 THEN GOTO 150).

VIDEO BASIC-64 DEVELOPMENT PACKAGE

VII. GAME FEATURES

VIDEO BASIC-64 has several features that are especially suitable for use in games. The features are broken into input functions, collision detection and timer functions.

A. INPUT FUNCTIONS

JOY(p) - joystick function

The JOY function reads either joystick port p.

p = 1, joystick port 1
2, joystick port 2

The value returned is the sum of the following:

1 = North
2 = South
4 = West
8 = East
16 = Fire button

For example if the joystick were being held in the southeast direction and the fire button was being pressed, the value would be 2 (South) + 8 (East) + 16 (Fire button) = 26.

PADDLE(p,d) - game paddle function

The PADDLE function reads the position of the game paddle. There are two paddles connected to each of the two game ports. The ports are designated 1 and 2. The paddles connected to each port are designated as X and Y.

p = 1, port 1
2, port 2

d = X, paddle X
Y, paddle Y

NOTE - the fire buttons on the paddles are read by the JOY function. The fire button on the X paddle returns a 4 value and the Y paddle returns a value of 8 (12 if both).

VIDEO BASIC-64 DEVELOPMENT PACKAGE

PEN(c) - lightpen function

The position of the lightpen may be read by the PEN function. Either the x-coordinate or y-coordinate may be read.

c = X, to read the X-coordinate
= Y, to read the Y-coordinate

NOTE - If your lightpen has a "trigger", then the JOY function can be used to determine if the lightpen has trigger has been activated. Joystick function JOY(1) will return a value of 16 if the trigger has been pressed. Some lightpens have a tip switch hooked to the WEST button line so the test may have to be IF JOY(1)=4 THEN....

We have also found that the EDUMATE lightpen is wired so that the keyboard does not function properly (BCMZ and other keys are disabled).

To facilitate calibration of lightpens, two bytes of memory contain the initial offset values and may be adjusted though POKEs.

30595 is the x-offset and initially contains 42.

30596 is the y-offset and initially contains 11.

To align pens, simply display a point on the screen (say at x=160 and y=100) and instruct the operator to put the pen at the point. Then read the lightpen values through the PEN commands and increase the x-value in the 30595 location if the value read is too low (less than 160), or decrease the value in that location if the value is too high. Do the same with the y-value, only increase the value in 30596 if the value read is too high (above 100).

VIDEO BASIC-64 DEVELOPMENT PACKAGE

B. TIMERS

VIDEO BASIC-64 makes it very convenient to keep track of the duration of different events. Ten timers (called counters) are available. They all count down (like an oven timer).

Counters 0 through 4 count down in jiffies (1/60th of a second). Counters 5 through 9 count down in seconds.

A counter is set using the SCTR command and read using the CTR function.

SCTR c,v - set counter

c = counter number(0-9)
v = value to set counter (1-65000)

CTR(c) - read counter

c = counter to be read(0-9)

NOTE - both the HIRIS and MULTI commands reset the counters. You should be aware of this if you are timing some events and must also use either of those commands.

C. SPRITE COLLISION FUNCTION

Collisions between sprites and between a sprite and a background pattern can be detected with the SCOLL and BCOLL functions. The function is true (-1) if the specified collision(s) have occurred, and false (0) if not.

SCOLL(s1,s2[,s3,..s8]) - sprite/sprite collision function

s1, s2, etc. are the sprite numbers to be tested. The test is true only if ALL sprites specified are in collision with each other.

For example:

SCOLL(2,3) is true only if sprites 2 and 3 collide.

SCOLL(1,4,5) is true only if sprites 1 and 4 and 5 are in collision.

NOTE - The test is true as long as all sprites listed in the command are in collision. If additional sprites not specified in the command are also in collision, the test will still be true.

BCOLL(s1[,s2,..s8]) - sprite/background collision function

s1, s2, etc. are the sprites to be tested for collision with anything in the background. Unlike SCOLL, this function is true if ANY of the sprites are in collision with the background.

NOTE - In multicolor mode, only objects in the lxx or 2xx colors will be detected by this test. Objects written with 'regular colors (1-16)' will not be seen by this test.

Examples:

BCOLL(2) is true if sprite 2 hits a lxx or 2xx color object in the background.

BCOLL(1,3) is true if EITHER sprite 1 or 3 hits a lxx or 2xx object in the background.

NOTE - The COMMODORE 64 collision registers are set (latched) upon a collision and stay set until the register is "read" during a SCOLL or BCOLL function. If the collision still exists after the test, the registers are set again. If your program takes a while to test for a collision, the sprites may have been moved apart; but the register is still set until the collision test is made. To avoid any problems, test for collisions very soon after each sprite movement occurs.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

VIII. SOUND FEATURES

There are three sound generators in the COMMODORE 64. Each is designated by the number - 1, 2 or 3. To make it easy to use, VIDEO BASIC-64 sets up generator 1 to be a standard square-wave, 2 is set to a sawtooth waveform (reedy sound) and 3 is set up as a noise burst sound. The tone characteristics of each generator may be changed by the GEN and VOL commands.

The pitch of a tone or the pitch of the noise generator is determined by a value between 0 and 95. The values correspond to the steps in the chromatic music scale. 0 corresponds to the lowest C on a piano keyboard, 1=C# and so on. The value 48 gives a pitch of middle C. The complete list of values and notes is given in TABLE 2 and also in APPENDIX B.

There are two ways to make sound - by using the SOUND command or by starting a TUNE pattern.

The SOUND command is the easiest to use:

SOUND a,p,d - start a SOUND

The SOUND command lets you specify the generator (1-3), the pitch (1-127, see TABLE 2 or APPENDIX B for note correspondence) and the duration (0-255 60ths of a second).

a = generator - 1, 2, or 3.

p = pitch (0 = low, 127 = high)

d = duration (in 60ths of a second)

VIDEO BASIC-64 DEVELOPMENT PACKAGE

All numbers are in decimal

Octave	Note											
	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
1	-	-	-	0	1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27	28	29	30	31	32
4	33	34	35	36	37	38	39	40	41	42	43	44
5	45	46	47	[48]	49	50	51	52	53	54	55	56
6	57	58	59	60	61	62	63	64	65	66	67	68
7	69	70	71	72	73	74	75	76	77	78	79	80
8	81	82	83	84	85	86	87	88	89	90	91	92
9	93	94	95									

TABLE 2 -TONE/NOTE Chart

If you want to change the characteristics of any of the tone generators, you can use the GEN command as outlined below:

GEN a,b,c,d,e,f,g,h,i - set up generator

a = generator - 1, 2, or 3

b = waveform:

1 = triangle

2 = sawtooth

4 = pulse

8 = noise

(You can add up several values to make new sounds, but the results are unpredictable)

c = attack speed

0 = instant

15 = slowest attack on each note

d = decay speed (how fast the initial sound drops to the sustain level)

0 = instant

15 = very slow decay

e = sustain level (the level of the sound after the initial attack and decay)

VIDEO BASIC-64 DEVELOPMENT PACKAGE

0 = silent
15 = full volume
f = release speed (how fast the sound dies away after it is stopped)
0 = instant
15 = very slow decay
g = duty cycle, for pulse waveforms
0-15 A 50% duty cycle (7) results in a square wave
h = synchronization control
The generators may be locked together (synchronized). This parameter determines if this generator is or is not locked.
0 = no synchronization
1 = synchronize
i = ring modulator control.
The sounds of two generators may be modulated by one another if this control is on.
0 = no modulation
1 = ring modulation on

VOL a,b,c,d,e - volume-filter control

The VOL command sets the controls which apply to all generators

a = overall sound level
0 = no sound
15 = loudest sound
b = filter mode
1 = low pass filter mode
2 = band pass filter mode
4 = high pass filter mode
8 = turns off the sound from generator 3 (useful with synch and ring mod).
These values may be added together for multiple modes.
c = filter control
1 = send generator 1 output thru filter
2 = send generator 2 output thru filter
4 = send generator 3 output thru filter
8 = send external sound input thru filter

VIDEO BASIC-64 DEVELOPMENT PACKAGE

d = filter frequency - control the center or cutoff
frequency of the filter
0 = low
15 = high

e = filter resonance (controls how sharp the filter is)
0 = very flat cutoff
15 = very sharp cutoff

The other method of making sound is by starting a TUNE pattern. The pitch of each tone generator can be programmed, started and timed separately, and all of this runs simultaneously with the rest of your VIDEO BASIC-64 program!

A pattern is a set of instructions for controlling the generator. A pattern is composed of several segments (or parts). There are two numbers needed for each segment of a tune - the pitch or increment and the duration.

Because it is compact, all of the information to control the TUNE is written in hexadecimal notation in a TDATA pattern. If you are not familiar with hexadecimal notation, see APPENDIX F.

A TDATA pattern starts with the opening pitch value in hexadecimal (such as 38). The next hexadecimal number specifies how many jiffies (1/60th of a second) to hold the pitch. When the time has expired, VIDEO BASIC-64 automatically looks at the next value and time segment within that literal. From this point onward, the value is the amount that the pitch should be increased (or decreased) each jiffy. The time determines how long the incrementing (or decrementing) is to continue. Again, when the time is up, VIDEO BASIC-64 looks at the next value and time segment. The tone is shut off if the time in the literal is 00.

For example:

The TDATA pattern "38090108FF0800100000" is processed as such --

3809 set the tone generator to pitch 38 hexadecimal
(56 decimal)
0108 increment the pitch by 1 each jiffy for 8 jiffies
FF08 decrement the pitch by 1 each jiffy for 8 jiffies
(FF hexadecimal = -1 decimal)
0010 no change to pitch (increment 00) for 10 hex
(16 decimal) jiffies
0000 turn off the tone since the time value is 00

Additional features are included to expand the flexibility. An increment value of hex 80 turns the generator off for the

VIDEO BASIC-64 DEVELOPMENT PACKAGE

specified time. Another hex 80 turns it back on. If the quote (") at the end of the literal is found before the 00 time entry, VIDEO BASIC-64 goes back to the beginning of the literal and starts over. This allows the sound pattern to repeat as long as you wish. Since the first entry in the literal sets the starting pitch, it can be bypassed if the first hex digit is 8-F (greater than 127 decimal).

For example, a high-low siren is:

```
10 SET 1,99
20 TUNE 1,255
30 END
99 TDATA"C010FB01001005010010"
```

C010 sets the pitch to 40 hex (64 decimal) and bypass on repeat

FB01 lower the pitch by 5 for 1 jiffy

0010 hold the pitch for 16 jiffies (10 hex is 16 decimal)

0501 raise the pitch by 5 for 1 jiffy

0010 hold the pitch for 16 jiffies

" quote says to repeat, bypassing C010. Therefore the next step is

FB01 to lower the pitch by 5, and on and on.

If you create a tune which does not shut itself off, then you can press the Function 1 key (F1) which turns off all of the generators.

TABLE 3 presents the same TONE/NOTE Chart of TABLE 2, but has the values presented in hexadecimal for use in the following TDATA commands.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

All values are hexadecimal numbers

Octave	Note											
	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
1	-	-	-	00	01	02	03	04	05	06	07	08
2	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
3	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20
4	21	22	23	24	25	26	27	28	29	2A	2B	2C
5	2D	2E	2F	[30]	31	32	33	34	35	36	37	38
6	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44
7	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
8	51	52	53	54	55	56	57	58	59	5A	5B	5C
9	5D	5E	5F									

TABLE 3 - TONE/NOTE Chart

The TDATA pattern is used with any of the three tone generators. It is attached to the tone generator with the SET command and started with the TUNE command.

TDATA"PTTIITTIITT....." - define TUNE pattern

This statement defines a pattern that can be used to control the generators.

The format of the data within the quotes " " is:

" P T : I D : I T : : : I T :
segment1 segment2 segment3segmentN

P = Hexadecimal value to set original pitch 00-7F.

To bypass this segment on repeat, use 80-FF.

T = Duration of segment. If T=00, signals the end of the tune

I = Increment (decrement) value each 1/60th of second.

If I=00, pitch is held constant

If I=80, pitch is turned off (second 80 turns it on again).

If quote is encountered, pointer is sent to the beginning of the literal (will bypass 1st segment if P is 80-FF).

VIDEO BASIC-64 DEVELOPMENT PACKAGE

SET a,11111 - set up **TUNE** pattern

a = generator - 1, 2 or 3

11111 = line number of the **TDATA** pattern to be used
by the **TUNE** command

TUNE a,d - start **TUNE** pattern

This command starts the designated tone generator. The **TUNE** command turns the generator on only for a specified time unless the time value is 255, in which case the generator stays on until the **TDATA** program or f1 key turns it off.

a = generator - 1, 2, or 3

d = duration (60ths of a second)

If you need to coordinate screen action with the completion of sound, there is a location for each tone generator which is set to 0 when the sound or tune is finished-

PEEK(30811) for generator 1

PEEK(30813) for generator 2

PEEK(30815) for generator 3

VIDEO BASIC-64 DEVELOPMENT PACKAGE

IX. OTHER FEATURES

A. REPEATING commands

Because there are often times when a pattern must be repeated (especially when using TURTLE graphics commands), the following repeat commands have been included.

They work as follows :

The sequence of commands to be repeated are enclosed between a **BRACKET[** and a **BRACKET]** commands (like a FOR - NEXT loop). The number of times the group of commands is to be repeated is indicated by a number and colon (:) following the **BRACKET[**.

BRACKET[3: :BRACKET] indicates that the commands will be repeated three times. There must be a colon(:) before the **BRACKET]**

Repeats may be nested to a depth of 5. An error message "[] STACK ERROR" is displayed if this limit is exceeded. The **BRACKET[** and the **BRACKET]** may be on different lines. Any BASIC or VIDEO BASIC-64 commands may be included.

**BRACKET[n :
BRACKET]** - repeat between the brackets.

n = number of times to repeat the commands

B. EXIT from REPEAT

It is sometimes necessary to leave a REPEAT loop before the repeat count has completed. The **:BRACKETE** command (bracket exit) is used for this purpose.

An example:

```
5
.
.
10 CHAR 1,50,75,2,"START"
20 BRACKET[ 15: TURN 30: IF TPOS(A) > 180 THEN : BRACKETE
30 BRACKET]
40 CHAR 1,50,50,2,"DONE"
```

VIDEO BASIC-64 DEVELOPMENT PACKAGE

The turtle command TURN 30 (meaning turn 30 degrees) is performed 15 times unless the turtle's angular position is greater than 180. In this case the REPEAT loop is exited and statement number 40 is performed. Note that the ending bracket :BRACKET] is on a separate line in order for the IF to work properly. Here's why--

```
20 BRACKET[ 15 : TURN 30 : IF TPOS(A) > 180 THEN : BRACKET :  
BRACKET]
```

This line 20 will not work correctly because the BRACKET] is never encountered when the turtle's angle is > 180 because the IF statement is not true. Therefore BASIC goes to the NEXT LINE, skipping the BRACKET]. By placing that statement on a separate line, it will work correctly.

:BRACKETE - exit from a REPEAT loop

C. HARDcopy of graphics screen

VIDEObASIC-64 can reproduce the graphics screen onto your Commodore, Epson MX-80, FX-80 or RX-80, Gemini, Prowriter 8510A or Okidata Microline printer. (The proper module for your printer is selected when the program is being loaded). The entire screen is sent to the printer. A small size screen requires 4 1/2 minutes to reproduce on the Commodore 1525 printer and about 1 1/2 minutes on an Epson FX-80 printer. The large size takes nearly 4 times as long.

HARD d,b,s,a - hardcopy of graphics screen

This command prints the graphics screen in either of two sizes onto your COMMODORE, EPSON, GEMINI or OKIDATA dot matrix printer in graphics mode.

d = device number (default is 4, the standard printer device)

b - big printout or small (big = 1, small = 0). Big is two points per screen pixel, small is only one per screen pixel, but take much less time to print. The default is small.

s - secondary address (if needed) to be sent with the printer OPEN command. This is usually needed to setup the printer interface. See APPENDIX A for more information. The default is secondary address 0.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

- a - ASCII translate. If your printer interface does not change the 8-bit codes in any way, then enter a 0. If it does change the codes, enter a 1 and VIDEO BASIC-64 will pre-translate the codes for you to compensate for the printer translation. See APPENDIX A for popular interfaces and their requirements.
-

D. RESET STACK - HIRES 99,0

This command resets the BASIC STACK to the normal point, clearing all GOSUB and FOR commands. For example the program statements:

```
10 GOSUB 25
```

```
25 ... IF I = H THEN ...
```

```
(go off somewhere, and then a GOTO
10, never RETURNing to the GOSUB
RETURN is bad practice, but using a
HIRES 99,0 to reset the stack in
line 10 (10 HIRES 99,0:GOSUB 25 )
will make it work.
```

BE CAREFUL, this command resets ALL GOSUBS, so it should never be used except at the main level of your program (such as at a Main menu).

In BASIC you should always exit a FOR by setting the loop variable to the top value and going through the corresponding NEXT.

HIRES 99,0 - reset the stack

no other parameters are required

VIDEO BASIC-64 DEVELOPMENT PACKAGE

E. RESET BASIC - HIRES 99,1

This command reset the Commodore 64 back to its state as if the computer was just turned on. It will "wipe out" any program that is in memory, so be sure that you want to reset the computer before issuing this command.

HIRES 99,1 - reset the computer

no other parameters are required

F. DISABLE STOP and STOP/RESTORE - HIRES 99,2

This command completely disables the STOP and STOP/RESTORE keys. This prevents anyone from interrupting a running program when VIDEO BASIC-64 is being used.

HIRES 99,2 - disable STOP and STOP/RESTORE keys

no other parameters are required

VIDEO BASIC-64 DEVELOPMENT PACKAGE

G. XFER - transfer memory

To permit fast and easy transfers of memory for screen saves, a memory transfer command is included. Except for the block of memory \$D000-DFFF which is specially controlled by the command, all ROMs are locked out during the transfer operation. This allows copying the graphic screen (which is located under the Kernal, to other areas of memory).

XFER Fh, Fl, Th, Tl, Nh, Nl [,D ,S]

h and l indicate the high and low order bytes of a two-byte address value (0-65535). To convert the address D from decimal, the high byte expression can be $\text{INT}(D/256)$. The corresponding low byte can be $D-(256*\text{INT}(D/256))$.

Fh, Fl - The high and low bytes of the starting address where the memory is to be transferred FROM.

Th, Tl - The high and low bytes of the starting address where the memory is to be transferred TO.

Nh, Nl - The high and low bytes of the number of bytes to be transferred.

D - \$D000 block of memory control:

0 - Reads the I/O ports and color memory in \$D000-DFFF.

1 - Reads the Character ROMs in \$D000-DFFF.

2 - Reads the RAM in \$D000-DFFF.

S - Swap control:

0 - normal operation (transfer from FROM area to TO area).

1 - swaps (exchanges) the FROM and TO areas.

NOTE - No checking is performed on the memory addresses or number of bytes transferred by this command. Be careful to make frequent saves of your program during testing before issuing this command, as erroneous transferring of memory can cause the system to crash!

VIDEO BASIC-64 DEVELOPMENT PACKAGE

H. REGION - copying graphic screen

To facilitate copying of portions of a graphic screen, a REGION command is included. This command copies a rectangular area (specified by x and y coordinates of two diagonal corners), from the graphic screen to a workarea (\$A000 and upward) or from the workarea to the graphic screen.

To copy a section of the screen, two REGION commands are used. The first transfers the screen area to the workarea. The second REGION command transfers the image to the new location in the screen.

If multiple screens are used (as with CADPAK-64), the graphic screen data can be swapped after the rectangle is in the workarea, so graphics can be copied from screen to screen.

Like the BLOCK command, REGION operates on 8x8 cells, so the pixel coordinates specified in the command are adjusted to 8x8 boundaries.

When the bit mapped area is copied from the workarea to the graphic screen, you can control whether the area copied from is MOVED, ORed, ANDed or EXCLUSIVE-ORed (reversed) over the area being copied to.

When the data is copied to the workarea, the corresponding color information from the video matrix (all color information in HIRRES mode or paintbrush A and B information in MULTICOLOR mode) is also copied.

When the information is copied back, you can control whether the color information is replaced in the graphic screen or ignored.

Location 40960 (\$A000) contains the number of cells in the x direction and location 40961 (\$A001) contains the number of cells in the y direction.

REGION 0,x1,y1,x2,y2 - copy graphic to workarea

x1,y1 - the x-y coordinates of one corner of the rectangle to be copied to the workarea.

x2,y2 - the x-y coordinates of the other corner of the rectangle to be copied.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

REGION l,x,y,m,c - copy graphic FROM workarea

x,y - the x-y coordinates of the UPPER LEFT screen coordinates where the graphic is to be copied.

m - graphic copy mode:

- 0 - MOVE graphic
- 1 - OR graphic
- 2 - EOR graphic
- 3 - AND graphic

c - color (video matrix) copy:

- 0 - no copy
- 1 - copy color

NOTE - If you copy more than 900 cells (of the 1000 on the screen) the copy will overwrite some of the video matrix area. Limit any region to less than a full screen. If you want to copy the entire screen, use an Xfer command.

X. PROGRAMMING NOTES

A. GENERAL INFORMATION

The VIDEO BASIC-64 interpreter is a 6502 machine language program which "wedges" itself into BASIC. After loading the interpreter, type RUN and press the RETURN key. The COMMODORE 64 will display the program title. Any of your programs may now be loaded using the standard LOAD commands (LOAD"" or LOAD"",8). Graphic displays may be restored (READ) to the screen by pressing function key F4 (F3 and the SHIFT key).

The display commands may be used in BASIC programs as you would use any BASIC commands. There is one restriction:

IF and REM commands require a colon (:) before the graphic command for proper operation. For example --- IF A=5 THEN DOT 2,3,5 will plot the point at 2,3 every time, regardless of the value of A. For proper operation of the IF, change the statement to - IF A=5 THEN : DOT 2,3,5. This will plot the point only if A=5. The REM also needs the colon to bypass any graphic command following the REM.

BASIC programs which use VIDEO BASIC-64 may be saved to tape or disk just like other BASIC programs. When later reloading these program, be sure that the VIDEO BASIC-64 interpreter is loaded and linked to BASIC (by typing RUN).

Because VIDEO BASIC-64 sometimes disables the clock, the timekeeping of your COMMODORE 64 may not be precise. You should keep this in mind if you are using VIDEO BASIC-64 and the built-in clock at the same time.

When you are finished using this program, reset the COMMODORE 64 by turning the power off and then on again or by typing SYS 64738. This frees all memory that is occupied and unlinks VIDEO BASIC-64 from BASIC.

B. THREE-DIMENSIONAL PLOTTING

Three dimensional function plotting is not really very difficult. There are two FOR loops set up, one going from left to right (X) and within that, a loop scanning from front to back (Y). The Y-axis is tilted by multiplying the Y-value by approximately .6. The computed Z-value is added to the Y-value. To hide the lines which should not be visible, plotting starts with a low value of y (towards the front) and increases (towards the back). If the total of .6Y + the function value (Z) is less than the prior point, it is not shown, as it would be hidden from view. Below is a sample program 3D plotting routine.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

```

1600 REM 3-D GRAPHICS
1610 DEF FNA(Z) = 90*EXP(-Z*Z/1500)
1620 FOR A=1 TO 2: HIRES 1,7
1630 TIC 10,10,2: G=160: K=60: FOR X=-100 TO 0: L=-60:
      H=5*INT(SQR(10000-X*X)/5)
1640 FOR Y=H TO -H STEP -5: Z=25+FNA(SQR(X*X+Y*Y))-.6*Y
1650 IF Z>L THEN L=Z: DOT G+X,Z+K,4: DOT G-X,Z+K,4
1660 NEXT Y,X: GOSUB 1680
1670 DEF FNA(Z)=38*(SIN(Z/24)+.48*SIN(3*Z/24))+20:
      NEXT: GOTO 10
1680 FOR B=1 TO 4000: NEXT: RETURN

```

SAMPLE 3D program

C. MEMORY ORGANIZATION

HEX ADDR	DEC. ADDR	USE
\$0400 - \$07FF	1024-2047	BASIC SCREEN AREA
\$0801 - \$73FF	2049-29695	VIDEO BASIC PROGRAM AREA
\$7400 - \$9FFF	29696-40959	VIDEO BASIC-64 MODULE
\$A000 - \$BFFF	40960-49151	REGION TRANSFER AREA; PATTERN FILL WORKAREA; DUMP/GREAD workarea
\$C000 - \$C3FF	49152-50175	VIDEO MATRIX AREA (all colors in HIRES; Colors A + B in MULTICOLOR)
\$C400 - \$C42F	50176-50223	GRAPHICS CHIP REG. SAVE AREA
\$C430 - \$C82F	50224-51247	COLOR MEMORY SAVE AREA (Color C in MULTICOLOR)
\$C840 - \$CBFF	51264-52223	SPRITE PATTERNS (15)
\$E000 - \$FFFF	57344-65535	GRAPHIC SCREEN SPACE

VIDEO BASIC-64 DEVELOPMENT PACKAGE

In addition to the areas shown on the memory map, VIDEO BASIC-64 uses the following locations in page 0:

HEX.	DECIMAL
\$03	3
\$26-\$29	38-41
\$4B-\$4E	75-78
\$52-\$53	82-83
\$9B	155
\$9E-\$9F	158-159
\$A5-\$A6	165-166
\$A8-\$AB	168-171
\$B0	176
\$B4-\$B6	180-182
\$BD	189
\$C3	195
\$F7-\$F8	247-248

Because Commodore BASIC is running along with VIDEO BASIC-64, most of the remainder of the page 0 is not available for use.

VIDEO BASIC-64 also uses the following other locations:

\$0334-\$03FF

D. COLOR MEMORY PROBLEM

There is only one color memory area inside the COMMODORE 64, so it must be shared by both the graphic screen and the BASIC screen. To conserve time and space, the color information from the BASIC screen is not saved. Therefore if you write something using a cursor color, it will come back in whatever colors you select in lines 63000+ after flipping the screen.

The second problem is that the BASIC screen is always active, even if you're looking at the graphic screen. This causes no problem, except that 200+ colors in multicolor have to put their color information into the common color memory area. If your graphic program writes to the BASIC screen, no problem, unless it causes the BASIC screen to scroll. When scrolling takes place, the color memory area is scrolled also---although it may contain your graphic colors. When this happens, you'll see the 200+ colors scroll up through your display. This often happens when your program ends and BASIC says READY.

There are two ways to prevent this---

First, make sure the BASIC screen stays clean by PRINTing a

VIDEO BASIC-64 DEVELOPMENT PACKAGE

CLR (shift HOME) at the start of your graphic program.

Second, don't let BASIC say READY until you want it to. Do this by putting an intentional loop into the program at the end -- such as 999 GOTO 999. BASIC will hang here. Shift the screen to BASIC (use F5) and then press STOP. The display can be viewed by pressing the F7 key.

E. TESTING AND DEVELOPMENT OF APPLICATIONS

There are two versions of the main VIDEO BASIC-64 program on the disk: the development version and the final version.

To develop a program, first LOAD "VB.DEV",8 and RUN it. After the copyright message you will be asked to select a printer. Key in one of the numbers and press <RETURN>. When loaded it will stop with the message "BREAK IN 62999". List the program. (See sample LISTING 1). These statements must be in your final version. Simply key in your program using any line numbers between 2 and 62998. There are 27190 BYTES FREE for your program.

Lines 63020-63024 control the BASIC screen colors and upper/lower case. Note that the color numbers in these lines must be the same as the color POKE values (VIDEO BASIC-64 color numbers - 1). Line 63024 must be either 21 for UPPER CASE/GRAPHICS character set or 23 for upper/lower character set.

After making these changes, you may delete all the remaining REMS to conserve space. If you delete line 1, you must change line 63025 GOTO to the new first line of your program.

It's a very good idea to SAVE your program frequently during development. Here's a skeleton program for your application:

```
0 IF PEEK(669)<>3 THEN 63000
1 REM  START OF YOUR BASIC PROGRAM
2 REM  *****
3 REM  *
4 REM  *      YOUR BASIC PROGRAM WOULD
5 REM  *      USE LINES 2 - 62999
6 REM  *
7 REM  *****
62999 STOP:REM END OF YOUR BASIC PROGRAM
63000 POKE 669,PEEK(669)+1
63002 ON PEEK(669) GOTO 63005,63015,63020
63005 PRINT "<CLR><CUR DN><CUR DN>,"VIDEO BASIC-64"
```

VIDEO BASIC-64 DEVELOPMENT PACKAGE

```
63006 PRINT,"<CUR DN>SELECT PRINTER:":PRINT,"<CUR DN>1=
      EPSON":PRINT,"2= OKIDATA
63007 PRINT,"3= CBM 1525/MPS801":PRINT,"4= PROWRITER":
      PRINT,"<CUR DN>";:INPUT A
63010 IF A<5 AND A>0 THEN LOAD 'VBPR'+STR$(A),8,1
63011 GOTO 63005
63015 LOAD "VBMAIN",8,1
63020 POKE 30581,2 :REM BASIC BORDER COLOR
63022 POKE 30582,0 :REM BASIC SCREEN COLOR
63023 POKE 30547,7 :REM BASIC CURSOR COLOR
63024 POKE 30573,21 :REM 21=UPPER CASE; 23=LOWER CASE
63025 SYS30467: CLR: GOTO 1:REM INITIALIZE VB-64
```

F. MEMORY MANAGEMENT

If you need an area of memory for screen saves, special character sets, etc., simply lower the top of BASIC. Do this by inserting two POKES on line 63025 as follows:

```
63025 SYS30467:POKE55,t1:POKE 56,th:CLR:GOTO 1
```

where t1 is the low byte of the top of memory address and th is the high byte of the top of memory address. The normal values in 55 and 56 are 0 and 116 (\$7400). See the MEMORY MAP for details.

G. MAKING A DISTRIBUTION VERSION

To make a distribution diskette (one that you can sell or give to friends), you will have to load and RUN a program called **MAKEDISK**. This program optionally initializes (NEW) the diskette and copies the necessary **VIDEO BASIC-64** modules onto it. Follow the instructions that **MAKEDISK** displays on the screen.

Then you must insert the diskette with your program and LOAD your program into memory (**VIDEO BASIC-64** does not have to be running).

List line 63015 of your program and change it to LOAD "VBXXXX",8,1 and SAVE your program under its final name on the diskette created by **MAKEDISK**. Reset the '64 and try the program from the newly created diskette.

The runtime version of **VIDEO BASIC-64** has only a few keywords it can recognize from the keyboard. All others cause **?SYNTAX ERROR**. The keywords recognized are:

VIDEO BASIC-64 DEVELOPMENT PACKAGE

RUN

GOTO

LOAD

DUMP

GREAD

NORM

GRAPH

This should prevent anyone from SAVING, LISTING or modifying your programs.

There is no protection against disk copying.

XI. ERRORS

As VIDEO BASIC-64 reads your commands, it checks to see if the required parameters are present.

In most cases it will indicate an error with a ?SYNTAX ERROR message. If the command was issued in a running program, the line number of the bad statement is shown. Generally, the problem will be too many or too few parameters. Simply check the statement against the proper form in the manual, correct it and retry.

The COPY statement is slightly different.

If the target line number (1st line of HEX, SDATA, BIT or COLORS is missing or not one of those words) an ?UNDEFINED LINE NUMBER message will appear. The line number shown will be that of the COPY statement.

If the first character after the HEX, BIT or COLORS isn't a " (quote) then the message COPY LINE LENGTH will appear, along with a ?SYNTAX ERROR referencing the COPY statement.

If there are not enough characters in the line of the HEX (42), BIT (24) or COLORS (12), then the COPY LINE LENGTH error will also appear.

If there are fewer lines than needed in the HEX (3), BIT (21) or COLORS (21) or if all lines are not the same type, the message COPY LINE COUNT error will appear, along with a ?SYNTAX ERROR pointing to the COPY statement.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

TURTLE GRAPHICS COMMANDS

TURTLE c,x,y	- initialize turtle
TCOLOR c	- set turtle line color
TUP	- lift the turtle pen
TDOWN	- put the turtle pen down
TURN d	- turn the turtle
TURNT0 d	- point the turtle
MOVE a	- move turtle
BYE	- make turtle disappear
TPOS(v)	- return position or direction

INPUT FUNCTIONS

JOY(P)	- return joystick position
PADDLE(p,d)	- return paddle value
PEN(c)	- return lightpen value

TIMERS

SCTR c,v	- set counter
CTR(c)	- read counter

COLLISION FUNCTIONS

SCOLL(s1,s2[,s3,...s8])	- return sprite/sprite collision value
BCOLL(s1[,s2,...s8])	- return sprite/background collision value

SOUND COMMANDS

SOUND a,p,d	- play simple sound
GEN a,b,c,d,e,f,g,h,i	- set up generator control
VOL a,b,c,d,e	- common generator control
SET a,lllll	- set generator pattern
TDATA"ppttiittiitt...."	- define TUNE pattern
TUNE a,d	- play complex pattern

OTHER COMMANDS

BRACKET[- repeat commands within
BRACKET]	brackets
BRACKETE	- leave repeat
HARD d,b,s,a,	- hard copy to printer
HIRES 99,0	- reset stack
HIRES 99,1	- reset computer
HIRES 99,2	- disable STOP key
XFER fh,fl,th,tl,nh,nl,[d,s]	- transfer memory
REGION 0,x1,y1,x2,y2	- copy graphic to workarea
REGION 1,x,y,m,c	- copy workarea to graphic

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX A - PRINTER/INTERFACE SUPPORT

VIDEO BASIC-64 supports the Commodore 1515 and 1525E printers if they are connected directly to the Commodore 64 or 1541 disk drive via the serial cable.

VIDEO BASIC-64 supports the Epson MX-80 and MX-100 with Grafrax, the FX-80, FX-100 and RX-80, and the Gemini 10 and 15 printers.

VIDEO BASIC-64 supports the following OKIDATA printers:

MICROLINE 92
MICROLINE 82A with OKIGRAPH kit
MICROLINE 83A with OKIGRAPH kit
MICROLINE 93
MICROLINE 84 STEP 2

VIDEO BASIC-64 also supports the C. Itoh Prowriter 8510A.

For the non-Commodore printers, you must connect the Commodore 64 to the printer with one of the following parallel printer interfaces:

MANUFACTURER	MODEL	USE THIS FORM OF THE HARD COMMAND	SWITCHES ON
CARDCO Wichita, KS	CARD?/A	HARD d,b,5,0	
CARDCO Wichita, KS	CARD?/G+	HARD d,b,25,0	
ECX, Inc. Walnut Creek, CA	C-6401	HARD d,b,0,1	*
MICROWORLD ELECTRONIX Lakewood, CO	MW-302	HARD d,b,0,1 HARD d,b,0,0	3,4 3
MSD, Inc. Dallas, TX	CPI	HARD d,b,0,0	1,3,5

* requires that the three-position switch is set to the center position

in the above commands:

d is the device number (usually 4)
b is 0 for small size; or 1 for large size

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX B - TONE/PITCH CHART

* * DECIMAL * *

Octave	Note											
	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
1	-	-	-	0	1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15	16	17	18	19	20
3	21	22	23	24	25	26	27	28	29	30	31	32
4	33	34	35	36	37	38	39	40	41	42	43	44
5	45	46	47	[48]	49	50	51	52	53	54	55	56
6	57	58	59	60	61	62	63	64	65	66	67	68
7	69	70	71	72	73	74	75	76	77	78	79	80
8	81	82	83	84	85	86	87	88	89	90	91	92
9	93	94	95									

* * HEXADECIMAL * *

Octave	Note											
	A	A#	B	C	C#	D	D#	E	F	F#	G	G#
1	-	-	-	00	01	02	03	04	05	06	07	08
2	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
3	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20
4	21	22	23	24	25	26	27	28	29	2A	2B	2C
5	2D	2E	2F	[30]	31	32	33	34	35	36	37	38
6	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44
7	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50
8	51	52	53	54	55	56	57	58	59	5A	5B	5C
9	5D	5E	5F									

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX C - CONVERTING FROM ULTRABASIC-64 TO VIDEO BASIC-64

If you have a program already written using **ULTRABASIC-64**, it can be easily converted for **VIDEO BASIC-64** as follows:

Review and alter the **ULTRABASIC-64** program so it has no lines numbered 0 or 1, nor any lines greater than 62998 and save it onto disk.

Reset the computer and load the **VIDEO BASIC** development version (**VBDEV**) into the '64.

Load **VBSTART** into the computer, then **LIST** it onto the screen.

Move the cursor up one line into the (**READY**), then load the revised **ULTRABASIC-64** program, being careful not to destroy the lines of **VBSTART** on the screen.

Place the cursor on line 0 and press **<RETURN>**. Continue to press **<RETURN>** for each of the lines of **VBSTART**. This will add those lines to the revised **ULTRABASIC-64** program.

Locate any **[,]** or **EXIT** commands in the revised **ULTRABASIC-64** program. These will show as **XFER**, **REGION** or **BRACKET** commands in the listing. Replace these with **BRACKET[, BRACKET]** and **BRACKETE** respectively.

SOUND and **TDATA** commands will have to be changed for the new tuning. A rough guide is :

OLD VALUE = NEW VALUE.

<u>HEXADECIMAL</u>	<u>DECIMAL</u>	<u>to</u>	<u>DECIMAL</u>	<u>HEXADECIMAL</u>
01	1		0	00
02	2		11	0B
03	3		18	12
04	4		23	17
05	5		27	1B
06	6		30	1E
07	7		33	21
08	8		35	23
09	9		37	25
0A	10		39	27
0B	11		41	29
0C	12		42	2A
0D	13		44	2C
0E	14		45	2D
0F	15		46	2E

VIDEO BASIC-64 DEVELOPMENT PACKAGE

HEXADECIMAL	DECIMAL	to	DECIMAL	HEXADECIMAL
10	16		47	2F
11	17		48	30
12	18		49	31
13	19		50	32
14	20		51	33
16	22		52	34
17	23		53	35
18	24		54	37
1A	26		55	37
1B	27		56	38
1D	29		57	39
1E	30		58	3A
20	32		59	3B
22	34		60	3C
24	36		61	3D
26	38		62	3E
28	40		63	3F
2B	43		64	40
2D	45		65	41
30	48		66	42
32	50		67	43
35	53		68	44
38	56		69	45
3C	60		70	46
3F	63		71	47
43	67		72	48
47	71		73	49
4B	75		74	4A
50	80		75	4B
55	85		76	4C
5A	90		77	4D
5F	95		78	4E
64	100		79	4F
6A	106		80	50
71	113		81	51
77	119		82	52
7F	127		83	53

SAVE the revised program under its new VIDEO BASIC-64 name.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX D - DUMP FORMAT ON DISK

Position	Contents
1-2	Disk sector link bytes
3-4	Starting save address (\$A000)
5-8196	8192 Bit map graphic screen Dump mode 1 stops here
8197-9220	1024 Video matrix Dump mode 2 stops here
9221-9268	48 VIC II CHIP registers DUMP mode 3 stops here
9269-10292	1024 COLOR MEMORY SAVE AREA DUMP mode 4 stops here
10293-10308	16 Unused area
10309-11268	960 15 SPRITE PATTERNS DUMP mode 0 stops here

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX E - USING XREF-64 for VIDEO BASIC-64 programs

XREF-64 available from ABACUS software is a powerful software development tool that provides a complete list of all variables, line numbers and BASIC keywords and shows the line numbers where each is used. It may be customized to handle extended keyword lists (such as those in VIDEO BASIC-64). To make it easy to handle the VIDEO BASIC-64 keywords with XREF-64, the distribution diskette includes a program which will create a VIDEO BASIC keyword file for XREF-64.

Reset the '64.

Insert the VIDEO BASIC-64 distribution diskette into drive 8 and type LOAD "VBKEYWORDS" ,8.

Remove the tape covering the diskette protection notch and insert your XREF-64 diskette into drive 8.

Type RUN. You will be asked for the name the keyword file is to be given. You may use any name but ZZ.VB is suggested.

When you press <RETURN> the file will be written to disk.

Remove the XREF-64 diskette from the drive and replace the tape over the protection notch.

When running XREF-64 for VIDEO BASIC-64 programs, key the name you gave (ZZ.VB) the keyword file to select it instead of the standard keywords.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX F

A short lesson in hexadecimal numbers

Computers represent data in binary format. In the binary number system, a digit may take on a value of either 0 or 1. Each digit position in the binary number system has a value that is a power of 2, just as each digit in the decimal number system has a value that is a power of 10.

BINARY NUMBER SYSTEM

Power of 2	7	6	5	4	3	2	1	0
Value	128	64	32	15	8	4	2	1

DECIMAL NUMBER SYSTEM

Power of 10	5	4	3	2	1	0
Value	100000	10000	1000	100	10	1

So if you want to represent the decimal number 17 in the binary number system, you would use the string of binary digits 10001 which would stand for:

$$\begin{array}{ccccccccc} & 4 & & 3 & & 2 & & 1 & & 0 \\ 1 & \times & 2 & + & 0 & \times & 2 & + & 0 & \times & 2 & + & 0 & \times & 2 & + & 0 & \times & 2 \\ = & 16 & + & 0 & + & 0 & + & 0 & + & 0 & + & 1 \\ = & 17 \end{array}$$

Representing numbers as a string of binary digits (commonly called "bits") would look as shown:

Decimal	Binary	Decimal	Binary
0	0	13	1101
1	1	14	1110
2	10	15	1111
3	11	16	10000
4	100	17	10001
5	101	18	10010
6	110	19	10011
7	111	20	10100
8	1000	21	10101
9	1001	22	10110
10	1010	23	10111
11	1011	24	11000
12	1100	25	11001

Eight bits (remember - binary digits) is called a byte. A byte is the smallest accessible unit of data in the COMMODORE 64. Eight bits is capable of representing a value equivalent to decimal 255.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

A string of eight bits however, is not easy to read or write. Because they are awkward to interpret, bits are often represented in the hexadecimal numbering system (base 16). Each hexadecimal digit stands for four bits.

Hexadecimal notation uses 16 symbols to represent the 16 different values. These symbols are the numerals 0 through 9 and the letters A through F. Below is an equivalency chart:

DECIMAL	BINARY	HEXADECIMAL
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

To convert binary numbers to hexadecimal notation, divide the binary number into groups of four bits, starting at the righthand side. Replace each group of four bits by the corresponding hexadecimal symbol. If the left most group of bits is not a full four bits, then fill in with zeros. The example below illustrates this:

101001101011010110	=	[binary number]
0010/1001/1010/1101/0110	=	[4-bit groups]
2 9 A D 6		[hexadecimal number]

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX G - COLOR NUMBERS

<u>COLOR</u>	<u>NUMBER</u>	<u>COLOR</u>
	1	BLACK
	2	WHITE
	3	RED
	4	CYAN
	5	PURPLE
	6	GREEN
	7	BLUE
	8	YELLOW
	9	ORANGE
	10	BROWN
	11	LIGHT RED
	12	DARK GRAY
	13	MEDIUM GRAY
	14	LIGHT GREEN
	15	LIGHT BLUE
	16	LIGHT GRAY

Note that these are not the same color numbers that you
POKE to color memory.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

APPENDIX H - Tutorial 3

The advanced features of VIDEO BASIC-64 are reprinted here in this tutorial for your reference convenience.

One outstanding feature of VIDEO BASIC-64 is its ability to setup two graphic screens. This can be done either of two ways:

One is to set up two save areas and transfer the active screen to and from these areas. This has the advantage that either screen save can be copied into the active screen. This can provide a "retry" function if the active screen is saved before an operation, as in CADPAK-64. The easiest way is to setup GOSUB routines, one pair to transfer the active screen areas to each save area, and another pair to transfer save to active. Then GOSUB to the proper routine to move the screen to or from either save area. The commands and memory allocation are similar to the second method.

The second method uses the swap option of the memory transfer command to exchange the active screen with one save area. This saves memory and is faster, but can't provide a "retry" function. Because it is more complicated, we will demonstrate this approach. For either method, make room by moving the top of basic down.

You will need the following memory for each screen save area:

8192 (DEC)	\$2000 (HEX)	- bit map
1024 (DEC)	\$0400 (HEX)	- video matrix [color info]
48 (DEC)	\$0030 (HEX)	- chip regs [always]
1024 (DEC)	\$0400 (HEX)	- color 3 save [multi only]
0960 (DEC)	\$03C0 (HEX)	- sprites [only if sprites]

Note that the last two areas are optional. The total is 11248 bytes. For the swap example we need only one save area, and will save everything:

	HEX	DECIMAL
		HI LOW
Top of video basic	\$7400	[116 0]
Less:		
for bit map	\$2000	[32 0]
for video matrix	\$0400	[04 0]
for registers	\$0030	[00 48]
for color mem	\$0400	[04 0]
for sprites	\$03C0	[03 192]

VIDEO BASIC-64 DEVELOPMENT PACKAGE

It is easier to do arithmetic using the decimal columns (don't forget carries). Add up the space required:

Top of Video Basic	\$7400	
less space required:	- \$2BF0	[43 240]
yields:	-----	\$4810 [72 16]

To move the top of basic down, POKE the decimal value-hi into address 56 and the decimal value-low into address 55.

POKE 56,72: POKE 55,16 , then do a CLR command.

Note how the free changes after this: Before free 11625, after free 11633. (The CLR also clears out any variables, so the difference between the two is slightly less than the amount of screen save memory allocated)

Look at the memory map in the manual:

The bit map area is located	\$E000-\$FFFF
The video matrix is located	\$C000-\$C3FF
The sprites are located	\$C840-\$CBFF

These are the active areas when VIDEO BASIC-64 is showing the graphic screen in addition to the C-64 requirements.

Video chip registers	\$D000-\$D02F
Color memory	\$D800-\$DBFF

When the video screen is switched to normal (F5 key), the registers are saved in the save area \$C400-C42F, and color memory is saved into \$C430-C82F.

When the screen is switched back to graphics (F7 key), the contents of \$C400-C42F are moved to the chip regs and \$C430-C82F are moved to color memory.

Blocks of memory are transferred with the XFER command. The format is:

XFER FH,FL,TH,TL,NH,NL,[D,S]

H and L are the hi and lo bytes of addresses IN DECIMAL

FH,FL - from address
TH,TL - to address
NH,NL - number of bytes to move

D controls the \$D000-\$DFFF memory:

0 - sees I/O ports and color memory
1 - sees character ROMS
2 - sees RAM in \$D000-\$DFFF

VIDEO BASIC-64 DEVELOPMENT PACKAGE

S controls swapping:

- 0 - normal (transfers from to)
- 1 - swaps from and to areas

To swap screens we need XFERs, the formats are as follows:

XFER 224,0,72,16,32,0,0,1 for the bits

- 224, 0 = \$E000 (from bit map area)
- 72,16 = \$4810 (to bit save area above basic)
- 32, 0 = \$2000 (number of bytes)
- 0 = D block rom (doesn't matter here)
- 1 = swap two areas

XFER 192,0,72+32,16,4,0,0,1 for the video matrix

- 192, 0 = \$C000 (from area)
- 72+32,16 = \$4810+\$2000 (to save area above bits)
- 4, 0 = \$0400 (number of bytes)
- 0 = D block rom (doesn't matter)
- 1 = swap two areas

Note that it only swaps bit-map and video matrix areas, not the registers (border color is in registers and therefore is not swapped). Now try it and watch the screen.

The actual video chip registers are at address \$D000-D02F. You must be very careful when xfering to the active chip registers that no junk is sent there because the regs also control interrupts and will quickly hang up the computer.

For example, if we just swap the registers with our new save area above basic, we will move whatever junk happens to be there into the regs on the first swap...and probably hang up the computer!

The way to beat this is to do a one-way transfer from the registers to the screen save area first, after setting up the first graphic screen. Thereafter, we can use swaps, since the data is valid. Note also that we will be xfering to the active registers, so the screen must be in graph mode or we will get the basic screen registers instead.

The one-way transfer is as follows:

GRAPH:XFER 208,0,72+32+4,16,0,48[,0,0]

issue it once after the first HIRES or MULTI command.

- 208, 0 = \$D000 registers
- 72+32+4,16 = next space in save
- 0,48 = \$30 number of bytes of regs
- [0, = normal D block(I/O)
- 0] = no swap

VIDEO BASIC-64 DEVELOPMENT PACKAGE

Now adding the following:

```
XFER 208,0,72+32+4,16,0,48,0,1
    to the swap routine will swap border colors, sprite
    controls, etc.
```

The color memory for color 3 is at address \$D800-DBFF. Just swap \$D800-DBFF and our save area:

```
XFER 216,0,72+32+4,64,4,0,0,1

216,0 = $D800 color mem
72+32+4,64 = start of next in save
4, 0 = $400 bytes (1024)
0, 1 = $D000 block = I/O and swap
```

Sprites are at \$C840-CBFF. These may be swapped just like the bit area:

```
XFER 200,64,72+32+4+4,64,3,192,0,1

200,64 = $C840 start sprites
72+32+4+4,64 = next in save
3,192 = $3C0 = number of bytes (960)
0, 1 = $D000 block=I/O and swap
```

Because all registers are saved, sprites, positions, hires/multicolor are swapped.

The block command has an expanded form which allows you to do several interesting things:

```
BLOCK x1,y1,x2,22,c,m,p1,p2...,p8

X1,Y1,X2,Y2,C are explained before

m = mode
    0 = normal mode
    1 = reverse video matrix (color)
    2 = pattern filling

p1,p2...,p8 are pattern values.
```

The reverse operation works somewhat differently in hires and multicolor. In hires, the background and foreground colors are switched in the area of the coordinates in the BLOCK. When a hires screen is setup with the HIRSES command, the bit map area is cleared and the video matrix is filled with two colors, the background which you see (since screen is clear), and the foreground color everywhere which you don't see, since the screen is clear. The default color for the foreground color is white, but in this example it was set to blue. All video basic graphics commands which

VIDEO BASIC-64 DEVELOPMENT PACKAGE

include colors (draw,box,etc.) automatically put the command color into the video matrix foreground color cell(s) so the graphic shows proper color. This BLOCK command simply switches the background and foreground colors in an area. Note that the sprite was not affected since it's not really part of the graphic screen. In multicolor mode, colors 1 and 2 are switched in the area. Note how the edges are affected (also the sprite doesn't change). A second command will undo because it reverses it back. These can be used for flashing titles.

Patterns are defined as 8 numbers each between 0-255 and are created as follows:

Draw the pattern on paper, and code each row:

XXXX....	128+64+32+16	= 240
...X..XX	16+0+0+2+1	= 19
..X..X.X	32+00+0+4+0+1	= 37
.XX.XX..	64+32+00+8+4	= 108
..X..X.X	32+00+0+4+0+1	= 37
...X..XX	16+0+0+2+1	= 19
....XX..	8+4	= 12
.....X..	4	= 4

so the result of the command:

BLOCK 50,50,200,140,1,2,240,19,37,108,37,19,12,4

Note that block mode = 2 for patterns

There is a pattern filling option of the fill command. Patterns are defined the same way:

FILL x,y,c,b,m,p1,p2...,p8

x,y,c are as explained before

b must be 0 in hires or border paintbrush #1 in MULTI

m = mode

0 = solid fill

1 = pattern fill

p1,p2,...,p8 are the pattern values.

The last new command is the REGION command. This lets you extract or copy back a rectangle of a graphic. The area under BASIC ROM (\$A000) is used to save the graphic image. This way the screens can be switched after the graphic is copied out, allowing transfers from screen to screen.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

The copy out command is:

```
REGION 0,x1,y1,x2,y2
```

0 = specifies the copy out
x1,y1 = one corner of rectangle
x2,y2 = other diagonal corner

The copy back command is:

```
REGION 1,xul,yul,m,c
```

1 = copy back

xul,yul = coordinate of upper left corner of where graphic is to go.

m = mode

0 = move graphic
1 = OR graphic with screen
2 = EXCL-OR graphic with screen
3 = AND graphic with screen

c = color move control

0 = no color copy
1 = copy color (only 1&2 in multi)

For example, let's note the following:

First, copy out with:

```
REGION 0,199,80,300,180
```

199, 80 = X and Y coordinate of one corner
300,180 = X and Y coordinate of the other

then, copy back (in a loop)

```
FOR I = 10 TO 50 STEP 8  
REGION 1,I,190,0,0  
NEXT
```

I,190 = X and Y coordinate of upper left
0 = move mode
0 = no color

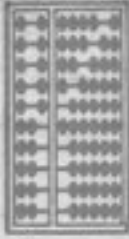
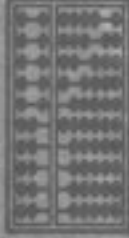
Note that when the pattern overlaps anything else, the new one covers the old. Now try doing the same in OR mode, where the old pattern(s) show thru. Note how the color of the pattern is whatever color happens to be in the area. This is the reason for the extra options on the HIRRES and MULTI commands to set colors all over the screen (refer to section of manual on this). The same pattern can be transferred to the other hires screen after swapping it.

VIDEO BASIC-64 DEVELOPMENT PACKAGE

The lower left image was copied with move and color, the one on right was OR'd only, no color.

This completes tutorial 3 for VIDEO BASIC-64. For an in-depth preview of the many examples discussed here, refer to the Video Basic tutorial included with your Video Basic diskette (just LOAD "VBTUTOR3",8 and RUN). We hope you enjoy using it.

Abacus Software



(C) 1985 RC WAINWRIGHT

VIDEO BASIC

56735

SN:

MFD: JANUARY 20, 1986

Abacus Software • P.O. Box 7211 • Grand Rapids, MI 49510 • (616) 241-5510

REGISTRATION CARD

Registration # 56735 Product: _____
Name _____
Address _____

City _____ State _____ Zip _____

Purchase Information:

Dealer _____
Address _____
City _____ State _____ Zip _____

Returning this registration card entitles you to phone support for the above product. You may also obtain a backup copy of the diskette for a handling charge of \$10.00. This card and a check, money order or credit card number must accompany this request. Purchase orders are not acceptable.

BACKUP COPY? ☐ No, do not send a backup, but register my purchase
☐ Yes, send a backup copy, payment is enclosed

Credit card# _____
Expiration Date / /

You Can Count On

Abacus Software



P.O. Box 7219
Grand Rapids, MI 49510

**GET THE MOST OUT OF YOUR
COMMODORE-64**



VIDEO BASIC 64

Graphics & Sound development package

This superb package is for software developers using graphics, music or sound effects. You can develop your software using VIDEO BASIC 64 and distribute the runtime version without having to pay any royalties. Video commands for hires, multicolor, sprite and turtle graphics. Audio commands for simple or complex music or sound effects. Hardcopy to 1525/801, Epson, Gemini, Okidata or Prowriter printers. Game features for sprite collision detection, lightpen, joystick, game paddles, etc. Memory management for multiple graphics screens, screen copy, much more. For C-64 and 1541 disk drive. Printer optional.