Basic 8 Manual Errata

Basic 8.0

The Enhanced Graphics System For The C128

Developed by

LR Wallace, DP Darus

WALRUSOFT

March 19, 1987

Dear Valued Customer,

Thank you for purchasing Basic 8 with Basic Paint. Basic Paint was created using Basic 8 and has been included with the package as an added bonus to show what can be accomplished with Basic 8. Also included is an Icon based, Desk-top utility which enables you to access your Basic 8 programs quickly and conveniently.

We think you will find Basic 8 to be one of the most sophisticated and powerful software packages available for the C-l28. It breaks new ground in unleashing hidden graphic capabilities rivaling those of l6-Bit Computers! Your purchase enables us to continue to offer software of the highest quality to users of the Commodore 64 and 128.

Patech Software is founded on Three Basic Principles:

1. To offer the best quality software at a fair price. (One which accurately reflects its true value to the user.)

2. To offer realistic and usable support to each Registered user after purchase, to ensure that he gets full value from the software.

3. To Present the software in its original state, without copy-protection of any kind! This enables you to make back-ups for your convenience and the protection of your investment (as well as your equipment!).

   You put your trust in Patech when you purchased Basic 8. In turn, we are putting our trust in you to help ensure that we are fairly compensated for our software. Only through this mutual trust can we continue to serve your software needs. Please don't force us to add copy protection to our software. That would only hurt both of us!

Patech is a very open company. If you have any questions, comments or suggestions, please write to us. We will do our best to respond promptly.

If you have written any programs (with or without the aid of Basic 8) which you think have merit, or if you have an idea for one which you think fills a genuine need, please contact us. We are always interested in good programs and ideas. Who knows, you may just become the newest member of the Patech Software Development Team!

To keep fully informed of future products and updates; and to become a Registered User, please mail the enclosed Registration Card. Whenever you contact us for support, please reference the Serial Number printed on the card. (Make a record of the number before you mail the card!)

I hope you find that Basic 8 fulfills your expectations. Thanks again for your support.

Sincerely,

Paresh Patel, President
Patech Software, Inc.

P.O. Box 5208

Somerset, NJ 08873

(1)

TABLE OF CONTENTS

	Page
TABLE OF CONTENTS	1
HARDWARE REQUIREMENTS	2
MAKING BACKUPS	3
PHILOSOPHY	4
FOREWORD	6
1. INTRODUCTION	8
1.1 COMMAND SUMMARY	12
1.2 STARTING BASIC 8.0	13
2. SCREENS AND MODES	14
3. DRAWING COMMANDS	20
4. THREE DIMENSIONAL GRAPHICS	25
5. MEMORY MANAGEMENT	31
5.1 BUFFERS	31
5.2 STRUCTURES	32
A. Saving and recalling STRUCTURES	33
B. Patterns	35
C. Logo	36
D. Character Font	36
E. Brush	37
6. Exotic Commands	38
7. Making A Distributable Disk - The Run Time System	42
8. BASIC PAINT	44
9. WIOS WORKBENCH And Other Supplied Utilities	62
10. Basic 8.0 COMMAND ENCLYOPEDIA	65
11. Printer Drivers	122

Appendix A: The 8563 VIDEO CHIP - Ram Expansion 173

Appendix B: VIDEO RAM MEMORY MAPS (MODE's 0-3) 175

Appendix C: C128 Basic 8.0 MEMORY MAP 177

Appendix D: File Formats, Naming Conventions 179

Appendix E: Data Compression Algorithms 185

Appendix F: Other Suggested Standards 187

The Players 188

(2)

Required Equipment

Commodore C128 (or C128D) Computer in C128 80 column Mode only

At Least 1 disk drive (1541, 1571, 1581 and compatibles)

RGB Monitor

Supported Hardware Options

External RAM Expansion

Model 1700 128K RAM cartridge, or

Model 1750 512K RAM cartridge

Internal 64K RAM Expansion for the 8563 Video Display Chip

(New C128D model comes with the 64K already built in.)

Model 1351 Proportional Mouse

Standard Joystick

Dot Matrix Printers (See Chapter 11 for specific printers)

(3)

MAKING BACKUPS OF YOUR BASIC 8.0 DISK

Even though it is copyrighted, your BASIC 8.0 disk is not copyprotected. In fact, it comes with a built in backup facility. The original disk you buy is not meant to be used for work. The first time you boot the disk, have three formatted blank disks ready. Put the original BASIC 8.0 disk into the drive and either turn on the computer or press the reset button (if the computer is already on). You vill be prompted with a menu which has three choices. These menu options allow you to make a BASIC 8.O Work Disk, a BASIC PAINT Work Disk or a BASIC 8.0 RunTime Disk. You should make one of each, then put the original BASIC 8.0 System Disk away, where it will be safe. Use it at another time to make additional work disks or runtime disks. Of course, you can also use your favorite disk copy program to back up your work disks and runtime disks.

The BASIC 8.O Work Disk is for writing and editing your BASIC 8.0 programs. You may not give this disk away! Make a backup of this disk, and work from your backup work disk. This disk contains the complete BASIC 8.0 editor system, and is the disk you will use for creating your own BASIC 8.0 programs. To use this disk, just put it in the drive and boot the computer. The BASIC 8.0 system will load and become initialized. You may then write programs, run them and save them as you normally would. The difference lies in the many new BASIC 8.O commands available.

The BASIC PAINT Work Disk contains the included 80 column color paint program, as well as menus, fonts and requestors. It is a RunTime Program, meaning it contains only the runtime library, not the editor. You cannot edit or list the program,

and you cannot sell or give it away. The BASIC PAINT disk is

also an autoboot disk. Just plug in a 1351 mouse into port one or a joystick into port two and boot the disk. You will be presented with a series of menus and requestors that allow you to configure the program to your system.

The BASIC 8.0 RunTime Disk is the only legal way to sell or give away programs you have written with BASIC 8.0. It contains the RunTime Library, which is a program that allows BASIC 8.0 programs to run, but not edited or listed. This disk also contains the WORKBENCH program, which is simply a BASIC 8.0 program that allows you to use a mouse to load and run BASIC 8.0 programs. This disk too is a autoboot disk. It will load and activate the RunTime Library, and then it loads and runs the WIOS WORKBENCH. You can put your own programs and their required files on this disk, where they can be loaded by pointing the mouse and selecting them with the mouse buttons. Or you can alter the booting sequence of the RunTime Disk in order to directly load and run your program. See Chapter Seven for details.

OUR PHILOSOPHY ON SOFTWARE COPY PROTECTION

We made a very deliberate decision to not put any type of copy protection on the disk, so you could easily make working disks. At this time, nearly all new Commodore software is protected. So why not BASIC 8.0? We believe that C128 users deserve to be trusted. We believe in giving you value for your money. So on this disk is not only the first truly unique C128 software package (BASIC 8.0), there is a FREE run time system that can be used by registered owners to produce program disks that will run your BASIC 8.0 programs without using the BASIC 8.0

editor disk. These run time systems can be given away or sold

without you paying us royalties! So if you write an exceptional program you want to share, you can do so legally without infringing on our copyright. (These run time programs can only be ran, not edited or altered by the user.)

If that isn't enough, there is also a bonus program written in BASIC 8.O called BASIC PAINT. This is a very powerful graphic paint package, actually more powerful than most commercial systems available for sale. This program is also copyrighted, so you may not give it away or sell it. Use BASIC PAINT just as a paint system, or use it to create pictures, brushes and patterns for use in your BASIC 8.0 programs.

Besides informing you that making copies of BASIC 8.0 and BASIC PAINT and giving them away (or selling them) is illegal and makes you subject to criminal penalties, please consider this. We spent two years developing this system for the C128, at a time when most software developers were ignoring your software needs. We have given you one of the most powerful graphic languages ever produced on eight bit machines, a free run time system for the programs you develop, as well as a powerful paint package. We have trusted you by allowing you to make copies of the disk for your own use. If we find that all this is not respected and piracy occurs anyway, other software being developed by us may well never show up, or may be protected if it does.

But if you show the proper restraint and do not give out illegal copies, rest assured we will follow with more powerful, unique and unprotected software for your C128. And who knows, other companies may follow our lead!

(6)

FOREWORD

The Commodore 128 personal computer was released in mid 1985 as the logical upgrade path to the popular C64 computer. In

order to address the wants and needs of the potential C128 computer buyers, Commodore asked for suggestions about what the users would like in this new computer system. The major suggestions were implemented. These were C64 compatibility, a faster high capacity disk drive, more standard memory and ram expansion capability, a higher level BASIC language, and a professional 80 column color display. All these improvements over the C64 allowed Commodore to create what is perhaps the most powerful 8 bit microcomputer yet released to the public, and they explain why in a little more than a year over one million C128's have been sold.

One problem associated with the C128 has been the limited amount of software developed for its native C128 mode. This is mainly because of the C64 compatibility as software developers wanted to play it safe and only released software that would run in C64 mode. This lack of real C128 mode software has led to a good deal of anger and disappointment in the C128 buying public, yet they have stayed with their chosen computer system. And of the one hundred or so native C128 packages available after a year, most have been productivity packages like word processors, spreadsheets and databases. Interestingly enough the ones that have been successful have all catered to the C128's 80 column display capabilities.

While the built in Basic 7.0 has a very large base of commands that take advantage of the C128's 40 column graphics capability, almost nothing is available to the BASIC programmer

(7)

to access the C128's 80 column graphics possibilities, which are quite extensive and actually more powerful than the supported 40 column graphics. In fact Commodore originally stated that there would be no high resolution RGB graphics (except text) as the

8563 chip was not designed to support them and any that could be added via software would be too slow to be usable. The authors of Basic 8.0 demonstrated that high speed monochrome graphics were not only possible, but quite practical in an early copyrighted language extension called Ultra Hires. This small graphics language generated such an avalanche of world-wide interest that we decided to see just how extensive the graphics of the 8563 were. Our criteria was to produce the most powerful and productive graphics system ever developed for a 8 bit computer. With Basic 8.0 we more than succeeded.

David P. Darus

Louis R. Wallace

(8)

CHAPTER ONE

INTRODUCTION TO THE BASIC 8.0

PROGRAMMING LANGUAGE

Basic 8.0 is a new and exciting programming environment for the C128 computer owner. It allows you to use an entirely new aspect of graphics on the C128, graphics that actually exceed the high resolution and power that the C128 and C64 user have come to expect. Yet it does so while being compatible with the existing advanced non-graphic commands of Basic 7.0 that are built into the computer.

Why is Basic 8.0 so powerful? Well, first of all it offers an extremely high resolution graphics display of 640 X 200 pixels on the screen at the same time (twice the resolution of 40 column mode). This screen memory is completely separate from the normal system ram contained in the computer, and unlike the 40 column graphic screen it does not use any of the users valuable

programming ram. Secondly, the normal bitmap graphics used on microcomputers are a two dimensional system. With Basic 8.0 however, its drawing commands work in a three dimensional environment, and the BASIC programmer has a great deal of control over perspective, rotation, viewing angles, the origin and depth of view. Furthermore, it gives the programmer commands to generate three dimensional solid surfaces, a power usually only available on much more expensive computers. Third, Basic 8.0 offers the RGB graphics user five different types of graphic displays, four of which can use all 16 colors on the screen at the same time. It is also possible to have several screens in memory simultaneously, each with a different color resolution and/or screen size. Fourth, Basic 8.0 has many other special

and unique features. For example you can make Virtual screens that are much larger than the normal 640 X 200 displayable screen. By using the SCRDEF command you can create custom screens that are up to 2040 pixels wide or over 800 pixels tall. Conversely, by using the MODE command you can easily switch between any of the 32 predefined graphic screens. There are also commands to SCROLL a screen, save and load screens and brushes, define a LOGO, PAINT with exotic color PATTERNS of your own design, create WINDOWS, make printed copies of your images on many different printers (and in many different sizes as well as in color) and much more.

Another important feature of Basic 8.0 is it's total support of the C128 and its hardware expansion capabilities. Obviously the 80 column RGB display is used. And of course the Basic 8.O user can make use of the 1541,1571 and 1581 disk drives. Support for a number of different dot matrix printers allow programs to

create hardcopy of the graphic screens. Basic 8.0 also is one of the first commercial software packages to support the 1700 and 1750 ram expansion modules. You will find that the new 1351 proportional mouse has found a home here in Basic 8.0 programs, and you can still use a joystick too. You will even find that the extra 64K video ram expansion for the 8563 chip being installed in tens of thousands of C128's by their owners (and already present in the new C128D) is completely supported. Basic 8.0 can use in one way or another the whole 704 K ram that can be installed in the C128, as well as virtually every peripheral device. In fact every time you upgrade your C128 computer, you will find Basic 8.0 waiting with even more capabilities for you to use!

Chapter One

(10)

The following chapters will help you learn about Basic 8.0 and its commands and techniques. They are divided into several sections: screens and modes, 3d graphics, solid shapes, memory management, data structures, the mouse/joystick/pointer system, printers and more. Following these chapters you will find a Basic 8.0 ENCYLOPEDIA, which is an organized directory of the commands and their syntax. Here every command is listed and each parameter explained. Then comes the Basic 8.0 MEMORY MAP, which simply shows the areas of the computer that are used by the ROMS, RAM and Basic 8.0 itself. Finally the appendix contains information on the data compression algorithms used, the file format for the graphics, 3d data file structures and more!

The hardware in the C128 that lets Basic 8.0 generate all these special graphics is the 8563 Video Display Chip. This is a very powerful RGB display processor that is a new addition to the

Commodore computer family. It is controlled through 37 registers that are completely separate from the normal C128 operating system. If is only possible to address the 8563 registers and its video display ram through two locations in the C128. These are $D600 (Address Register) and $D601 (Data Register). Programming the 8563 can be quite complex, and explains why graphics were thought to be infeasable, though obviously not impossible. On a standard C128 the 8563 video chip has 16k of dedicated display ram you can use, but it is quite possible to expand that ram to 64k, opening a whole new world of C128 and Basic 8.0 graphics.

Basic 8.0 allows you to access these new capabilities as well as creating many additional features in your own programs. It does this while still supporting the built in Basic 7 commands

(except of course the 40 column graphics commands).

How does Basic 8.0 work? Well, the language is what is known as a wedge. A wedge is a machine language program that is wedged into the computers BASIC interpreter where the new commands can be executed (and evaluated) along with the old. One of the problems associated with these types of languages is the fact they slow down the normal basic interpreter simply by adding more commands. Basic 8.0 does not work this way because we did not want to slow down your programs. Instead we choose a method called a syntax error wedge. This technique prefixes the new commands with a character that is not used by BASIC in any of its' normal commands. So when this character is encountered in a statement, the operating system jumps to the routine that handles syntax errors. We have taken over that function, and check for one of our commands when an error occurs. If it is a real syntax

error the program passes it on to the operating system for evaluation. If however we find it to be a Basic 8.0 command it jumps to our new interpreter. Only then is the new command interpreted. This technique saves a great deal of time in processing the program, especially when (like in Basic 8.0) the added language consists of a large number of new commands. This time savings means your programs run as fast as is possible.

The special character used to prefix all Basic 8.0 commands is the @ character. It is easy to see in your programs, and does not require pressing the shift key on the C128 keyboard to use. And, finally it should make it far easier for a compiler to be developed for the Basic 8.0 language, should a developer wish to do so.

Chapter One

(12)

Basic 8.0 COMMAND SUMMARY

@ANGLE

@ARC

@BOX

@BRUSHPATRN

@BUFFER

@CBRUBH

@CHAR

@CIRCLE

@CLEAR

@COLOR

@COPY

@CYLNDR

@DIR$

@DISPLAY

@DOT

@DRWMODA

@DRWMODB

@FETCH

@FLASH

@GR0W

@HCOPY

@LINE

@LOGO

@LSTRUCT

@MODE

@MOUSE

@ORIGIN

@PAINT

@PATTERN

@PIXEL

@PTR

@SCALE

@SCLIP

@SCRDEF

@SCREEN

@SCROLL

@SDAT

@SEND

@SPHERE

@SPOOL

@SSTRUCT

@STASH

@STORE

@STRUCT

@STYLE

@TEXT

@TOROID

@VIEW

@WALRUS

@WINDOWCLOSE

@WINDOWOPEN

@ZOOM

Chapter One

(13)

STARTING BASIC 8.0

To start BASIC 8.0 turn on your disk drive and RGB monitor. If you are using a dual monitor (like the 1902) make sure it is in RGB mode. Press the 40/80 column key on the C128 down. Put the BASIC 8.0 Work Disk in the drive, and turn on the computer. The system will autoboot (load and run itself). When it is finished loading, you can then run one of the supplied utility programs or write your own.

Chapter One

(14)

CHAPTER TWO

SCREENS AND MODES

There are five different screen formats supported by BASIC 8. The key difference between them lies not in the screen pixel resolution, but rather in the color resolution or color cell size. In monochrome graphics mode there is no color information stored for a color cell, so there is only two colors, foreground and background, available to the screen. Therefore a 640 X 200

monochrome screen requires 16000 bytes of ram for the bitmap, and no ram for color information. To use many colors at once you must have somewhere in memory to store the color information, or attributes. On the C128 a single attribute byte controls the color information for a color cell. Each attribute is responsible for storing the color information for a specific area of the screen known as the color cell. A color cell is always 8 bits (pixels) wide, and can be either 2, 4, 8 or 16 scanlines (pixels) tall. Each color cell can have any 2 colors, a foreground color and a background color, from a palette of 16 colors. The larger the color cell, the smaller the amount of ram required for color attributes. The smaller the color cell the more ram required for color attributes. So when using the multicolor modes some ram in addition to the bitmap ram is required to store the color information, with the exact amount depending on the color cell and screen size.

The five possible display modes are defined as such:

Mode	Type	Color Resolution
0	Monochrome	No color cell
1	Color Mode 1	8 X 2 pixel color cell
2	Color Mode 2	8 X 4 pixel color cell
3	Color Mode 3	8 X 8 pixel color cell
4	Color Mode 4	8 X 16 pixel color cell

For some point of reference look at display mode 3. It uses an 8 X 8 pixel color cell, which means that each of the 8 X 8 cells on the screen in this display mode can have their own unique pair of colors, one for the foreground and the other for

the background. This is the same color resolution as the C64 or C128 composite high resolution screens color capacity, yet the bitmap has twice the horizontal screen resolution of the composite mode. Each color cell requires 1 byte of attribute memory in addition to the bitmap ram requirements. Using a 640 X 200 display with an 8 X 8 color cell requires 16000 bytes for the bitmap and 2000 bytes for the color attributes totaling 18000 bytes of ram. This is obviously more than the 16K built in for the 8563 to use and since none of the C128's system ram can be directly accessed for 80 column graphic purposes it is not possible to display a full 640 X 200 color screen in any of the color modes with only the standard 16K. That's why we strongly recommend you have the full 64K video ram installed in your C128 (see appendix). The ram requirements for a 640 X 200 screen in each display mode are:

DISPLAY MODE	DESCRIPTION	COLOR CELL SIZE	RAM REQUIRED
	Monochrome	0	16000
	Color Mode 1	8 X 2	24000
	Color Mode 2	8 X 4	20000
	Color Mode 3	8 X 8	18000
	Color Mode 4	8 X 16	17000

As you can see it is not possible to have these full screen colors without the extra ram. If you have installed the full 64K of ram (or have a C128D) you can use any display mode as well as virtual screens larger than 640 x 200. However in a standard C128 there is only 16K, so to get around this limitation and still provide color, you must define screens that are less than

(16)

200 pixels high, which frees some ram to be used for color information. You can define your own screens using the @SCRDEF (SCReen DEFinition) command, or simply use the @MODE,O command to define 8 separate screens designed for the 16K C128. These screens already make allowances for the limited memory, and create screen displays of various heights and resolutions. They are as follows:

SCREEN 0	640 X 200 Monochrome
SCREEN 1	640 X 192 Color 8 X 16
SCREEN 2	640 X 192 Color 8 X 8
SCREEN 3	640 X 192 Color 8 X 4
SCREEN 4	640 X 192 Color 8 X 2
SCREEN 5	640 X 192 Color 8 X 8 Interlaced
SCREEN 6	640 X 192 Color 8 X 4 Interlaced
SCREEN 7	640 X 192 Color 8 X 2 Interlaced

After using the @MODE,O command you can work on any of the above screens by just using the @SCREEN,n command, where n is 0-7. We have found the @SCREEN,2 (640 X 176 Color, 8 X 8 color cell) the most generally useful color display in the standard, unexpanded C128. If your needs are such that a smaller display (640 X 152 or 640 X 104) is useful, by all means use those screens. The extra color capability is exceptional, and it allows for strikingly beautiful color displays. If you need a larger color display in a 16K system, use @SCREEN,1 for a 640 X 192 color display using a 8 X 16 color cell. You can use only one of the @MODE,O screens at any one time, but you can switch very easily between the different screen types. You can also maintain the different screens in memory as brushes, or @STORE them to disk until needed. Depending on the size of your

program, you can @STASH several screens in the C128's ram banks 0 and 1. However, if you have the 1700 or 1750 Ram Expansion cartridges you can have dozens of screens, fonts, brushes, logos

Chapter Two

(17)

and patterns in memory at one time.

If you have the 64K video ram expansion installed in your C128 (see Appendix) or own a C128D your screen possibilities are enhanced virtually 1000%! You can now use all four @MODE commands. @MODE,1-3 offers twenty four spectacular screen combinations. Besides 640 X 200 monochrome screens there is every possible 640 X 200 color screen (8 X 16, 8 X 8, 8 X 4 and 8 X 2). There is a large variety of VIRTUAL SCREENS (screens larger than what can be displayed on the monitor at once), including monochrome displays such as 1280 X 409, 640 X 819, 2040 X 252 and 800 X 655 pixels. There are many virtual color screens as well, up to 640 X 728 pixels. @MODE, 1 and @MODE,2 both contain combinations that allow two to four screens in memory at once, regardless of the screen size or type. Of course, you can use the @SCRDEF command to custom design screens of any size using any of the five color types. When used it overrides the definition given that screen number by @MODE,n. Take a look at the various screens in each mode, because it is likely most of the screens you want will be there; if not, then make some of your own design. You can find a detailed list of the screens in each mode in the chapter BASIC 8 COMMAND ENCYCLOPEDIA under @MODE; the complete syntax of @SCRDEF is also listed there.

You activate your chosen graphic screen with the command @SCREEN, draw screen, view screen. The draw screen and view screen signifies which of the possible (at one time) 8 screen

definitions to use as your current drawing screen and viewing screen. They do not have to be the same, so you can be looking at one screen while drawing on another (in 64K video ram systems) . This is ideal for many applications, and can be used

Chapter Two

(18)

for double buffered displays, a technique often used for animation. And screens can be saved or loaded with the @STORE and @DISPLAY commands.

At times you will need the use of the normal text screen, and you can return to it quite easily with @TEXT. This initializes the C128 80 column text display, using the standard character fonts. If you want to install custom fonts in text mode, using the @FONT command will allow you to load any two fonts into memory for use with the 80 column text display.

Another powerful command is @WINDOWOPEN. This command allows you to define special subwindow areas of the screen. The upper left corner of the window becomes the screen location 0,0. Both the windows size and location is user defined, and once defined all subsequent drawing commands default to it. It stays the default output until another @WINDOWOPEN or @SCREEN command is issued or you give the command @WINDOWCLOSE. Once opened, the window can be cleared, filled with color, drawn into, etc. If clipping is on, objects drawn outside the window are clipped.

Most, but not all, of the graphic commands work on the current window. Those that do not (like @STASH, @FETCH, @COPY etc.) are not drawing commands. In the BASIC 8.0 COMMAND ENCLYCLOPEDIA each command is defined and if it does not respond to the @WINDOWOPEN command, it will be clearly labeled as such.

Here is a simple example program that will define a @MODE,O

640 X 176 color screen, clear the display and set up some default drawing colors.

```
10 @WALRUS,0:REM define the 16K ram mode
20 @MODE,0:REM Select the 8 screens available to 16K users
30 @COLOR,0,8,0:REM black background, red drawing color, black border
40 @SCREEN,2,2:REM use the 640 x 176 color display 8x8 cells
50 @CLEAR,0:REM clear the display
60 SLEEP 5:REM wait 5 seconds
```

Chapter Two

(19)

```
70 @TEXT:REM return to text mode
```

A final note. The different color screen are not compatible. You cannot display an 8 x 8 screen in an 8 x 2 mode. The same is true for brushes. They can only use the color mode from which they were created. Monochrome screens and brushes can be loaded into any screen, although they will not contain any color information. Color screens and brushes of any color cell resolution can be displayed in a monochrome screen, with the color information ignored by the monochrome display. By going through a monochrome intermediary, you can transfer the bitmap of screens and brushes from different color modes.

Chapter Two

(20)

CHAPTER THREE

GRAPHICS

BASIC 8.0 has all the graphics shapes you would expect from

a graphics oriented language. It has @DOT, @LINE, @BOX, @ARC, @CIRCLE, @PAINT and more. These are normally rather standard commands, but this is BASIC 8.0, not some simple graphics system. Each of these commands offers far more than found in any other language. For example, each of them works in 3 dimensions. That's right, 3D! Each point is defined as an element of a X,Y,Z coordinate system. This is far more advanced than the simpler 2D system used on other graphic systems. It allows you to define and draw complex 3D objects without complex (and slow) mathematical calculations. It doesn't stop there either. BASIC 8.O has some powerful pattern capabilities that allows you to define patterns n bytes wide by m pixels deep and in color. This is especially useful with the @PAINT command, but also the drawing commands (and the character command) can be told to draw in the current pattern. That means your lines, circles, boxes, dots, arcs and letters can all be patterned (like the AMIGA). There's more! Each of the drawing commands (except @DOT) can be given a height parameter that allows you to multidraw it. You can define both the direction and the step value used by the height parameter with the @GROW command. The syntax is @GROW,x step,y step,z step; these step parameters can be positive or negative values. Therefore it is possible to grow in several dimensions and directions, all at the same time! If the step value is 1 or -1, the line is drawn as a solid. If it is 0 there is no growth in that direction. If greater or less than 1 or -1 then each layer of the graphic is stepped by that number of lines. What can you do with this? Well, for one thing it allows

lines of different weights. For another, when used with the

@CIRCLE command it allows you to draw cylinders. Used with the @ARC or @BOX commands it allows you to draw 3D bars of various shapes, or use a subtended arc and make 3D pie wedges for graphs or charts.

@BOX allows shearing which means to pull the base left or right from its position, while holding the top still. These parameters allow you to specify a shear direction (X or Y) and a shear value (how far to shear it). This allows you to make complex 3D bars very easily.

@ARC offers a few special features too. While it can be used for circles or ellipses, it can also be used for generating segments of a curve by specifying the start and ending angles of the arc. You can set the increment of the arc (angle between the lines used to draw the arc) to create other geometric shapes. The smaller the increment, the smoother the line. The larger the increment, the more noticeable the lines. If you define an increment of 45 degrees, the arc becomes an octagon (8 sides). Use 60 degrees and it's a hexagon (6 sides), 90 degrees and you have a diamond, and 120 degrees for a triangle.

When using pixel oriented graphics, a problem arises. The pixel is not symmetrical, rather it is bigger in the Y direction than the X. This means if you draw a 100 pixel horizontal line and a 100 pixel vertical line, the vertical line will be physically longer. To compensate for this BASIC 8.0 has the command @SCALE. There are three scale modes available. The first, @SCALE,0 is the normal mode. Here the Y pixel is longer than the X, so a 640 X 200 pixel screen is just that. @SCALE,1 tells the graphic primitives to use a different pixel scale. The

640 X 200 physical pixel screen becomes a 640 X 512 logical screen. There are no more real pixels displayed but they are addressed differently. This new graphic scale means that in the example above, both 100 pixel lines are the same physical length. @SCALE,2 doubles the logical scale of @SCALE,1. The screen now becomes 1280 X 1024 logical pixels. Again here we have no additional pixels, only a better logical addressing where all the addressable points are the same size. It is very important not to confuse these logical screens with virtual screens. Virtual screens really are physically larger in terms of real pixels. If you are working in a large virtual screen and change the scale the virtual screen is adjusted accordingly. To determine the new pixel ranges, use these formulas:

@SCALE,1

NewX = O1dMaxX, NewY = OldMaxY/.39

@SCALE,2

NewX = O1dMaxX*2, NewY = (OldMaxY/.39]*2

OldMaxX and OldMaxY are the largest X and Y values in the current screen.

A word about using @SCALE is in order. While it does compensate for the C128's non-symmetrical pixel, it has the drawback of slowing the rate of drawing. That's because the scaled points must be converted to the physical screen pixels. So don't use @SCALE 1 or 2 in situations where speed is important.

@PATTERN allows you to specify one of the 192 structures as the current pattern. You can fill an enclosed area with the @PAINT command. @PAINT uses the currently defined pattern, if pattern mode has been selected in @DRWMODA, otherwise @PAINT uses the current color for a solid paint. @PATTERN allows you to create incredibly detailed patterned fills, and of course the

Chapter Three

(23)

pattern can be in color. Anything you can define as a brush can be used as a pattern. Just @STASH the screen area you want in memory as a brush, then use @BRUSHPATRN to make that brush into a pattern. For information on defining a pattern structure see the chapter on structures or the BASIC 8.0 ENCYCLOPEDIA.

You can also find out information on any specified pixel with the command @PIXEL. It allows you to tell if a given pixel is on, and what the colors are within its color cell.

When using the multicolor modes, it is easy to set the drawing, background and border color with the @COLOR command. It allows each to be any of 16 colors. Normally there is no different border color in the C128 RGB mode, but while in color mode BASIC 8.0 was able to get around that limitation. You can also use the @CLEAR command to fill the current screen or window with the colors of your choice, or even to fill it with one of 255 different simple patterned shades based upon the numbers 0-255.

Another graphic command is @CHAR. @CHAR allows you to write on the graphics screen using any character font you want. You can use the two built in fonts, or you can load a new font into a structure and use it. The characters can be enlarged up to 16 times in both the x and y directions. You can position the character string anywhere on the active drawing screen. If you are using the color modes you can specify the foreground and/or the background color of the character by inserting control codes in the string. You can also give the direction to print the character string. The direction can be left, right, up, down or combinations of them, like up and to the right. And there are

many special character control codes in addition to colors

Chapter Three

(24)

available. For example, you can underline, rotate, flip, mirror, reverse, pattern, complement, inverse and blank under the character string, just by including the proper control code. For details of the @CHAR command see the BASIC 8.0 ENCYCLOPEDIA (CHAPTER 10).

There are other commands used with the graphic commands that modify the way they are used. The commands are @DRWMODA and @DRWMODB. Each allows you to turn on and off different drawing modes. For example, you can use @DRWMODA to draw, erase or use the XOR (complement) mode when drawing pixels. You can turn on and off patterned drawing. You can merge a pattern with the screen, and set the clipping flag. (Clipping chops off the drawing commands when they leave the edges of the current screen or window.) You can also change from perspective to parallel 3D drawing modes with the @DRWMODB command as well as setting the Unplotlast and Unplotvertex flags, so that when drawing multiheight objects they have a more obvious 3D appearance. All the parameters of @DRWMODA and @DRWMODB are explained in the BASIC 8.0 COMMAND ENCYCLOPEDIA.

```
lO @WALRU6,0:REM 16K video ram system
20 @MODE,O:REM use 16K screens
30 @SCREEN,2,2:REM use 640 x 176 screen
40 @DRWMODA,1,0,0,0,0,0,0:REM define drawmodea using JAM1 color
50 @DRWMODB,O,O,O
60 @COLOR,O,8,O:@CLEAR,O:REM set colors, clear screen
70 FOR I=l TO 100:REM Define random lines and colors
```

```
80 Xl=INT(RND(1)*640)
90 Yl=INT(RND(1)*176)
100 Zl=INT(RND(1)*100)
110 X2=INT(RND(1)*640)
110 Y2=INT(RND(1)*176)
130 Z2=INT(RND(1)*100)
140 C=INT(RND(1)*15)+1
150 @COLOR,O,C,O
160 @LINE,X1,Yl,Z1,X2,Y2,Z2,1
170 NEXT I
l80 SLEEP 5:REM wait 5 seconds
190 @CLEAR:@TEXT:REM return to text mode
```

Chapter Three

(25)

CHAPTER FOUR

THREE. DIMENSIONAL GRAPHICS

Commands and Theory

Many of BASIC 8.0's graphic commands work in three dimensions, or 3 space. In order to increase the graphics power available, there are a number of special commands that effect how the graphics are drawn. But before discussing them, a short explanation of 3 dimensional graphics and its terms is necessary.

Three dimensional graphics means there are three coordinate axes used to define a point. In normal microcomputer graphics only 2 dimensions are used. They are labeled X and Y, where the X axis is horizontal to the screen and Y is vertical (up and down). In three space, a new axis is defined, Z. This Z axis is directed into and out of the screen. (See Figure 4.1)

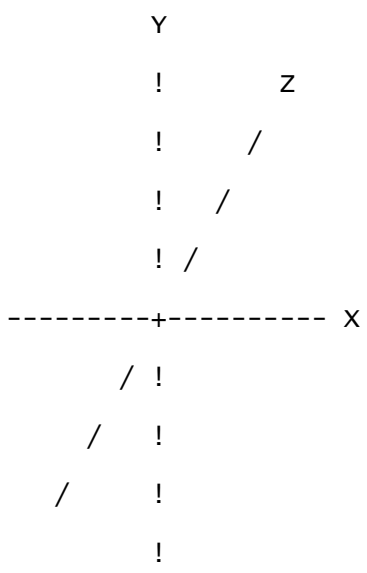

Figure 4.l 3D Coordinate Axis

There are two viewing methods commonly used in 3D graphics. One is parallel, where the Z coordinate is simply used to offset the point when converted to the 2D space of the screen. The other is perspective viewing, which uses a point in 3 space as

(26)

its vanishing point. Perspective makes objects drawn near to the viewer (Z values decrease) appear bigger, and objects drawn away from the observer toward the vanishing point (Z values increase) appear smaller. You can select perspective or parallel mode by using the @DRWMODB command. The vanishing point has no effect when using parallel mode.

When drawing in perspective mode, objects that approach the vanishing point appear to be smaller. If they reach the vanishing point they become a single point, and if they exceed it they become larger (to our eyes). By being able to define the

vanishing point (with the @ORIGIN command) you can control the degree of three dimensional perspective your images will assume.

Another aspect of 3 space that effects the way we perceive a 3D object is its origin. The origin is essentially the center of the universe to your object, line or point. When you rotate, you are rotating around the origin. The default center of rotation is the 3D point 0,0,0. Sometimes you don't want an object to rotate around this point, because, depending on your position in space relative to this default origin, you can end up with your object completely off the screen. If you wanted to rotate an object around its own center and stay in the same screen location you would have a problem. BASIC 8.0 removes this difficulty by allowing you to define any spot in 3 space as the center of rotation. To control where the origin is as well as the vanishing point, use the @ORIGIN command. For example, if your object was a cube, by setting the center of origin to the center of. the cube your object would rotate around its own center, and stay in the same spot while doing so.

In BASIC 8.0 you can make a 3D object or point appear

Chapter Four

(27)

different with parallel drawing mode set by using the @VIEW command. @VIEW sets the position of your eye around the Y axis. It has the effect of a simple rotation of your eye (not your data) around the Y axis, so when you draw you see your object at a different point of view.

In order to have complete rotation control, BASIC 8.0 uses the @ANGLE command. It allows you to set global rotation values that are used by all the drawing commands (except @CHAR), as well as the rotation sequence.

There are three different axes in 3 space, the X, Y and Z axes. You can rotate your objects in any of these three directions, both in the positive and negative directions. You can rotate in more than one axis at once, which results in a multiple transformation of the object. When rotating in several directions at once, the end results differs based on what sequence you rotate (XYZ, XZY, YXZ, YZX, ZXY or ZYX). An object rotated in the sequence XYZ will appear different from one rotated in XZY, even if the angles of rotation are the same. All of these parameters are definable with the @ANGLE command. When you draw with the graphic commands they respond to the currently defined @ANGLE setting. The default is @ANGLE,0,0,0,0 (no rotation). Once set to another value, all the drawing commands output their functions accordingly. For example, if you are drawing an arc, and the @ANGLE is defined to be a 45 degree rotation around the Y axis (@ANGLE,0,45,0,0), the resulting arc will appear to be rotated 45 degrees in the Y axis. The commands @DOT, @LINE, @BOX, @ARC, and @CIRCLE all will respond to the given @ANGLE. Most importantly, you can change the rotation angle as many times as your program requires. However, remember

that complex rotations will decrease the speed of your graphics (although only a tiny fraction of what they would be if mathematically rotated in basic).

Here is an example of a BASIC 8.0 program that draws a rotated line using the @ANGLE command to rotate it around the Z axis. Notice that the line definition is never changed, only the @ANGLE values change.

```
10 @WALRUS,O:REM 16 K VIDEO MODE
20 @MODE,O:REM USE FIRST 8 SCREEN DEFINITIONS
30 @SCREEN,O,O:REM MONOCHROME SCREEN 640 X 200
40 @SCALE,O:REM USE NORMAL SCALE
50 @DRWMODA,1,0,0,0,0,0,0: @DRWMODB,0,0,0: REM STANDARD MODES
60 @ANGLE,O,O,O,O:REM DEFAULT NO ROTATION
70 @0RIGIN,320, 100,0,320,100,200:REM ORIGIN IN SCREEN CENTER
80 @CLEAR,O:@COLOR,O,8,O:REM CLEAR SCREEN, SET COLOR TO RED
9O FOR AN=O TO 360 STEP 20 :REM CHANGE ANGLE BY 20 DEGREES
100 @ANGLE,O,O,AN,O:REM ROTATE LINE AN DEGREES AROUND Z
110 @LINE,200,100,0,440,100,0,1: REM DRAW LINE
120 NEXT AN
130 GETKEY A$:REM WAIT FOR A KEY PRESS
l4O REM CLEAR ALTERED ROTATION VALUES
150 @ANGLE,O,O,0,0
160 REM RETURN TO TEXT MODE
170 @TEXT
```

In addition to these 3D drawing commands, there are several designed to be used to generate SOLID 3D OBJECTS. These are based upon techniques developed for the C64 computer by Mr. Richard Rylander and first published in the May 1985 Dr. Dobb's Journal. Because of the 3D nature of our work, Mr. Rylander allowed us to convert his solids algorithms to the C128 for use in BASIC 8.0. We wish to thank him for his great work and generosity.

The Rylander Solid commands allow you to generate textured (halftone or random) solid images such as SPHERES, TOROIDS, CYLINDERS and SPOOLS. You have a great deal of control over clipping the objects, shading and texture. It cannot be

(29)

emphasized enough the incredible graphics possibilities of the solid generation commands.

The solid commands differ from the regular 3D graphic commands in that they are defined using the normal 2D X,Y system. You can not use a Z coordinate value when defining their positions.

Perhaps the most interesting of the Rylander shapes is the sphere. It allows you to generate a solid spherical object, which can be used in many drawings or illustrations for quite a dramatic effect. The command is @SPHERE,X,Y,RADIUS. It will place the sphere at the given coordinate X,Y (remember these commands have no Z axis control). You can change the clipping, shading and lighting with the @STYLE and @SCLIP commands.

You can create a toroidal shape (somewhat donut shaped) with the command @TOROID. This allows you to define both the inside radius and the outside radius, as well as the view (horizontal, vertical or top) so the shape can be quite different looking with the same command.

Another of the Rylander shapes is SPOOL. A spool surface can be thought of as the inside of the toroid surface. Created with the @SPOOL command, it allows you to specify the inner and outer radius, and two viewpoints, horizontal and vertical.

The final Rylander solid shape is the CYLINDER shape. This will create a horizontal or vertical cylinder, with you defining its radius, halflength and view.

To make these even more useful, there are two command that can be used to define the characteristics of the solids in much the same way as @DRWMODA and @DRWMODB define the drawing commands.

Chapter Four

(30)

@STYLE sets up the shading (textured or halftoned), scaling (symmetrical or elongated) and lighting (normal or backlite). @SCLIP allows you to only draw part of the solid, and clip the rest. This makes it easier to combine the shapes into more complex solids.

While the solids commands do not respond to the 3D commands (@ANGLE, @VIEW etc.) they do respond to the @COLOR, @WINDOWOPEN/@WINDOWCLOSE and bitmap screen (@COPY, @STASH/@FETCH) commands.

Here is a simple example that creates a green textured sphere.

```
10 @MODE,0
20 @SCREEN,2
30 @COLOR, 0, 5, 0:@CLEAR,0
40 @DRWMODA,1,0,0,0,0,0,0:@DRWMODB,0,0,0
50 @STYLE,0,1,1
60 @SPHERE,320,200,50
70 GETKEY A$
80 @TEXT
```

An important point should be made about the 3D drawing and 3D solid commands. Your program, when it finishes up and exits, should set the ANGLE, VIEW, DRWMOD's, ORIGIN etc. back to the default values. Otherwise the next program will begin in these modes. Always finish with @ANGLE,0,0,0,0 and @DRWMODA,0,0,0,0,0,0,0. (See Appendix D)

Chapter Four

(31)

CHAPTER FIVE

MEMORY MANAGEMENT

BUFFERS, STRUCTURES AND OTHER DELIGHTS

BASIC 8.0 is able to utilize up to 10 different 64K banks of ram for storage. Two of these are ram banks 0 and 1, which are the two standard ram banks used by the C128 itself. The other 8 are made available when you install external ram via the 1700 or 1750 ram expansion cartridges. If you have added the 1700 ram expansion cartridge, which has 128K of ram, you have then added two addition banks, bringing the total available 64K banks to 4. If you have added the 1750 ram expansion cartridge it adds 512K additional ram, giving you 8 additional banks of 64K for a total of 10 banks.

Basic 8.0 allows you to access that ram by the use of a data structure system. Each of the 10 banks is referred to as a BUFFER. The BUFFER numbers are 0-9, where 0 and 1 are the standard C128 banks 0 and 1, and BUFFERS 2-9 are the external banks added with the ram expansion. You can also define where in the ram bank a BUFFER will begin and it's size in order to effectively use the system ram banks 0 and 1. Obviously you cannot just use all the ram in banks 0 and 1, as that would leave no room for programs and variables; yet very few programs use every byte in those banks. That leaves some for use as BUFFERS; but it is far better to have the external ram for structures, and leave internal ram for programs. For example, let's define a BUFFER in ram hank 1. Since this bank is also used for basic program variables you should adjust the variable start or end pointers so that your buffer doesn't corrupt your variables or

visa versa.

```
POKE 47,0:POKE 48,68:CLR
@BUFFER,1,1024, 16384
```

Chapter Five

(32)

This example uses ram bank 1 as the BUFFER, with it starting at decimal 1024 and is 16384 bytes long. It moves the start of variables in bank 1 up 16K to make room for the data buffer. That's enough for several fonts, or even a full screen monochrome picture. If you have the external ram expansion installed, you could use several of its ram banks instead. For example;

```
@BUFFER,2,0,65535 :rem use external ram bank 0, use all 64k
@BUFFER,3,0,65535 :rem use external ram bank 1, use all 64k
@BUFFER,4,0,65535 :rem use external ram bank 2, use all 64k
@BUFFER,5,0,65535 :rem use external ram bank 3, use all 64k
```

This gives you a total of 256k of ram to use as BASIC 8.0 STRUCTURE storage. You can use all 10 banks if you wish, or any combination. With the external ram installed and used for storage, you can use banks 0 and 1 just for your program and its variables, allowing for very large and graphically complex programs.

STRUCTURES

Once you have declared your BUFFERS, you can define structures to use that memory. There are 4 types of structures you can define. There is the PATTERN structure, used to define a bitmap pattern (of any size) and its' color (if any). There is

the LOGO type, which is used to generate one or more character strings. For example, you could define a structure that would print to a specific place on the screen, with a particular font, and any defined size. You can define many character fonts with the FONT structure, allowing you to have an unlimited number of different character fonts all at once. Finally there is the BRUSH type of structure, which is a rectangular area of bitmap (including color if it is present). Brushes allow you to pick up any area of the screen and save them to memory or disk, flip

Chapter Five

(33)

them, mirror or reverse them, then put them back on the screen where ever you want. If you are using color the color is saved and manipulated as well. You can even compress them to their smallest possible size for compact storage. Even more impressive is that a brush can be copied to another STRUCTURE as a pattern type. This allows ANYTHING you can place on the screen to be used as a pattern!

The data structure is defined using the commands @STRUCT, @SDAT, and @SEND. These commands work together to create different types of usable structures. The command @STRUCT includes the structure number (0-191) used to identify which structure when using later commands. Then the structure type (1-4), followed by a BUFFER number, and an address in the buffer. Once you have defined the type, use the command @SDAT (Structure DATa) to send whatever data is appropriate. You can use Basic 7.0 commands within the @STRUCT:@SDAT:@SEND block, for example FOR NEXT, DO WHILE LOOP etc., to generate data. You can use the @SDAT and @SEND commands to find the next available address in the buffer, as they return that information value when used. For

example, if you are using the @SDAT command to send pattern data, the next available address in the BUFFER is found by using @SDAT like this;

```
@BUFFER,2,0,65535 :rem external ram, 64k buffer
@STRUCT,2,1,2,0:rem struct 2,type pattern, buffer 2,address 0
AD = @SDAT,pattern data, pattern data, pattern data
rem AD = next address available
@SEND :rem finished defining pattern structure
```

A) Loading and Saving STRUCTURES

STRUCTURES come in four types, pattern, logo, character and

brush. You use the commands @LSTRUCT and @SSTRUCT to load and save all four types of STRUCTURES. When you want to load you define a BUFFER that can be used to store the data, then use @LSTRUCT to load it into memory. If you have a STRUCTURE in memory you want to save, then @SSTRUCT can be used to store it to disk. Here is a small example that loads the supplied 160 column character font into a STRUCTURE to be used with the @CHAR command.

```
@BUFFER,9,0,65535 : REM use external ram bank 9, all 64 k
@LSTRUCT,0,8,9,0,"FNT.160 COL":REM STRUCTO, DRIVE 8,BUFFER 9,
                                                Address 0,filename
AD=@SEND :REM AD now has the next address in BUFFER 9
```

This short program segment has now added a 160 column

character font to external ram memory as structure 0, where you can use it with the @CHAR command. If we wanted to add another font as well, the program would continue this way.

```
@LSTRUCT,1,8,9,AD,"FNT.COMPUTER":REM STRUCTl,DRIVE8,BUFFER
                                    9,Address AD,filename
AD=@SEND :REM AD now has the next address in BUFFER 9
```

We could now begin using the different fonts with the @CHAR command, indicating which one we wanted with it's STRUCTURE number. However, perhaps we want to load some BRUSH types also. We still use @LSTRUCT to load.

```
@LSTRUCT,2,8,9,AD,filename :REM STRUCT#2,DRIVE8,BUFFER 9,
                                    Address AD,filename
AD=@SEND :REM AD now has next available address in BUFFER 9
```

We now have easily loaded two fonts and a BRUSH in as user recallable STRUCTURES. We could have loaded LOGO or PATTERN types just as easily. The @LSTRUCT command allows you total control, for example, you might have created on the screen an object that you want to save as a BRUSH. Continuing our program above (still using BUFFER 9) we will @STASH it to a STRUCTURE, then save it to disk.

Chapter Five

(35)

```
OA=AD :REM Save starting address of this Structure
AD=@STASH,3,9,AD,0,0,100, 100, 1 :REM STASH area of screen
                                    0, 0 to 100, 100 to
                                    STRUCTURE 3 in BUFFER 9
                                    in compressed format. AD
```

```
                                    now contains the next
                                    address in BUFFER 9.
    @SSTRUCT,3,8,filename :REM SAVE STRUCTURE
    @SEND :REM Save the file
```

We have now saved the BRUSH as a disk file, but it is also still available in memory to be used. We could @FETCH it to the screen, use @CBRUSH (ChangeBRUSH) to flip, reverse or mirror it, or even use @BRUSHPATRN (BRUSH to PATTERN) to make a pattern out of it. Of course, since it is stored on the disk we can always reload it at another time.

B) PATTERN STRUCTURES

BASIC 8.0 has a very extensive pattern capability. You can define a pattern from 1 to 255 bytes wide, and up to 255 pixels deep. If you wish, it can also contain foreground and background information for each color cell in the pattern. Once defined, the pattern can be loaded to and from disk, and used to fill an enclosed area with the @PAINT command. It can also be used in the graphic drawing commands as draw pattern by setting the appropriate parameter in @DRWMODA. The syntax of the PATTERN STRUCTURE is:

```
    @STRUCT,4,1,9,AD:REM STRUCT44,pattern,BUFFER 9,Address AD
    @SDAT,2,8, 1,1 :REM 2 bytes X 8 pixels bitmap, 1 X 1 color
    @SDAT,255,255,128,0,128,0,128,0,255,255,0,128,0,128,0,128
      :REM That is the bitmap data for a brickwall
    @SDAT,206 :REM color (brown on lite grey), AD=next address
    AD=@SEND
```

STRUCTURE 4 is now defined as a PATTERN type. You can use it by specifying STRUCTURE 4 when you use the PATTERN command.

You can make small simple patterns, or very complex ones. The easiest way to make a pattern is to convert a BRUSH to a PATTERN

Chapter Five

(36)
with the command @BRUSHPATRN.

C) LOGO STRUCTURES

LOGO STRUCTURES are a special form of the @CHAR command. They allow you to print a number of different character strings out by a single command. So certain often used strings can be saved and loaded directly from the disk. A good example of a LOGO STRUCTURE is a menu of options. It could be stored as a BRUSH, but a LOGO requires much less memory because it is text data, not bitmap data. Here is an example of a LOGO STRUCTURE.

```
OA=AD:REM SAVE AD FOR NEXT EXAMPLE
LA=XXXXX :REM LA (LOGO Address in BUFFER 1)
@STRUCT,5,2,1,LA :REM STRUCT#5, LOGO, BUFFER 9, Address LA
@SDAT, 1,CHARSET STRUCT#,l4,16,0,2,2,2,13,"MENU OPTION 1"
@SDAT, 1,CHARSET STRUCT#,l4,32,0,2,2,2,13,"MENU OPTION 2"
@SDAT, 1,CHARSET STRUCT#,l4,48,0,2,2,2,13,"MENU OPTION 3"
@SDAT, 1,CHARSET STRUCT#,l4,64,0,2,2,2,13,"MENU OPTION 4"
@SDAT,O :REM Finished the LOGO
AD=@SEND :REM complete this STRUCTURE, AD is next address
@SSTRUCT,5,8,"LOGO.MENU":REM Save LOGO STRUCTURE
@SEND :REM save LOGO to disk
LA=AD:REM SAVE NEXT ADDRESS IN BUFFER 1
AD=OA:REM RESTORE AD FOR NEXT EXAMPLE
```

When defining LOGO STRUCTURES, the first parameter of the

@SDAT command should be either 0 or l. A 0 indicates you are finished, while a 1 means continue, I have more data to give. And, because the character string data in LOGO's are accessed by the CHAR command (in much the same way as a string variable) ALL LOGO structures must be stored in BUFFER 1 (system bank 1). Remember, LOGO structures in BUFFER 1 only!

D) CHARACTER FONT STRUCTURES

BASIC 8.0 has the capability of using virtually unlimited fonts. These fonts can be defined with any standard character editor used for the C64 or C128. They must be converted to the proper BASIC 8.0 format for a CHARACTER STRUCTURE with the utility program Font Converter supplied on your BASIC 8.0 disk.

Chapter Five

(37)

All this program does is add a small header file to the beginning of the font so BASIC 8.O can tell what kind of STRUCTURE it is. (For information on file formats see the Appendix.) CHARACTER STRUCTURES must be loaded into memory with the @LSTRUCT/@SEND command. After that they are available whenever you need them. Continuing our program examples above:

```
@LSTRUCT,6,8,9,AD,"FNT.TECH":REM STRUCT#,DRIVE#,BUFFER9,AD,name
AD=@SEND :REM Get the font, AD = next Address in BUFFER 9
```

E) BRUSH STRUCTURES

Perhaps the most useful type of STRUCTURE is the BRUSH STRUCTURE. This type allows you to @STASH and @FETCH a piece of the screen, any size, back and forth from memory. Of course you can also save it to disk, where it can be recalled as a BRUSH

with the @LSTRUCT, or even displayed directly with the @DISPLAY command. BRUSHES can also be manipulated with the @CBRUSH (ChangeBRUSH) command. This can make flipped, reversed or mirrored images of your brush. Or use the @BRUSHPATRN command to make a copy of a BRUSH as a PATTERN STRUCTURE.

```
@STASH,7,9,AD,0,0,100,100,0 :REM STASH screen 0,0-100,100 as
                             brush structure #7. Do not
                             compress the data.
                             AD=next BUFFER 9 address
@CBRUSH,7,0,0, 1:REM FLIP BRUSH IN MEMORY
@SSTRUCT,7,8,"BRUSH.TWO" :REM Prepare to save BRUSH
@SEND :REM Save To Disk as BRUSH.TWO
AD=@BRUSHPATRN,7,8,9,AD :REM Make Pattern Structure #8
@FETCH,7,0,100,0 :REM FETCH FLIPPED BRUSH TO SCREEN
```

A BRUSH can be loaded to a STRUCTURE and saved from a STRUCTURE. It can be returned to the screen with the @FETCH command, or loaded directly to the screen from disk with @DISPLAY. When loaded with the @DISPLAY command, the brush is returned to exactly where it was stored from, or you can specify the X and Y to load it to. Other areas of the screen are not affected.

Chapter Five

(38)

CHAPTER SIX

ADDITIONAL BASIC 8.0 COMMANDS

BASIC 8.0 offers a number of other interesting advanced commands. For example, if you have the extra 64K video ram

installed for the 8563 video chip (the new C128D cones with it standard, older C128's do not) you can make use of the @SCROLL command. @SCROLL allows you to move in one of eight directions at any of 255 speeds and up to 255 units. The directions are up, up/right, right, down/right, down, down/left, left and up/left. The speed is from 0-254, where 0 is the fastest. Each unit of speed adds a 0.0255 second delay between increments. The X scroll unit is on the byte level, while the Y scroll unit is in pixels. Or, by beginning your parameter list for SCROLL with 255, you can move directly to a specified scanline. Use @SCROLL with virtual screens (screens wider than 640 or higher than 200) to move around.

One very important aspect of any graphics system (language or application) is in getting what is on your screen to a printer. In order to make it easy, we have created installable printer drivers that allow you to create a hardcopy of the screen. The programmable command to do so is called HCOPY, and just as BASIC 8.0 is very flexible, so is it's hardcopy capabilities.

There are 11 different printers that are directly supported by BASIC 8.0. The default (built in) is EPSON, but if your printer is different merely load in one of the other drivers. (See chapter on printers for instructions on installing a printer driver.) Since the screen size you can use with BASIC 8.0 can differ, and also since many printers have different dot densities and different horizontal resolutions, HCOPY was designed to allow

you to print in a variety of different ways. The syntax of HCOPY is:

@HCOPY,Secondary Address,Height (l-4),Density (1-7),Rotation Flag

The secondary address for printer bitmap mode differs from printer to printer. (See chapter on BASIC 8.0 printer drivers for secondary address of your printer.) The height flag is how large to make your printout, with 1 the smallest and 4 very tall. The density parameter allows you to change the dot density of your printer, if possible. (Not all printers have more than one density, but some like the EPSON and PANASONIC have seven different settings. By setting to a larger density, you can print an image smaller than it would have been in a lower density.) Finally, the rotation flag allows you to rotate the printed image 90 degrees. With some printers, like the Commodore MPS 801 or Seikosha 1000, the image is always rotated because these printers cannot even print 640 dots per line (the smallest horizontal screen resolution).

To make it possible for even more printers to be supported, Chapter Eleven lists the assembly language source code for every printer driver. Experienced programmers can use these as a base to develop even more printer drivers.

MOUSE, JOYSTICK and POINTERS

BASIC 8.0 also offers mouse and joystick control, along with a very precise graphic pointer function. You choose between the mouse (the 1351 proportional mouse, NOT the 1350 joystick mouse) and a joystick with @MOUSE,1,DEVICE,X,Y[,Joystick speed]. This activates the IRQ controller reading routine. As long as it remains on it will then constantly update (internally) the position of the mouse or joystick. You can then find the current

(40)

X and Y value with the @MOUSE function. It will return the X or Y value, where it can be used to position the arrow with the @PTR,1,X,Y,PTR DEF# command. For example;

```
@MOUSE,1,0,320,100     starts the mouse reader using the mouse
                       and positions it at 320,100
@MOUSE,1,1,320,100,2   starts the mouse reader using the joystick
                       and positions it at 320,100 moving 2 pixels
@MOUSE,O               turns off the IRQ mouse reader
MX=@MOUSE,2,0          returns the current mouse x position to MX
MY=@MOUSE, 2, 1        returns the current mouse y position to MY
```

The pointer function has three basic forms. One turns off the pointer while the other two are used for positioning.

@PTR,O Turn off the pointer and restore the old image.
@PTR,1,MX,MY,PTR DEF (O-15) Turn on at NX,MY using the pointer definition given and "floats" over bitmap.
@PTR,2,MX,MY,PTR DEF (O-15) Turn on at MX,MY using the pointer definition given and draws on the bitmap

Programming the mouse/pointer system is quite easy. First you turn on the proper mode (mouse or joystick) at the location you want to start. Then just use the @MOUSE,2 command to find the X and Y positions, and set the pointer position to those locations. Here is a small example program that will initialize the mouse, and move the pointer as you move the mouse.

```
10 TRAP 140:REM PRESS RUN/STOP KEY TO EXIT PROGRAM
20 @WALRUS,O:REM 16K MODE
30 @MODE,O:REM USE MODE 0 SCREENS
```

```
40 @SCREEN,O:REM USE SCREEN 0, MONOCHROME
50 @COLOR,0,4,0:REM SET COLOR
60 @CLEAR,O:REM CLEAR SCREEN
70 @DRWMODA,1,0,0,0,0,0,0:@DRWMODB,0,0,0:REM JAM 1 MODE
80 MX=320:MY=100:REM INITIAL X,Y
85 OX=MX:OY=MY:REM SAVE VALUES
90 @MOUSE, 1,0,MX,MY:REM USE 1351 MOUSE
100 @PTR, 1,MX,MY,O:REM USE ARROW POINTER (0) AT MX,MY
110 MX=@MOUSE,2,0:MY=@MOUSE,2,1:REM FIND NEW MX,MY
115 IF MX=OX AND MY=OY THEN GOTO 110:REM ANY MOVEMENT?
120 @PTR,1,MX,MY,O:REM REP0SITION POINTER AT NEW MX,MY
125 OX=MX:OY=MY:REM UPDATE OLD VALUES
130 GOTO 110:REM CONTINUE TO CHECK FOR MOVEMENT
140 @PTR,O:@MOUSE,O:REM TURN OFF POINTER AND MOUSE READER
150 @TEXT:REM RETURN TO TEXT MODE
```

Another useful command is @FLASH. This command allows you

to reverse a rectangular area of the screen, from 1 to 255 times. It is very useful for bringing something to a user's attention. If you are in monochrome mode, it reverses the bitmap. If you are in color mode only the color information is reversed (which is very fast compared to the bitmap flash).

Another BASIC 8.0 command that is both exotic as well as useful is the command @ZOOM. This command allows you to fetch an uncompressed brush from memory as an enlarged version of the brush. And it can do so in a number of different sizes.

Since BASIC 8.0 is a bitmap graphics language, it offers very little for use in 80 column text mode. However, there are

two text commands. The first is @TEXT, which returns your program to text mode from graphics mode. It also re-initializes some of the various parameters back to their default values. For example, @ANGLE is set to 0,0,0,0 and @SCALE is set to 0.

The other text mode command is @FONT. This command allows you to set the two character sets available in text mode to any font you wish. Just load in the font as a structure and specify in the @FONT command which structure to use as a font. It is downloaded to the 8563 chip, where it becomes available in text mode. Once a @TEXT command has been issued, the standard character fonts are restored, so it will be necessary to re-issue the @FONT command.

Chapter Six

(42)

Chapter Seven

Making A Distributable Disk

The Run-Time Module

The ability to give away (or sell) programs written with BASIC 8.0 centers around the RunTime Library supplied with your BABIC 8.O system. When you first booted your original disk, you were offered the option of making a Basic 8.0 Run Time Disk. If you. have not yet done so, you should get a blank disk and create one. If you have already done so, make a backup of the Run Time Disk. It is this disk that you are allowed to distribute.

The Run Time Disk is a autoboot disk, meaning it will load and run when you boot the computer (if the disk is in the drive). The normal startup sequence will install the BASIC 8.0 Run Time System, then load and run the WIOS WORKBENCH (see Chapter Nine). To use your program under the WORKBENCH, save your program on the Run Time Disk

with a filename that starts with the letters B8. and is followed with a filename less than 12 characters. For example, if your program is called 'CHARTMAKER' you would save it with the name

B8.CHARTMAKER

on the WORKBENCH disk. When the WIOS WORKBENCH is automatically ran on bootup, it will be able to find the CHARTMAKER program.

However, you may want to have your program be loaded directly, bypassing the WIOS WORKBENCH. In that case, you will need to change one of the programs on the Run Time Disk.

To do this, insert your working copy of the Run Time Disk. Save your program to this disk. Then load the program called:

STARTUP

and type LIST and press return. You will see the single line:

10 RUN "WORKBENCH"

To change the bootup sequence in order to autoboot your

Chapter Seven

(43)

BASIC 8.0 program, type the following line

10 RUN "filename" (RETURN)

(the (RETURN) means press the RETURN KEY). The filename should be the name of your program.

To make sure the new line has been entered correctly, type LIST and then RETURN. You should see

10 RUN "filename"

as the only line in the program. If it is wrong, retype the line as instructed above and check it again. If it is okay, then type:

SCRATCH "STARTUP" (RETURN) The C128 will ask:

ARE YOU SURE? Y (RETURN) You reply Y and press return

You have now erased the startup file. At this point you

need to save the newly modified (with your filename in place of WORKBENCH) program as STARTUP. To do so type:

DSAVE "STARTUP" (RETURN)

You have now created a BASIC 8.0 Run Time Disk that will automatically load and run your program. As a registered owner of BASIC 8.0, you may give away this Run Time Disk, or sell it if you wish. Remember, this is the only way you may give away programs using the BASIC 8.0 language!

Chapter Seven

(44)

Chapter Eight

BASIC PAINT

The 80 Column Color Paint Program

Along with all the added programming capabilities of BASIC 8.0, you also get a complete graphics application. This program, written completely in BASIC 8.0, is called BASIC PAINT ((c) 1987 WALRUSOFT). BASIC PAINT has several very important uses. First and foremost, it is meant to be a stand alone graphics paint program, complete in every detail. You can use it simply to draw or paint pictures for your amusement or as a means of artistic expression. Secondly, it is meant to be an accessory to the BASIC 8.0 programmer. You can use it to create pictures, brushes, icons or patterns that can be used later in your own BASIC 8.0 programs. And finally, BASIC PAINT is meant to be an example of what you can do with the programming power offered by BASIC 8.0.

As mentioned, BASIC PAINT offers all the standard computer paint package graphic functions (freehand draw, line, box, circle and polygon) found in most commercial programs. You can add

text, fill areas (with solids and patterns) and print your pictures. But BASIC PAINT goes further than that. You can add 3D solids to your pictures, cut and paste to and from a clipboard, zoom in for precise editing, and even lock your image for later recall (lock acts like an oops function). You control the program with a pointer using either a mouse or joystick, you can change your pointer to eight different shapes, and can set the drawing modes in many combinations of draw, erase, complement and pattern. You can also draw using thick lines or very thin lines, with the exact size being up to you.

BASIC PAINT works in all five graphic modes supported by

Chapter Eight

(45)

BASIC 8.0. You have your choice of monochrome and the four color resolutions (8 x 16, 8 x 8, 8 x 4 and 8 x 2). And just like BASIC 8.O, BASIC PAINT supports the 1700 and 1750 external ram expansion modules. Even more exciting, if your C128 has the full 64K of video ram you can use BASIC PAINT to draw on several size virtual vertical screens.

BASIC PAINT is designed to be simple to use, even intuitive In nature. It is essentially a mouse driven program, controlled By the mouse, the left and right mouse buttons and graphic icons. (Icons are pictures that represent a particular function.) By pointing to a icon and pressing the left mouse button (or the joystick button) you select that icon's function. At times you will want to indicate that you are finished doing something, or perhaps you will want to change your mind and return to the previous option. This is accomplished by clicking the right mouse button (or the ESCAPE key if you are using a joystick). This left button (for selection) and right button (for ESCAPE) system becomes very natural after a while. In fact, if you are

using a mouse, the only keyboard input is when you are saving an image or if you want to shear a box. (If you are using a joystick the keyboard ESCAPE key is used instead of the mouse right button.)

To use BASIC PAINT, simply insert the BASIC PAINT Work Disk In your C128 and boot it, either by turning on the computer or (if it is already on) by pressing the reset button. The program will autoboot, and you will quickly be presented with a special type of menu called a requestor (see Figure 1). This requestor menu is requesting information on how your computer is equipped, BASIC PAINT will work best with your system. (You should

always keep your working copy of the BASIC PAINT disk in drive 8 when using it as it will often need to load requestors and menus from the disk.)

```
--------------------------------------
! Press M for Mouse   J for joystick !
! Video RAM:  O  16k    O 64k        !
! System RAM: O 128k  O 256k  O 640k !
!         Continue                   !
--------------------------------------
```

Figure 1

The first thing to do is indicate what type of controller you are using (see figure 1). If you have the 1351 mouse (inserted in port 1 only), press M. If you are using a joystick (inserted in port 2 only), press J. Once you have done so, you will notice an arrow pointer appear on the screen. You can now use your chosen control device to move this pointer, and the

select button (left mouse button or the joystick button) to indicate your choices to the computer.

Next tell the computer how much ram your 80 column video chip has installed. If your computer is a standard C128 and has not been modified, you have 16K. That is the default selection. If your C128 has had the 64K video ram installed (the C128D comes with the extra ram), point to the 64K box and click the select button (left mouse button). Notice how the box becomes black. This indicates you have now selected 64K of video ram. (WARNING! If you do NOT have 64K, it does no good at all to make this choice and choosing it will cause the program to work incorrectly.)

Next, choose the amount of system ram your C128 has

Chapter Eight

(47)

installed. If your computer does not have any extra ram you have 128K, the default. If you have the 1700 expansion cartridge you have a total of 256K, and if you have the 1750 cartridge you have a total of 640K. Point to the appropriate box and click the select button.

If you have made a mistake and selected the wrong options, you can change them quite easily by simply pointing to the correct box and clicking the select button. Once you have the proper settings, point to the CONTINUE box and click the select button, and these values will be used by BASIC PAINT during this session.

Once you have selected the CONTINUE box, BASIC PAINT will display a new requestor menu. Depending on your previous choices, you will be offered a selection of graphic screen type to choose from.

```
! O 640 x 200 monochrome   !
! O 640 x 192 color 8 x 16 !
! O 640 x 176 color 8 x 8  !
! O 640 x 152 color 8 x 4  !
! O 640 x 104 color 8 x 2  !
!      Continue            !
----------------------------
```

Figure 2

If your system has 16K of video ram, the second requestor menu (Figure 2) allows you to choose one of five different screens. These are the same screens allowed in MODE 0 of BASIC 8.0 They differ on the size of the color cell and the total size of the screen. (See Chapter Two for more information on

screens.) Choose the screen you wish to work with by pointing to the proper box and clicking the select button, then click on the CONTINUE box. You will then proceed to the main drawing screen.

```
--------------------
! O Monochrome    !
! O 8 x 16 Color  !
! O 8 x 8  Color  !
! O 8 x 4  Color  !
! O 8 x 2  Color  !
!   Continue      !
--------------------
```

Figure 3

If your system has the full 64K of video ram, you are also presented with a requestor menu (Figure 3) asking for the screen type, however no screen sizes are listed. Choose the screen type

you want, you will then be presented with a third requestor menu that asks for the actual size of the screen you want. These third menus (see Figure 4-8) offer screens greater than 200 pixels high, although only a viewport of 640 x 200 pixels is visible at one time. When you are drawing the screen will automatically scroll up and down to keep up with the mouse.

```
---------------   ---------------   ---------------
!  Monochrome !   ! 8 x 16 Color!   !  8 x 8 Color!
! 0 640 x 200 !   ! 0 640 x 200 !   ! 0 640 x 200 !
! 0 640 x 300 !   ! 0 640 x 400 !   ! 0 640 x 300 !
! 0 640 x 400 !   ! 0 640 x 600 !   ! 0 640 x 400 !
! 0 640 x 500 !   ! 0 640 x 700 !   ! 0 640 x 600 !
! 0 640 x 600 !   !  Continue   !   !  Continue   !
! 0 640 x 800 !   ---------------   ---------------
!  Continue   !
---------------
```

Figures 4-6

```
---------------   ---------------
!  8 x 4 Color!   !  8 x 2 Color!
! 0 640 x 200 !   ! 0 640 x 200 !
! 0 640 x 300 !   ! 0 640 x 300 !
! 0 640 x 400 !   ! 0 640 x 400 !
! 0 640 x 600 !   ! 0 640 x 500 !
!  Continue   !   !  Continue   !
---------------   ---------------
```

Figures 7-8

Chapter Eight

(49)

Figure 4-8

At this point the screen will display the WALRUS logo screen

And indicate the amount of video ram you have chosen. It then clears and the BASIC PAINT drawing screen is created.

<<irreproducible figure>>

Figure 9

This screen consists of the drawing area and a strip of icons (pictures) at the top of the screen (Figure 9). Each icon represents one or more graphic functions, and they can be selected by pointing with the arrow and clicking the select button. When additional choices are required, submenus will appear. You can make the additional selections, or use the right button to ESCAPE back one step. Once you have made your selection, the icon strip disappears and the screen is made available for your drawing activity. When you are finished with hat function, click the right button of the mouse to escape back to the menu(s) (or use the ESCAPE key if you are using the joystick). Here is a description of each function.

FREEHAND DRAW

To draw in a freehand mode, move the pointer to the first Icon in the icon strip. The wiggly line represents drawing. With the pointer on the icon, click the select (left mouse) button. The icon strip will disappear, leaving the entire screen available for drawing. To draw hold down the left mouse button and move the mouse. A line will follow the pointer's tip. When you wish to stop drawing, let off the button. You can then move

to another location and start drawing again, or click the right mouse button (or ESCAPE key if using a joystick) to restore the icon strip. You will notice that the lines you have drawn are

black on the grey screen. These are the default colors. You can change them with the color icon (be sure and set the JAM options in the drawmode icon properly).

LINE

The second icon represents the line function. To draw lines, point to this icon and click the select button on your controller. As before, the icon strip disappears. Point to where you want the line to start and click the select button. The pointer will disappear, leaving you with a flashing line. As you move the mouse this line will follow you, a process called rubber-banding. Move the tip of the line to where you want it to end, and click the select button. The line will quit rubber-banding, but will continue to flash. At this point you can click the select button to accept this line, causing it to be drawn in the current color and mode, or if you decide it is not what you want you can click the right mouse button and it will disappear. You can then either draw another line, or click the right button and escape back to the icon strip. You can also use the right button to escape while actually rubber-banding the line. You will notice that when you escape while rubber-banding the pointer returns to the exact spot where you started the line. And when you have actually drawn a line, it starts exactly where the line ends.

BOX

Box allows you to draw rectangles and works very much like the line function. Point to the box icon and click the select

button. The icon strip disappears, and you are free to move the

pointer to any spot on the screen. When you have the pointer where you want one of the corners of the rectangle to be, click the select button. Just as with line the pointer goes away, only this time you have a box that rubber-bands, not a line. If the starting location was not correct, click the right button to escape and start the box again, otherwise move it's corner to create the size and position you want. When it is just as you want it, click the select button and again you have a flashing box that is waiting for the select button to accept and be drawn or the escape button to cause it to be erased. However, box has a couple of extra features at this point. You can press the X or Y key, and this will allow you to shear the flashing box in the indicated direction. (Shearing means to move one side while holding the opposite side still.) This allows you to create more complex shapes. Once you have begun shearing you have the same options as in normal box mode, that is to select or escape. Once selected you can still escape and start over as before. When you have finished drawing your boxes you can use the right button to escape back to the icon strip.

POLYGONS

<<<irreproducible figure>>>

Figure l0

The circle icon represents the polygon menu. Select this icon and a new submenu (Figure 10) appears. This submenu has six icons, circle, ellipse, triangle, diamond, hexagon and octagon. You can select one of these by pointing and clicking at the

desired object, or use the right button to escape back to the

icon strip.

If you chose circle, the icons disappear and you are free to move anywhere in the screen. Place the pointer on the spot where the center of the circle is to be and click the left button. The pointer is replaced with a rubber-banding line. This line represents the radius of the circle, which is the distance from the center of the circle to its edge. When you have the radius the size you want, click the left button and the circle will be drawn over and over. If it is correctly sized and placed, click the left mouse button, otherwise use the right button to escape and start over. You can repeat this as often as you wish. When you are finished, use the right button to escape back to the polygon submenu, where you can select another shape or escape again back to the main icon strip.

The other five shapes (ellipse, triangle, diamond, hexagon and octagon) work somewhat differently. When they are selected, you again point to where the center is to be and click the left button. Only here, instead of a single line you get two lines. These represent the two different radius required for these types of shapes, the X radius and the Y radius. As you move the mouse (or joystick) you will notice they each change in length. With a little practice you can quickly become quite skilled at using this technique for these complex polygons. And as always, use the left button to select and the right button to escape back to the polygon submenu and main icon strip.

SOLIDS

Chapter Eight

(53)

<<<irreproducible figure>>>

Figure 11

The solids icon also represents a more complex submenu. When the solid icon is selected, a submenu (Figure 11) is presented that offers eight different shapes and two drawing styles. The shapes are sphere, horizontal, vertical and topview toroids, horizontal and vertical spools and horizontal and vertical cylinders. The two style options on the right side of the menu represent the type of texture (random or halftone). You change the texture freely from the menu by pointing and clicking with the left selection button.

Sphere is controlled just like circle. Point to where the center is to be and click the left button. The pointer is replaced with a single line, (the radius). Move the line to make the proper radius, then click the left button. A beautiful solid sphere will be quickly drawn. Once it is finished, you can create another or click the right button to escape back to the solids submenu.

The other solids use the two line radius method. In the case of the horizontal and vertical spools, cylinders and toroids, they are fairly straight forward and easy. The topview toroid also uses two lines, but in this case they represent the inside radius and the outside radius. It is possible to select it such that the outside radius is smaller than the inside, an obvious impossibility. In that case the pointer appears as a question mark for a couple of seconds, meaning essientially "Say What?". Practice using the solids for a while and their use will

become quite clear and easy.

Use the right button to escape to the submenu, where you can select a different shape or escape again to the main icon strip.

CUT

The icon shaped like a pair of scissors is used to cut a rectangular area of the screen to the clipboard area, where it can be copied (pasted) to another location, or saved to disk as a brush (from the disk menu), or even used as a pattern for the fill function. CUT does not erase the area you have selected.

To cut an area, select the cut icon and position the pointer at one corner of the desired rectangle and click the left button. The pointer is replaced with a rubber-banded box, which you then use to enclose the area you want to put in the clipboard. Again use the left button to select, and the box will be adjusted to the boundaries required by the graphic mode you are in (color modes always round up and down to the edges of the color cell, and the rectangle must always begin and end on an even byte boundary. For more information on this see GRID mode in the drawmodes section.) At this point you can accept by clicking with the left button, or reject and escape with the right button. Once you have selected, you can then escape back to the icon strip, or replace the current contents of the clipboard with a new object. You can only have one item in the clipboard at a time.

PASTE

The paste jar icon is used to paste (copy) the contents of the clipboard back on the screen. To use point to the paste jar

icon and click the left button. Then point to the position where you want the upper left corner of the clipboards object to be and click the left button to paste it on the screen. You can repeat this as often as you wish, making many copies of a single object.

When you are finished, click the right button to escape back to the icon strip.

When pasting while in the color modes, keep in mind that the color memory of the object has been stored in the clipboard as well as the bitmap. So if your pasted object overlaps upon an area with a different set of colors than that found in the clipboard, the old screen colors in that area will be replaced with those of the stashed object.

TEXT

Choosing the text icon loads another submenu. This one offers a number of different fonts, as well as allowing you to set the height and width of the font you are using. To select a font, point to the one you want and click the select button. It will be loaded into the memory of the computer where it can be used. To change the height, point to the letters Ht and click once. Then click on the up or down arrows to change the height. To change the width, point to the letters Wd and click once. Again, the arrows are used to increase or decrease the width of the character.

If you have selected the special 160 column font, set your width to 0. (This is the only time the width should be less than 1.)

When you are ready to type your characters, point to the word DRAW and click once. The menus disappear, and you can position your letters with the mouse pointer or with the keyboard

cursor keys. Wherever the pointer rests is the top of the character cell. You may type whatever you wish. Pressing RETURN moves the cursor down one line and flush left on the screen (if

there is enough room to do so). To exit this mode press the escape button. Remember, the text responds to the draw modes DRAW and COMPLEMENT in the DRAWMODE submenu.

FILL

The FILL icon is used to paint an enclosed area with either a solid color or the current pattern, depending on the drawmode in use at the time of the fill action. To fill just point at the icon and click with the select button. The icon strip disappears and the pointer can be moved to any spot on the screen. Point to a spot within the area to fill and click the select button. (Remember to only fill a completely enclosed area, as it will leak out and fill the whole screen!) The area will then be filled. When it is finished you can repeat it again, or escape back to the main icons.

ZOOM

The ZOOM icon (it looks like a magnifying glass) allows you to zoom into a small area for editing. Point to it and click on the select button. The icon strip is removed and you can then move to the general area you wish to edit. Once there click the select button and a box will appear. You can then move the box to enclose the exact area you wish. Click the select button again, and a ZOOM window will appear, and it will fill with the magnified image of the smaller box's contents.

Once it is filled you can move the mouse to the points you wish to edit and change them by clicking with the select button. If the pixel is already on it is turned off, if off it is turned

on. The dot will be in the current color and drawmode. To

change these there are icons to change color and drawmode outside the ZOOM window. Once you are finished editing, use the escape button to close the ZOOM window. You can then ZOOM into another area, or escape back to the main icon strip. NOTE: ZOOM does not work in 8 x 16 mode!

DRAWMODE <<<irreproducible figure>>>

Figure 12

The DRAWMODE icon represents a second submenu (Figure 12). Once selected this menu appears and you can select the functions You want. One set of functions is called the COLOR JAMS. These Set the various JAM modes. Jam0 uses the color information found On the screen, and it ignores your choice of colors in the icon strip. Jaml uses your foreground color choice, but ignores your background color. Instead it uses the background colors it finds in the area you are drawing in. JAM2 uses both of your current color choices, foreground and background.

The other set of modes concern the type of bitmap drawmode You wish. These are DRAW, ERASE, COMPlement, Pattern, Trail and GRID.

Draw is the normal drawing mode. This simply means you want To draw in normal mode using a solid line.

Complement mode produces a reverse image. The points that

are already off are turned on, those on are turned off. (You will notice that the pointer also responds to this mode.)

Erase sets the mode to turn off all points that you draw on. Use this mode to remove (unpaint) an area. If you want you can change the pointer from the arrow to another definition (from the main icon strip) in order to make a larger eraser.

Pattern mode turns on and off the pattern drawmode. When off all drawing and filling is done in solid mode. When on the current pattern is used when drawing.

Trail mode is used when in freehand drawing. Instead of drawing it leaves a trail of the current pointer definition. Trail always uses the JAM0 mode, regardless of the settings in the JAM options.

Grid sets the drawing mode to a grid. The grid is always 8 pixels wide, and is as tall as the color cell size. This is very useful when cutting and pasting, as it allows you more precision when performing these functions.

One addition feature is found in the DRAWMODE menu. This is the line height. You can change the line thickness by pointing to the up or down arrows, changing the line size found in the box.

LOCK

Lock allows you to store a copy of the screen in memory (if you have the available memory free in the form of the 1700 or 1750 ram cartridges) in case you wish to restore it later after making some change you don't like. Once a screen is locked, you can restore it by choosing the CLEAR SCREEN option.

UNLOCK

Unlock turns off LOCK mode, allowing you to completely clear

the screen.

DISK

The DISK icon loads a submenu that makes available many disk functions, as well as the ability to convert brushes in the

clipboard into patterns.

LOAD PICTURE	Loads a directory of available brushes to be loaded as pictures. Just point and select, or return to the MAIN MENU.
SAVE PICTURE	An input box opens where you type in a filename to save the current picture to the currently selected drive. The name should start with 'PICT.'.
LOAD BRUSH	Load a directory of brushes that can be loaded into the clipboard as brushes, or you can return to the MAIN MENU.
SAVE BRUSH	An input box opens where you type in a filename to save the current clipboard to the currently selected drive. The name should start with 'BRUS.'.
LOAD PATTERN	Load a directory of patterns that can be loaded into the pattern clipboard, or you can return to the MAIN MENU.
SAVE PATTERN	An input box opens where you type in a filename to save the current pattern clipboard to the current drive. The file should begin with 'PATR.'.
BRUSHTOPATTERN	This copies the contents of the brush (CUT & PASTE) clipboard to the pattern clipboard, where it can be used as a pattern (if pattern is set in DRAWMODES) or it can be saved as a pattern using SAVE PATTERN.
DIRECTORY	Loads a directory of all files on the current disk.

CHANGE DEVICE Change your disk drive from drives 8-12. CAUTION: BASIC PAINT requires the disk with it's files and requestors to always be in device 8. By changing the device, you only change the input./output of user created files (brushes, pictures and patterns). When a menu requestor is needed it will still be loaded from device 8.

CHANGE DRIVE Switches between drive 0 and 1 for users of dual drives. BASIC PAINT menu requestors will always be loaded from drive 0.

PRINT

The PRINT icon displays a submenu that requests several

types of information. It asks for the HEIGHT of the printout, which can be up to 4. It allows you to set the DENSITY of the printout from 1-7. You can set the hardcopy to be rotated or unrotated, and you can set the secondary address to whatever your printer (and interface) requires for graphics screen dumps. And it allows you to switch between the various printer drivers. (If you change the printer type the appropriate driver will be loaded from disk and installed.) When you have the settings you want press the ESCAPE button and you will be presented with a small requestor that simply says PRINT. If you still want to print, point at it and click the select button. (Make sure you have your printer hooked up and turned on.) If you wish to change your mind click the ESCAPE button and you will abort the printing

process. The next time you return to the PRINT submenu it will remember your last settings.

See Chapter 11 on printers for information on your specific printer.

COLOR

Colors are set by the FG/BG color box. To change these colors, point at the top of the box (for ForeGround) or the bottom (for BackGround) and click the select button. If you are using monochrome, the entire display will change as a result. If you are in color mode, only the contents of the color box will reflect the new colors. However, if you draw (while in JAM1 or JAM2 DRAWMODES) you will use these selected colors.

CLR

The CLR icon is used to clear the display. If in color mode it clears the image using the current colors for foreground and background. However, if the screen has been LOCKed, clicking on

clear will simply restore it to the LOCKed image.

EXIT

The EXIT icon will open a small window in the upper left side of the screen. You may exit the program completely start it again (perhaps to draw in a different mode). Or, if you change your mind you can click the ESCAPE button and return to drawing.

****

Finally, if you are at the main icon strip there are a couple of additional features you can access from the keyboard.

1) If you are using a joystick, you can change your drawing

speed by pressing the +/- keys. (The faster you draw with the joystick, the coarser the resolution.)

2) You can change the pointer to one of eight definitions by pressing the ALT key and one of the keys 0-7 (at the same time). There are several interesting shapes, some of which you may prefer to the arrow.

If your pointer seems to disappear when it is inside a solid color area, remember it is NOT a sprite. It shares the same color memory as the image it rests upon. To make it always visible, set the draw mode to COMP. However, when drawing the complement mode draws on areas where the bitmap is empty, and reverses where it is already on. COMPLEMENT mode is not always the best choice when, drawing, but it does offer some interesting effects.

If you have questions about BASIC PAINT's screens and modes, see Chapter 2 for further information.

Chapter Eight

(62) Chapter Nine

WIOS WORKBENCH

And Other Supplied Utilities

WIOS WORKBENCH

The WIOS (Walrusoft Icon Operating System) WORKBENCH is really nothing more than a BASIC 8.0 program. It uses a few of the new commands, along with some standard BASIC 7 commands, to emulate the increasingly popular graphics user interface. It will load and run any program written in BASIC 8.0 that starts with the letters B8. and are followed by a filename that does not exceed 12 characters.

The WIOS WORRBENCH supports up to 4 disk drives, including the 1541, 1571 and 1581 drives. When it is first run it checks to see how many disk drives are present, then puts up on the

screen a simple requestor that asks if you are using a mouse (port 1) or a joystick (port 2). Press the appropriate key (M or J) and you will then see one or more disk icons labeled DRIVE N (N is 8-11). The arrow pointer is also active. By moving the control device (mouse or joystick) the pointer will move. To see the contents of a drive, move the pointer to the drive you want and press the left mouse button (or the joystick fire button). (Make sure there is a disk in the drive!)

Once you have pressed the button, a window will open and some brief disk activity will occur. If any files are found (that start with BB.XXXXX) the name of the file and a small icon that looks like a flowchart is shown in the window. A total of ten files can he seen in the window at one time. If there is more than 10 files, you can see t.hem by putting the arrow pointer on the little scroll arrows (at the bottom of the window) and clicking the left mouse button (or joystick fire button). If

Chapter Nine

(63)

there are any more files, the screen will shift and the new files be displayed. To move back the other direction, click on the other scroll arrow. If you see the program you want, just point to it and click the button and it will load and run.

If the file you want is not there, point to the close gadget In the upper left corner of the window (NOT THE SCREEN!) and click the button. This will instantly close the window. You can point to another disk icon, or remove the disk from the drive and insert another and repeat the process.

If you want to exit the WIOS WORKBENCH, use the close gadget In the upper left corner of the screen. Clicking on this gadget will close the screen and return to text mode. If you want to run it again, insert the disk with the WORRBENCH and type:

RUN "WORKBENCH"

The WORKBENCH program, like many of the supplied programs written in BASIC 8.0, is meant to be both useful and instructive. It is a real (although somewhat limited) graphic interface. It allows a user to load BASIC 8.0 applications quickly and easily, even if they don't even know how to find a directory by the normal methods. For that reason, it is the standard startup program on your BASIC 8.0 RUNTIME DISK (It is also on the BASIC 8.0 WORK DISK). All you (the BASIC 8.0 programmer) have to do to share your programs is to DSAVE them to the runtime disk with the name B8.XXXXX and the WORKBENCH will allow users to simply boot, point, click and go!

However, like all the supplied BASIC 8.0 programs, the WORKBENCH is meant to be instructive. It demonstrates many aspects of programming in BASIC 8.0, and is a good place to look for examples of using the language.

Chapter Nine

(64)

BASIC 8.0 Utilities

PICT CONVERTER/FONT CONVERTER/LOGO MAKER

This is a set of utility programs that are used for various special functions.

The PICT CONVERTER allows you to convert pictures made with ULTRA HIRES (an early graphics language for the 80 column C128) to BASIC 8.0 format.

The FONT CONVERTER will convert a standard 40 column font created with any C64/C128 font editor to the BASIC 8.0 structure format. The font must be a simple binary file. Once converted, it can be loaded with the @LSTRUCT command in BASIC 8.0 programs.

The LOGO MAKER utility allows you to make LOGO structures that can be loaded directly with the @LSTRUCT command. Just follow the prompts and it will do all the work of creating your

LOGO.

Chapter Nine

(65)

CHAPTER TEN - THE BASIC 8.0 COMMAND ENCYCLOPEDIA 04/10/87

@ANGLE Angle will perform a rotation of how your object (line, box, circle, dot and arc) is viewed around the X,Y,Z axis. Your data remains unchanged, but the result is one of a 3D transformation around the given axis in the given rotation sequence. This rotation occurs when you render graphics.

SYNTAX:

@ANGLE,X angle, Y angle, Z angle, Rotation Sequence

X angle - -360 to 360 X angle of rotation

Y angle - -360 to 360 Y angle of rotation

Z angle - -360 to 360 Z angle of rotation

Rotation Sequence - 0-5 Order in which rotation about the various axis is performed. Different sequences give different results. If rotation sequence is greater than 5 than rotation sequence defaults to 0.

Sequence #	Rotation Sequence
0	XYZ
1	XZY
2	YXZ
3	YZX
4	ZXY
5	ZYX

Chapter 10 - The BASIC 8.0 Library

(66)

@ARC Used to draw ellipse, arcs, polygons, pie wedges, subtended arcs

SYNTAX:

@ARC, Center X, Center Y, Center Z, X radius, Y radius, Starting Angle,
Ending Angle, Increment, Thickness, Subtend Flag

Center X - 0-65535 X center of arc.
Center Y - 0-65535 Y center of arc
Center Z - -32768 to 32767 Z center of arc

X radius - 0-65535 Radius of arc in horizontal direction.
Y radius - 0-65535 Radius of arc in vrtical direction.

Starting angle -360 to 360 angle at which to start arc
Ending angle -360 to 360 angle at which to end arc

Increment 1-180 smoothness of the arc (angle between lines)
1-8 very smooth
30 12 sides
45 8 sides
60 6 sides
90 4 sides
120 3 sides

Thickness - 0-65535 Number of times the @ARC is drawn stepping
by the value defined by the @GROW command.

Subtend flag - 0-1 0 = do not subtend arc
Non 0 = draw lines from starting and ending angle
To center of arc.

ARC is rotated, scaled and clipped (if necessary) before being drawn.

Chapter 10 - The BASIC 8.0 Library

(67)

@BOX Draw a box or rectangle

SYNTAX:

@BOX,X1,Y1,Z1,X2,Y2,Z2,Shear direction, Shear value, Thickness

X1 - 0-65535	Corner 1 X coordinate of box.
Y1 - 0-65535	Corner 1 Y coordinate of box.
Zl - -32768 to 32767	Corner 1 Z coordinate of box.
X2 - 0-65535	Opposite diagonal corner X coordinate of box.
Y2 - 0-65535	Opposite diagonal corner Y coordinate of box.
Z2 - -32768 to 32767	Opposite diagonal corner Z coordinate of box.

Shear direction 0-2	0=no shear 1=X shear (move corners 3&4 right or left) 2=Y shear (move corners 2&3 up or down) 3 X & Y shear 4 or greater produces a value of 0-3, depending on the valueAND3.
Shear value -32768 to 32767	number of pixels to move the affected corners depending on direction of shear.
Thickness - 0-65535	Number of times the box is drawn stepping by the value defined by the @GROW command.

BOX is rotated, scaled and clipped (if necessary) before being drawn.

Chapter 10 - The BASIC 8.0 Library

(68)

@BRUSHPATRN Convert a brush to a pattern. Allows any piece of a screen to be used as a pattern for the paint command.

SYNTAX:

@BRUSHPATRN,Brush Structure #, Pattern Structure #, Buffer #, Address

Brush Structure # - Brush # to convert to pattern
Pattern Structure # - Destination pattern structure 0-191
Buffer - Buffer to store the structure
Address - Address in Buffer for pattern

@BRUSHPATRN works on any uncompressed brush that is no more than 255 bytes (2040 pixels) wide and 255 scan lines (pixels) deep. If the brush is color, the pattern is color.

A=@BRUSHPATRN,0,1,2,1024
IF A=-1 then wrong structure type or brush is compressed.
ELSE A=next address in the destination pattern buffer.

Chapter 10 - The BASIC 8.0 Library

(69)

@BUFFER Defines an area in each bank to be used as data storage of Structures.

SYNTAX:

@BUFFER, Buffer #, Beginning Address, Size
Buffer # - 0-9 Indicates which of the 10 banks to use
0 System Ram Bank 0

1 System Ram Bank 1

2-9 Expansion Ram Banks 0-7

* Buffer numbers greater than 9 default to 9. *

Beginning Address - 0-65535 Indicates where BUFFER will begin
in designated bank.

Size - 0-65535 Indicates the amount of continuous ram
to be used as a BUFFER.

NOTE: @BUFFER doesn't check if size is greater than the amount
left after the beginning address is defined.

For example:

@BUFFER, 1,32000,65535 Buffer starts at 32000 for 65335 bytes

This will give unpredictable (and probably unpleasant) results

Chapter 10 - The BASIC 8.0 Library

(70)

@CBRUSH Change a non-compressed BRUSH in memory

SYNTAX:

@CBRUSH,STRUCTURE #,REVERSE, REFLECT, FLIP

Structure # 0-191 Number of STRUCTURE with BRUSH data

REVERSE 0-1 0 No Reverse
1 Reverse

REFLECT 0-1 0 No Reflect

```
            1 Reflect

FLIP 0-1    0 No Flip
            1 Flip

Once a BRUSH is modified with @CBRUSH, you can @FETCH it to the screen
     in its new form. You can combine the commands to do several
     actions at once.

If structure is wrong type or compressed, -1 is returned.

ex., e=@chrush,structure#,reverae,reflect,flip

Chapter 10 - The BASIC 8.0 Library

(71)
@CHAR Outputs a text string to the 8563 bitmap screen.

SYNTAX:
@CHAR, Structure #,Column,Row,Height,width,Direction,"Character string"

structure #              0-191 indicate a ram resident font
                               in a user defined structure.
                         254   indicates uppercase/lowercase
                               ROM font.
                         255   indicates uppercase/graphics
                               ROM font.
Column                   0-255 Horizontal column number
Row                      0-819 Vertical scan line (pixel)
Height                   1-16  Height of character
Width                    0-16  Width of character
                               0 indicates 160 col char set
```

Values greater than 16 default to 16.

Direction 0-7 Indicates direction to print

0 Print up

1 Print up & right

2 Print right

3 Print down & right

4 Print down

5 Print left & down

6 Print left

7 Print left & up

Greater then 7 becomes 0-7

String Text string to print. String can contain special control codes embedded within the character string to be printed. These control codes are created by pressing the CTRL key and the indicated letter.

CONTROL CODES

CTRL-B	Blank cell under character
CTRL-I	Inverse cell (AND) with character
CTRL-O	Overwrite cell (OR) with character
CTRL-X	Complement cel1 (XOR) with character
CTRL-F	Flip character upside down (Toggle)
CTRL-P	Pattern character (Toggle)
CTRL-U	Underline character (Toggle)
CTRL-Y+	Rotates character left 90 degrees
CTRL-Y-	Rotates character right 90 degrees
CTRL-Y	Resets character upright
CTRL- Z	Mirror Image character (toggle)

CTRL-RVS ON Reverse on

CTRL-RVS OFF Reverse off

COLOR CTRL CODES

CTRL-C By itself turns Jam 0 color control on

CTRL-CForeground Color Sets character color (Jam l)

CTRL-CForeGroundBackground Color This is Jam 2 mode.

Foreground or Background color is set by pressing

Chapter 10 - The BASIC 8.0 Library

(72)

the CTRL Key or CBM Key and the appropriate color key.

CTRL-Cursor	Up,down,left,right
	These work on a 8 X 8 cell only
SHIFTED CLR	Clears screen with last FC/BG
	colors.
HOME	Places cursor at 0,0 position
	in window.

FULL CHARACTER WRAP IS SUPPORTED IN OUTPUT WINDOW IN ALL DIRECTIONS.

Chapter 10 - The BASIC 8.0 Library

(73)

@CIRCLE Draws a fast circle.

SYNTAX:

@CIRCLE,Center X,Center Y,Center Z,Radius,Thickness

Center X - 0-65535	X center of circle
Center Y - 0-65535	Y center of circle
Center Z - -32768 to 32767	Z center of circle
Radius 0-65535	radius of circle

Thickness - 0-65535 Number of times the circle is drawn stepping by the value defined by the @GROW command.

CIRCLE is rotated, scaled and clipped (if necessary) before being drawn.

The @CIRCLE command produces a symmetrical circle, but in SCALE 0 it will appear to be an ellipse (due to the unsymmetrical y direction). Setting SCALE to 1 or 2 will allow you to make a proper circle. Or use @ARC with the x radius and y radius adjusted accordingly.

Chapter 10 - The BASIC 8.0 Library

(74)
@CLEAR Clear the bitmap screen

SYNTAX:
@CLEAR,Bitmap Fill Value[,Background Color,Foreground Color]

Bitmap Fill Value - 0-255 Fills bitmap area with given value
0 clear window
1 Only the windows color information
is affected, bitmap image is untouched.
255 solid window
2-254 various vertical striped windows result

Color parameters are optional. If omitted current colors are used.

Background Color - 0-15 Sets Background color
Foreground Color - 0-15 Sets Foreground color

If a window was opened with the border flag set, every time the window is cleared the border is drawn in the current drawmode on the

inside boundaries of the window. This will continue to occur until the window is closed.

Chapter 10 - The BASIC 8.0 Library

(75)

@COLOR Defines colors for foreground, background and border

SYNTAX:

@COLOR, background color, foreground color, outline (border) color

Background color : 0-15 See color table

Foreground color : 0-15 See color table

Outline Color : 0-15 See color table

NOTES: Outline color achieved only in multicolor modes.

COLOR TABLE

0 Black

1 Dark Grey

2 Blue

3 Lite Blue

4 Green

5 Lite Green

6 Cyan

7 Lite Cyan

8 Red

9 Lite Red

10 Purple

11 Lite Purple

12 Brown

13 Yellow

14 Lite Grey

15 White

If using monochrome mode the effect of the @COLOR command will be to change the display colors immediately. In color modes the border will change at once, but only subsequent graphic commands will use the new colors.

Chapter 10 - The BASIC 8.0 Library

(76)

@COPY @COPY moves a rectangular area of screen to any other defined screen area. It uses the 8563 block write capabilities, and is very fast. It works on the column (8 pixel) level horizontally and scan lines (pixels) vertically when in monochrome. When in color the y values are adjusted to the color cell size.

SYNTAX:

@COPY, Source Screen, Start X, Start Y, DX, DY, Destination Screen, EX, EY

Source Screen : 0-7 as defined by @MODE or @SCRDEF

Start X : 0-2047 Defined as pixels, but rounded down to nearest column.

Start Y : 0-819 Defined as scan lines (pixels) but rounded down to color cell.

DX : Distance in pixels, rounded up to columns

DY : Distance in scan lines (pixels) rounded up

to next color cell.

Destination Screen : 0-7 as defined by @MODE or @SCRDEF

EX : Ending X at destination screen, rounded down
to nearest column

EY : Ending Y at destination screen, in scan lines
(pixels)

**** NOTE **** @COPY WILL NOT WORK RELIABLY WHEN VIEWING A
VIRTUAL HORIZONTAL SCREEN GREATER THAN 640 PIXELS.
YOU MAY VSE @COPY ON A VERTICAL VIRTUAL SCREEN,
BUT @COPY WILL ONLY WORK ON VIRTUAL HORIZONTAL
SCREENS IF YOU ARE VIEWING A NON-VIRTUAL
HORIZONTAL SCREEN. FOR EXAMPLE, IF YOU HAVE
TWO 8CREENS, ONE THAT IS A HORIZONTAL VIRTUAL
OF 1280 X 200 AND ANOTHER THAT IS 640 X 400
VERTICAL VIRTUAL. YOU CAN @COPY ON THE 1280 X 200
IF YOU ARE VIEWING THE 64O X 400. ATTEMPTS TO
@COPY OM THE 1280 X 200 WHILE VIEWING IT WILL
CAUSE THE COPIED BITMAP TO BE DISTORTED. THIS
IS DUE TO A BUG IN THE 8563 ADDRESS INCREMENT
PER ROW REGISTER.

IF START X,Y PLUS DX,DY IS OUT OF BOUNDS THEN
SOURCE RECTANGLE COPIED IS CLIPPED. IP END X,Y
PLUS DX,DY IS OUT OF BOUNDS THEN DESTINATION
RECTANGLE IS CLIPPED AND X,Y,DX,DY,EX,EY ARE
SCALED IF NECESSARY.

(77)

@CYLNDR Draw solid cylinder

SYNTAX:

@CYLNDER,X,Y,RADIUS,HALFLEN,VIEW

X	0-Xmax
Y	0-YMax
radius	1-127
Halflen	1-128
View	0/1 type of cylinder
	0 horizontal
	1 vertical

Chapter 10 - The BASIC 8.0 Library

(78)

@DIR$ Read a directory entry

Requires considerable setup from basic

SYNTAX:

L=@DIR$

@DIR$ returns a string to the first variable in your program. The variable must be a string type. @DIR$ returns a value equal to the length of the string found. You must use the LEFT$(VAR$,L) to remove any trailing null values. Here is an example routine that will search the directory for all BRUS. files.

VARIABLE DESCRIPTIONS

```
dn=device#
sd=drive side (0 or 1)
dp$= directory pattern to seek
de=directory index variahle = ends up as last index in de$()
de$=directory entry variable...This must be first variable
    in your program.

 10 de$="                ":rem de$=16 spaces
 20 dim de$(244):rem maximum number of possible files
 30 dp$="brush.":rem look for brush files
 40 dn=8:sd=0:rem drive 8, side 0
 50 de=0:rem number of directory entries
 60 open 3,dn,0,"$"+mid$(str$(sd),2)+":"+dp$+"*" :rem open file
 70 1=@DIR$:rem 1 is the length of the string now in de$
 80 if st<>0 then 110
 90 de$(de)=left$(de$,1)
100 de=de+1:goto 70
110 close 3
120 rem print out all found entries
130 for i=0 to de:print de$(i):next i
```

The pattern used in the open command is a wildcard used to search for specific files. To get all the files on a directory, just use the $ as the search string. The variable sd is used in case of a dual drive. And the array used to store the filenames, DE$(244) is dimensioned to the number 244 in order to work with the 1581 disk drive, which can store a larger number of file entries than the 1541 or 1571 disk drives (144 each).

@DISPLAY Recalls a @STOREd screen or brush to the designated screen at the location from which it was stored or a specified location.

SYNTAX:

@DISPLAY, SCREEN #, DEVICE #, DRAWMODE, "FILENAME" [,X,Y]

SCREEN# - The number of the screen the file is to be loaded to. It does not have to be the screen you are currently viewing.

DEVICE # - The drive number you are using, ex. 8

DRAWMODE - 0-3

0 Erase under (replace)

1 Merge with (OR)

2 Common (AND)

3 Complement (XOR)

FILENAME - The name of the screen or brush to load. It will be loaded to the area of the screen it was saved from (i.e., the same X,Y coordinates).

X,Y - Optional screen locations to display brush to.

NOTE: If you use the optional X,Y parameters for displaying a brush to a position other than the default values, there is one thing to watch for. If Y is greater than the original Y it was saved from, (0 in the case of a picture, the Y hotspot in the case of a brush), then THERE IS NO CLIPPING PERFORMED IN THE Y DIRECTION!. So be sure your brush or picture will fit in the area and not overrun the bottom of the current screen display. If it does, unusual (and probably visually unpleasant) screen effects occur.

This will happen only with @DISPLAY.

Chapter 10 - The BASIC 8.0 Library

(80)

@DOT Plots a single pixel in 3 space

SYNTAX:

@DOT,X,Y,Z

X - 0-65535 X coordinate of point. Highest visible X is equal to Xmax of virtual screen, or Window width.

Y - 0-65535 Y coordinate of point. Highest visible Y is equal to Ymax of virtual screen, or Window depth.

Z - -32768 to 32767 Z coordinate of point. Z value of 0 is the plane of the screen. Positive values are directed into the screen, while negative vales are directed towards the viewer.

DOT is rotated, scaled and clipped (if necessary) before being drawn.

Chapter 10 - The BASIC 8.0 Library

(81)

@DRWMOD Two Commands to define global drawing modes to be used with graphic commands.

SYNTAX: @DRWMODA,Jam,InverseVideo,Complement,Undraw,Pattern,Merge,Clip

@DRWMODB,Zview,Unplotlast,Unplotvertex

```
Jam : 0-2 Updates color memory when drawing on a color screen.
0    Uses foreground/background colors found in cell.(Fastest)
1    Uses background color in cell and foreground color set
     by @COLOR command.
2   Uses foreground/background colors set by @COLOR command.

InverseVideo : 0-1 Reverses colors set by Jam option.
0 off
1   on

Complement   : 0-1 Turns on XOR mode. Pixels on are turned off,
                   pixels off are turned on.
               0   off
               1   on

Undraw       : 0-1 Turns on erase mode. Actually unplots all
                   pixels.
               0   off
               1   on

Pattern      : 0-1 Uses previously defined pattern when drawing.
               0   off
               1   on

Merge        : 0-1 Merge pattern with screen using OR mode,
                   pixels on are unaffected.
               0   off
               1   on

Clip         : 0-1 Turns on and off global screen/window clipping.
               0   on
               1   off
```

NOTE The default is 0, CLIPPING ON. Turn it off for
an increase in drawing speed.

***** DRWMODB values may be used in any combination *****

Zviev : 0-1 Turns on either parallel or perspective 3D
0 Perspective on (objects drawn toward vanishing
point appear smaller as they approach it).
1 Parallel on. Retains object shape and size but
it is offset based upon Z coordinate.

Unplotlast : 0-1 Unplots last iteration of a multi-height
drawing of a graphic primitive (i.e., lines,
circles, boxes etc). Used to make 3D bars
from boxes.
0 off
1 on

Unplotvertex : 0-1 Unplots vertexs of graphic primitives (i.e.,
circles, arcs, boxes etc) Also used to
give 3D appearances of bars, tubes etc.

Chapter 10 - The BASIC 8.0 Library

(82)
@FETCH To recall a @STASHed area to the display screen.

SYNTAX:
@FETCH,STRUCTURE #,X,Y,DRAW MODE

STRUCTURE # - 0-191 The structure # used when @STASHing

```
X            -  The screen X location to recall the
                BRUSH to. X is rounded down to the
                start of the current byte.
Y            -  The screen Y location to recall the
                BRUSH to. Y is rounded up to the top
                of the current color cell.
Draw Mode    -  0-3
                0  Erase under (replace)
                1  Merge with (OR)
                2  Common (AND)
                3 Complement (XOR)
```

Chapter 10 - The BASIC 8.0 Library

(83)

@FLASH Reverse a rectangular area of the screen 1-255 times

SYNTAX:

@FLASH,X,Y,DX,DY,Number Of times to flash[,FAST]

```
X  -  Screen X 0-639
Y  -  Screen Y 0-Ymax
DX -  Number of pixels wide (X+DX)<640
DY -  Number of pixels down (If using color, rounded up
      to color cell)
Number - Number of times to reverse the rectangle
         1-255
FAST  0-1  OPTIONAL PARANETER (DEFAULT IS 1)
      0    No fast FLASH, use bitmap regardless of graphic
           mode.
      1    Use fast FLASH, means reverse color cells only
           when in color graphic mode.
```

***** WARNING ***** @FLASH, like @COPY will not work on a
horizontal virtual screen. The screen
must be 640 pixels wide, no wider. Y depth is
irrelevent.

Chapter 10 - The BASIC 8.0 Library

(84)

@FONT Loads a custom font to 8563 chip ram for use on the text screen.
Two complete fonts can be resident at one time (512 total chars).

SYNTAX:

@FONT,Charset #,Structure #

Charset # - 0-1 Indicates which character set to replace
(Character set consist of 256 characters)
Value greater than 1 defaults to 1

Structure # - 0-191 Indicates the data structure that contains
the ram resident character font data.

254 Uppercase/Lowercase ROM font in bank 14

255 Uppercase/Graphics ROM font in bank 14

Chapter 10 - The BASIC 8.0 Library

(85)

@GROW Step value for the X, Y and Z axis when using the multidrawing
(thickness) parameter of the graphic drawing commands. Minus value
indicates negative direction.

SYNTAX:

@GROW,X step, Y step, Z step

X step - -32768 to 32767 Step value in X direction (i.e., left and right). A value of 1 indicates a solid line, n>1 steps lines by n.

Y step - same as X step except up and down

Z step - same as X step except towards or away from viewer.

Chapter 10 - The BASIC 8.0 Library

(86)

@HCOPY Print a hardcopy of the current screen on a dot matrix printer.

SYNTAX:

@HCOPY, Sec Address,height,density,rotation

Sec Address	See chapter on printer drivers for specifics
Height 1-4	Height of printed output
Density 1-7	Allows for variable densities, if printer supports different densities. See chapter on printer drivers for specific capabilities of supported printers.
Rotation 0-1	0 means no rotation 1 means rotate 90 degrees. Some printers (like MPS 801) are ALWAYS rotated.

Chapter 10 - The BASIC 8.0 Library

(87)

@LINE Draw a line in 3 space

SYNTAX:

@LINE,X1,Y1,Z1,X2,Y2,Z2,Thickness

Xl - 0-65535 Startinq X coordinate of line.

Yl - 0-65535 Starting Y coordinate of line.

Zl - -32768 - 32767 Starting Z coordinate of line.

X2 - 0-65535 Ending X coordinate of line.

Y2 - 0-65535 Ending Y coordinate of line.

Z2 - -32768 - 32767 Ending Z coordinate of line.

Thickness - 0-65535 Number of times the line is drawn stepping
by the value defined by the @GROW command.

LINE is rotated, scaled and clipped (if necessary) before being drawn.

Chapter 10 - The BASIC 8.0 Library

(88)

@LOGO Draws a logo using the @CHAR command. LOGO screen position is
defined within the LOGO itself.

SYNTAX:

@LOGO,Structure number

Structure number 0-191 selects which structure to use as @LOGO data.
If Structure is of wrong type it returns a value of -1

example A=@LOGO,STRUCTURE Number
IF A=-1 THEN ERROR IN STRUCTURE TYPE

@LOGO PARANETER CHANGES FOR USE WITH @STRUCT/@SDAT/@SEND

You DO NOT follow the parameters with a string of characters. Instead follow the parameters with a series of numbers that are the ASCII equivilent of the characters. For example, if your letters are "abc" you would send the following parameters after the Length parameter;,65,66,67.

This will create the logo string "abc". Also, while there are no required parameters for LOGO types, you must still follow the @STRUCT command with a @SDAT on a line by itself. For example;

```
@STRUCT,5,2,1,AD
@SDAT
@SDAT,l,254,0,0,0,1,1,2,3,65,66,67
@SDAT,0
@SEND
```

This creates the LOGO string "abc".

NOTE Because LOGO uses the CHAR command, (which get string data from the variable bank 1) all LOGO structures MUST be in BUFFER 1 (System Bank 1). Use the @BUFFER command to define a buffer in bank l.

Chapter 10 - The BASIC 8.0 Library

(89)

@LSTRUCT Loads a structure into memory

SYNTAX:

@LSTRUCT,STRUCT #, DEVICE #,BUFFER #,BUFFER ADDRESS,FILENAME

STRUCT # - 0-191

DEVICE # - 8-11

BUFFER # - 0-9

BUFFER ADDRESS - 0-65500

FILENAME - name of structure on disk you want to load

*** @LSTRUCT MUST BE FOLLOWED BY @SEND ***

Chapter 10 - The BASIC 8.0 Library

(90)

@MODE Specify's which of the four group of predefined screens
       (graphic modes) to be used as display areas. For unexpanded
       8563 chip ram (16K) only MODE 0 (default) is available.

SYNTAX:

@MODE,mode #[,Interlace Flag]

Mode #   -   0-3 Which of the 4 screen sets to use

Interlace Flag - 0/1 Interlace option [Optional parameter]
                  0 No Interlace Sync
                  1 Use Interlace Sync (Not all monitors
                     can display interlace sync.)

Mode # : 0-3
  0   For use with 16K or 64K chip ram Mixed Types

*** NOTE *** Only 1 screen is available at a time in Mode 0

     @SCREEN,0 - 640 X 200 Monochrome screen
     @SCREEN,1 - 640 X 192 Color 8 X 16 cell
     @SCREEN,2 - 640 X 176 Color 8 X 8 cell

```
     @SCREEN,3 - 640 X 152 Color 8 X 4 cell
     @SCREEN,4 - 640 X 104 Color 8 X 2 cell
     @SCREEN,5 - 640 X 176 Color 8 X 8 cell interlaced
     @SCREEN,6 - 640 X 152 Color 8 X 4 cell interlaced
     @SCREEN,7 - 640 X 104 Color 8 X 2 cell interlaced

  1 For use with 64K chip ram only - Mixed Types

*** NOTE ***  SCREENS 0, 1, and 2 can be used at the same time, or
              SCREENS 0, 1, and 3 can he used at the same time, or
              SCREENS 4, 5, and 6 can be used at the same time, or
              SCREEN 7.

     @SCREEN,O - 640 X 200 Monochrome
     @SCREEN,1 - 640 X 200 Color 8 X 8 cell
     @SCREEN,2 - 640 X 200 Color 8 X 2 cell
     @SCREEN,3 - 640 X 300 Monochrome Virtual
     @SCREEN,4 - 640 X 200 Color 8 X 8 cell
     @SCREEN,5 - 640 X 200 Color 8 X 8 cell
     @SCREEN,6 - 640 X 200 Color 8 X 8 cell
     @SCREEN,7 - 640 X 728 Color 8 X 8 cell Virtual

  2 For use with 64K chip ram only

*** NOTE ***  SCREENS 0, 1, 2 and 3 can he used at the same time, or
              SCREENS 3, 4 and 5, or
              SCREENS 4, 6 and 7, or

     @SCREEN,O - 640 X 200 Monochrome
     @SCREEN,1 - 640 X 200 Monochrome
```

(91)

@SCREEN,2 - 640 X 200 Monochrome

@SCREEN,3 - 640 X 200 Monochrome

@SCREEN,4 - 640 X 200 Color 8 X 2 cell

@SCREEN,5 - 640 X 200 Color 8 X 2 cell

@SCREEN,6 - 640 X 200 Color 8 X 4 cell

@SCREEN,7 - 640 X 200 Color 8 X 4 cell

3 For use with 64K chip ram

*** NOTE *** ONLY ONE SCREEN CAN BE USED AT ONCE!

@SCREEN,O - 1280 X 409 Monochrome Virtual

@SCREEN,1 - 640 X 819 Monochrome Virtual

@SCREEN,2 - 2040 X 252 Monochrome Virtual

@SCREEN,3 - 800 X 655 Monochrome Virtual

@SCREEN,4 - 640 X 546 Color 8 X 2 cell Virtual

@SCREEN,5 - 640 X 655 Color 8 X 4 cell Virtual

@SCREEN,6 - 640 X 728 Color 8 X 8 cell Virtual

@SCREEN,7 - 640 X 768 Color 8 X 16 cell Virtual

*** NOTE *** Interlaced mode screens merely change the display
to Interlaced Sync Mode. These SCREENS are
duplicates of a non-interlaced display. For more
information on Interlace sync mode see the appendix.

Chapter 10 - The BASIC 8.0 Library

(92)

@MOUSE commands to turn on and off the interrupt Mouse and Joystick, and
return the current X and Y position.

```
SYNTAX:
@MOUSE,1,DEVICE,X,Y[,Joystick Increment]

1        -  Activate IRQ reader
DEVICE   -  0/1
            0 mouse (Port 1)
            1 joystick (Port 2)
X        -  0-2047 horizontally, depending on Screen X Max

Y        -  0-819 Vertically, depending on Screen Y Max

Joystick Increment  -  Required ONLY FOR JOYSTICK - NOT MOUSE!
                       1-255 Controls number of pixels the
                       arrow moves when using the joystick.
                       If increment=0 then no movement is possible.

@MOUSE,0   Turn OFF interrupt reader

@MOUSE,2,0 Returns X coordinate of mouse (or joystick)

   X=@MOUSE,2,0

@MOUSE,2,1 Returns Y coordinate of mouse (or joystick)

   Y=@MOUSE,2

READING THE MOUSE BUTTONS (PORT 1)

To read the LEFT MOUSE button on the 1351 mouse:
   10 IF JOY(1)>127 THEM PRINT "LEFT MOUSE BUTTON IS PUSHED"
```

To read the RIGHT MOUSE button on the 1351 mouse:

```
10 IF JOY(1)=1 THEN PRINT "RIGHT MOUSE BUTTON IS PUSHED"
```

READING THE JOYSTICK BUTTON (PORT 2)

To read the JOYSTICK FIREBUTTON:

```
10 IF JOY(2)>127 THEN PRINT "JOYSTICK FIRE BUTTON PRESSED!"
```

Chapter 10 - The BASIC 8.0 Library

(93)

@ORIGIN Establishes center of rotation and vanishing point (used for perspective draw mode). Vanishing point determines where an object will appear as a single point if drawn in perspective drawmode. Allows varied depths in Z axis.

SYNTAX:

@ORIGIN,Center X, Center Y, Center Z, Vanish X, Vanish Y, Vanish Z

Center X - 0-65535 (Value can be larger than Virtual X)

Center Y - 0-65535 (Value can be larger than Virtual Y)

Center Z - -32768 to 32767

Vanish X - 0-65535

Vanish Y - 0-65535

Vanish Z - -32768 to 32767

NOTE: All X,Y coordinates are scaled to current scale factor.

Chapter 10 - The BASIC 8.0 Library

(94)

@PAINT Paint an area with pattern or solid.

SYNTAX:

@PAINT,X,Y,Bank#,Address,Size

X	= 0-Xmax	x coordinate to start paint from.
Y	= 0-Ymax	y coordinate to start paint from.
Bank#	= 0-1 C	128 system ram bank for PAINT stack area. The more complex the area to be filled, the larger the stack area you need. If the stack fills up the paint action stops, but the program continues from the next statement. Make sure you don't use memory where your program resides.
Address	= 0-65535	Startinq address in designated bank for stack area
Size	= 0-65535	Size of stack area in bytes

To determine if an error condition occurs, use;

ER=@PAINT,X,Y,BANK#,BANK ADDRESS,STACK SIZE

IF ER=O then no error occurs

IF ER=-1 then an error occured, PAINT aborted.

NOTES:

1. @PAINT works on SCALED coordinates.
2. For many programs we suggest you use 1K of ram in Bank 0 starting at 57343.

ex; @PAINT,X,Y,0,57343, 1024 :REM Use 1K stack at 57343

Chapter 10 - The BASIC 8.0 Library

(95)

@PATTERN

SYNTAX:

@PATTERN, Structure number

Structure number - 0-191 Indicates which of the possible 192
STRUCTURES to use as pattern data. If
indicated STRUCTURE is not of a pattern
type then error status is -l.

example A=@PATTERN,STRUCTURE number
If A=(-1) then Structure was wrong type.
If A <>(-1) then Pattern was successfully defined.

Chapter 10 - The BASIC 8.0 Library

(96)

@PIXEL Function to tell if a pixel is on/off or the color of that cell.

SYNTAX:

@PIXEL,X,Y,Mode

X - 0-65535 x coordinate of point to get value of.
Y - 0-65535 y coordinate of point to get value of.
Mode - 0-1 0 = Bitmap pixel status returns
O=off l=on -l=out of bounds

1 = Color attribute status returns value
with foreground/background color combined

foreground color = value and 15
background color = (value and 240)/16

PIXEL is rotated, scaled and clipped (if necessary) before being drawn.

```
  A=@PIXEL,320,100,0
  IF A=O THEN PIXEL NOT ON ELSE PIXEL IS ON

  A=@PIXEL,320,100,1
  FC=A AND 15:REM Foreground color = FC
  BC=INT((A AND 240)/16)
```

Chapter 10 - The BASIC 8.0 Library

(97)

@PTR Plots a sprite-like pointer at position X and Y

SYNTAX:

@PTR,O Turns off the pointer

@PTR,1,X,Y,DEF#[,hgt] Turns an the pointer and positions it at X,Y
The mode of 1 indicates the pointer 'floats' over
the bitmap images.

X - X coordinate of arrow

Y - Y coordinate of arrow

DEF# - 0-15 Sixteen pointer definition shapes

HGT - 1-16 Height of pointer, system defaults to eight but
you can change it to be larger. By settinq it to
16 you can use two definitions as one. This allows
pointer definitions of 16 x 16 pixels. In this case
the pointers are paired, with n and n+1 acting as a
single pointer.

POINTER will automatically restore what was underneath it
when moved or turned off when using mode l.

@PTR,2,X,Y,DEF# Turns on the arrow pointer and positions it at X,Y
                 The mode of 2 indicates the pointer leaves a trail
                 on the bitmap of the pointer image.

X   - See above
Y   - See above
DEF# - See above
HGT - See above

POINTER will not restore what was underneath it when moved using mode 2.

See Appendix C for information on defining your own pointers.

NOTE: POINTER RESPONDS TO DRAW MODES UNDRAW AND COMPLEMENT.

Chapter 10 - The BASIC 8.0 Library

(98)

RYLANDER SOLIDS

These are a special subset of BASIC 8.0 commands. They were developed by Richard Rylander for the Commodore 64, and he allowed us to adapt them to the BASIC 8.0 graphics system. However, they are not an integral part of the language, and do not respond to all the possible 3D parameters. We consider them important enough to have included them, and have made a number of changes to allow them to work on the C128 BASIC 8.0 hires display screen. You will find they offer a very powerful graphics capability, and we wish to thank Mr. Rylander for his gracious permission for their use.

For more information on the Rylander Solids, see the commands @SHERE, @TOROID, @SPOOL, @CYLNDR, @SCLIP and @STYLE.

Chapter 10 - The BASIC 8.0 Library

(99)

@SCALE Changes bitmap plotting area to logical equidistant units. This results in X and Y coordinate ranges being equally proportioned.

SYNTAX:

@SCALE,scale number

Scale number - 0-2 Sets scale units.

0 No scale, screen remains pixel oriented.

1 Sets Y pixel values to .39 of a real pixel. For example, a 640 X 200 physical screen becomes a 640 X 512 logical screen.

2 Doubles logical screen size. Same logical screen (640 X 512) at SCALE 1 becomes 1280 X 1024 logical points at 6CALE 2.

* NOTE! Logical screens are not the same as Virtual screens. Larger virtual screens result in larger logical screens. Scale values greater than 2 default to 0.

Chapter 10 - The BASIC 8.0 Library

(100)

@SCLIP Solids Clip, sets the clipping boundaries for the Rylander Solids

SYNTAX:

@SCLIP,left,right,down,up

left 0-255 based upon center of solid object

right 0-255 based upon center of solid object

up 0-255 based upon center of solid object

down 0-255 based upon center of solid object

radius is from 1-127

Chapter 10 - The BASIC 8.0 Library

(101)

@SCRDEF Defines an area of memory in the 8563 chip ram to be used as a custom screen for display. This overrides the @MODE command for the specified screen #.

SYNTAX:

@SCRDEF,screen#,display type,color size,size x,size y,
bitmap beg. addr,color beg. addr

screen# : 0-7 indicates which screen definition to be used by the @SCREEN command. Values greater than 7 are converted to 0-7 (eg., 8=0, 9=1, 10=2 etc.). This is always the case.

display type : 0 = normal RGB
non 0 = Interlace Sync - area between scan lines filled to yield a sharper image.
(some monitors will not display this mode properly)

color size : 0-4 Indicates the color resolution to be used.
0 Use monochrome mode
1 Use 8 X 2 color cell : requires 8000 bytes
2 Use 8 X 4 color cell : requires 4000 bytes
3 Use 8 X 8 color cell : requires 2000 bytes
4 Use 8 X 16 color cell : requires 1000 bytes

Number of bytes is in addition to 16000 required for screen bitmap of 640 X 200 pixels. Larger virtual screens require more ram. Color sizes greater than 4 default to type 4. To determine the amount of ram needed for a bitmap, use the formula

Bitmap Ram = ((maximum x)/8)*(maximum y)

Color Ram = 80*(maximum y)/sizecode

Screen Type	Sizecode
mono	no color ram
color 8 X 16	16
color 8 X 8	8
color 8 X 4	4
color 8 X 2	2

Total ram for a color screen=Bitmap ram + color ram

SizeX : 640 to 2040 in increments of 8 pixels
Indicates horizontal size of virtual screen area. Only 640 pixels can be seen on one line at time. Values greater than 640 only possible if 64K 8563 chip ram installed. Virtual screens greater than 640 pixels work in monochrome mode only.

SizeY : 200 X 819 in increments of scan lines (1 pixel)
Indicates vertical size of virtual screen area. Only 200 lines can be seen at one time. Values greater than 200 only possible if 64K 8563 chip ram is installed. Virtual screens greater than 200 are allowed in both color and monochrome modes.

Bitmap Address : 0 if only 16K 8563 chip ram installed.
0-65535 if 64K 8563 chip ram installed.

49535 last available address for a contiguous
16K bitmap. (65535 - 16000 = 49535)

Color Address : 0-16383 if only 16K 8563 chip ram installed.
0 - 63535 if 64K 8563 chip ram installed.

Chapter 10 - The BASIC 8.0 Library

(102)
@SCREEN select which screens to use when drawing and/or viewing.

SYNTAX:
@SCREEN,draw screen #[,view screen #]

draw screen #: 0-7 Indicates which is active drawing screen.
view screen #: 0-7 Indicates which screen is displayed.

If both parameters are the same or view screen 4 is omitted then the view screen becomes the active drawing screen. If they differ this allows you to draw on one screen while viewing another. This can be used for DOUBLE BUFFERING, which is a common animation technique.

Chapter 10 - The BASIC 8.0 Library

(103)
@SCROLL Scroll bitmap area
SYNTAX 1:
@SCROLL,Direction,Number of Units,Speed

Direction - 0-7 0 = up
1 = up & right

```
                2 = right
                3 = down & right
                4 = down
                5 = down & left
                6 = left
                7 = up & left
                Greater then 7 becomes 0-7

Number of Units - 0-255 number of columns and/or color cells to
                        move in given direction.

Speed             0-255 time delay between each scroll increment
                  0 no delay
                  255 = .255 seconds
          (delay = Speed * .001 seconds)

SYNTAX 2: Move the display immediately
@SCROLL,255,LByte of Color Cell,HByte of Color Cell

255 - Indicates you want Absolute Positioning
LByte - Lo byte of color cell (Color Row) to move to
HByte - Hi Byte of color cell (Color Row) to move to

This @SCROLL allows you to move directly to the start of any color cell
     or scanline (in monochrome).

For Example:
  (In monochrome)
     @SCROLL,255,25,0    Move to scanline 25

  (In 8x8 color)
     @SCROLL,255,2,0     Move to scanline 16 (color row 2)
```

Chapter 10 - The BASIC 8.0 Library

(104)

@SDAT Structure DATA statements to read data into current structure buffer

SYNTAX:

@SDAT, required parameters for the Structure Type

@SDAT, optional data depending on Structure Type

*** Required parameters MUST be on a separate @SDAT command line than the optional data. If buffer fills before end of Structure data then status of -1 is returned to status variable. If variable is not -1 then variable contains the next available address in buffer. (16 bit address, 0-65535)

example A=@SDAT,data...
IP A=-1 then BUFFER ERROR, ELSE A=next available address

------------------ Required Parameters -------------------------

Type 1 @SDAT,pattern width, pattern height, color width, color height

Pattern width is in BYTES (8 pixels)
Pattern height is in PIXELS
Color width is in BYTES (8 pixels)
Color height is based on size of current color cell height. For example, if color cell height is 4 pixels, then each color height of 1 is equal to 4 pixels.

Type 2 NO Required Parameters

```
Type 3 NO Required Parameters

Type 4 No Required Parameters


*------------------ Optional Parameters -----------------------*

Type 1 @SDAT,Pattern bitmap data  pattern width X pattern height
                                  optional parameters needed.


  For example, to define a brick wall pattern
  which is 2 bytes wide and 8 pixels deep you
  would use


@BUFFER,2,0,65535 : rem Buffer 2 is 64K buffer in ext. ram bank 0

@STRUCT,0,1,2,0 : rem Struct 0, the pattern, buffer 2, address 0

@SDAT,2,8,1,1 : rem 2 bytes X 8 pixels, 1 X 1 color
@SDAT,255,255,128,0,128,0,128,0,255,255,0,128,0,128,0, 128

@SDAT,206 : rem brown on lite grey using JAM 2 color

@SEND


Type 2 @SDAT,Flag,Structure #, column, y(Lobyte), y (Hibyte),char
          height, char width, direction, length, "string of
          characters"

Chapter 10 - The BASIC 8.0 Library


(105)
   If Flag = 0 then structure is done, else if
     not 0 then continue.
   Other parameters described in @CHAR command.


Type 3 No optional parameters as character set should be loaded
       from disk.


Type 4 No optional parameters as brushes should be loaded from
```

disk.

Chapter 10 - The BASIC 8.0 Library

(106)

@SEND Terminates the current structure (Structure END).

SYNTAX:

@SEND

Special note: The @STRUCT @SDAT @SEND commands are to be used together. Once a @STRUCT command has been given @SDAT commands and then a @SEND command is expected. BASIC 7.0 commands only can be used in between these commands such as:

FOR..NEXT loop to pass data to the @SDAT command.

Use variable to find next byte in buffer with I/O

@LSTRUCT,0,8,2,0,"FNT.ROMAN":AD=@SEND

AD=next free byte in BUFFER.

USING BASIC 8.0 COMMANDS WITHIN THE @STRUCT @SDAT and @SEND COMMANDS WILL GIVE UNPREDICTABLE RESULTS.

Chapter 10 - The BASIC 8.0 Library

(107)

@SPHERE Plot a 3D solid sphere

SYNTAX:

@SPHERE,X,Y,Radius

X - 0-2047 Depending on Screen Max X

Y - 0-819 Depending on Screen Max Y

Radius - 1-127

Chapter 10 - The BASIC 8.0 Library

(108)

@SPOOL Draws a solid spool structure

SYNTAX:

@SPOOL,X,Y,INNER RADIUS,OUTER RADIUS,VIEW

X - 0-Xmax Center X of spool

Y - 0-Ymax Center Y of spool

Inner Radius - 1-127 inner radius of spool

Outer Radius - 1-127 outer radius of spool

View - 0-1 Type of spool

0 Horizontal

1 Vertical

Chapter 10 - The BASIC 8.0 Library

(109)

@SSTRUCT Saves a structure from memory to disk

SYNTAX:

@SSTRUCT,STRUCT #,DEVICE #,FILEMAME

STRUCT # - 0-191

DEVICE # - 8-11

FILENAME - filename to store to disk

*** @SSTRUCT MUST BE FOLLOWED BY @SEND ***

Chapter 10 - The BASIC 8.0 Library

(110)

@STASH Used to place a designated rectangle of screen display in a brush structure for later recall or storage to disk.

SYNTAX:

@STASH,STRUCTURE#,BUFFER#,BUFFER ADDRESS,X,Y,DX,DY,COMPRESSION

STRUCTURE # - 0-191 The designated structure number used to recall the data with the @FETCH conunand.

BUFFER # - 0-9 The ram bank to be used as a buffer. This parameter is just like the @BUFFER command.

Buffer Address - 0-65535 The address within the buffer to begin storing the data.

X - 0-Virtual X The beginning X coordinate of the rectangular area to store. X will be rounded down to an even byte boundary. For example, if X=26 then the actual X will be 24 (INT(26/8)=24)

Y - 0-Virtual Y The beginning Y coordinate of the rectangular area to store. Y will be rounded down to the top of the current color cell.

DX - The length in pixels of the rectangle. The right side of the rectangle will

be adjusted to the end of the byte
it falls on.

DY - The height in pixels of the rectangle.
The endpoint in Y will be adjusted to
the end of the current color cell.

Compression - 0/1 0 indicates no data compression,
while 1 instructs the system to
compress the data. The amount of
compression depends upon the data
itself.

To find the next address in the buffer use the syntax
AD=@STASH,STRUCURE# BUFFER#,BUFFER ADDRESS,X,Y DX DY,COMPRESSION
AD=Next available address, or -1 indicating STASH failure.

The X,Y,DX,DY coordinates are scaled if necessary.

Chapter 10 - The BASIC 8.0 Library

(111)
@STORE Saves an entire screen as a BRUSH to disk for later @DISPLAY as a
screen or loading into a STRUCTURE as a BRUSH for use with @FETCH.

SYNTAX:
@STORE,SCREEN#,DEVICE #,COMPRESSION FLAG,"FILENAME"

SCREEN# - The number of the screen to store. See @SCREEN
command for information on screen numbers.

DEVICE # - The drive number you are using, ex. 8

```
COMPRESSION -  0 No data compression
               1 Compress data
** NOTE ** See appendix for data compression algorithm

FILENAME    -  The name the screen display is to be stored
               under on disk. It must be within double
               quotes (").

Chapter 10 - The BASIC 8.0 Library

(112)
@STRUCT  Defines data structure used in various BASIC 8.0 commands
         and where in the indicated BUPFER it is at.

SYNTAX:
@STRUCT,Structure #,Type,Buffer #, Beginning address in buffer

Structure #  -   0-191 Indicates which of 192 possible structures
                      you are defining. Structure numbers
                      greater than 191 are converted to 0-191.

Type         -   1-4  Indicates Structure data type
                   1  Pattern type
                   2  Logo type
                   3  Character Font type
                   4  Brush type

Buffer #     -   0-9 Indicates which of ten buffer banks to use.
                     0-1 is internal ram banks 0 and 1
                     2-9 are external ram banks 0-7

Buffer Address - 0-Buffer Size (Address to store data in buffer.)
```

Chapter 10 - The BASIC 8.0 Library

(113)

@STYLE Define characteristics for Rylander Solids

SYNTAX:

@STYLE,SHADE,SCALE,LIGHTING

Shade - 0/1 Sets shade style

0 textured

1 halftone

Scale - 0/1

0 Indicates no scaling, images are elongated

1 Images are symmetrical

Lighting - 0/1 Set lighting style

0 Normal

1 Backlite

Chapter 10 - The BASIC 8.0 Library

(114)

@TEXT

SYNTAX:

@TEXT

Clears text screen, initializes standard fonts, enters text mode. It also turns off the window border flag. @SCALE is set to 0,

all parameters of drawmodes A and B are set to 0, @ANGLE
is set to 0,0,0,0 and current @MODE is removed from system,
so @MODE must be re-issued or another @SCRDEF issued.

Chapter 10 - The BASIC 8.0 Library

(115)
@TOROID Draw a solid toroid shape

SYNTAX:
@TOROID,X,Y,INSIDE RADIUS,OUTSIDE RAD1US,VIEW

X - 0-XMax
Y - 0-YMax
Inside radius - 1-127
Outside radius - 1-127
View - 0/1/2
0 horizontal
1 vertical
2 top

Chapter 10 - The BASIC 8.0 Library

(116)
@VIEW

SYNTAX:
@VIEW,Angle

Angle - -360 to 360 Angle of view used when drawing
in parallel draw mode (zview flag of
@DRWMODB is equal to 1). Angle

is the rotation of your eye around
the Y axis. Angle is given in degrees.
Angles between -360 and 360 yield
all possible results. Greater values
are redundant.

Chapter 10 - The BASIC 8.0 Library

(117)

@WALRUS WALRUS logo, sets up the BASIC 8.0 system for 16K or 64K video ram.
@WALRUS must be executed at the beginning of a program.
Logo screen can be used as a title screen for BASIC 8.0 programs.

SYNTAX:

@WALRUS,RAM TYPE

RAM TYPE 0-1
0=16K 8563 video ram
1=64K 8563 video ram

NOTE: WALRUS command automatically goes to graphic mode and should
be used before any other BASIC 8 command.
If used within a program more than once, the @MODE command will
need to be issued again.

Chapter 10 - The BASIC 8.0 Library

(118)

@WINDOWOPEN Windowopen allows you to define a subscreen within the main
virtual screen area. The upper left corner of the window
becomes the pixel coordinates 0,0. All graphic commands
translate to this new origin. Any drawing that falls

outside the window boundaries is clipped (not rendered). This new window remains active until a @WINDOWCLOSE command is encountered ,a new window is defined or a @SCREEN command is issued.

SYNTAX:

@WINDOWOPEN,X,Y,window width,window height,border flag

X : 0-2046 This is top lefthand corner of window. Value indicates a pixel number of the virtual screen. Values greater than 639 only possible with 64K 8563 chip ram installed. While X is given in pixels, it rounds down to the next lowest column value. (1 column = 8 pixels)

Y : 0-818 This is the top scan line (pixel) of the window. Values greater than 200 are only possible with 64K 8563 chip ram installed.

window width : 1-2040 This is the width of the window in pixels and rounds up to the nearest column number. Values greater than 640 only possible with 64K 8563 chip ram installed.

window height: 1-819 This is the height of the window in scan lines (1 pixel). Values greater than 200 are only possible with 64K 8563 chip ram installed.

border flag : 0-l If set to one the window will have a border drawn at the outer coordinates of the window when a CLEAR command is issued.

NOTE: If window is opened in a color screen, the beginning scan line

number is rounded down to the closest color cell boundary. The window height is rounded up to the closest color cell boundary. This is done in order to prevent color bleed outside the window. @WINDOWOPEN does not save the contents of the area underneath the new window. Programmers who wish to preserve and restore this area should use the @STASH & @FETCH as appropriate.

Chapter 10 - The BASIC 8.0 Library

(119)

@MINDOWCLOSE Closes last window created with @WINDOWOPEN command and redirects graphic output of screen to the current virtual screen boundaries.

SYNTAX:

@WINDOWCLOSE

NOTES: Turns border flag off, does not remove contents of window from screen viewing. You must have previously @STASHed the screen area where the new window will appear and then restore the old screen area with a @FETCH command.

Chapter 10 - The BASIC 8.0 Library

(120)

@ZOOM This command enlarges a STASHed structure from memory to the screen.

SYNTAX:

E=@ZOOM,STRUCTURE#,SIZE,DestX,DestY

STRUCTURE# - 0-191 Number of an uncompressed brush structure

SIZE - 1-15 Height of ZOOMed brush in color cells

DestX - 0 MaxX Destination X of enlarged brush structure

DestY - 0-MaxY Destination Y of enlarged brush structure

ZOOM does little in the way of error correcting. YOU must make sure the new enlarged brush will fit into the destination area. The enlargement is constant in the X direction (1 pixel->8 pixels) and is variable in the Y. For example, a SIZE of 2 for a brush in the 8x2 mode would make each pixel in the Y direction enlarge a total of 4 pixels (2 scanlines per cell * size 2 = 4 pixels). For a brush in the 8x8 mode, a SIZE of 4 would make each pixel equal to 32 pixels. (8 scanlines per cell * size 4 = 32 pixels).

Chapter 10 - The BASIC 8.0 Library

(121)

CHART OF COMMANDS

EFFECTS AND RETURN VALUES

COMMAND	ROTATION	XYZ2XY	SCALE	CLIP	WINDOW	V RETURN	MMU	BUFFERS/BANKS
ANGLE								
ARC	+	+	+	+	+			
BOX	+	+	+	+	+			
BRUSHPATRN						-1		
BUFFER								0-10
CBRUSH						-1		
CHAR				+	+	-1		0-10,14
CIRCLE	+	+	+	AL	+			
CLEAR					+			
COLOR								
COPY				+				

CYLNDR	!		!		!		!		!		!		!		!
DIR$	!		!		!		!		!		!		!		!
DISPLAY	!		!		!		!	+	!		!	-1	!		!
DOT	!	+	!	+	!	+	!	AL	!	+	!		!		!
DRWMODA	!		!		!		!		!		!		!		!
DRWMODB	!		!		!		!		!		!		!		!
FETCH	!		!		!	+	!	+	!		!	-1	!		!
FLASH	!		!		!	+	!	+	!	+	!		!		!
FONT	!		!		!		!		!		!	-1	!		!
GROW	!		!		!		!		!		!		!		!
HCOPY	!		!		!		!		!		!		!		!
LINE	!	+	!	+	!	+	!	+	!	+	!		!		!
LOGO	!		!		!		!	+	!	+	!	-1	!		!
LSTRUCT	!		!		!		!		!		!		!		!
MODE	!		!		!		!		!		!		!		!
MOUSE	!		!		!		!		!		!		!		!
ORIGIN	!		!		!		!		!		!		!		!
PAINT	!		!		!	+	!	+	!	+	!	-1	!		!
PATTERN	!		!		!		!		!		!	-1	!		!
PIXEL	!		!		!	+	!	+	!	+	!		!		!
PTR	!		!		!		!		!		!		!		!
SCALE	!		!		!		!		!		!		!		!
SCLIP	!		!		!		!		!		!		!		!
SCRDEF	!		!		!		!		!		!		!		!
SCREEN	!		!		!		!		!		!		!		!
SCROLL	!		!		!		!		!		!		!		!
SDAT	!		!		!		!		!		!	-1/Adr	!		!
SEND	!		!		!		!		!		!		!		!
SPHERE	!		!		!		!		!		!		!		!
SPOOL	!		!		!		!		!		!		!		!
SSTRUCT	!		!		!		!		!		!		!		!
STASH	!		!		!		!	+	!		!	-1/Adr	!		!

STORE	!	!	!	!	+ !	!	-1	!	!	!
STRUCT	!	!	!	!	!	!	!	!	0-10	!
STYLE	!	!	!	!	!	!	!	!	!	!
TEXT	!	!	!	!	!	!	!	!	!	!
TOROID	!	!	!	!	!	!	!	!	!	!
VIEW	!	!	!	!	!	!	!	!	!	!
WALRUS	!	!	!	!	!	!	!	!	!	!
WINDOWCLOSE	!	!	!	!	!	!	!	!	!	!
WINDOWOPEN	!	!	!	!	!	!	!	!	!	!
ZOOM	!	!	!	!	!	!	!	!	!	!

Chapter 10 - The BASIC 8.0 Library

(122)

CHAPTER ELEVEN

PRINTER SUPPORT

BASIC 8.0 has some of the most flexible printer drivers available to users of computer graphics. It supports most of the major printers used by Commodore 128 owners. Since the different printers have different capabilities and require different programming techniques to print bitmap graphics, different printer specific drivers are required. These are installable printer drivers, which means you (or a program written by you) can easily change the specific printer being used by loading a driver with the command BLOAD "PRINTER NAME",B0,P26064.

You access the printer with the BASIC 8.0 command called @HCOPY. This command is the same regardless of what printer you are using, so your programs will always be compatible. The parameters of @HCOPY control the secondary address of the printer you are using, the size of the printout, the printer density to use, and the orientation of the printout (rotated or unrotated).

Since not all printers are the same (some offer more dots per horizontal line, some have several densities, and others

offer colors besides black) there may be limitations on what your printer can do. For example, the MPS 801 can print only 480 dots per horizontal line. So it cannot print out even the smallest unrotated BASIC 8.0 image (which would be at least 640 pixels wide.) In order to use it, the image is automatically rotated 90 degrees, regardless of the value of the rotate parameter in the @HCOPY command. This also means you cannot have a height parameter greater than 2 (400 pixels) with a 200 scanline hiqh screen. (A height of 3 would be 600, too large for the MPS 801.) And it offers only one density, so values greater than one are ignored. On the other extreme, the PANASONIC 1091 and EPSON FX
Chapter Eleven

(123)
both support seven dot densities. That means you could have a horizontal image upto 1920 dots wide (using density 7) before you had to rotate it. (The different density settings are not linear. This means a larger density value is not necessarily going to print a smaller picture. Instead it may instruct the printer to use more dots per inch, but still give the same size image as that of a smaller density setting. The best way of knowing what to expect from the various densities is to check with your printer manual, or simply print out a picture using different settings. We gave you the power to determine what the final printed output will look like.)

On the following pages is information concerning the eleven printers supported directly by BASIC 8.0. This information includes the assembly language source code, so other printer drivers could be written for BASIC 8.0 by experienced programmers. For additional information on the @HCOPY command, see the BASIC 8.0 COMMAND ENCYCLOPEDIA in chapter 10.
Chapter Eleven

(124)

EPSON FX-80

PANASONIC 1091

STAR NX-10

This printer driver is on your BASIC 8.0 disk with the name P.HC-EPSON. This is the default driver built into the BASIC 8.0 system, so it is not necessary to install it again unless another driver has been loaded.

The secondary address we used for the CARDCO G and G-WIZ interfaces is 5. Other printer interfaces may require a different secondary address.

This driver can use the full range of heights (1-4) in the height parameter of @HCOPY, as well as the full range of densities (1-7). For a 640 x 200 pixel screen a height of 2 with a density of 2 gives a very good proportional hardcopy of the screen. Or use a height of 1 with a density of 1 and your 640 x 200 image prints just about the size of a large business card. As mentioned, a density parameter of seven allows you to print a horizontal unrotated image of 1920 pixels, larger than most applications will require.

You can also print your images rotated 90 degrees, which is very useful if the screen is more than 1920 pixels wide, or you want to use a lower density in order to get a larger image. Just set the rotation parameter to one and the picture will be rotated.

This is the source code (written with the Commodore C128 Assembler Development Package). If you have a near EPSON compatible printer that doesn't work quite right, it may be possible for you (if you are a machine language programmer) to convert it to a new driver for your printer.

This source code is supplied as a courtesy to all registered

Chapter Eleven

(125)

BASIC 8.0 owners.

```
     EPSON/PANASONIC/STAR NX-10 BASIC 8.0 PRINTER SOURCE CODE
.PAGE
;NAME      HARD COPY
;CREATED   12/20/86
;UPDATED
;AUTHOR    DAVID DARUS
;ASSEMBLER HCD65
;COMPUTER  C-128
;REMARKS   BASIC 8 HARD COPY DRIVER FOR EPSON FX-NN,PANASONIC 1091,STAR NX
;                  X * 880 UNROTATED & ROTATED
*=$65D0

INCWRD .MACRO ?IARG1
       INC ?IARG1
       BNE 255$
       INC ?IARG1+1
255$
 .ENDM

DECWRD .MACRO ?DARG1
       LDA ?DARG1
       BNE 254$
       DEC ?DARG1+1
254$
       DEC ?DARG1
 .ENDM
```

```
MMU = $FF00

CHROUT = $FFD2
CLRCHN = $FFCC
CLOSE  = $FFC3
SETLFS = $FFBA
OPEN   = $FFC0
CHKOUT = $FFC9

CLCADR     = $26A8
SWAPNYBL   = $25DB
GETPIXSTAT = $2652
GETCOLSTAT = $266E

CURCOLOR = $1318
SCRCSIZE = $131A
SCRSIZEY = $131C
SCRSIZEX = $1324
XO       = $136B
YO       = $136D
MASK     = $137B
PARM1    = $13EF
PARM2    = $13F0
PARM3    = $13F1
PARM4    = $13F2

Chapter Eleven

(126)
MLHCOPY
  LDA PARM4
  BEQ 99$
```

```
  JMP MLHCOPYR
99$
  LDA PARM1
  STA SECADR
  LDA PARM2
  STA HCHT
  LDX PARM3       ;1=SINGLE  2=DOUBLE  3-7 SEE EPSON FX-80 MANUAL
  DEX
  STX ESCDAT+6

  LDA SCRSIZEX
  STA ESCDAT+7
  LDA SCRSIZEX+1
  STA ESCDAT+8

  LDX HCHT
  DEX
  LDA CNVTHT,X
  STA HCHT
  LDA #8
2$
  DEX
  BMI 1$
  LSR A
  JMP 2$
1$
  STA STEP

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
```

```
  LDA #7
  STA BN
  LDA #0
  STA BV    ;BYTE VALUE TO CHROUT=0
  STA YY1    ;Y0=0
  STA YY1+1
NXTY
  JSR PRTESC  ;DO FOR EACH ROW = 8*HEIGHT
  LDA #0
  STA X0
  STA X0+1
NXTX
  LDA YY1
  STA Y0
  LDA YY1+1
  STA Y0+1
  CLC
  LDA Y0
  ADC STEP
  STA ENDY0
  LDA Y0+1
  ADC #0
  STA ENDY0+1
```

Chapter Eleven

```
(127)
NXTY0
  LDA #0
  STA HTCNTR
NXTHT
  JSR GETPIX   ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
```

```
  ROL BV       ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN       ;BIT#=BIT#-1
  BPL 1$
  LDA BV
  JSR CHROUT
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INC HTCNTR
  LDA HTCNTR
  CMP HCHT
  BNE NXTHT
  INCWRD Y0
  LDA Y0
  CMP ENDY0
  BNE NXTY0
  LDA Y0+1
  CMP ENDY0+1
  BNE NXTY0
 .LOCAL
  INCWRD X0
  LDA X0
  CMP SCRSIZEX
  BNE NXTX      ;IF X0< SCRSIZEX THEN GOTO NXTY DO NEXT SCANLINE
  LDA X0+1
  CMP SCRSIZEX+1
  BNE NXTX
  CLC
  LDA YY1
  ADC STEP
```

```
  STA YY1
  LDA YY1+1
  ADC #0
  STA YY1+1
  LDA YY1
  CMP SCRSIZEY
  LDA YY1+1
  SBC SCRSIZEY+1
  BCC NXTYJ
HCERR
  LDA #27       ;ESC
  JSR CHROUT
  LDA #'@       ;RESET CODE
  JSR CHROUT
  JSR CLRCHN
  LDA #4
  JMP CLOSE
NXTYJ
  JMP NXTY

OPNCHN
```

Chapter Eleven

```
(128)
  LDA #0
  STA $B7
  LDX #4
  TXA
  LDY SECADR
  JSR SETLFS
  JSR OPEN
```

```
  LDA #4
  JMP CHKOUT

PRTESC
  LDA #0
1$
  LDA ESCDAT,X
  JSR CHROUT
  INX
  CPX #9
  BNE 1$
  RTS

GETPIX
  LDA Y0
  CMP SCRSIZEY
  LDA Y0+1
  SBC SCRSIZEY+1
  BCS 1$
  JSR CLCADR
  JSR GETPIXSTAT
  AND MASK
  BEQ 1$
  SEC
  RTS
1$
  CLC
  RTS

ESCDAT
 .BYT 13,10,27,'l',27,'*',0,<640,>640
CNVTHT
```

```
 .BYT 1,2,4,8    ;TRANSLATE HT OF 1,2,3,4 TO 1,2,4,8

SECADR
 .BYT 0
HCHT
 .BYT 0

BN
 .BYT 0
BV
 .BYT 0
HTCNTR
 .BYT 0
STEP
 .BYT 0
ENDY0
 .BYT 0,0
YY1
 .BYT 0,0
```

Chapter Eleven

(129)

```
MLHCOPYR
  LDA PARM1
  STA SECADR
  LDA PARM2
  STA HCHT
  LDX PARM3     ;1=SINGLE  2=DOUBLE
  DEX
  STX ESCDAT+6
```

```
  LDA SCRSIZEY
  STA ESCDAT+7
  LDA SCRSIZEY+1
  STA ESCDAT+8
  LDX HCHT
  DEX
  LDA ESCDAT+7
4$
  DEX
  BMI 3$
  ASL A
  ROL ESCDAT+8
  JMP 4$
3$
  STA ESCDAT+7

  LDX HCHT
  DEX
  LDA CNVTHT,X
  STA HCHT
  LDA #8
  STA STEP

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #7
  STA BN
  LDA #0
  STA BV       ;BYTE VALUE TO CHROUT=0
  STA XX1       ;X1=0
```

```
NXTX1R
  JSR PRTESC   ;DO FOR EACH ROW = 8*HEIGHT
  LDA SCRSIZEY
  STA Y0
  LDA SCRSIZEY+1
  STA Y0+1
  DECWRD Y0
NXTYR
  LDA #0
  STA HTCNTR
NXTHTR
  LDA XX1
  STA X0
  LDA XX1+1
```

Chapter Eleven

```
(130)
  STA X0+1
  CLC
  LDA X0
  ADC STEP
  STA ENDX0
  LDA X0+1
  ADC #0
  STA ENDX0+1
NXTXR
  JSR GETPIX    ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV        ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN        ;BIT#=BIT#-1
  BPL 1$
  LDA BV        ;IF BN<0 THEN PRINT BN:BN=7
```

```
  JSR CHROUT
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INCWRD X0
  LDA X0
  CMP ENDX0
  BNE NXTXR
  INC HTCNTR
  CMP HCHT
  BNE NXTHTR
 .LOCAL
  DECWRD Y0
  LDA Y0
  CMP #$FF
  BNE NXTYR     ;IF Y = 0 THEN GOTO NXTY  DO NEXT SCANLINE
  LDA Y0+1
  CMP #$FF
  BNE NXTYR
  CLC
  LDA XX1
  ADC STEP
  STA XX1
  LDA XX1+1
  ADC #0
  STA XX1+1
  LDA XX1
  CMP SCRSIZEX
  LDA XX1+1
  CMP SCRSIZEX+1
```

```
  BCC NXTXJR
HCERRR
  LDA #27         ;ESC
  JSR CHROUT
  LDA #'@         ;RESET CODE
  JSR CHROUT
  JSR CLRCHN
  LDA #4
```

Chapter Eleven

(131)

```
  JMP CLOSE
NXTXJR
  JMP NXTXR1

ENDX0
 .BYT 0,0
XX1
 .BYT 0,0

 .END
```

Chapter Eleven

(132)

OLIVETTI PR 2300

This printer driver is on your BASIC 8.0 disk with the name P.HC-OLIVETTI. This is not the default driver built into the BASIC 8.0 system, so it is necessary to install it with the Command BLOAD "P.HC-OLIVETTI",B0,P26064.

The secondary address we used for the CARDCO G and G-WIZ

interfaces is 5. Other printer interfaces may require a different secondary address.

This driver can use the full range of heights (1-4) in the height parameter of @HCOPY, but only 1 density is supported by the Olivetti. For a 640 x 200 pixel screen a height of 2 with a density of 1 gives a very good proportional hardcopy of the screen. Or use a height of 1 with a density of 1 and your 640 x 200 image prints just about the size of a large business envelope. The Olivetti PR2300 can print unrotated images upto 880 horizontal pixels wide.

You can also print your images rotated 90 degrees, which is very useful if the screen is more than 880 pixels wide. Just set the rotation parameter to one and the picture will be rotated.

This is the source code (written with the Commodore C128 Assembler Development Package). If you have a near OLIVETTI PR2300 compatible printer that doesn't work quite right, it may be possible for you (if you are a machine language programmer) to convert it to a new driver for your printer.

This source code is supplied as a courtesy to all registered BASIC 8.0 owners.

Chapter Eleven

(133)
OLIVETTI PR2300 BASIC 8.0 PRINTER SOURCE CODE

```
.PAGE
;NAME      HARD COPY
;CREATED   12/09/86
;UPDATED
;AUTHOR    DAVID DARUS
```

```
;ASSEMBLER HCD65
;COMPUTER  C-128
;REMARKS   BASIC 8 HARD COPY DRIVER FOR OLIVETTI PR2300
;                  880 * Y UNROTATED  X * 880 ROTATED
*=$65D0

INCWRD .MACRO ?IARG1
       INC ?IARG1
       BNE 255$
       INC ?IARG1+1
255$
 .ENDM

DECWRD .MACRO ?DARG1
       LDA ?DARG1
       BNE 254$
       DEC ?DARG1+1
254$
       DEC ?DARG1
 .ENDM

MMU = $FF00

CHROUT = $FFD2
CLRCHN = $FFCC
CLOSE  = $FFC3
SETLFS = $FFBA
OPEN   = $FFC0
CHKOUT = $FFC9

CLCADR     = $26A8
SWAPNYBL   = $25DB
```

```
GETPIXSTAT = $2652
GETCOLSTAT = $266E

CURCOLOR = $1318
SCRCSIZE = $131A
SCRSIZEY = $131C
SCRSIZEX = $1324
XO       = $136B
YO       = $136D
MASK     = $137B
PARM1    = $13EF
PARM2    = $13F0
PARM3    = $13F1
PARM4    = $13F2
MLHCOPY

Chapter Eleven

(134)
  LDA PARM4
  BEQ 99$
  JMP MLHCOPYR
99$
  LDA PARM1
  STA SECADR
  LDA PARM2
  STA HCHT

  LDA HCHT
  DEX
  LDA MULT8,X
  STA SSLN        ;SAVE SCANLINE#
```

```
  LDA HTABL,X
  STA ESCDAT+8
  LDA MMU
  JSR OPNCHN
  LDA #7
  STA BN
  LDA #0
  STA SLN          ;SCANLINE#=0
  STA BV           ;BYTE VALUE TO CHROUT = 0
  STA Y0           ;Y0=0
  STA Y0+1
NXTY
  LDA #0
  STA HTMULT       ;HEIGHT COUNTER
4$
  LDA SLN
  BNE 3$
  JSR PRTESC       ;DO FOR EACH ROW = 8*HEIGHT
3$
  INC SLN          ;INCREMENT SCANLINE# IN ROW
  LDA SLN
  CMP SSLN         ;IF SLN=SSLN THEN END OF ROW REACHED
  BNE 6$           ;BR IF <>
  LDA #0           ;ELSE
  STA SLN          ;RESET SCANLINE# TO 0
6$
  LDA #0
  STA X0
  STA X0+1
2$
  JSR GETPIX       ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN
CLC
```

```
  ROL BV          ;ROLL CARRY INTO BYTE VALUE  0 OR 1
  DEC BN          ;BIT#=BIT#-1
  BPL 1$
  LDA BV          ;IF BN<0 THEN PRINT BV;BN=7
  JSR CHROUT
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INCWRD X0
  LDA X0
Chapter Eleven

(135)
  CMP #<880
  BNE 2$          ;IF X0<880 THEN GOTO 2$
  LDA X0+1
  CMP #>880
  BNE 2$
  INC HTMULT      ;ELSE END OF LINE
  LDA HTMULT
  CMP HCHT
  BNE 4$          ;REPEAT SCANLINE HCHT TIME (HEIGHT)
 .LOCAL
  INCWRD Y0
  LDA Y0
  CMPSCRSIZEY
  BNE NXTY        ;IF Y0<SCRSIZEY THEN GOTO NXTY  DO NEXT SCANLINE
  LDA Y0+1
  CMP SCRSIZEY+1
  BNE NXTY
```

```
HCERR
  JSR CLRCHN
  LDA #4
  JMP CLOSE

OPNCHN
  LDA #4
  TXA
  LDY SECADR
  JSR SETLFS
  JSR OPEN
  LDX #4
  JMP CHKOUT

PRTESC
  LDX #0
1$
  LDA ESCDAT,X
  PHP
  AND #$7F
  JSR CHROUT
  INX
  PLP
  BPL 1$
  RTS

GETPIX
  LDA X0
  CMP SCRSIZEX
  LDA X0+1
  SBC SCRSIZEX+1
  BCS 1$
```

```
  JSR CLCADR
  JSR GETPIXSTAT
  AND MASK
  BEQ 1$
  SEC
  RTS
1$
  CLC
  RTS
```

Chapter Eleven

```
(136)
ESCDAT
 .BYT 27,71,'1;110;2;2',27,90+$80

SECADR
 .BYT 0
MULT8
 .BYT 8,16,24,32
HTABL
 .BYT 49,50,51,52
BN
 .BYT 0
BV
 .BYT 0
HCHT
 .BYT 0
HTMULT
 .BYT 0
SLN
 .BYT 0
SSLN
```

```
 .BYT 0

MLHCOPYR
  LDA PARM1
  STA SECADR
  LDA PARM2
  STA HCHT

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDX HCHT      ;1-4 GOES TO
  DEX           ;0-3 FOR INDEXING PURPOSES
  LDA HTABLR,X
  STA HCHT      ;1,2,4,8
  LDA #<880
  STA NUMSL
  LDA #>880
22$
  DEX
  BMI 11$
  LSR A
  ROR NUMSL
  JMP 22$
11$
  STA NUMSL+1
  LDA #7
  STA BN
  LDA #0
  STA SLN      ;SCANLINE#=0
  STA BV       ;BYTE VALUE TO CHROUT = 0
```

```
  STA X0        ;X0=0
NXTXR
  LDA SLN
Chapter Eleven


(137)
  BNE 3$
  JSR PRTESCR   ;DO FOR EACH ROW=8*HEIGHT
3$
  INC SLN       ;INCREMENT SCANLINE# IN ROW
  LDA SLN
  CMP #8        ;IF SLN=8 THEN END OF ROW REACHED
  BNE 6$        ;BR IF <>
  LDA #0        ;ELSE
  STA SLN       ;RESET SCANLINE# TO 0
6$
  LDA NUMSL
  STA Y0
  LDA NUMSL+1
  STA Y0+1
  DECWRD Y0
NXTYR
  LDA #0
  STA HTMULT
4$
  JSR GETPIX       ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN
CLC
  ROL BV           ;ROLL CARRY INTO BYTE VALUE  0 OR 1
  DEC BN           ;BIT#=BIT#-1
  BPL 1$
  LDA BV           ;IF BN<0 THEN PRINT BV;BN=7
  JSR CHROUT
```

```
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INC HTMULT
  LDA HTMULT
  CMP HCHT
  BNE 4$
  DECWRD Y0
  LA Y0
  CMP #$FF
  BNE NXTYR        ;YO <> 0 THEN GOTO NXTY
  LDA Y0+1
  CMP #$FF
  BNE NXTYR
 .LOCAL
  INCWRD X0
  LDA X0
  CMP SCRSIZEX
  BNE NXTXTR       ;IF X0<SCRSIZEX THEN GOTO NXTY  DO NEXT SCANLINE
  LDA X0+1
  CMP SCRSIZEX+1
  BNE NXTXTR

HCERR
  JSR CLRCHN
  LDA #4
  JMP CLOSE

NXTXTR
  JMP NXTXR
```

Chapter Eleven

```
(138)
PRTESCR
  LDX #0
1$
  LDA ESCDATR,X
  PHP
  AND #$7F
  JSR CHROUT
  INX
  PLP
  BPL 1$
  RTS

GETPIXR
  LDA Y0
  CMP SCRSIZEY
  LDA Y0+1
  SBC SCRSIZEY+1
  BCS 1$
  JSR CLCADR
  JSR GETPIXSTAT
  AND MASK
  BEQ 1$
  SEC
  RTS
1$
  CLC
  RTS

ESCDATR
```

```
 .BYT 27,71,'1;110;1;2',27,90+$80
HTABLR
 .BYT 1,2,4,8    ;TRANSLATE HT OF 1,2,3,4 TO 1,2,4,8
NUMSL
 .BYT 0,0

 .END
```

Chapter Eleven

(139)

STAR NX-10C

This printer driver is on your BASIC 8.0 disk with the name P.HC-NX-10C. This is not the default driver built into the BASIC 8.0 system, so it is necessary to install it with the Command BLOAD "P.HC-NX-10C",B0,P26064.

The secondary address to use is 8.

This driver can use the full range of heights (1-4) in the height parameter of @HCOPY, but only 2 densities are supported by the NX-10C. For a 640 x 200 pixel screen a height of 2 with a density of 2 gives a very good proportional hardcopy of the screen. Or use a height of 1 with a density of 1 and your 640 x 200 image prints just about the size of a small envelope. The NX-10C can print unrotated images upto 1280 horizontal pixels wide.

You can also print your images rotated 90 degrees, which is very useful if the screen is more than 1280 pixels wide. Just set the rotation parameter to one and the picture will be rotated.

This is the source code (written with the Commodore C128 Assembler Development Package). If you have a near NX-10C compatible printer that doesn't work quite right, it may be possible for you (if you are a machine language programmer) to

```
convert it to a new driver for your printer.
    This source code is supplied as a courtesy to all registered
BASIC 8.0 owners.

Chapter Eleven
(140)
NX-10C BASIC 8.0 PRINTER SOURCE CODE

.PAGE
;NAME      HARD COPY
;CREATED   12/20/86
;UPDATED
;AUTHOR    DAVID DARUS
;ASSEMBLER HCD65
;COMPUTER  C-128
;REMARKS   BASIC 8 HARD COPY DRIVER FOR STAR NX-10C
;                  X * 880 UNROTATED
*=$65D0

INCWRD .MACRO ?IARG1
       INC ?IARG1
       BNE 255$
       INC ?IARG1+1
255$
 .ENDM

DECWRD .MACRO ?DARG1
       LDA ?DARG1
       BNE 254$
       DEC ?DARG1+1
254$
       DEC ?DARG1
```

```
 .ENDM

MMU = $FF00

CHROUT = $FFD2
CLRCHN = $FFCC
CLOSE  = $FFC3
SETLFS = $FFBA
OPEN   = $FFC0
CHKOUT = $FFC9

CLCADR     = $26A8
SWAPNYBL   = $25DB
GETPIXSTAT = $2652
GETCOLSTAT = $266E

CURCOLOR = $1318
SCRCSIZE = $131A
SCRSIZEY = $131C
SCRSIZEX = $1324
XO       = $136B
YO       = $136D
MASK     = $137B
PARM1    = $13EF
PARM2    = $13F0
PARM3    = $13F1
PARM4    = $13F2

MLHCOPY
  LDA PARM4
  BEQ 99$
```

```
(141)
  JMP MLHCOPYR
99$
  LDA PARM1
  STA SECADR
  LDA PARM2
  STA HCHT
  LDX PARM3        ;1=SINGLE  2=DOUBLE
  DEX
  LDA DENSITY,X
  STA ESCDAT+6

  LDA SCRSIZEX
  STA ESCDAT+7
  LDA SCRSIZEX+1
  STA ESCDAT+8

  LDX HCHT
  DEX
  LDA CNVTHT,X
  STA HCHT
  LDA #8
2$
  DEX
  BMI 1$
  LSR A
  JMP 2$
1$
  STA STEP

  LDA MMU
```

```
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #7
  STA BN
  LDA #0
  STA BV     ;BYTE VALUE TO CHROUT=0
  STA Y1    ;Y0=0
  STA Y1+1
NXTY
  JSR PRTESC  ;DO FOR EACH ROW = 8*HEIGHT
  LDA #0
  STA X0
  STA X0+1
NXTX
  LDA Y1
  STA Y0
  LDA Y1+1
  STA Y0+1
  CLC
  LDA Y0
  ADC STEP
  STA ENDY0
  LDA Y0+1
  ADC #0
  STA ENDY0+1
NXTY0
  LDA #0
```

Chapter Eleven

(142)

```
  STA HTCNTR
```

```
NXTHT
  JSR GETPIX   ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV       ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN       ;BIT#=BIT#-1
  BPL 1$
  LDA BV
  JSR CHROUT
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INC HTCNTR
  LDA HTCNTR
  CMP HCHT
  BNE NXTHT
  INCWRD Y0
  LDA Y0
  CMP ENDY0
  BNE NXTY0
  LDA Y0+1
  CMP ENDY0+1
  BNE NXTY0
 .LOCAL
  INCWRD X0
  LDA X0
  CMP SCRSIZEX
  BNE NXTX      ;IF X0< SCRSIZEX THEN GOTO NXTY DO NEXT SCANLINE
  LDA X0+1
  CMP SCRSIZEX+1
  BNE NXTX
  CLC
```

```
  LDA Y1
  ADC STEP
  STA Y1
  LDA Y1+1
  ADC #0
  STA Y1+1
  LDA Y1
  CMP SCRSIZEY
  LDA Y1+1
  SBC SCRSIZEY+1
  BCC NXTYJ
HCERR
  LDA #27       ;ESC
  JSR CHROUT
  LDA #'@       ;RESET CODE
  JSR CHROUT
  JSR CLRCHN
  LDA #4
  JMP CLOSE
NXTYJ
  JMP NXTY

OPNCHN
  LDA #0
  STA $B7
```

Chapter Eleven

```
(143)
  LDX #4
  TXA
  LDY SECADR
  JSR SETLFS
```

```
  JSR OPEN
  LDA #4
  JMP CHKOUT

PRTESC
  LDA #0
1$
  LDA ESCDAT,X
  JSR CHROUT
  INX
  CPX #8
  BNE 1$
  RTS

GETPIX
  LDA Y0
  CMP SCRSIZEY
  LDA Y0+1
  SBC SCRSIZEY+1
  BCS 1$
  JSR CLCADR
  JSR GETPIXSTAT
  AND MASK
  BEQ 1$
  SEC
  RTS
1$
  CLC
  RTS

ESCDAT
 .BYT 13,27,'3',21,27,'K',<640,>640
```

```
CNVTHT
 .BYT 1,2,4,8    ;TRANSLATE HT OF 1,2,3,4 TO 1,2,4,8
DENSITY
 .BYT 'K','L'

SECADR
 .BYT 0
HCHT
 .BYT 0

BN
 .BYT 0
BV
 .BYT 0
HTCNTR
 .BYT 0
STEP
 .BYT 0
ENDY0
 .BYT 0,0
Y1
 .BYT 0,0
Chapter Eleven

(144)
MLHCOPYR
  LDA PARM1
  STA SECADR
  LDA PARM2
  STA HCHT
  LDX PARM3     ;1=SINGLE  2=DOUBLE
  DEX
```

```
  LDA DENSITY,X
  STA ESCDAT+6

  LDA SCRSIZEY
  STA ESCDAT+7
  LDA SCRSIZEY+1
  STA ESCDAT+8
  LDX HCHT
  DEX
  LDA ESCDAT+7
4$
  DEX
  BMI 3$
  ASL A
  ROL ESCDAT+8
  JMP 4$
3$
  STA ESCDAT+7

  LDX HCHT
  DEX
  LDA CNVTHT,X
  STA HCHT
  LDA #8
  STA STEP

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #7
  STA BN
```

```
  LDA #0
  STA BV      ;BYTE VALUE TO CHROUT=0
  STA X1      ;X1=0
  STA X1+1
NXTX1R
  JSR PRTESC   ;DO FOR EACH ROW = 8*HEIGHT
  LDA SCRSIZEY
  STA Y0
  LDA SCRSIZEY+1
  STA Y0+1
  DECWRD Y0
NXTYR
  LDA #0
  STA HTCNTR
NXTHTR
  LDA X1
  STA X0
```

Chapter Eleven

(145)

```
  LDA X1+1
  STA X0+1
  CLC
  LDA X0
  ADC STEP
  STA ENDX0
  LDA X0+1
  ADC #0
  STA ENDX0+1
NXTXR
  JSR GETPIX    ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV        ;ROLL CARRY INTO BYTE VALUE 0 OR 1
```

```
  DEC BN        ;BIT#=BIT#-1
  BPL 1$
  LDA BV        ;IF BN<0 THEN PRINT BN:BN=7
  JSR CHROUT
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INCWRD X0
  LDA X0
  CMP ENDX0
  BNE NXTXR
  LDA X0+1
  CMP ENDX0+1
  BNE NXTXR
  INC HTCNTR
  LDA HTCNTR
  CMP HCHT
  BNE NXTHTR
 .LOCAL
  DECWRD Y0
  LDA Y0
  CMP #$FF
  BNE NXTYR     ;IF Y = 0 THEN GOTO NXTY  DO NEXT SCANLINE
  LDA Y0+1
  CMP #$FF
  BNE NXTYR
  CLC
  LDA X1
  ADC STEP
  STA X1
```

```
  LDA X1+1
  ADC #0
  STA X1+1
  LDA X1
  CMP SCRSIZEX
  LDA X1+1
  SBC SCRSIZEX+1
  BCC NXTXJR
HCERRR
  LDA #27        ;ESC
  JSR CHROUT
  LDA #'@        ;RESET CODE
  JSR CHROUT
  JSR CLRCHN

Chapter Eleven

(146)
  LDA #4
  JMP CLOSE
NXTXJR
  JMP NXTXR1

ENDX0
 .BYT 0,0
X1
 .BYT 0,0

 .END

Chapter Eleven
```

(147)

MANNESMANN TALLY SPIRIT 80

This printer driver is on your BASIC 8.0 disk with the name P.HC-SPIRIT80. This is not the default driver built into the BASIC 8.0 system, so it is necessary to install it with the Command BLOAD "P.HC-SPIRIT80",B0,P26064.

The secondary address to use is 5, if you are using the CARDCO G or G-WIZ interfaces. Other interfaces may require a Different secondary address.

This driver can use the full range of heights (1-4) in the height parameter of @HCOPY, but only 2 densities are supported by the SPIRIT 80. For a 640 x 200 pixel screen a height of 2 with a density of 2 gives a very good proportional hardcopy of the screen. Or use a height of 1 with a density of 1 and your 640 x 200 image prints just about the size of a small envelope. The SPIRIT 80 can print unrotated images upto 1280 horizontal pixels wide.

You can also print your images rotated 90 degrees, which is very useful if the screen is more than 1280 pixels wide. Just set the rotation parameter to one and the picture will be rotated.

This is the source code (written with the Commodore C128 Assembler Development Package). If you have a near SPIRIT 80 compatible printer that doesn't work quite right, it may be possible for you (if you are a machine language programmer) to convert it to a new driver for your printer.

This source code is supplied as a courtesy to all registered BASIC 8.0 owners.

Chapter Eleven

(148)

MANNESMANN TALLY SPIRIT 80 BASIC 8.0 PRINTER SOURCE CODE

```
.PAGE
;NAME      HARD COPY
;CREATED   12/20/86
;UPDATED
;AUTHOR    DAVID DARUS
;ASSEMBLER HCD65
;COMPUTER  C-128
;REMARKS   BASIC 8 HARD COPY DRIVER FOR SPIRIT80
;                  X * 880 UNROTATED
*=$65D0

INCWRD .MACRO ?IARG1
       INC ?IARG1
       BNE 255$
       INC ?IARG1+1
255$
 .ENDM

DECWRD .MACRO ?DARG1
       LDA ?DARG1
       BNE 254$
       DEC ?DARG1+1
254$
       DEC ?DARG1
 .ENDM

MMU = $FF00

CHROUT = $FFD2
CLRCHN = $FFCC
CLOSE  = $FFC3
```

```
SETLFS = $FFBA
OPEN   = $FFC0
CHKOUT = $FFC9

CLCADR     = $26A8
SWAPNYBL   = $25DB
GETPIXSTAT = $2652
GETCOLSTAT = $266E

CURCOLOR = $1318
SCRCSIZE = $131A
SCRSIZEY = $131C
SCRSIZEX = $1324
XO       = $136B
YO       = $136D
MASK     = $137B
PARM1    = $13EF
PARM2    = $13F0
PARM3    = $13F1
PARM4    = $13F2

MLHCOPY
  LDA PARM4
  BEQ 99$
  JMP MLHCOPYR
Chapter Eleven

(149)
99$
  LDA PARM1
  STA SECADR
  LDA PARM2
```

```
  STA HCHT
  LDX PARM3        ;1=SINGLE  2=DOUBLE
  DEX
  LDA DENSITY,X
  STA ESCDAT+5

  LDA SCRSIZEX
  STA ESCDAT+6
  LDA SCRSIZEX+1
  STA ESCDAT+7

  LDX HCHT
  DEX
  LDA CNVTHT,X
  STA HCHT
  LDA #8
2$
  DEX
  BMI 1$
  LSR A
  JMP 2$
1$
  STA STEP

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #7
  STA BN
  LDA #0
  STA BV     ;BYTE VALUE TO CHROUT=0
```

```
  STA Y1    ;Y0=0
  STA Y1+1
NXTY
  JSR PRTESC  ;DO FOR EACH ROW = 8*HEIGHT
  LDA #0
  STA X0
  STA X0+1
NXTX
  LDA Y1
  STA Y0
  LDA Y1+1
  STA Y0+1
  CLC
  LDA Y0
  ADC STEP
  STA ENDY0
  LDA Y0+1
  ADC #0
  STA ENDY0+1
NXTY0
  LDA #0
  STA HTCNTR
```

Chapter Eleven

```
(150)
NXTHT
  JSR GETPIX   ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV       ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN       ;BIT#=BIT#-1
  BPL 1$
  LDA BV
  JSR CHROUT
```

```
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INC HTCNTR
  LDA HTCNTR
  CMP HCHT
  BNE NXTHT
  INCWRD Y0
  LDA Y0
  CMP ENDY0
  BNE NXTY0
  LDA Y0+1
  CMP ENDY0+1
  BNE NXTY0
 .LOCAL
  INCWRD X0
  LDA X0
  CMP SCRSIZEX
  BNE NXTX      ;IF X0< SCRSIZEX THEN GOTO NXTY DO NEXT SCANLINE
  LDA X0+1
  CMP SCRSIZEX+1
  BNE NXTX
  CLC
  LDA Y1
  ADC STEP
  STA Y1
  LDA Y1+1
  ADC #0
  STA Y1+1
  LDA Y1
```

```
  CMP SCRSIZEY
  LDA Y1+1
  SBC SCRSIZEY+1
  BCC NXTYJ
HCERR
  JSR CLRCHN
  LDA #4
  JMP CLOSE
NXTYJ
  JMP NXTY

OPNCHN
  LDA #0
  STA $B7
  LDX #4
  TXA
  LDY SECADR
  JSR SETLFS
  JSR OPEN
```

Chapter Eleven

```
(151)
  LDA #4
  JMP CHKOUT

PRTESC
  LDA #0
1$
  LDA ESCDAT,X
  JSR CHROUT
  INX
  CPX #8
```

```
  BNE 1$
  RTS

GETPIX
  LDA Y0
  CMP SCRSIZEY
  LDA Y0+1
  SBC SCRSIZEY+1
  BCS 1$
  JSR CLCADR
  JSR GETPIXSTAT
  AND MASK
  BEQ 1$
  SEC
  RTS
1$
  CLC
  RTS

ESCDAT
 .BYT 13,10,27,'1',27,'K',<640,>640
CNVTHT
 .BYT 1,2,4,8    ;TRANSLATE HT OF 1,2,3,4 TO 1,2,4,8
DENSITY
 .BYT 'K','L'

SECADR
 .BYT 0
HCHT
 .BYT 0

BN
```

```
 .BYT 0
BV
 .BYT 0
HTCNTR
 .BYT 0
STEP
 .BYT 0
ENDY0
 .BYT 0,0
Y1
 .BYT 0,0

MLHCOPYR
  LDA PARM1
  STA SECADR
Chapter Eleven

(152)
  LDA PARM2
  STA HCHT
  LDX PARM3     ;1=SINGLE  2=DOUBLE
  DEX
  LDA DENSITY,X
  STA ESCDAT+5
  LDA SCRSIZEY
  STA ESCDAT+6
  LDA SCRSIZEY+1
  STA ESCDAT+7
  LDX HCHT
  DEX
  LDA ESCDAT+6
4$
```

```
  DEX
  BMI 3$
  ASL A
  ROL ESCDAT+7
  JMP 4$
3$
  STA ESCDAT+6

  LDX HCHT
  DEX
  LDA CNVTHT,X
  STA HCHT
  LDA #8
  STA STEP

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #7
  STA BN
  LDA #0
  STA BV      ;BYTE VALUE TO CHROUT=0
  STA X1      ;X1=0
  STA X1+1
NXTX1R
  JSR PRTESC   ;DO FOR EACH ROW = 8*HEIGHT
  LDA SCRSIZEY
  STA Y0
  LDA SCRSIZEY+1
  STA Y0+1
  DECWRD Y0
```

```
NXTYR
  LDA #0
  STA HTCNTR
NXTHTR
  LDA X1
  STA X0
  LDA X1+1
  STA X0+1
  CLC
  LDA X0
  ADC STEP
```

Chapter Eleven

```
(153)
  STA ENDX0
  LDA X0+1
  ADC #0
  STA ENDX0+1
NXTXR
  JSR GETPIX    ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV        ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN        ;BIT#=BIT#-1
  BPL 1$
  LDA BV        ;IF BN<0 THEN PRINT BN:BN=7
  JSR CHROUT
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INCWRD X0
  LDA X0
```

```
  CMP ENDX0
  BNE NXTXR
  LDA X0+1
  CMP ENDX0+1
  BNE NXTXR
  INC HTCNTR
  LDA HTCNTR
  CMP HCHT
  BNE NXTHTR
 .LOCAL
  DECWRD Y0
  LDA Y0
  CMP #$FF
  BNE NXTYR     ;IF Y = 0 THEN GOTO NXTY  DO NEXT SCANLINE
  LDA Y0+1
  CMP #$FF
  BNE NXTYR
  CLC
  LDA X1
  ADC STEP
  STA X1
  LDA X1+1
  ADC #0
  STA X1+1
  LDA X1
  CMP SCRSIZEX
  LDA X1+1
  SBC SCRSIZEX+1
  BCC NXTXJR
HCERRR
  JSR CLRCHN
  LDA #4
```

```
  JMP CLOSE
NXTXJR
  JMP NXTXR1

ENDX0
 .BYT 0,0
X1
 .BYT 0,0
```

Chapter Eleven

```
(154)
 .END
```

Chapter Eleven

```
(155)
```

GEMINI II

This printer driver is on your BASIC 8.0 disk with the name P.HC-GEMINI2. This is not the default driver built into the BASIC 8.0 system, so it is necessary to install it with the Command BLOAD "P.HC-GEMINI2",B0,P26064.

The secondary address to use is 8.

This driver can only use the range of heights (1-2) in the height parameter of @HCOPY, and only 2 densities are supported by the Gemini II. For a 640 x 200 pixel screen a height of 2 with a density of 1 gives a very good proportional hardcopy of the screen, although the image is always rotated 90 degrees. Or use a height of 1 with a density of 1 and your 640 x 200 image prints just about the size of a small envelope. The Gemini II can only print unrotated images upto 480 horizontal pixels wide, and since the smallest screen is at least 640 pixels wide the image is always rotated.

This is the source code (written with the Commodore C128

Assembler Development Package). If you have a near GEMINI II compatible printer that doesn't work quite right, it may be possible for you (if you are a machine language programmer) to convert it to a new driver for your printer.

This source code is supplied as a courtesy to all registered BASIC 8.0 owners.

Chapter Eleven

(156)

GEMINI II BASIC 8.0 PRINTER SOURCE CODE

```
.PAGE
;NAME      HARD COPY
;CREATED   12/20/86
;UPDATED
;AUTHOR    DAVID DARUS
;ASSEMBLER HCD65
;COMPUTER  C-128
;REMARKS   BASIC 8 HARD COPY DRIVER FOR GEMINI II
;                  X * 480 ROTATED
*=$65D0

INCWRD .MACRO ?IARG1
       INC ?IARG1
       BNE 255$
       INC ?IARG1+1
255$
 .ENDM

DECWRD .MACRO ?DARG1
       LDA ?DARG1
```

```
        BNE 254$
        DEC ?DARG1+1
254$
        DEC ?DARG1
 .ENDM

MMU = $FF00

CHROUT = $FFD2
CLRCHN = $FFCC
CLOSE  = $FFC3
SETLFS = $FFBA
OPEN   = $FFC0
CHKOUT = $FFC9

CLCADR     = $26A8
SWAPNYBL   = $25DB
GETPIXSTAT = $2652
GETCOLSTAT = $266E

CURCOLOR = $1318
SCRCSIZE = $131A
SCRSIZEY = $131C
SCRSIZEX = $1324
XO       = $136B
YO       = $136D
MASK     = $137B
PARM1    = $13EF
PARM2    = $13F0
PARM3    = $13F1
PARM4    = $13F2
```

```
MLHCOPY
  LDA PARM1
  STA SECADR
  LDA PARM2
Chapter Eleven

(157)
  STA HCHT
  LDX PARM3       ;1-2
  DEX             ;CONVERTED TO 0-1
  LDA DENSITY,X   ;DENSITY CODE 8 OR 9
  STA ESCDAT+1

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #6
  STA BN
  LDA #0
  STA BV      ;BYTE VALUE TO CHROUT=0
  STA X1      ;X1=0
  STA X1+1
NXTX1
  JSR PRTESC
  LDA SCRSIZEY
  STA Y0
  LDA SCRSIZEY+1
  STA Y0+1
  DECWRD Y0
NXTY
  LDA #0
```

```
  STA HTCNTR
NXTHT
  LDA X1
  STA X0
  LDA X1+1
  STA X0+1
  CLC
  LDA X0
  ADC STEP
  STA ENDX0
  LDA X0+1
  ADC #0
  STA ENDX0+1
NXTX
  JSR GETPIX    ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV        ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN        ;BIT#=BIT#-1
  BPL 1$
  SEC
  ROR BV
  LDA BV        ;IF BN<0 THEN PRINT BN:BN=7
  JSR CHROUT
  LDA #6
  STA BN
  LDA #0
  STA BV
1$
  INCWRD X0
  LDA X0
  CMP ENDX0
```

```
(158)
  BNE NXTX
  LDA X0+1
  CMP ENDX0+1
  BNE NXTXR
  INC HTCNTR
  LDA HTCNTR
  CMP HCHT
  BNE NXTHT
 .LOCAL
  DECWRD Y0
  LDA Y0
  CMP #$FF
  BNE NXTY     ;IF Y = 0 THEN GOTO NXTY  DO NEXT SCANLINE
  LDA Y0+1
  CMP #$FF
  BNE NXTY
  CLC
  LDA X1
  ADC STEP
  STA X1
  LDA X1+1
  ADC #0
  STA X1+1
  LDA X1
  CMP SCRSIZEX
  LDA X1+1
  SBC SCRSIZEX+1
  BCC NXTXJ
HCERR
  LDA #13        ;CARRIAGE RETURN TO CLEAR OUT BUFFER
  JSR CHROUT
```

```
  LDA #15         ;SET TO NORMAL TEXT MODE
  JSR CHROUT
  JSR CLRCHN
  LDA #4
  JMP CLOSE
NXTXJ
  JMP NXTX1

OPNCHN
  LDA #0
  STA $B7
  LDX #4
  TXA
  LDY SECADR
  JSR SETLFS
  JSR OPEN
  LDA #4
  JMP CHKOUT

PRTESC
  LDA #0
1$
  LDA ESCDAT,X
  JSR CHROUT
  INX
  CPX #2
  BNE 1$
Chapter Eleven

(159)
  RTS
```

```
GETPIX
  LDA X0
  CMP SCRSIZEX
  LDA X0+1
  SBC SCRSIZEX+1
  BCS 1$
  JSR CLCADR
  JSR GETPIXSTAT
  AND MASK
  BEQ 1$
  SEC
  RTS
1$
  CLC
  RTS

ESCDAT
 .BYT 13,8       ;CR,GRAPHIC MODE

SECADR
 .BYT 0

HCHT
 .BYT 0
HTCNTR
 .BYT 0

DENSITY
 .BYT 8,9

BN
 .BYT 0
```

```
BV
 .BYT 0
STEP
 .BYT 0
ENDX0
 .BYT 0,0
X1
 .BYT 0,0

 .END
```

Chapter Eleven

(160)

MPS 801/SEIKOSHA 1000/GEMINI 10X

This printer driver is on your BASIC 8.0 disk with the name P.HC-MPS801. This is not the default driver built into the BASIC 8.0 system, so it is necessary to install it with the Command BLOAD "P.HC-MPS801",B0,P26064.

The secondary address to use is 8 for the MPS 801, 8 for The GEMINI 10X and 0 for the SEIKOSHA.

This driver can only use the range of heights (1-2) in the height parameter of @HCOPY, and only 1 density is supported by these printers. For a 640 x 200 pixel screen a height of 2 with a density of 1 gives a very good proportional hardcopy of the screen, although the image is always rotated 90 degrees. Or use a height of 1 with a density of 1 and your 640 x 200 image prints just about the size of a small envelope. The MPS 801 and compatibles can only print unrotated images upto 480 horizontal pixels wide, which is why the image is always rotated.

This is the source code (written with the Commodore C128 Assembler Development Package). If you have a near MPS 801 compatible printer that doesn't work quite right, it may

be possible for you (if you are a machine language programmer) to convert it to a new driver for your printer.

This source code is supplied as a courtesy to all registered BASIC 8.0 owners.

Chapter Eleven

(161)

GEMINI II BASIC 8.0 PRINTER SOURCE CODE

```
.PAGE
;NAME      HARD COPY
;CREATED   12/20/86
;UPDATED
;AUTHOR    DAVID DARUS
;ASSEMBLER HCD65
;COMPUTER  C-128
;REMARKS   BASIC 8 HARD COPY DRIVER FOR MPS-801,SEIKOSHA,GEMINI 10X
;                  X * 480 ROTATED
*=$65D0

INCWRD .MACRO ?IARG1
       INC ?IARG1
       BNE 255$
       INC ?IARG1+1
255$
 .ENDM

DECWRD .MACRO ?DARG1
       LDA ?DARG1
       BNE 254$
       DEC ?DARG1+1
```

```
254$
       DEC ?DARG1
 .ENDM

MMU = $FF00

CHROUT = $FFD2
CLRCHN = $FFCC
CLOSE  = $FFC3
SETLFS = $FFBA
OPEN   = $FFC0
CHKOUT = $FFC9

CLCADR     = $26A8
SWAPNYBL   = $25DB
GETPIXSTAT = $2652
GETCOLSTAT = $266E

CURCOLOR = $1318
SCRCSIZE = $131A
SCRSIZEY = $131C
SCRSIZEX = $1324
XO       = $136B
YO       = $136D
MASK     = $137B
PARM1    = $13EF
PARM2    = $13F0
PARM3    = $13F1
PARM4    = $13F2

MLHCOPY
  LDA PARM1
```

```
  STA SECADR
  LDA PARM2
Chapter Eleven

(162)
  STA HCHT

  LDA #7
  STA STEP

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #6
  STA BN
  LDA #0
  STA BV      ;BYTE VALUE TO CHROUT=0
  STA X1      ;X1=0
  STA X1+1
NXTX1
  JSR PRTESC
  LDA SCRSIZEY
  STA Y0
  LDA SCRSIZEY+1
  STA Y0+1
  DECWRD Y0
NXTY
  LDA #0
  STA HTCNTR
NXTHT
  LDA X1
```

```
  STA X0
  LDA X1+1
  STA X0+1
  CLC
  LDA X0
  ADC STEP
  STA ENDX0
  LDA X0+1
  ADC #0
  STA ENDX0+1
NXTX
  JSR GETPIX    ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV        ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN        ;BIT#=BIT#-1
  BPL 1$
  SEC
  ROR BV
  LDA BV        ;IF BN<0 THEN PRINT BN:BN=7
  JSR CHROUT
  LDA #6
  STA BN
  LDA #0
  STA BV
1$
  INCWRD X0
  LDA X0
  CMP ENDX0
  BNE NXTX
  LDA X0+1
  CMP ENDX0+1
  BNE NXTXR
```

```
(163)
  INC HTCNTR
  LDA HTCNTR
  CMP HCHT
  BNE NXTHT
 .LOCAL
  DECWRD Y0
  LDA Y0
  CMP #$FF
  BNE NXTY     ;IF Y = 0 THEN GOTO NXTY  DO NEXT SCANLINE
  LDA Y0+1
  CMP #$FF
  BNE NXTY
  CLC
  LDA X1
  ADC STEP
  STA X1
  LDA X1+1
  ADC #0
  STA X1+1
  LDA X1
  CMP SCRSIZEX
  LDA X1+1
  SBC SCRSIZEX+1
  BCC NXTXJ
HCERR
  LDA #13
  JSR CHROUT
  LDA #15
  JSR CHROUT
  JSR CLRCHN
```

```
  LDA #4
  JMP CLOSE
NXTXJ
  JMP NXTX1

OPNCHN
  LDA #0
  STA $B7
  LDX #4
  TXA
  LDY SECADR
  JSR SETLFS
  JSR OPEN
  LDA #4
  JMP CHKOUT

PRTESC
  LDA #0
1$
  LDA ESCDAT,X
  JSR CHROUT
  INX
  CPX #2
  BNE 1$
  RTS

GETPIX
  LDA X0
```



```
(164)
  CMP SCRSIZEX
```

```
  LDA X0+1
  SBC SCRSIZEX+1
  BCS 1$
  JSR CLCADR
  JSR GETPIXSTAT
  AND MASK
  BEQ 1$
  SEC
  RTS
1$
  CLC
  RTS

ESCDAT
 .BYT 13,8       ;CR,GRAPHIC MODE

SECADR
 .BYT 0

HCHT
 .BYT 0
HTCNTR
 .BYT 0

BN
 .BYT 0
BV
 .BYT 0
STEP
 .BYT 0
ENDX0
 .BYT 0,0
```

```
X1
 .BYT 0,0

 .END
```

Chapter Eleven

(165)

CANON PJ-1080A

This printer driver is on your BASIC 8.0 disk with the name P.HC-CANNON. This is not the default driver built into the BASIC 8.0 system, so it is necessary to install it with the Command BLOAD "P.HC-CANNON",B0,P26064.

The CANNON PJ1080A is a special printer because it is a Color inkjet printer. It allows the BASIC 8.0 user to generate a Reasonable quality color hardcopy at a minimal cost. It supports Only 8 colors, while the C128 80 column screen has 16 colors. However, the C128's 16 colors are really only 8 colors in two Intensities, so the two shades will print out as one color. For Example, red and light red both print as red.

The secondary address we used for the CARDCO G and G-WIZ Interfaces is 5. Other printer interfaces may require a Different secondary address.

This driver can only use the range of heights (1-4) in the height parameter of @HCOPY, but only 1 density is supported by the CANNON PJ1080A. For a 640 x 200 pixel screen a height of 2 with a density of 1 gives a very good proportional hardcopy of the screen. Or use a height of 1 with a density of 1 and your 640 x 200 image prints just about the size of a large business envelope. The CANNON PJ1080A can print images upto 640 horizontal pixels wide.

The CANNON PJ1080A driver does not support the rotation Parameter. Images greater than 640 cannot be printed.

This is the source code (written with the Commodore C128

Assembler Development Package). If you have a near CANNON PJ1080A compatible printer that doesn't work quite right, it may be possible for you (if you are a machine language programmer) to

Chapter Eleven

(166)

convert it to a new driver for your printer.

This source code is supplied as a courtesy to all registered BASIC 8.0 owners.

CANNON PJ1080A BASIC 8.0 SOURCE CODE

```
.PAGE
;NAME      HARD COPY
;CREATED   12/09/86
;UPDATED
;AUTHOR    DAVID DARUS
;ASSEMBLER HCD65
;COMPUTER  C-128
;REMARKS   BASIC 8 HARD COPY DRIVER FOR CANNON PJ-1080A
;                  640 * Y UNROTATED  COLOR INK JET
*=$65D0

INCWRD .MACRO ?IARG1
       INC ?IARG1
       BNE 255$
       INC ?IARG1+1
255$
 .ENDM

DECWRD .MACRO ?DARG1
       LDA ?DARG1
```

```
        BNE 254$
        DEC ?DARG1+1
254$
        DEC ?DARG1
 .ENDM

MMU = $FF00

CHROUT = $FFD2
CLRCHN = $FFCC
CLOSE  = $FFC3
SETLFS = $FFBA
OPEN   = $FFC0
CHKOUT = $FFC9

CLCADR     = $26A8
SWAPNYBL   = $25DB
GETPIXSTAT = $2652
GETCOLSTAT = $266E

CURCOLOR = $1318
SCRCSIZE = $131A
SCRSIZEY = $131C
SCRSIZEX = $1324
XO       = $136B
YO       = $136D
MASK     = $137B
PARM1    = $13EF
PARM2    = $13F0
Chapter Eleven

(167)
```

```
PARM3    = $13F1
PARM4    = $13F2

MLHCOPY
  LDA PARM1
  STA SECADR
  LDA PARM2
  STA ESCDAT+2        ;HEIGHT

  LDA MMU
  AND #$FE
  STA MMU
  JSR OPNCHN
  LDA #7
  STA BN
  LDA #0
  STA BV              ;BYTE VALUE TO CHROUT = 0
  STA Y0              ;Y0=0
  STA Y0+1
NXTY
  LDA #0
  STA RGBMODE
  JSR PRTESC  ;DO FOR EACH ROW = 8*HEIGHT
6$
  LDA #0
  STA X0
  STA X0+1
2$
  JSR GETPIX   ;GET PIXEL VALUE AT X0,Y0  IF X0>SCRSIZEX AND <880 THEN CLC
  ROL BV       ;ROLL CARRY INTO BYTE VALUE 0 OR 1
  DEC BN       ;BIT#=BIT#-1
  BPL 1$
```

```
  LDA BV       ;IF BN<0 THEN PRINT BN;BN=7
  JSR CHROUT
  LDA #7
  STA BN
  LDA #0
  STA BV
1$
  INCWRD X0
  LDA X0
  CMP #<640     ;X0<640 THEN GOTO 2$
  BNE 2$
  INC RGBMODE
  LDA RGBMODE
  CMP #3
  BNE 6$        ;REPEAT SCANLINE TO GET 3 (RGB) COLOR ELEMENTS
 .LOCAL
  INCWRD Y0
  LDA Y0
  CMP SCRSIZEY
  BNE NXTY      ;IF Y0< SCRSIZEY THEN GOTO NXTY  DO NEXT SCANLINE
  LDA Y0+1
  CMP SCRSIZEY+1
  BNE NXTY
```

Chapter Eleven

```
(168)
HCERR
  JSR CLRCHN
  LDA #4
  JMP CLOSE

OPNCHN
```

```
  LDA #0
  STA $B7
  LDX #4
  TXA
  LDY SECADR
  JSR SETLFS
  JSR OPEN
  LDX #4
  JMP CHKOUT

PRTESC
  LDA #0
1$
  LDA ESCDAT,X
  JSR CHROUT
  INX
  CPX #4
  BNE 1$
  RTS

GETPIX
  JSR CLCADR        ;X,Y TO ADDR
  LDA SCRCSIZE      ;COLOR SCREEN
  BEQ 1$            ;BR IF MONO
  JSR GETCOLSTAT    ;GET BC/FC
  JSR SWAPNYBL      ;SWAP TO FC/BC
  JMP 2$
1$
  LDA CURCOLOR      ;GET CURRENT FC/BC
2$
  STA HCOPYCOLR     ;SAVE COLOR NYBBLES
  JSR GETPIXSTAT    ;GET PIXEL ON/OFF STAT
```

```
  AND MASK
  BEQ 3$           ;BR IF OFF
  LDA HCOPYCOLOR   ;USE FC BECAUSE PIXEL IS ON
  LSR A            ;PUT ON 0-15 SCALE TO BE USED AS AN INDEX
  LSR A
  LSR A
  LSR A
  JMP 4$
3$
  LDA HCOPYCOLR
  AND #%00001111  ;USE BG
4$
  TAX
  LDA RGBMODE     ;0=RED  1=GREEN  2=BLUE
  BEQ 5$
  CMP #1
  BEQ 6$
  LDA BLUETAB,X
  BEQ 10$
  BNE 11$
```



```
(169)
6$
  LDA GREENTAB,X
  BEQ 10$
  BNE 11$
5$
  LDA REDTAB,X
  BEQ 10$
11$
  SEC
```

```
  RTS
10$
  CLC
  RTS

REDATB
 .BYT 0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1

GREENTAB
 .BYT 0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1

BLUETAB
 .BYT 0,0,1,1,0,0,1,1,0,0,1,1,0,0,1,1


ESCDAT
 .BYT 27,114,0,80

SECADR
 .BYT 0
BN
 .BYT 0
BV
 .BYT 0
RGBMODE
 .BYT 0
HCOPYCOLR
 .BYT 0

 .END
```

Chapter Eleven

(170)

PRINTER	DRIVER	SEC ADR*	SIZE	DENSITY	ORIENTATION	MAX X
EPSON FX-80	P.HC-EPSON	5	1-4	1-7	B	1920
PANASONIC	P.HC-EPSON	5	1-4	1-7	B	1920
STAR NX-10	P.HC-EPSON	5	1-4	1-7	B	1920
OLIVETTI	P.HC-OLIVETTI	5	1-4	1	B	880
STAR NX-10C	P.HC-NX-10C	8	1-4	1-2	B	1280
SPIRIT 80	P.HC-SPIRIT80	5	1-4	1-2	B	1280
GEMINI II	P.HC-GEMINI2	8	1-2	1-2	R	480
MPS 801	P.HC-MPS801	8	1-2	1	R	480
SEIKOSHA	P.HC-MPS801	0	1-2	1	R	480
GEMINI 10X	P.HC-MPS801	8	1-2	1	R	480
CANNON PJ	P.HC-CANNON	5	1-4	1	U	640

SYNTAX OF @HCOPY

@HCOPY,SEC ADR,SIZE,DENSITY,ROTATION

* Where a printer interface is required, these secondary addresses represent the values required with the CARDCO B, G+ and G-WIZ interface. Other interfaces may require different secondary address values.

(171)

PRINTER DRIVER PSEUDO CODE

The following program code segments represent pseudo-code, which means it is not any specific computer language. Rather, it is meant to demonstrate programming techniques and logic. These pseudo-code segments describe the logic required for various printer drivers, and they are here to make it easier to understand the supplied assembly language source code.

There are several examples, some of which are appropriate for several printers.

```
OLIVETTI UNROTATED
  FOR Y=0 TO SCRSIZEY
     FOR HTMULT=0 TO HGHT
        FOR X=0 TO SCRSIZEX
           GETPIXEL(X,Y)
           BUILDBYTE        ;RIGHT TO LEFT
           BIT#=BIT#+1
           IF BIT#=8 THEN PRINT BYTE:BIT#=0
        NEXT X
     NEXT HTMULT
  NEXT Y

OLIVETTI ROTATED
  FOR X=0 TO SCRSIZEX
     FOR Y=SCRSIZEY TO 0 STEP -1
        FOR HTMULT=0 TO HGCT
           GETPIXEL(X,Y)
           BUILDBYTE        ;RIGHT TO LEFT
           BIT#=BIT#+1
           IF BIT#=8 THEN PRINT BYTE:BIT#=0
        NEXT HTMULT
```

```
    NEXT Y
  NEXT X

EPSON,PANASONIC,SPIRIT80,NX-10,NX-10C UNROTATED
  INCRM=2^(HEIGHT-1)
  FOR Y=0 TO SCRSIZEY STEP INCRM
    FOR X=0 TO SCRSIZEX
      FOR TEMPY=Y TO Y+INCRM
        FOR HT=0 TO HEIGHT
          GETPIXEL(X,Y)
          BUILDBYTE        ;RIGHT TO LEFT
          BIT#=BIT#+1
          IF BIT#=8 THEN PRINT BYTE:BIT#=0
        NEXT HT
      NEXT TEMPY
    NEXT X
  NEXT Y
```

Chapter Eleven

(172)

```
EPSON,PANASONIC,SPIRIT80,NX-10,NX-10C ROTATED
  INCRM=2^(HEIGHT-1)
  FOR X=0 TO SCRSIZEX STEP INCRM
    FOR Y=SCRSIZEY TO 0 STEP -1
      FOR HT=0 TO HEIGHT
        FOR TEMPX=X TO X+INCRM
          GETPIXEL(X,Y)
          BUILDBYTE        ;RIGHT TO LEFT
          BIT#=BIT#+1
          IF BIT#=8 THEN PRINT BYTE:BIT#=0
        NEXT TEMPX
```

```
        NEXT HT
     NEXT Y
  NEXT X

GEMINI 2,MPS 801,SEIKOSHA 1000 ROTATED
  BIT#=6
  FOR X=0 TO SCRSIZEX
     FOR Y=SCRSIZEY TO 0 STEP -1
        FOR HT=0 TO HEIGHT
           FOR TEMPX=X TO X+7
              GETPIXEL(X,Y)
              BIT#=BIT#-1
              BUILDBYTE        ;LEFT TO RIGHT
              IF BIT#=8 THEN PRINT BYTE:BIT#=6
           NEXT TEMPX
        NEXT HT
     NEXT Y
  NEXT X

CANNON PJ-1080A UNROTATED
  FOR Y=0 TO SCRSIZEY
     FOR RGB=0 TO 2
        FOR X=0 TO 639
           GETPIXEL(X,Y)
           BIT#=BIT#+1
           BUILDBYTE        ;RIGHT TO LEFT
           IF BIT#=7 THEN PRINT BYTE:BIT#=0
        NEXT X
     NEXT RGB
  NEXT Y
```

(173)

APPENDIX A

The 8563 Video Chip

RAM Expansion

The 8563 Video Chip in the C128 has it's own ram bank that it uses for displaying graphics. In the standard C128 there is 16K of ram. This ram is in the form of two chips on the motherboard of the C128, inside the small metal box that contains the C128's two video chips (the 8563 and VIC II). These ram chips are either 4416 or 4164 rams. This means that each chip contains 8192 bytes of ram (for a total of 16384 bytes). As you have read in many places in this manual, you can use this 16K for some interesting graphic displays, but we constantly mention the option of having 64K of video ram installed (the new C128D comes from the factory with this extra ram). Having the full 64K of ram offers a great enhancement in your C128's graphics, as is obvious throughout the BASIC 8.0 manual. What do you need to get this extra ram and how should you have it installed?

To get the full 64K, you need to have the 4416's (or 4164's) replaced with a different ram chip, the 4464. This 4464 contains 32 Kbytes of ram per chip, so by substituting two of these for the two 16 Kbyte ram chips increases your available 80 column graphic ram to 64K. These 4464 rams can be found in ads for computer chips in magazines like BYTE, COMPUTER SHOPPER and many other magazines. You can also find them (or have them ordered) by your local electronic parts store.

One problem in removing the old 4416 (or 4164) chips is the fact the chips are not socketed. They are soldered to the board, and the pins are crimped (bent) on the bottom. Removing them is a tedious job that requires a skilled technician, and is not something you should attempt yourself. It is very easy to damage

APPENDIX A

(174)

the board and ruin your C128, not something you want to do. If you decide to upgrade to the 64K video ram, we strongly suggest you contact your local CBM service center and ask them to install the 4464 ram chips. If you are not skilled at this type of work you can count on some serious harm coming to the computer. It is that easy to damage!

Also, have the service center install sockets before they put in the 4464 chips. This makes it easy to replace the chips in the future (if you ever need to).

If you would like more technical information on replacing your 8563 ram chips, there have been articles published in both the Computer Shopper and Transactor magazines that detail their installation.

A final warning. Do not attempt to change these ram chips yourself! Your most likely result will he serious damage to the C128. Have the 4464 rams installed at a reputable commodore Service Center.

APPENDIX A

(175)

APPENDIX B

8563 VIDEO RAM MEMORY MAPS

MODE's 0-3

The MODE command configures the available bitmap ram for the 8563 into 8 different screens per mode. In MODE 0 each screen represents the one screen available in the 16K ram area. In MODE's 1-2 there are some combinations that allow multiple concurrent screens. MODE 3, while a 64K graphics ram mode, does not allow more than one screen to exist at the same time because each screen uses the entire 64K ram

available.

The following memory maps show the memory usage of each screen and mode. You can create screens like these by using the SCRDEF command. For more information, see Chapter 10, the BASIC 8.0 COMMAND ENCYLOPEDIA for information on the MODE and SCRDEF commands.

MODE 0

Screen #	Type	Width	Height	Bitmap Address	Color Address
0	M	640	200	0-15999	0
1	C 8x16	640	192	0-15359	15360
2	C 8x8	640	176	0-14079	14080
3	C 8x4	640	152	0-12159	12160
4	C 8x2	640	104	0-8319	8320
5	C 8x8 Lace	640	176	0-14079	14080
6	C 8x4 Lace	640	152	0-12159	12160
7	C 8x2 Lace	640	104	0-8319	8320

MODE 1

Screen #	Type	Width	Height	Bitmap Address	Color Address
0	M	640	200	0-15999	0
1	C 8x8	640	200	16000-31999	32000
2	C 8x2	640	200	34000-49999	50000
3	M	640	300	34000-33999	34000
4	C 8x8	640	200	0-15999	16000
5	C 8x8	640	200	18000-33999	34000
6	C 8x8	640	200	36000-51999	52000
7	C 8x8	640	728	0-58239	58240

APPENDIX B

(176)

MODE 2

Screen #	Type	Width	Height	Bitmap Address	Color Address
0	M	640	200	0-15999	0
1	M	640	200	16000-31999	0
2	M	640	200	32000-47999	0
3	M	640	200	48000-63999	0
4	C 8x2	640	200	0-15999	16000
5	C 8x2	640	200	24000-39999	40000
6	C 8x4	640	200	24000-39999	40000
7	C 8x4	640	200	44000-59999	60000

MODE 3

Screen #	Type	Width	Height	Bitmap Address	Color Address
0	M	1280	409	0-65439	0
1	M	640	819	0-65519	0
2	M	2040	252	0-64259	0
3	M	800	655	0-65499	0
4	C 8x2	640	546	0-43679	43680
5	C 8x4	640	655	0-52399	52400
6	C 8x8	640	728	0-58239	58240
7	C 8x16	640	768	0-61439	61440

APPENDIX B

(177)

Appendix C

BASIC 8.0 MEMORY MAP

ZERO PAGE LOCATIONS USED:

$9E,9P

$AC-$B5

$FA,FB,FC,FD

```
BUFFER AREA USED: $0F00-$0FFF

SYSTEM CODE: $1300-$6FFF

FREE USER BYTES: $1464-$146B

PRINTER DRIVERS: $65D0-$68CF

SOLID ROUTINES: $68D0 $6FFF


16 POINTER DEFINTIONS: 16 bytes definable by user with pokes

    #0 - $0E00

    #1 - $0E10

    #2 - $0E20

    #3 - $0E30

    #4 - $0E40

    #5 - $0E50

    #6 - $0E60

    #7 - $0E70

    #8 - $0E80

    #9 - $0E90

   #10 - $0EA0

   #11 - $0EB0

   #12 - $0EC0

   #13 - $0ED0

   #14 - $0EE0

   #15 - $0EF0


Pointers consist of a 16 pixel by 8 pixel matrix. For example:


     1111111100000000   255,0

     1111100000000000   248,0

     1100110000000000   204,0

     1100011000000000   198,0

     0000001100000000     3,0

     0000000110000000     1,128
```

0000000011000000 0,192

0000000001100000 0,96

However, by using the optional last parameter in the @PTR command (see Chapter Ten) you can change the height used in the pointer. The number is definable from 1 to 16, with the default

APPENDIX C

(178)

on powerup as 8. If you were to change the default by using the value 16, all subsequent uses of the @PTR command would use 16 scanlines to define the pointer, until a different height value is used.

If you wish, you can use the larger height values to create bigger, sprite-like objects. In the case of a height of 16, each of the new bigger pointers would consist of pairs of subsequent pointers. In this case, a pointer definition of 0 would use pointer def's 0 and 1, a definition of 2 would use pointer def's 2 and 3, etc.

If you create sets of pointer definitions you want to save And reuse, remember that the pointer definitions are stored in The same place as the 40 column sprite definitions. So you can Save and load them with the Basic 7.0 commands BLOAD and BSAVE.

BLOAD "PTRS.B8",B0,P3584

BSAVE "PTRS.B8",B0,P3584 TO P3839

One final note. When programming with datafiles or devices, It is important that you are aware of the file numbers used by BASIC 8.0.

When performing a hardcopy with @HCOPY, BASIC 8.0 uses file number 4. And when performing disk Input/Output it uses file

number 2. You should not use these filenumbers in your programs when using these BASIC 8.0 commands.

APPENDIX C

(179) APPENDIX D

File Formats & Naming Conventions

BASIC 8.0 has four specific file types (brushes, patterns, fonts and logo). Each of these has a specific defined structure, in some ways like the Amiga PC's IFF system. This section documents these four formats, as well as suggesting a structure for 3D graphic data files. We also make suggestions on filenames to be used for each format, in order to make different applications written by different programmers recognize and use each others datafiles.

NAMING CONVENTIONS:

BRUSH - A BRUSH is really any piece of the bitmap screen. It can be as small as a single byte, or as large as the entire screen. We suggest the name BRUSH be used for sections smaller than a screen, in order to identify them as different in size from pictures.

Start each brush with the letters BRUS. For example;

BRUS.DISK

BRUS.CHART

BRUS.OBJECT8X2 {Indicates it is in 8x2 color format)

The structure of a BRUSH file is:

Load Address (2 bytes) Lo/Hi

FILE IDENTIFIER (4 bytes) BRUS

FILE TYPE (1 byte) 4

Column # (1 byte)

Scanline (2 bytes) Lo/Hi

Compression Flag (1 byte)

ColorSize (1 byte)

DX Columns (1 byte)

DY (2 bytes) Lo/Hi

BC (background color 1 byte)

FC (foreground color 1 byte)

OC (outline color 1 byte)

Brush data ...

PICTURE - A PICTURE is really a brush as large as the entire screen. We suggest the name PICTURE be used for brushes that are full screen, in order to identify them as different in size from brushes.

Start each picture with the letters PICT. For example;

PICT.SCENE

PICT.GRAPH

PICT.PLOT8X4 (Indicates it is in 8X4 color format)

The structure of a PICTURE file is exactly the same as a BRUSH.

PATTERNS

Start each pattern with the letters PATR. For example;

PATR.BRICKWALL

PATR.DIAGONALSx8 (Indicates it is in 8x8 color format)

APPENDIX D

(180)

PATR.HAPPYFACE

The structure of a PATR datafile is:

LOAD ADDRESS (2 bytes) Lo/Hi

FILE IDENTIFIER (4 bytes) PATR

FILE TYPE (1 byte) 1

COLUMN DX (1 byte)

SCANLINE DY (1 byte)

COLOR DZ {1 byte)

COLOR DY CELLS (1 byte)

Pattern data ...

FONTS

Start each font with the letters FONT. For example;

FONT.Roman

FONT.160Column

FONT.Technical

The structure of a FONT datafile is:

Load Address (2 bytes) Lo/Hi

FILE IDENTIFIER (4 bytes) FONT

FILE TYPE (1 byte) 3

FONT DEFINITION COLUNNS WIDE (1 byte)

FONT DEFINITION SCANLINES DEEP (1 byte)

FONT data ...

LOGO

Start each logo with the letters LOGO. For example;

LOGO.MENU1

LOGO.REQUESTOR

LOGO.TITLEPAGE

The structure of a LOGO datafile is:

LOAD ADDRESS (2 bytes) Lo/Hi

FILE IDENTIFIER (4 bytes) LOGO

FILE TYPE (1 byte) 2

FLAG (1 byte, 0 indicates end of data)

FONT STRUCT# (1 byte)

COLUMN (1 byte)

ROW LO (1 byte)

ROW HI (1 byte)

HEIGHT (1 byte)

WIDTH (1 byte)

DIRECTION (1 byte)

STRING LENGTH (1 byte)

STRING ... (string length)

APPENDIX D

(181)

3D DATAFILES

In addition to these system file formats, it is likely users will generate additional datafiles to store 3D line drawings. We suggest two formats which should cover most programmers needs, and if the filename convention is used will facilitate the use by others of user generated 3D objects.

Format one (3DF1.) concerns storing discrete lines as the six points used to define the line. The file consists of one word that identifies the total number of 3D lines (N) to follow. Then follow six*N points of data. This format is easy to read and write, but has the drawback of requiring a larger amount of file (and memory) space.

3DP1. An example of writing format 1

```
 0 read n
10 dim x1(n), x2(n), y1(n), y2(n), z1(n), z2(n)
20 rem read n 3D lines from data statements in program
```

```
 30 for i=1 to n
 40 read x1(i), y1(i), z1(i), x2(i), y2(i), z2(i)
 50 next i
 60 rem open seq disk file and write data as format 3DF1.
 70 open 8,8,8,"3DFl.object,s,w"
 80 print#8,"3DF1" : REM WRITE THE FILETYPE AS FIRST WORD
 90 print#8,str$(n)
100 for i=1 to n
110 x1$=str$(x1(i)):print#8,x1$
120 y1$=str$(y1(i)):print#8,y1$
130 z1$=str$(z1(i)):print#8,z1$
140 x2$=str$(x2(i)):print#8,x2$
150 y2$=str$(y2(i)):print#8,y2$
160 z2$=str$(z2(i)):print#8,z2$
170 next I
180 close 8
190 rem this file defines n lines that can be drawn in
200 rem the BASIC 8.0 format LINE,X1,Y1,Z1,X2,Y2,Z2,1
```

Format 2 (3DF2.) is somewhat more complex, but can store more information in a smaller file. It consists of a system that stores information as groups of connected points. The file begins with a number (NG) that specifies the number of groups. Each group begins with a number (NP) that indicates the number of

points in that group. These points are connected, with point 1 starting the object, point 2 as the endpoint of point 1, point 3 as the endpoint of point 2, and continuing with point NP as the endpoint of point NP-1.

Here is an example of reading a 3DF2 datafile.

```
 10 open 8,8,8,"3DF2.datafile,s,r"
 20 input#8,ft$:ree filetype 3df2
 30 input#8,ng$:ng = val(ng$)
 40 for i=1 to ng
 50 input#8,np$:np = val(np$)
 60 rem get first point in this group
 70 input#8,x1$:x1 = val(x1$)
 80 input#8,y1$:y1 = val(y1$)
 90 input#8,z1$:z1 = val(z1$)
100 for j=2 to np
110 input#8,x2$:x2 = val(x2$)
120 input#8,y2$:y2 = val(y2$)
130 input#8,z2$:z2 = val(z2$)
140 rem draw line
150 @line,x1,y1,z1,x2,y2,z2,1
160 x1=x2:y1=y2:z1=z2
170 next j
180 next I
190 close 8
200 end: REM OF FILEREADER
```

EXAMPLE OP 3DF1 PILE:

```
3DF1
4
0,0,0,10,10,10
100, 50, -10, 37, 199, 100
639,199,199,639, 199,-199
0,100,0,0,100,100
```

APPENDIX D

(183)

EXAMPLE OF #DF2 FILE:

```
3DF2
3
4
0,0,0
100,10,100
0,0,100
0,0,-100
3
25,25,-25
50,25,-25
50,50,-25
6
0,0,0
100,0,0
100,100,0
100,100,100
100,100,-100
100,100,200
```

APPENDIX D

(184)

A Standard Basic 8.0 Program

This is a simple program that introduces you to some of the basics of programming in BASIC 8.0. It demonstrates the proper way to start and end a BASIC 8.0 program, and the best way to handle error trapping in your C128 BASIC 8.0 programs.

```
   10 TRAP 1000:REM Get out if you have trouble
   20 @WALRUS,0:REM Set up for 16K mode
   30 @MODE,0:REM Use the 16K MODE screens
   40 @SCREEN,2:REM 8x8 640x176 Color screen
   50 @CLEAR,0,2,0
   60 @COLOR,2,4,0
   70 @DRWMODA,1,0,0,0,0,0,0
   80 @DRWMODB,0,0,0
   90 @ANGLE,0,0,0,0:@ORIGIN,320, 100,100,320,100,200
  100 REM YOUR PROGRAM BEGINS HERE
      .
      .
      .
 1000 @DRWMODA,1,0,0,0,0,0,0:REM RESTORE DRAWMODES TO DEFAULT
 1010 @DRWMODB,0,0,0
 1020 @ANQLE,0,0,0,0:REM SET DRAWING ANGLES TO 0
 1030 @ORIGIN,320,100,100,320,100,200:REM RESET ORIGINS
 1040 @TEXT:END
 1050 END
```

This program demonstrates the proper way to begin and end a BASIC 8.0 program. It begins with the TRAP command. TRAP instructs the computer what to do if an error occurs, for example syntax error or a press of the STOP key. In this case it goes to line 1000 and re-initializes the DRWMOD's, ANGLE and ORIGIN to the defaults, and returns the C128 to TEXT mode. This is a necessity in EVERY BASIC 8.0 program, as you will most likely have an error occur when you are in BITMAP mode. Without the TRAP command, you could get stuck and have to reset the C128 and re-boot BASIC 8.0 in order to regain command of the keyboard.

APPENDIX D

(185)

Appendix E

Data Compression

BASIC 8.0 offers as an option in it's storage of brushes and pictures the ability to compress the data into a smaller, more compact date structure. This data compression can be in memory as a structure or on disk as a file. This section offers some information for programmers who may need to be able to compress and uncompress data in the BASIC 8.0 format.

Data compression is the process of compacting a set of data into a smaller set of data; yet still retaining the same meaning. This means that a set of data once uncompressed will be identical to the original data set. Besides requiring less disk space or ram to store the data, it also allows faster input/output then the original size data set.

The algorithm used to compress brushes in the BASIC 8.0 system is a toggled run-length, meaning that if a value is repeated more than the break even point the value is compressed to a run count and run value. If a run is below the break even point then the values are retained as they are. The toggle aspect of the algorithm stems from the fact the data can toggle in and out of compressed format encoding within a single data file.

The details of the compression algorithm are:

Compressed format:

byte 1 - bit 7 = 1

bit 0-6 = count (4-127) of the number of times to repeat the following byte value when uncompressing data.

4 is the break even point. 4 or more repeating values can be compressed.

```
     byte 2 - bitmap data value (0-255)
APPENDIX E


(186)
Uncompressed format:
  byte 1 - bit 7 = 0 bit 0-6 = count n (1-127) number of uncompressed
                               unique data values to follow.
byte 2-n uncompressed data values


  1 rem A sample uncompression program in Basic
  5 dim a$(11)
 10 open 2,8,2,"datafile,p,r"
 20 get#2,a$,b$:rem get load address
 30 for t=1 to 4:get#2,a$:next:rem BRUS header
 40 get#2,a$:rem file type
 50 for t=1 to 11:get#2,a$(t):next:rem get header info
 60 if a$(i)="1" then 120
 70 rem data not compressed
 8O get#2,a$:s=st
 90 a=asc(a$+chr$(0)):rem store A somewhere
100 if s=0 then 80
110 goto 200
120 rem compressed data
130 get#2,a$:s=st:a=asc(a$+chr$(0))
140 if (a and 127) = 0 then 210
150 get#2,a$:s=st:rem run value
160 for t=1 to (a or 127):rem run count
170 rem store asc(a$+chr$(0)) somewhere
180 next
190 if s=0 then 130
200 closa 2:end
210 rem uncompressed toggle
```

```
220 for t=1 to (a or 127)
230 get#2,a$:s=st
240 rem store asc(a$+chz$(0)) somewhere
250 it s<>o then t=(a or 127):goto 200
260 next t
270 goto 190
```

APPENDIX E

(187)

Appendix F

STANDARD STRUCTURE # USES

Suggested standard locations for data structures.

STRUCTURE#	USE
0	FONT.CURSIVE
1	FONT.COMPUTER
2	FONT.ROMAN
3	FONT.GOTHIC
4	FONT.FANCY
5	FONT.THIN
6	FONT.TECH
7	FONT.SCRIPT
8	FONT.CURSIVE
9	FONT.160 COLVMN
10-19	User supplied FONT's
20-29	LOGO.
30-39	PATR.
40-99	BRUS.

These are suggestions only, you may vary them as your programming requirements dictate.

APPENDIX F

(188)

The Players

WALRUSOFT Inc. - Company started and owned by Louis R. Wallace and
David P. Darus (with a little help from our friends)

David P. Darus - President, Senior Programmer, Designer for Walrusoft
Writer and Consultant
Computer Science Graduate, University of Florida 1984
Fleet Analyst, Gainesville FL Regional Utilities

Darius R. Wallace - Secretary/Treasurer, Programmer, Designer and
Documentation Specialist, Writer, Editor
Research Chemist and Computer Specialist,
Veterans Administration Medical Center, Gainesville

Ken French - A good friend who provided great insight and help in
the design and implementation of the printer drivers.

Richard Rylander - Original author of the 3D solids commands for C64

Basic 8 Manual Errata

Basic Calc, Write and Print are on the back side of the disk. To use them, flip

the Basic 8 disk over and reboot your machine.

1

BASIC PRINT

Program and Documentation by: Loren J. Lovhaug

1. Disclaimer and Distribution: BASICC PRINT was written for the enjoyment of C-

128 owners everywhere. You may freely distribute BASIC PRINT and its

subordinate files. You may also freely distributed the BASIC 8 RUN T1MELIBRARY

required for non-BASIC 8 owners to run BASIC PRINT. You may NOT freely

distribute the BASIC 8 editor, BASIC 8 is the copyrighted property of WALRUSOFT.

Please don't destroy software development on the C-128 by stealing software.

Neither the author nor PATECH SOFTWARE shall be responsible or liable for any

defects in this program or its documentation.

2. BASIC PRINT OVERVIEW: Okay now that you got the "rules" concerning BASIC

PRINT, here is how you use it. First I suggest you copy the BASLC PRINT files

onto a disk of their own. The files you need to copy are: B8.BPRINT V1.04,

PICT.BPRINTFRONT, PICT.BPRINTED, CONFIG.BPRINT, and PTRS.ARROWS. I also

recommend that you place a few of your favorite fonts on the disk as well. If

you do not own the BASIC 8 package the disk you place the BASIC PRINT files on

should be a BASIC 8 RTL (Run Time Libriry) disk. You can obtain a BASIC 8 RTL

disk from most Commodore user groups or by downloading the BASIC 8 RTL files

from Quantum Link or Genie. If you are running BASIC PRINT from an RTL disk

simply click on the icon for BASIC PRINT from the workbench. If own the BASIC

PRINT package and prefer running the program after the BASIC S editor is

installed, you can do so by typing RUN"B8.BPRINT V1.04" from BASIC.

Once you have executed BASIC PRINT the screen will turn black and after a short

delay you will see the BASIC PRINT title screen. Towards the bottom of the

title screen you will inside black box the instruction: "Press M to use a mouse

or J for joystick". Press the appropriate letter for the input device you wish

to use. (Note: From this point onward both in this documentation and in the

program itself the word "mouse" will be used when referring to the selected

input device, even though you may use a joystick.) Once you have selected your

input device, you will be asked whether you wish to load the preset fonts

defined in the config.bprint file or whether you would wish to load fonts

individually at a later time. Press L at this prompt will load the fonts

from the config.bprint file, press I to load your fonts individually at a later

time.

The config.bprint file is a simple Commodore ASCII sequential file which tells

BASIC PRINT what printer driver it should use in conjunction with your printer

and what fonts you wish to have loaded as "pre-sets" should using the option

listed above. It may be edited with most C-128 word processors or via a BASIC

program. The first line of the config.bprint file should contain the name of

the printer driver required for your printer followed by a return (ASCII 13 -

denoted below as a <CR>) On subsequent Lines are the names of the fonts that

2

will be loaded if the loading of preset fonts is requested again followed by a

return mark. Here is how the default config.bprint file is set up:

P.HC-EPSON<CR>

FONT.CURSIVE<CR>

FONT.COMPUTER<CR>

FONT.GOTHIC<CR>

FONT.ROMAN<CR>

FONT.SCRIPT<CR>

FONT.STYLE<CR>

FONT.THIN<CR>

You may have noticed the message located on the upper portion of the black box

where the two prompts appeared denoting the amount of video random access memory

(RAM) and system random access memory you have BASIC PRINT available on your C-

128 or C-128D. BASIC PRINT automatically recognizes how much RAM is available

in your system and takes advantage of it accordingly. BASIC PRINT users who

have C-128s equipped with 64K video display RAM or C-128Ds will notice that

screen changes are instant and the overall operation of the program is faster

than those users who have only l6K of video display memory. Likewise owners of

either the 1700 or 1750 System RAM expansion units will be blessed with better performance than those lacking expanded system RAM.

After answering the input device and load fonts prompts the BASIC PRINT Control

Panel screen will be displayed. On the right side of the screen you will see

the BASIC PRINT Control Panel. There are five buttons (options) on the Control

Panel that you may choose by moving the mouse over one of the buttons and

pressing the left mouse button (or joystick fire button) to push (select) the

button (option) of your choice. The five Control Panel buttons are as follows:

OUTSIDE - Edit outside portion of your greeting card.

INSIOE - Editing inside portion of your greeting card.

LOAD BRUSH - Load BASIC 8 graphics for later placement.

LOAD FONT - Loading of BASIC 8 fonts for use when editing text.

PRINT - Outputs the specified creation to the printer.

EXIT - Exits BASIC PRINT, BASIC PRINT still in memory.

Below the Control Panel you will see a bright red box. If you told BASIC PRINT

to load preset fonts from disk when prompted above you will see the names of the

fonts that were specified in the config.bprint file displayed inside of the red

box. These are the fonts currently available with which you will be able to

place text on the inside or outside of your greeting card.

3. OUTSIDE and INSIDE: After selecting the OUTSIDE or INSIDE option you will be

presented with the creating card editor. The greeting card editor has nine

3

options which are outlined below. Each option is accessed by typing its first

letter.

(G)et The(G)et option allows you to retrieve previously saved portions of

your greeting card. If you are working on the outside of your greeting card,

the (G)et command will retrieve previously stored front and back portions of

greeting cards denoted by the pict.bpo prefix . If you are working on the

inside of the greeting card, the (G)et command will retrieve previously stored

inside left and inside right portions of greeting cards denoted by the pict.bpi

prefix. File retrieval from the (G)et command is accomplished by using the

mouse. The files from which you may choose will be presented on a file

selection menu. To select a file simply position the mouse pointer over the

desired selection and click on the left mouse button. Should you wish to view

other choices that may be chosen on the current disk drive position the mouse

pointer over the down arrow on lower right portion of file selection menu and

press the left button, this will scroll the file selection menu down. To scroll

the file selection menu back up, simply position the mouse pointer over the

right button on the upper right portion of the file selection menu and press

the left mouse button. Should you wish to access a file on another disk drive

simply position the mouse pointer over the word UNIT on the right hand side of

the file selection menu and press the left mouse button.

(S)ave The (S)ave option allows you to store portions of your greeting card

for printing or for later editing. You will be asked on the bottom of the

screen to provide a name for the portion of your greeting card you wish to

store. The proper outside or inside prefix is provided for you.

(A)lter The (A)lter option allows you to change the current font, adjust the

font height and width and adjust the line width for the line and box drawing

commands. Altering the above settings is accomplished by placing the mouse

button on the appropriate green "button" in the center of the screen and

pressing the left mouse button. Your new choices will be displayed on the lower

part of the screen. Here are the buttons and their meanings: FT = change font,

HT = change font height, WD = change font width, LW = change line width, X =

Exits the alter settings option.

(T)ext The (T)ext option allows you to place text on your greeting card.

After selecting the (T)ext option you will be asked whether you want to put your

text on the left or the right window on the editor screen. Press L if wish to

place your text on the left side of the screen, press R if you wish to place

your text on the right side of the screen. Once you have selected which side of

the screen you wish to place your text on, you will be asked to position the

pointer with the mouse at the location at which you would like to begin placing

your text. After positioning the mouse pointer to the position where you would

like to begin typing your text, press the mouse button again and you will be

4

able to place text in the current font and size settings on your card. In

addition to the obvious alphanumeric keys you can also use the cursor keys

and the delete keys to help you place your text.

(P)lace The (P)lace brush option allows you to place a picture (loaded with

the load brush option) onto the screen. The picture must be a monochrome or 8 x

8 color cell brush. Be careful, this version of BASIC PRINT does not do any

error checking on the size and color cell make up of the brush.

(L)ine This option allows you to place lines on the screen, Operation of

this command is similar in nature to the (T)ext command except that in addition

to selecting a starting point with the mouse you will be asked to select both a

starting as well as an ending point. Line thickness is determined by the line

width setting.

(B)ox This option allows you to place Boxes on the screen, Operation of

this command is similar in nature to the (L)ine command. You will be asked to

specify the upper left and lower right corners of the box you wish to draw on

the screen with the mouse. Line thickness on the box is determined by the line

width setting.

(E)rase This option allows you to erase areas on the screen, Operation of

this command is similar in nature to the (L)ine command. You will be asked to

specify the upper left and lower right corners of the area you wish to erase on

the screen with the mouse.

(Q)uit The (Q)uit option allows you to exit back to the Control Panel. Before

you are allowed to return to the Control Panel however you will be asked if you

wish to save the current portion of the greeting card you are working on. Type

a Y if you wish to save the current portion to disk. Keep in mind that area

must be saved to disk in order to print it.

4. LOAD BRUSH & LOAD FONT For both of these options you will be presented with a

file selection menu as described in the greeting card editor's (G)et command and

allowed to choose the brush or font you wished to have loaded into memory.

5. PRINT Before you print your greeting card, you will be asked for the

filenames of the outside and inside portions of the greeting card you wish to

print and the settings for your individual printer. In most cases it will

require some experimentation with these settings before the desired results are

achieved.

5

BasicCalc

BasicCalc is an electronic spreadsheet program written in Basic 8.0 for home or small business use. It is loosely based on a C64 basic spreadsheet published in RUN Magazine by Trent Busch, but has been heavily modified and enhanced. Everything from balancing a checkbook to complex investment analysis can be done with a spreadsheet. Besides calculating data, it can be used to represent that data as a bar, pie or line graph. It can also save the screen to disk as a brush or page for use within BasicPaint or BasicWrite, or print the screen to your printer. No matter what your needs, BasicCalc will be a valuable addition to your program library.

This document will take you step by step through the features of BasicCalc. Let us RUN the program and examine the display. Load and run the program from the Workbench or directly from the keyboard. (Basic 8 or the Basic 8 RunTime Library must be installed in memory.) BasicCalc will default from whatever drive you load it from. It starts off with a black screen with the comment "Initializing Variables" displayed. There will be a short pause, then the title screen appears. The program will know how much ram is present both in the system and the video display memory area. BasicCalc will run on a minimal C128, meaning the normal 128 K of ram and 16 K of VDC ram. If you have a 1700 or 1750 ram cartridge, or your C128 has 64K of video ram (as the C128D has) the program will run smoother because more items will be available from memory.

Once the program displays the title screen it will begin to Load its fonts, menus, and help screens. This will take only a few seconds. When it is complete the words BASICCALC will begin to scroll and make a sound. It will continue to scroll until you press a key. Once a key has been pressed the

program starts.

The flashing arrow at the top left of the screen represents the data entry Line. Below that is a solid Line running across the screen. This contains some useful information that is always present on the spreadsheet page. (If you have a 64K VDC C128, the bottom 3 lines of the screen contain additional reminders.) The numbers 0-6 represent columns. The letters A through S are the rows. BasicCalc has thirty columns (0-29) and twenty-six rows (A-Z). Each column can display up to nine characters. Notice that only seven columns are displayed on the screen. All thirty are there, you just cannot see them all at once. Imagine that you are looking through a window and that you can only see a portion of the overall picture. Each cell in the spreadsheet is identified by the letter of the row followed by the column number. For example, the first cell is A0.

The cursor keys allow you to move this window around the spreadsheet. Press 'CURSOR DOWN' and the spreadsheet will be quickly redrawn with rows B through T. Experiment with the cursor keys until you can place the viewing

6

window over all the columns and rows. Pressing 'HOME' will return the window to cell A0.

As mentioned, the intersection of a column and a row is called a cell. There are 780 cells that you can use, A0 through Z29. And there are three types of information that you can enter into a cell, text, numeric, or formulas. In order to enter information into a cell you need to follow a specific procedure.

Type in the cell location, row first and column second. Please leave out any

spaces. Next type a colon. This separates the cell location from the data. Now

you can type in text or numeric data up to nine characters in length.

Here are some examples.

A0:BUDGET 84 (enter a text label in cell A0)

C12:250 (enter the value 250 in cell C12)

D5:-23.12

Text information can contain almost any character on the keyboard, but

must not begin with a number, or a plus or minus sign. Numeric information must

start with a number or a plus or minus sign followed by numbers. Decimal points

are allowed in the numbers.

After typing in the information that you want in a cell, press RETURN.

If everything was typed in correctly you should see the data in the proper cell.

It is very important that you enter information in the proper method. If not,

BasicCalc will complain by displaying an error message on the display line. The

messages are designed to help you locate the problem. When you are ready to

continue press a key and simply retype the line correctly. Text data is

automatically left justified. You can insert spaces to move the text over if you

desire. Numeric data will be right justified.

To clear a cell, simply type the cell coordinates followed by a colon

and then press RETURN. Note that this will not clear a formula. Pressing

SHIFT-CLEAR will clear the entire spreadsheet. For safety reasons this is a two

step process. First press SHIFT-CLEAR and then answer the question on the

comment line. Pressing 'Y' will clear the spreadsheet. Press 'N' to exit the

clear mode.

Up to now all we have done if create neat columns and rows. The real

power of BasicCalc is its ability to compute mathematical equations using the

data in each cell. For example, we can add cell A0 to cell A1 and put the answer

in cell A2. This is accomplished by putting the formula A1+A2 into cell A2. Here

is the proper format.

A2:[F1]A1+A2

The [F1] appears as a reverse 'F' and is printed by pressing the 'F1'

function key (not the brackets in the examples). This key is used to access

7

special features of BasicCalc. If you forget to press 'F1' when entering a

formula, the formula will be entered as text and displayed in the cell.

Normally, a formula should not be displayed in a cell, only the result of the

computation should be present. If you can see the formula, it was entered

incorrectly as a text label. A special command allows you to view the formula in

a particular cell.

A2:[F1]V

If a formula resides in cell A2, it will be printed on the comment

line. The full value of the numeric data in cell A2 will be printed also. This

is important because each column is limited to nine characters. BasicCalc will

fill the cell with asterisks if the numeric data is longer than nine characters.

You will then need to use the view command to examine that cell. These are the

formulas that this spreadsheet can compute.

addition : cell + cell or cell + constant

subtraction : cell - cell or cell - constant

multiplication : cell * cell or cell * constant

division : cell / cell or cell / constant

exponentiation : cell ^ cell or cell ^ constant

BasicCalc cannot handle complex formulas involving more than two cells at once. A more involved computation can be done by storing the intermediate answer in a spare cell. When typing in a formula please leave out all spaces and you must enter a cell first and the constant second. If your needs require complex formulas, spread them out. For example, if you need the to divide the sum of two cells by another cell, it is a two step process.

A2:[F1]A0+A1

A4:[F1]A2/A3

This adds A0 and A1, then divides that value by the contents of A3, with the results in A4. Formulas are not calculated immediately. In order to calculate the spreadsheet you must press the left arrow key (<-). (This is the top left key on the keyboard under the ESC key.) Calculating takes a while depending upon the number of formulas in the spreadsheet. Normally you will make all your changes and calculate just once. Calculations are done column by column from top to bottom. Column one will be completely done before column two. This is an important point to remember, and if you forget it your calculations may come out incorrect!

For example, let cell A0=F9*G6. If cell F9 has a formula in it, the

resulting answer will be figured after cell A0 is computed. To overcome this,

8

you should press the left-arrow key twice (calculate twice). After all computations are complete, the spreadsheet will be redrawn with the results displayed in the proper cells. Attempts to divide by zero will be noted in that cell and if a exponentiation calculation is too large an overflow note will be displayed in that cell. BasicCalc has a few extra special formulas which are very useful.

A1:[F1]SUMA2-Z2 This command puts the sum of cells A2 through Z2 into cell A1. Very useful for adding up long columns or rows.

Z29:[F1]AVGB3-B12 This command calculates the average of cell B3 through B12 and puts the answer into cell Z29

C12:[F1]MIND0-G0 This command looks for the minumum figure over a range of cells and puts the answer in cell C12

F5:[F1]MAXZ0-Z29 This is similar to the MIN command except it returns the maximum value in a range.

You can use any cells that you wish but they must be either in a straight column or row. Here is an example of an incorrect format.

Z29:[F1]SUMA0-D29 This formula will not work because cells A0 through D29 are in a diagonal line, not a straight line.

Here are the rest of 8asicCalc's special features.

C15:[F1]C This command will clear an individual cell including the formula, text, and numeric data.

F25:[F1]J This command jumps the display to a particular area of the spreadsheet. Sometimes this is faster than using the cursor keys to move the display window.

D3:[F1]COPD4-D29 This command is used when you are entering lots of identical information. In this example the contents of cell D3 will be copied into cells D4 through D29. Only text or numeric information will be copied. Formulas must be typed individually. This works with rows or columns.

FUNCTION KEYS

Press F6 and you will see a maximum precision display on the comment line. This command does not affect the accuracy of the calculations. It rounds

the number for display purposes. Use the View command to see the full value.

Press a number from zero to six. Zero means integers and six means six decimal places. BasicCalc is automatically set up for two decimal places automatically.

This command is only for numbers that are computed by a formula. If you want two place decimals on all the numbers you must type them that way.

Press F4 and follow the screen directions to save the spreadsheet template to disk. Pick out a logical filename for the template. It will have the prefix SPRD. added to it.

Press F2 and follow the screen directions to load the spreadsheet

template from tape or disk. Or you can use the directory menu (F8) and choose

load a spreadsheet. This will show you all the spreadsheets on the current

disk. To load one just use the cursor keys to highlite your selection and press

return.

To print the spreadsheet template on paper, press F5 and follow the

screen directions. You can print the whole template or any portion of it. You

will need to know the top-left cell coordinates and the bottom-right cell

coordinates of the area that you want printed out. If you specify more than

seven columns, BasicCalc will automatically break the printout into sections for

you. To print just the formulas on paper, press F3 and follow the screen

directions.

GRAPHS

F7 puts you in the BasicCalc graph screen. Here you can generate

simple bar, pie and line graphs from your data. The data must be in a line, in

other words the same row or column.

Bar graphs are three dimensional. Their size depends on the number of

points you are graphing. The height is set so that the largest data item nearly

reaches the top of the graph area. The value of each bar is printed above it.

The pie chart creates a segmented pie graph, with different patterns for

each item. The largest data item is pulled out from the main pie. Each pie

wedge has its value printed near it.

The line graph is a form of non-linear regression, or best fit curve.

It generates a curved line that attempts to show the relationship of the data.

This graph is not that useful for most applications. There are no labels in

this graph.

Once your graph is completed, a new set of cursor driven commands appear. These allow you to either return to the spreadsheet, graph again, or save the screen to disk (snapshot) as a picture which can be used in programs like BasicWrite or BasicPaint. You can also print the screen on your printer.

PRINTERS

10

Before attempting to print any screens you should make sure you have installed the correct printer driver. To do that, from the main spreadsheet press F8. This takes you to the directory menu. Here you can look at a directory of all the files on the current disk, all the spreadsheets available, or all the printer drivers on the disk. When you first start up BasicCalc, it is a good idea to install the proper printer driver. So press F8, select P for printers, and use the cursor keys to select the proper driver. It will be loaded into memory.

After installing the printer driver, you should set the proper secondary address for your printer and its interface, as well as the height and density of the output. You can even set the printed picture to be rotated if your printer driver supports rotation.

To do this you must use the extended commands. These are entered from the spreadsheet command line and are prefixed with the asterisk (*). Here is a list of the extended commands.

*dir :gives the directory menu, just like pressing F8

*quit :exits the spreadsheet program and returns to basic

*drive= :changes default drive and accepts drive numbers 8-11

*settime= :set the C128 clock...enter in 24 hour format HHMMSS

*clockon :display the time (it is displayed in 12 hour format)

*clockoff :turn off the clock display

*graph :go to graphing screen...same as F7

*snapshot :save spreadsheet screen to disk as a picture

*print :print spreadsheet screen to the printer

*secadd= :set secondary address required for graphics on your printer. Values are 0-9.

*height= :set height of printout. Max height varies from printer to printer. Values are from 1-4

*density= :set printer density for printout. Max density varies from printer to printer. Values from 1-7

*rot= :set printout rotation flag. Values 0-1. Some printers (MPS 801 is an example) will always rotate the printed image.

Finally, you can get help from the spreadsheet display by pressing the HELP key. This displays a help screen with most of the commands present. You exit this screen by pressing any key.

11

<<the following page has been reformatted from the original manual>>

Basic Calc Help

Basic Calc is a spreadsheet of 780 cells (26 rows by 38 columns).

The rows are a-z tha columns 0-29.

Labels, data and commands are entered at the command line.

Extended commands can print screen, set SecAdd, Height, Density or Rotation of HCopy), the time (in 24 hour format HHMMSS), the default drive, save screen as a brush (for use in BasicWrite or BasicPaint), The Dir command offers a menu of directories, You can load a spreadsheet from here, install a printer driver, or just look at the contents of the disk.

Operators	Function Keys	Extended Cmds	Function Cmds
+,-,*,/,^ (Exp)	F1 Formula Key	*dir *quit	cell:[F1]j jump
Exit, *quit	F2 Load Sheet	*graph	cell:[F1]v view
Functions	F3 Print Formulas	*drive=	cell:[F1]a clear
Sum ex; c10:[F1]suma0-z0	F4 Save Sheet	*clockon	(Here cell can
Avg ex; z29:[F1]avgk10-k25	F5 Print Sheet	*clockoff	be any cell.
Max ex; y2:[F1]maxa9-a22	F6 Set Precision	*settime=	Jump moves to
Min ex; z29:[F1]mina0-z0	F7 Graph Mode	*snapshot	area of cell,
Cop ex; a7:[F1]copz0-z29	F8 Dir Menu	*print	View shows
(copy places source	HELP HelpScreen	*secadd=,*height=	formulas, Clear
in a range of cells)		density=,rot=	clears cell.)

12

<<the following page has been reformatted from the original manual>>

Basic Write

In the beginning the BasicWrite program was to be an accessory to BasicPaint

and was not meant to grow in size beyond the bare minimum. However, it has

become in its own right another major part of your BASIC 8 library.

What exactly is this BasicWrite? It is certainly not a word processor.

Nor is it one of those 'Desktop Publishing' programs you have no doubt heard

of. Yet it has aspects of all of these and more.

I prefer to call it a more meaningful title- A Graphic Text Editor. By

that I mean it can combine many of the features found in traditional text

editors, plus it has the ability to incorporate graphic images into the body

of the document.

It does even more than that of course. You can include many different

text styles (fonts) on the same screen, in 16 colors and in 25 different text
sizes. You can also use the special graphic characters that are built into all Commodore 8 bit computers to create boxes, borders and other graphic accents that will enhance the page appearance.

The rest of this document will discuss the various abilities of BasicWrite.

There are two major screens you will use in BasicWrite. It has the WORK
page and the ESC page. The WORK page is where you actually create your documents. It has a large number of commands that you can access by pressing
the CTRL KEY and one of the predefined key commands. The other screen, called the ESC screen is where you go to change the various screen values such as left or right margins, tabs, page length or the printer driver. It is also here that you save your documents, load documents, brushes or pictures. And it is at the ESC screen you can print documents.

Another feature of the ESC screen is the FONT change keys. You can switch between the fonts currently in memory by pressing one of the function
keys. And you can load additional fonts into memory.

Finally there is a builtin help screen to refer to.

Work Page

When you first begin using the editing commands they will seem a little strange to you. In some ways they are quite easy to remember, in other cases
they may take a little practice for you to get used too.

Many of them are those you are already familiar with. If you want to move around the WORK screen use the four cursor kegs. To move to the top of the page press the HOME key. And if you want to CLEAR a page you would of course use the SHIFT/HOME key. The TAB key can be used to move your cursor in increments more than one letter at a time. And the SHIFT/TAB key will

move it backwards for the same mount.

Others just take a little memorization to get used to. The ESC key will

take you to the ESC page, and when on the ESC page the ESC keg will take you

back to the WORK page. Another of the easy keys to remember is the CTRL-F key. It is used to FETCH to the WORK page a brush to the current cursor position. (Of course the brush must have been loaded from the LOAD menu on the ESC page.)

CTRL-C can change the current character color, and CTRL-P the

13

color of the paper under the letter. This allows you to use the <<unreadable>> found on the 40 column screen. Only here there are absolutely

NO limits on the number of characters or fonts you can have at once when using the extra color features.

You can also use the INSERT Key to put a single space under the cursor and move the letters to the right (within the defined margins) over one letter. Other normal keys are the REVERSE and REVERSE OFF keys to switch from normal to reverse modes.

There are also many other keys that have special purposes. You can get a HELP MENU when you are on the ESC screen by pressing F3. You can see this help menu on the right. <<moved to bottom of page>>

On PAGE KEYS allow you to do many functions of text editors and standard word processors you may already be familiar with and use.

Besides the keys mentioned already, there are many more CTRL keys that make using this editor useful and unique.

For example, the CTRL-D keys will delete from the current cursor position to the right margin.

Another set of commands is for changing how the text is output to the screen. CTRL-B will set the text mode to BLANK everything under the cursor when it prints on the page. CTRL-O causes the text to be merged with

anything else under it. And CTRL-Y causes the text to EXCLUSIVE OR the text with the page under it.

Another set of CTRL KEYS is designed to make it easy to move around the page. CTRL-A will scroll the screen up 16 lines, and CTRl-Z will scroll it down 16 lines. If you want to get to the bottom of the page fast you can use CTRL-E. And of course the HOME key takes you to the top.

For extra emphasis you can cause all text to be underlined by using CTRL-U. This toggles under lining on and off. And for even more emphasis you can put graphic images in the form of BASIC 8 BRUSHES in the 8x8 color or monochrome modes. The command for fetching the graphic to the screen is CTRL-F. The brush is loaded from the ESC page load menu.

One of the most important commands allow you to change the character height and its width. The key commands that do it are CTRL-W (width) and CTRL-H. You can have five heights (1, 2,4,8,16) or five widths (1,2,4, 8,16). And you can mix these sizes so you really have 25 different sizes to choose from. When you change sizes, the cursor shape changes with the size. It will direct1y show 1x1, 1x2, 2x1 and 2x2 size characters. If the size exceeds these in a direction, the cursor will indicate that by losing its right side, bottom or both. And you can see the current height and width by pressing the ESC key. The ESC bar at the top of the screen has this information.

BASIC 8 Text Editor Key Commands

ESC Strip Commands	On Page Commands
F1-Font Forward	CTRL C Text Color
F2-Font Backward	CTRL P Page Color
F3-This Menu Box	CTRL D Delete Right
F4-Print Page	CTRL U Underline
F5-Load Menu	CTRL F Fetch Brush
F6-Save Menu	CTRL Rvs On or Off

F7-Load Fonts CTRL O OR Text
F8-Quit CTRL B Blank Text
CTRLQ-Page Parms CTRL Y XOR Text
CTRLW-Printer Setup CTRL A Scroll Up
+/- Change Drive CTRL Z Scroll Down
CTRL E Goto Bottom
CTRL W/H Widt/Hght
Clr Home
TAB/Shift TAB
ESC-Menu Strip

14

The last WORK page commands are accessed by the function kegs at the top right side of the C128 keyboard. They allow you easy movement around the monitor screen. The keys are: F1 Move to left margin. F3 Move to Right Margin. F5 Move to Screen Top. F7 Move to Screen Bottom. These keys, along with the HOME TAB SHIFT/TAB and CTRL-E keys will make your editing a lot easier.

ESC PAGE

The ESC page is where you do much of the input and output of BasicWrite. if you want to LOAD a page, picture or brush, you do it here. Pressing F5 brings up the LOAD MENU. It will ask you to press O (pages) P (pictures) or B for brushes. It will then look at the current drive for all files that start with one of the three prefixes (page., pict., or brus.). If there are any, they will be in the small window, and a black bar will be across the top one. You can choose one to load by putting the black bar on it and pressing return. It will be loaded. If you want to see more than the ten in the window, pressing the SPACEBAR will list more (if there any). And pressing the ESC key will bring you to the WORK page and abort the loading.

To SAVE your work, press the F6 key. The SAVE MENU appears and you

can type in the name of the document and save it to the current disk drive. You should always use the prefix PAGE. with your documents. You can use the prefix PICT. if you plan on using the documents in BasicPaint.

To change character fonts use F1 and F2, moving you forward and backward through the fonts in memory. If you need more than will fit in memory at one time, insert a disk with additional fonts and press F7. This will replace the ones currently in the memory of the system. When you want the old fonts back, change to the disk with them and press F7.

To change the drive you are using, at the ESC page press + or - And you will switch between drives 8-11.

If you want to print your page, press F4 and it will be output to the printer that is currently defined.

To change printers press (at the ESC page) CTRL-W. It will then search the drive for all the printer drivers it can find. You can change yours by pressing the + or - keys. When you have the one you want just press RETURN. You can then change the size, secondary address and height of the printed output. Once chosen, it is used until you change it again.

To change the page size and attributes, press CTRL-Q. This allows you to change the TAB value, height of the page in lines (which can be from 25 -66), and the margins (which are used to make columns).

F3 gives you a HELP menu, which can he used to refer to when you have forgotten a command.

And finally, there is F8. This function key is used to quit BasicWrite. You can use this from both the WORK page and the ESC Page.

I believe that this documents everything about BasicWrite. If I have forgotten some command or you find a bug, let me know.

Oh yes, BasicWrite is written completely in Basic 8!

Hope you like it!

Louis M. Wallace