

SYSTEM DRIVER FOR COLOR GRAPHICS COMPUTER

by

ru

MARILYN MCCORD DILLINGER

B.S., Kansas State University, 1961

M.S., Kansas State University, 1978

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1980

Approved by:

William J. Hankley
Major Professor

SPEC
COLL
LD
2668
.R4
1980
M32
C.2

TABLE OF CONTENTS

LIST OF FIGURES	iv
I. INTRODUCTION	
A. OVERVIEW	1
B. PAPER ORGANIZATION	2
C. BACKGROUND	2
D. MOTIVATION	
1. Human Engineering	3
2. Learning & Extension	4
3. Remote Teaching	5
E. DESCRIPTION OF PROJECT	
1. Program Structure	5
2. Evaluation of Results	9
3. Learnings	10
II. USERS' GUIDE	
A. THE MACHINE	
1. The Screen	12
2. The Disks	16
3. The Tablet	16
B. THE PROGRAM	
1. Initial Conditions	17
2. Loading and Running	17
C. THE KEYBOARD AND THE COMMANDS	
1. Introduction to the Keyboard	23
2. Redefined Keys	23
3. Shift	26

4. States	26
5. Mode Codes	29
6. Control Codes	32
7. Escape Codes	33
8. Miscellaneous Keys	33
D. THE PRIMITIVES	
1. Graphics Mode	35
2. Character Mode	35
E. OTHER FUNCTIONS & COMMANDS	
1. Create Buffer	39
2. Zoom	40
3. Complex Fill	40
4. Windowing	41
III. PROGRAM INFORMATION AND POSSIBLE MODIFICATIONS	
A. PROGRAM DETAILS	
1. Explanation of variables	44
2. Explanation of Code Sections	49
B. EXPANSION OF PRIMITIVES	
1. Built-in Primitives	56
2. Extended Primitives	56
C. TABLET SIZE AND SCALING	
1. User-Defined Tablet Size	57
2. Scaling to Avoid Distortion	57
D. FUNCTION KEYS	58
E. MULTIPLE WINDOWS	59
BIBLIOGRAPHY	60

APPENDIX

A. SOURCE CODE WITH COMMENTS

1. Part 1	62
2. Part 2	63
3. Part 3	68
4. Part 4	73
5. Alternate Tablet Initialization Code	81

B. SAMPLE SESSION 1	82
---------------------------	----

C. SAMPLE SESSION 2	87
---------------------------	----

D. SAMPLE SESSION 3	94
---------------------------	----

LIST OF FIGURES

FIGURE 1.	Control Flow of Program	6
FIGURE 2.	Logical Structure of Main Driver	6
FIGURE 3.	Overlay Routines	8
FIGURE 4.	Relationship to Chromatics Software	8
FIGURE 5.	The Chromatics Terminal	13
FIGURE 6.	The Display Screen	13
FIGURE 7.	Chromatics Disk Drives	14
FIGURE 8.	Diskette	14
FIGURE 9.	The Tablet	15
FIGURE 10.	4-Button Cursor	15
FIGURE 11.	Key Summary	19
FIGURE 12.	Keyboard Layout	24
FIGURE 13.	Redefined Keys	25
FIGURE 14.	The Primitives	36
FIGURE 15a.	Upper Case, Alternate-Character-Set	37
FIGURE 15b.	Shift, Alternate-Character-Set	38
FIGURE 16.	Flags Used in Part 2	45
FIGURE 17.	Overlay Routine Parameters	46
FIGURE 18.	Code Values for Special Function Keys	51
FIGURE 19.	Summary of Machine States	52
FIGURE 20.	Determination of Background Color	55
FIGURE 21.	Built-in Primitives Available on Chromatics	55
FIGURE 22a.	Stars and Bars	92
FIGURE 22b.	Adjusted Stars and Bars	93
FIGURE 23.	Concentric Circle Complex Fill	98

**THIS BOOK
CONTAINS
NUMEROUS PAGES
WITH DIAGRAMS
THAT ARE CROOKED
COMPARED TO THE
REST OF THE
INFORMATION ON
THE PAGE.**

**THIS IS AS
RECEIVED FROM
CUSTOMER.**

I. INTRODUCTION

A. OVERVIEW

This report describes a system driver designed for the Chromatics CG (Computer Graphics) 1999. The program allows the user to extend the performance of the Chromatics in two primary areas: input capability and extension of graphics primitives.

Using Chromatics commands under program control, the interface suppresses keys which would kill the program while redefining some keys for new commands.

Coordinate input for graphics primitives is accepted in any of four modes:

- (1) from the keyboard as digit coordinates,
- (2) from the keyboard as cursor position.
- (3) from the tablet, positioning from the four-button cursor on the tablet,
- (4) from the tablet, positioning from the cursor on the screen.

The program expands the set of graphics primitives available. While built-in primitives such as circles and rectangles are available on the Chromatics itself, extended primitives such as thickened lines and arrows are accessed from disk using an overlay.

A Users' Guide is provided along with sample programs and diagrams which illustrate the use of the driver. Source code and details related to programming options give opportunity for modification to suit individual applications.

B. PAPER ORGANIZATION

Chapter one is an introduction which provides some background information and describes the general project effort, results and evaluation.

Chapter two is a Users' Guide for the resultant program. It is intended for the reader who desires to use the driver on the Chromatics color graphics microcomputer.

Chapter three provides details of the program development which would enable the reader either to duplicate the interface or modify it to expand the subset of graphics primitives, rescale the tablet, redefine the function keys or add other features not included in this project.

Appendices include the source code for the system driver and sample programs that support or expand upon various items in the text of chapters one, two and three.

C. BACKGROUND

In 1972 the Computer Science Department of Kansas State University acquired a Computek graphics terminal. It was a dumb terminal, black and white with 256 x 256 discrete points and required 8K of refresh memory.

By 1978 a Compucolor 8080-based terminal with eight colors, light pen, mini disks and 64K of memory was added. Its 160 x 190 discrete points required 15K of refresh memory; about 48K of memory was available to the user.

The further addition in summer, 1979, of a Chromatics Z80-based terminal with two disk drives, tablet, and high resolution from 512 x

512 discrete points expanded opportunities for more sophisticated work in color graphics. While the refresh memory alone required 131K, about 40K of memory was available for BASIC users.

Through the efforts of students in the fall 1979 KSU graphics class and the work of several other individuals, the capabilities and possibilities of the KSU graphics systems have become increasingly diversified. The past year has included development of demonstration programs, Chromatics hardcopy capability with an interface to the Perkin-Elmer minicomputer printer, and introductory users' guides for the Chromatics graphics functions. Current efforts include developing a comprehensive users' manual, attaching a camera, expanding the set of primitives available, working toward hard copy color graphics materials and developing editing capability for the create buffer.

This paper is a report on one additional step in the development of a comprehensive graphics system at KSU.

D. MOTIVATION

Human Engineering

The decision to undertake this particular project resulted primarily from short-range considerations. There were many respects in which the graphics system as it was did not provide a very "human" interface for users.

Documentation was scattered within more than four different manuals. In various instances the documentation was confusing, non-existent, or even inaccurate. Although production of a comprehensive, usable set of instructions for the whole Chromatics system was considered beyond the scope of this research, it was desired

to produce a guide which would simplify and organize the information necessary to successfully execute a program such as the one described in this report.

Use of the tablet was limited as there was inadequate software available to link the tablet to the full range of keyboard possibilities. Providing a smooth interface would allow users to input information from the tablet while intermixing the usual keyboard commands and data.

Some keys or input sequences would cause undesirable effects or even kill the program; these possibilities needed to be avoided while at the same time providing users the opportunity for some changes of mind in order to provide a tolerant environment.

Several improvements and additions to the system were hoped for. Separate work was in progress on routines to expand the basic set of graphics primitives; there needed to be some way to make these available to users. One of the most often used primitives, the circle, contained a "flaw" in one mode of inputting coordinates. A few changes would allow the user much greater accuracy and flexibility in defining circle size. Certain special function keys could be programmed for special effects to offer a greater range of easy-to-use features.

Learning and Extension

Very little existed in the way of documented programs which would encourage student experimentation and learning. The Computer Science Department could benefit from a model which could be "played with" and "added to" by students. Development of a driver for the Chromatics would provide another layer to the "onion" of software available for

such purposes.

Remote Teaching

One future possibility for use of graphics packages such as this one is in helping to provide visual aids for remote teaching. As features such as a camera, hard copy in color, etc., become available, demand for easy-to-use methods of producing quality graphics materials will increase.

E. DESCRIPTION OF PROJECT

Program Structure

The program, written in BASIC, consists of four parts (see Figure 1, page 6). The initialization in "Part 1" is executed only once; at its conclusion "Part 2", the Main Driver, is loaded. If the user requests extended graphics primitives (basic figures generated by special software routines) control moves to "Part 3" or "Part 4" which contain the routines for WIDE_VECTOR, SLANT_RECTANGLE, ARROW, and DOUBLE-HEADED_ARROW. These extended graphics routines were developed at Kansas State University by Maxine Yee (Yee80) and they are a significant contribution to the development of this report.

The main driver of the program is a simple interrupt handler (see Figure 2, page 6). Input is accepted from the keyboard one character at a time. A large case statement constructed of "if-then" statements checks the input and branches to the appropriate place in the program. The main driver handles state changes such as "blink" and "fill", color changes, and selects and sets up the primitives.

Coordinates for the primitives are gathered by a subroutine which

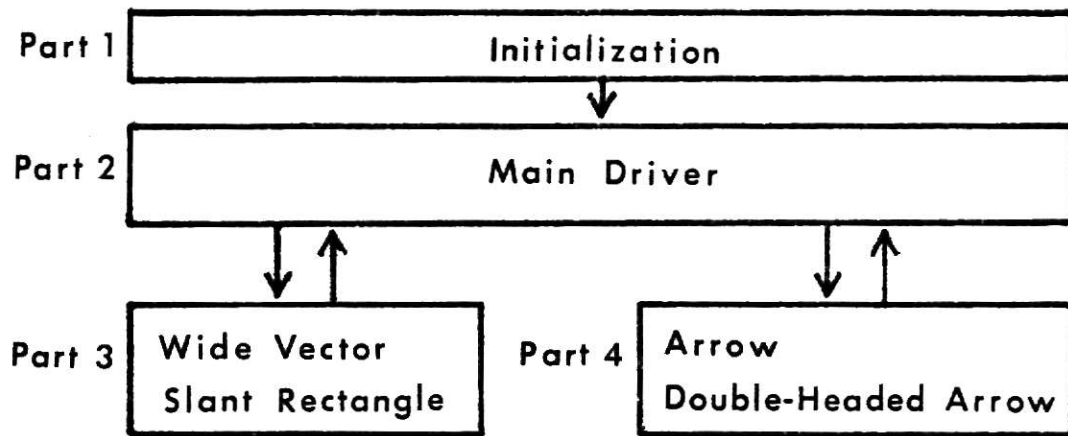


FIGURE 1.

CONTROL FLOW OF PROGRAM

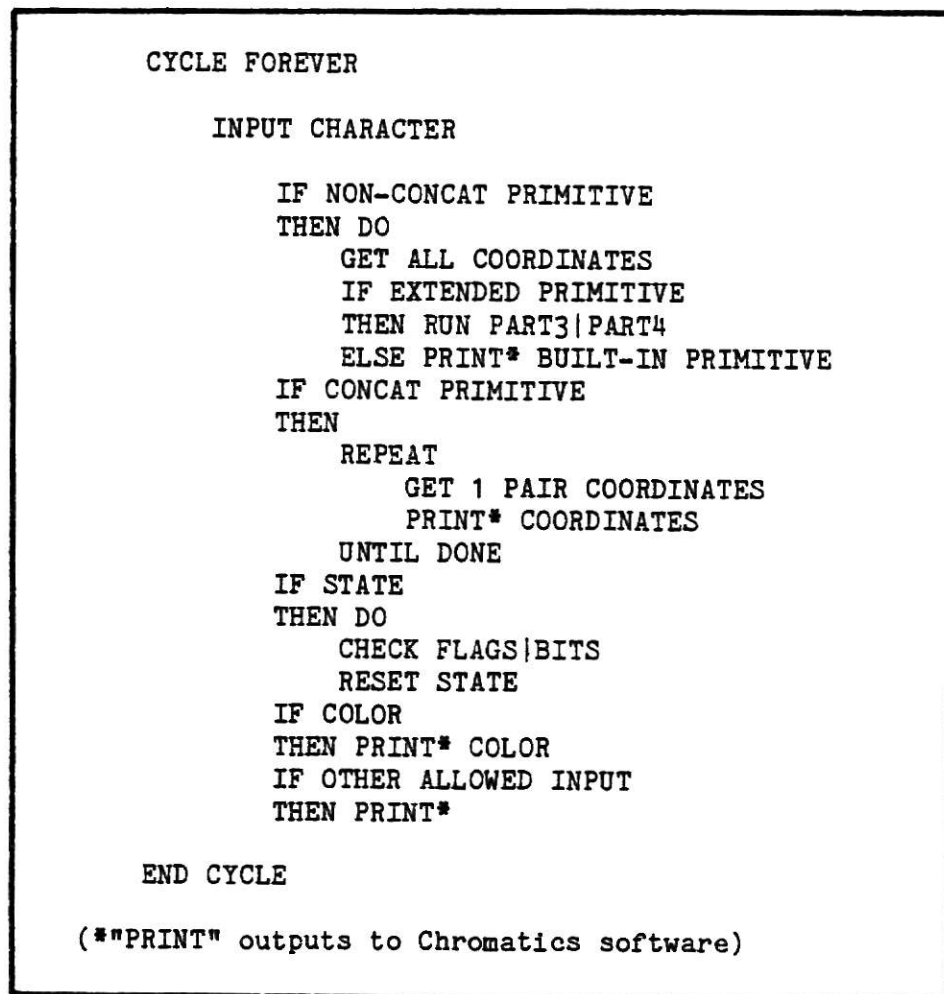


FIGURE 2.

LOGICAL STRUCTURE OF MAIN DRIVER

accepts input from the keyboard using the cursor position on the screen, from the keyboard using digits, from the tablet using the cursor position on the screen and from the tablet using positioning of the four-button cursor on the tablet.

The tablet is available on reprogrammed keys in either of two modes: stream mode or point mode. Stream mode inputs coordinates at the rate of 70 x-y pairs per second and facilitates tracking of the cursor on the screen as well as small increments in primitives such as the concatenated vector. Point mode inputs a single coordinate pair at a time. The position of the four-button cursor does not appear on the screen since the positioning in this mode needs to come from the tablet. Since only one point is input at a time, the chance of an error in coordinate selection due to holding the button down too long is eliminated.

A "create buffer" is available in the main driver. If the user selects this option the program itself handles turning the buffer off and on to eliminate the storage of unnecessary information (such as cursor movements) which might fill up the buffer space. Data in the create buffer may be redisplayed on command or saved as a file on disk.

Extensibility of the set of primitives is demonstrated with "Part 3" and "Part 4", overlay routines called by the main program (see Figure 3, page 8). Coordinates are fetched and stored in an array before the overlay is loaded; at the conclusion of an overlay routine the main driver is reloaded and the program continues. The relationship of this program to the underlying Chromatics software is illustrated in Figure 4, page 8.

Part 3

```

IF FILL THEN DO ROUTINES FOR FILL
IF DOTTED THEN DO ROUTINES FOR DOTTED
IF COMMAND = "VEC" THEN DO ROUTINES FOR WIDE VECTOR
IF COMMAND = "REC" THEN DO ROUTINES FOR RECTANGLE
RERUN PART 2

```

Part 4

```

IF FILL THEN DO ROUTINES FOR FILL
IF DOTTED THEN DO ROUTINES FOR DOTTED
IF COMMAND = "ARR"
    THEN IF HEAD = FIXED
        THEN DO FIXED-HEAD ARROW ROUTINES
        ELSE DO VARIABLE-HEAD ARROW ROUTINES
IF COMMAND = "DAR"
    THEN IF HEAD = FIXED
        THEN DO FIXED-HEAD DOUBLE-HEADED ARROW ROUTINES
        ELSE DO VARIABLE-HEAD DOUBLE-HEADED ARROW ROUTINES
RERUN PART 2

```

FIGURE 3.
OVERLAY ROUTINES

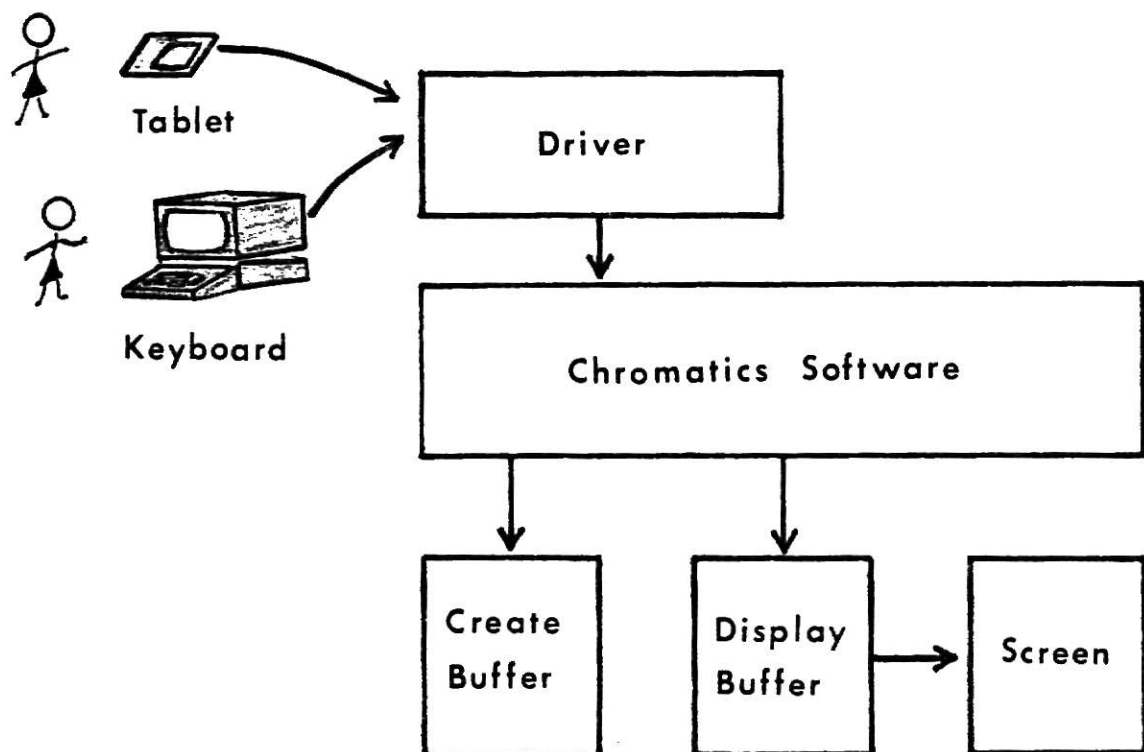


FIGURE 4.

RELATIONSHIP TO CHROMATICS SOFTWARE

Evaluation of Results

The primary objective was achieved in that the program provides an easy way to produce a graphics file. The user is afforded a reasonable set of primitives which can handle a wide range of applications. Many functions which previously required a sequence of instructions have been programmed to operate through use of a single key which has been redefined specifically for one purpose. For example, the eight function keys on the top right of the Chromatics keyboard combine the complex fill commands with characters from the alternate character set to give special shading effects. This feature could be particularly useful in maps or graphs.

A smooth transition is provided from tablet to keyboard, and vice-versa, with pre-set initialization simplifying use of the tablet. Switching from stream to point mode requires only a single key.

A relatively tolerant environment has been achieved. Input which kills the program appears to have been successfully "weeded out". Once a primitive has been designated, attempts to select a new primitive are ignored until the correct number of coordinates has been received and the original primitive has been printed. This helps ensure correct completion of primitives and prevents the program from stacking up a large number of subroutine calls for coordinates. However, users can redecide a "state" at any time, even in the midst of gathering coordinates, or even "cancel" the primitive in progress.

Although the program prevents the user from turning off plot mode while getting coordinates from the screen, the same precaution did not appear to work from the tablet. Therefore, plot control needed to be available when the tablet was enabled so that if plot was turned off,

the user could restore plot mode before continuing coordinate input.

Another difficulty became obvious when using the tablet for copying. The square tablet with equal X and Y distances between points caused distortion when figures were transferred to the rectangular screen having unequal X and Y distances.

Some optimization was done on the create buffer so that only the minimum set of instructions needed to reproduce the display was stored in the create buffer.

The program is composed of small, identifiable modules. Similar code is grouped together and duplication avoided where possible. Use of a single subroutine for coordinates channels all coordinate requests to the same place even though there are four options in the way coordinates may be gathered. The structure makes the code easy to comment, easy to read and easy to modify. This feature should enhance the objective relative to student learnings and extensibility of the program.

Learnings

Experience in developing the software for the project yielded lessons in: (1) the equipment, (2) the language, (3) graphics, (4) use of commercial documentation and (5) programming techniques.

Many aspects were "first-time" experiences: using a microcomputer, programming in BASIC, using graphics functions, using floppy disks, using a tablet, working with an alternate character set, designing an interrupt service routine, doing overlays and having to use source code as a mini-manual. "GO-TOs" had always been frowned on and had never really been necessary in previous programming experience. In BASIC, however, GO-TOs were essential but required effective use of block

structure and branching mechanisms. Logical to physical device assignments had to be made and rate codes set.

There was a heavy demand for experimenting and finding out what worked. Interactive debugging required special techniques such as inserting delays to allow time to monitor responses. Chromatics function keys had to be trapped to determine their special code on the machine so that their action could be redefined.

The 170 or so lines of code which contain the final product are too few to really be indicative of the learning which occurred.

II. USERS' GUIDE

A. THE MACHINE

1. The Screen

The CRT (cathode ray tube) screen (see sketch of terminal in Figure 5, page 13), is logically a rectangular grid of 512 x 512 points. It is helpful to image it as a coordinate graph where the X-axis goes from 0 to 511 and the Y-axis likewise goes from 0 to 511 (see Figure 6, page 13). Any pair of coordinates within this range is represented physically by some particular point on the screen, called a pixel (from "picture element"). The X coordinate represents the horizontal displacement from the lower left hand corner (the origin) and is always assumed to be the first element of the coordinate pair. The Y coordinate represents the vertical displacement up from the origin and is assumed to be the second element of the coordinate pair: (X,Y). If coordinate values higher than 511 are entered the result is the same as if these coordinates modulo 511 had been entered. For example: entering (700,800) would give the same result as entering (189,289).

The width and height of the CRT screen display area are not in the same ratio as the number of X dots and Y dots; five X dots approximately equal seven Y dots.

A window is logically a rectangular subset of the points displayed on the CRT screen. A single window defined as the full screen size is what the user sees. It is possible to define four overlapping windows (see USERS' GUIDE, Section E.4, for information on multiple windows).

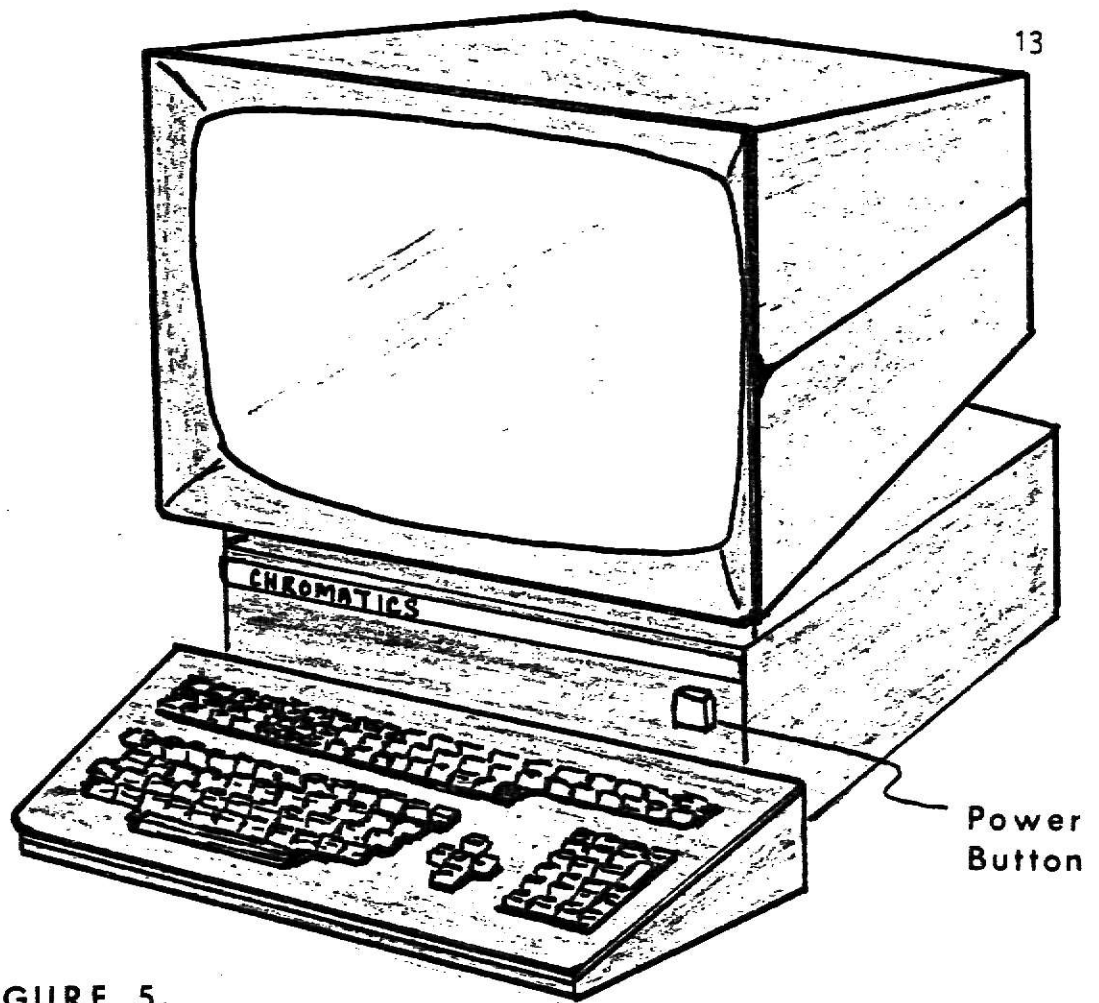


FIGURE 5.
THE CHROMATICS TERMINAL

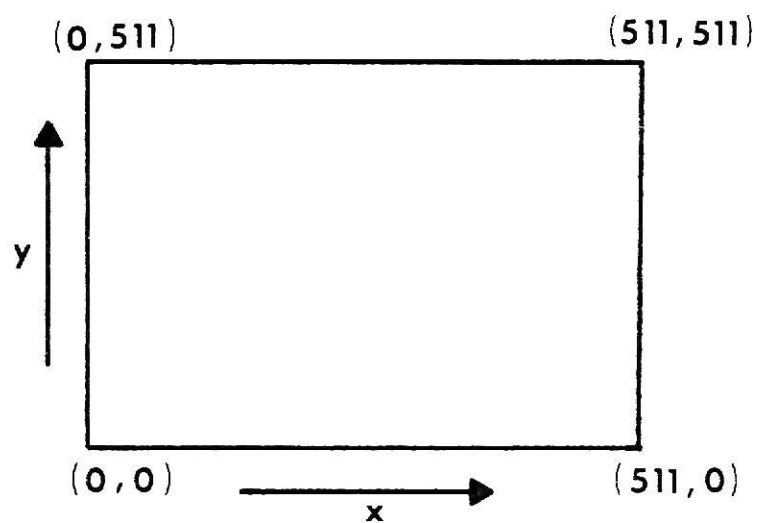


FIGURE 6.
THE DISPLAY SCREEN

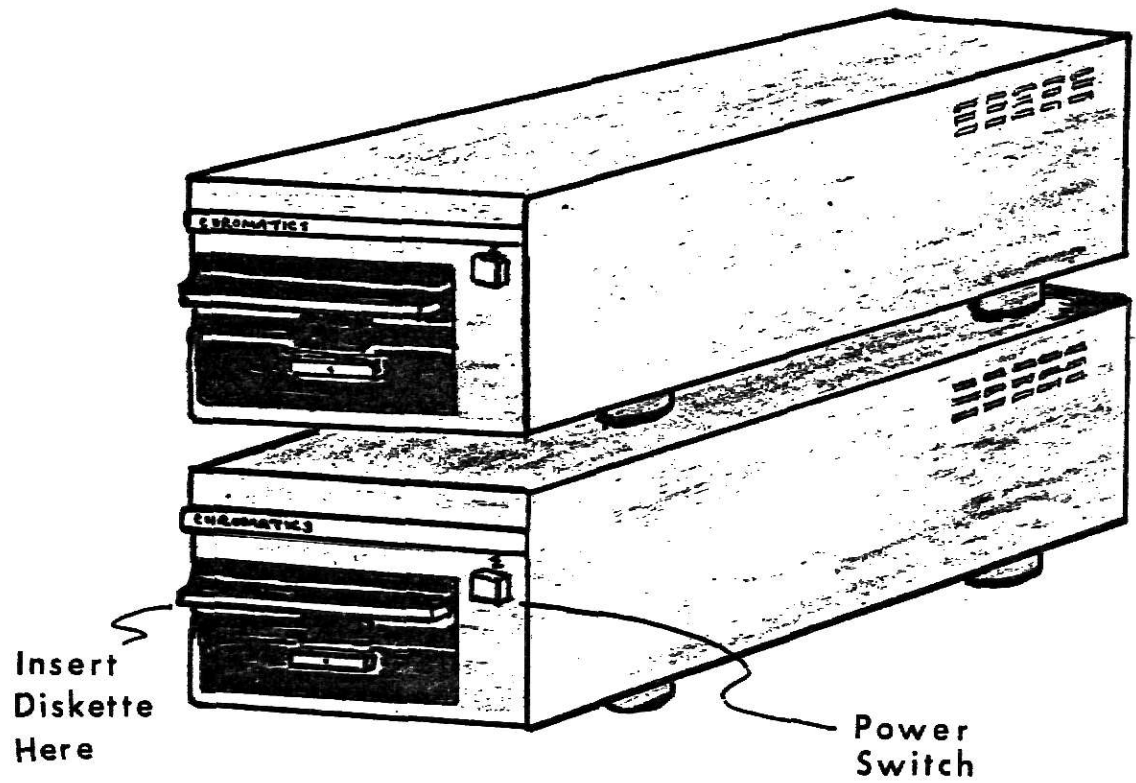


FIGURE 7.
CHROMATICS DISK DRIVES

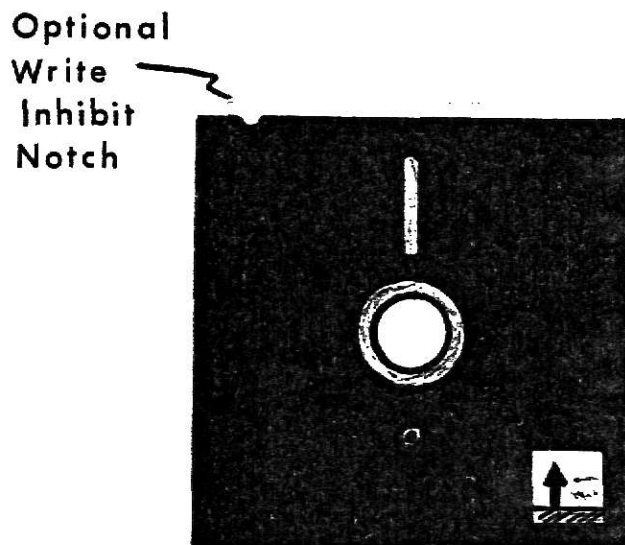


FIGURE 8.
DISKETTE

FIGURE 9.
THE TABLET

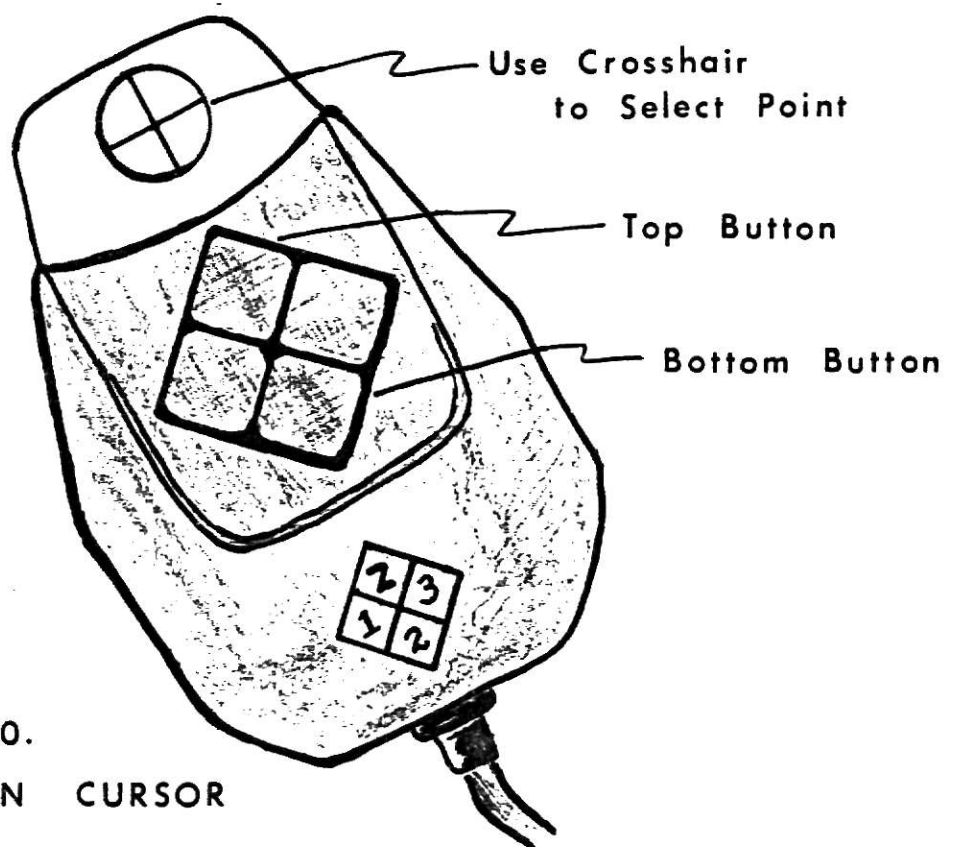
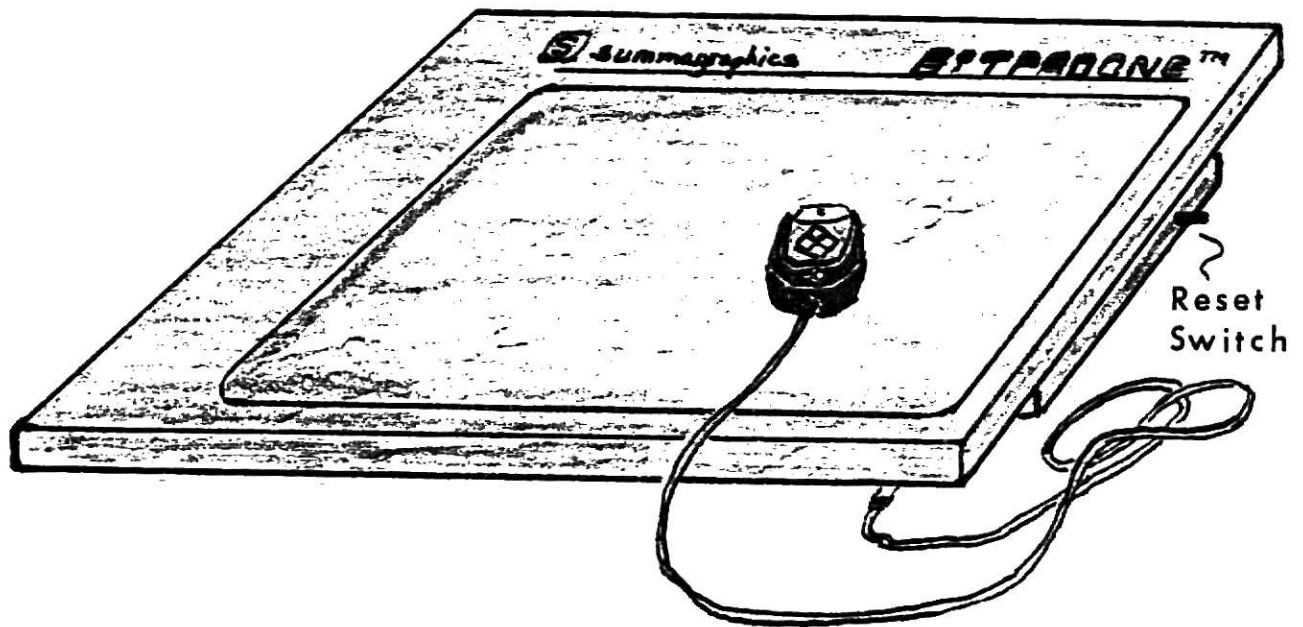


FIGURE 10.
4-BUTTON CURSOR

2. The Disks

The KSU Chromatics CG terminal has two disk drives attached (see Figure 7, page 14). Information is stored on eight-inch, single-density floppy disks (Figure 8, page 14) in concentric rings. New disks need to be formatted before use; check the Chromatics Disk Software Reference Manual (Chr78a), subsection 2.5.6, if you have an unformatted disk.

If the "create" option in this program is selected, graphics produced may be saved on disk as a buffer file. Sample Sessions in the appendix illustrate the saving of buffer files.

3. The Tablet

The tablet (see Figure 9, page 15) is conceptually similar to the screen described above. It is square in shape with raw coordinates ranging from (0,0) to (2200,2200). Here the X and Y distances are in a one to one ratio. Consequently, transferring pictures on a square tablet to a rectangular screen requires a scaling transfer. (For more information on scaling see page 57.)

The program documented in this report automatically maps the tablet onto the screen and coordinates may be input directly by using the 4-button cursor.

When the tablet is enabled, use the 4-button cursor by lining up the crosshairs over the desired point and pressing the topmost button for a "hit" (see Figure 10, page 15). When a point is selected on the tablet (and Point Mode is NOT enabled), a dot remains on the screen to mark the "hit". This "remembering" of points selected allows the user to place subsequent points in relationship to those already selected. If it is later desired to remove any extra dots, they may be "erased" by

lining up the cursor dot over the dot on the screen and overwriting with a dot of the background color.

The bottom button on the cursor is used to complete the selection of coordinates for certain primitives, such as the concatenated vector, where the desired number of coordinates is not known ahead of time. In addition, non-concatenated primitives may be "cancelled" by using the bottom button, which signals termination of the primitive. The other two buttons on the cursor are not used in this program and will give no effect when pressed.

B. THE PROGRAM

1. Initial Conditions

The program begins with an ERASE PAGE and sets up the following initial conditions:

- window size: full screen
- colors: foreground=green; background=black
- cursor: green, in home position
- BLINK off, ROLL off. PLOT on, FILL off,
- BACKGROUND off, CREATE off
- TABLET off, OVERSTRIKE off
- Character mode (not accessible until PLOT is off):
 - character size 1 by 1, standard character set,
 - horizontal mode

2. Loading and Running

- Insert disk into disk drive with label side up and with label

toward you, until the disk clicks and stays in place. Pull latch down to lock in the disk.

- Push "on" light on Chromatics and on disk drive, so that both lights appear lighted (refer to Figure 5, page 13, and Figure 7, page 14).
- Push RESET, BOOT/WIDE_VECTOR, BASIC/SLANT_RECT, and RETURN. The last item on the screen should be a prompt in green which says, "OK".
- Push the reset button on the right side of the tablet to reset it to initial states (refer to Figure 9, page 15). If you fail to do this you may get a "TYPE MISMATCH" error the first time you try to use the tablet.
- Type DOS"LOAD PART1"; push RETURN. The machine will load the program and return with the prompt "OK".
- Type RUN; push RETURN. Wait a minute to allow Part 1 to finish and load Part 2. (When the red light on the disk drive goes out the loading is complete.) The machine will begin execution of the program and set initial conditions: the "plot key" should be lighted and all others unlighted while the cursor for plot, ".__", should appear at the upper left of the screen in home position.

You are now ready to enter commands. Select commands from the Key Summary (Figure 11, pages 20-22) or consult the remaining sections of this chapter for additional information on how the various keys work. Also, the Sample Sessions in the Appendix demonstrate many of the commands. When finished with your session simply push the BREAK key and, if desired, save your file using DOS commands.

Extensions for editing, saving and retrieving display files are

currently being developed by Sharlene Mitchell.

NOTE: The remainder of this chapter contains a condensed, reorganized and corrected version of portions of the Chromatics Preliminary Operator's Manual. Information useful for this program has been included while unnecessary sections or details have been omitted.

KEY SUMMARY

20

KEY	SUMMARY	PAGE REF
RESET	Gains control from program	34
BOOT/wide vect*	BOOT: initializes system devices, etc./ wide vect: selects primitive	34 36
TEST/append*	Accomplished in this program with MODE T sequence/ append: appends to create buffer	31 39
TEXT EDIT/tablet*	(Not used in this program)/ tablet: selects tablet for coord input	27
ASMB/arrow*	(Not used in this program)/ arrow: selects primitive	36
PROM PGMR/point mode*	(Not used in this program)/ point mode: selects point mode as tablet submode	28
BASIC/slant rect*	Selects BASIC language interpreter and initializes memory for BASIC/ slant rect: selects primitive	18 36
COPY	(Not used in this program)	
F1-F8/F1-F8*	(Not available in this program as originally designed)/ Reprogrammed for special shading effects using complex fill	59 58
CPU OS	(Not used in this program)	
DISK OS/d-arrow*	Disk operating system/ d-arrow: selects primitive	34 36
CRT OS	(Not used in this program)	
CREATE	Sets up special buffer for files	39
ZOOM	Selects portion of window to enlarge	40
REDRAW	Maps contents of create buffer to window	39
XMIT	(Not used in this program)	
WINDOW	Allows defining of multiple windows	41
CURSOR X-Y	Moves cursor to indicated coords	34

FIGURE 11.

KEY	SUMMARY	PAGE REF
PLOT	Turns on plot; necessary for primitives	28
DOT	Selects primitive	36
X BAR	Selects primitive	36
Y BAR/concat vec*	(Use the " to select Y-BAR primitive)/ concat vec: selects primitive	36 36
VECTOR	Selects primitive	36
RECT	Selects primitive	36
CIRCLE	Selects primitive	36
ROLL	Scrolls window up or down	28
BLINK	Sets mode to blink; may be used for foreground and/or background color	26
BACKGROUND	Future color commands affect background	26
FILL	Primitives appear in solid color	27
BLACK - WHITE	Selects colors	26
MODE	Establishes operating mode of a window (there are many options)	29
$\bar{\Lambda}$	Λ = operator which squares functions ~ = mode code (same as above)	29
ERASE PAGE	Clears window	33
ESC	For high level control functions	33, 39
L F	Line Feed; moves cursor to line below	33
RETURN	Carriage return	
CIRL B	Selects fixed head option on arrow	32
CIRL C	Cancel current primitive	32
CIRL D	Selects dotted state for primitives	27, 32
CIRL F	Deletes character	32
CIRL G	"Bell"	

KEY	SUMMARY	PAGE REF
CTRL I	Tabulation	32
CTRL N	Selects alternate char set	33
CTRL O	Turns off alternate char set	33
CTRL W	Inserts character	32
CTRL X	End of Record (EOR) mark	39
CTRL A	End of File (EOF) mark	32
BREAK	Leaves BASIC program	33
SHIFT	Selects alternate char sets; allows cursor to move 1 pixel	26
SHIFT CTRL /	Underline	26
SHIFT CTRL O	Delete	26
SPACE BAR	Inserts space	
HOME	Moves cursor to first position of top line	34
↑,→,↓,←	Moves cursor one character space (or one pixel with SHIFT) in direction indicated by arrow	
Note: *indicates options used under this program		

C. THE KEYBOARD AND THE COMMANDS

1. Introduction to the Keyboard

The keyboard is the primary input device for the terminal. If a key is held for about 3/4-ths of a second, the key is repeated.

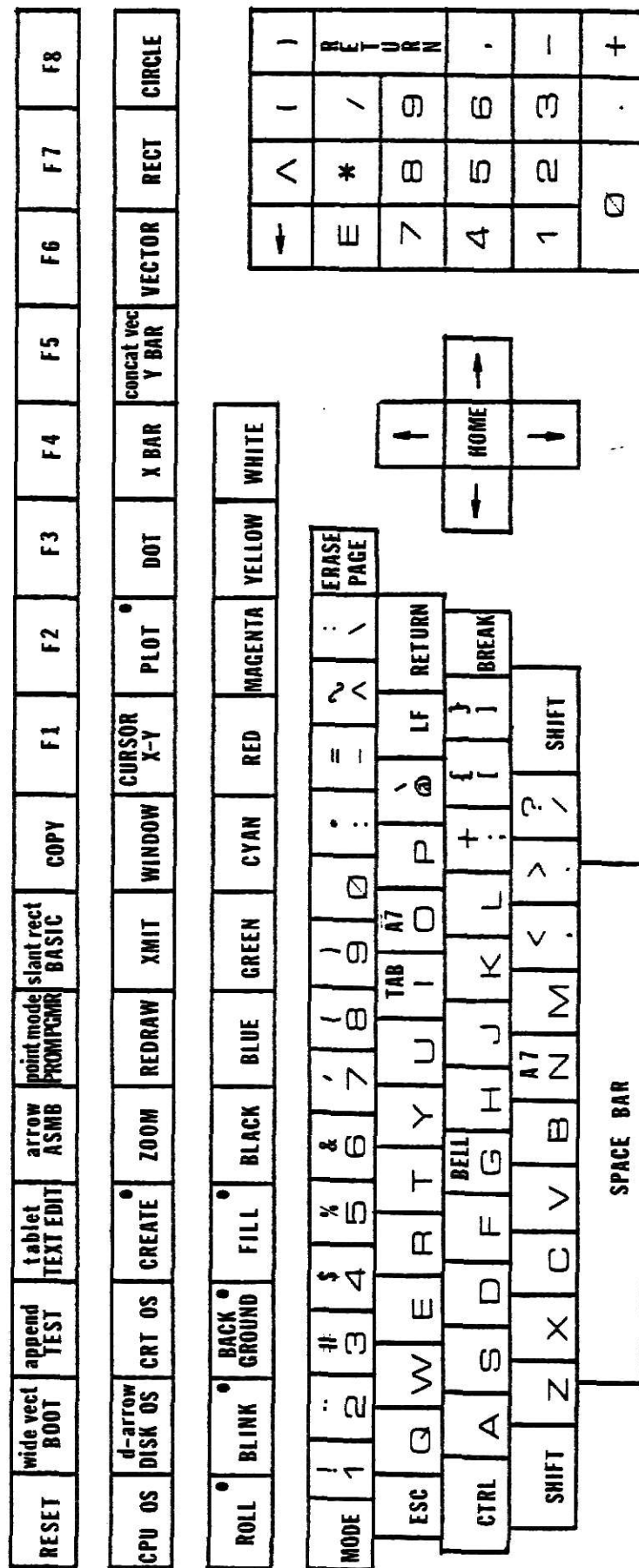
There are four groups of keys on the 128-key keyboard (see Figure 12, page 24). The upper three rows are special function keys, used for functions specifically defined for the Chromatics CG series or redefined for this particular program. The numeric keypad is the block of keys on the lower right; it is arranged in adding machine format to allow easy entry of numeric information and equations. The group of five keys to the left of the numeric keypad is the cursor control, which allows for easy positioning of the cursor anywhere on the screen. The final group of keys is the basic keyboard, which is the typewriter-like block of keys on the lower left.

Since the program requires a significant amount of checking on each input character, typing too fast may cause some input to be skipped--be sure your input rate is acceptable.

2. Redefined Keys

Several of the special function keys have been redefined for special uses in this program. (Redefined keys appear with the new definitions in lower case letters above the original key definitions in Figure 12.) The user may wish to tape over those keys and mark them with the new symbols. The new keys and their actions are given in the table in Figure 13, page 25.

KEYBOARD LAYOUT



- indicates an illuminated key
- lower case lettering indicates programmed driver changes
- UPPER case lettering indicates CHROMATICS names

FIGURE 12.

REDEFINED KEYS			
OLD KEY	NEW KEY	ACTION	TO NEGATE ACTION
TEXT EDIT	TABLET	State key; tells the program to accept stream mode coordinates from the tablet instead of keyboard coordinates	Push TABLET
PROM PGMR	POINT MODE	State key; indicates a switch from stream mode to point mode, which accepts only one pair of X-Y coords at a time; only effective if TABLET is already on	Push POINT MODE
CTRL B	(none)	State key; sets "fixed" as attribute for arrow heads by assigning AH\$="FIX"	(none)
CTRL C	(none)	CANCEL key; cancels last primitive if not on tablet (bottom button=tablet cancel)	(none)
CTRL D	(none)	State key; sets dotted line as option for wide vectors and arrows	(none)
Y BAR	CONCAT VEC	Primitive; sets code for successive vectors joined at their endpoints; only effective in plot mode	If tablet mode, bottom button; if screen push CONCAT VEC
BOOT	WIDE VECT	Primitive; draws wide lines	(none)
ASMB	ARROW	Primitive; draws arrows	(none)
DISK OS	D-ARROW	Primitive; draws double-headed arrows	(none)
BASIC	SLANT RECT	Primitive; draws rectangle in any orientation (as opposed to built-in rect which uses only vertical and horizontal lines)	(none)
F1-F8	F1-F8	Complex fill; these now provide 8 shading effects	(none)
TEST	APPEND	Appends to create buffer	(none)

FIGURE 13.

3. Shift

Shift provides access to two alternate character sets: lower case for the basic keyboard and an alternate character set (as shown in Figure 15b, page 35), if the command for the alternate character set has been issued. (See Section D.2 on CHARACTER MODE.)

Shift in combination with the cursor control keys moves the cursor only one dot (pixel) in the direction indicated instead of the usual six dots left/right or ten dots up/down.

Shift also has a special use in combination with control for two codes, underline and delete in which SHIFT must be held down while pressing CONTROL. The sequence is:

```
SHIFT  CTRL  / = _ (underline)
SHIFT  CTRL  O = DEL (delete symbol,
                    a non-printing symbol)
```

4. States

States define the status of a window.

Colors - Colors are set by pushing the color keys.

Background - The Background command sets background mode on and lights the BACKGROUND key. Future color commands will affect the background color. Pushing the key a second time sets background mode off so that future color commands will now affect the foreground color.

Blink - The Blink command lights the BLINK key and sets the screen to foreground blink (or background blink if the BACKGROUND key is lighted). Pushing the key a second time turns off the light and resets to non-blink mode. The following set of

commands provides an example of the variations possible using BLINK:

```

BOOT  MODE X 5,  MODE Y 5,  BACKGROUND
BLUE  ERASE PAGE  CYAN  BACKGROUND(off)
RED   -->  -->  ↓  ↓  F  BLINK  -->
F     BACKGROUND  -->  BACKGROUND(off)
BLINK(off)  -->  F

```

When entered correctly, all four possibilities for blinking should be displayed: the first character should be a large red "F" on a cyan block; the second character is a blinking, red "F"; in the third character both the "F" and the background blink; and in the fourth character only the background blinks while the letter "F" is stationary.

Fill - The Fill command lights the FILL key and subsequent primitives will appear in solid color (all interior dots lighted) rather than in outline form. Pushing the FILL key a second time turns off the light and resets the state to non-fill.

Dotted Line - The Dotted Line command is available as a state option for wide vectors and arrows. If the CTRL D sequence is pushed and the next primitive to be printed is a wide vector or an arrow, the primitive will appear with dotted (dashed) lines. This state, or attribute, needs to be reselected each time it is to be used.

Tablet - Use of the Tablet command changes coordinate input for primitives from the keyboard as a source to the tablet. In stream mode (the default tablet submode), the coordinate

position of the cursor is continually sent to the screen so that the user may follow the movement of the cursor. However, if the top button on the cursor is pressed, a "hit" is recorded and the coordinates at the position of the hit are selected. Care must be taken in doing selection; if the cursor is moved even slightly during a "hit", additional coordinates are likely to be selected.

Point Mode - This tablet submode may be selected by pushing the POINT MODE key. The cursor position is no longer tracked on the screen and one pair of coordinates is selected at a time by pushing the top button. Point mode is less risky than stream mode since a "hit" is guaranteed to pick up only one pair of coordinates. This submode is useful when a figure is to be copied: simply tape your paper securely to the tablet itself and select the appropriate primitives. An illustration of the use of POINT MODE is presented in Sample Session 2 in the Appendix.

ROLL - When the Roll key is lighted the information in the window is automatically scrolled up (or down) as the cursor moves past the bottom (or top) of the window. If the user is entering text this is particularly useful. For most graphics applications, however, it is dangerous in that a roll state could lose the contents of the topmost (or bottommost) lines of the window; therefore this program initializes the roll state to "off" and it is recommended that the user only enable roll with caution.

Plot - All graphics primitives become active in PLOT mode. When

the machine is in Plot state, the PLOT key light is on and the cursor is displayed as a blinking dot followed by a line. This form of the cursor reminds the user of the current state and enables selection of a particular pixel.

Pushing the PLOT key so that the light is turned off returns the machine to Character Mode and also acts as a mode cancel by ignoring any partially entered mode commands.

5. Mode Codes

Mode code functions are used to establish the operating mode of a window. If no windows have been defined, the mode codes describe the whole screen which is the window #0. The more common mode codes follow:

Character Size - The character size can be expanded an integer number of times in either the horizontal or vertical directions. The format is

MODE X num1, MODE Y num2.

Keying "MODE X2, MODE Y3," for example, changes the dimensions of the characters displayed to twice the width and three times the height. The maximum value of num1 is 85 and the maximum value of num2 is 51.

Example key sequence:

A B C RETURN LF LF

MODE Y 3, A B C RETURN LF LF

MODE X 3. A B C

When keyed correctly, this sequence should produce three lines with "ABC" displayed. The first line is normal sized, the second tall and narrow and the third, normal characters three

times larger in both dimensions. To return to normal character size simply key in "MODE X1, MODE Y1,".

Vertical Display of Characters - Occasionally it is convenient to display the characters vertically down the page. "MODE V" sets for vertical mode.

Example key sequence:

A B C MODE V A B C

This should result in: ABCA
 B
 C

To return to horizontal mode simply key in "MODE H".

Interline Spacing - Normally, each line of characters is directly adjacent to the line of characters above it. "MODE A num" sets the number of points between lines of characters to num (range 0 to 255). The default value of num is 0, so "MODE A 0" resets interline spacing to normal.

Note that the lines of points between characters are not written, (set to foreground or background color), when characters are sent with non-zero interline spacing. An ERASE PAGE command, however, will always set all window lines to background color.

Cursor Display Control - The cursor may be made invisible on the screen by keying "MODE K". However, the cursor still exists and will be moved by the appropriate commands. To make the cursor visible and blinking again, enter "MODE J". To change the cursor color enter "MODE Q colnum" where colnum is 0|1|2|3|4|5|6|7 and the color numbers match the order of color keys on the keyboard moving from left to right.

Erase Line - The "MODE @" command causes the current line to be cleared to the background color; the cursor is not moved.

Erase to End of Line - The "MODE 3" command causes the remainder of the current line from the cursor to the right hand edge of the window to be erased to background color; the cursor is not moved.

Overstrike - The command "MODE 0" (notice this is 0 and not zero!) places the window in overstrike mode for the next display character. Only the foreground dots of the character are written; this allows multiple characters to be overlapped in the same location. The command "MODE]" places the window in overstrike mode for all subsequent characters until overstrike mode is turned off. This function is useful in underlining as well as in special applications. "MODE [" places the window in normal (not overstrike) mode. An example key sequence is given:

```

MODE X 3, MODE Y 3, --> --> -->  ↓  ↓  ↓
BACKGROUND BLUE BACKGROUND(off)
C H R O M A T I C S RETURN --> --> -->
RED MODE ] SHIFT CTRL / SHIFT CTRL /
SHIFT CTRL / SHIFT CTRL / SHIFT CTRL /
SHIFT CTRL / SHIFT CTRL / SHIFT CTRL /
SHIFT CTRL / MODE [

```

Test - The command "MODE T char" causes an ERASE PAGE followed by the filling of the window with the indicated character (char), which cannot be a control character. (See the first example

in section E.4 on Windowing.)

Delay - The command "MODE ? num", where num is the duration of the delay in tenths of a second and must be three digits long, causes a delay in execution. (Example: if num = 050 the delay is 5 seconds.) This command could be useful in animation sequences as any delay from 0 up to 99.9 seconds is possible. However, if this command is put into the create buffer, expect to wait it out!

6. Control Codes

Most control functions can be keyed using the special function keys. The following functions can only be keyed using the CTRL key:

CTRL A = end of file (EOF)

CTRL W = insert character

CTRL F = delete character

CTRL B = fixed-head option for arrow or double-headed arrow

CTRL C = cancel last primitive

(if entering tablet coordinates, use bottom button
on cursor to cancel the last primitive)

CTRL D = dotted line mode for extended primitives

TABULATION is set up on multiples of eight and is accomplished by keying "CTRL I". The cursor is then moved to the next character position on the current line which has a number divisible by eight, where the numbers are assigned left to right beginning with the first character position. If this action would move the cursor outside the window, the cursor is moved to the first character position on the next line; if the next line would be outside the window, the cursor is moved

to the Home position.

Another significant use of the CTRL key includes access to the alternate character sets. Pushing "CTRL N" turns on a signal so that all characters are taken from the alternate character set (see Figures 15a and 15b, pages 37 and 38). The command "CTRL O" turns the signal off, (the normal condition), and causes all characters to be taken from the standard set (as shown in Figure 12).

7. Escape Codes

Escape code functions are used primarily for high level control functions such as making logical to physical device assignments and transferring control between various main programs. The only application of escape codes to this program is windowing, discussed in Section E.4, and escape codes used with the create buffer, discussed in in Section E.1.

8. Miscellaneous Keys

Keys not previously explained are noted below:

ERASE PAGE - This key clears the window.

BREAK - The only action taken for this key is a break in an output line created by a timing delay. The system will leave the BASIC program and allow the user to do editing and enter commands. (For an illustration of using this key to leave the program and enter commands to save the create buffer, see Sample Session 1, Part III.)

LF - Line feed moves the cursor to the position immediately below the current one. This command is identical to the down arrow,

"↓".

HOME - The cursor is moved to the first character position of the top line.

CURSOR X-Y - The cursor is moved directly to the coordinates indicated:

CURSOR X-Y coord coord

where "coord" represents a three-digit number. This command ignores window boundaries and should be used with care.

RESET - The RESET key is different from all other keys on the keyboard. It is a system control function which gains control from a program without losing internal information. It returns the system to the basic operating program supplied with the standard system.

BOOT - This key is like RESET but it additionally causes all devices and tables in the system to be initialized to default conditions. Sometimes a RESET preceeding the BOOT is necessary for proper operation.

DISK OS - The DOS (Disk Operating System) allows the user to manipulate data to and from the floppy disk drives. DISK OS (DOS) prompts the user with an asterisk (*) to indicate that a disk command is expected. Some of the more useful DOS commands are:

- DIR - Type DIR, push RETURN, and the directory of files on your disk is displayed.
- BUFF - Type BUFF filename, push RETURN, and the create buffer is saved as a buffer file on your disk.
- DRAW - Type DRAW filename, push RETURN; when the prompt

(*) reappears the file is loaded. Push REDRAW and the file is displayed on the screen.

The DOS commands may also be executed under BASIC by typing DOS" command and then pushing RETURN. See the examples in the Sample Sessions or consult (Chro78a) for more information.

D. THE PRIMITIVES

1. Graphics Mode

The intrinsic primitives (built-in functions for generating graphics figures) supported by this program are DOT, VECTOR, RECTANGLE, CIRCLE, CONCATENATED VECTOR , Y-BAR and X-BAR. Extended primitives (those using software routines for generating the figures) demonstrated in this program are WIDE_VECTOR, SLANT_RECTANGLE, ARROW and DOUBLE-HEADED_ARROW.

The primitives, the information required to produce them and examples of results are shown in Figure 14, page 36. Note the position relationships between the coordinates and the final figures.

The plot mode, or state, must first be enabled for graphics primitives to print. When in plot mode the cursor is displayed as a blinking dot (representing the pixel location) followed by a line, ".__".

2. Character Mode

Characters entered will be displayed on the screen in the position indicated by the cursor (blinking lines indicating the top and bottom of the character position). A character position is ten dots high and six

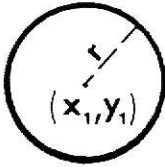

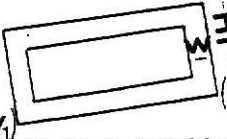
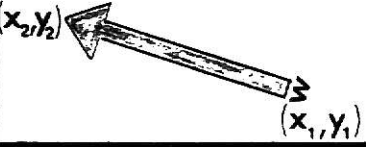
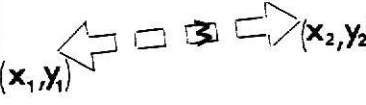
PRIMITIVE	KEYS, COORDINATES & INFORMATION REQUIRED	RESULTING FIGURE
DOT	DOT X1 Y1	$\cdot (x_1, y_1)$
VECTOR	VECTOR X1 Y1 X2 Y2	$(x_1, y_1) \xrightarrow{\quad} (x_2, y_2)$
RECTANGLE	RECT X1 Y1 X2 Y2	$(x_1, y_1) \boxed{\quad} (x_2, y_2)$
CIRCLE	CIRCLE X1 Y1 R (R may be input with digits or cursor positions & hits)	(x_1, y_1) 
CONCATENATED VECTOR	concat vec X1 Y1 X2 Y2 X3 Y3 . . . Xn Yn	$(x_1, y_1) \xrightarrow{\quad} (x_2, y_2) \xrightarrow{\quad} (x_3, y_3)$ (x_4, y_4)
X-BAR	X-BAR X1 Y1 X2	$(x_1, y_1) \text{---} (x_2)$
Y-BAR	" X1 Y1 Y2	(x_1, y_1) (y_2)
WIDE VECTOR (DOTTED) (FILLED)	wide vect X1 Y1 X2 Y2 W (CTRL D) (0 1 2 3 4 5 6 7 for color#)	$(x_1, y_1) \xrightarrow{\quad} (x_2, y_2)$ 
SLANT RECT	slant rect X1 Y1 X2 Y2 HT, W (and color # if FILL on)	$(x_1, y_1) \boxed{\quad} (x_2, y_2)$ 
ARROW (var head) (fixed head)	arrow X1 Y1 X2 Y2 W (color # if FILL on) (Default: head = 1/10 length) (CTRL B; head is 10 pixels)	$(x_1, y_1) \xrightarrow{\quad} (x_2, y_2)$ 
DOUBLE-HEADED ARROW	d-arrow X1 Y1 X2 Y2 W (color # if FILL on) (var and fixed head options)	$(x_1, y_1) \xleftrightarrow{\quad} (x_2, y_2)$ 
Notes: PLOT must be enabled for all primitives. R, HT and W may be input as digits or as cursor positions and "hits". Options such as BLINK, FILL, DOTTED and fixed head should be selected in advance of the primitive selection.		

FIGURE 14.
THE PRIMITIVES

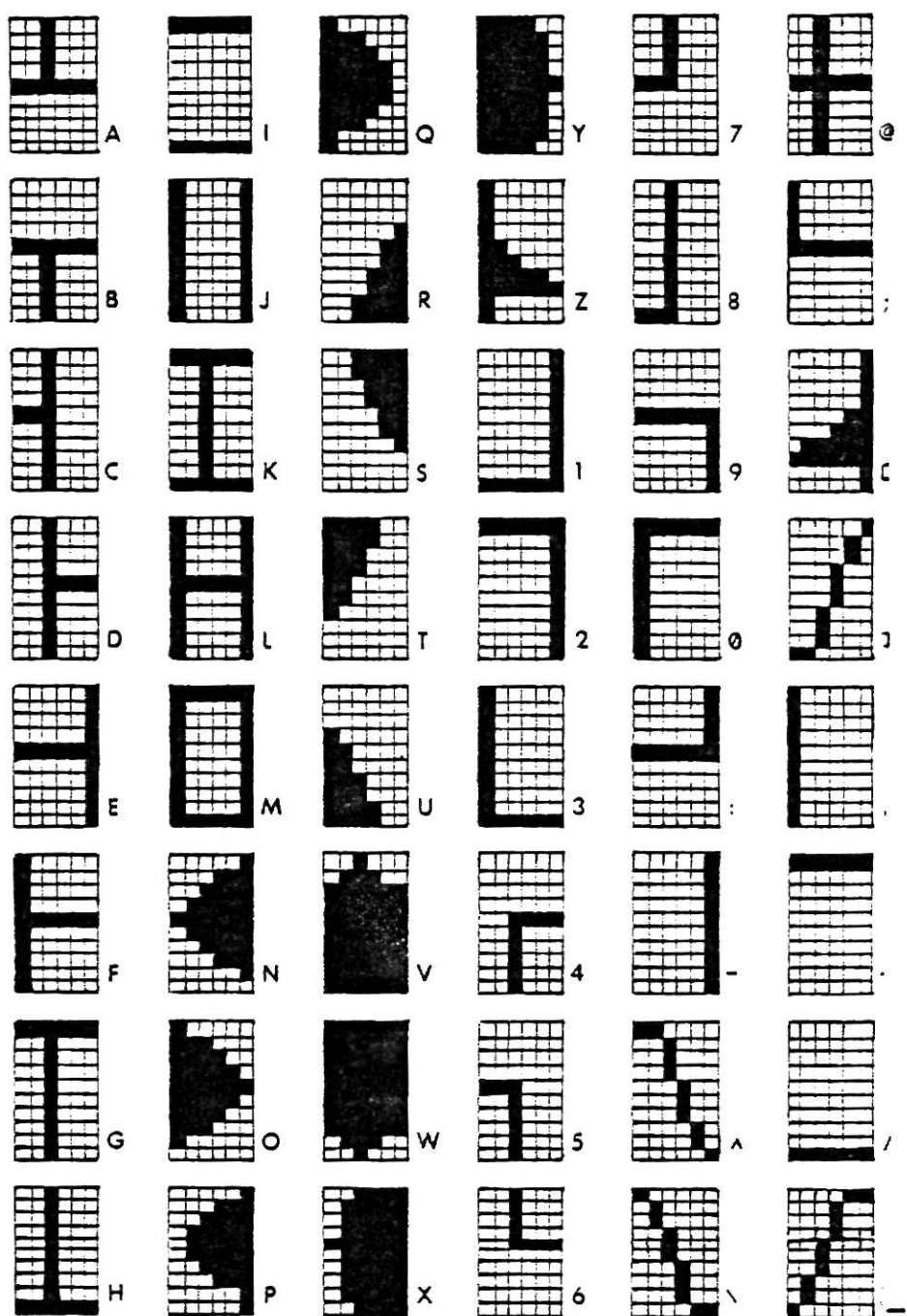


FIGURE 15a.
UPPER CASE, ALTERNATE-CHARACTER-SET

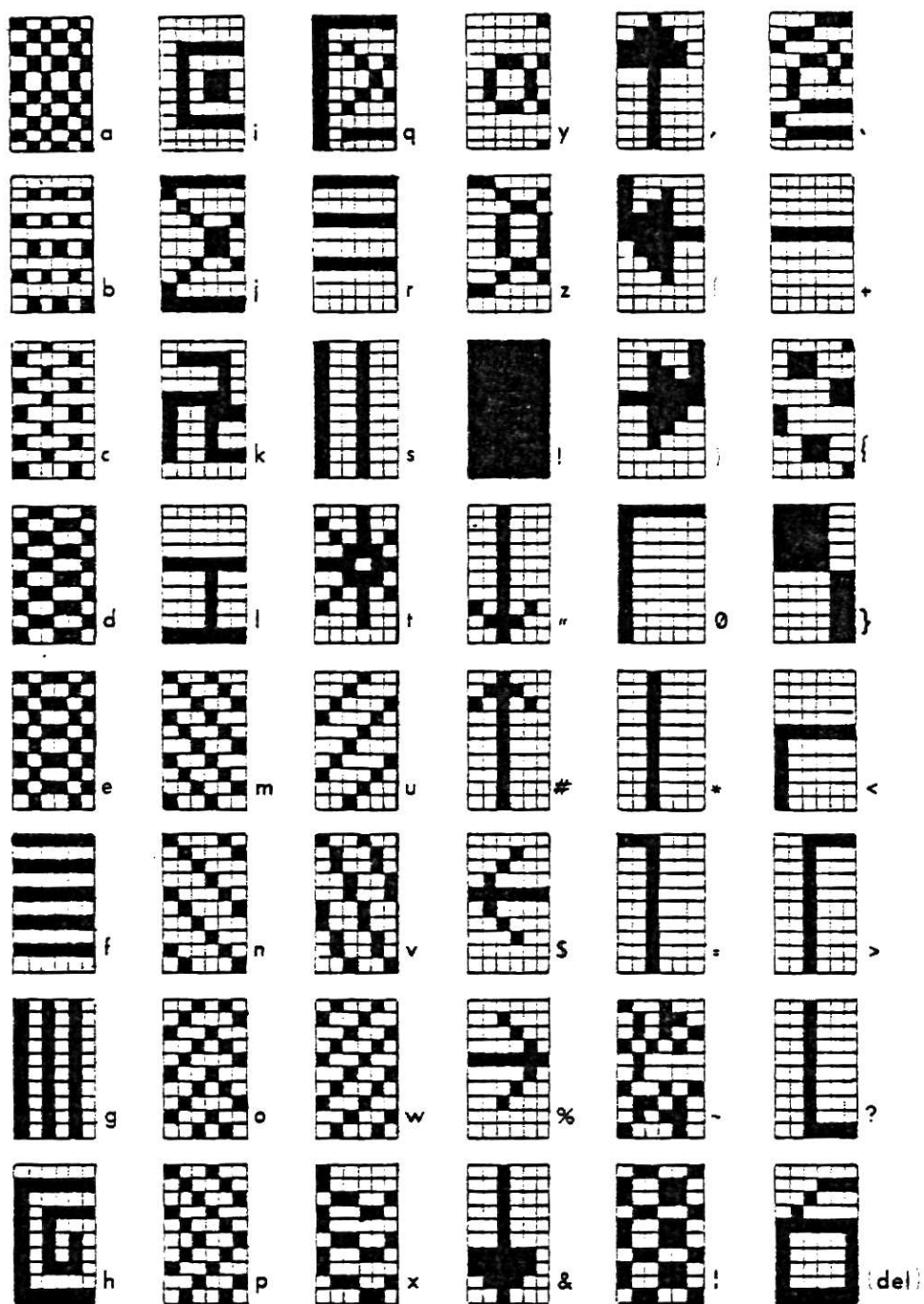


FIGURE 15b.

SHIFT, ALTERNATE - CHARACTER - SET

dots wide and includes inter-character and interline spacing.

Any of four display character sets of 96 characters each can easily be displayed on the terminal. (See Figures 12, 15a and 15b.) Note that Figure 12 represents two different character sets: upper case and lower case standard characters.

E. OTHER FUNCTIONS AND COMMANDS

1. Create Buffer

Use of the CREATE key sets up a buffer which stores all subsequent characters used for displays as they are entered into the terminal. Multiple subbuffers (up to 255) may be created by separating the data with an end of record (EOR) character, CTRL X. Hitting the CREATE key when the light is on will terminate the function and cause the light to go out. The command "ESC K hnum" kills only a portion of the buffer: the data between EOR marks hnum and hnum + 1 is erased and the following data is compressed, leaving additional room at the end of the buffer. (hnum is always entered in HEX and can be any number from 00 to FF.)

If it is later desired to add to the create buffer, the APPEND key may be used. Subsequent characters are added to the end of the buffer rather than at the first location. The function is terminated as above.

The REDRAW key maps the contents of the buffer to a window, usually the full screen unless windows have been redefined. There is a command to view only a portion of the create buffer: "ESC V hnum" works like REDRAW except that only the data between EOR marks hnum and hnum + 1 is sent to the display screen.

2. Zoom

The ZOOM key allows the user to select any rectangular subset of the screen and enlarge it to the full window area. The original contents of the window are destroyed by this function unless CREATE and REDRAW are used to restore the old contents.

Example:

```

CREATE  ERASE PAGE  RED  PLOT  RECT  .
--> (key 12 right arrows)  ↓ (key 3 down arrows)
.  RETURN  ↓  ↓  -->  PLOT(off)  CYAN
C  H  R  O  M  A  T  I  C  S
HOME  CREATE(off)  ZOOM  .  --> (12 arrows)
↓ (3 arrows)  .  REDRAW

```

3. Complex Fill

This function is useful for filling in polygons or producing special shading effects. A two-tone fill is possible by using both a background

fill color for the solid portion and a foreground fill color for the pattern. Although this command is a bit complicated the effects possible are a nice addition to the capabilities of the system; it is worth some effort to understand the requirements:

- Be sure ploygon border is completely closed,
- Polygon border must "contain" the foreground fill color,
- Place cursor inside polygon to be filled,
- Set background NOT black,
- Set foreground fill color for character or pattern,
- Key in "MODE > char" where char is any character.

(Or use the preprogrammed F1 through F8 keys)

If "char" is a blank, the polygon fills solidly with the background color only. To determine which colors are "contained" within another color, some experimenting may be helpful. There are three primary color planes: red, green and blue. Yellow contains red and green; magenta contains red and blue; cyan contains green and blue; white contains all three color planes. Black is the absence of any of the three color planes. (Note: the instructions for Complex Fill in Chro78b, Sect 4.1.8, are in error.) Sample Session 3 in the Appendix illustrates use of Complex Fill, and notes the importance of registering cursor position if the create buffer is used. The CURSOR X-Y function key is sometimes useful for this.

4. Windowing

The WINDOW key allows defining of up to four windows: #0, #1, #2, #3. The following key sequence defines window #0 to be the upper left quadrant of the screen and fills the window with +'s:

```
WINDOW 000 511 255 255  MODE T +
```

In order to display to other windows, they must be assigned to the appropriate logical output devices. For example:

WINDOW 000 511 255 255	window #0=upper left
BACKGROUND GREEN ERASE PAGE	make window visible
ESC O A 1	set up window #1
WINDOW 256 511 511 256	window #1=upper right
MODE J	make cursor visible
RED A B C	write to window #1

Note: assigning a window does not affect the status of the lights on

the illuminated keys; therefore, the lights on the keys do not indicate the mode of the window but instead reflect the meaning of the key. Although the BACKGROUND key is illuminated in the example above, the final line does NOT set the background color to red, but rather the foreground color. Striking the BACKGROUND key will realign it with the mode of the window. To change the background color of window #1, the BACKGROUND key must be struck twice.

Continued example:

```
BACKGROUND(off) BACKGROUND RED ERASE PAGE
BACKGROUND(off) BLACK foreground w #1=black
MODE X 3, MODE Y 3, char size = 3 x 3
A B C write to window #1
```

Window #1 has now been set up with entirely different characteristics from window #0. Window #0 retains its identity as shown by the continuation of the example:

```
ESC O A 0 set up window #0
A B C write to it
```

The keyboard synchronization command "MODE 0" (zero), causes the status of the addressed window to be sent to the keyboard illuminated keys. After a keyboard "sync" command, the window status will be accurately reflected by the illuminated keys.

Windows may be defined to be overlapping. When two windows share the same point on the screen, whichever window writes the point last is the one that affects that point.

Continued example:

```
ESC O A 2 set up w #2
BACKGROUND MAGENTA ERASE PAGE color it
```

ESC O A 1

move back to w#1

ERASE PAGE

note difference

The following illustration shows what happens when a window boundary is exceeded:

WINDOW . --> --> --> --> ↓ . ↓

BACKGROUND BLUE ERASE PAGE

LF --> SHIFT ↓ SHIFT ↓ SHIFT ↓ SHIFT ↓

ABCDEFGHI

ERASE PAGE

This sequence uses the cursor control to define the window. Note that the information written outside the window is not cleared by the ERASE PAGE.

III. PROGRAM INFORMATION AND POSSIBLE MODIFICATIONS

A. PROGRAM DETAILS

1. Explanation of Variables

Four different one-dimensional arrays are set up as follows:

PS\$ - This string array stores data associated with the special function keys on the upper three rows of the keyboard. The functions of each of these keys (except for the RESET key) can be duplicated from the basic keyboard using a combination of keys; therefore the upper three rows are merely a convenience for the user. The array is first initialized so that each position contains CHR\$(7), the ASCII code for the "bell". Keys which need to be suppressed under program control (because they would kill the program) are then trapped and the "bell" is printed instead of the usual function. Keys which do not interfere with the program and are an advantage to the user need to be enabled; the array positions for these keys are refilled with appropriate data.

C - This array is used to store user input such as coordinates and other parameters for primitives like radius (for circle) or height (for slant rectangle).

X, Y - These two arrays are used for plotting of coordinates in the overlay routines.

A number of flags are used in this program; they are summarized in Figure 16, page 45. The last two flags in the table became necessary

USE	NAME	VALUES
Tablet Flag	FL	0 = Tablet off -1 = Tablet enabled
Tablet Initialization	TI	0 = Tablet uninitialized -1 = Tablet initialized and devices assigned
Create Buffer	CR	0 = Create mode off -1 = Create mode enabled
Point Mode	PM	0 = Point mode off -1 = Point mode enabled
Background Flag	BF	0 = Background color stable -1 = Store current background color
Point Flag	PF	0 = Not first time in point mode -1 = First time in point mode
Other Coordinates	OC	0 = No coordinates yet fetched from tablet -1 = Other coordinates fetched

FIGURE 16.

FLAGS USED IN PART 2

TYPE OF VARIABLE	NAME	VALUES
command	CMD\$	"REC" = SLANT RECTANGLE "VEC" = WIDE VECTOR "ARR" = ARROW "DAR" = DOUBLE-HEADED ARROW
state	BL\$	"XB" = BLINK off "B" = BLINK on
	FI\$	"XF" = FILL off "F" = FILL on
	DO\$	"XD" = DOTTED LINE off "D" = DOTTED LINE on
coordinates	XS YS XN YN	Starting X coordinate Starting Y coordinate Ending X coordinate Ending Y coordinate
attributes	W HT BC AH\$	width of fill space height of rectangle fill color "FIX" = fixed head (for arrows) "VAR" = variable head (for arrows)

FIGURE 17.

OVERLAY ROUTINE PARAMETERS

due to the unusual way the tablet handles input/output in stream mode. There is an apparent delay in registering the coordinate values from the tablet. After a "hit" in selecting tablet coordinates, the first coordinates gathered are not the ones desired but are instead the coordinates just previous to the ones selected. A double input statement is used to "discard" the first coordinates and store the ones actually selected.

If the user is in point mode the delay does not occur and the double input statement is unnecessary. The PM flag is TRUE and the first input statement is bypassed.

A complication occurs when moving from stream to point mode. The first time coordinates are selected in point mode after having selected coordinates in stream mode, the coordinate pair from the last "hit" is still stored and erroneously picked up as the first "hit" in point mode. The following table summarizes the action taken to avoid this undesirable selection of a point:

OC	PF	ACTION
0	0	none
0	-1	none
-1	0	none
-1	-1	Use up old point

Note that in BASIC the NOT of 0 is -1, so it is necessary to use these values for logical operations involving NOT. A flag with value 0 is defined as FALSE while a flag with any other value is defined as TRUE.

Some overlay routine variables are necessary in the main driver for the passing of input data. These are detailed in Figure 17, page 46.

Tablet variables include the following:

SP\$ - Speed of picking up coordinates:

"0" = stream mode, 70 coordinate pairs per second

"P" = point mode

"S" = idle mode (stop)

Other speeds are available. (See Chro79a.)

XX, YY - Used to store raw coordinates from the tablet; their range is 0 to 2200.

X1, Y1 - Used to store scaled coordinates to pass from tablet to screen.

OX, OY - Used to copy values of coordinates input so comparison can be made with next input to see if conditions have changed.

F\$ - Used to store value of 4-button cursor key used:

"0" = no hit

"1" = top button hit

"2" = left button hit - unused in this program

"4" = bottom button hit

"8" = right button hit - unused in this program.

PR\$ - Used as a temporary to store code for primitive selected while outputting dots for points selected in tablet stream mode.

Other variables used include the following:

K - Used to store the input from the keyboard,

LO - Used to store contents of the location in memory which keeps bit information for state conditions.

COL - Used to store contents of the location in memory which

keeps bit information for background color.

BG - Used to determine background color.

NC - Used to indicate the number of coordinates plus other variables needed as input for a primitive.

CT - Used to count the number of coordinates in digit input mode; since 3 digits = 1 coordinate, the digits are stored in array C when CT indicates 3 digits have been input.

I, M - Used as subscripts to array C when filling C and printing from C, respectively.

2. Explanation of Code Sections

PART1 contains the initialization and is executed only once. The "MODEOFF" statement is used to enable reading of the keyboard directly. Mode codes are now echoed as tildes and escape codes are echoed as reverse apostrophes. Without this command a mode code would stop program execution and echo further input to the output device until the RETURN key was typed.

Logical to physical device assignments are made and initial states are set. Data, such as the color codes and primitive codes, is placed in the array P\$. Part 1 then "chains" to Part 2.

PART2 contains the main driver. The first section is the interrupt handler. The program loops on a statement until an interrupt is detected. If it is indeed a keyboard interrupt (ERR=24), then the input is stored. Trapping of further interrupts is disabled by the "ON ERROR GOTO 0" statement which causes BASIC to print an error message on any subsequent errors and terminate execution.

Some of the ASCII codes remain unassigned and could be used for new

input functions (see ASCII chart in Chro78b). The interrupt handler determines which key caused the interrupt (see Figure 18, page 51. for key code values) and branches to the appropriate place in the program. Some types of keyboard interrupts are disallowed during certain routines, such as coordinate gathering, to prevent nested subroutine calls from being stacked up. A "RESUME" statement reenables the keyboard interrupts.

The primitive handler section is entered if the interrupt handler detects a primitive. The number of coordinates and attributes for that primitive is set with an assignment to the variable NC. Since it is impossible to determine ahead of time how many coordinates are expected for concatenated primitives, NC is arbitrarily set at 999. This section then branches to a subroutine which fetches the desired input and returns it in array C. If the primitive is a built-in primitive, the contents of C are then output to the underlying Chromatics software with a "PLOT" statement. If the primitive is one of the extended primitives, the contents of array C are assigned to the Part 3 or Part 4 parameters and the appropriate overlay is loaded.

Concatenated primitives are again an exception: it is assumed the user would like to see the results as coordinate selection occurs and therefore these primitives are printed out as selection occurs.

At the conclusion of this section, NC is reset to zero to indicate that no coordinates are presently being sought and the full range of keyboard interrupts is enabled.

State changes are handled by accessing a fixed memory location, &H3B52, and performing a logical AND with the appropriate hexadecimal value to check bits. The condition of the state is reversed and any

FIGURE 18.

CODE VALUES FOR SPECIAL FUNCTION KEYS

RESET	wide vect BOOT	append TEST	tablet TEXT EDIT	arrow ASMB	point mode PROMPCMR	slant rect BASIC	COPY	F1	F2	F3	F4	F5	F6	F7	F8
-------	-------------------	----------------	---------------------	---------------	------------------------	---------------------	------	----	----	----	----	----	----	----	----

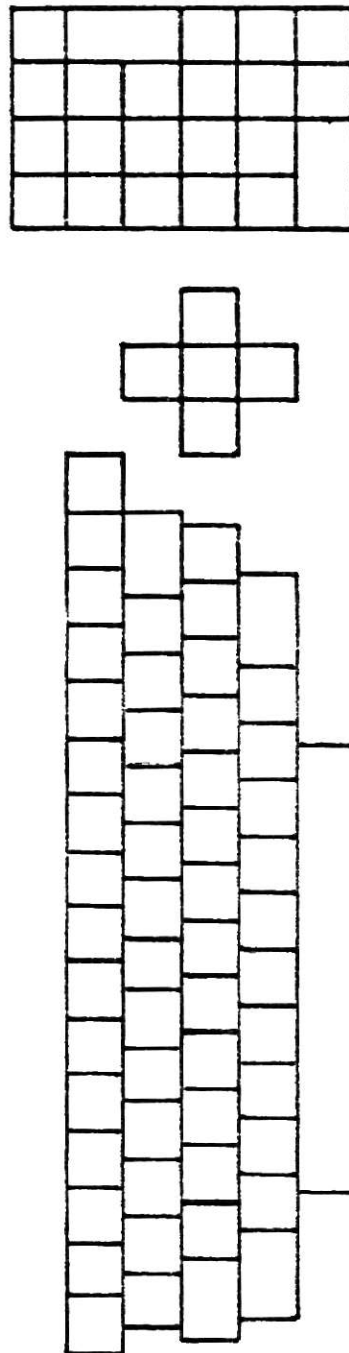
--- 199 244 216 193 208 194 205 160 161 162 163 164 165 166 167

CPU OS	d arrow DISK OS	edit CRT OS	CREATE	ZOOM	REDRAW	XMIT	WINDOW	CURSOR X-Y	PLOT	DOT	X BAR	concat vec Y BAR	VECTOR	RECT	CIRCLE
--------	--------------------	----------------	--------	------	--------	------	--------	---------------	------	-----	-------	---------------------	--------	------	--------

218 196 212 129 250 215 213 247 245 130 149 145 146 151 155 154

ROLL	BACK	FILL	BLACK	BLUE	GREEN	CYAN	RED	MAGENTA	YELLOW	WHITE
------	------	------	-------	------	-------	------	-----	---------	--------	-------

131 132 133 134 176 177 178 179 180 181 182 183



STATE	ACTION TO ENABLE	ACTION TO DISABLE	ASSOCIATED FLAG
Create	CREATE key ESC "C"	CREATE(off) (EOF)	CR
Plot	PLOT key MODE "G"	PLOT(off) (ASCII 21)	
Roll	ROLL key MODE "P"	ROLL(off) MODE "R"	
Blink	BLINK key MODE "1"	BLINK(off) MODE "2"	BL\$
Background	BACKGROUND key	BACKGROUND(off)	
Fill	FILL key MODE "F"	FILL(off) MODE "L"	FI\$, BF
Tablet	TABLET key	TABLET(off)	FL
Point Mode	POINT MODE key	POINT MODE(off)	PM
Dotted	CTL "D"	(automatically disabled after each use)	DO\$

FIGURE 19.

SUMMARY OF MACHINE STATES

flags are switched. The states are summarized in Figure 19, page 52.

When the tablet is first selected the code branches to a small subsection which does tablet initialization, including device assignments.

The create mode is not an initial condition; it is turned on only as a user option. The program automatically resets ROLL off and PLOT on so that these states are recorded in the create buffer. A flag is set so that it is possible to test for user request for create mode irrespective of whether the create light itself is on or off.

Many commands, such as tablet cursor movements, are not needed in recreating a display; some commands, such as the "bell", tend to be annoying when reexecuted from the buffer in a REDRAW command. These reasons in addition to space considerations provide incentive for elimination of unnecessary items from the create buffer. Therefore, the program continually turns create mode off unless the input information is necessary for a display. Since proper placement of characters depends on knowing the position of the cursor (including any backspacing to correct errors!), the cursor movements are recorded in the create buffer if character mode is on.

The end of file (EOF) character, CHR\$(30), turns the create buffer off. Create mode is reenabled with ESCape "Q", which appends to the last EOF in the buffer; note that ESCape "C" or the CREATE key always begins at the first buffer position.

Background color is also stored in a fixed location in memory. A "PEEK" statement transfers the contents into COL, which is then ANDed with &H55 to check which bits are on in each of the four memory planes, and the result stored in BG. The actual background color is shown by

Figure 20, page 55. Since the use of the FILL command in an overlay routine changes the background color, it is necessary to determine and store the original color before loading the overlay. This will allow restoration of the original environment.

The coordinate subroutine gathers all coordinates. Variables are initialized. The tablet flag is checked and if on, the routine branches to handle input from the tablet. Otherwise, the routine loops waiting for keyboard input. When information comes from the keyboard it is first checked by the interrupt handler section. Any coordinates using cursor position are determined by checking NC. If NC is not equal to zero, then coordinates are being fetched. A "." is then interpreted as a "hit" instead of a period, and the present cursor position is stored in the array C. Digit coordinates are gathered a digit at a time until there are three digits and then stored in array C. (Forcing the user to input three digits is an arbitrary decision but it avoids errors such as 10, 30 being interpreted as 103.0-- if a comma is accidentally omitted.) Tablet coordinates may be gathered in either stream or point mode, but only stream mode "tracks" the cursor on the screen. This portion loops until either the number of preset coordinates is reached or the function is terminated. Selected points are recorded when F\$ = 1; concatenated primitives (both dots and vectors) are terminated by "4" in F\$.

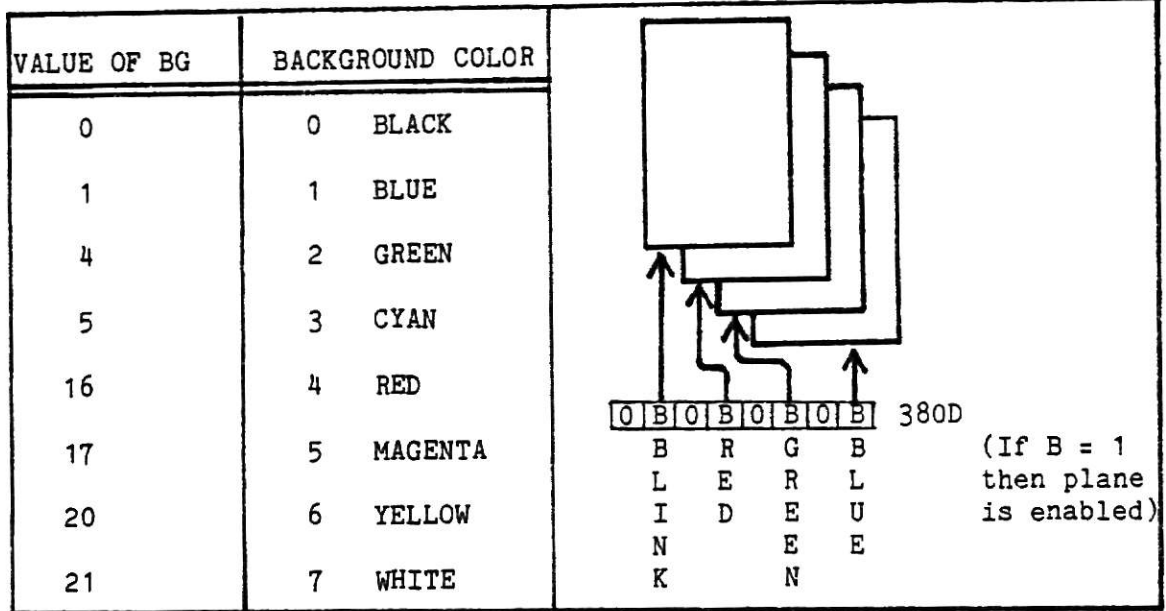


FIGURE 20.

DETERMINATION OF BACKGROUND COLOR

PRIMITIVE	CODE	COMMENTS
RECTANGLE	+	
CIRCLE	*	
ARC)	Not Available in this program
CONCAT VECTOR	((2 ways to turn off in this program)
VECTOR	,	
DOT	%	May be used as concatenated dots
INCR DOT	&	Not available in this program
X-BAR	!	
INCR X-BAR	#	Not available in this program
Y-BAR	"	Achieved in the program with PLOT " sequence; NC = 3. (BASIC does not support "" so array P\$ not used.)
INCR Y-BAR	\$	Not available in this program

FIGURE 21.

BUILT-IN PRIMITIVES AVAILABLE ON CHROMATICS

B. EXPANSION OF PRIMITIVES

1. Built-in Primitives

The basic set of built-in primitives which is available on the Chromatics includes the figures listed in Figure 21, page 55. Of these, all are available within this program except for the ones indicated in Figure 21. The ARC was considered to be of lower priority than most of the other primitives and no effort is made to program in the necessary code; it would not be difficult to add it. The same is true for INCRemental X-BAR and INCRemental Y-BAR.

The INCRemental DOT is somewhat more complicated in implementation and especially in user information; for those reasons plus anticipated low user demand it is not included in this program.

The CIRCLE command contained a flaw if cursor coordinate input was used. The first "." selected the coordinates for the center of the circle; a second "." selected the radius. Previously, the first three digits (the X coordinate) of the second point were automatically picked up as the value of the radius--generally producing results which were not what the user would have been expecting. The standard distance formula from mathematics, $\sqrt{(X1-X2)^2 + (Y1-Y2)^2}$, is used to determine the radius, with the $(Y1-Y2)$ being divided by $\sqrt{2}$ to adjust for the difference in X-dot and Y-dot distances.

2. Extended Primitives

The set of extended graphics primitives developed by Maxine Yee (Yee80) includes routines for SLANT_RECTANGLE, WIDE_VECTOR, ARROW, and DOUBLE-HEADED_ARROW with the latter three being available in solid or dotted line mode. (See Figure 14 for details on these primitives.) The

possibility for further software extensions is clear.

The overlay techniques used are fairly simple. The command DOS"CHAIN filename loads in the new file while saving all variables--with their values--from the previous file. The "chained-to" program may access these variables, write to them and leave them to again be referenced or written to by the original program file (or a new overlay) with another DOS"CHAIN filename command to reload (load) the next file. Any variables which need to be local to a file should therefore have unique identifiers.

Since the approach to the user interface in the overlay routines (Part3 and Part4) varied considerably from the main driver (Part2), some revision was necessary for smooth transitions.

C. TABLET SIZE AND SCALING

1. User-Defined Tablet Size

It is possible to define a small portion of the tablet and transfer it to the full screen as an enlargement technique. The appendix contains alternate code for the "tablet initialization" section (lines 1150 to 1200) of the main driver. This alternate section uses point mode and bases scaling on the initial input coordinates with which the user defined the tablet size.

2. Scaling to Avoid Distortion

Some effort was made to scale the square tablet to a square portion of the screen. Enlarging the Y coordinates was unsuccessful since points near the top of the tablet exceeded the 511 limit of the screen and created a "wrap-around" effect. Scaling the X coordinates by

dividing by $\sqrt{2}$ (which makes X-dot distances equal to Y-dot distances on the screen) and adding 75 was successful except in the case of circles, where the radius needed to be able to be taken in any direction. (The addition of 75 to the X-coordinate allowed centering of the resulting square area on the screen. When the X-dot distances were limited to the same horizontal length as the corresponding vertical Y-dot distances, there were 150 more X-dots than necessary--this excess number of dots was divided equally between the left and right hand sides of the screen so that the resulting area would be centered and the unused portions would be on the edges of the screen.) Work in this area was not considered a priority and was not pursued although a comprehensive scaling transformation from tablet to screen coordinates would be helpful.

D. FUNCTION KEYS

The function keys marked F1 through F8 at the upper right of the keyboard are programmed to provide special shading effects. Functions for these keys are stored in the P\$ array. Each of these keys performs a complex fill using an arbitrary selection from the alternate character set (Figure 15b) as the fill character. It would be easy to make other selections from the alternate character set or otherwise reprogram these keys. When one of these keys is detected in the interrupt handler section, the code plots the cursor position for window 0, shifts to the alternate character set, prints the complex fill command, returns to the standard character set and branches to the end of the main driver. All that the user has to do is set the background and foreground colors as directed in the section on complex fill.

These function keys, when not under program control, can be programmed by the user to define specific sequences of commands in the following way:

F1 F1 Fn (where n = 2 to 8)

sequence of commands

Fn

Now, whenever the Fn key is hit, the entire sequence defined is executed. Passing these keys through the program in their original state would add to the flexibility of the program for a sophisticated user.

E. MULTIPLE WINDOWS

Although it is possible to define multiple windows within this program, there are some limitations. Statements #500 and #610 plot the cursor position from window #0 only. It would be possible to set up a variable for the window number and plot the cursor from any of the four windows.

BIBLIOGRAPHY

- Chr78a Chromatics Disk Software Reference Manual, Chromatics, Inc., Atlanta. Ga., 1978.
- Chr78b Chromatics Preliminary Operator's Manual (Revised), Chromatics, Inc., Atlanta. Ga., November, 1978.
- Chr79a Chromatics Bit Pad One Reference Manual, Chromatics, Inc., Atlanta. Ga., 1979.
- Chr79b Chromatics CG BASIC Reference Manual. CG File Handling BASIC Version 3.0. Chromatics, Inc., Atlanta. Ga., January 24, 1979.
- New79 Newman. W. and Sproull, R. Principles of Interactive Computer Graphics, Second Edition, McGraw-Hill Book Co., 1979.
- Yee80 Yee, Maxine. Implementation of Extended Graphics Primitives, Master's Report, KSU Department of Computer Science, 1980.

APPENDIX

PART1C.SRC

```

10 MODEOFF
20 PRINT CHR$(27);"IAF";CHR$(27);"ID9"; 'ALLOW MODES TO BE PICKED UP BY INTERRUPT HANDLER
30 PRINT CHR$(12);"^P^G"; 'LOGICAL TO PHYSICAL DEVICE ASSGNMYS
40 CLEAR 2000 'INITIALIZE WITH ERASE PAGE, ROLL OFF, PLOT ON
50 DIM P$(127),C(15),X(60),Y(60) 'SET STRING SPACE FOR LARGER THAN DEFAULT OF 50 BYTES
60 A$="VAR":DO$="XD" 'SET DIMENSIONS OF ARRAYS NEEDED
70 FOR I=0 TO 127 'SET OPTION DEFAULTS FOR EXTENDED PRIMITIVES
80 P$(I)=CHR$(7) 'FILL ARRAY POSITIONS WITH BELL
90 NEXT
100 FOR I=0 TO 24
110 READ J,P1$
120 P$(J-128)=P1$
130 NEXT
140 '
150 DATA 176,"^C0"
160 DATA 177,"^C1"
170 DATA 178,"^C2"
180 DATA 179,"^C3"
190 DATA 180,"^C4"
200 DATA 181,"^C5"
210 DATA 182,"^C6"
220 DATA 183,"^C7"
230 DATA 149,"^Z"
240 DATA 145,"^J"
250 DATA 151,"^I"
260 DATA 155,"^+"
270 DATA 154,"^*"
280 DATA 146,"^{"
290 DATA 245,"^U"
300 DATA 247,"^W"
310 DATA 250,"^Z"
320 DATA 160,"^>A"
330 DATA 161,"^>B"
340 DATA 162,"^>C"
350 DATA 163,"^>D"
360 DATA 164,"^>E"
370 DATA 165,"^>F"
380 DATA 166,"^>G"
390 DATA 167,"^>H"
400 DOS"CHAIN PART2" 'INITIALIZATION COMPLETE: LOAD PART 2

'CURSOR X-Y FUNCTION
'WINDOWING FUNCTION
'ZOOM FUNCTION
'COMPLEX FILL WITH ALTERNATE CHAR SET

'STORE PRIMITIVE CODES IN ARRAY

'STORE COLOR CODES IN ARRAY

'REFILL SELECTED ARRAY POSITIONS

'FILL ARRAY POSITIONS WITH BELL

```


PART2C.SRC

```

410 '-----INTERRUPT HANDLER-----
420 ON ERROR#1 GOTO 440          *SET JUMP TABLE
430 GOTO 430                    *WAIT FOR INPUT
440 IF ERR=24 THEN K=INP(2H4A)
      ELSE ON ERROR #0 GOTO 0 *CHECK IF ERROR IS INTERRUPT

450 IF K=2H80 THEN PRINT CHR$(30):CHR$(27):"IA9":STOP *IF BREAK KEY THEN STOP
460 LO=PEEK(2H3B52)             *LOCATION OF STATE INFORMATION
470 IF K=5 OR K=8 OR K=10 OR K=11 OR K=22 OR K=25 OR K=28 OR K=29 OR K=31
      THEN IF (LO AND 2H4) THEN PRINT CHR$(K):GOTO 1140 *CURSOR & PLOT

480 IF NOT FL AND NC<>0 THEN IF 48<=K AND K<=57
      THEN ON CT+1 GOTO 1270,1280,1290 *DIGIT MEANS COORD INPUT

490 IF BF THEN IF 48<=K AND K<=55
      THEN BC=(K-48):BF=0:RESUME 1620 *SAVE BACKGROUND COLOR

500 IF (FL=0 AND NC<>0 AND CT=0) THEN IF CHR$(K)="."
      THEN C(I)=CURSX(0):C(I+1)=CURSY(0):I=I+1
      :IF I>=NC OR NC=999 THEN RETURN ELSE RESUME *"." MEANS COORD

510 IF NC=999 THEN IF K=146 OR K=149
      THEN NC=0:PRINT CHR$(7):RETURN *FINISH CONCAT PRIMITIVE

520 *CATCH FOR CB
530 IF K<215 AND CR THEN PRINT CHR$(27):"Q": *IF NOT REDRAW THEN APPEND
540 IF FL AND K=208 THEN IF PM THEN SP$="O":PM=0:PF=0
      ELSE SP$="P":PM=-1:PF=-1 *RESET POINT MODE FLAG

550 IF K>=129 AND K<=134 THEN ON K-128
      GOTO 1000,1020,1040,1060,1080,1100 *STATE CHANGE

560 IF K>=176 AND K<=183 THEN PRINT P$(K-128):GOTO 1130 *COLOR CHANGE
570 IF K=4 THEN DO$="O" *SET DOTTED STATE
580 IF K=2 THEN AH$="FIX" *SET FIXED-HEAD ARROWS
590 IF K=3 AND UC<>0 THEN RETURN *CANCEL PRIMITIVE IN PROGRESS
600 IF UC<>0 THEN GOTO 1130 *IF LOOKING FOR COORDS, SKIP REMAINING OPTIONS

```

```

610 IF K>=160 AND K<=167 THEN PRINT "U":PLOT CURSX(0),CURSY(0):
    PRINT CHR$(14):PS(K-128):CHR$(15):GOTO 1130 'COMPLEX FILL

620 IF K=216 THEN PRINT CHR$(7):GOTO 1120 'TABLET FLAG
630 IF (LO AND &H4) AND K=34 THEN PR$=CHR$(K):PRINT CHR$(K):GOTO 800 'Y-BAR
640 IF K>127 THEN PR$=PS(K-128):PRINT PS(K-128):CHR$(7):
    ELSE PRINT CHR$(K):GOTO 1130 'SET FOR PRIMITIVES OR PASS THRU

650 IF K>=193 AND K<=196 THEN ON K-192 GOTO 1490,1500,1130,1520 'EXTENDED PRIMITIVES
660 IF K=199 THEN GOTO 1510 'EXTENDED PRIMITIVE
670 IF K=149 OR K=146 THEN PRINT CHR$(7):GOTO 740 'PRIMITIVES
680 IF K>=151 AND K<=155 THEN ON K-150 GOTO 780,1130,1130,670,780 'PRIMITIVES
690 IF K=145 THEN PRINT CHR$(7):GOTO 850 'PRIMITIVE
700 IF K=244 THEN PRINT CHR$(7):CHR$(27):"Q":CR=-1 'APPEND TO CREATE BUFFER
710 IF K=215 THEN PRINT CHR$(30):CHR$(12):CHR$(27):"W": 'REDRAW
720 GOTO 1130 'BRANCH AROUND
730 '----HANDLE PRIMITIVES----
740 NC=999:GOSUB 1230:IF K=3 THEN NC=0:GOTO 980
750 IF CR THEN PRINT CHR$(27):"Q":
760 IF NOT FL AND NC<>0 THEN PLOT C(I-2),C(I-1):PRINT CHR$(30):
770 IF NC=999 THEN GOTO 740 ELSE IF FL THEN GOTO 420 ELSE RESUME 420
780 NC=4:GOSUB 1230:NC=0:IF K=3 THEN GOTO 980
790 GOTO 930
800 NC=4:GOSUB 1230:NC=0:IF K=3 THEN GOTO 980
810 IF CR THEN PRINT CHR$(27):"Q":
820 PLOT C(0),C(1),C(3)
830 IF CR THEN PRINT CHR$(30):
840 GOTO 980
850 NC=3:GOSUB 1230:NC=0:IF K=3 THEN GOTO 980
860 GOTO 930
870 NC=3:GOSUB 1230:NC=0:IF K=3 THEN GOTO 980
880 IF K>=48 AND K<=57 THEN 930 ELSE 890
890 IF CR THEN PRINT CHR$(27):"Q":
900 PLOT C(0),C(1):PLOT SQR((C(0)-C(2))^2+(C(1)-C(3))^2)/SQR(2))^2)
910 IF CR THEN PRINT CHR$(30):
920 GOTO 980
930 FOR M=0 TO (I-1) STEP 2
940 IF CR THEN PRINT CHR$(27):"Q":
950 PLOT C(M),C(M+1)
960 IF CR THEN PRINT CHR$(30):
970 NEXT
980 IF FL THEN 420 ELSE RESUME 420

```

```

990 '-----HANDLE STATES-----
1000 IF CR THEN CR=0 ELSE PRINT CHR$(27);"C^P^G";:CR=-1 'CREATE
1010 GOTO 1130
1020 IF (LO AND 8H4) THEN PRINT CHR$(21); ELSE PRINT "^G"; 'PLOT
1030 GOTO 1130
1040 IF (LO AND 8H2) THEN PRINT "^P"; ELSE PRINT "^R"; 'ROLL
1050 GOTO 1130
1060 IF (LO AND 8H40) THEN BL$="XB":PRINT "^2"; ELSE BL$="B":PRINT "^1"; 'BLINK
1070 GOTO 1130
1080 IF (LO AND 8H1) THEN PRINT "^N"; ELSE PRINT "^M^2"; 'BACKGROUND
1090 GOTO 1130
1100 IF (LO AND 8H80) THEN FI$="XF":PRINT "^L"; ELSE FI$="F":PRINT "^f"; 'FILL
1110 GOTO 1130
1120 IF FL THEN FL=-1:IF IF=0 THEN GOTO 1160 'TABLET
1130 PRINT CHR$(30):CHR$(7);
1140 RESUME
1150 '-----FINISH MAIN-----
1155 '
1160 '-----SUB: TABLET INIT-----
1170 SP$="O":PM=0 'SET SPEED: POINT MODE INITIALIZED OFF
1180 PRINT CHR$(30):CHR$(27);"R1C":CHR$(27);"OE5":CHR$(27);"IES"; 'DEVICE ASSIGNMENTS
1190 IF CR THEN PRINT CHR$(27);"Q"; 'APPEND
1200 IF=-1 'SIGNAL TABLET INIT COMPLETE
1210 RESUME 420
1220 '
1230 '-----SUB: COORD-----
1240 CT=0:I=0:F$="0":OX=0:OY=0:OF$="0"
1250 IF FL THEN 1310 ELSE 1260
1260 RESUME 430
1270 Z=100*(K-48):CT=CT+1:RESUME 'FOR DIGIT COORDS
1280 Z=Z+10*(K-48):CT=CT+1:RESUME
1290 C(I)=Z+(K-48):CT=0:I=I+1:IF I>=NC OR (NC=999 AND I=2) THEN RETURN ELSE RESUME
1300 '
1310 'TABLET COORD
1320 PRINT #4:SP$;
1330 RESUME 1340
1340 IF NOT PM THEN INPUT #4:XX,YY,F$ 'DOUBLE INPUT NEEDED FOR STREAM MODE
1350 INPUT #4:XX,YY,F$

```

```

1360 PRINT #4;"S"; 'SET SPEED TO IOLE
1370 X1=INT((XX/4.3):Y1=INT((YY/4.3) 'MAP TABLET TO SCREEN COORDS
1380 IF OX=X1 AND OY=Y1 AND OF$=F$ THEN 1450 'CHECK CHANGES FROM LAST ENTRY
1390 IF NC=999 AND F$="1" AND CR THEN PRINT CHR$(27);"Q";
1400 IF (NOT PF OR NOT OC) AND NC=999 THEN IF F$="1"
      THEN PLOT X1,Y1:PRINT CHR$(30);CHR$(7);GOTO 1450
1410 IF F$="4" THEN NC=0:K=3:RETURN 'TURN OFF PRIMITIVE
1420 IF (NOT PF OR NOT OC) AND F$="1" THEN IF NOT PM
      THEN PRINT "X";:PLOT X1,Y1 'PRINT DOTS FOR STREAM MODE HITS
1430 IF (NOT PF OR NOT OC) AND F$="1" THEN PRINT CHR$(7);:C(I)=X1:C(I+1)=Y1:
      I=I+2:IF I>=NC THEN PRINT PR$;:RETURN ELSE 1450 'LOAD ARRAY
1440 PRINT "^U";:PLOT X1,Y1
1450 OX=X1:OY=Y1:OF$=F$:PF=0:OC=-1 'SAVE VALUES FOR NEXT COMPARISON
1460 PRINT #4:SP$;:GOTO 1340 'RESET SPEED
1470 '
1480 '-----EXTENDED PRIMITIVES-----
1490 CMD$="ARR":NC=5:GOTO 1530
1500 CMD$="REC":NC=6:GOTO 1530
1510 CMD$="VEC":NC=5:GOTO 1530
1520 CMD$="DAR":NC=5:GOTO 1530
1530 IF FL THEN IF CMD$="REC" THEN NC=8 ELSE NC=6 'INCREASE # COORDS IF TABLET ON
1540 GOSUB 1230:NC=0:XS=C(0):YS=C(1):XN=C(2):YN=C(3):W=C(4)
1550 IF K=3 THEN RESUME 420 'CANCEL PRIMITIVE
1560 IF NOT FL AND CHR$(K)="." AND CMD$="REC"
      THEN NC=2:GOSUB 1230:NC=0:C(6)=C(0):C(7)=C(1) 'GET TWO MORE COORDS
1570 IF FL OR CHR$(K)="." THEN W=INT((SQR((C(2)-C(4))^2+(C(3)-C(5))^2)/SQR(2))^2)
1580 IF FL OR CHR$(K)="." THEN IF CMD$="REC"
      THEN C(5)=INT((SQR((C(4)-C(6))^2+(C(5)-C(7))^2)/SQR(2))^2)
1590 IF CMD$="REC" THEN HT=W:W=C(5)
1600 IF FL THEN 1610 ELSE RESUME 1610
1610 IF F1$="F" THEN BF=-1:GOTO 430
1620 GOSUB 1660
1630 IF CMD$="VEC" OR CMD$="REC" THEN DOS="CHAIN PART3
1640 IF CMD$="ARR" OR CMD$="DAR" THEN DOS="CHAIN PART4
1650 '

```

```
1660 '-----SUB: SAVE BACKGROUND COLOR-----  
1670 COL=PEEK(&H380D)  
1680 BG=COL AND &H55  
1690 IF BG=0 THEN B1=0  
1700 IF BG=1 THEN B1=1  
1710 IF BG=4 THEN B1=2  
1720 IF BG=5 THEN B1=3  
1730 IF BG=16 THEN B1=4  
1740 IF BG=17 THEN B1=5  
1750 IF BG=20 THEN B1=6  
1760 IF BG=21 THEN B1=7  
1770 RETURN  
1780 .
```

```

PART3.SRC
5 *AUTHOR-MAXIMIE YEE
6 *PROGRAM-EGP(EXTENDED GRAPHIC PRIMITIVES)
7 *LANGUAGE-CHROMATICS BASIC VERRSION 3.0
8 *PLACE-KANSAS STATE UNIVERSITY,DEPT. OF COMPUTER SCIENCE
9 *DATE-DEC..1979
10 *****
11 *
12 * THIS IS A PROGRAM FOR THE GENERATION OF EXTENDED GRAPHIC PRIMITIVES
13 *
14 *****
114 SQ=SQR(2)
115 IF BL$="B" THEN PRINT"^1";ELSE PRINT "^2"
120 IF CMD$="VEC" THEN GOSUB 1500:GOTO 155
150 IF CMD$="REC" THEN GOSUB 20010
155 DOS$="XD":W=1:PRINT "^M^C":CHR$(48+B1):"^N":DOS"CHAIN PART2
1500 *
1501 *****
1502 *
1503 * THIS IS A SUBROUTINE FOR DRAWING WIDE VECTORS
1504 *
1505 *****
1520 I=1
1525 IF DO$="D" THEN GOSUB 2200:RETURN
1527 YS=YS/SQ:YN=YN/SQ
1530 X=XN-XS
1540 Y=YN-YS
1550 L=SQR(X^2+Y^2)
1560 X(I+2)=XS-(W*Y)\L
1570 Y(I+2)=YS+(W*X)\L
1580 X(I+1)=XN-(W*Y)\L
1590 Y(I+1)=YN+(W*X)\L
1592 Y(I+2)=Y(I+2)*SQ
1594 Y(I+1)=Y(I+1)*SQ
1596 YS=YS*SQ:YN=YN*SQ
1600 IF FI$<>"F" THEN GOSUB 54000
      ELSE GOSUB 54500
      *SUB FOR PLOTTING ENV
1615 *IF DO$="D" THEN GOTO 1625
1620 PLOT XS,YS,XN,YN
1625 PLOT XN,YN,X(I+1),Y(I+1)
1630 PLOT X(I+1),Y(I+1),X(I+2),Y(I+2)
1640 PLOT X(I+2),Y(I+2),XS,YS
1660 IF FI$="F" THEN XMH=(XS+X(I+1))/2:YMH=(YS+Y(I+1))/2:GOSUB 55000
1680 RETURN

```

```

*CAL.LENGTH OF GIVEN LINE
*CAL X2 COORD
*CAL.Y2 COORD
*CAL.X1 COORD
*CAL.Y1 COORD

```

```

2200 *
2201 * *****
2202 *
2203 *      THIS IS A SUBROUTINE FOR DRAWING DOTTED LINES
2204 *
2205 * *****
2206 *
2207 *      YS=YS/SQ:YN=YN/SQ
2208 *
2209 *      J=1
2210 *
2211 *      X(J)=XS
2212 *
2213 *      Y(J)=YS
2214 *
2215 *      XB=XS:YB=YS
2216 *
2217 *      D = 12
2218 *
2219 *      SEG = D
2220 *
2221 *      SP = SEG\2
2222 *
2223 *      L1= SQR((XN-XS)^2 + (YN-YS)^2)
2224 *
2225 *      IF L1<SP AND CMD$="VEC" THEN PRINT CHR$(21);"2315";:RETURN
2226 *
2227 *      COW = (XN-XS)/L1
2228 *
2229 *      SNW = (YN-YS)/L1
2230 *
2231 *      K2= (L1\((SP+SEG))*2 + 1
2232 *
2233 *      FOR J = 1 TO K2 STEP 1
2234 *
2235 *      X(J+1) = XB + D*(COW)
2236 *
2237 *      Y(J+1) = YB + D*(SNW)
2238 *
2239 *
2240 *
2241 *      'ASSIGN SOLID SEGMENT OF THE DOTTED LINE
2242 *
2243 *      'ASSIGN SPACE OF THE DOTTED LINE
2244 *
2245 *      'CAL LENGTH OF GIVEN LINE
2246 *      CHR$(21);"2315";:RETURN
2247 *
2248 *      'CAL COSINE OF AN ANGLE
2249 *
2250 *      'CAL SINE OF AN ANGLE
2251 *
2252 *      'CAL TERMINAL VALUE FOR THE LOOP
2253 *
2254 *      'CAL X(I) COORD
2255 *
2256 *      'CAL Y(I) COORD
2257 *
2258 *
2259 *
2260 *
2261 *
2262 *
2263 *
2264 *
2265 *
2266 *
2267 *
2268 *
2269 *
2270 *
2271 *
2272 *
2273 *
2274 *
2275 *
2276 *
2277 *
2278 *
2279 *
2280 *
2281 *
2282 *
2283 *
2284 *
2285 *
2286 *
2287 *
2288 *
2289 *
2290 *
2291 *
2292 *
2293 *
2294 *
2295 *
2296 *
2297 *
2298 *
2299 *
2300 *
2301 *
2302 *
2303 *
2304 *
2305 *
2306 *
2307 *
2308 *
2309 *
2310 *
2311 *
2312 *
2313 *
2314 *
2315 *
2316 *
2317 *
2318 *
2319 *
2320 *
2321 *
2322 *
2323 *
2324 *
2325 *
2326 *
2327 *
2328 *
2329 *
2330 *
2331 *
2332 *
2333 *
2334 *
2335 *
2336 *
2337 *
2338 *
2339 *
2340 *
2341 *
2342 *
2343 *
2344 *
2345 *
2346 *
2347 *
2348 *
2349 *
2350 *
2351 *
2352 *
2353 *
2354 *
2355 *
2356 *
2357 *
2358 *
2359 *
2360 *
2361 *
2362 *
2363 *
2364 *
2365 *
2366 *
2367 *
2368 *
2369 *
2370 *
2371 *
2372 *
2373 *
2374 *
2375 *
2376 *
2377 *
2378 *
2379 *
2380 *
2381 *
2382 *
2383 *
2384 *
2385 *
2386 *
2387 *
2388 *
2389 *
2390 *
2391 *
2392 *
2393 *
2394 *
2395 *
2396 *
2397 *
2398 *
2399 *
2400 *
2401 *
2402 *
2403 *
2404 *
2405 *
2406 *
2407 *
2408 *
2409 *
2410 *
2411 *
2412 *
2413 *
2414 *
2415 *
2416 *
2417 *
2418 *
2419 *
2420 *
2421 *
2422 *
2423 *
2424 *
2425 *
2426 *
2427 *
2428 *
2429 *
2430 *
2431 *
2432 *
2433 *
2434 *
2435 *
2436 *
2437 *
2438 *
2439 *
2440 *
2441 *
2442 *
2443 *
2444 *
2445 *
2446 *
2447 *
2448 *
2449 *
2450 *
2451 *
2452 *
2453 *
2454 *
2455 *
2456 *
2457 *
2458 *
2459 *
2460 *
2461 *
2462 *
2463 *
2464 *
2465 *
2466 *
2467 *
2468 *
2469 *
2470 *
2471 *
2472 *
2473 *
2474 *
2475 *
2476 *
2477 *
2478 *
2479 *
2480 *
2481 *
2482 *
2483 *
2484 *
2485 *
2486 *
2487 *
2488 *
2489 *
2490 *
2491 *
2492 *
2493 *
2494 *
2495 *
2496 *
2497 *
2498 *
2499 *
2500 *
2501 *
2502 *
2503 *
2504 *
2505 *
2506 *
2507 *
2508 *
2509 *
2510 *
2511 *
2512 *
2513 *
2514 *
2515 *
2516 *
2517 *
2518 *
2519 *
2520 *
2521 *
2522 *
2523 *
2524 *
2525 *
2526 *
2527 *
2528 *
2529 *
2530 *
2531 *
2532 *
2533 *
2534 *
2535 *
2536 *
2537 *
2538 *
2539 *
2540 *
2541 *
2542 *
2543 *
2544 *
2545 *
2546 *
2547 *
2548 *
2549 *
2550 *
2551 *
2552 *
2553 *
2554 *
2555 *
2556 *
2557 *
2558 *
2559 *
2560 *
2561 *
2562 *
2563 *
2564 *
2565 *
2566 *
2567 *
2568 *
2569 *
2570 *
2571 *
2572 *
2573 *
2574 *
2575 *
2576 *
2577 *
2578 *
2579 *
2580 *
2581 *
2582 *
2583 *
2584 *
2585 *
2586 *
2587 *
2588 *
2589 *
2590 *
2591 *
2592 *
2593 *
2594 *
2595 *
2596 *
2597 *
2598 *
2599 *
2600 *
2601 *
2602 *
2603 *
2604 *
2605 *
2606 *
2607 *
2608 *
2609 *
2610 *
2611 *
2612 *
2613 *
2614 *
2615 *
2616 *
2617 *
2618 *
2619 *
2620 *
2621 *
2622 *
2623 *
2624 *
2625 *
2626 *
2627 *
2628 *
2629 *
2630 *
2631 *
2632 *
2633 *
2634 *
2635 *
2636 *
2637 *
2638 *
2639 *
2640 *
2641 *
2642 *
2643 *
2644 *
2645 *
2646 *
2647 *
2648 *
2649 *
2650 *
2651 *
2652 *
2653 *
2654 *
2655 *
2656 *
2657 *
2658 *
2659 *
2660 *
2661 *
2662 *
2663 *
2664 *
2665 *
2666 *
2667 *
2668 *
2669 *
2670 *
2671 *
2672 *
2673 *
2674 *
2675 *
2676 *
2677 *
2678 *
2679 *
2680 *
2681 *
2682 *
2683 *
2684 *
2685 *
2686 *
2687 *
2688 *
2689 *
2690 *
2691 *
2692 *
2693 *
2694 *
2695 *
2696 *
2697 *
2698 *
2699 *
2700 *
2701 *
2702 *
2703 *
2704 *
2705 *
2706 *
2707 *
2708 *
2709 *
2710 *
2711 *
2712 *
2713 *
2714 *
2715 *
2716 *
2717 *
2718 *
2719 *
2720 *
2721 *
2722 *
2723 *
2724 *
2725 *
2726 *
2727 *
2728 *
2729 *
2730 *
2731 *
2732 *
2733 *
2734 *
2735 *
2736 *
2737 *
2738 *
2739 *
2740 *
2741 *
2742 *
2743 *
2744 *
2745 *
2746 *
2747 *
2748 *
2749 *
2750 *
2751 *
2752 *
2753 *
2754 *
2755 *
2756 *
2757 *
2758 *
2759 *
2760 *
2761 *
2762 *
2763 *
2764 *
2765 *
2766 *
2767 *
2768 *
2769 *
2770 *
2771 *
2772 *
2773 *
2774 *
2775 *
2776 *
2777 *
2778 *
2779 *
2780 *
2781 *
2782 *
2783 *
2784 *
2785 *
2786 *
2787 *
2788 *
2789 *
2790 *
2791 *
2792 *
2793 *
2794 *
2795 *
2796 *
2797 *
2798 *
2799 *
2800 *
2801 *
2802 *
2803 *
2804 *
2805 *
2806 *
2807 *
2808 *
2809 *
2810 *
2811 *
2812 *
2813 *
2814 *
2815 *
2816 *
2817 *
2818 *
2819 *
2820 *
2821 *
2822 *
2823 *
2824 *
2825 *
2826 *
2827 *
2828 *
2829 *
2830 *
2831 *
2832 *
2833 *
2834 *
2835 *
2836 *
2837 *
2838 *
2839 *
2840 *
2841 *
2842 *
2843 *
2844 *
2845 *
2846 *
2847 *
2848 *
2849 *
2850 *
2851 *
2852 *
2853 *
2854 *
2855 *
2856 *
2857 *
2858 *
2859 *
2860 *
2861 *
2862 *
2863 *
2864 *
2865 *
2866 *
2867 *
2868 *
2869 *
2870 *
2871 *
2872 *
2873 *
2874 *
2875 *
2876 *
2877 *
2878 *
2879 *
2880 *
2881 *
2882 *
2883 *
2884 *
2885 *
2886 *
2887 *
2888 *
2889 *
2890 *
2891 *
2892 *
2893 *
2894 *
2895 *
2896 *
2897 *
2898 *
2899 *
2900 *
2901 *
2902 *
2903 *
2904 *
2905 *
2906 *
2907 *
2908 *
2909 *
2910 *
2911 *
2912 *
2913 *
2914 *
2915 *
2916 *
2917 *
2918 *
2919 *
2920 *
2921 *
2922 *
2923 *
2924 *
2925 *
2926 *
2927 *
2928 *
2929 *
2930 *
2931 *
2932 *
2933 *
2934 *
2935 *
2936 *
2937 *
2938 *
2939 *
2940 *
2941 *
2942 *
2943 *
2944 *
2945 *
2946 *
2947 *
2948 *
2949 *
2950 *
2951 *
2952 *
2953 *
2954 *
2955 *
2956 *
2957 *
2958 *
2959 *
2960 *
2961 *
2962 *
2963 *
2964 *
2965 *
2966 *
2967 *
2968 *
2969 *
2970 *
2971 *
2972 *
2973 *
2974 *
2975 *
2976 *
2977 *
2978 *
2979 *
2980 *
2981 *
2982 *
2983 *
2984 *
2985 *
2986 *
2987 *
2988 *
2989 *
2990 *
2991 *
2992 *
2993 *
2994 *
2995 *
2996 *
2997 *
2998 *
2999 *
3000 *

```

```

3010 *
3020 *THIS IS A SUBROUTINE FOR DRAWING DOTTED WIDE LINES
3030 XN=X(J+1)
3040 YN=Y(J+1)
3050 XS=X(J)
3060 YS=Y(J)
3070 GOSUB 1527
3080 RETURN
3081 *
3100 *
3110 *THIS IS A PLOTTING SUBROUTINE FOR A SINGLE DOTTED LINE
3120 PLOT X(J),Y(J),X(J+1),Y(J+1)
3130 RETURN
20010 *****
20020 *
20030 *THIS IS A SUBROUTINE FOR DRAWING RECTANGLES
20040 *
20050 *****
20060 *
20082 YS=YS/SQ:YN=YN/SQ
20090 X=XN-XS
20100 Y=YN-YS
20120 L=SCR(X^2+Y^2)
20130 CO#X/L
20140 SN#Y/L
20150 *
20160 *FOLLOWING SECTION OF CODE CALCULATES THE VARIOUS COORD. OF THE RECTANGLE
20170 *
20180 I=1
20190 X(I)=XS+W*CO#
20200 Y(I)=YS+W*SN#
20210 X(I+1)=X(I)-W*SN#
20220 Y(I+1)=Y(I)+W*CO#

```



```

20230 X(I+2)=X(I)+(W-HT)*SN#
20240 Y(I+2)=Y(I)+(HT-W)*CO#
20250 X(I+3)=XS-HT*SN#
20260 Y(I+3)=YS+HT*CO#
20270 X(I+4)=XS+(L-W)*CO#
20280 Y(I+4)=YS+(L-W)*SN#
20290 X(I+5)=X(I+4)-W*SN#
20300 Y(I+5)=Y(I+4)+W*CO#
20310 X(I+6)=X(I+4)+(W-HT)*SN#
20320 Y(I+6)=Y(I+4)+(HT-W)*CO#
20322 X(I+7)=XN-HT*SN#
20324 Y(I+7)=YN+HT*CO#
20326 YS=YS*SQ:YN=YN*SQ
20328 Y(I)=Y(I)*SQ
20330 Y(I+1)=Y(I+1)*SQ
20332 Y(I+2)=Y(I+2)*SQ
20334 Y(I+3)=Y(I+3)*SQ
20336 Y(I+4)=Y(I+4)*SQ
20338 Y(I+5)=Y(I+5)*SQ
20340 Y(I+6)=Y(I+6)*SQ
20342 Y(I+7)=Y(I+7)*SQ
20350 GOSUB 54000:GOSUB 20700
20360 RETURN
20700 .
20710 *THE FOLLOWING SECTION OF CODE PLOTS THE RECTANGLE
20720 .
20730 PLOT XS,YS,XN,YN
20740 PLOT XN,YN,X(I+7),Y(I+7)
20750 PLOT X(I+7),Y(I+7),X(I+3),Y(I+3)
20760 PLOT X(I+3),Y(I+3),XS,YS
20770 PLOT X(I+1),Y(I+1),X(I+5),Y(I+5)
20780 PLOT X(I+5),Y(I+5),X(I+6),Y(I+6)
20790 PLOT X(I+6),Y(I+6),X(I+2),Y(I+2)
20800 PLOT X(I+2),Y(I+2),X(I+1),Y(I+1)
20810 IF FIS="F" THEN XMH=(XS+X(I+1))/2:YMH=(YS+Y(I+1))/2:GOSUB 55000
20820 RETURN

```

```

54000 .
54010 .
54020 *THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT AND SET COLOR
54050 PRINT "^M";
54060 PRINT CHR$(1):"C":CHR$(48+BC);
54070 PRINT "^N";
54085 PRINT "^G";
54087 PRINT " ";
54090 RETURN
54500 .
54520 *THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT
54530 PRINT "^G";
54535 PRINT " ";
54550 RETURN
55000 .
55020 *THIS SUBROUTINE COMPLEX FILLS AN OBJECT
55030 PRINT "^U": : PLOT XM#,YM#
55035 PRINT "^J";
55040 PRINT "^M";
55050 PRINT CHR$(1):"C":CHR$(48+BC);
55060 PRINT "^N";
55080 PRINT ">"; CHR$(32);
55090 RETURN
55500 .
      *BACKGROUND LIGHT ON
      *SET BACKGROUND COLOR
      *BACKGROUND LIGHT OFF
      *VECTOR SUBMODE
      *MOVE CURSOR TO COORD
      *BACKGROUND LIGHT ON
      *SET BACKGROUND COLOR
      *BACKGROUND LIGHT OFF
      *FILL OBJECT WITH SOLID COLOR

```

PART 4.SRC

```

5 *AUTHOR-MAXINE YEE
6 *PROGRAM-EGP(EXTENDED GRAPHIC PRIMITIVES)
77 *LANGUAGE-CHROMATICS BASIC VERSION 3.0
8 *PLACF-KANSAS STATE UNIVERSITY,DEPT. OF COMPUTER SCIENCE
9 *DATE-DEC.,1979
10 *****
11 .
12 * THIS IS A PROGRAM FOR THE GENERATION OF EXTENDED GRAPHIC PRIMITIVES
13 .
14 *****
52 .
114 SQ=SQRT(2)
115 IF BL$="B" THEN PRINT"^1":ELSE PRINT "^2"
130 IF CMD$="ARR" THEN GOSUB 5000:GOTO 150
140 IF CMD$="DAR" THEN GOSUB 10000
150 DO$="XD":AH$="VAR":W=1:PRINT "^M^C":CHR$(48+B1):"^N":DOS"CHAIN PART2
1500 .
1501 *****
1502 .
1503 .
1504 .
1505 *****
1506 I=1
1525 IF DOS="D" THEN GOSUB 2200:RETURN
1530 X=XN-XS
1540 Y=YH-YS
1550 L=SGR(X^2+Y^2)
1560 X(I+2)=XS-(W*Y)\L
1570 Y(I+2)=YS+(W*X)\L
1580 X(I+1)=X(I)-(W*Y)\L
1590 Y(I+1)=Y(I)+(W*X)\L
1600 IF F1$<>"F" THEN GOSUB 54000
      ELSE GOSUB 54500
      *SUB FOR PLOTTING ENV
1615 IF DOS="D" THEN GOTO 1625
1620 PLOT XS,YS,XN,YN
1625 PLOT XH,YH,X(I+1),Y(I+1)
1630 PLOT X(I+1),Y(I+1),X(I+2),Y(I+2)
1640 PLOT X(I+2),Y(I+2),XS,YS
1660 IF F1$="F" THEN XMH=(XS+X(I+1))/2:YMH=(YS+Y(I+1))/2:GOSUB 55000
1670 IF DOS="D" THEN RETURN ELSE GOSUB 55500
1675 GOTO 1510
1680 RETURN

```

*CAL.LENGTH OF GIVEN LINE

*CAL X2 COORD
 *CAL.Y2 COORD
 *CAL.X1 COORD
 *CAL.Y1 COORD

```

2200 *
2201 * *****
2202 *
2203 * THIS IS A SUBROUTINE FOR DRAWING DOTTED LINES
2204 *
2205 * *****
2206 *
2207 J=1
2230 X(J)=XS
2240 Y(J)=YS
2241 XB=XS:YB=YS
2270 D = 12
2280 SEG = D
2290 SP = SEG\2
2310 L1= SQR((XN-XS)^2 + (YN-YS)^2)
2317 IF L1<SP AND CMD$="ARR" THEN RETURN
2319 IF L1<SP AND CMD$="DAR" THEN RETURN
2320 CO# = (XN-XS)/L1
2330 SN# = (YN-YS)/L1
2340 K = (L1\SP + SEG)*2 + 1
2350 FOR J = 1 TO K STEP 1
2360 X(J+1) = XB + D*(CO#)
2370 Y(J+1) = YB + D*(SN#)
2380 *
2390 *
2400 * THIS SECTION OF CODE TESTS THE VARIOUS END POINTS OF A GIVEN LINE
2410 E = J MOD 2
2420 IF E=0 THEN IF (L1-D) < SEG
      THEN D=L1:NEXT J
      ELSE D=D+SEG:NEXT J
      ELSE GOSUB 54000
2430 *
2440 *
2450 PRINT " ";
2460 PLOT X(J),Y(J),X(J+1),Y(J+1)
2465 IF W>1 THEN GOSUB 3000
2472 IF (L1-D)<=SP THEN RETURN *REMAINING SEG TOO SHORT TO PLOT
2480 D = D + SP
2485 IF E=1 AND L1-D<=SP THEN RETURN
2490 NEXT J
2500 RETURN

```

*ASSIGN SOLID SEGMENT OF THE DOTTED LINE

*ASSIGN SPACE OF THE DOTTED LINE

*CAL LENGTH OF GIVEN LINE

*CAL COSINE OF AN ANGLE

*CAL SINE OF AN ANGLE

*CAL TERMINAL VALUE FOR THE LOOP

*CAL X(I) COORD

*CAL Y(I) COORD

*VECTOR SUBMODE

*PLOT THE DOTTED LINE SEGMENT

*SUB FOR DRAWING DOTTED UNFILLED LINE

```

3000 .
3010 .
3020 *THIS IS A SUBROUTINE FOR DRAWING DOTTED WIDE LINES
3030 XN=X(J+1)
3040 YN=Y(J+1)
3050 XS=X(J)
3060 YS=Y(J)
3070 GOSUB 1530
3080 RETURN
3081 .
5000 *****
5001 .
5002 *      THIS IS A SUBROUTINE FOR DRAWING ARROWS
5003 .
5004 *****
5005 .
5010 .
5100 X = XN-XS
5110 Y = YN-YS
5120 L2= SQR((XN-XS)^2 + (YN-YS)^2)      'CAL LENGTH OF GIVEN LINE
5130 W1= W/2                             'CAL WIDTH OF TAIL
5140 S1# = Y/L2                           'CAL SINE OF AN ANGLE
5150 CS# = X/L2                           'CAL COSINE OF AN ANGLE
5160 IF AH$ = "FIX" AND W1<=1 THEN GOSUB 5410: GOTO 5260
5165 IF AH$ = "FIX" AND W1 >1 THEN GOSUB 5410
5167 IF AH$="FIX" AND D0$="O" THEN 5290
5170 .
5180 .
5190 *THIS SECTION OF CODE CALCULATES THE VARIABLE SIZE OF AN ARROW HEAD
5191 P=2
5200 X(P) = XS + 9/10*X
5210 Y(P) = YS + 9/10*Y
5220 X(P+1) = X(P) - Y/10
5230 Y(P+1) = Y(P) + X/10
5240 X(P+4) = X(P) + Y/10
5250 Y(P+4) = Y(P) - X/10
5255 IF D0$="O" AND W1<=1 THEN 5505
5260 IF W1<=1 THEN GOSUB 54000:GOSUB 5610:RETURN
5270 .

```

```

5280 .
5290 *THIS SECTION OF CODE CALCULATES COORD FOR THE TAIL OF AN ARROW
5300 X(P+2) = X(P) - W1*SI#
5310 Y(P+2) = Y(P) + W1*CS#
5320 X(P+3) = X(P) + W1*SI#
5330 Y(P+3) = Y(P) - W1*CS#
5340 X(P+5) = XS - W1*SI#
5350 Y(P+5) = YS + W1*CS#
5360 X(P+6) = XS + W1*SI#
5370 Y(P+6) = YS - W1*CS#
5372 IF DO$="D" AND AH$="FIX" THEN 5507
5375 IF DO$="D" AND AH$="VAR" THEN 5507
5380 GOSUB 54000:GOSUB 5720:GOSUB 5670:RETURN
5400 RETURN
5410 .
5420 .
5430 *THIS SUBROUTINE CALCULATES FIXED ARROW HEAD
5440 H = 10
5445 P = 2
5450 X(P) = XS + (L2-H)*CS#
5460 Y(P) = YS + (L2-H)*SI#
5470 X(P+1) = X(P) - H*SI#
5480 Y(P+1) = Y(P) + H*CS#
5490 X(P+4) = X(P) + H*SI#
5500 Y(P+4) = Y(P) - H*CS#
5502 IF DO$<>"D" THEN 5260
5505 IF W1<=1 THEN GOSUB 54000:GOSUB 5640:XN=X(P):YN=Y(P):GOSUB 2200:RETURN
5506 IF AH$="FIX" THEN 5290
5507 IF W1>1 THEN GOSUB 54000:GOSUB 5640:GOSUB 6100:GOSUB 2270:RETURN
5510 RETURN
5610 .
5611 *THIS IS A PLOTTING SUBROUTINE
5630 PLOT XS,YS,X(P),Y(P)
5640 PLOT X(P+1),Y(P+1),X(P+4),Y(P+4)
5650 PLOT X(P+4),Y(P+4),XN,YN
5660 PLOT XN,YN,X(P+1),Y(P+1)
5670 IF FL$="F" THEN XMH=(X(P)+XN)/2:YMH=(Y(P)+YN)/2:GOSUB 55000
5710 RETURN
5720 .

```

```

5730 *
5930 *THIS IS A PLOTTING SUBROUTINE FOR WIDE ARROW
5950 PLOT X(P+2),Y(P+2),X(P+1),Y(P+1)
5960 PLOT X(P+1),Y(P+1),XN,YN
5970 PLOT XN,YN,X(P+4),Y(P+4)
5980 PLOT X(P+4),Y(P+4),X(P+3),Y(P+3)
5990 PLOT X(P+3),Y(P+3),X(P+6),Y(P+6)
6000 PLOT X(P+6),Y(P+6),X(P+5),Y(P+5)
6010 PLOT X(P+5),Y(P+5),X(P+2),Y(P+2)
6020 RETURN
6100 *
6101 *
6102 *THIS IS A SUBROUTINE FOR DRAWING DOTTED WIDE ARROWS
6110 XN=X(P+3):YN=Y(P+3)
6130 XS=X(P+6):YS=Y(P+6)
6131 J=1
6141 XS=X(P+6):YB=Y(P+6)
6142 X(J)=X(P+6):Y(J)=Y(P+6)
6150 RETURN
10000 *
10001 ******
10002 *
10003 *THIS IS A SUBROUTINE FOR DRAWING DOUBLE HEADED ARROWS
10004 *
10005 ******
10060 X=XN-XS
10070 Y=YN-YS
10080 L3=SQR((XN-XS)^2 + (YN-YS)^2)
10090 W1=W/2
10100 SI#Y/L3
10110 CS#X/L3
10120 IF AH#="FIX" AND W1<=1 THEN GOSUB 10500:GOSUB 10900:RETURN
10125 IF AH#="FIX" AND W1>1 THEN GOSUB 10500:GOTO 10300
10130 *THIS SECTION OF CODES CALCULATES THE VARIABLE SIZE OF THE ARROW HEADS
10140 *
10150 P=2
10160 X(P)=XS+9/10*X
10170 Y(P)=YS+9/10*Y
10180 X(P+1)=X(P)-Y/10
10190 Y(P+1)=Y(P)+X/10
10200 X(P+4)=X(P)+Y/10
10210 Y(P+4)=Y(P)-X/10
10220 X(P+5)=XS+X/10
10230 Y(P+5)=YS+Y/10
10240 X(P+6)=X(P+5)-Y/10
10250 Y(P+6)=Y(P+5)+X/10
10260 X(P+9)=X(P+5)+Y/10
10270 Y(P+9)=Y(P+5)-X/10

```

```

10280 IF DO$="D" AND W1<=1 THEN GOSUB 54000:GOSUB 10940:GOTO 11300
10282 IF DO$="D" AND W1>=1 THEN 10310
10290 IF W1<=1 THEN GOSUB 54000:GOSUB 10920:RETURN
10300 .
10310 'THIS SECTION OF CODE CALCULATES WIDE TAIL OF THE DOUBLE HEADED ARROWS
10320 .
10330 X(P+2)=X(P)-W1*SI#
10340 Y(P+2)=Y(P)+W1*CS#
10350 X(P+3)=X(P)+W1*SI#
10360 Y(P+3)=Y(P)-W1*CS#
10370 X(P+7)=X(P+5)-W1*SI#
10380 Y(P+7)=Y(P+5)+W1*CS#
10390 X(P+8)=X(P+5)+W1*SI#
10400 Y(P+8)=Y(P+5)-W1*CS#
10405 IF DO$="D" THEN GOSUB 54000:GOSUB 10940:GOSUB 11300:GOSUB 2270:RETURN
10410 GOSUB 54000:GOSUB 11100:RETURN
10420 .
10430 .
10500 .
10510 .
10520 'THIS SECTION OF CODE CALCULATES FIX ARROW HEAD
10530 H=12
10540 P=2
10550 X(P)=XS+(L3-H)*CS#
10560 Y(P)=YS+(L3-H)*SI#
10570 X(P+1)=X(P)-H*SI#
10580 Y(P+1)=Y(P)+H*CS#
10590 X(P+4)=X(P)+H*SI#
10600 Y(P+4)=Y(P)-H*CS#
10610 X(P+5)=XS+H*CS#
10620 Y(P+5)=YS+H*SI#
10630 X(P+6)=X(P+5)-H*SI#
10640 Y(P+6)=Y(P+5)+H*CS#
10650 X(P+9)=X(P+5)+H*SI#
10660 Y(P+9)=Y(P+5)-H*CS#
10665 IF DO$="D" THEN GOSUB 54000:GOSUB 10940:GOSUB 10300
10670 RETURN
10680 .
10690 .
10900 .
10910 .

```



```

10920 *THIS IS A PLOTTING SUBROUTINE FOR THE DOUBLE HEADED ARROWS
10930 PLOT X(P+5),Y(P+5),X(P),Y(P)
10940 PLOT X(P+1),Y(P+1),XN,YN
10950 PLOT XN,YN,X(P+4),Y(P+4)
10960 PLOT X(P+4),Y(P+4),X(P+1),Y(P+1)
10970 PLOT X(P+6),Y(P+6),X(P+9),Y(P+9)
10980 PLOT X(P+9),Y(P+9),XS,YS
10990 PLOT XS,YS,X(P+6),Y(P+6)
11000 IF FI$="F" THEN XMH=(X(P)+XN)/2:YMH=(Y(P)+YN)/2:GOSUB 55000
11010 IF FI$="F" THEN XMH=(X(P+5)+XS)/2:YMH=(Y(P+5)+YS)/2:GOSUB 55000
11020 RETURN
11030 *
11100 *
11110 *
11120 *THIS IS A PLOTTING SUBROUTINE FOR WIDE DOUBLE HEADED ARROWS
11130 PLOT X(P+2),Y(P+2),X(P+1),Y(P+1)
11140 PLOT X(P+1),Y(P+1),XN,YN
11150 PLOT XN,YN,X(P+4),Y(P+4)
11160 PLOT X(P+4),Y(P+4),X(P+3),Y(P+3)
11170 PLOT X(P+3),Y(P+3),X(P+8),Y(P+8)
11180 PLOT X(P+8),Y(P+8),X(P+9),Y(P+9)
11190 PLOT X(P+9),Y(P+9),XS,YS
11200 PLOT XS,YS,X(P+6),Y(P+6)
11210 PLOT X(P+6),Y(P+6),X(P+7),Y(P+7)
11220 PLOT X(P+7),Y(P+7),X(P+2),Y(P+2)
11230 IF FI$="F" THEN XMH=(X(P)+XN)/2:YMH=(Y(P)+YN)/2:GOSUB 55000
11240 RETURN
11300 *
11310 *
11320 *THIS SECTION OF CODE PLOTS DOTTED LINE FOR THE DOUBLE HEADED ARROWS
11330 IF W1<=1 THEN 11500
11340 XS=X(P+8):YS=Y(P+8)
11350 XN=X(P+3):YN=Y(P+3)
11360 XB=X(P+8):YB=Y(P+8)
11370 J=1:X(J)=X(P+8):Y(J)=Y(P+8)
11380 RETURN
11500 *
11510 *
11520 *THIS SECTION OF CODE PLOTS SINGLE DOTTED LINE FOR THE ARROW HEADS
11530 XS=X(P+5):YS=Y(P+5)
11540 XN=X(P):YN=Y(P)
11550 GOSUB 2200:RETURN
20000 *
20010 *
30000 *

```

```

54000 .
54010 .
54020 *THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT AND SET COLOR
54050 PRINT "M";
54060 PRINT CHR$(1);"C":CHR$(48+BC);
54070 PRINT "N";
54085 PRINT "G";
54087 PRINT "M";
54090 RETURN
54500 .
54510 .
54520 *THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT
54530 PRINT "G";
54535 PRINT "M";
54550 RETURN
55000 .
55010 .
55020 *THIS SUBROUTINE COMPLEX FILLS AN OBJECT
55030 PRINT "U": PLOT XM#.YM#
55035 PRINT "J";
55040 PRINT "M";
55050 PRINT CHR$(1);"C":CHR$(48+BC);
55060 PRINT "N";
55080 PRINT ">": CHR$(32);
55090 RETURN

```

*MOVE CURSOR TO COORD

*BACKGROUND LIGHT ON

*SET BACKGROUND COLOR

*BACKGROUND LIGHT OFF

*FILL OBJECT WITH SOLID COLOR

ALT.SRC

```

1160 '-----ALTERNATE TABLE INITIALIZATION-----
1170 SP$="0" 'SET SPEED FOR STREAM MODE
1180 PRINT CHR$(27);"R1C":CHR$(27);"OE5":CHR$(27);"IE5": 'DEVICE ASSIGNMENTS
1190 IF CK THEN PRINT CHR$(27);"Q": 'APPEND IF CREATE ON
1200 PRINT#4;"P": 'PUT IN POINT MODE
1210 ON ERROR#4 GOTO 1260 'SET JUMP TABLE FOR TABLET INTERRUPT
1220 RESUME 1230 'REENABLE INTERRUPTS: GOTO 960
1230 IF TF<>1 THEN 1230 'IF NOT DONE WITH INIT THEN LOOP FOR INTERRUPT ELSE NEXT
1240 GOTO 420 'IF DROP THROUGH, RETURN TO MAIN INTERRUPT HANDLER
1250 .
1260 IF ERR=24 THEN INPUT #4:X0,Y0,F$ ELSE ON ERROR #0 GOTO 0 'IF TABLET INTERRUPT THEN READ POINT
1270 PRINT CHR$(7): 'SOUND BELL TO INDICATE POINT ACCEPTED
1280 ON ERROR #4 GOTO 1310 'RESET JUMP TABLE FOR TABLET INTERRUPTS
1290 RESUME 'REENABLE INTERRUPTS: GOTO 960 (PLACE WHERE LAST INTERRUPT OCCURRED)
1300 .
1310 IF ERR=24 THEN INPUT #4:X1,Y1,F$ ELSE ON ERROR #0 GOTO 0 'IF TABLET INTERRUPT THEN READ POINT
1320 S1=(X1-X0)/512:S2=(Y1-Y0)/512 'SET SCALING FACTORS BASED ON X DIFF, Y DIFF
1330 PRINT CHR$(7): 'SOUND BELL TO INDICATE POINT RECEIVED
1340 PRINT #4:SP$: 'RESET PORT FOR STREAM MODE
1350 TF=1 'INDICATE INITIALIZATION COMPLETE
1360 RESUME 'REENABLE INTERRUPTS: GOTO 960 WHICH DROPS THROUGH AND GOES TO 420

```

SAMPLE SESSION 1

- I. GOALS: (1) To demonstrate the built-in primitives;
- (2) To demonstrate coordinate input in three modes;
- (3) To demonstrate state changes;
- (4) To demonstrate use of the create buffer.

II. GETTING STARTED

- Insert disk into disk drive with label side up and with label toward you, until the disk clicks and stays in place.
- Pull latch down to lock in the disk.
- Push "on" light on Chromatics and on disk drive, so that both lights appear lighted (refer to Figures 5 and 7).
- Push RESET, BOOT/WIDE VECTOR, BASIC/SLANT RECT. and RETURN. The last item on the screen should be a prompt in green which says, "OK".
- Push the reset button on the right side of the tablet to reset it to initial states (refer to Figure 9). (If you fail to do this you may get a "TYPE MISMATCH" error the first time you try to use the tablet.)
- Type DOS"LOAD PART1; push RETURN. (The machine will load the program and return with the prompt "OK".)
- Type RUN; push RETURN. (The machine will begin execution of the program and set initial conditions: the "plot key" should be lighted and all others unlighted while the cursor for plot. ".__", should appear at the upper left of the screen in home position.)

III. ENTERING COMMANDS

Push the following keys in the KEY column in the order given. The returning prompt and/or explanation is given in the MACHINE RESPONSE column. The SCREEN CHANGES, if any, are noted.

GOAL	KEY	MACHINE RESPONSE	SCREEN CHANGES
4	CREATE	create light blinks on, then off	
	BACKGROUND	background light goes on	
	BLACK	sets background color	
	BACKGROUND	background light off	
	ERASE PAGE	clears screen	clears screen
	GREEN	sets foreground color	
1	RECTANGLE		
2	010010510510	gives coordinates for (010,010), (510,510)	green rectangle
	<--	(notice split cursor is now whole)	
	↓		
1	RECTANGLE		
2	. (period)	sets coordinates of first corner	
	↓ (hold key down until near bottom)		cursor moves down until ↓ is released
	--> (4 times)		cursor moves right, then reappears at left of screen
	↓ (one or two times)		cursor moves inside previous rectangle
	. (period)	(sets coordinates of second corner)	2nd green rectangle

GOAL	KEY	MACHINE RESPONSE	SCREEN CHANGES
3	TABLET	create on; bell	
1	CIRCLE		cursor moves to new position
3	RED	(create light goes off as side effect)	
2	(Move the 4-button cursor over the tablet to desired center of circle; push top button)	("bell" indicates a hit with the button)	
	(Move the 4-button cursor over tablet to desired edge of circle; push top button)	("bell")	red circle
1	CONCAT VECTOR	("bell")	cursor moves to new position
	CYAN	(changes color)	
	(Hold top button down or push at intervals and move cursor across tablet)		concatenated vectors appear on screen
3	WHITE	(changes color)	
	(Continue holding cursor button down while moving)		concatenated vectors continue with new color
	(Push bottom button on cursor)	(Turns concatenated vector off)	cursor becomes stationary

GOAL	KEY	MACHINE RESPONSE	SCREEN CHANGES
1	CIRCLE		
3	YELLOW	(change color)	
3	FILL	fill light goes on	
2	(Move to desired center and push top button) (Move to desired edge and push top button)		filled yellow circle will overwrite any previous figure
3	FILL	fill light goes off	
1	DOT		cursor jumps to new position
	GREEN	(new color)	
2	(Hold top button down while moving cursor) (Push bottom button)	(Turns dots off)	trail of dots will appear; spacing depends on your speed cursor becomes stationary
1	X-BAR RED (Position cursor at left of any circle and push top button) (Position cursor at right of same circle and push top button)	(change color) (Select left end of horizontal vector) (Select right end of horizontal vector)	horizontal line appears across circle; horizontal lines are difficult from the tablet without X-BAR

GOAL	KEY	MACHINE RESPONSE	SCREEN CHANGES
	TABLET	("bell"; this turns tablet off.)	
1	VECTOR MAGENTA (Use cursor arrows to position for one end of vector)	(change color)	
2	. (Position cursor at other end of vector)	(selects coordinates)	
2	.	(selects coordinates)	magenta vector
4	REDRAW	(create buffer is dumped out)	screen recreates your selections
4	REDRAW	(you can do it again!)	screen recreates your selections

III. SAVING YOUR FILE

Push the BREAK key. You will get a prompt; if it is not on a new line push RETURN (ignore any "SYNTAX ERROR" messages at this point). Type DOS"BUFF fn where "fn" stands for filename, that is, any name you wish to give to your file. Then push RETURN. The information you stored in the create buffer should now be in the process of being stored on the disk. You can access your file at any time with the commands RESET, BOOT/WIDE VECTOR, DISK OS/DOTTED LINE, RETURN. You will get a prompt, "*". Then type DRAW fn (again using your unique filename). When another "*" prompt appears it means that your file is loaded into memory. Now simply push REDRAW and your selections will be recreated.

SAMPLE SESSION 2

I. GOALS: (1) To demonstrate use of point mode.

(2) To demonstrate blink mode,

(3) To demonstrate priority of overlapping regions,

(4) To demonstrate "erasing" by using background color,

(5) To demonstrate character size change.

(6) To demonstrate positioning of characters for create buffer.

II. GETTING STARTED

Securely fasten a copy of Figure 22b to the tablet (taping at opposite corners should be adequate). Figure 22b is an adjustment which allows for the distortion of translating tablet coordinates in a 1:1 ratio to screen coordinates in a 5:7 ratio; Figure 22a illustrates the desired result. Follow the sequence of directions given in Sample Session 1, part II.

III. ENTERING COMMANDS

Execute the commands given in the COMMAND column in the order given. Be sure to "turn off" the concatenated figures before trying to proceed. The RESULTS and/or EXPLANATION is given for each sequence.

GOAL	COMMAND	RESULTS/EXPLANATION
1	CREATE	(Initialize create buffer or not; background color picked up will be current machine setting for background when file is redrawn.)
	BLUE	Set foreground color
	TABLET	Turn tablet on
	POINT MODE	Select coordinates 1 pair at a time
	FILL	Print solid figures
	RECT hit #1 hit #2	Move 4-button cursor to position indicated on figure 16a; push top button for "hit". Blue rectangle appears
3	RED	Change foreground color
	RECT hit #3 hit #4	Red rectangle
	RECT hit #5 hit #6	2nd red rectangle
	RECT hit #7 hit #8	3rd red rectangle
	RECT hit #9 hit #10	4th red rectangle
	RECT hit #11 hit #12	5th red rectangle
3	WHITE	Change foreground color
	CIRCLE hit #13 hit #14	White circle
4	BLUE	Set foreground same as background
	RECT hit #13 hit #15	Removes part of circle

GOAL	COMMAND	RESULTS/EXPLANATION
	RECT hit #16 hit #17	Creates "S" by overlaying with background color
3	WHITE CIRCLE hit #18 hit #19	Change foreground color 2nd white circle
3	RECT hit #19 hit #20	Overlay bottom of "A"
3	RECT hit #21 hit #22	Overlay bottom of "A"
4	BLUE RECT hit #20 hit #23	Change foreground to background color Erase center of "A"; if any other white color remains, use another blue rectangle to erase it.
3	WHITE RECT hit #24 hit #25 RECT hit #25 hit #26	Reset foreground color First stroke of "L" Rest of "L" (if bottom uneven, X-BAR corrects)
	RECT hit #27 hit #28 RECT hit #28 hit #29 RECT hit #30 hit #31 RECT hit #32 hit #33	First stroke of "E" Bottom of "E" (X-BAR corrects unevenness) Middle of "E" Top of "E" (use X-BAR if necessary)
	FILL(off)	Turn fill light off

GOAL	COMMAND	RESULTS/EXPLANATION
2	BLINK	Blink light goes on
2	CONCAT VECT hit #34-40 bottom button X-BAR hit #40 hit #41 X-BAR hit #43 hit #34 CONCAT VECT hit #41-43 Bottom button	"Hit" each point in this sequence Portion of blinking star Bottom button turns concat off--MUST DO! Horizontal line for star Horizontal line for star Finish blinking star Don't connect to next point!
2	BLUE CONCAT VECT hit #44,45,... ...60,44 Bottom button BLINK(off)	Change foreground for center of "A" Small blinking star for center of "A"; be sure to hit unnumbered points on star. Turn concat off Blink light goes off
	TABLET(off)	Leave tablet; there is no machine response
	PLOT(off)	Leave plot mode; notice cursor change
5	MODE X2, MODE Y2,	Change character size; notice that cursor enlarges
6	HOME	Fixes cursor position for create buffer
6	↓ and -->	Move cursor with arrows until approximately at #54, just inside right edge of 1st red bar; use shift with arrow if necessary. (Positioning must be in create buffer for letters to print in correct location, and cursor movements are recorded in character mode only!)
	BACKGROUND BLUE BACKGROUND(off) BLACK	Set background for lettering so it is the same as the large blue rectangle behind the bars. Set foreground for letters.
3	(Type ...)	Type the letters: STARS & BARS

GOAL	COMMAND	RESULTS/EXPLANATION
	MODE K	This makes the cursor invisible
	BREAK	This interrupts the program and allows DOS commands to be entered.
	(Type ...)	Insert disk on which you wish to save file. Save buffer on your disk by typing: DOS"BUFF filename
	BACKGROUND BLACK BACKGROUND(off)	Reset the environment for a REDRAW command. The color set will be background for REDRAW.
	RESET BOOT RETURN DISK OS	Change to DOS environment
	(Type ...)	After prompt (*), type: DRAW filename
	REDRAW	Display the file

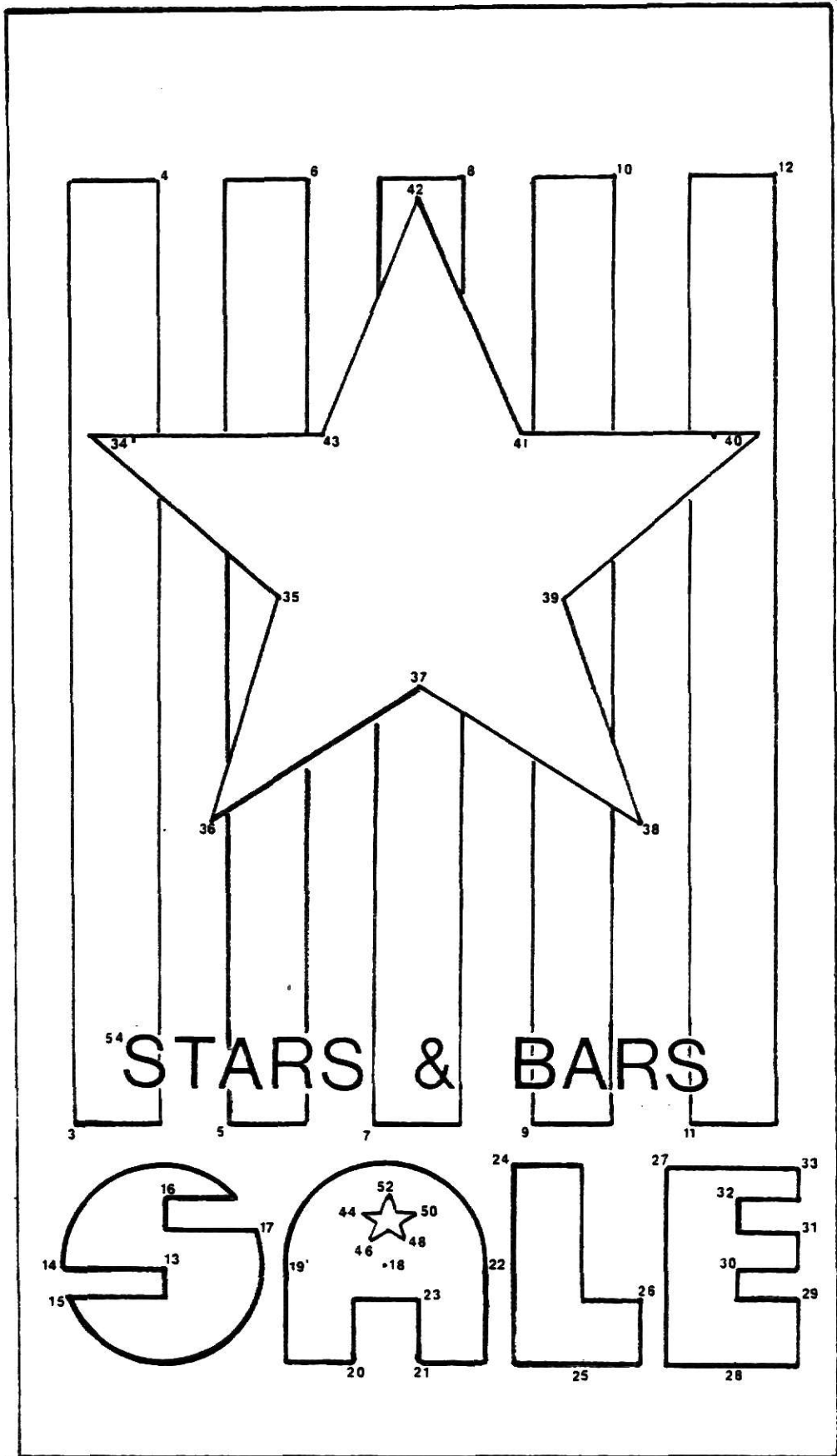


FIGURE 22a.
STARS & BARS

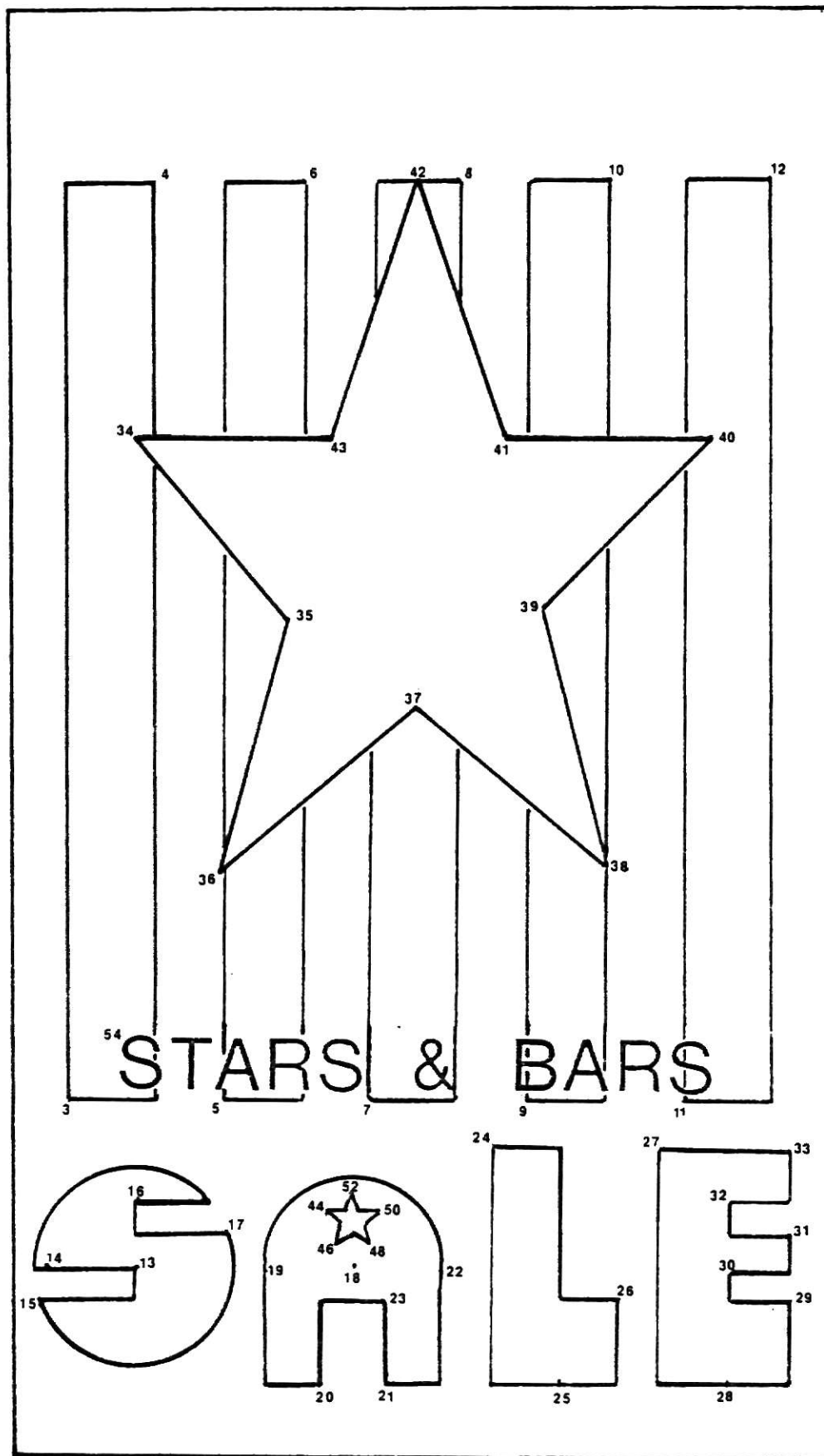


FIGURE 22b.

ADJUSTED STARS & BARS

SAMPLE SESSION 3

- I. GOALS: (1) To demonstrate use of the create buffer;
 (2) To demonstrate complex fill;
 (3) To demonstrate CURSOR X-Y command;
 (4) To demonstrate use of alternate character set.

II. GETTING STARTED

Follow the directions from Sample Session 1 to load the program and prepare for data entry. It will probably be helpful to refer to Figure 23 when positioning the cursor.

III. ENTERING COMMANDS

Push the following keys in the KEY SEQUENCE column in the order given. The results and/or explanations are noted.

GOAL	KEY SEQUENCE	RESULTS/EXPLANATIONS
1	CREATE BACKGROUND BLACK BACKGROUND(off) ERASE PAGE	Set Create Buffer on (These 4 steps are optional; background color is picked up from machine setting prior to REDRAW in any event!)
2	WHITE CIRCLE 128 170 030 CIRCLE 128 170 050 CIRCLE 128 340 030 CIRCLE 128 340 050 CIRCLE 255 170 030 CIRCLE 255 170 050 CIRCLE 255 340 030 CIRCLE 255 340 050 CIRCLE 384 170 030 CIRCLE 384 170 050 CIRCLE 384 340 030 CIRCLE 384 340 050	White "contains" all other colors and works well for Complex Fill White circle appears at lower left 2nd circle around the first White circle appears at upper left 2nd circle around last one White circle appears lower center 2nd circle around last one White circle appears upper center 2nd circle around last one White circle appears at lower right 2nd circle around last one White circle appears at upper right 2nd circle around last one

GOAL	KEY SEQUENCE	RESULTS/EXPLANATIONS
	RECT (position) ". " (position) ". "	Position cursor at upper right corner; position diagonally from first dot at lower left corner; rectangle results.
	RECT (position) ". " (position) ". "	Position cursor for border to first rectangle; result is 2nd rectangle.
3	CURSOR X-Y 128 170	Sends cursor inside first circle
2	BACKGROUND BLUE BACKGROUND(off) YELLOW	Turn on background; set background color; turn background light off; set foreground color.
4	CTRL N	Turn on the alternate character set
2,4	MODE > shift 0	Watch first circle fill!
3	CURSOR X-Y 128 230	Send cursor inside 2nd circle
2	BACKGROUND CYAN BACKGROUND(off) BLUE	Set background on; set background color; turn background off; set foreground color.
2,4	MODE > CTRL shift 0	Hold both ctrl and shift down, then press 0; border circle fills.
3	CURSOR X-Y 128 340	Send cursor to upper left circle
2	BACKGROUND BLUE BACKGROUND(off) CYAN	Set background on; set background color; turn background off; set foreground color.
2,4	MODE > shift N	Complex fills circle
3	CURSOR X-Y 128 400	Send cursor to border circle
2	MAGENTA	Use same background; reset foreground
2,4	MODE > shift Y	Border circle complex fills
3	CURSOR X-Y 255 340	Send cursor to upper center
2	BACKGROUND RED BACKGROUND(off) WHITE	Turn background on; set background color; turn background off; set foreground color.

GOAL	KEY SEQUENCE	RESULTS/EXPLANATIONS
2,4	MODE > shift P	Upper center circle fills
3	CURSOR X-Y 255 400	Send cursor to border circle
2	BLUE	Same background; reset foreground
2,4	MODE > shift J	Border circle fills
3	CURSOR X-Y 255 170	Send cursor to lower center
2	YELLOW	Same background; reset foreground
2,4	MODE > shift X	Lower center circle fills
3	CURSOR X-Y 255 230	Send cursor to border circle
2,4	MODE > shift L	Same colors; border fills
3	CURSOR X-Y 384 340	Send cursor to upper right
2	BACKGROUND GREEN BACKGROUND(off) BLUE	Turn background on; set background color; turn background off; set foreground color.
2,4	MODE > shift N	Upper right circle fills
3	CURSOR X-Y 384 400	Send cursor to border circle
2	RED	Same background; reset foreground
2,4	MODE > shift Q	Border circle fills
3	CURSOR X-Y 384 170	Send cursor to lower right
2	BLUE	Same background; reset foreground
2,4	MODE > shift S	Lower right circle fills
3	MODE U 384 230	Note alternate code to send cursor
2	WHITE	Same background; reset foreground
2,4	F8	Note: the 8 function keys are set for complex fill; they use the alternate character set and reset to the standard set automatically. (You may wish to try more of these keys although they are generally more suitable for subtle shading than for dramatic effect; F8 = exception.)

GOAL	KEY SEQUENCE	RESULTS/EXPLANATIONS
3	CURSOR X-Y 211 255	Send cursor inside rectangle
2	BACKGROUND BLUE BACKGROUND(off) CYAN	Turn background on; set background color; turn background off; set foreground color.
2,4	MODE > *	Note: if F8 had not turned off alternate character set, CTRL O would have been necessary before "*" would have printed in fill.
3	CURSOR X-Y 255 450	Check to make sure cursor is in rectangle border. Reposition cursor as necessary.
2	BACKGROUND GREEN BACKGROUND	Turn background on; set background color; turn background off; no foreground for solid fill.
2	MODE > (space)	When no character is specified the polygon fills with solid color.
	MODE K	Makes cursor invisible
	BREAK (Type)	Leave program to enter commands Type the command: DOS"BUFF filename and the buffer is saved.

The disk may be reinserted in the disk drives at any time and redrawn with the following sequence: DISK OS, (type) REDRAW filename, REDRAW. However, make sure the background is set to BLACK before doing the REDRAW.

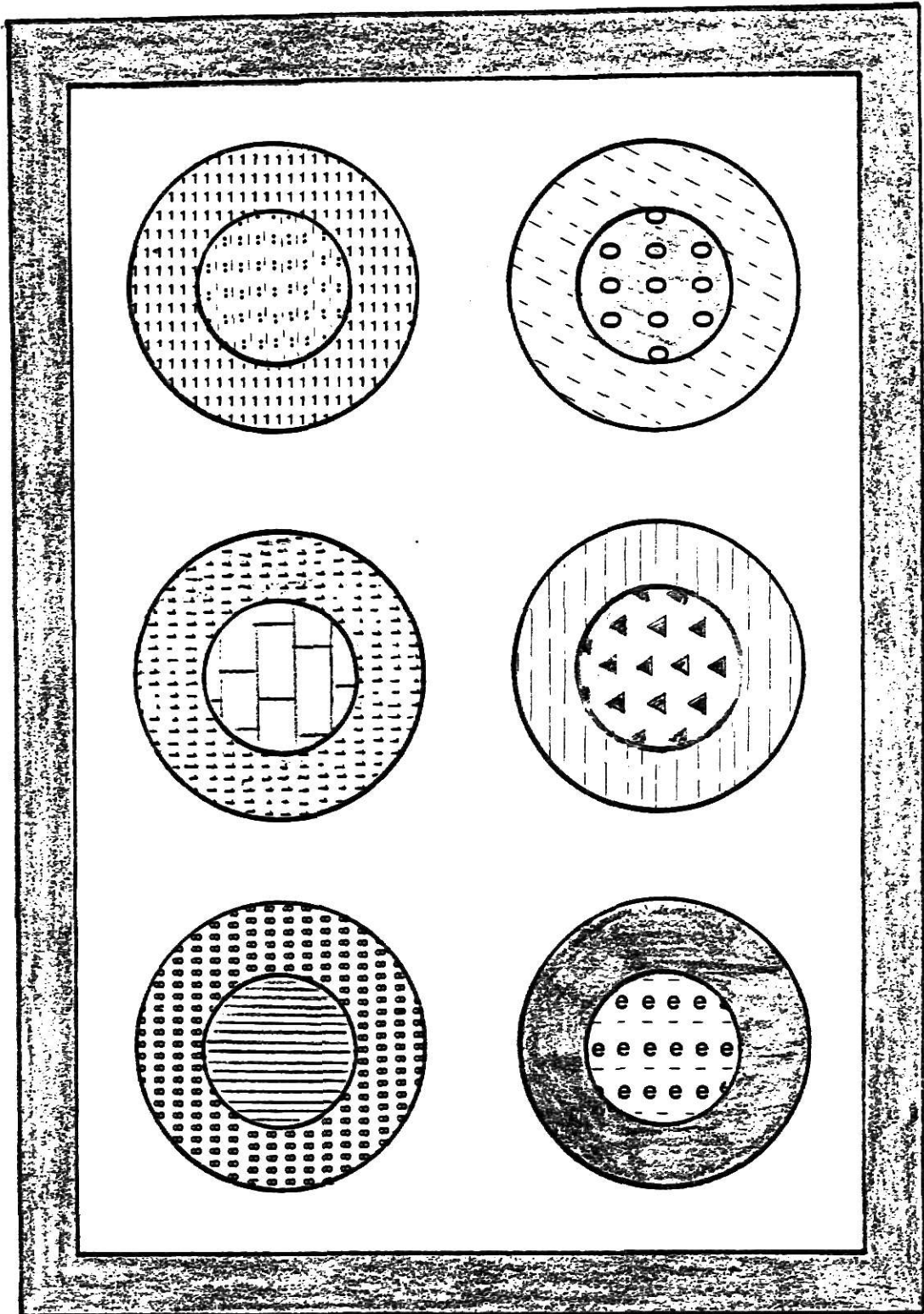


FIGURE 23.

CONCENTRIC CIRCLE COMPLEX FILL

SYSTEM DRIVER FOR COLOR GRAPHICS COMPUTER

by

MARILYN MCCORD DILLINGER

B.S., Kansas State University, 1961
M.S., Kansas State University, 1978

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1980

ABSTRACT

This report describes a system driver designed for the Chromatics CG (computer_graphics) 1999. The program allows the user to extend the performance of the Chromatics in two primary areas: input capability and extension of graphics primitives.

Using Chromatics commands under program control, the interface suppresses keys which would kill the program while redefining some keys for new commands.

Coordinate input for graphics primitives is accepted in any of four modes:

- (1) from the keyboard as digit coordinates,
- (2) from the keyboard as cursor position.
- (3) from the tablet. positioning from the four-button cursor on the tablet,
- (4) from the tablet. positioning from the cursor on the screen.

The program expands the set of graphics primitives available. While built-in primitives such as circles and rectangles are available on the Chromatics itself, extended primitives such as thickened lines and arrows are accessed from disk using an overlay.

A Users' Guide is provided along with sample programs and diagrams which illustrate the use of the driver. Source code and details related to programming options give opportunity for modification to suit individual applications.