YOU CAN COUNT ON

Abacus Software

==========

Ultrabasic-64

software development package for high resolution, multicolor, sprite and turtle graphics, sound and game features for the COMMODORE-64

==========

YOU CAN COUNT ON

Abacus Software

P.O. BOX 7211, Grand Rapids, MI 49510

Third Printing - November 1983

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 90 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

ULTRABASIC-64 is the "ultimate" software package for those that want to use the wonderful features for which you bought the COMMODORE 64. **ULTRABASIC-64** gives you easy to use graphics, fun to use TURTLE graphics, POKEless music and sound, convenient game features and a screen dump to the printer. It's a very powerful bag of tools for software development.

The Roy Wainwright family lives in a quiet rural area of Pennsylvania - probably 700 miles or so from Grand Rapids. Therefore most of the details of Roy's software development is handled over the phone or by mail. But that does not seem to slow Roy down. **ULTRABASIC-64** is Roy's third major package for the COMMODORE 64. And he's done this in less than 8 months in addition to his duties as a husband, father and data processing executive.

And he has more software packages close to completion.

The final specifications for this package were completed in a hotel room 2500 miles from both Pennsylvania and Michigan. But we've been busy putting the finishing touches on the software and readying the documentation. We feel that **ULTRABASIC-64** is the finest package for software development available for the COMMODORE 64.

Once again we owe a great deal of thanks to Roy and Ann Wainwright and family. They have a great deal of patience to put up with our late night calls and last minute changes.

Arnie Lee
May 4, 1983
Grand Rapids, MI

TABLE OF CONTENTS

I. INTRODUCTION

The COMMODORE 64 is a marvelous computer. It has excellent color graphics, quality sprite animation, unsurpassed music and sound synthesis capabilities, sophisticated game features, and much more - all at a very affordable price. But many of these features are hard to use without resorting to POKEs, PEEKs, trial and error. **ULTRABASIC-64** is a software package that makes it simple to use these features.

ULTRABASIC-64 makes it easy to do any of the following:

* Plot in either high resolution or multicolor modes. You can plot dots, lines, boxes and circles.
* Define and manipulate sprites in either high resolution or multicolor modes.
* Create sound effects and music using any or all of the three voices.
* Detect sprite/sprite or sprite/background collisions
* Read the joystick, game paddle or lightpen ports directly.
* Draw using a very friendly set of TURTLE graphics commands.
* Get hardcopy of the graphics screen in either of two sizes on a Commodore, or EPSON compatible or OKIDATA printer.

ULTRABASIC-64 adds 50 commands to BASIC. **ULTRABASIC-64** does not perform magic - it merely makes it easy to use those features of your COMMODORE 64 which made you buy the computer in the first place. Once **ULTRABASIC-64** is initialized these extra commands may be used with the other standard BASIC commands to create spectacular programs.

Commands may be used in a running program or directly from the keyboard. Programs which use any **ULTRABASIC-64** commands may be SAVEd, LISTed or LOADed, just like any other BASIC program. (However **ULTRABASIC-64** must first be initialized).

By using **ULTRABASIC-64** commands in a simple BASIC program you can show off the truly amazing capabilities of your computer.

The **ULTRABASIC-64** package consists of several parts:

* this user's manual
* a program for loading **ULTRABASIC-64** for use with a COMMODORE 1515 or 1525 printer called **UBCBM**
* a program for loading **ULTRABASIC-64** for use with an EPSON MX-80 or FX-80 printer **UBEPSON**
* a program for loading **ULTRABASIC-64** for use with several OKIDATA Microline printers **UBOKI**

tically RUN by one of the above initializers) called **ULTRA**

* a demonstration program called **UBDEMO**
* Part I of a tutorial on using **ULTRABASIC-64** called **UBTUTOR1.**
* Part II of a tutorial on using **ULTRABASIC-64** called **UBTUTOR2.**

The next section is **GETTING STARTED USING ULTRABASIC-64.** It will show you how to set up and use the **ULTRABASIC-64** commands, demos and tutorials. The rest of the manual is organized in sections that describe the various groups of features.

II. GETTING STARTED USING ULTRABASIC-64

In this section we show you how to use **ULTRABASIC-64.** You will learn how to initialize **ULTRABASIC-64** and run the sample programs that are part of the package.

The distribution cassette or diskette contains the following:

NAME	CONTENTS
UBCBM	the initializer for ULTRABASIC when using a COMMODORE 1515 or 1525 printer
UBEPSON	the initializer for ULTRABASIC when using an EPSON MX-80 or FX-80 printer
UBOKI	the initializer for ULTRABASIC when using an OKIDATA Microline printer
ULTRA	the ULTRABASIC-64 interpreter which adds the 50 commands to BASIC.
UBDEMO	a program which shows off most of the features of the ULTRABASIC-64 package
UBTUTOR1	tutorial on the use of the new commands
UBTUTOR2	tutorial (cont) on the use of the new commands

Follow these directions and you will be able to run the sample programs:

1. If using cassette, insert the distribution cassette intothe cassette drive making sure it is rewound.

   or

   If using diskette, carefully insert the distribution diskette into the drive and close the disk drive door.

2. There are three different "initializers" for **ULTRABASIC-64.** If you are not using a printer, it does not matter which initializer you use.

   One initializer sets up **ULTRABASIC-64** for use with a COMMODORE 1515 or 1525 printer for hardcopy. This program is called **UBCBM.**

   A second initializer sets up **ULTRABASIC-64** for use with an EPSON MX-80, FX-80 RX-80 or GEMINI series printer. This program is called **UBEPSON.**

   A third initializer sets up **ULTRABASIC-64** for use with an OKIDATA Microline series printer. This programis called UBOKI.

   If using cassette type: **LOAD "UBCBM"**, **LOAD "UBEPSON"** or **LOAD "UBOKI"** and press the **RETURN** key. Press PLAY on the cassette recorder when asked to do so by the computer. When the message **FOUND** "UBCBM" (or UBEPSON or UBOKI) appears on the screen, press the **C=** key to continue the loading.

   or

If using diskette type: **LOAD "UBCBM",8**, **LOAD "UBEPSON",8** or **LOAD "UBOKI",8** and press the **RETURN** key.

3. When the appropriate initializer is loaded, **READY** will reappear on the screen. Type **RUN** and press the **RETURN** key. If using **UBEPSON** or **UBOKI** then the computer prompts you to enter a:

SECONDARY ADDRESS NUMBER (0-255)

If you printer interface does not need or respond to secondary addresses with the printer OPEN command, then press <RETURN>. If a secondary address is needed to configure the interface, then key in the number followed by <RETURN>. See APPENDIX D for more information. The next prompt is:

ASCII TRANSLATE? (Y/N)

If your interface does not change the 8 bit codes in any way, then press N (no). If your interface translates Commodore ASCII to standard ASCII, then answer Y, and **ULTRABASIC-64** pre-translates the graphic data so your interface prints properly.

Next **ULTRABASIC-64** interpreter is automatically loaded and run from either cassette or diskette. At this point

ULTRABASIC is initialized

A short introduction will appear on the screen. Press STOP on the cassette recorder if using cassette.

4. Now you can run the demonstration program. If using cassette type: **LOAD "UBDEMO"** and press the **RETURN** key to load the demonstration program from cassette. Press PLAY on the cassette recorder when asked to do so by your computer. When **FOUND** "UBDEMO" appears on the screen, press the **C=** key to continue the loading.

or

If using diskette type: **LOAD "UBDEMO",8** and press the **RETURN** key to load the demonstration program from diskette.

5. Using either cassette or diskette type **RUN** and press the **RETURN** key. The demo program will begin. The screen displays will appear in both high resolution and multicolor mode! This demo shows off most of the features of **ULTRABASIC-64** and will run for several minutes as it goes through its paces. You may interrupt it by pressing the RUN/STOP key on the keyboard. Press the **F5** key to switch to the text screen.

6. If you **LIST** the program you will see the commands that do the plotting. This is an easy way to learn the use

of the **ULTRABASIC-64** commands.

7. A easier way to learn about the new graphics commands is from the tutorial. It contains a step-by-step description of most of the commands.

8. If using cassette type: **LOAD"UBTUTOR1"** and press the **RETURN** key to load part I of the tutorial program. When **FOUND** "UBTUTOR1" appears on the screen press the **C=** key to continue loading.

   or

   If using diskette type: **LOAD "UBTUTOR1",8** and press the **RETURN** key. The tutorial will begin.

   Just follow the directions and learn how to use **ULTRABASIC-64.** Part II of the tutorial is loaded the same way but the name is **UBTUTOR2.**

III. COMMAND FORMATS

ULTRABASIC-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. 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.

COLOR NUMBER	COLOR
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 <u>not</u> the same color numbers that you POKE to color memory.

TABLE 1 COLOR NUMBERS

IV. HIRES/MULTICOLOR GRAPHICS

A. GRAPHICS DISPLAY FORMAT

The **ULTRABASIC-64** display screen is laid out corresponding to normal graphing arrangements. 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. 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 section entitled **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. 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) allowed plus three different plot command colors. You can think of it easiest 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 number).

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

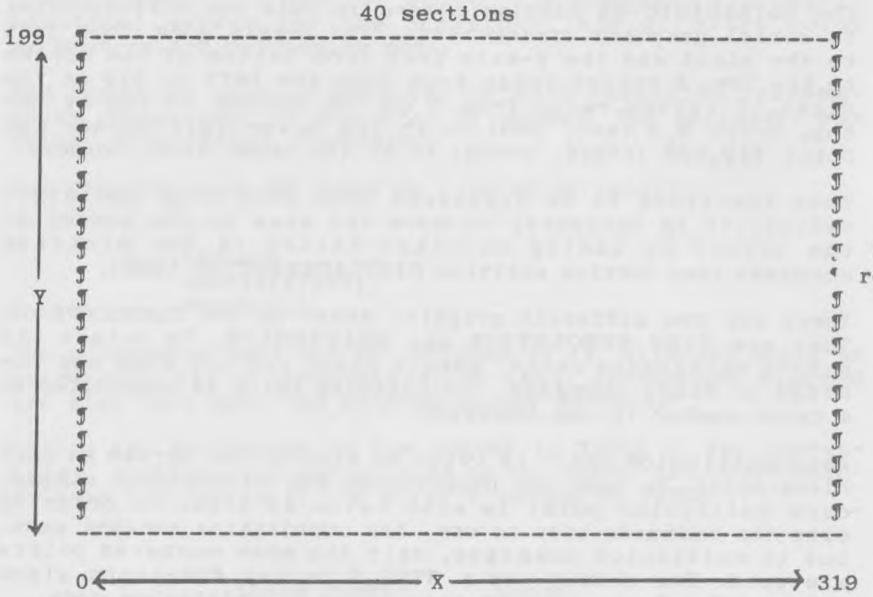


FIGURE 1 SCREEN LAYOUT

P. DISPLACMENT 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

ULTRABASIC-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 GRAPHIC DISPLAYS

Graphic displays can be dumped (saved) directly to tape or disk . Pressing Function key 2 (shift and F1) during a graphic display program will switch the screen to normal (text) mode, and display a filename and device prompt. Simply key the filename enclosed in "" and press the **RETURN** key ("" is ok). The standard device is the cassette recorder but you may add a ',' and a different device number after the filename (such as disk: ',8'). The graphic display will reappear as dumping takes place.

Graphic displays can be read from tape or disk by pressing Function key 4 (SHIFT & F3). The prompt and actions are the same as with the dump function key.

E. HIRES/MULTICOLOR COMMANDS

HIRES a,b - setup HIgh RESolution screen

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

a - this is a value between 1 and 16 which sets the screen color. This number is chosen from TABLE 1.

b - this is a value between 1 and 16 which sets the border color and is also chosen from TABLE 1.

MULTI a,b - setup MULTIcolor screen.

This command clears the screen and sets the screen and border colors for subsequent multicolor graphics.

a - this is a value between 1 and 16 which sets the screen color. This number is chosen from TABLE 1.
b - this is a value between 1 and 16 which sets the border color and is also chosen from TABLE 1.

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).

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-cooordinate 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 - The is the color of the CIRCLE (see TABLE 1).

xf - The x-factor to multiply times X-coordinate

yf - The y-factor to multiply times Y-coordinate

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.

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 literal 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).

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. There-

fore, to reset plotting to normal, give a MODE 0 command.

FILL x,y,c,e - 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.

e - This parameter is NOT allowed in HIRES filling. In MULTICOLOR filling, this is the hundreds (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.

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

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 .....
```

F. SCREEN CONTROL COMMANDS

DUMP "filename"[,dev] - DUMP graphic display

This command saves the entire graphic display to tape or disk.

"filename" - this is the name of the graphic display which is to be dumped to tape or disk. For tape, it is valid to specify "" in which case there will be no filename.

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

GREAD "filename"[,dev] - READ the graphic display

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

"filename" - this is the name of the graphic display which is to be read from the tape or disk.

[,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 spcify ",1" since this is the default. To read the graphic display from disk, specify ",8" which is the device number of the disk drive.

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.

V. SPRITE GRAPHICS

A. INTRODUCTION TO SPRITES

In addition to all of the graphic capabilites described above, **ULTRABASIC-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. **ULTRABASIC-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). **ULTRABASIC-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.

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"000000000000000000000000"
```

A hires sprite picture can 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 different 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"0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F"
```

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"000000011111111111000000"
903   BIT"000000111111111111100000"
904   BIT"000001111110001111100000"
905   BIT"000001111101100111110000"
906   BIT"000011111011111111110000"
907   BIT"000011111011001111110000"
908   BIT"000001111100111111100000"
909   BIT"000001111111111111100000"
910   BIT"000001111111111111100000"
911   BIT"000000101111111110100000"
912   BIT"000000101111111101000000"
913   BIT"000000100111111001000000"
914   BIT"000000010011100100000000"
915   BIT"000000010011100100000000"
916   BIT"000000000100100100000000"
917   BIT"000000000100100100000000"
918   BIT"000000000011111000000000"
919   BIT"000000000011111000000000"
920   BIT"000000000011111000000000"
921   BIT"000000000001110000000000"
```

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,11111 - COPY sprite image

a - sprite slot number 1-15.

11111 - 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.

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).

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,sx,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 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)
-------------------------------------------------------------
```

D. SPRITE EXAMPLE

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

After **ULTRABASIC-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.

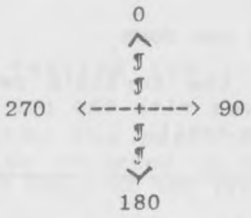
Now experiment with changing the pattern, x & y expand, color.

VI. TURTLE GRAPHICS

ULTRABASIC-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:



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 spirte 8 nor use slots 12-15 while you are using any of the TURTLE commands.

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 commands 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

TURNTO 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)

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 reutrns 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).

VII. GAME FEATURES

ULTRABASIC-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).

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.

B. TIMERS

ULTRABASIC-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 **HIRES** 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.

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

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, **ULTRABASIC-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.

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

The SOUND command is the easiet to use:

SOUND a,p,d - start a SOUND

The SOUND command lets you specify the generator (1-3), the pitch (1-127, see Appendix A 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)

If you want to change the characteristics of the 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)
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

The VOL command sets the control which apply to all generators:

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
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 **ULTRABASIC-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 B.

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, **ULTRABASIC-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, **ULTRABASIC-64** looks at the next value and time segment. The tone is shut off if the time in the literal is 00.

For example:

TheTDATA pattern "38090108FF0800100000" is processed as such --

```
3809  set the tone generator to pitch 38 hexadecimal
        (56 decimal)
0108 increment the pitch by 1 eachjiffy 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 spcified time. Another hex 80 turns it back on. If at the " at the end of the literal is found before the 00 time entry, **ULTRABASIC-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 is 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"8010FB01001005010010"
```

E010 sets the pitch to 96 decimal and bypass on repeat
FB01 lower the pitch by 5 for 1 jiffy
0010hold the pitch for 16 jiffies (10hexis 16 decimal)
0501 raise the pitch by 5 for 1 jiffy
0010 hold the pitch for 16 jiffies
" quote says to repeat, bypassing 8010. 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.

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"PPTTIITTIITT........" - 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 ¶
 segment1segment2segment3.....segmentN
```

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 segement if P is 80-FF).

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 turns it off.

a = generator - 1, 2, or 3
d = duration (60ths of a second)

IX. OTHER FEATURES

A. REPEATing commands

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

They work as follows:

The commands to be repeated are enclosed in brackets [ ]. The number of times the group of commands is to be repeated is indicated by a number and colon (:) following the left bracket ---

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

Repeats may be nested to a depth of 30. The **[ ]** may be on different lines. Any BASIC commands may be included.

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

n = number of times to repeat the commands .....

B. :EXIT from REPEAT

It is often desirable to leave a **REPEAT** loop before the repeat count has completed. The **:EXIT** command is used for this purpose.

An example:

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

The turtle command TURN 30 (meaning turn 30 degrees) will be 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 will be performed. Note that the ending bracket **:]** is on a separate line in order for the **IF** statement to work correctly.

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

The above example will not work correctly because the ending bracket is never encountered whenf the turtle's angular position is not greater than 180 because the **IF** statement is not satisfied. Therefore the ending bracket should be on a separate line when an **IF** statement is present within the **REPEAT** loop.

:EXIT] - exit from a REPEAT loop

C. HARDCOPY of graphics screen

ULTRABASIC-64 can reproduce the graphics screen onto your COMMODORE, EPSON MX-80, FX-80 or RX-80, Gemini or OKIDATA Microline printer. The entire screen is sent to the printer. A full screen requires 4 1/2 minutes to reproduce on the COMMODORE 1515 or 1525 printer and about 1 1/2 minutes on an EPSON FX-80 printer.

HARD d,s - hardcopy to printer

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)
s - size 0 = 1X display, prints horizontally
1 = 2X display, prints vertically

X. PROGRAMMING NOTES

A. GENERAL INFORMATION

The **ULTRABASIC-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 COMMMODORE 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 color to bypass any graphic command following the **REM.**

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

Because **ULTRABASIC-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 **ULTRABASIC-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 **ULTRABASIC-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 .7. 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 .7Y + the function value (Z) is less than the prior point, it is not shown, as it would be hidden from view. **UBDEMO** has a sample 3D plotting routine.

C. MEMORY ORGANIZTION

DECIMAL	HEX	USAGE
2048-7424	$0800-$2C00	ULTRABASIC-64 INTERPRETER
7552-32767	$2C00-$7FFF	your BASIC program area
32768-33727	$8000-$83BF	SPRITE PATTERN AREA
33728-33791	$83C0-$83FF	GRAPHICS CHIP REGISTER SAVE AREA
33792-34815	$8400-$87FF	GRAPHIC SCREEN POINTERS
34816-35839	$8800-$8BFF	GRAPHIC SCREEN COLOR SAVE
35840-36863	$8C00-$8FFF	AREA FOR BASIC SCREEN
36894-40959	$9000-$9FFF	CHARACTER ROM
40960-49151	$A000-$BFFF	BIT MAPPING AREA
55296-56319	$D800-$DBFF	COLOR MEMORY DYNAMIC AREA

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 scren is not saved. Therefore if you write something using a cursor color, it will come back standard blue 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 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 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.

XI. ERRORS

As **ULTRABASIC-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.

XIII. COMMAND SUMMARY

HIRES/MULTICOLOR COMMANDS

```
HIRES a,b              - setup high resolution screen
MULTI a,b              - setup multicolor screen
TIC a,b,c              - tic mark the screen edges
DOT x,y,c              - plot a single point
DRAW x1,y1,x2,y2,c     - plot a line
BOX x1,y1,x2,y2,c      - plot a box (rectangle)
CIRCLE x,y,r,c         - plot a circle
CHAR g,x,y,c,"string"  - display graphic text characters
BLOCK x1,y1,x2,y2,c    - block fill
MODE a                 - set plot mode
FILL x,y,c[,a]         - fill a region
PIXEL(x,y)             - function which tests for points
                           on/off
```

SCREEN CONTROL COMMANDS

```
DUMP "filename"[,dev] - save graphic display to tape/disk
GREAD "filename"[,dev]- restore graphic display from
                           tape/disk
NORM                  - switch screen to normal text screen
GRAPH                 - switch screen to graphic display
```

SPRITE COMMANDS

```
COPY a,llllll          - copy sprite picture from line
SPRITE n,s,m,p,x,y,c1,c2,c3 - turn on sprite
OFF n                  - turn off sprite
PLACE n,sx,xy          - move sprite
ROTATE d,m,s           - rotate sprite
```

SPRITE PICTURE DEFINITIONS

```
BIT"bbbbbbbbbbbbbbbbbbbbbbbb"   - hires definition
                                  requires 21 lines
COLORS"cccccccccccc"             - multicolor definition
                                  requires 21 lines
HEX"hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh" - alternate
                                  hires definition requires
                                  3 lines.
SDATA"val,val,val,val,val,..."  - alternate data definition
                                      requires 63 values
```

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
TURNTO 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 collison 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	- setupgenerator control
VOL a,b,c,d,e	- common generator control
SET a,llll	- set generator pattern
TDATA"pptt1itt1itt...."	- define TUNE pattern
TUNE a,d	- play complex pattern

OTHER COMMANDS

[N: :]	- repeat commands within brackets
:EXIT	- leave repeat
HARD d	- hard copy to printer

APPENDIX A - NOTE EQUIVALENTS

NOTE	DECIMAL VALUE	HEX VALUE
G3	12	0C
A3	14	0E
B3	16	10
C4	17	11
D4	19	13
E4	21	15
F4	22	16
G4	25	19
A4	28	1C
B4	32	20
C5	34	22
D5	38	26
E5	43	2B
F5	45	2D
G5	51	33
A5	57	39
B5	64	40
C6	68	44
D6	76	4C
E6	86	56
F6	91	5B
G6	102	66
A6	115	73

APPENDIX B

A short lesson in hexadecimal numbers

Computers represent data in binary format. It 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:

$$1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0$$
$$= 16 + 0 + 0 + 0 + 1$$
$$= 17$$

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.

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]
```

APPENDIX C - COLOR NUMBERS

COLOR NUMBER	COLOR
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.

APPENDIX D - PRINTER/INTERFACE SUPPORT

ULTRABASIC-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. To support these printers you should load and run **UBCBM** at startup.

ULTRABASIC-64 supports the Epson MX-80 and MX-100 with Graftrax, the FX-80, FX-100 and RX-80, and the Gemini 10 and 15 printers. To support these printers you should load and run **UBEPSON** at startup.

ULTRABASIC-64 also supports the following OKIDATA printers:

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

To support these printers you should LOAD and RUN **UBOKI** at startup.

For either **UBEPSON** or **UBOKI** you must have one of the following parallel printer interfaces:

MANUFACTURER	MODEL	SECONDARY ADDRESS	TRANSLATE (Y/N)	SWITCHES ON
CARDCO Wichita, KS	CARD?	5	N	
ECX, Inc. Walnut Creek, CA	C-6401	0	N	*
MICROWORLD	MW-302	0	Y	3,4
ELECTRONIX Lakewood, CO		0	N	3
MSD, Inc. Dallas, TX	CPI	0	N	1,3,5

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

The **SECONDARY ADDRESS** and **ASCII TRANSLATE?** prompts should be answered as per the above table.

Abacus Software

P.O. Box 7211, Grand Rapids, MI 49510 616/241-5510