TABLE OF CONTENTS

INTRODUCTION

CHROMA BASIC is a new program for use with a CHROMAtrs (T.M.) COLOR ADD-ON. Included in the CHROMA BASIC program are many, easy-to-use, graphics commands that can either be written into any Basic program or used independently. The advantage of this is that graphics programs can be run and the screen can be modified without rerunning the same program.

CHROMA BASIC is a driver program for the CHROMAtrs (T.M.) COLOR ADD-ON. CHROMA BASIC loads the required information for the commands into the CHROMAtrs (T.M.) COLOR ADD-ON and interprets the graphics functions. These functions make graphics programing much faster. Machine language or peeks and pokes are no longer necessary when using the CHROMAtrs (T.M.) COLOR ADD-ON.

CHROMA BASIC is compatable with most disk operation systems and allows use of editor functions and all the usual DOS capabilities with only a few restrictions. CHROMA BASIC's versatility and wide range of graphics commands will be pleasing to anyone who owns a CHROMAtrs (T.M.) COLOR ADD-ON.

OPERATION

DISK:

Now that you have received your Chroma Basic disk BACK IT UP at least twice. On the disk you received you will find the following programs:

*	CXXDISK/CMD -	Chroma Basic.
*	COLXX/CMD -	The Color Driver.
	DEMO/BAS -	A demo program written in Basic using the color driver. The color driver must be loaded to use this program. (Instructions to do this are in the CHROMAtrs users manual.)
	MLOGO/BAS -	A minilogo language containing a subset of LOGO commands. This is an updated version written in Chroma Basic. Chroma Basic must be loaded to use this program.
	MLOGO/DMO -	A demo program written in MLOGO. It can be loaded through the recall function of MLOGO. MLOGO must be loaded to use this program.
	UFO/BAS -	A game program written in Chroma Basic. Chroma Basic must be loaded to use this program.
	HOUSE/BAS -	A program written in Chroma Basic to demonstrate the use of the DRAW command. Chroma Basic must be loaded to use this program.
*	COLOR/ASM -	The commented source code of COLXX/CMD.

Now take a system DOS disk with normal disk Basic on it and copy Chroma Basic (CXXDISK/CMD) onto it. The Chroma Basic disk is so filled that normal disk Basic is NOT on it. You will use the disk you just created to enter Chroma Basic, it will be referred to from now on as the Chroma Basic disk. (One drive TRSDOS customers you will not be able to make a copy of CXXDISK/CMD, just go on to the TRSDOS procedure that follows.)

Chroma Basic is compatible with the following disk operating systems:

For the Model I:	NEWDOS/80 2.0, TRSDOS 2.3, LDOS.
For the Model III:	NEWDOS/80 2.0, TRSDOS 1.3, DOSPLUS 3.4, LDOS.

* - XX should be substituted with 32 or 48, 32 for use with a 32K system and 48 for use with a 48K system.

It is possible that Chroma Basic is compatible some disk operating systems not listed. Chroma Basic can be entered using the general procedure below. Following that are specific procedures for entering Chroma Basic through various disk operating systems.

1) Boot the Chroma Basic disk.
2) Enter normal disk Basic with one of the following memory sizes:
   For a Model I with 32K use 36100.
   For a Model III with 48K use 52100.
   Note: This memory size may be varied by a few hundred to allow Chroma Basic to work with a particular DOS.
*3) Execute the file CXXDISK/CMD.
4) The Chroma Basic logo will appear on the screen connected to the CHROMAtrs. When the READY prompt appears on the TRS-80 screen Chroma Basic will be entered.

TRSDOS:

1) Boot the Chroma Basic disk.
   (If you have one drive boot a TRSDOS system disk with normal disk Basic on it.)
2) Type 'BASIC' and press ENTER.
3) When 'HOW MANY FILES?' appears press ENTER.
   When 'MEMORY SIZE?' appears respond one of the following ways:
   If you have 32K type 36100 and press ENTER.
   If you have 48K type 52100 and press ENTER.
   If you have a 48K Model IV type 50000 and press ENTER.
*4) Type 'CMD"I","CXXDISK"' and press ENTER.
   (If you have one drive swap the Chroma Basic disk in the drive before entering the command above.)
5) The Chroma Basic logo will appear on the screen connected to the CHROMAtrs. When the READY prompt appears on the TRS-80 screen Chroma Basic will be entered. Now turn to page four of this manual.
   (If you have one drive now swap the system disk back into your drive.)

* - XX should be substituted with 32 or 48, 32 for use with a 32K system and 48 for use with a 48K system

NEWDOS/80 2.0

1. Boot the Chroma Basic disk.

*2. Type 'BASIC #####,CMD"CXXDISK"' and press ENTER.
 ##### - should be substituted with a memory size from the chart below.
 If you have 32K substitute 37200
 If you have 48K substitute 52800

3. The Chroma Basic logo will appear on the screen connected to the CHROMAtrs. When the READY prompt appears on the TRS-80 screen Chroma Basic will be entered. Now turn to page four of this manual.

Note: NEWDOS is not completely compatible with Chroma Basic. The following limitations are invoked while using Chroma Basic:

1) Do not enter mini-dos using 'DFG'.
2) Renumber does not work.
3) Renew does not work.

LDOS - the following procedure works on a 48K system but is not completely compatible with LDOS 5.1.3, the TEXT command of Chroma Basic will not work if the extensions are used. Although the color can be set by the COLOR command and a text magnification factor may be poked into E522H.

1. Boot the Chroma Basic disk.
2. Type 'LOAD C48DISK/CMD' and press ENTER.
3. Type 'LBASIC (M=50000)' and press ENTER.
4. Type '10 DEFUSR=&HD53F' and press ENTER.
 Type '20 X=USR(0)' and press ENTER.
 Type '30 CMD"R"' and press ENTER. (Model I only!)
 Type 'RUN' and press ENTER.
5. The Chroma Basic logo will appear on the screen connected to the CHROMAtrs. When the READY prompt appears on the TRS-80 screen Chroma Basic will be entered. Now turn to page four of this manual.

Note: The above Basic program may be saved on disk and then run each time Chroma Basic is loaded.

* - XX should be substituted with 32 or 48, 32 for use with a 32K system and 48 for use with a 48K system.

OPERATION

TAPE:

On the tape you received you will find the following programs:

SIDE A -	Chroma Basic -	Chroma Basic. (ONLY ONE COPY!)
SIDE B -	UFO -	A game program written in Chroma Basic. Chroma Basic must be loaded to use this program. (Not included on 16K cassette.)

The following steps will allow you to enter Chroma Basic.

1) Turn on the computer.
   (If you have a Model III When 'Cass?' appears type 'L' to set the baud rate to 500.)
   When 'MEMORY SIZE?' type in one of the following numbers:
   - 21698 - For a 16K Model I.
   - 21394 - For a 16K Model III.
   - 36100 - For a 32K TRS-80. (Model I or III)
   - 52100 - For a 48K TRS-80. (Model I or III)

   Afterwards press ENTER.
2) Place the tape in the tape recorder, rewind it, and press play.
3) Type 'SYSTEM' and press ENTER.
4) When '*?' appears type one of the following:
   - 'C16CS1' - For a 16K Model I.
   - 'C16CS3' - For a 16K Model III.
   - 'C32CAS' - For a 32K TRS-80. (Model I or III)
   - 'C48CAS' - For a 48K TRS-80. (Model I or III)

   Afterwards press ENTER.
   (Note: Two stars should appear in the upper right hand corner of the screen. The right most one should flash on and off. If it does not flash make sure you have typed in the right 'C' name from the chart above.)
5) After about three minutes '*?' should appear again. When it does type '/' and press ENTER. The Chroma Basic logo should appear on the screen connected to the CHROMAtrs. When the READY prompt appears on the TRS-80 screen Chroma Basic will be entered. Now turn to page four of this manual.

Note: If '*?' never appears a second time try sdjusting the volume of the tape recorder, make sure the cables are in properly, and TRY THE PRECEDING PROCEDURE A SECOND TIME. Only if all else fails call us.

By now Chroma Basic will have been loaded. The READY prompt should have appeared and everyday Basic programs can be run as usual (if you didn't know any better you might not realize Chroma Basic had been loaded.) However, 68 commands have now been added to Basic that facilitate easy control of the CHROMAtrs color add-on. The following pages contain an overview of Chroma Basic's commands.

LIMITATIONS:

Although normal Basic will operate there are some limitations that must be kept in mind while using Chroma Basic.

1) Never exit to DOS and then reexecute Chroma Basic unless the Exit command is used.
2) Never do a CMD "S".
3) Never enter the edit mode by the ',' single-character command.
4) Never use the 'A' command in the edit mode. If you need to do this, quit with the 'Q' command and reenter with 'E'. All other edit functions can be used with Chroma Basic.
5) Chroma Basic loads into high memory. This should be considered with use with DOS.
6) Certain enhancements to the Model III do not work.
7) The 16K cassette Model I version of Chroma Basic does not include any commands dealing with sprite velocities because of a lack of interrupts.
8) The 16K cassette and 32K diskette versions do not include 3D commands because of a lack of memory.
9) NEWDOS/80 2.0 is not completely compatible with Chroma Basic - see page 2c.
10) LDOS 5.1.3 is not completely compatible with Chroma Basic - see page 2c.

GRAPHICS COMMANDS

INIT

INIT will reinitialize the CHROMAtrs(T.M.) COLOR ADD-ON. It clears the screen and backdrop color to transparent, sets the last plotted location to (0,0), clears the automatic velocity table, sets LARGE, NOZOOM, reinitializes the graphics mode, etc. (i.e. it returns to the power-up condition.

INIT n

n is a number (0 or 1) that determines the graphics mode. When n is 0 the CHROMAtrs (T.M.) COLOR ADD-ON is in multi-color mode - a 64x48 block matrix. When n = 1 the CHROMAtrs is initialized for Graphics II mode - a 256x192 pixel matrix.

INIT 1
Presets the CHROMAtrs(T.M.) COLOR ADD-ON in Graphics II mode.

COLOR

Color changes the current foreground and background colors.

COLOR f (,b)
f is the foreground color (0-15) and b is the optional background color (0-15). If (,b) is not specified, the old background color is used. Colors used in this command must always be entered as decimal numbers (Items in () are optional).

COLOR 5,6
Sets the foreground color to light blue and the background color to dark red. The color will appear as an 8-bit code that will ocasionally bleed into the points next to those plotted. This is a limitation of the video processor chip and cannot be overcome.

SCREEN

SCREEN is used to change the color of the backdrop plane.

SCREEN c
c is the desired color of the backdrop plane (0-15).

SCREEN 8
This will set the screen to medium red.

CLR

CLR will clear the screen to the current background color.

CLR
If the background color had been set to 8 previously, then the screen would be cleared to medium red.

PLOT and UNPLOT

PLOT and UNPLOT are used to draw and erase lines and points.

(UN) PLOT x,y - Set (or reset) the given point. PLOT will change the color of the "sliver", while UNPLOT only resets the bit.

(UN) PLOT x,y TO a,b - Draw (or erase) a line from x,y to a,b.

(UN) PLOT TO x,y - Draw (or erase) a line from the last plotted point to x,y.

The above formats may be chained. For example:
PLOT 0,0,10,10 will plot two points, (0,0) and (10,10).
PLOT 0,0 TO 225,0 TO 255,191 TO 0,191 TO 0,0 will draw a box around the screen.

PAINT

PAINT will fill ANY closed figure, no matter how irregular, with a color. In graphics mode II the painting will stop at any SET point, no matter what color that point is.

PAINT x,y,c

x,y is the starting coordinate of the paint.
c is the color to paint to. This is only used in the multi-color mode. In the graphics II mode c is a dummy variable. The painting will stop at the first lit pixel in graphics mode II.

PAINT 100,100,8
In the graphics II mode, this statement will paint around the point (100,100) until it reaches a pixel that is set. In other words, if (100,100) was the center of a circle the paint statement would color the circle. In the multi-color mode this command would paint on the screen until a medium red (8) color is encountered. The paint command, in multi-color mode send an 4x4 pixel block to the screen. This occasionally causes the painted color to bleed over some of the lines of the figure. Graphics mode II only.(See COLOR).

POSITION

POSITION is used to position the invisible text cursor. This location will be used to define the upper-left corner of the next character to be displayed.

POSITION x,y

x is the x-coordinate
y is the y-coordinate

POSITION 10,10
10,10 is the coordinate for the upper-left hand corner of the next character to be displayed.

TEXT

TEXT is used to place letters, numbers, and symbols on the pattern plane. The starting posistion is determined by POSITION.

```
TEXT x$ (,c (,x size (,y size)))
```

x$ is the string to be printed. It may be a literal, string expression, or variable.

c is the optional FOREGROUND color, If it is not specified, the current color will be used.

x size
and
y size
are the optional scaling factor for each dot plotted. These default to 1. If they are specified, then c must also be specified.

```
TEXT "THIS IS A TEST",9,2,2
```

The message THIS IS A TEST will be sent to the screen in the color of light red, and magnified by 2. Care must be taken when displaying text. If any pixels are sent past the screen an overflow message will occur and the program will "bomb out".

WRITE

WRITE will save the entire pattern plane onto a disk file. The sprite patterns and locations will not be saved. The backdrop color will, however, be saved. The screen is saved in a compressed format.

```
WRITE "FILENAME"
```

FILENAME is a string containing the name of the file in standard DOS format.

SREAD

SREAD will load the entire pattern plane from disk. The backdrop color will be restored.

SREAD "FILENAME"

FILENAME is a string containing the name of the file in standard DOS format.

DASH

DASH will cause all lines drawn from then on (PLOT, 3-D rotation, circles, polygons, etc.) to be dashed by a certain ratio.

DASH x,y

x is the number of ON pixels
y is the number of OFF pixels
If either x or y is zero then the dashing is cancelled.

DASH 3,2
For every three pixels drawn the next two will be skipped.

POLYGON

POLYGON will draw a polygon with the number of sides specified, at the place specified, and at the angle specified. The polygon will be drawn in the current color.

POLYGON c,x,y,xr,yr,s(,a(,n))

c is 0 for drawing, 1 for erasing,
x and y are the coordinates for the center points,
xr and yr are the x and y radii,
s is the number of sides (2 is less than or equal to s
  which is less than or equal to 89),
a is the optional starting angle (in DEGREES),
n is the optional number of lines to be drawn (1 is less
  than or equal to n which is less than or equal to s).
  When n is specified only n sides of the polygon will be
  drawn.(Note when n is specified a must also be specified.

POLYGON 0,100,100,30,30,8
This statement draws an octagon (8-sided polygon) around the point (100,100) at a distance of 30 pixels.

CIRCLE

CIRCLE will draw a circle or oval. The parameters for circle are almost the same as polygon with the exception of s,a, and n which are ommitted for obvious reasons.

```
CIRCLE c,z,y,xr,yr(,b,e)
```
see POLYGON for parameters except (,b,e).
b is the beginning angle (in degrees) of the arc,
e is the ending angle (in degrees) for the arc.

```
CIRCLE 0,60,70,40,40
```
When this circle statement is executed a circle will be drawn with the point (60,70) as the center and a radius of 40 pixels.

PIE

This command combines line drawing with arc drawing. It will draw the arc, and then connect the ends of the arc to the center point of the arc with two lines, thus forming a "pie slice."

```
PIE s,x,y,xr,yr,b,e
```
See CIRCLE above.

```
PIE 0,60,70,40,40,0,180
```
This will create a semicircle (half the circle created in the circle example).

STORE

STORE is used to save a block of pixels in an integer array. This command can only be used in graphics II mode. Care must be taken when dimensioning the array or errors will occur.

```
STORE X1,Y1 TO X2,Y2,A%(0)
```

X1,Y1	are the coordinates of one corner of the block
X2,Y2	are the coordinates of the opposite corner
A%(0)	is an integer array subcriated zero. It must be dimensioned to the absolute value of (X2-X1+1)*(Y2-Y1+1)/16.

STORE 10,10 to 30,30,A%(0)
This command will take the pixels inside the rectangle defined by the points (10,10),(10,30),(30,10),(30,30) and store them in the array A%. A% must have been previously dimensioned to 55 in the program.

RETRIEVE

RETRIEVE is used to restore a block of pixels that was stored by the STORE command (see above). If the width and height of the block of pixels is different, the results will not be those expected. GR II only.

RETRIEVE x1, y1 TO x2,y2,a% (0)
See STORE above.

RETRIEVE 100,100 TO 120,120,A%(0)
The rectangle defined by (100,100) , (100,120) , (120,100) , (20,120) will be filled by the pixel block that was previously stored in array A%(0).

DRAW

The DRAW command will draw, scale, and rotate any figure on the screen that is being created with the draw command.

DRAW x$

where x$ contains:
Mx,y - move to x,y location
Ux, Dx, Rx, Lx, Ex, Fx, Gx, Hx - draw in the corresponding direction x units according to this direction chart:

```
        U
    H   :   E
 L  -   *   -   R
    G   :   F
        D
```

Ax - rotate every movement from now on x degrees clockwise.
Tx - rotate every movement x degrees clockwise until cancelled (T0) or until the end of the DRAW string (temporary rotation). The actual direction that will be drawn in is the direction (U,R, etc.) plus the offset of the A parameter PLUS the offset of the T parameter.
N - don't update next line. The line will be drawn, but the next position to draw at will remain the same.
B - don't draw. The next movement will not draw, but only move the graphics cursor.
Zx - change drawing mode: 0=set points, 1=reset points.
Cx,y - set foreground/background color: x=foreground, y=background.
; - separator for multiple commands.

The N and B commands are not to have a semicolon separating them from the respective commands that they are being used with. (See the sample program below.)

To help with the draw command a small sample program is included here. Type this program in and run it to see how the draw command works. (It is included on disk, named HOUSE/BAS.)

SAMPLE PROGRAM

```
 10 CLEAR 500
 20 INIT 1
 30 COLOR 10
 40 REM DRAW THE SUN
 50 CIRCLE 0,30,20,10,10
 60 PAINT 30,20,10
 70 A$ = ";NU20;NE20;NR20;NF20;ND20;NG20;NL20;NH20"
 80 DRAW "M30,20;C10,0"+A$
 90 DRAW "M30,20;A22"+A$
100 REM DRAW THE HOUSE
110 DRAW "M100,140;C12,0;A0;U60;E40;F40;NL54;D60;L22;U20;L10;
          D20;NR11;L22"
120 CIRCLE 0,129,132,2,2
130 PAINT 129,132,12
140 REM DRAW THE WINDOWS
150 A$ = ";U8;R14;D8;L14;BR7;U8;BD4;R7;L14"
160 DRAW "M104,130"+A$
170 DRAW "M136,130"+A$
180 DRAW "M136,110"+A$
190 DRAW "M104,110"+A$
200 PAINT 124,90,12
210 REM PAINT THE HOUSE
220 COLOR 5
230 PAINT 101,139,5
240 REM DRAW AND PAINT THE CHIMNEY
250 DRAW "M148,89;C6,0;U20;L12;D9"
260 PAINT 146,85,6
270 END
```

BLANK

This command blanks the entire display with the exception of the background color without erasing the CHROMAtrs' (T.M.) COLOR ADD-ON memory.

```
BLANK
```

Clears the screen and leaves the memory intact.

NOBLANK

This command will restore the CHROMAtrs' screen to its previous contents

```
NOBLANK
```

If the screen was filled with stars before the BLANK command it will again be filled with stars after the NOBLANK command.

DUMP

DUMP will take the entire Graphics II or multi-color plane and dump it to any Epson MX-70, MX-80, or MX-100 equipped with GRaftrax-80 or Graftrax-Plus. The resulting picture is approximately 16 cm by 14 cm, with each pixel being represented by a 3 x 2 block of dots, with their density depending upon the color of the pixel. Sprites are not printed.

```
DUMP
```

Do not dump without a line printer and Graftrax-80 of Graftrax Plus because the routine is not error protected. To abort the dump press the BREAK key.

WINDOW

WINDOW is used to limit all pixel plotting to a certain window. Depending upon the opiton selected, all pixels plotted will be within (or outside of) a specified box. All pixels inside (or outside) of this box will be completely ignored. (unless they are plotted past the screen).

```
WINDOW x1,y1 TO x2,y2,m
```

x1, y1 is the coordinate of the upper-left corner of the box x2,y2 is the coordinate of the lower-right corner. 0 is the option m: 0 - draw only inside, 1 - draw only outside.

```
WINDOW 10,10 TO 150,150,0
```

The execution of this statement will only allow graphics to be drawn within the square window defined by the points (10,10),(10,150),(150,10),(150,150).

THREE-D COMMANDS

These CHROMA BASIC commands will manipulate and/or display three dimensional objects. A very rigid set of rules must be followed to use these commands. A few changes have been made in the accepted manner of representing three dimensional figures for ease and speed of programming and running the CHROMA BASIC driver.

The three dimensional figure is stored in two arrays. The first array must be a single precision array dimensioned (n-1,2), where n is the number of vertices in the figure. (0 counts as a subscript, hence there will be enough room to store the coordinates of every vertex).

The second array must be an integer array and must also be subscripted (m-1,1), where m is the number of lines in the figure. The subscript of 1 is to allow for two numbers (the two vertices that are connected). Every pair of numbers read into the array will determine which set of vertices are connected by lines. The vertex number is determined by the first subscript of the single precision array.

Take for example this square represented in three dimensions:

```
     (0,-10,-10)   a    (0,-10,10)
         +------------------+
         : 0              1 :
       b :                  : c
         : 3              2 :
         +------------------+
     (0,10,-10)    d     (0,10,10)
```

Note that the numbers in parentheses are the three dimensional coordinates. The first coordinate in this representation is always 0 since a square has no depth. WARNING! The coordinate system has been shifted for efficiency in programming. Remember that the first coordinate is depth, the second is height, and the third is width. In other words the coordinates look like (z,y,x).

The numbers 0-3 are the vertex numbers (note that they start at zero). a-d are the lines connecting the vertices. To set this up the following must be taken care of:

First, the single precision array must be arranged. For this example the array is A(x,x). The A array is dimensioned A(3,2) for 4 vertices with 3 coordinates each. Next the second array must be initialized. In this example the integer array will be B%(x,x) and this must be dimensioned B%(3,1) for 4 lines connecting two points for each line.

Now the arrays are set up as follows:

VERTEX NUMBER	X DEPTH	Y HEIGHT	Z WIDTH	CONNECTION NUMBER	1st VERTEX	2nd VERTEX
X	A(X,0)	A(X,1)	A(X,2)	Y	B%(Y,0)	B%(Y,1)
0	0	-10	-10	0	0	1
1	0	-10	10	1	1	2
2	0	10	10	2	2	3
3	0	10	-10	3	3	0

Here is a small program to set up these arrays and display the square.

```
10 INIT 1: SCREEN 5: COLOR 6
20 DIMA(3,2),B%(3,1)
30 FOR X = 0 TO 3:FOR Y = 0 TO 2:
   READA(X,Y):NEXTY,X
40 FOR X = 0 TO 3:FOR Y = 0 TO 1:
   READB%(X,Y):NEXTY,X
50 DATA 0,-10,-10,0,-10,10,0,10,10,0,10,-10
60 DATA 0,1,1,2,2,3,3,0
70 DISPLAY A(0,0),1,B%(0,0),128,96
80 END
```

Note: see appendix a for another sample 3-D program.

DISPLAY

The small program above illustrates the DISPLAY command. This command places the three dimensional figure on the screen at the position specified in the command. The command format is:

DISPLAY arraya(0,0),scale,arrayi(0,0),x,y where

arraya	is the single precision array holding the vertex coordinates (it must be subscripted (0,0)).
scale	is the scaling factor for displaying the figure (0 - infinity). The scale must, however, be small enough so that the figure can be drawn within the boundary of the screen. If not an error message will occur.
arrayi	is the integer array of vertex connections.
x,y	the center point for the figure.

```
ROTATEX array(0,0),degrees
ROTATEY array(0,0),degrees
ROTATEZ array(0,0),degrees
```

The three rotate commands all perform close to the same functions. First they all rotate the three dimensional figure around a specified axis, and second they all have the same parameters. In all three rotate commands the parameters are:

array(0,0)	which is the single precision array
degrees	the amount of degrees rotation given as an integer.
ROTATEX	will rotate the three-d figure around the x-axis. REMEMBER that the X-axis in CHROMA BASIC is depth.
ROTATEY	will rotate the three-d figure around the Y-axis. The Y-axis is height in CHROMA BASIC.
ROTATEZ	This will rotate the three-d figure around the Z-axis which in CHROMA BASIC is width.

REMEMBER in CHROMA BASIC the (x,y,z) coordinate system is adjusted to be (z,y,x).

STRETCH

The stretch command stretches a figure by the factor put in the command. The array can be magnified by any amount along the x,y,z axes.

```
STRETCH array(0,0),x,y,z
```

array	is the single precision array.
x,y,z	are the magnification factors. They must have a decimal point with at least one digit following them or else the stretch will not work properly.

SOUND COMMANDS

For all of the folowing, v is the volume (0-31), d is the duration (0-65535, with 32768-65535 being expressed as the negative numbers -32768 to -1), and p and q are pitches (0-255, with 0 used as 256. Note that 1 is the highest and 0 (256) is the lowest).

The interrupts are disabled for the sound commands. This means that time is taken away from the central processor to run the sound commands, and therefore all sprite movement is halted for the duration of the respective sounds. Most sound commands may be aborted by pressing (BREAK).

SOUND

Sound is used to create a pure tone.

SOUND p, d, v

This small program will quickly play each pitch and print the pitch number on the terminal screen.

```
10 FOR X = 0 TO 255
20 PRINT X;" ";
30 SOUND X,100,20
40 NEXT
50 END
```

DUTY

DUTY is used to produce sounds of varying tone quality. This is accomplished by changing the duty cycle of the waveform produced.

DUTY p,q,d,v
(see the first paragraph of sound commands.)

The actual tone produced will have a frequency of (P+Q)/2. d is the duration for the first half of the wave, and Q is the duration for the second half.

DUTY 100,121,1000,20
This will produce a sound unlike anything the SOUND

EFFECT

EFFECT is used to output two tones at once.

EFFECT p,q,d,v
(see the first paragraph of sound commands)

The duration of this sound must be much higher than the other durations above. This is because the duration here does not specify the number of wave periods that reach the speaker, but instead specifies the actual number of machine language cycles used to produce the sound.

EFFECT 10,200,-1,31
This will output the tones 10 and 200 to the TV/monitor for the greatest amount of time and volume as possible.

HARMONY

HARMONY is almost the same as EFFECT, except that the pitch is lower and has a different tone quality.

HARMONY p,q,d,v

See EFFECT.

HARMONY 10,200,-1,31
This will give the same duration of the sound as the EFFECT command with a lower tone. Execute this little program to compare the two sounds.

```
10 EFFECT 10,200,-1,31
20 FOR X = 1 TO 100: NEXT X
30 HARMONY 10,200,-1,31
40 END
```

STATIC

Static is used to generate static. This is the only sound command that does not turn off the interrupts.

STATIC p,d,v

STATIC 100,2000,31
Just plain static.

SIRENUP AND SIRENDOWN

SIRENUP and SIRENDOWN will produce an up or down siren.

```
SIRENUP v (,d (,start (,end)))
SIRENDOWN v (,d (,start (,end)))
```

d	is the duration of EACH tone produced. It defaults to 1.
SIRENUP	will normally produce tones starting at 255 and ending at 1.
SIRENDOWN	will produce tones starting at 1 and ending 255.

When (,start) is specified the (255) SIRENUP or (1) SIRENDOWN is changed to the value specified. If start is specified, (,d) must be specified. When end is specified the (1) SIRENUP OR (255) SIRENDOWN is changed to the value specified. If end is specified, then, (start) and (,d) must also be specified.

SIRENUP and
SIRENDOWN
Use this small program for a demonstration of these two commands.

```
          10 SIRENUP 31,7
          20 SIRENDOWN 31,7
          30 END
```

STEREO

STEREO will output two tones simultaneously - one to the TV/MONITOR speaker, and the other to the amplifier connected to the cassette port.

STEREO p,q,d,v

p is the pitch for the amplifier
q is the pitch for the TV/MONITOR

STEREO 10,200,10000,31
If you have some amplifier connected to the cassette port this command will send a tone similar to SOUND 10,10000,31 to the amplifier and a tone similar to SOUND 200,10000,31 to the TV/MONITOR.

SPRITE HANDLING COMMANDS

For all of the following commands, p is a sprite pattern number (0-63), s is a sprite number (0-31), and c is a color (0-15).

A sprite pattern, as referred to above, is a constant 8x8 or 16x16 pixel mapping of a bit pattern. A sprite consists of binary ones and zeros that, for an 8x8 sprite, look like:

BINARY	=	HEXEDECIMAL
0 0 0 0 1 0 0 0	=	08
0 0 0 0 1 1 0 0	=	0C
0 0 0 0 1 1 1 0	=	0E
1 1 1 1 1 1 1 1	=	FF
1 1 1 1 1 1 1 1	=	FF
0 0 0 0 1 1 1 0	=	0E
0 0 0 0 1 1 0 0	=	0C
0 0 0 0 1 0 0 0	=	08

The ones above represent the "on" pixels for the sprite to be created. A group of 8 binary digits are taken together as two hexedecimal digits to form one of the eight lines of the sprite. The hexedecimal digits for the above sprite are (080C0EFFFF0E0C08) and the sprite created by these digits is a right arrow.

A 16x16 sprite consists of four 8x8 sprite patterns (represented as I - IV below) set up like:

```
  I  ! III
-----!-----
 II  ! IV
```

To make a 16x16 sprite the four patterns should be read in from I to IV in numerical succession. To make a 16x16 sprite that has on the first row a right-arrow (sprite I) pointing to a left-arrow (sprite III) and the second arrow a right arrow (sprite II) pointing to a left arrow (sprite IV) this pattern should be read in:
(780C0EFFFF0E0C08080C0EFF0E0C08
80C0E0FFFFE0C8080C0E0FFFFE0C08).

SMALL

Set up the 8 x 8 sprite mode. The sprite sizes (SMALL and LARGE) are universal and effect all sprites immediately.

SMALL
Sets sprites to small (8x8).

LARGE

Set up the 16 x 16 sprite mode.

LARGE
Sets sprites to large (16x16).

ZOOM

Cause all of the sprites to be magnified by a factor of four (8x8 sprites become 16x16, 16x16 sprites become 32 x32). The sprite magnification modes (ZOOM and NOZOOM) effect all sprites immediately.

ZOOM
Enlarges all the sprites.

NOZOOM

Return all sprites to their original size (8x8 or 16x16).

NOZOOM
Shrinks all the sprites.

CREATE

CREATE is used to define a sprite pattern.

```
CREATE p, x$...
x$ is a string of either 16 or 64 hex character
   (depending on whether you are in the 8x8 or
   16 x 16 sprite mode) that defines the sprite.
   Each hex character defines one four-pixel wide
   area of the sprite.
```

In CREATE and all of the following commands that have "..." after them, the command may be replicated. For example:

CREATE 0, "FFFFFFFFFFFFFFFF",1, "0000000000000000" will define two patterns. 0 and 1.

DEFPAT

DEPPAT is used to define a sprite pattern from an integer array.

```
DEFPAT p,a%(0) ...
```

a%(0) is a zero-subscripted integer array consisting of integer numbers from evaluated hexedecimal pixel codes. The array must be dimensisoned at least (7) for 8x8 sprites and at least (31) for 16x16 sprites. For example:

```
DEFPAT 5,G%(0)
```

This will assign the pattern in G%(0), if it is set up correctly, to the sprite pattern number 5.

UNPAT

UNPAT will copy a sprite pattern into an integer array. It is the exact opposite of DEFPAT.

```
UNPAT p, a%(0) ...
```

See DEFPAT.

```
UNPAT 5,A%(0)
```

If the array is dimensioned properly (either 7 or 31), sprite 5 will be stored in array A%(0).

ASSIGN

ASSIGN is used to assign a sprite pattern to a sprite.

ASSIGN p TO s (,c)...

If the sprite color, (c), is specified, then the sprite will become that color. If it is not specified, then the sprite will retain its old color. The sprite will always maintain its old position.

In order to replicate the ASSIGN command with no (,c) specified, an extra comma must be included to differentiate the (,c) parameter from the next pattern number. For example:

ASSIGN 6 TO 2,5,8 TO 23,,10 To 0

This will take the sprite pattern 6 and assign it to sprite number 2 and the color will be color number 5. The second section will assign the pattern number 8 to sprite number 23, no color is specified for this sprite. The last sprite definition also sets no color for the sprite. Pattern number 10 will be assigned to sprite number 0.

SCOLOR

SCOLOR is used to change the color of a sprite.

SCOLOR s,c ...

s is the sprite number
c is the color the sprite will become.

SCOLOR 8,6

This will give sprite number 8 a color of dark red.

MOVE

MOVE is used to move a sprite.

MOVE s TO x,y ...

x,y are the coordinates of the upper-left hand point of the sprite.

MOVE 5 TO 100,100

This would place the sprite with its upper left-hand corner at the point (100,100).

MREL

MREL is used to relatively move a sprite.

MREL s, x offset, y offset...

x offset and y offset are the numbers that will be added to the x and y coordinates of the sprite. They are in the range (-255 to 255).

MREL 5,20,20
Calculating from the previous example, the upper left-hand corner of sprite 5 will be at 120,120.

PLACE

PLACE will copy a sprite from its current x,y location into the pattern. The sprite will be copied dot by dot, taking into account LARGE or SMALL and ZOOM or NOZOOM. The copy will be the same color as the original sprite. The sprite is not changed in any way. GR II only.

PLACE s

s is the sprite number.

PLACE 5
Places sprite 5 on the multi-color plane.

ERASE

Erase will clear all of the sprite patterns to zero, and move all sprites off the screen.

ERASE
Clears all sprites from the screen.

VELOC

VELOC is used to set the velocity of a sprite. The sprite will be moved automatically at this velocity by the real time clock interrupts. Any number of sprites can have velocities. The sprites will all be moved simultaneously except under the following circumstances (when all sprite movement will halt):

1) During the execution of most CHROMA BASIC sound commands (with the exception of STATIC).
2) When the interrupts are turned off (Model I CMD"T").
3) Whenever a disk drive's motor is on (this is to speed up disk access).
4) Whenever a PAUSE command is in effect.
5) When at Basic READY.
6) During the execution of many CHROMA BASIC commands.

```
VELOC s, x, y ...
```

x and y are the x and y velocities MULTIPLIED BY 256. Thus, to make a sprite move one pixel every interrupt (40 times per second), you would specify a velocity of 256. To make it move one half pixel every interrrupt, you would specify 128.

```
VELOC 5,256,0
```

Sprite 5 will now move at a horizontal velocity of 256 (40 pixels per second.)

CLRVELOC

CLRVELOC will set the velocity of all sprites to 0,0.

```
CLRVELOC
```

PAUSE

PAUSE will temporarily suspend movement of all sprites until a NOPAUSE command is executed. The interrupts, and sound commands, etc. are not affected.

NOPAUSE

NOPAUSE will cause a resumption of sprite movement.

NOPAUSE

The sprites will begin halt movement again.

LEFT

LEFT will set the bit in the sprite attribute table that causes a sprite to be displayed 32 pixels to the left of its actual posistion. This can be used to slide a sprite in from the left side of the screen. To compensate for this move, the sprite's x coordinate is incremented by 32. Thus, the sprite appears not to move. This movement is also transparent to the program because the MOVE and &SPRX commands compensate for this (the &SPRX command may return a negative number -32 to -1). However, the sprite can no longer slide off the right side of the screen (it wraps around at physical position 224).

LEFT n

n is the number of the sprite (0-31)

LEFT 5

This will cause sprite 5 to disappear at x-coordinate 224 and reappear at x-coordinate 0 once it has reached that position.

NORMAL

NORMAL will reverse the effect of LEFT (see above). It resets the bit, and decrements the x coordinate by 32, again with no visible effect in the program.

NORMAL n

n is the sprite number (0-31)

NORMAL 5

Normal 5 will reset sprite 5 so that it wraps around at x-coordinate 255.

EXIT

CHROMA BASIC's sprite handling commands are automatically processed. This comand will restore the normal interrupt vector table and allow the processor to accept interrupts and process them faster. (i.e. the keyboard wil return to normal speed.) The computer may also crash if a high-memory routine is executed.

EXIT
This command wil ask for verfication. The sprites will stop moving, and your TRS-80 will return to the Disk Operating System.

SPRSAVE

SPRSAVE is used to save all sprite patterns, attributes, and size information onto a disk file.

SPRSAVE x$

x$ is the filename in standard DOS format.

SPRSAVE "SPRITE"
Saves all the sprites on the screen in the file SPRITE.

SPRLOAD

SPRLOAD is used to restore the sprites saved on a disk file with the SPRSAVE command. The size and zoom switches are restored.

SPRLOAD x$,p

x$ is the filename in standard DOS format.
p is the loading option: if p = 0, then the sprite's y-coordinate will be set to 192 (off the screen). If p is 1 the sprites will be loaded on the screen where they were when they were saved.

MISCELLANEOUS FUNCTIONS

The following function must be used on the RIGHT side of an equation.

For example:

```
A=&POINT (0,0)
PRINT &SPRX (0)
```

&POINT

POINT will return the on/off status of a graphics pixel. The value returned will be -1 if the pixel is off, or the color (0-15) if the pixel is on.

&POINT (x,y)

x,y are the coordinates of the pixel to be tested.

Z = &POINT(100,100)
If point (100,100) is on then, &POINT will return the color (0-15) and if not it will return a -1.

&JOYSTK

JOYSTK will return the status of a joystick. The fire button will NOT be returned. If the specified joystick is inactive, then the keyboard will be scanned for the arrow keys. The returned values will be the same.

&JOYSTK (n)

n is number of the joystick (0-1).

The following numbers will be returned depending on the position of the joystick:

```
    10  8  9
      !  !  !
    2 -  *  - 1
      !  !  !
      6  4  5
```

X = &JOYSTK(0)
This will return a number (0,1,2,4,5,6,8,9,10) depending on the position of joystick 0.

&PDL

PDL will return a number (0-255) that corresponds to the position of a paddle knob.

&PDL(n)

n is the number of the paddle (0-3).

Y = &PDL(1)
This polls paddle (1) and returns the position into Y.

&STRIG

STRIG will return the status of a joystick's fire button. The value will be -1 if the button is depressed, and 0 if it is not pressed. If the joystick trigger is inactive, then the space bar will be tested instead.

&STRIG(n)

n is the number of the joystick(0-1).

Z = &STRIG(0)
If joystick (0)'s fire button is depressed then z = -1 if not, z will be 0.

&PTRIG

PTRIG will return the status of a paddle's fire button. The value returned is the same as &STRIG above.

&PTRIG(n)

n is the paddle number.

W = &PTRIG(1)
If paddle one's fire button is depressed then W will equal negative 1.

&SPRX and &SPRY

SPRX and SPRY will return a sprite's current x and y coordinates, respectively.

```
&SPRX (s)
&SPRY (s)
```

s is the number of the sprite (0-31).

```
X1 = &SPRX(5)
Y1 = &SPRY(5)
```

These two commands will, when executed together, return the current coordinates of sprite 5. If the sprite was at (100,100) then X1 and Y1 would both equal 100.

&SPRC

SPRC will return the current color of a sprite (0-15).

```
&SPRC (s)
```

s is the number of the sprite (0-31).

```
C = &SPRC(5)
```

If sprite 5 was black then C would equal 1.

&SPRP

SPRP will return the current pattern assigned to a sprite (0-63).

```
&SPRP (s)
```

s is the number of the sprite (0-31).

```
P = &SPRP(5)
```

If sprite 5's pattern number was 0, then p would equal 0.

&UNCREATE

UNCREATE will take a sprite pattern and convert it into a hex string that is either 16 or 64 characters long (depending on whether you are in the 8 x 8 or 16 x 16 sprite mode). UNCREATE is the exact opposite of CREATE.

&UNCREATE (n)

n is the sprite pattern number (0-63).

X$ = &UNCREATE(5)
    X$ will equal the hexadecimal string for pattern number 5.

&COINCE

COINCE will return a 0 if there are not two sprites coinciding (pixel by pixel) on the screen, and -1 if there are two or more.

&COINCE (0)

0 is a dummy argument

C = &COINCE(0)
    If there were two sprites coinciding c would equal -1.

&FIFTH

FIFTH checks to see if there are 5 or more sprites on a single horizontal TV scan line. If there are less than 5 sprites on the same line, then a -1 is returned. Otherwise, the number of the lowest priority sprite on that line is returned.

&FIFTH (0)

0 is a dummy argument

F = &FIFTH(0)
    F will equal -1 unless 5 sprites are on a line, then it will equal the lowest priority sprite's number.

&XVELOC and &YVELOC

XVELOC and XVELOC will return the current x and y velocity of a sprite.

```
&XVELOC (n)
&YVELOC (n)
```

n is the number of the sprite (0-31).

```
X = &XVELOC(5)
Y = &YVELOC(5)
```

The execution of these two commands will return both the x-velocity and the y-velocity of sprite 5 in x and y respectively.

We here at South Shore Computer Concepts feel that this software modification to the Basic interpreter (with its new commands) will enhance programing of the CHROMAtrs immensely. Enjoy using the enhanced commands of CHROMA BASIC. If you have any problems please call us.

APPENDIX A

Here are a few sample programs to help initiate the excitement of CHROMA BASIC programming.

This first program is for a three dimensional figure. It designs a three dimensional figure and then uses the rotate commmands to give different views of this figure.

```
10 INIT 1: SCREEN 9: COLOR 1
20 DIMA(7,2),B%(13,1)
30 FOR X = 0 TO 7: FOR Y = 0 TO 2
40 READ A(X,Y)
50 NEXT Y: NEXT X
60 FOR X = 0 TO 13: FOR Y = 0 TO 1
70 READ B%(X,Y)
80 NEXT Y: NEXT X
90 FOR X = 1 TO 3: ROTATEXA(0,0),25: GOSUB 500: NEXT X
100 FOR X = 1 TO 3: ROTATEYA(0,0),25: GOSUB 500: NEXT X
110 FOR X = 1 TO 3: ROTATEZA(0,0),25: GOSUB 500: NEXT X
120 END
130 DATA -10,-10,-10,-10,-10,10,10
140 DATA -10,-10,10,-10,10,-10,10
150 DATA -10,10,10,-10,10,10,10,-10
160 DATA 10,10
170 DATA 0,1,1,3,3,2,2,0,0,4,4,5,5,2
180 DATA 5,6,6,7,7,4,6,3,1,7,0,3,2,1
500 CLR: DISPLAYA(0,0),7,B%(0,0),128,96: RETURN
```

This program will create a pie and display the parts of the pie according to the amounts you enter. The main purpose of this program is to encourage modifications to familiarize you to CHROMA BASIC.

```
10 INIT 1: COLOR 15: DIM A(50)
20 CLS: PRINT"PIE CHART": PRINT
30 PRINT"ENTER THE NUMBER OF ITEMS IN EACH GROUP, -1
TERMINATES."
40 PRINT: X=1
50 INPUTA(X): IF A(X) = -1 THEN X= X-1: GOTO 80
60 S = S + A(X)
70 X = X + 1: GOTO 50
80 G = 360/S
90 P = 0
100 FOR Y = 1 TO X
120 IF Y = X THEN L = 360 ELSE L = A(Y) * G + P
130 PIE 0,128,96,95,95,P,L: P = L
140 NEXT Y
```

We hope that these programs help ease the transition into CHROMA BASIC programming. If you have any questions please do not hesitate to call.