Byte Paint™

By Phil Montoya

dedicated to my wife, Annette

-Phil Montoya

Apple Computer, Inc., makes no warranties, either expressed or implied, regarding the enclosed computer software package, its merchantability, or its fitness for any particular purpose.

ProDOS and Applesoft are registered trademarks of Apple Computer, Inc.

Limited Warranty - Subject to the below stated limitations, Byte Works hereby warrants that the program contained in this unit will load and run on the standard manufacturer's configurations for the computer listed for a period of ninety (90) days from the date of purchase. Except for such warranty, this product is supplied on an "as is" basis without warranty as to merchantability or its fitness for any particular purpose. The limits of warranty extend only to the original purchaser.

Neither Byte Works, nor the author of this program are liable or responsible to the purchaser and/or user for loss or damages caused, or alleged to be caused, directly or indirectly by this software and its attendant documentation, including (but not limited to) interruption of service, loss of business, or anticipatory profits.

To obtain the warranty offered, the enclosed purchaser registration card must be completed and returned to The Byte Works within ten days of purchase.

Important Notice - This is a fully copyrighted work and as such is protected under the copyright laws of the United States of America. According to these laws, consumers of copywritten material may make copies for their personal use only. Duplication for any other purposes whatsoever would constitute infringement of copyright laws and the offender would be liable to civil damages of up to $50,000 in addition to actual damages, plus criminal penalties of up to one year imprisonment and/or a $10,000 fine.

This product is sold for use on a single computer at a single location. Contact the publisher for information regarding licensing for use at multiple-workstation or multiple-computer installations.

ORCA/M Libraries - This program makes use of libraries contained in ORCA/M. These libraries are fully copyrighted, and are used here by permission.

Use of the Graphics Library in Other Programs - It is the policy of the Byte Works to License the graphics libraries to purchasers of Byte Paint

free of charge. A copy of the program, along with any documentation, is required at the time of licensing, and the documentation must give credit for using the libraries. For details, please contact the Byte Works.

I would like to express many thanks to Mike Westerfield. First of all I would like to thank him for giving me the chance to do Byte Paint, secondly, I am grateful for all the guidance, supervision and enthusiasm he provided throughout this project, and finally, I am grateful for the many lessons in structured assembly language programming Mike has so patiently dictated.

The graphics routines were adapted from the graphics libraries provided with ProDOS ORCA/M 4.0. It should be noted that many changes were made to optimize the routines for our particular purpose.

TABLE OF CONTENTS

Introduction to Byte Paint 11

The Byte Paint Drawing Program 11
- Selecting Colors 12
- The Icon Bar 12
- Drawing on the Full Screen 13
- The Menu Bar 14
- Help 14
- Disk 14
- Edit Aids 15

The Byte Paint Shape Editor 16

Using Byte Paint Graphics from BASIC 18
- Installing the Graphics Package 18
- The Graphics Screens 19
- The Viewport 20
- Pushing and Poping the Screen 20
- Color Mapping 20
- Shape Tables 21
- Turtle Graphics 21
- Page 3 21

Graphics Commands 22
- &BOX - Draw a Box 24
- &CHGR - Initialize Color Graphics 25
- &CMAPOFF - Color Map Off 26
- &CMAPON - Color Map On 27
- &COLOR= - Set the Pen Color 28
- &COPY - Copy Shape 30
- &CUSR - Draw System Cursor 32
- &CUT - Cut a Shape From the Screen 35
- &DHGR - Initialize Double High Res Graphics 36
- &DRAWBLOCK - Draw a Block on the Screen 37
- &DRAWTO - Draw a Line 38
- &ELLIPSE - Draw an Ellipse 39
- &FILL - Fill the Window 40
- &FIND - Find the Cursor Position 41
- &FLOOD - Flood Fill 42
- &GROUT - Graphics Output of Text 43
- &HOME - Clear the Screen 45

&INPUT - Select the Mouse, Joystick or Keyboard 46
&INVERSE - Invert the Screen 47
&MOVETO - Move the Pen 48
&NORMAL - Restore Output Hook 49
&PDL - Read the Paddle 50
&PLOT - Plot a Point 51
&POP - Pop the Screen 52
&PUSH - Push the Screen 53
&READ - Read a Pixel 54
&SETCMAP - Set Color Map 55
&SETCUSR - Create Cursor Mask 57
&TEXT - View the Text Screen 58
&TRTLDRAW - Turtlegraphics Draw Command 59
&TRTLMOVE - Turtlegraphics Move Command 60
&TRNTO - Turtlegraphics Turn To Command 61
&TURN - Turtlegraphics Turn Command 62
&VIEW - Set the Viewport 63

Introduction to Byte Paint

Byte Paint is really three programs in one. First, it is a full color drawing program that will use a mouse, joystick, or the keyboard to help you create full size color drawings on you Apple //. The new double high resolution graphics page is used, so you can draw with sixteen vibrant colors.

Next, Byte Paint includes a sophisticated editor for editing character fonts and shape tables. Byte Paint shape tables are not the same as Applesoft shape tables - with Byte Paint, shapes can be multicolored.

Byte Paint is also a complete graphics language that you can use from your own BASIC programs. From BASIC, you can draw on either the color or black and white double high resolution screens. Since Byte Paint hides itself in the alternate 64K of your computer, your BASIC programs can be as large as ever. The graphics language is very extensive, including turtle graphics, automatic clipping windows, push and pop for doing pull down menus and windows, color mapping, and much more. Complete descriptions are given in the programming guide, along with examples.

There are also two very extensive examples of the graphics package. The drawing program and shape editor are written in uncompiled Applesoft BASIC, using the graphics package. If you haven't seen the program running, you simply won't believe how fast it is, and yet, you can list it at any time, and make changes, if you like.

So, prepare yourself for a pleasant change to graphics on the Apple // - here is a program that lets you draw, and the things that you draw can be used in your own BAS programs!

The Byte Paint Drawing Program

The drawing program is based on icons, pull down menus and windows. This makes it a very easy program to learn to use, so this section of the manual is relatively short. If you have used programs like MousePaint, simply skimming the documentation will tell you all that you need to know.

To use the drawing program, simply insert the Byte Paint disk, boot your computer and select the BYTEPAINT option. Next, you will be asked if you would like to use a mouse, joystick, or the keyboard. All will work, but we should point out that a joystick works better than the keyboard, and the

mouse works best of all. Select the drawing tool of your choice by typing an M, J or K, as indicated.

After a few moments, your screen will show the Byte Paint menu bar, icon bar, and color selection bar. A small arrow will be on the screen, and will respond to the mouse, joystick, or the arrow keys on the keyboard, depending on which drawing device you selected. Take a moment-to move the arrow around, getting a feel for the program.

Throughout the rest of this section, we will refer to "moving the mouse" and "clicking" or "holding down" the mouse button. If you are using a joystick, moving the mouse is equivalent to using the joystick to move the cursor on the screen. The keyboard arrow keys are used to move the cursor if you select the keyboard option. If you are using the keyboard, note that holding the closed apple key down while using the arrow keys causes the cursor to move faster across the screen. The open apple key on the keyboard is equivalent to the mouse button, as is game paddle button zero for the joystick driver.

Selecting Colors

If you hold the mouse button down as you move it, you will draw a black line across the screen. Only the drawing window will be effected. Along the bottom of the screen, you will see a series of seventeen colored boxes. The one on the far left is larger than the others; it indicates the current color, and starts out black.

Move the arrow cursor down to one of the colored boxes and click the mouse button. The color of the left-hand box will change to the color of the box that the arrow was on when you clicked the mouse button. Now try drawing again - you will be drawing a colored line!

All of the drawing commands will use the color you select using the color bar.

The Icon Bar

Along the left-hand edge of the screen is the icon bar. It contains your drawing tool box. To change tools, simply move the cursor over the tool you want and click the mouse button. The tool will be surrounded in gray, indicating your choice.

There are eight tools in all. The one you have been using so far is the pencil, shown at the top of the screen. It draws a thin line across the screen whenever the cursor is moved while the mouse button is held down.

Just below the pencil is the eraser. If you never make mistakes, you won't need this one. As you move it across the drawing area, it erases the picture (assuming that the mouse button is held down). Since the screen starts out white, it always leaves a white screen behind when you erase.

Next is a spray can. The spray can is a very fun tool, which sprays a spotted pattern. Moving the mouse very quickly will give a very fine mist, while slowly moving across a small area will eventually fill it in. Don't worry about getting carried away - it will never drip.

The last of the conventional painting tools is the paintbrush. It works like a wide pencil, filling in a large area as you move the mouse. Like the spray can, moving too fast can cause it to skip. Later, we will look at ways to change the shape of the paint brush.

You can type information onto the screen using the letter icon, which shows up as a large Z. After selecting it, your cursor will look like an oversized 1. Move it to where you want your text information to be and click the mouse button. This locks the cursor in place it stays there until you click the mouse button again. As long as you leave it locked in place, anything you type will show up on the screen. Later, we will see how to choose character sets.

The slanted line is a drawing tool that lets you draw straight lines. Move the cursor to a location in the drawing area and hold down the mouse button. While holding it down, move the mouse until the line is as long as you want it, then let up on the mouse button. A straight, colored line will appear on the screen.

The last two tools let you draw rectangles and ellipses on the screen. Like the line drawing tool, you start by selecting one point, then hold the button down while you expand the shape out to its full size. Letting up on the button completes the shape.

Drawing on the Full Screen

So far, you have only been able to draw on a part of the screen. The color bar, menu and icon bar are using part of your drawing area.

The solution to this problem is found with the ESC key. When you type it, all of the screen covered by program options goes away, and you can draw on the full screen. To switch instruments or change the pen color, you must switch back, again with the ESC key.

The Menu Bar

Across the top of the screen is a menu bar. Each of the words on the menu bar represent a list of drawing aids that you can use. To get at this list, move the mouse up to the word that you want to see a menu for, push the mouse button, and hold it down.

To activate one of the options, hold the mouse button down and move the cursor down until the option you want is shown in inverse, then release the button. Releasing the button without an option active will cause the menu to disappear, and no action to be taken.

Help

The first menu is simply a help menu. Under it you will find several entries, each of which causes a help screen to appear with a brief comment about the topic.

Disk

The disk menu is used to load and save the drawing screen or shape tables, to catalog the disk, or delete a file. In each case, a file name is asked for. In the case of the CATALOG command, the name is the name of the directory you want to see cataloged, and can be blank if the default directory is wanted. In all other cases, a specific file name is required. It can be a simple file name or a full or partial path name.

By the way, you don't need to know much about ProDOS to use this program, but to use the disk options, you do need to know the rules for file names and path names under ProDOS. If you don't know about ProDOS file names, consult your ProDOS reference manuals for help.

The functions of the CATALOG and DELETE options should be fairly familiar: the first shows you the files on a directory, while the second destroys one of those files. LOAD PIC and SAVE PIC are also fairly obvious, once you know that PIC is an abbreviation for picture: these commands save a picture that you have drawn to disk, or load one from disk for further editing.

LOAD TABLE and SAVE TABLE are a bit more complex. Byte Paint makes extensive use of what it calls shape tables. These are very different from the shape tables used in Applesoft BASIC programs. In Byte Paint, a shape table is a collection of up to 256 different shapes. These can be created using the shape editor or the drawing program. The first 128 of the shapes are generally used as the ASCII character set. This means that loading a shape table will cause the drawing program to change character sets, and that is the principle reason for using the LOAD TABLE command. In the directory /BYTEPAINT/FONTS on the Byte Paint disk you will find a series of shape tables that implement various character sets when used with the character icon drawing tool. Use the CATALOG command to see what they are, and LOAD TABLE to use them.

Another use of shape tables is to add shapes to your BASIC programs. This will be discussed more fully in the next section. Once a shape table is created, SAVE TABLE places it on disk, where it can be found later.

Edit Aids

Under the edit aids menu, you will find a variety of commands to do special things to your drawing. The first three options, CUT, PASTE, and DELETE, are used to copy and move rectangular areas of the screen. When you select CUT or DELETE your cursor will change to a small cross hair. You can now select a rectangular area of the screen the same way that you draw a box with the box drawing tool: first select one corner of the area, then hold the mouse button down while you move the cursor to the other corner, then release the mouse button. If you are using DELETE, the area marked disappears. CUT doesn't make any change to the drawing. Either operation copies the area into the shape table as shape 255, which is popped back onto the screen using the PASTE command. PASTE will place the area back onto the screen as many times and in as many places as you like. Once PASTE is selected, you get an arrow cursor. Place the arrow where you want the top left of the area to be placed, then click the mouse button.

CUT and DELETE can also be used to build a shape table for use in your BASIC programs, or for use as a clip art set in later drawing sessions. Once you CUT or DELETE an area, you can SAVE it to a different area of the shape table. SAVE asks for the number of a shape, then moves the last shape that you cut out into the shape table at the location you give it. To move a shape from somewhere in the shape table into shape 255, use GET. It will also ask for a shape number, this time the one you want to fetch.

Once it is moved into shape 255, you can use PASTE to place it on the screen.

If you place a shape into one of the shapes numbered 0 to 127, there is a second way to get it out to the screen. In the alpha mode, typing the key that corresponds to the shape will cause it to be printed. For example, the ASCII character code for an "A" character is 65 ($41). Try cutting an area from the screen and saving it to shape number 65 ($41). Now use the Z icon to type something on the screen. Be sure that you use a capitol A - you will see your new shape instead of the character!

The last option in the edit aids menu is the only one that has nothing to do with the shape table. The SET BRUSH command lets you choose among sixteen different brush shapes. Once selected, the brush shape will be used by the brush icon. After you choose a brush from those drawn on the screen, move the cursor to the small box at the upper left corner of the brush window and click the button. This will cause the brushes to disappear and allow you to start painting with the brush you have chosen.

The Byte Paint Shape Editor

The drawing program that we just looked at is a very good program for drawing full size pictures, but it is not easy to use to edit character sets. There are also some unique characteristics about shapes when they are used as characters that must be kept in mind, For those reasons, Byte Paint includes a special program for editing character sets. It is also good at editing shapes.

First, let's take a look at how colors are used by Byte Paint. When you are using the Z icon to write characters on the screen, Byte Paint makes use of color mapping so that your characters show up in the current pen color, and so that the background shows through. Color number five (counting from zero, with black as zero) is a gray color. If you look at the color bar, you will see that color number ten is also gray. Byte Paint uses the first gray (number five) as an invisible color, so that any part of a character defined as color five will not be drawn when you type. Similarly, black is used as a mapped color. Any point in a character that is black in the shape editor will be drawn as the current pen color on the screen when Byte Paint uses the table. All of the other colors show up on the Byte Paint screen the same way that they do in the shape editor. This gives you the ability to define full color character sets, with the limitation that color five cannot be used, and the pen color must be black when the character is drawn for black to be

available as a specific color. The Rainbow font provided with Byte Paint gives, ahem, a graphic illustration of this.

To use the shape editor, simply select the shape editor from the main menu and choose your drawing tool. You will see a familiar looking menu bar across the top, and the color bar across the bottom. These are used just like they were used in the drawing program. The icon bar, on the left, has changed a bit, but still serves the purpose of expanding the number of things you can do to a character.

To get started, pull down the options menu. There you will find some of the same commands that were available in the drawing program. DELETE and CATALOG are used to see what is on the disk, and to remove existing files. LOAD TABLE and SAVE TABLE let you load an existing shape table, or save one that you have edited. Shape tables are loaded from and saved to the /BYTEPAINT/FONTS directory.

Two other options are available there, so let's look at them now. The first is SET SIZE. This option lets you set the size of the character that you are looking at, expanding or contracting it. After selecting that option, you will be asked for a height (in pixels) and width (also in pixels). The character will be expanded by adding rows to the bottom and columns on the right of the existing character, or contracted by deleting columns from the right or rows from the bottom. Any added rows or columns will use color number five, the invisible gray.

The last menu option is COPY SHAPE. It will ask for the number of an existing shape, and will copy that shape onto the one you are editing.

Once a table is loaded, you will want to start editing the character set. To get a character to edit, simply type the appropriate keyboard character. The keyboard will normally use the standard ASCII character set, so that a 65 will be an A. If you want to edit shapes with numbers higher than 128, you can still use the font editor; simply hold down the closed apple key when you type the character. The font editor will reverse the high bit, which effectively adds 128 to the ASCII character set number.

By the way, you can type control characters this way. For example, the RETURN key is a control D, which will edit shape number 13.

The icon bar shows some editing options which effect the entire shape. The arrows will shift the shape one pixel in the direction of the arrow, filling the

new row or column with the invisible gray and destroying the row or column shifted off of the edit area. The double arrows will flip the shape.

Finally, use the mouse (or keyboard or joystick) to actually edit the shape. After selecting the appropriate color (black for the current pen color), simply move the cursor to the pixel you want to change and click the mouse button. This must be done on the expanded window, not on the actual size shape.

Using Byte Paint Graphics from BASIC

If you have been trying out the programs, it may be necessary to remind you that they were written in BASIC. Despite their speed, both were really written entirely in Applesoft BASIC, and they have not been compiled.

The way that we did this was to start by implementing a very powerful graphics package using Amper commands. As it turns out, Applesoft can be extended by writing subroutines in assembly language, then using a special hook to cause Applesoft to call the assembly language subroutines. Every time Applesoft finds an "&" character in the program, it calls the assembly language routines, which read the rest of the line and execute a new command. The Byte Paint graphics commands are implemented this way, so you can use them from your own BASIC programs.

Installing the Graphics Package

There are two ways to install the amper package that lets you use the Byte Paint graphics language from BASIC. The easiest way is to boot the program disk and select the QUIT option from the initialization menu, After typing NEW to clear out the old program, you can begin typing in and running your BASIC program. One important note is in order: the Byte Paint initialization routines change some zero page locations so that BASIC programs start at $6000 instead of the normal $800. This gives more room for the BASIC program, since it will not run into the graphics page. It also gives a good spot to load shape tables, since the memory from $800 to $1FFF (decimal 2048 to 8191) and $4000 to $5FFF (decimal 16384 to 24575) is free.

The other way to install the amper package is by running a binary program called INIT.PROGRAM. From the BASIC command line, just type

```
BRUN INIT.PROGRAM
```

INIT.PROGRAM will need to access another binary file, called ASM.CODE. For that reason, ASM.CODE must be on the default prefix when you run INIT.PROGRAM. You can do all of this from a BASIC program, if the BASIC program is short. If your program is fairly long, you will need to use a special boot program, like the one below. Using this method, your BASIC programs will start at $800 (decimal 2048) unless you have already changed this yourself.

```
100 REM SET UP BYTE PAINT GRAPHICS
110 REM AND CALL ANOTHER PROGRAM
120 D$ = CHR$(4)
130 PRINT D$;"BRUN INIT.PROGRAM"
140 PRINT D$;"RUN MYPROG"
```

The Graphics Screens

The Byte Paint graphics routines draw on the double high resolution graphics screen. Double high resolution graphics doubles the number of pixels that are on the high resolution graphics screen by putting twice as many pixels on each line. The high resolution graphics page accessed by Applesoft has 280 points on each line; with Byte Paint, you get 560 points per line.

Because of the unique way that the Apple // computer sends out its video signal to the monitor, the graphics screen looks very different on a color monitor or television set than it does when displayed on a one color monitor. On a one color monitor, each of the 560 points on a line is distinct. The better the monitor, the more each of the points looks like a little square box. On a color monitor or television set, you can only see 140 distinct points, but each of those points can appear in any one of sixteen colors. There is absolutely no restriction about which colors can appear next to one another. On the other hand, you should keep in mind that most color monitors and all television sets will show some bleed. Thus, even though you can put a white dot next to a brown one, you get something that looks more like a yellow dot beside a black one.

Because of the big difference between what the screen looks like on a monitor and what it looks like on a color display, Byte Paint gives two different ways of drawing on the screen. The one used in the drawing program addressed the screen as if it were a color screen, 192 dots high and 140 dots wide, with each dot having sixteen colors. Another method lets you address the screen as a black and white screen with 560 dots on each of

the 192 rows. The commands for the two screens are the same; you choose which you will use when you initialize the graphics screen. The &CHGR command sets up the color screen, while the &DHGR command sets up the black and white screen.

As you would expect, the X coordinates on the screen start at zero on the left edge, and increase as you move to the right. The Y coordinate works differently than it does in Applesoft, In Applesoft, the top of the screen was 0 and the bottom was 191. Byte Paint uses a more standard way of addressing the screen, placing 0 at the bottom of the screen, and 191 at the top. This makes the screen coordinates work just like the first quadrant of a Cartesian coordinate system, which is the way most expensive graphics computers work.

The Viewport

The graphics commands contain several features that you may not have seen before unless you have programmed in on of the more advanced languages, like Pascal or Logo. The viewport is one of these. The viewport is a rectangular area on the screen. It is used to protect the screen from changes, without needing to carefully check the positions you are about to plot to. By default, the viewport is set to the entire screen. If you change it to a smaller area using the &VIEW command, you cannot change the area outside of the viewport. The graphics commands still function, and you don't get an error, but only the part of the screen in the viewport is changed.

Pushing and Poping the Screen

In order to make pull down menus and windows work, you need a way to save part of the screen. Byte Paint's &PUSH command will do that. It can save any rectangular area on the screen, after which &POP can restore it. These commands work like a stack, so you can push more than one area of the screen, but you must restore them in the reverse order of the way they were stored. By the way, the memory used to push and pop large areas can add tip to several kilobytes very quickly. Byte Paint uses the alternate 64K for its stack, so the memory used is not taken away from your BASIC program.

Color Mapping

One of the most powerful features of the Byte Paint graphics language is the ability to define color maps. A color map lets you define exactly what will happen when a point is plotted using any of the drawing commands. With

color mapping enabled, you can set the color map so that the color drawn depends not only on the current pen color, but also on the color of the screen where the point will be drawn. The commands &CMAPON, &CMAPOFF and &SETCMAP create this ability, and fully describe how it works.

Shape Tables

Byte Paint allows you to define shape tables, which work with the &DRAWBLOCK command and the &GROUT command. (&GROUT lets you tell Byte Paint to draw text characters on the graphics screen,) These shape tables work quite differently from the shape tables in standard Applesoft. The &CUT command is used to define the shape table by placing areas of the screen into the table.' &COPY can then be used to move them around.

Before using a shape table, you must decide where in memory it will be located, and how long it is. The graphics commands will give an error if you try to use shapes without doing this. Once you select an area of memory and protect it with appropriate settings of LOMEM and HIMEM, you should clear the memory to all zeros with a loop of POKE statements. Next, place the address of the first byte in the area at $3C2 (decimal 962), then place the address of the last byte in the table area at $3C4 (964). After that, you can create shapes, or even use BLOAD and BSAVE to load and save shape tables from and to disk.

Turtle Graphics

Turtle graphics uses the concept of a turtle placed on the screen which will move across the screen drawing lines. You can tell the turtle to turn or move, and can tell it to draw a line or not to. The Byte Paint commands &TRTLMOVE, &TRTLDRAW, &TRNTO and &TURN implement turtle graphics for double high resolution graphics.

Page 3

The Byte Paint graphics package hides itself in the alternate 64K of your 128K Apple. By doing this, it doesn't use up any of the memory that would normally be available for BASIC programs. Byte Paint uses page 3 to communicate between your running BASIC program and the graphics subroutines in the alternate memory bank, It also places certain values in page 3, so that you can use the PEEK command from BASIC to read them.

The table below summarizes the use made of page 3. Individual command descriptions tell what use is made of them, and the sample programs give examples of how to use them.

$300 - $3BC	768 - 956	Code used to switch control between BASIC and Byte Paint.
$3BD	957	BUTTON - 0 if the button is not pressed, 1 if it is.
$3BE - $3BF	958 - 959	ABSX - set by &PDL to the X position of the mouse, joystick or keyboard cursor.
$3CO - $3Cl	960 - 961	ABSY - set by &PDL to the Y position of the mouse, joystick or keyboard cursor.
$3C2 - $3C3	962 - 963	SHAPEADR - The address of the first byte in the shape table.
$3C4 - $3C5	964 - 965	SHAPEEND - The address of the last byte available to the shape table.
$3CA - $3CB	970 - 971	PENX - X location of the graphics pen; set by &FIND.
$3CC - $3CD	972 - 973	PENY - Y location of the graphics pen; set by &FIND.
$3C9	969	POINT - The color of the point last read by &READ.
$3C6 - $3C7	966 - 967	ANGLE - Current angle of the turtle.
$3C8	968	INVISIBLE - Flag that indicates if color 5 is to be an invisible color.

Graphics Commands

The rest of this section explains the commands that are available with Byte Paint.

The command descriptions have four parts. The first part lists the parameters needed by the command, in the order that they appear in the BASIC program. When using the command in a program, multiple parameters are separated by commas. The next section tells what the command does. If the command can cause an error (other that a syntax error), the third section tells you what those errors are. The last section is a short sample program showing how the command is used.

As you are no doubt aware, Applesoft puts blanks wherever it pleases, and takes out any blanks that you might put in to make the program easier to read. Some of the names for the graphics commands contain Applesoft

keywords, This does not effect the running of the program, but does cause blanks to occasionally show up inside of the graphics command.

&BOX - Draw a Box

Parameters:

LEFT - left edge of the box
RIGHT - right edge of the box
TOP - top edge of the box
BOTTOM - bottom edge of the box

Result:

A box is drawn on the graphics screen. The color of the box will match the current pen color, and its size is determined by the four parameters. The box is not filled in.

Errors:

Illegal Quantity Error

Ex amp l e:

```
10 L = 139 / 2 - 2:R = 139 / 2 + 2
20 B = 191 / 2 - 2:T = 191 / 2 + 2
30  & C HGR
40  & COLOR= 15
50  FOR J = 0 TO 33
60  & BOXL,R,B,T
70 L = L - 2:R = R + 2:B = B 2:T = T + 2
80 NEXT J
90 INPUT A$: HOME : & TEXT LIST
100 & TEXT
```

&CHGR - Initialize Color Graphics

Parameters:

none

Result:

This command sets up the screen for color double high resolution graphics. It is used just like the HGR command in Applesoft. The viewport is set to the full screen, which means that the top is set to 191, the bottom to 0, the left to 0, and the right to 139. The pen color is set to black, and the push and pop stacks are initialized.

Errors:

none

Example:

See &BOX for an example.

&CMAPOFF - Color Map Off

Parameters:

none

Result:

Color mapping is turned off. See &CMAPON for a description of color mapping.

Errors:

none

Example:

SEE &SETCMAP

&CMAPON - Color Map On

Parameters:

none

Result:

Color mapping is turned on. Color mapping is a very powerful feature of the graphics package that lets you define what will happen when one color is drawn over a point that used to be a different color. For example, if you need to make sure that a red line is seen, but you do not know what color the background is, you could use color mapping to make sure that when a red line is drawn on a red background, it shows up as white.

This command simply turns color mapping on. By default, all of the colors are set up so that they will show up as the pen color, no matter what the background color is. In plain English, that means things work just like they did before color mapping was turned on, The &SETCMAP command is used to change the color map; it is described later.

By the way, color mapping works by reading a point from the graphics screen, deciding what color to draw, and then drawing a point. Deciding what color to draw doesn't take much time, but reading a point from the screen takes about as much time as plotting one. For that reason, all of the graphics commands work about half as fast with color mapping turned on.

Errors:

None

Example:

SEE &SETCMAP

&COLOR= - Set the Pen Color

Parameters:

NUM1 - Color Number

Result:

This command is used to set the color that all of the drawing commands will use. The parameter is the number of the color to set the pen to. The numbers range from zero to fifteen, and match the colors from the low resolution graphics screen.

color number	color name
0	black
I	magenta
2	dark blue
3	purple
4	dark green
5	gray
6	medium blue
7	light blue
8	brown
9	orange
10	gray
11	pink
12	green
13	yellow
14	aqua
15	white

Errors:

Illegal Quantity Error

Example:

```
10 X = 9
20 L = O:R = X
30 COLR = 0
40 & C HGR
50 & COLOR= COLR
60 & FILL
70 FOR COLR = I TO 15
80 & COLOR= COLR
90 & VIEWL,R,0,191
100 & FILL
110 L = L + X:R = R + X
120 NEXT COLR
130 INPUT A$: HOME : & TEXT : LIST
```

&COPY - Copy Shape

Parameters:

FROM - location to copy from
TO - location to copy to

Result:

The graphics routines can keep track of 256 different shapes at a time. Each of these shapes has a number; the numbers start at 0 and go up. to 255. &COPY takes the shape whose number is specified by the FROM parameter and copies it into the shape specified by the TO parameter. If either parameter is not in that range, an illegal quantity error results. If the &COPY results in overflowing the shape table area, you will get an out of memory error.

Errors:

Illegal Quantity Error
Out of Memory

Example:

```
10 BL = 962:BH = 964:FL = 968
20 & C HGR
30 & COLOR= 15: & FILL
40 REM USE MACHINE CODE TO CLEAR FONT AREA
50 PRINT CHR$ (4)"-ASM.CODE/CLEAR.CODE"
60 REM LOAD A FONT TABLE
70 PRINT CHR$ (4)"BLOAD FONTS/OUTLINE.FONT"
80 REM SET THE BEGINNING ADDRESS OF THE TABLE
90 POKE BL,O: POKE BL + 1,64
100 REM SET THE ENDING ADDRESS OF THE TABLE
110 POKE BH,O: POKE BH + 1,96
120 REM USING THE INVISIBLE COLOR
130 POKE 968,1
140 X = O:Y 180
150 FOR J 65 TO 81
160 & DRAW BLOCKX,Y,J
170 X = X + 8
180 NEXT J
190 REM NOT USING THE INVISIBLE COLOR
200 POKE 968,0
210 X = O:Y 165
220 FOR J 65 TO 81
230 & DRAW BLOCKX,Y,J
240 X = X + 8
250 NEXT J
260 REM
270 REM EXAMPLE OF COPY
280 REM
290 & COPY66,65: REM COPY "B" INTO "All
300 & DRAW BLOCK60,150,65: & DRAW BLOCK75,150,66
310 REM
320 REM EXAMPLE OF CUT
330 REM
340 & CUTO,70,166,180
350 & DRAW BLOCK35,135,255
360 REM CAN COPY SHAPES OF DIFFERENT SIZES
370 & COPY255,65
380 & DRAW BLOCK35,120,65
390 INPUT A$: HOME : & TEXT
```

&CUSR - Draw System Cursor

Parameters:

X - X coordinate of the top left corner of the cursor
Y - Y coordinate of the top left corner of the cursor

Result:

The graphics routines have a special cursor that they can draw very quickly, since it is maintained in a preshifted table. The cursor is always drawn in the EOR mode, so that it will show up on any background. To erase the cursor, simply draw it in the same position a second time. X and Y must be in the legal ranges for a coordinate on the screen, which means that Y must be in the range 0-191, and X can range from 0-139 on the color screen or 0-559 on the black and white screen. See &SETCUSR for information on how to set the cursor shape. If it is not set beforehand, the command will not do anything.

&CUSR is one of the few commands that ignores the setting of the viewport. It will draw the cursor to any screen location.

Errors:

Illegal Quantity Error

Example:

```
10 REM THIS PROGRAM MUST BE ENTERED AFTER EXECUTING
20 REM STARTUP. THE REASON FOR THIS IS BECAUSE THE
30 REM PROGRAM USES A SHAPE TABLE WHICH IS LOAD[D
40 REM A $800, THEREFORE THE PROGRAM MUST BE
50 REM RELOCATED TO $6000 BY THE STARTUP PROGRAM.
60 D$ = CHR$ (4)
70 PRINT D$"BLOAD FONTS/PROG. FONT"
80 BL = 962: REM LOCATION OF BLOCK ADDRESS
90 BH = 964: REM LOCATION OF END OF BLOCK TABLE ADDRESS
100 IN = 2: REM INPUT DRIVER "MOUSE"
110 BT = 957: REM LOCATION OF BUTTON STATUS ON PADDLE READ
120 CR = 130: REM CURSOR SHAPE NUMBER
130 DX = 958: REM LOCATION OF X VALUE ON PADDLE READ
140 DY = 960: REM LOCATION OF Y VALUE ON PADDLE READ
150 CX = 22:CY = 170
160 GOSUB 330: REM INITIALIZE SCREEN
170 REM MOVE THE CURSOR
180 & SETC USR CR
190 & C USR CX,CY
200 REM READ THE INPUT DEVICE
210 & PDL
220 REM CHECK FOR A CHANGE IN EITHER X OR Y
230 IF PEEK (DX) = CX AND PEEK (DY) = CY THEN 210
240 REM SAVE THE CURRENT CX,CY
250 TX = CX:TY = CY
260 REM RETRIEVE THE NEW CX,CY
270 CX = PEEK (DX):CY = PEEK (DY)
280 REM ERASE OLD CURSOR
290 & C USR TX,TY
300 REM IF BUTTON THE PLOT POINT
310 IF PEEK (BT) = I THEN -& PLOT TX,TY
320 GOTO 190
330 REM SCREEN INITIALIZATION
340 REM SELECT INPUT DRIVER
350 & INPUT IN
360 & C HGR
370 REM POKE TABLE ADDRESS
380 POKE BL,O: POKE BL + 1,9
390 REM SET THE PEN COLOR
400 & COLOR= 15
410 RETURN
```

&CUT - Cut a Shape From the Screen

Parameters:

LEFT - left edge of the area to cut
RIGHT - right edge of the parameter to cut
TOP - top edge of the parameter to cut
BOTTOM - bottom edge of the parameter to cut

Result:

This command takes a rectangular area of the screen and places it into the shape table in position 255. The &COPY command can then be used to move the shape to a different location, and &DRAWBLOCK can be used to draw the shape onto the screen. All of the parameters must be valid screen coordinates, and LEFT must be less than or equal to RIGHT, and BOTTOM must be less than or equal to TOP. If any of these rules are broken, an error results. An out of memory error can result if you try to cut out an area that will not fit into the buffer holding the shape table.

Errors:

Illegal Quantity Error Out of Memory

Example:

SEE &COPY

&DHGR - Initialize Double High Res Graphics

Parameters:

none

Result:

This command sets up the screen for black and white double high resolution graphics. It is used just like the HGR command in Applesoft. The viewport is set to the full screen, which means that the top is set to 191, the bottom to 0, the left to 0, and the right to 559. The pen color is set to black, and the push and pop stacks are initialized. In this mode, only two colors are available: black and white. &COLOR will not let you set the pen color to any other color.

Errors:

none

Example:

See &DRAWTO for an example.

&DRAWBLOCK - Draw a Block on the Screen

Parameters:

X - X coordinate of the top left corner of the block
Y - Y coordinate of the top left corner of the block
NUM - number of the shape to draw

Res u I t:

NUM is the number of one of the shapes in the shape table, and must be in the range of 0..255. The shape is drawn at X, Y, which must be on the screen. If the shape table pointer in page 3 is zero, or if the table doesn't have a shape in spot number NUM, an error results. A null shape is legal, so if a shape was defined with a width and height of zero, no error results, and nothing is drawn.

If the invisible flag in page 3 is set to true (1), color number 5 is treated as an invisible color. This means that whenever a gray point is found in the shape, it is not plotted on the screen. By careful use of this feature, odd shapes can be drawn on a patterned background. If the invisible flag is set to zero, color number five is treated like any other color.

Errors:

Illegal Quantity Error Illegal Direct Error

Example:

See &COPY for an example.

&DRAWTO - Draw a Line

Parameters:

X - X coordinate to draw to
Y - Y coordinate to draw to

Results:

Draws a line from the current pen location to the X, Y specified, and sets the pen to the new X, Y. The line is drawn using the current pen color. X and Y must be on the screen.

Errors:

Illegal Quantity Error

Example:

```
10 MX = 559
20 MY = 191
30 CX = MX / 2:CY = MY / 2
40 & D HGR
50 & COLOR= 15
60 FOR J = 0 TO MX STEP 8
70 & MOVE TO CX,CY
80 & DRAW TO J.0
90 & DRAW TO J,MY
100 & DRAW TO MX,O
110 & MOVE TO J,MY
120 & DRAW TO 0,0
130 NEXT J
140 INPUT A$: HOME : & TEXT LIST
```

&ELLIPSE - Draw an Ellipse

Parameters:

LEFT - left edge of the ellipse
RIGHT - right edge of the ellipse
TOP - top edge of the ellipse
BOTTOM - bottom edge of the ellipse

Results:

An ellipse is drawn on the screen, filling the area specified by the parameters. All of the parameters must be on the screen, and LEFT must be less than or equal to RIGHT and BOTTOM less than or equal to TOP. The current pen color is used. Only the outline of the ellipse is drawn; it is not filled in.

Errors:

Illegal Quantity Error

Example:

```
10 MX = 139
20 MY = 191
30 L = MX / 2 - 2: R = MX / 2 + 2
40 B = MY / 2 - 2:T = MY / 2 + 2
50 & C HGR
60 & COLOR= 15
70 FOR J = 0 TO 22
80 & ELIPSEL,R,B,T
90 L = L - 3:R = R + 3
100 B = B - 3:T = T + 3
110 NEXT J
120 INPUT A$: HOME : & TEXT : LIST
130 & TEXT
```

&FILL - Fill the Window

Parameters:

none

Results:

The current viewport is filled with the current pen color. Color mapping is ignored during the fill. This fill is used to set large rectangular background areas; for a fill that will handle irregular shapes, see &FLOOD.

Errors:

none

Example:

```
10 X = 7
20 L = O:R = 139
30 B = O:T = 191
40 COLR = 0
50 & C HGR
60 FOR COLR = 15 TO 6 STEP - 1
70 & COLOR= COLR
80 & VIEWL,R,B,T
90 & FILL
100 L = L + X:R = R - X
110 B = B + X:T = T - X
120 NEXT COLR
130 INPUT A$: HOME : & TEXT : LIST
```

&FIND - Find the Cursor Position

Parameters:

none

Results:

The X and Y coordinates of the pen are placed in page 3, where they can be read using PEEK statements. The X location is placed in $3CA and $3CB (decimal locations 970 and 971). The Y location is at $3CC and $3CD (decimal locations 972 and 973). Although the Y value is returned as a two byte value, it is not really necessary to read both bytes, since the screen only goes to 191, and the second byte will be zero for all values below 256. For the same reason, you only need to read the second byte of the X value if you are using color double high resolution graphics. With black and white graphics, the X value can range up to 559, while with color graphics, 139 is the largest possible X value.

Errors:

none

Example:

```
10 PY = 972:PX = 970
20 HOME
30 & C HGR
40 & COLOR= 15
50 FOR X = 0 TO 15
60 & MOVE TO X,Y
70 & PLOT X,Y
80 Y = Y + 1
90 & FIND
100 PRINT PEEK (PY), PEEK (PX)
110 NEXT X
120 INPUT A$: & TEXT
```

&FLOOD - Flood Fill

Parameters:

X - X location to begin the fill
Y - Y location to begin the fill

Result:

The flood fill command will change the color of a simply connected region from what it is now to the current pen color. Simply connected means that the area that will be filled is all of the points on the screen that are the same color and can be reached by moving up, down, left or right without leaving the color being filled. It doesn't matter how large the region is. If the area is extremely complicated, the command may not reach every point, but will not give an error. X and Y must be on the screen,

Errors:

Example:

Illegal Quantity Error

```
10 & C HGR
20 & COLOR= 15
30 & FILL
35 & COLOR= 0
40 & ELIPSE30,60,40,90
50 & BOX70,110,80,150
60 & COLOR= 2
70 & FLOOD45,65
80 & COLOR= 13
90 & FLOOD71,81
100 & VIEW20,119,20,171
110 & COLOR= 5
120 & FLOOD21,21
```

&GROUT - Graphics Output of Text

Parameters:

none

Result:

Before using this command, a shape table must be defined. In most shape tables, the first 127 shapes are characters, with the number of the shape matching the numbers used in the ASCII character set. Once the &GROUT command is used, anything that would normally have been printed on the text screen will be printed on the graphics screen instead, using the current pen color. The shapes for characters are usually defined with backgrounds made up of the gray color (number 5) that is used as the invisible color. The character output routine uses the invisible color, so that the background that the character is drawn over is not effected by the writing of the character.

The character will appear on the screen with the upper left corner of the character at the current pen position. The pen is automatically moved to the right after the character is drawn. Control keys have no effect - the RETURN key, for example, will print like any other character. It does not cause the cursor to move to a new line.

While it is plotting characters, the program uses color mapping to cause most characters to appear as the current pen color, and to allow the background to show past the parts of the character which are not plotted. The first gray color (color number five) is used as an invisible color, so that whatever was on the screen stays there. Black is used as a mapped color, so that any pixel which is black in the character definition is drawn using the current pen color.

&NORMAL is used to change the output hook back, so that text again appears on the text screen.

Errors:

Illegal Direct Error

Example:

```
10 BL = 962
20 PRINT CHR$ (4)"BLOAD FONTS/OUTLINE.FONT"
30 & D HGR
40 POKE 968,1
50 POKE BL,O: POKE BL + 1,64
60 & COLOR= 15
70 & FILL
80 & COLOR= 0
90 & GR OUT
100 & MOVE TO 10,180
110 PRINT "THIS IS A TEST";
120 & MOVE TO 10,170
130 PRINT "Please type anything you wish..";
140 PRINT "a carriage return terminates this test"
150 & MOVE TO 10,150
160 INPUT A$: & NORMAL : HOME : & TEXT : LIST
```

&HOME - Clear the Screen

Parameters:

none

Result:

The viewport is set to the full screen size, the pen color set to black, and the screen is cleared. The cursor is set to 0, 0 (the lower-left corner of the screen).

Errors:

none

Example:

```
10 & C HGR
20 & VIEW20,109,20,171
30 & COLOR= 15
40 & FILL
50 & HOME
60 INPUT A$: HOME : & TEXT : LIST
```

&INPUT - Select the Mouse, Joystick or Keyboard

Parameters:

NUM - number of the device to read

Result:

The graphics package has a uniform interface that your program can use so that one read command can be used with either the mouse, joystick, or the keyboard. This command selects which of those devices the &PDL command will read. It must be used before the &PDL command.

The value for the parameter is zero for the keyboard, one for the joystick, and two for a mouse. If you select &INPUT 2 and a mouse is not connected, you will get an illegal quantity error. Any value other than zero, one or two will cause the same error,

Errors:

Illegal Quantity Error

Example:

SEE &CUSR

&INVERSE - Invert the Screen

Parameters:

LEFT - left edge of the area to invert
RIGHT - right edge of the area to invert
BOTTOM - bottom edge of the area to invert
TOP - top edge of the area to invert

Result:

The rectangular area defined by the input parameters is reversed (exclusive or-ed with a $FF). A second use of the command will cause the area to return to its original state. This command is used in the drawing program to reverse the pull down menus as the cursor moves across them. The command ignores the window and any color mapping.

The parameters must be on the screen, and LEFT must be less that or equal to RIGHT, and TOP greater than or equal to the BOTTOM. If not, an error will result.

Errors:

Illegal Quantity Error

Example:

```
10 X = 9
20 L = O:R = X
30 COLR = 0
40 & C HGR
50 & COLOR= COLR
60 & FILL
70 FOR COLR = 1 TO 15
80 & COLOR= COLR
90 & VIEWL,R,0,191
100 & FILL
110 L = L + X:R = R + X
120 NEXT COLR
130 FOR PAUSE = 0 TO 1000: NEXT PAUSE
140 & INVERSE 0,135,0,191
150 INPUT A$: IF A$ .... THEN 140
160 HOME : & TEXT LIST
```

&MOVETO - Move the Pen

Parameters:

X - X coordinate to move the pen to
Y - Y coordinate to move the pen to

Res u I t:

The pen is moved to the position specified. The screen is not effected in any way. The next use of any command that uses the current pen position will use the new value. Examples of such commands are &DRAWTO and text output after an &GROUT command. If the coordinate given is not on the screen, an error will result.

Errors:

Illegal Quantity Error

Example:

For a sample program, see &DRAWTO or &PLOT.

&NORMAL - Restore Output Hook

Parameters:

none

Result:

Restores the output hook, so any future text output will go to the text screen, not the graphics screen, This command can be used several times in a row, or it can be used before the &GROUT command, with no problems.

Errors:

none

Example:

See &GROUT.

&PDL - Read the Paddle

Parameters:

none

Result:

This command provides a single, unified way to read a keyboard, joystick or mouse and get back an X, Y value and a flag that tells if the button is down. In the case of the mouse, the button is the mouse button. For the joystick, the button is game paddle button 0. For the keyboard driver, the open apple key is used.

The current input device is read, and the result placed in locations ME to $3C1 (decimal 958 to 961). An error will result if the &INPUT command has not been used to select the input device. The value returned is always on the screen. Location $3BD (decimal 957) is set to 0 if the button is not being pushed, and to a non-zero value if the button is being pushed.

Errors:

Illegal Direct Error

Example:

SEE &CUSR

&PLOT - Plot a Point

Parameters:

X - X coordinate to plot at
Y - Y coordinate to plot at

Result:

A point is placed on the screen at the location specified, using the current pen color. An error will result if the coordinate given is not on the screen.

Errors:

Illegal Quantity Error

Example:

```
10 & D HGR
20 & COLOR= 15
30 FOR J = 0 TO 559 STEP 3
40 & PLOT J,O
50 & PLOT J,191
60 NEXT J
70 FOR J = 0 TO 191 STEP 3
80 & PLOT 0,J
90 & PLOT 559,J
100 NEXT J
110 INPUT A$: HOME : & TEXT LIST
```

&POP - Pop the Screen

Parameters:

none

Result:

The screen area on the top of the graphics stack is popped back onto the screen at the same location it came from. Color mapping and the viewport are ignored. This removes the screen area from the top of the stack, so a subsequent &POP will pop a different area.

Using an &POP with nothing on the stack has no effect.

Errors:

none

Example:

```
10 X = 9
20 L = O:R = X
30 COLR = 0
40 & C HGR
50 & COLOR= COLR
60 & FILL
70 FOR COLR = 1 TO 15
80 & COLOR= COLR
90 & VIEWL,R,0,191
100 & FILL
110 L = L + X:R = R + X
120 NEXT COLR
130 & PUSH20,120,100,180
140 & VIEW20,120,100,180
150 & COLOR= 15
160 & FILL
170 FOR PAUSE = 1 TO 500: NEXT PAUSE
180 & POP
190 INPUT A$: HOME : & TEXT : LIST
```

&PUSH - Push the Screen

Parameters:

LEFT - left edge of the area to invert
RIGHT - right edge of the area to invert
BOTTOM - bottom edge of the area to invert
TOP - top edge of the area to invert

Result:

The area defined by the parameters is pushed onto the graphics stack. After using this command, an &POP command can be used to restore the area of the screen, no matter what has been drawn there in the mean time. This command, along with &POP, can be used to handle pull down menus. To do this, start by pushing the area that will be covered by the pull down menu. Next, draw the pull down menu and do any processing that you need to. When it is time to pull the menu up, simply do an &POP.

The command can also be used to do windows, using basically the same idea. First, the area to be used by the window is pushed, and the window is drawn. When it is time to erase the window, just do an &POP,

Keep in mind that the areas are pushed onto a true stack. This means that if you push two areas, say area A and then area B, that there is no way to pop A without first popping B~ By the way, the stack is located in the alternate 64K of memory, so it can get quite large, yet never takes room away from your program.

Errors:

Illegal Quantity Error

Example:

SEE &POP

&READ - Read a Pixel

Parameters:

X - X coordinate to read
Y - Y coordinate to read

Result:

The color number for the point at X, Y is placed in page 3 location $3C9 (decimal 969). The point must be on the screen, or an error will result.

Errors:

Illegal Quantity Error

Example:

```
10 X = 9
20 L = 0: R = X
30 COLR = 0
40 & C HGR
50 & COLOR= COLR
60 & FILL
70 FOR COLR = I TO 15
80 & COLOR= COLR
go & VIEWL,R,0,191
100 & FILL
110 L = L + X:R = R + X
120 NEXT COLR
130 HOME
140 FOR J = 2 TO 139 STEP 9
150 & READ J,20
160 PRINT PEEK (969)
170 NEXT J
180 INPUT A$: & TEXT
```

&SETCMAP - Set Color Map

Parameters:

NUM1 - color number for pen
NUM2 - color number for screen
NUM3 - resulting color to plot

Result:

This command is used to set the color map, which can then be enabled with an &CMAPON command. When the color map is enabled, plotting a point using color NUM1 on a location that is currently set to color NUM2 will result in the screen being changed to color NUM3. NUM3 can be the same as NUM1, in which case no difference is seen between using the color map and not using it. The default table is set up that way, so this command must be used before color mapping has any effect.

All color values must be in the range 0-15, or an error will result,

Errors:

Illegal Quantity Error

Example:

```
10 X = 9
20 L = O:R = X
30 COLR = 0
40 & C HGR
50 & COLOR= COLR
60 & FILL
70 FOR COLR = I TO 15
80 & COLOR= COLR
90 & VIEWL,R,0,191
100 & FILL
110 L = L + X:R = R + X
120 NEXT COLR
130 & VIEWO,139,0,191
140 GOSUB 300
150 REM TURN ON THE COLOR MAP
160 & CMAPON
170 & COLOR= 0
180 FOR J = 180 TO 160 STEP - 1
190 & MOVE TO 0,J: & DRAW TO 139,J
200 NEXT J
210 REM TURN OFF THE COLOR MAP
220 & CMAPOFF
230 FOR J = 150 TO 130 STEP - 1
240 & MOVE TO O,J: & DRAW TO 139,J
250 NEXT J
260 INPUT A$: HOME : & TEXT : END
270 REM SETS COLOR MAP SO THAT BLACK ON
280 REM ANY COLOR BACKGROUND PLOTS THE
290 REM EXCLUSIVE OR OF THE BACKGROUND
300 & SETCMAPO,0,15: & SETCMAPO,1,14
310 & SETCMAPO,2,13: & SETCMAPO,3,12
320 & SETCMAPO,4,11: & SETCMAPO,5,10
330 & SETCMAPO,6,9: & SETCMAPO,7,8
340 & SETCMAPO,8,7: & SETCMAPO,9,6
350 & SETCMAPO,10,5: & SETCMAPO,11,4
360 & SETCMAPO,12,3: & SETCMAPO,13,2
370 & SETCMAPO,14,1: & SETCMAPO,15,0
380 RETURN
```

&SETCUSR - Create Cursor Mask

Parameters:

NUM - shape number

Result:

NUM is the number of a shape that will be used as the cursor for the &CUSR command. The &CUSR command allows drawing of a very fast cursor, which can be up to eight pixels wide and ten pixels high. The shape used must exist, and must consist of only black and white pixels. If the shape is too big, this command automatically clips it to the proper size.

Errors:

Illegal Quantity Error

Example:

SEE &CUSR

&TEXT - View the Text Screen

Parameters:

none

Result:

The text screen is shown instead of the graphics screen. The graphics screen and text screen are unaffected. The output hook is not changed, so if you want to write to the text screen, be sure and use the &NORMAL command if the &GROUT command has been used.

Errors:

none

Example:

See &GROUT for an example.

&TRTLDRAW - Turtlegraphics Draw Command

Parameters:

NUM - distance to draw

Result:

A line of the current pen color is drawn from the current pen position in the direction of the turtle for a length of NUM. The cursor is set to the new location. NUM must be between 32768 and 32767.

Errors:

Illegal Quantity Error

Example:

```
10 AN = 0
20 DX = 125:DY = 155
30 & C HGR
40 & COLOR= 15
50 & MOVE TO DX,DY
60 & TRN TO AN
70 & TRTL DRAW 2
80 & TURN15
90 & FIND
100 IF PEEK (970) = 0 OR PEEK (972) 0 THEN 120
110 GOTO 70
120 AN = AN + 90
130 DY = DY - 20
140 IF AN < > 360 THEN 50
150 INPUT A$: HOME : & TEXT : LIST
```

&TRTLMOVE - Turtlegraphics Move Command

Parameters:

NUM - distance to move

Result:

The turtle position (which is also the cursor position) moves NUM pixels in the current turtle direction. The screen does not change. NUM must be in the range -32768 to 32767.

Errors:

Illegal Quantity Error

Example:

```
10 & C HGR
20 & COLOR= 15
30 & MOVE TO 70,85
40 & TRN TO 90
50 FOR J = 1 TO 4
60 & TRTL DRAW 15
70 & TRTLMOVE15
80 & TURN90
90 NEXT J
```

&TRNTO - Turtlegraphics Turn To Command

Parameters:

NUM - number of degrees to turn

Result:

The direction of the turtle is changed to NUM. Zero degrees is to the right; ninety degrees is straight up. NUM must be in the range -32768 to 32767. If it is outside of the range 0 to 359, it will be adjusted to an equivalent angle in that range.

Errors:

Illegal Quantity Error

Example:

SEE &TRTLDRAW or &TRTLMOVE

&TURN - Turtlegraphics Turn Command

Parameters:

NUM - number of degrees to change

Result:

The direction of the turtle is changed by NUM degrees. Turning a positive number of degrees causes the turtle direction to change in the counterclockwise direction. Negative turns are allowed; turning 180 degrees is equivalent to turning -180. NUM must be in the range -32768 to 32767.

Errors

Illegal Quantity Error

Example:

SEE &TRTLDRAW or &TRTLMOVE

&VIEW - Set the Viewport

Parameters:

LEFT - left edge of the viewport
RIGHT - right edge of the viewport
BOTTOM - bottom edge of the viewport
TOP - top edge of the viewport

Result:

No change is visible on the screen, but any future drawing commands cannot effect the area outside of the box defined by the parameters. Certain commands ignore the window; these include &POP and &CUSR.

The box must be on the screen. LEFT must be less that or equal to RIGHT, and TOP must be greater than or equal to BOTTOM.

Errors:

Illegal Quantity Error

Example:

See &FILL and &COLOR for sample programs.