

EDITING AND SEGMENTING
DISPLAY FILES FOR COLOR GRAPHICS

by

SHARLENE KAY MITCHELL

B.S., Kansas State University, 1968.
M.S., Kansas State University, 1971.

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
1981

Approved by:


Major Professor

SPEC
COLL
LD
2668
.R4
1981
M58
c.2

TABLE OF CONTENTS

	page
List of Figures	iv
Chapter One: Introduction	1
A. Overview.	1
B. Paper Organization.	1
C. An Explanation of the Create Buffer	2
D. Rationale for this Project.	5
1. Editing.	5
2. Segmenting	8
E. This Project in Relation to Current Developments.	8
F. Results and Evaluation.	10
G. The Future	12
H. Summary	13
Chapter Two: Users' Guide	14
A. The EDIT and TABLET EDIT Keys	14
B. The Segmenting Facility	16
C. Drawing Segmented Pictures Stored on Disk	20
D. Summary	23
Chapter Three: Implementation	24
A. Introduction.	24
1. Create Buffer Location and Size	24
2. Reading the Create Buffer.	26

3. Reading the Color Byte	26
4. Determining the Machine State.	29
B. EDITING the Create Buffer	30
1. Changes to the Original Driver	30
2. Variables and Flags.	32
3. Explanation of Code Sections in Editor	33
C. SEGMENTING the Create Buffer.	36
1. Changes to the Original Driver	36
2. Variables and Flags.	38
3. Explanation of Code Sections in Segmentor.	39
D. Summary	41
Appendices.	42
Appendix A (ANSI ASCII Chart).	43
Appendix B ("SEGDRAW" Program)	44
Appendix C ("Lister" Program).	45
Appendix D (Color Byte Chart).	46
Appendix E (Part1-Original Driver)	47
Appendix F (Part2-Original Driver)	48
Appendix G (Part3-Original Driver)	51
Appendix H (Part4-Original Driver)	54
Appendix I (Extended Part2-With Editor Shaded)	58
Appendix J (Extended Part2-With Segmentor Shaded).	64
Bibliography.	70

LIST OF FIGURES

	page
One: Relationship of User, Chromatics, and the Programmed Driver	2
Two: Entries in Create Buffer which Produce the Picture in Figure 3	3
Three: Picture Produced by Commands in Figure 2.	4
Four: Example of Holes Which Develop in Drawings	7
Five: Examples of Extended Primitives.	9
Six: Keyboard Layout	15
Seven: Pseudocode for Editing Capability	17,18,19
Eight: Pseudocode for Segmenting of Display File	21
Nine: Pseudocode for REDRAWing Segments.	22
Ten: Diagram of Memory Organization.	25
Eleven: Diagram of Locations of Color Byte and Pointers to Create Buffer	27
Twelve: Diagram of Color Byte and What Each Bit Represents.	28
Thirteen: Diagram of Color Byte AND &H55 Results	29
Fourteen: Diagram of Special Function Keys with Corresponding Chromatics Input Number	31
Fifteen: Built-In Primitive Recognized by Chromatics	35
Sixteen: Default Conditions of the Screen.	37

CHAPTER ONE: Introduction

A. OVERVIEW

This report describes two extensions of a system driver designed for the color graphics computer in the Computer Science Department. One extension allows the user to edit drawings or text as they are being developed. The second extension allows the user to create complex pictures which previously could not be done because of limited buffer space. This is accomplished by segmenting the create buffer and maintaining these segments on disk storage.

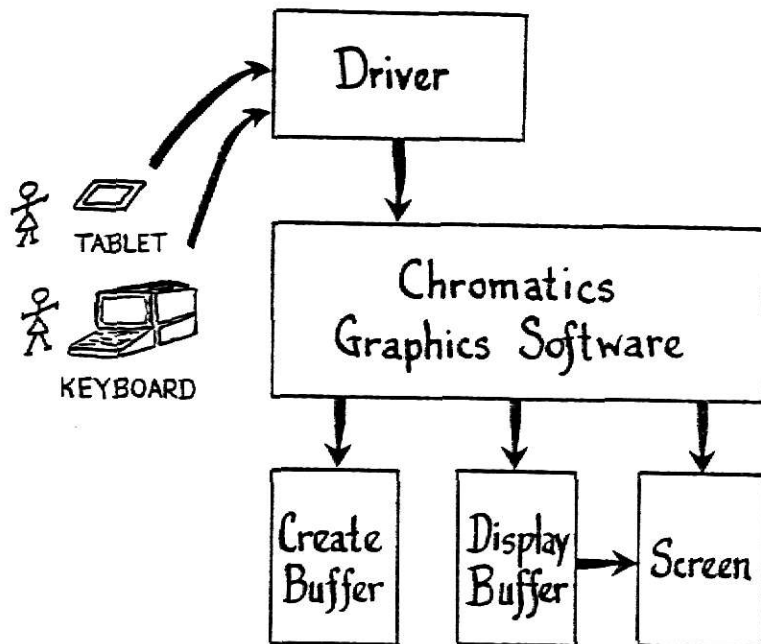
B. PAPER ORGANIZATION

This first chapter is an introduction which explains the rationale for the project, gives a short tutorial on what a create buffer is, explains where this project fits into current developments here at Kansas State University, and gives an overview of what was accomplished with its results and an evaluation based on how well the implementation works. The second chapter includes all information necessary for the person who wants to use the editing and/or segmenting portions of the driver developed for the Chromatics color graphics microcomputer. The third chapter provides the internal design to either duplicate the editing and segmenting processes or to modify them to include windows

other than 0. Appendices include the source code for the editing and segmenting sections within the system driver, and other materials that support or expand information found in the text of the paper.

C. AN EXPLANATION OF THE CREATE BUFFER

As shown in Figure 1, the user enters commands through the keyboard and possibly the tablet. The commands are first handled by the system driver originally developed by Dillinger(Dil80)



which calls on the Chromatics routines. The display file

FIGURE 1: RELATIONSHIP OF USER, CHROMATICS, AND THE PROGRAMMED DRIVER

and the refresh memory can each receive the picture developed by the user. The display file is developed in the Chromatics CG1999 when the CREATE key has been turned "on." The commands which include all characters, mode codes, control characters, and plot submodes entered into the terminal are placed in the display file (create buffer is the

**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.**

term used by Chromatics). Chromatics uses ANSI ASCII code to represent the input. If the display file is saved, the picture can be recreated by stepping through the command sequence in the create buffer. Figure 2 provides a short example of entries in the create buffer when the user has pressed a sequence of keys to produce the drawing in Figure 3. A translation from ASCII to English is provided to help the user readily understand the input. Appendix A provides the complete ANSI ASCII chart.

1	mode	roll off
80	P	
1	mode	plot on
71	G	
1	mode	background on
77	M	
1	mode	blink off
50	2	
1	mode	white
67	C	
55	7	
1	mode	background off
78	N	
1	mode	black
67	C	
48	0	
12	erase page	
42	*	circle
50	2	
53	5	x-coordinate
54	6	
50	2	
53	5	y-coordinate
54	6	
48	0	
53	5	radius
48	0	
8	BS	back space
8	BS	back space
8	BS	back space
21	mode cancel	change to character
67	C	
73	I	
82	R	
67	C	
76	L	
69	E	
30	EOF	end of file

Column 1 gives the numbers found in the create buffer.

Column 2 gives the translation from ANSI ASCII (refer to Appendix A for further ANSI ASCII).

Column 3 gives added interpretation.

The horizontal lines divide the entries into separate commands. (i.e. by pressing the 'WHITE' key-- 3 entries were made).

FIGURE 2: ENTRIES IN CREATE BUFFER WHICH PRODUCE THE PICTURE IN FIGURE 3

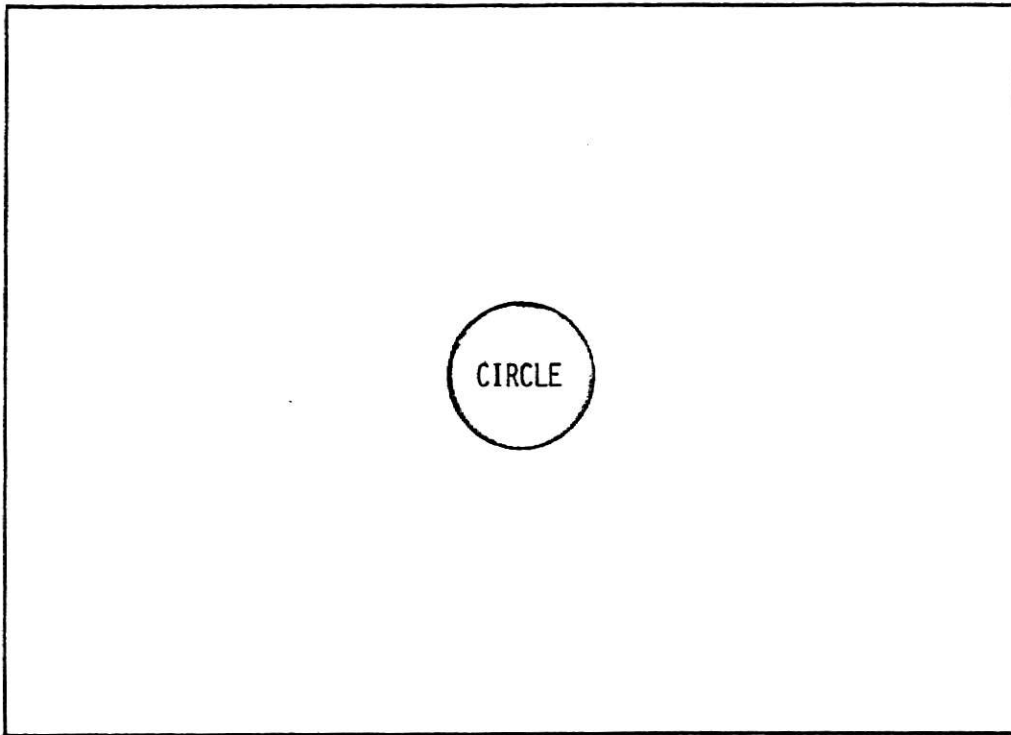


FIGURE 3: PICTURE PRODUCED BY COMMANDS IN FIGURE 2

The refresh memory has four entries (blink/non-blink, red, blue, and green) for each dot on the screen. The screen is 512 dots by 512 dots (262,144 dots) in size. Because of its speed, the refresh memory is used by the machine to constantly replace the picture on the screen so the viewer does not see any decay of the picture (New79). Obviously the create buffer will take much less space in storage if the user wants to save the picture to be shown later.

If the user has turned CREATE "off," an end-of-file (EOF) marker is placed at the end of the commands in the create buffer. The picture can be continued by APPENDING to it. The extended system driver has simplified this task, the user merely presses the APPEND key. If the user should press the CREATE key again, the pointers are reinitialized and the create buffer starts at the beginning overwriting the previous picture.

D. RATIONALE FOR THIS PROJECT

1. Editing

Often in developing a picture, people have problems with misspelled text, the incorrect positioning of text or geometric figures, or unexpected results due to incorrect commands. With our current machine capabilities, the only options open to the user to correct the mistake are:

- 1) to start over, which can be extremely disheartening, especially in a detailed picture;

- 2) to remove the object or characters by blocking the area out by filling it in with background color, which works as long as there is no overlap of objects. However, as the picture is reproduced at a later time, the mistake and the cover up have been entered into the create buffer and are shown. The final "static" picture does not show the mistake but it is not possible to do away with lines or objects which overlap.

It would be extremely valuable to the users to be able to edit their input. Two kinds of editing are needed:

- 1) the ability to remove any figure(s)/character(s) from the picture. At this time, this particular capability is beyond the scope of this project.
- 2) the ability to remove the most immediate (last) entered figure/character one at a time as far back as the user wants to go. This is part of what this project does.

The editing capability needs to be easy to use so the novice user can have the capability readily available to him/her. One key needs to be defined for the editing function. By pushing the one key the user can remove the previous character/figure from the screen by drawing over it in the background color and removing the instructions from the create buffer. By pushing the key repeatedly, many character(s)/figure(s) can be removed. If any abnormalities develop in the design (see Figure 4) the user can press the REDRAW key built into the machine. When the REDRAW key is pressed, the screen is reinitialized and the picture which is stored in the create buffer is drawn on the screen. By having removed the character(s)/figure(s) from the create buffer, the picture is what it was before the last item(s) was entered.

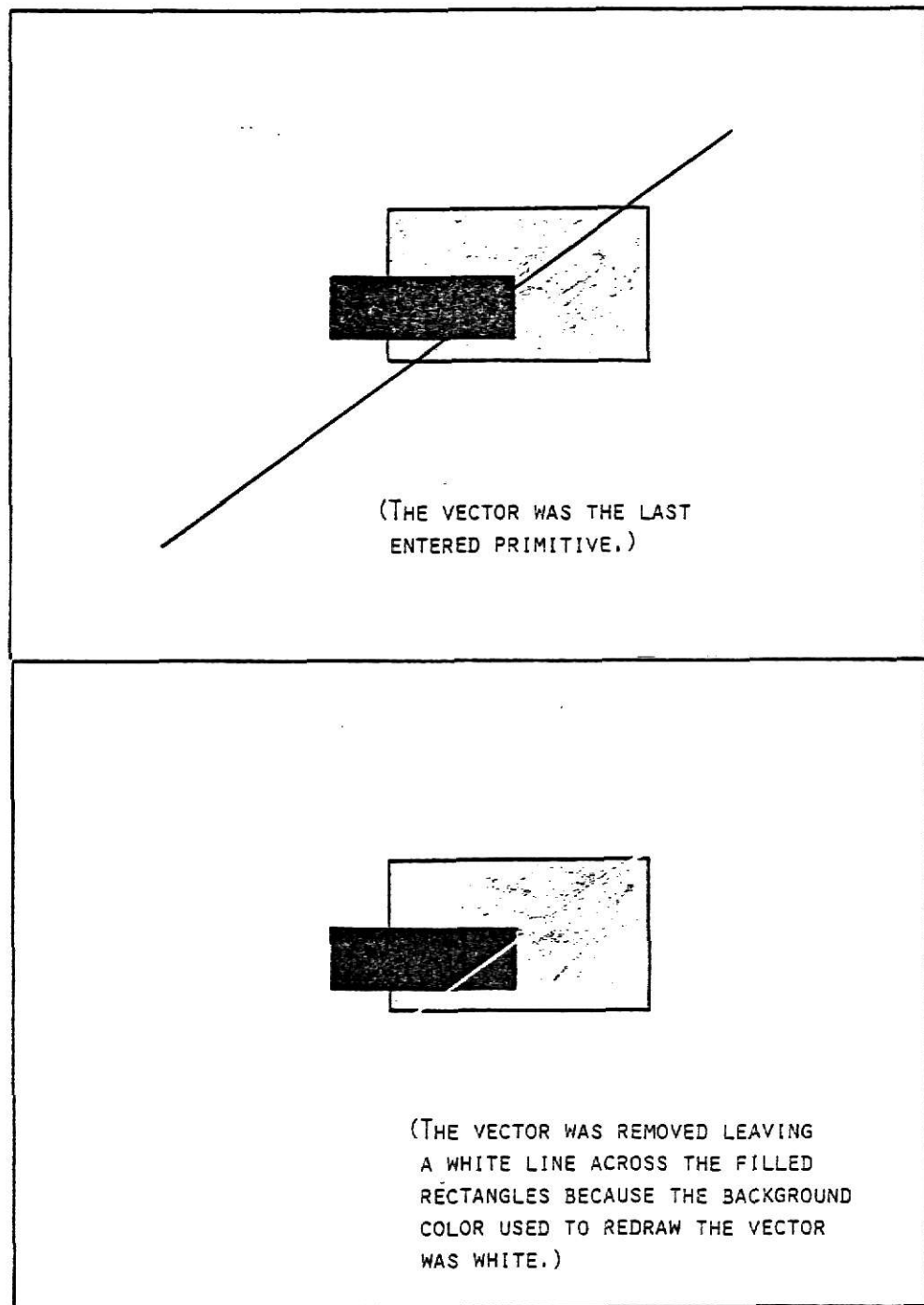


FIGURE 4: EXAMPLE OF HOLES WHICH DEVELOP IN DRAWINGS

2. Segmenting

Another problem encountered by users is that the create buffer has limited size. Only 16K ASCII characters can fit at any one time. In making detailed pictures, especially if a BASIC program is already taking up part of the memory space, the memory locations are not available and an "out of memory error" results. As an example, Chromatics supplied several sample pictures with the machine. Most of them are very complex and will not fit into the create buffer if the BASIC interpreter has been entered into memory. Therefore, it is not possible to write a program having the samples shown in sequence. Likewise, if one desires overlays, movement, or stream input from the tablet, the space fills quickly. The solution to this was to have the machine maintain segments of the create buffer on disk space and then read through the segments as needed when producing the picture, thus extending the length of the create buffer to the number of segments which will fit on a disk.

E. THIS PROJECT IN RELATION TO CURRENT DEVELOPMENTS

A driver developed by Dillinger (Dil80) for the Chromatics CG 1999 has made many options available to the user in a much less complex manner. Using Chromatics commands under program control, the interface suppresses keys which would put an end to the program while redefining some keys for new commands. The coordinate input for graphics primitives is accepted in any of four modes: 1) from the keyboard as

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

Included within the driver is an interface with a program developed by Yee (refer to Part3 and Part4 in Appendices G and H) which extends the primitives available to include: wide vectors, arrows, bubbles, thickened rectangles, dotted lines, double headed arrows, and slanted rectangles (see Figure 5). This program is a further extension of the driver to include editing and segmenting of the create buffer.

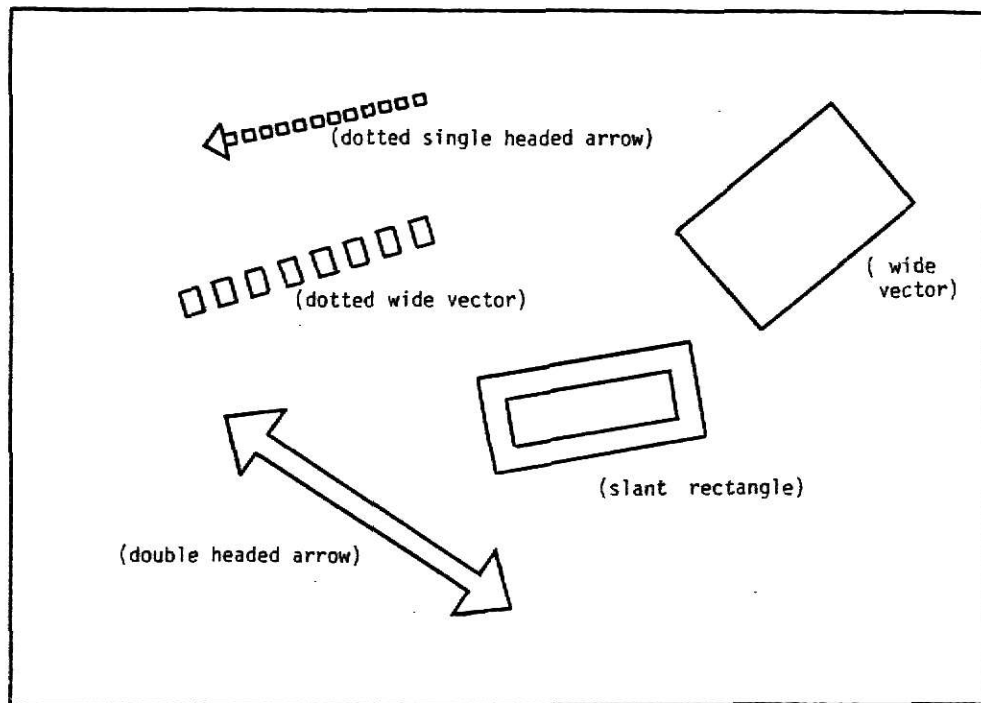


FIGURE 5: EXTENDED PRIMITIVES

F. RESULTS AND EVALUATION

This project was chosen because I had taken the graphics course offered by the Computer Science Department at Kansas State University and readily recognized the need for the additions because of personal experience. The project was a learning experience because of several aspects; working with something already developed to make changes or extensions, working with reference manuals with errors and misleading information, and making the extension work in spite of the unforeseen obstacles.

It became quite evident that working with an already developed piece of software was a challenge especially when the editing extension was not planned for when the original driver was developed. The original software was written with concern for the product on the screen whereas the editing extension deals with both the product on the screen and with the manipulation of the display buffer. The tablet input compared to keyboard input is an example because a primitive entered as tablet input puts the primitive symbol in the create buffer twice whereas if entered through the keyboard the symbol is put in the create buffer only once. Had the editor been part of the original plan one edit key might have been enough.

The reference manuals (Chr78a), (Chr78b), (Chr79) had a great deal of information, but they were not written for the beginner. Also, some of the information is wrong or misleading. The Chromatics model owned by Kansas State University would not support the MOD function. The STR\$ function placed a blank in front of the number which was being converted to a string. The INPUT statement would not enter the filename in the

segmentor. The machine would lock and could only be restored by starting over and re-entering the BASIC program thereby losing the picture.

When the first code was written for the editor, it was naively short. Not all the "what if's" were taken into consideration, and there are many. The editing code started out as a small section tacked onto the bottom of the driver. It is now almost as large as the driver itself.

By testing the extensions with five novice users, four of whom had not previously seen the Chromatics machine, the extensions were found to be fairly robust. Very little trouble was encountered in removing character input. The editor takes cursor movements into consideration and compensates for them. The one novice user who had used the Chromatics in a very limited capacity had learned to draw pictures by changing the background color and blocking out spaces with the space bar. Since the editor treated the cursor movements as space finders, it could back over an entire picture formed as previously mentioned without finding a primitive or character. This was easily changed by making the space bar appear to be a character. In many cases when instructions are input which cannot be counteracted, the editor prints a message to press REDRAW. When removing figure input, it was not possible to make all editing successful with overdraws so it was necessary to put in statements telling the user to press REDRAW. Not all of the situations could be specified, so the user must be aware of what he/she has done and press REDRAW if they have changed states or made movements which may not be recorded in the display file. To be safe, the novice should do a REDRAW whenever in doubt. It is not possible to back up for editing

purposes past the segment boundaries, so it is extremely important for the user to be correct at those boundaries. When using tablet input, it is necessary to turn the tablet "off" before editing. A message had to be added to tell the user to do so if they forgot. If the user presses the wrong edit key, the changes may not be what is expected but the user can recover easily by pressing REDRAW.

The segmentor works well in the environment of the driver; however, the system will not allow duplicate file names for BUF files. It is very important for the user not to duplicate a previous name when naming their file. Another inconvenience arises because a separate program is needed to DRAW the completed picture on the screen.

G. THE FUTURE

There are several further extensions which would be helpful. The most obvious need is the ability to perform DOS (Disk Operating System) commands from within the driver. This would provide the ability to DRAW the segmented files from within the driver without the need of an extra program or the need to know how many segments there are in a picture. To accomplish this task, it would be necessary to put a "header" at the beginning of each segment indicating whether or not another segment follows it. Also the DOS command would need to be implemented by defining a DOS key and collecting the necessary information to be able to carry out the command, i.e. DOS"DRAW XANADU.

Another extension would be to trap errors such as the error of

naming a file the same name as one already on the disk. Another need is the ability to segment files already on the disk so they can be shown with the BASIC interpreter and program in memory.

One immediate problem is the lack of space. The current driver is on the verge of using all the available memory space. The example of the file in the Appendices I and J with numerous comments will not run because an "out of memory" error will occur. Additional memory could be obtained by stripping out all comments and leaving out all blanks, but that would make the program extremely difficult to read, especially for someone who had not originally developed the program trying to extend or modify it.

H. SUMMARY

This chapter defined the problem and showed the need for editing and segmentation. Since the term "create buffer" is used throughout the paper, a short tutorial was given to explain what it is and to show examples of what is contained in it. One section showed where this project fits into current developments at Kansas State University. Another section provided an explanation and an evaluation of the results. Finally, some suggestions for further extensions which would be helpful to users were given. In the next chapter, the user will be given specific instructions which are necessary to use the previously described extensions.

CHAPTER TWO: Users' Guide

To draw pictures or enter text with the driver, refer to the User's Guide, Chapter 2, "System Driver for Color Graphics Computer" by Dillinger (Dil80). The following sections explain how to use the editor and segmentor.

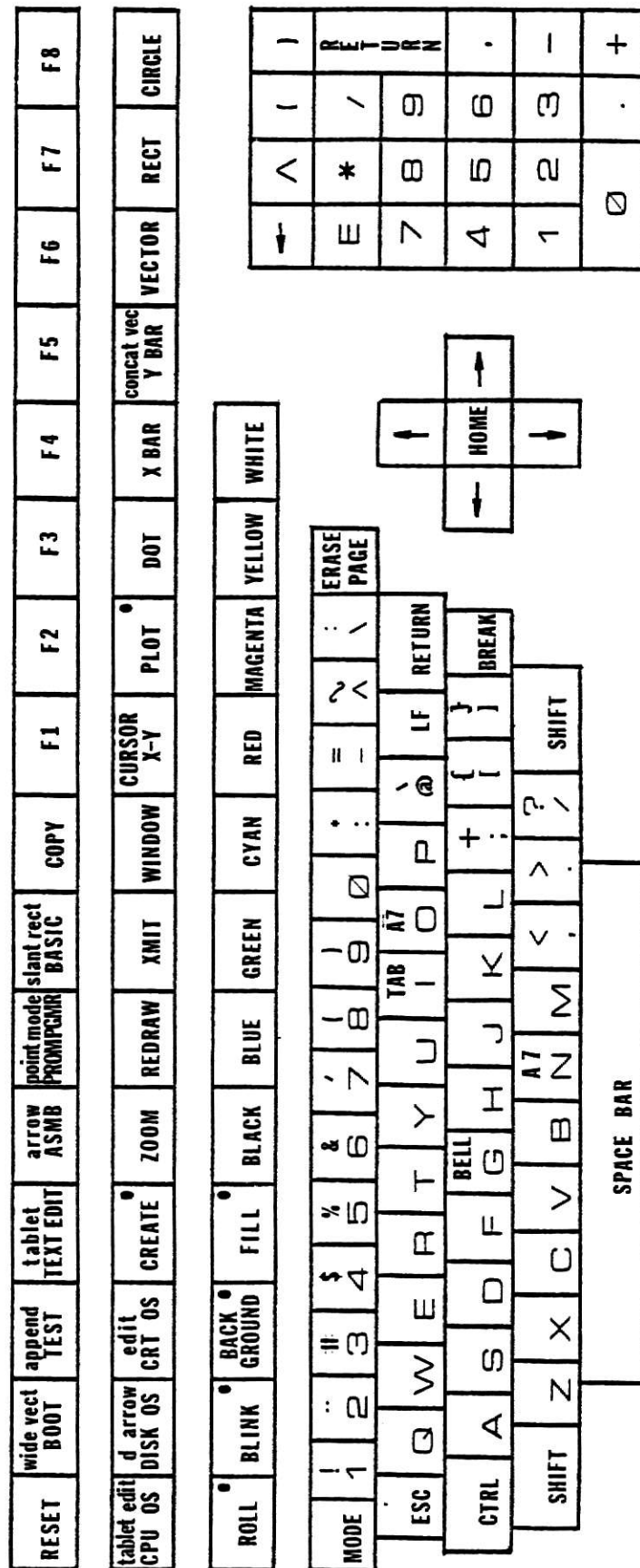
A. THE EDIT AND TABLET EDIT KEYS

The key CRTOS has been redefined to be an EDIT key and will now do an overdraw of the last entered character/figure if one is using keyboard rather than tablet input. The user merely presses this key (see Figure 6). If the user wants to remove the next previously entered character/figure, the user presses the key again, and so on.

To be able to use this extension, be sure to enter the primitive command each time you enter the coordinates of a primitive because the editor goes backwards through the entries in the create buffer until it finds the entry which designates a primitive. If you have entered numerous circles without re-entering the CIRCLE command, the editor will erase the first circle entered after the CIRCLE command from the screen but will have removed all subsequent circles from the create buffer. If this happens, press REDRAW (see Figure 6) to bring the picture back to what is in the create buffer.

There are several instances when it is not readily possible to

KEYBOARD LAYOUT



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

FIGURE 6: KEYBOARD LAYOUT

overdraw the figure/character on the screen. In these cases a message will appear on the screen telling the user to press REDRAW. If you have changed states (blink, fill, background, etc.) or color, changed from plot to character input, or produced holes in your picture, press REDRAW to get the picture to correspond with the entries in the create buffer.

The key CPUOS has been redefined to be a TABLET EDIT key (see Figure 6). To edit tablet input, turn the tablet "off" by pressing the TABLET/TEXT EDIT key, then press the TABLET EDIT key. If you have been entering concatenated vectors or dots through stream mode, more will be removed from the create buffer than shows on the screen so be sure to REDRAW the picture to correspond with the create buffer. To continue drawing with the tablet, press the TABLET key and continue.

Structured pseudocode of the action taken each time the EDIT/TABLET EDIT keys are pressed are represented in Figure 7. Figures 7A and 7B are expansions of sections in Figure 7.

B. THE SEGMENTING FACILITY

The ability to segment the create buffer will not be needed by most users, and it will require no user input unless the need to use it arises. Shortly before the create buffer runs out of space, a message will appear on the screen asking the user to enter a name for the picture (file). The user need only type in a name by following these

```
STOP THE CREATE BUFFER ENTRIES
FIND TOP AND BOTTOM OF PICTURE IN CREATE BUFFER
  IF BOTTOM = TOP
    THEN RETURN TO WAITING FOR INPUT
  FI
DETERMINE COLOR STATUS
CHANGE FOREGROUND COLOR TO BACKGROUND COLOR
  IF IN PLOT MODE
    THEN LOOK THROUGH THE CREATE BUFFER STEPPING
      BACKWARDS UNTIL A PRIMITIVE OR THE TOP
      IS REACHED (ALONG THE WAY CANCEL OUT
      COLOR CHANGES)
    IF BOTTOM NOT EQUAL TO TOP
      THEN REPRODUCE THE LAST PRIMITIVE
    FI
  ELSE REPRINT LAST CHARACTER
  FI
RETURN COLOR TO ORIGINAL SETTING
PUT BOTTOM OF CREATE BUFFER POINTER BACK TO WHERE
LAST PRIMITIVE OR CHARACTER STARTED
RESTART THE PICTURE AT NEW BOTTOM
```

FIGURE 7: PSEUDOCODE FOR EDITING

```

WHILE TOP NOT EQUAL TO BOTTOM AND BOTTOM NOT
EQUAL TO PRIMITIVE
    START AT BOTTOM AND PEEK AT EACH LOCATION
    IF CONTENTS OF BOTTOM NOT EQUAL TO A PRIMITIVE
        THEN
            IF CONTENTS EQUAL MODE
                THEN PEEK AT BOTTOM + 1
                    IF CONTENTS = C
                        THEN POKE NULLS AT BOTTOM,
                            BOTTOM+1, & BOTTOM+2
                    FI
                FI
            FI
            DECREMENT BOTTOM
        FI
    ENDWHILE

```

FIGURE 7A: EXPANSION OF PSEUDOCODE FOR (LOOKING THROUGH THE CREATE BUFFER STEPPING BACKWARDS UNTIL A PRIMITIVE OR THE TOP IS REACHED -- ALONG THE WAY CANCEL OUT ANY COLOR CHANGE)

```
WHILE LAST ENTRY WAS A MODE CHANGE  
    IGNORE IT AND KEEP GOING BACK  
ENDWHILE  
IF BOTTOM NOT EQUAL TO TOP  
    THEN  
        BACKSPACE  
        REPRINT CHARACTER  
        BACKSPACE  
FI
```

FIGURE 7B: EXPANSION OF PSEUDOCODE FOR (REPRINT
LAST CHARACTER)

rules:

1. Names should begin with an alphabetic character.
2. Length of name must be no greater than 6 characters.
3. Use only alphanumeric characters.
4. If the user chooses a name previously used as a file name on the user's disk, an error will occur and the picture will be lost.

After typing in the name, press RETURN. At that time the driver will take care of saving the first segment for the user and will save the following segments automatically as the create buffer refills. See Figure 8 for the pseudocode for saving the segments.

The REDRAW key has been trapped and takes care of putting the segments in the create buffer in the proper order and redraws each segment in sequence. See Figure 9 for the pseudocode for REDRAWing the segments.

C. DRAWING SEGMENTED PICTURES WHICH ARE STORED ON DISK

The user must know the name of his/her file and how many segments it has. To check the directory:

```

user: press RESET, press BASIC
machine response: Memory size?
user: press RETURN
machine response: Chromatics DISK BASIC Ver 3.0
                  Copyright (C) 1978 by Microsoft
                  1191 Bytes free
                  ok
user: type DOS"DIR/n (n is disk drive number)
user: press RETURN
machine response: (the disk directory is produced
                  on the screen)
```

```

WHEN CREATE FLAG IS TURNED "ON" SET FLAG TO INDICATE
THIS IS NOT A SEGMENTED FILE, SET SEGMENT COUNT TO 1
WHEN CREATE FLAG IS TURNED "OFF" BY USER, IF SEGMENT
COUNT IS GREATER THAN 1 THEN GO TO THE STORE SEGMENT SECTION
CHECK FOR MAXIMUM SIZE BEFORE EACH RPIMITIVE/CHARACTER
IS ENTERED INTO THE CREATE BUFFER IF GETTING NEAR THE
END THEN GO TO THE STORE SEGMENT SECTION

PUT EOF AT END OF FILE
IF SEGMENT = 1
    THEN
        MOVE CURSOR TO TOP OF PAGE
        SET CONDITIONS SO IT WILL PRINT CHARACTERS
        PRINT "CREATE BUFFER SEGMENTING -- If you wish
            TO CONTINUE, ENTER FILENAME"
        STORE SEGMENT WITH FILENAME
    ELSE
        STORE SEGMENT FILENAME(SEGMENT#)
FI
INCREMENT SEGMENT #
START AT TOP OF CREATE BUFFER AND GO ON

```

FIGURE 8: PSEUDOCODE FOR SEGMENTING OF DISPLAY FILE

```
IF FILE HAS NOT BEEN SEGMENTED
  THEN
    REDRAW
  ELSE
    STORE CURRENT SEGMENT UNDER FILENAME(SEG #)
    LOAD FIRST SEGMENT OF FILENAME
    REDRAW FIRST SEGMENT
    SET COUNTER WHICH WILL ACT AS SEGMENT NUMBER
    WHILE FILENAME(COUNT) IS FOUND
      LOAD FILENAME(COUNT)
      REDRAW FILENAME(COUNT)
      INCREMENT COUNT
    ENDWHILE
  FI
RETURN TO WAIT FOR NEW INPUT
```

FIGURE 9: PSEUDOCODE FOR REDRAWING SEGMENTS

You will see your filename and if it has been segmented it will also contain file(s) with number(s) preceding your filename.

```
Example: XANADU .BUF
         2XANADU .BUF
         3XANADU .BUF
```

You must also have SEGDRAW.BAS included in your directory. If it is not in the directory see Appendix B for code.

Assuming you are still in BASIC, the following interaction results in your picture being drawn on the screen.

```
user: type DOS"LOAD SEGDRAW
user: press RETURN
machine response: ok
user: type RUN; press RETURN
machine response: WHAT IS YOUR FILENAME?
user: type filename, press RETURN
machine response: HOW MANY SEGMENTS?
user: type number of segments;
press RETURN
machine response: (your picture will be
drawn on the screen)
```

D. SUMMARY

This chapter was designed as a users' guide. It explained how to use the editor and segmentor extensions. Pseudocode describing the editor and segmentor was provided to show the flow of logic in producing these extensions. The last section which told how to draw segmented pictures was needed because the stored pictures cannot be drawn from within the driver. The next chapter will give a more detailed explanation of how the editor and segmentor were added. It was written for the person who might want to duplicate or modify the extensions.

CHAPTER THREE: Implementation

A. INTRODUCTION

To be able to implement this project, a number of questions were addressed, several methods of reading input and output were learned, and Chromatics, Inc. (Chr80) was consulted for further documentation of the system. In the first section of this chapter, several questions and the information received from Chromatics are explored. Sections B and C describe the actual implementation.

1. Create Buffer Location and Size

One of the main purposes for developing the original driver was to make the Chromatics CG 1999 as easy for the novice user to operate as possible so the default command of carriage return is used when the machine asks for "memory size." In the default case, the maximum amount of space for the BASIC program is from &H43C0 to &H7FFF or 16K. This means that the create buffer will always begin at &H8000. Whether it starts at &H8000 or not, the create buffer will always end at &H7FFF, see Figure 10. In the editing extension the address of the top of the create buffer is read in case a user changed the memory size. Since the end of the create buffer will always be at the same place whether or not the starting point changes, a fixed address is used to determine how full the create buffer was getting.

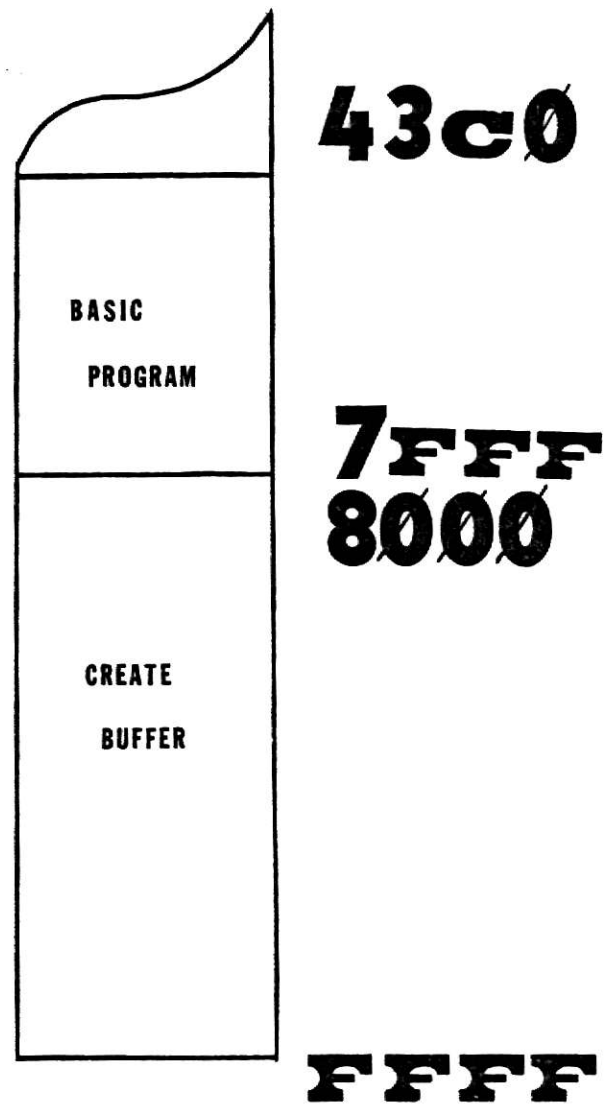


FIGURE 10: DIAGRAM OF MEMORY ORGANIZATION

2. Reading the Create Buffer

Being able to read what is in the create buffer is very important to the person trying to edit the create buffer. To be able to remove anything, it is first necessary to determine what is there. This kind of information is not readily available in the reference manuals. Also as the editor is being debugged, it is necessary to be able to see what you have changed or removed.

To read what is in the create buffer it is necessary to find out where it starts and where it ends. The starting location for window 0 is at 3B46 and 3B47 (see Figure 11). When PEEKing at those locations the operator finds an integer in each position. To translate these numbers into an address, multiply the contents of 3B47 by 256 then add the contents of 3B46. Likewise, to find the ending location of that picture for window 0 in the create buffer, one PEEKs at locations 3B44 and 3B45.

After finding the beginning and end of the picture, the operator can write a program which will PEEK at each location in the create buffer, translate it from ASCII code to English, then print out each of those entries. Appendix C is a program entitled "LISTER" written especially to do this process. As seen before, Figures 2 and 3 show an example of a picture and its list of what was in the create buffer.

3. Reading the Color Byte

One must be able to determine what colors are being used so the "overdraw" of the character/figure can be done in the background color. This will save time in drawing as long as holes do not develop in the

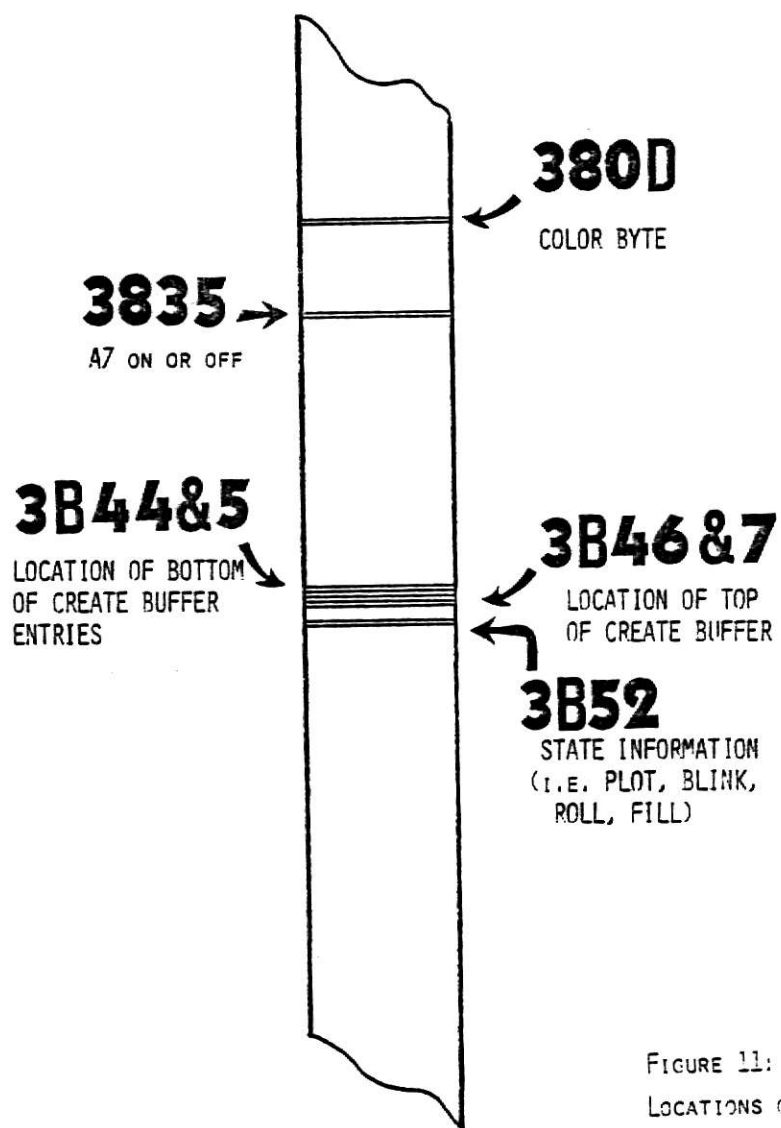


FIGURE 11: DIAGRAM OF
LOCATIONS OF COLOR BYTE
AND POINTERS TO CREATE
BUFFER

picture. After the "overdraw" has been done, the color setting will automatically be restored to what it was before the editing was done. If color changes were made during the last entry which was removed, the color either needs to be set to what the user wants to go on with or the REDRAW key pressed. Otherwise, the picture will continue on the screen in the color set from the color which has been removed but the picture in the create buffer will be the color preceding the color which was removed.

The color information for window 0 is contained at location 380D (refer to Figure 11). If the operator reads the contents of 380D by using the PEEK instruction, the resultant value when printed out is 0 through 255 in decimal integers. Appendix D gives the numbers with their corresponding meanings. This integer is a translation of the binary representation of "on" and "off" bits into decimal integers. The 8 bits each have the meaning shown in Figure 12.

BLINK		RED		GREEN		BLUE	
FOREGROUND	BACKGROUND	FOREGROUND	BACKGROUND	FOREGROUND	BACKGROUND	FOREGROUND	BACKGROUND
1	Ø	1	1	Ø	1	Ø	1

Ø DENOTES "OFF"
1 DENOTES "ON"

IN THIS EXAMPLE: FOREGROUND--BLINKING RED
BACKGROUND--RED+GREEN+BLUE=WHITE

FIGURE 12: DIAGRAM OF COLOR BYTE AND WHAT EACH BIT REPRESENTS

Therefore, an example of 00100110 means no blink, red & blue=magenta foreground, and green background. The binary representation 00100110 translates to 38 in decimal.

In programming the editing function, it is necessary to take the number received from the PEEK operation and AND it with &H55. The results possible are shown in Figure 13 along with the color those numbers represent.

Value of Color Byte AND &H55	Background Color
0	BLACK
1	BLUE
4	GREEN
5	CYAN
16	RED
17	MAGENTA
20	YELLOW
21	WHITE

FIGURE 13: DIAGRAM OF COLOR BYTE AND &H55 RESULTS AND WHAT THEY REPRESENT

4. Determining the Machine State

When editing the create buffer, one must know whether the item being removed is a character in text or a geometric primitive. In the case of character removal the last character is removed. The only additional checking needed is to make sure the last item is a character and not part of a mode change or cursor movement. When removing geometric primitives, the create buffer must be looked through stepping backwards until the primitive is found.

The machine state information for window 0 is found at location 3B52 (refer to Figure 11, page 27). The operator reads the contents of that location using the PEEK operation. The machine states available at this location include blink/non-blink, fill/non-fill, roll/non-roll,

foreground/background, and character/plot. In editing, one must know whether the machine is in character or plot mode. By ANDing the value at 3B52 with &H4, a boolean is developed which is "true" if in plot mode and "false" if in character mode.

As a picture is being edited, it is possible to back over plot/character mode changes. MODE G in the create buffer turns plot "on" and ASCII 21 is a "mode cancel" which turns plot "off."

B. EDITING THE CREATE BUFFER

1. Changes to the Original Driver

In the driver developed by Dillinger (Dil80), an array is set up in Part1 (see Appendix E) to disable all the interrupts caused from pressing the special function keys on the upper three rows of the keyboard. A "bell" is entered for each key, then those keys which are used in the driver are re-enabled or changed. The numbers in the array corresponding to the special function keys are shown in Figure 14.

The CRTOS and CPUOS keys were not being used so they were selected to be the EDIT and TABLET EDIT keys. It was necessary to differentiate between regular editing and editing tablet input because the driver printed the primitive twice when using tablet input. Also it was not possible to use the tablet flag as a determining factor in editing each kind of input because the tablet had to be turned "off" to edit the picture.

Appendix I shows a copy of the "extended" Part2 with those

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

tablet edit 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 *	BLINK *	BACK GROUND *	FILL *	BLACK	BLUE	GREEN	CYAN	RED	MAGENTA	YELLOW	WHITE
131	132	133	134	176	177	178	179	180	181	182	183

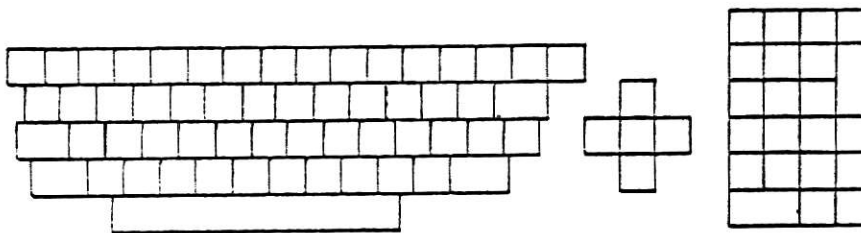


FIGURE 14: DIAGRAM OF SPECIAL FUNCTION KEYS WITH CORRESPONDING CHROMATICS INPUT NUMBER

additions and changes shaded which are necessary to do editing. Code line 580 was added to take care of the problem which arises when the edit keys are pressed without having turned off the tablet. Code lines 590 and 600 were added to re-enable the CRTOS key as the EDIT key and the CPUOS key as the TABLET EDIT key.

In the "handle primitives" section of Part2, lines of code were added to print the primitives. For example, the code IF EF AND P=155 THEN PRINT CHR\$(43); was added to print the rectangle if the edit flag is true and the last primitive found was a rectangle. In the original driver, the primitive handler returned to the point of waiting for the next keyboard interrupt. Line 1110 IF EF THEN RETURN returns control to the place from which it was called in the "last primitive found" section of the editor.

Changes were made to the "sub:coord" section, lines 1380 through 1430. This subroutine collects coordinates and stores them in arrays to be used in printing the primitive. If the edit flag is "true" then the collection is re-routed to the editor.

2. Variables and Flags Used

CNT	used in conjunction with BTM to keep track of which location in the create buffer is being examined
EF	"0" false boolean used as a flag to denote "-1" true that the editor has been entered and then exited
BTM	address of the bottom (end) of the create buffer for window 0 computed from values in &H3B44 and &H3B45

TP address of the top (beginning) of the create buffer for window 0 computed from values in &H3B46 and &H3B47

B1 value representing the background color (refer to Figure 13)

T1 "0" false flag denoting that TABLET
"-1" true EDIT key has been pressed and first time being sent to editor

T2 "0" false flag denoting that TABLET EDIT key
"-1" true has removed first occurrence of primitive and this is the second time through the editor

LO contents of location &H3B52 which contains state information for window 0

M
N ASCII representation found at specified
MN locations in the create buffer
NN

J incremental counter

A, Z addresses used in the subroutine which replaces mode changes with nulls

P temporary variable which holds the value which represents keyboard input for each primitive

A variables used in calculating and holding
X% values which represent the address
Y% to be placed in the bottom pointer

NBTM address of new bottom of create buffer after last character/primitive is removed

C, D variables used to hold values found in &H3B44 and &H3B45

COL contents of &H380D which contains color information

3. Explanation of Code Sections in the Editor

In initializing the "editor," lines 1910 through 1940, values of CNT and EF are set. An end-of-file mark is put at the end of the create

buffer. The top and bottom of the buffer are determined.

Next the "editor" branches to the subroutine which saves the contents of the color byte and determines the background color and returns to the editor. The foreground color is set to the background color. The state of the machine is checked to see if it is in character or plot mode.

The "character editor" section checks the create buffer for the last character entered. It is necessary to check the last four entries each time to see if it might be part of a mode change or cursor movement. The ASCII representations (refer to Appendix A) 1 through 32 either have individual meanings or are not used. Except for 0 and 1 which are null and mode respectively, each entry is specific. Those instructions which can be counteracted by another instruction are counteracted. Those which have no corresponding opposite effect are singled out and a message is sent to the users to let them know they need to press REDRAW so the picture on the screen will correspond to what is the create buffer. In the case of 21 which is "mode change," the "character editor" switches control to the "primitive editor." The last two entries are looked at because most mode changes are made up of two entries, i.e. 1,77=ModeM=background on. The last three entries are taken into consideration because color changes require three entries, i.e. 1,67,49=ModeC2=blue. The last four entries must be inspected because changes in letter size require four entries, i.e. 1,88,51,44=ModeX3,=make the width of the letter three times the usual width.

The "primitive editor" section checks back through the create buffer looking for a primitive. If the combination 1, 77=ModeG=plot on

is encountered control is given to the "character editor" section. Mode and color changes are replaced by nulls (0). When a primitive is encountered, ASCII 33 through 43, see Figure 15, the control is given to the "primitive handler" of the "main driver" which draws the primitive in the background color.

PRIMITIVE	CODE	ASCII CODE
X - BAR	!	33
Y - BAR	"	34
DOT	.	37
VECTOR	'	39
CONCATENATED VECTOR	<	40
CIRCLE	*	42
RECTANGLE	+	43

FIGURE 15: BUILT-IN PRIMITIVES RECOGNIZED
BY THE EDITOR

After the last character or primitive has been overwritten, the pointers are changed so that the last entry is no longer part of the create buffer, the beginning color is reinstated, the edit flag is turned "off," APPEND is instated so the user can continue and the "main driver" resumes waiting for keyboard input.

Several subroutines are used by the editor. One collects the coordinates of the last primitive. One subroutine put nulls in the create buffer to replace mode changes. Another subroutine prints out a message to the user in a color different from the background color to "press REDRAW."

C. SEGMENTING THE CREATE BUFFER

1. Changes in the Original Driver

Part2 of the original driver by Dillinger(Dil80) (see Appendix F) does not consider the size of the create buffer. Refer to Appendix J for a copy of the extended Part2 with changes and additions pertaining to the segmenting process shaded.

Code line 470, SIZE=PEEK(&H3B45), checks to see where the bottom of the create buffer is located. In line 480, IF SIZE>=254 THEN GOTO 2640, the address of the bottom of the create buffer entries is compared to an address near the bottom of the allowable space. If the entries in the create buffer have reached that address, the "main driver" sends control to the "segmentor."

Code line 440, IF SF THEN GOTO 2740 ELSE GOTO 450, was added to route keyboard input to the "segmentor" if the segmentor is in the process of naming the user's file, SF is the segment flag. Code line 760, IF K=215 AND SC=1 THEN PRINT CHR\$(30);"M~C0~N~C2~N~2~Q7~J~H~L~E";
"~W000000511511";CHR\$(28);CHR\$(15);CHR\$(12);"~P~X1,~Y1,~G~L";
CHR\$(27);"W"; ELSE IF K=215 AND SC>1 THEN GOSUB 2970 takes care of initializing the screen to the beginning (default) conditions (see Figure 16) so the picture drawn will correspond to the one drawn by the user if the segment being redrawn is an unsegmented picture. If the picture is segmented, control is routed to the "redraw segmented files" section found in the "segmentor."

In the "handle states" section of the "main driver" changes were made to the results of pressing the CREATE key. If a new picture is being started, CREATE is pushed. At that time it is necessary to

ATTRIBUTE	SCREEN CONDITION	COMMAND
COLORS	BACKGROUND BLACK; FOREGROUND GREEN	Mode M Mode C0 Mode N Mode C2
CURSOR	WHITE; VISIBLE; IN HOME POSITION	Mode Q7 Mode J CHR\$(28)
CHARACTER MODE	HORIZONTAL; CHARACTER SIZE 1x1	Mode H Mode X1, Mode Y1,
A7	OFF	CONTROL 0
BLINK	OFF	Mode 2
ROLL	OFF	Mode P
FILL	OFF	Mode L
BACKGROUND	OFF	Mode N
OVERSTRIKE	OFF	Mode L
COORDINATE INPUT	DECIMAL	Mode E
WINDOW SIZE	FULL SCREEN; CLEARED	Mode W 000 000 511 511 CHR\$(12)

FIGURE 16: DEFAULT CONDITIONS OF THE SCREEN

initialize the segment counter and the first file flag. If the picture is segmented and CREATE is pressed, the picture is being ended and it is necessary to route the processing to the segmentor to store that segment. Code lines 1140 and 1150 handle these operations. If a segmented file is APPENDED after CREATE is turned off it is necessary to remove the copy of the current segment from the disk, otherwise an error would result when an effort was made to put the appended segment on the disk.

2. Variables and Flags Used

SF	"0" false	segment flag denotes the segmentor
	"-1" true	has been entered and exited
SC	segment counter keeps track of how many segments in a picture; initialized to one (1) when a picture is started	
LO	contents of location 3B52 which contains state information for window 0	
K	keyboard input	
SN\$	filename entered by the user	
S\$	file number which is concatenated to the front of filename for second and subsequent files	
C,D	contents of addresses &H3B46 and &H3B47 used to calculate the address of top of buffer	
TP	the address of the top (beginning) of the create buffer calculated from C and D above	
RD	"0" false	redraw flag denotes that
	"-1" true	the segment, other than the first segment, has been stored already

3. Explanation of Code Sections in the Segmentor

Upon entering the segmentor, an "end of file" marker is put at the end of the file in the create buffer and a flag (SF) is set to denote entry. Next a check is made to see if the current segment is the first segment. If it is the first segment, the next action is to set conditions so a message is printed at the top of the screen informing the user that the create buffer is segmenting and she/he has to enter a filename to continue. The "segmentor" branches back to the "main driver" to receive keyboard input to collect the filename. After the message about segmenting is printed, it was necessary to put in a short FOR loop during which nothing is done. It merely takes up time so the user has a chance to realize something is happening. Otherwise, the user may end up with garbage as the filename.

The "collects filename" section of the "segmentor" is branched to from the "main driver" if the segmentor flag is true. The section accumulates letters for a filename until the carriage return is pressed. The section of code also takes into consideration backspaces so that if a typing error occurs, a change can be made. The LEN and LEFT\$ functions were used to take letters off the filename if a backspace occurred.

After collecting the filename, the first file segment is put on the disk and the segment counter is incremented. If the file being saved is not the first segment, it is necessary to concatenate the segment number to the front of the filename. It was necessary to convert the segment number into a character string by using the STR\$ function. Since the STR\$ function puts a blank character in front of the number and since blanks cannot be part of a filename, it was necessary to perform the

MID\$ function on the character string to remove the blank. If a portion of this particular segment has already been saved on the disk, it is necessary to KILL the segment before saving the updated file with the same name. The redraw flag is turned "off" to indicate that the segment is finished. The machine state is then determined, the segment is saved on the disk, and the segment number is incremented.

If the segment being saved is the first segment, it is necessary to reset initial machine conditions (refer to Figure 15) and to redraw the first segment to remove the message printed at the top of the screen.

To start a new segment, it is necessary to determine where the top of the file starts and then put that address in the location which stores the pointer to the bottom of the file. The picture is then appended to and the appended section starts at the top of the create buffer.

A need which was very apparent when dealing with segmented files was the need to change REDRAW. If the file has more than one segment and the user presses REDRAW, it is necessary to store the current segment. A flag is set to indicate that this file was stored, but since it is not finished, it will be necessary to store it again. The flag is necessary because the machine will not store a second BUF file with the same name. As mentioned previously the STR\$ and MID\$ functions are used to name the file. The first and subsequent segments are drawn by going through all the segments one at a time until the current segment has been redrawn.

After any redrawing of segmented files or storing of segmented files, the segment flag is turned "off," the states are reinstated, and control is resumed by the "main driver."

D. SUMMARY

This chapter was written for the person who is interested in duplicating or modifying the editing or segmenting extensions. In the introduction section, several ideas were addressed which must be understood before the extensions can be fully comprehended. The sections dealing directly with each extension first explained what changes had to be made to the "original driver." Then the variables and flags used in the extensions were listed and explained. Lastly, the code of the extensions was explained.

APPENDICES

APPENDIX A

ANSI ASCII CHART

null 0		space 32	48	a 64	P 80	` 96	p 112
mode 1		! 33	1 49	A 65	Q 81	a 97	q 113
		" 34	2 50	B 66	R 82	b 98	r 114
		# 35	3 51	C 67	S 83	c 99	s 115
		\$ 36	4 52	D 68	T 84	d 100	t 116
one dot up 5	mode cancel 21	% 37	5 53	E 69	U 85	e 101	u 117
delete character 6	one dot down 22	& 38	6 54	F 70	V 86	f 102	v 118
bell 7	insert character 23	' 39	7 55	G 71	W 87	g 103	w 119
back space 8	end of record for subbuffer 24	(40	8 56	H 72	X 88	h 104	x 120
tab 9	one dot left 25) 41	9 57	I 73	Y 89	i 105	y 121
line feed 10		* 42	: 58	J 74	Z 90	j 106	z 122
VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
erase page 12	home 28	, 44	< 60	L 76	\ 92	l 108	124
carriage return 13	cursor right 29	- 45	61	M 77] 93	m 109	} 125
A 7 on 14	EOF 30	. 46	= 62	N 78	^ 94	n 110	~ 126
A 7 off 15	one dot right 31	/ 47	? 63	O 79	_ 95	o 111	127

NOTE: THIS CHART SHOWS CODES IN REGULAR CHARACTER SET WITH A7 OFF.
 ENTRIES WITH INDICATE UNUSED ANSI ASCII CODES.

APPENDIX B
"SEGDRAW" PROGRAM

```
100 INPUT "WHAT IS YOUR FILENAME";F$
110 INPUT "HOW MANY SEGMENTS";N
120 A$=F$
130 PRINT CHR$(12);
140 DOS"DRAW "+A$+".BUF
150 PRINT CHR$(27);"W";
160 FOR I=2 TO N
170 B$=MID$(STR$(I),2)
180 DOS"DRAW "+B$+A$+".BUF
190 PRINT CHR$(27);"W";
200 NEXT
```

APPENDIX C

"LISTER" PROGRAM

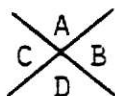
```
100 A=PEEK(&H3B47)
110 B=PEEK(&H3B46)
120 C=(A*256)+B
130 D=PEEK(&H3B45)
140 E=PEEK(&H3B44)
150 F=(D*256)+E
160 FOR X=C TO F
170 L=PEEK(X)
180 IF L=0 THEN PRINT "NULL"
190 IF L=1 THEN PRINT "MODE"
200 IF L=5 THEN PRINT "ONE DOT UP"
210 IF L=6 THEN PRINT "DELETE CHAR"
220 IF L=7 THEN PRINT "BELL"
230 IF L=8 THEN PRINT "BS"
240 IF L=9 THEN PRINT "TAB"
250 IF L=10 THEN PRINT "LF"
260 IF L=11 THEN PRINT "VT"
270 IF L=12 THEN PRINT "ERASE PAGE"
280 IF L=13 THEN PRINT "CR"
290 IF L=14 THEN PRINT "A7 ON"
300 IF L=15 THEN PRINT "A7 OFF"
310 IF L=21 THEN PRINT "MODE CANCEL"
320 IF L=22 THEN PRINT "ONE DOT DOWN"
330 IF L=23 THEN PRINT "INSERT CHAR"
340 IF L=25 THEN PRINT "ONE DOT LEFT"
350 IF L=27 THEN PRINT "ESC"
360 IF L=28 THEN PRINT "HOME"
370 IF L=29 THEN PRINT "CURSOR RIGHT"
380 IF L=30 THEN PRINT "EOF"
390 IF L=31 THEN PRINT "ONE DOT RIGHT"
400 IF L=32 THEN PRINT "SPACE"
410 IF L=33 THEN PRINT "X-BAR"
420 IF L=34 THEN PRINT "Y-BAR"
430 IF L=37 THEN PRINT "DOT"
440 IF L=39 THEN PRINT "VECTOR"
450 IF L=40 THEN PRINT "CONC VECTOR"
460 IF L=42 THEN PRINT "CIRCLE"
470 IF L=43 THEN PRINT "RECTANGLE"
480 IF L>43 THEN PRINT CHR$(L)
490 NEXT
500 RESUME
```

APPENDIX D

COLOR BYTE CHART

BACKGROUND COLORS

	BLACK	BLUE	GREEN	CYAN	RED	MAGENTA	YELLOW	WHITE
BLACK	0	1 193 129 65	4 196 132 68	5 197 133 69	16 208 144 80	17 209 145 81	20 212 148 84	21 213 149 85
BLUE	2 194 130 66	131 67	6 198 134 70	7 199 135 71	18 210 146 82	19 211 147 83	22 214 150 86	23 215 151 87
GREEN	8 200 136 72	9 201 137 73	140 76	13 205 141 77	24 216 152 88	25 217 153 89	28 220 156 92	29 221 157 93
CYAN	10 202 138 74	11 203 139 75	14 206 142 78	143 79	26 218 154 90	27 219 155 91	30 222 158 94	31 223 159 95
RED	32 224 160 96	33 225 161 97	36 228 164 100	37 229 165 101	176 112	49 241 177 113	52 244 180 116	53 245 181 117
MAGENTA	34 226 162 98	35 227 163 99	38 230 166 102	39 231 167 103	50 242 178 114	179 115	54 246 182 118	55 247 183 119
YELLOW	40 232 168 104	41 233 169 105	44 236 172 108	45 237 173 109	56 248 184 120	57 249 185 121	188 124	61 253 189 125
WHITE	42 234 170 106	43 235 171 107	46 238 174 110	47 239 175 111	58 250 186 122	59 251 187 123	62 254 190 126	191 127



- A = NON BLINK
 B = BLINK FOREGROUND AND NON BLINK BACKGROUND
 C = BLINK FOREGROUND AND BLINK BACKGROUND
 D = NON BLINK FOREGROUND AND BLINK BACKGROUND

APPENDIX E

PART1-ORIGINAL DRIVER

```

10 MODEOFF
20 PRINT CHR$(27);"IaF";CHR$(27);"ID9";
30 PRINT CHR$(12);"~P~G";
40 CLEAR 2000
50 DIM P$(127),C(15),X(60),Y(60)
55 AH$="VAR":DO$="XD"
60 FOR I=0 TO 127
70 P$(I)=CHR$(7)
80 NEXT
90 FOR I=0 TO 24
100 READ J,P1$
110 P$(J-128)=P1$
120 NEXT
130 '
140 DATA 176,"~C0"
150 DATA 177,"~C1"
160 DATA 178,"~C2"
170 DATA 179,"~C3"
180 DATA 180,"~C4"
190 DATA 181,"~C5"
200 DATA 182,"~C6"
210 DATA 183,"~C7"
220 DATA 149,"%"
230 DATA 145,"!"
240 DATA 151,"'"
250 DATA 155,"+"
260 DATA 154,"*"
270 DATA 146,"("
280 DATA 245,"~U"
290 DATA 247,"~W"
300 DATA 250,"~Z"
310 DATA 160,"~>a"
320 DATA 161,"~>b"
330 DATA 162,"~>c"
340 DATA 163,"~>d"
350 DATA 164,"~>e"
360 DATA 165,"~>f"
370 DATA 166,"~>g"
380 DATA 167,"~>h"
390 DOS"CHAIN PART2"

```


APPENDIX F

PART2-ORIGINAL DRIVER

```

410 '-----INTERRUPT HANDLER-----
420 ON ERROR#1 GOTO 440
430 GOTO 430
440 IF ERR=24 THEN K=INP(&H4A) ELSE ON ERROR#0 GOTO 0
450 IF K=&H80 THEN PRINT CHR$(30);CHR$(27);"Ia9";:STOP
460 LO=PEEK(&H3B52)
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 &H4) THEN PRINT CHR$(K);
    :GOTO 1140
480 IF NOT FL AND NC<>0 THEN IF 48<=K AND K<=57
    THEN ON CT+1 GOTO 1270,1280,1290
490 IF BF THEN IF 48<=K AND K<=55 THEN BC=(K-48):BF=0:RESUME 1620
500 IF (FL=0 AND NC<>0 AND CT=0) THEN IF CHR$(K)=". "
    THEN C(I)=CURSX(0):C(I+1)=CURSY(0):I=I+2:IF I>=NC OR
    NC=999 THEN RETURN ELSE RESUME
510 IF NC=999 THEN IF K=146 OR K=149 THEN NC=0:PRINT CHR$(7);:RETURN
520 'CATCH FOR CB
530 IF K>215 AND CR THEN PRINT CHR$(27);"Q";
540 IF FL AND K=208 THEN IF PM THEN SP$="O":PM=0:PF=0 ELSE SP$="P":PM=-1:PF=-1
550 IF K>=129 AND K<=134 THEN ON K-128 GOTO 1000,1020,1040,1060,1080,1100
560 IF K>=176 AND K<=183 THEN PRINT P$(K-128);:GOTO 1130
570 IF K=4 THEN DO$="D"
580 IF K=2 THEN AH$="FIX"
590 IF K=3 AND NC<>0 THEN RETURN
600 IF NC<>0 THEN GOTO 1130
610 IF K>=160 AND K<=167 THEN PRINT "-U";:PLOT CURSX(0),CURSY(0):
    PRINT CHR$(14);P$(K-128);CHR$(15);:GOTO 1130
620 IF K=216 THEN PRINT CHR$(7);:GOTO 1120
630 IF (LO AND &H4) AND K=34 THEN PR$=CHR$(K):PRINT CHR$(K);:GOTO 800
640 IF K>127 THEN PR$=P$(K-128):PRINT P$(K-128);CHR$(7);
    ELSE PRINT CHR$(K);:GOTO 1130
650 IF K>=193 AND K<=196 THEN ON K-192 GOTO 1490,1500,1130,1520
660 IF K=199 THEN GOTO 1510
670 IF K=149 OR K=146 THEN PRINT CHR$(7);:GOTO 740
680 IF K>=151 AND K<=155 THEN ON K-150 GOTO 780,1130,1130,870,780
690 IF K=145 THEN PRINT CHR$(7);:GOTO 850
700 IF K=244 THEN PRINT CHR$(7);CHR$(27);"Q";:CR=-1
710 IF K=215 THEN PRINT CHR$(30);CHR$(12);CHR$(27);"W";
720 GOTO 1130
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$(27);"Q";
900 PLOT C(0),C(1):PLOT SQR((C(0)-C(2))^2+((C(1)-C(3))/SQR(2))^2)
910 IF CR THEN PRINT CHR$(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$(27);"C~P~G";:CR=-1
1010 GOTO 1130
1020 IF (LO AND &H4) THEN PRINT CHR$(21); ELSE PRINT "~G";
1030 GOTO 1130
1040 IF (LO AND &H2) THEN PRINT "~P"; ELSE PRINT "~R";
1050 GOTO 1130
1060 IF (LO AND &H40) THEN BL$="XB":PRINT "~2"; ELSE BL$="B":PRINT "~1";
1070 GOTO 1130
1080 IF (LO AND &H1) THEN PRINT "~N"; ELSE PRINT "~M~2";
1090 GOTO 1130
1100 IF (LO AND &H80) THEN FI$="XF":PRINT "~L"; ELSE FI$="F":PRINT "~F";
1110 GOTO 1130
1120 IF FL THEN FL=0 ELSE FL=-1:IF TF=0 THEN GOTO 1160
1130 PRINT CHR$(30);CHR$(7);
1140 RESUME
1150 '----FINISH MAIN----
1160 '----SUB: TABLET INIT----
1170 SP$="O":PM=0
1180 PRINT CHR$(30);CHR$(27);"R1C";CHR$(27);"OE5";CHR$(27);"IE5";
1190 IF CR THEN PRINT CHR$(27);"Q";
1200 TF=-1
1210 RESUME 420
1220 '
1230 '----SUB: COORD----
1240 CT=0:I=0:F$="O":OX=0:OY=0:OF$="O"
1250 IF FL THEN 1310 ELSE 1260
1260 RESUME 430
1270 Z=100*(K-48):CT=CT+1:RESUME
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$;
1330 RESUME 1340
1340 IF NOT PM THEN INPUT #4;XX,YY,F$
1350 INPUT #4;XX,YY,F$
1360 PRINT #4;"S";
1370 X1=INT(XX/4.3):Y1=INT(YY/4.3)
1380 IF OX=X1 AND OY=Y1 AND OF$=F$ THEN 1450
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
1420 IF (NOT PF OR NOT OC) AND F$="1" THEN IF NOT PM THEN PRINT "%";
      :PLOT X1,Y1
1430 IF (NOT PF OR NOT OC) AND F$="1" THEN PRINT CHR$(7);:C(I)=X1:
      C(I+1)=Y1:I=I+2:IF I>=NC THEN PRINT PR$;:RETURN ELSE 1450
1440 PRINT "~U";:PLOT X1,Y1
1450 OX=X1:OY=Y1:OF$=F$:PF=0:OC=-1
1460 PRINT #4;SP$;:GOTO 1340
1470 '
1480 '-----EXTENDED PRIMITIVES-----
1490 CMD$="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
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
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)
1570 IF FL OR CHR$(K)="." THEN
      W=INT(SQR((C(2)-C(4))^2+((C(3)-C(5))/SQR(2))^2))
1580 IF FL OR CHR$(K)="." THEN IF CMD$="REC" THEN
      C(5)=INT(SQR((C(4)-C(6))^2+((C(5)-C(7))/SQR(2))^2))
1590 IF CMD$="REC" THEN HT=W:W=C(5)
1600 IF FL THEN 1610 ELSE RESUME 1610
1610 IF FI$="F" THEN BF=-1:GOTO 430
1620 GOSUB 1660
1630 IF CMD$="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-ORIGINAL DRIVER

```

10 'AUTHOR-MAXINE YEE
20 'PROGRAM-EGP(EXTENDED GRAPHIC PRIMITIVES)
30 'LANGUAGE-CHROMATICS BASIC VERSION 3.0
40 'PLACE-KANSAS STATE UNIVERSITY,DEPT. OF COMPUTER SCIENCE
50 'DATE-DEC.,1979
60 ' THIS IS A PROGRAM FOR THE GENERATION OF EXTENDED GRAPHIC PRIMITIVES
70 SQ=SQR(2)
80 IF BL$="B" THEN PRINT "~1";ELSE PRINT "~2"
90 IF CMD$="VEC" THEN GOSUB 130:GOTO 110
100 IF CMD$="REC" THEN GOSUB 780
110 IF CR THEN PRINT CHR$(27);"Q";
120 DO$="XD":W=1:PRINT "~M~C";CHR$(48+B1);"~N";CHR$(30);:DOS"CHAIN PART2
130 '
140 '      THIS IS A SUBROUTINE FOR DRAWING WIDE VECTORS
150 I=1
160 IF DO$="D" THEN GOSUB 360:RETURN
170 YS=YS/SQ:YN=YN/SQ
180 X=XN-XS
190 Y=YN-YS
200 L=SQR(X^2+Y^2)
210 X(I+2)=XS-(W*Y)\L
220 Y(I+2)=YS+(W*X)\L
230 X(I+1)=XN-(W*Y)\L
240 Y(I+1)=YN+(W*X)\L
250 Y(I+2)=Y(I+2)*SQ
260 Y(I+1)=Y(I+1)*SQ
270 YS=YS*SQ:YN=YN*SQ
280 IF FI$<>"F" THEN GOSUB 1280 ELSE GOSUB 1360
290 'IF DO$="D" THEN GOTO 310 (WHY IS THIS LINE IN HERE? IT'S A COMMENT!)
300 PRINT CHR$(27);"Q";:PLOT XS,YS,XN,YN
310 PRINT CHR$(27);"Q";:PLOT XN,YN,X(I+1),Y(I+1)
320 PLOT X(I+1),Y(I+1),X(I+2),Y(I+2)
330 PLOT X(I+2),Y(I+2),XS,YS:PRINT CHR$(30);
340 IF FI$="F" THEN XM#=(XS+X(I+1))/2:YM#=(YS+Y(I+1))/2:GOSUB 1400
350 RETURN
360 '
370 '      THIS IS A SUBROUTINE FOR DRAWING DOTTED LINES
380 YS=YS/SQ:YN=YN/SQ
390 J=1
400 X(J)=XS
410 Y(J)=YS
420 XB=XS:YB=YS
430 D=12
440 SEG=D
450 SP=SEG\2
460 L1=SQR((XN-XS)^2+(YN-YS)^2)
470 IF L1<SP AND CMD$="VEC" THEN RETURN
480 CO#=(XN-XS)/L1
490 SN#=(YN-YS)/L1
500 K2=(L1\((SP+SEG))*2+1
510 FOR J=1 TO K2 STEP 1
520 X(J+1)=XB+D*(CO#)
530 Y(J+1)=YB+D*(SN#)

```

```

540 '
550 'THIS SECTION OF CODE TESTS THE VARIOUS END POINTS OF A GIVEN LINE
560 E=J MOD 2
570 IF E=0 THEN IF (L1-D) <SEG
      THEN D=L1:NEXT J
      ELSE D=D+SEG:NEXT J
      ELSE GOSUB 1280
580 '
590 IF W<=1 THEN Y(J)=Y(J)*SQ:Y(J+1)=Y(J+1)*SQ:GOSUB 740
600 IF W>1 THEN GOSUB 660
610 IF (L1-D)<=SP AND CMD$="VEC" THEN RETURN 'REMAINING SEG TOO SHORT TO PLOT
620 D=D+SP
630 IF E=1 AND L1-D<=SP THEN RETURN
640 NEXT J
650 RETURN
660 '
670 'THIS IS A SUBROUTINE FOR DRAWING DOTTED WIDE LINES
680 XN=X(J+1)
690 YN=Y(J+1)
700 XS=X(J)
710 YS=Y(J)
720 GOSUB 170
730 RETURN
740 '
750 'THIS IS A PLOTTING SUBROUTINE FOR A SINGLE DOTTED LINE
760 PRINT CHR$(27);"Q";:PLOT X(J),Y(J),X(J+1),Y(J+1):PRINT CHR$(30);
770 RETURN
780 'THIS IS A SUBROUTINE FOR DRAWING RECTANGLES
790 YS=YS/SQ:YN=YN/SQ
800 X=XN-XS
810 Y=YN-YS
820 L=SQR(X^2+Y^2) 'CAL LENGTH OF GIVEN LINE
830 CO#=X/L 'CAL COSINE OF AN ANGLE
840 SN#=Y/L 'CAL SINE OF AN ANGLE
850 '
860 'FOLLOWING SECTION OF CODE CALCULATES THE VARIOUS COORD. OF THE RECTANGLE
870 '
880 I=1
890 X(I)=XS+W*CO#
900 Y(I)=YS+W*SN#
910 X(I+1)=X(I)-W*SN#
920 Y(I+1)=Y(I)+W*CO#
930 X(I+2)=X(I)+(W-HT)*SN#
940 Y(I+2)=Y(I)+(HT-W)*CO#
950 X(I+3)=XS-HT*SN#
960 Y(I+3)=YS+HT*CO#
970 X(I+4)=XS+(L-W)*CO#
980 Y(I+4)=YS+(L-W)*SN#
990 X(I+5)=X(I+4)-W*SN#
1000 Y(I+5)=Y(I+4)+W*CO#
1010 X(I+6)=X(I+4)+(W-HT)*SN#
1020 Y(I+6)=Y(I+4)+(HT-W)*CO#
1030 X(I+7)=XN-HT*SN#
1040 Y(I+7)=YN+HT*CO#

```

```

1050 YS=YS*SQ:YN=YN*SQ
1060 Y(I)=Y(I)*SQ
1070 Y(I+1)=Y(I+1)*SQ
1080 Y(I+2)=Y(I+2)*SQ
1090 Y(I+3)=Y(I+3)*SQ
1100 Y(I+4)=Y(I+4)*SQ
1110 Y(I+5)=Y(I+5)*SQ
1120 Y(I+6)=Y(I+6)*SQ
1130 Y(I+7)=Y(I+7)*SQ
1140 GOSUB 1280:GOSUB 1160
1150 RETURN
1160 '
1170 'THE FOLLOWING SECTION OF CODE PLOTS THE RECTANGLE
1180 PRINT CHR$(27);"Q";:PLOT XS,YS,XN,YN
1190 PLOT XN,YN,X(I+7),Y(I+7)
1200 PLOT X(I+7),Y(I+7),X(I+3),Y(I+3)
1210 PLOT X(I+3),Y(I+3),XS,YS
1220 PLOT X(I+1),Y(I+1),X(I+5),Y(I+5)
1230 PLOT X(I+5),Y(I+5),X(I+6),Y(I+6)
1240 PLOT X(I+6),Y(I+6),X(I+2),Y(I+2)
1250 PLOT X(I+2),Y(I+2),X(I+1),Y(I+1):PRINT CHR$(30);
1260 IF FI$="F" THEN XM#=(XS+X(I+1))/2:YM#=(YS+Y(I+1))/2:GOSUB 1400
1270 RETURN
1280 '
1290 'THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT AND SET COLOR
1300 PRINT CHR$(27);"Q~M";
1310 PRINT CHR$(1);"C";CHR$(48+BC); 'SET BACKGROUND COLOR
1320 PRINT "~N"; 'BACKGROUND LIGHT OFF
1330 PRINT "~G";
1340 PRINT "~";CHR$(30);
1350 RETURN
1360 '
1370 'THIS IS A SUBROUTINE TO HANDLE THE PLOTTING ENVIRONMENT
1380 PRINT CHR$(27);"Q~G'";CHR$(30);
1390 RETURN
1400 '
1410 'THIS SUBROUTINE COMPLEX FILLS AN OBJECT
1420 PRINT CHR$(27);"Q~U";:PLOT XM#,YM# 'MOVE CURSOR TO COORD
1430 PRINT "~J";
1440 PRINT "~M"; 'BACKGROUND LIGHT ON
1450 PRINT CHR$(1);"C";CHR$(48+BC); 'SET BACKGROUND COLOR
1460 PRINT "~N"; 'BACKGROUND LIGHT OFF
1470 PRINT ">";CHR$(32);CHR$(30); 'FILL OBJECT WITH SOLID COLOR
1480 RETURN
1490 '

```

APPENDIX H

PART4-ORIGINAL DRIVER

```

10 I=1
20 SQ=SQR(2)
30 IF BL$="B" THEN PRINT "~1"; ELSE PRINT "~2"
40 IF CMD$="ARR" THEN GOSUB 570:GOTO 55
50 IF CMD$="DAR" THEN GOSUB 1200
55 IF CR THEN PRINT CHR$(27); "Q";
60 DO$="XD":AH$="VAR":W=1:PRINT "~M~C";CHR$(48+B1); "~N";CHR$(30);:DOS"CHAIN PART2
70 IF DO$="D" THEN GOSUB 240:RETURN
80 X=XN-XS
90 Y=YN-YS
100 L=SQR(X^2+Y^2)
110 X(I+2)=XS-(W*Y)\L
120 Y(I+2)=YS+(W*X)\L
130 X(I+1)=XN-(W*Y)\L
140 Y(I+1)=YN+(W*X)\L
150 IF FI$<>"F" THEN GOSUB 2010 ELSE GOSUB 2070
155 IF CR THEN PRINT CHR$(27); "Q";
160 IF DO$="D" THEN GOTO 180
170 PLOT XS,YS,XN,YN
180 PLOT XN,YN,X(I+1),Y(I+1)
190 PLOT X(I+1),Y(I+1),X(I+2),Y(I+2)
200 PLOT X(I+2),Y(I+2),XS,YS
205 PRINT CHR$(30);
210 IF FI$="F" THEN XM#=(XS+X(I+1))/2:YM#=(YS+Y(I+1))/2:GOSUB 2090
220 IF DO$="D" THEN RETURN
230 RETURN
240 J=1
250 X(J)=XS
260 Y(J)=YS
270 XB=XS:YB=YS
280 D=12
290 SEG=D
300 SP=SEG\2
310 L1=SQR((XN-XS)^2+(YN-YS)^2)
320 IF L1<SP AND CMD$="ARR" THEN RETURN
330 IF L1<SP AND CMD$="DAR" THEN RETURN
340 CO#=(XN-XS)/L1
350 SN#=(YN-YS)/L1
360 K=(L1\((SP+SEG))*2+1
370 FOR J=1 TO K STEP 1
380 X(J+1)=XB+D*(CO#)
390 Y(J+1)=YB+D*(SN#)
400 E=J MOD 2
410 IF E=0 THEN IF (L1-D) < SEG THEN D=L1:NEXT J ELSE D=D+SEG:NEXT J ELSE GOSUB 2010
420 IF CR THEN PRINT CHR$(27); "Q";
430 PRINT " ";
440 PLOT X(J),Y(J),X(J+1),Y(J+1)
445 PRINT CHR$(30);
450 IF W>1 THEN GOSUB 510
460 IF (L1-D)<=SP THEN RETURN
470 D=D+SP
480 IF E=1 AND L1-D<=SP THEN RETURN
490 NEXT
500 RETURN

```

```

510 XN=X(J+1)
520 YN=Y(J+1)
530 XS=X(J)
540 YS=Y(J)
550 GOSUB 80
560 RETURN
570 X=XN-XS
580 Y=YN-YS
590 L2=SQR((XN-XS)^2+(YN-YS)^2)
600 W1=W/2
610 SI#=Y/L2
620 CS#=X/L2
630 IF AH$="FIX" AND W1<=1 THEN GOSUB 870: GOTO 740
640 IF AH$="FIX" AND W1>1 THEN GOSUB 870
650 IF AH$="FIX" AND DO$="D" THEN 750
660 P=2
670 X(P)=XS+9/10*X
680 Y(P)=YS+9/10*Y
690 X(P+1)=X(P)-Y/10
700 Y(P+1)=Y(P)+X/10
710 X(P+4)=X(P)+Y/10
720 Y(P+4)=Y(P)-X/10
730 IF DO$="D" AND W1<=1 THEN 960
740 IF W1<=1 THEN GOSUB 2010:GOSUB 1000:RETURN
750 X(P+2)=X(P)-W1*SI#
760 Y(P+2)=Y(P)+W1*CS#
770 X(P+3)=X(P)+W1*SI#
780 Y(P+3)=Y(P)-W1*CS#
790 X(P+5)=XS-W1*SI#
800 Y(P+5)=YS+W1*CS#
810 X(P+6)=XS+W1*SI#
820 Y(P+6)=YS-W1*CS#
830 IF DO$="D" AND AH$="FIX" THEN 980
840 IF DO$="D" AND AH$="VAR" THEN 980
850 GOSUB 2010:GOSUB 1060:GOSUB 1040:RETURN
860 RETURN
870 H=10
880 P=2
890 X(P)=XS+(L2-H)*CS#
900 Y(P)=YS+(L2-H)*SI#
910 X(P+1)=X(P)-H*SI#
920 Y(P+1)=Y(P)+H*CS#
930 X(P+4)=X(P)+H*SI#
940 Y(P+4)=Y(P)-H*CS#
950 IF DO$<>"D" THEN 740
960 IF W1<=1 THEN GOSUB 2010:GOSUB 1010:XN=X(P):YN=Y(P):GOSUB 240:RETURN
970 IF AH$="FIX" THEN 750
980 IF W1>1 THEN GOSUB 2010:GOSUB 1010:GOSUB 1140:GOSUB 280:RETURN
990 RETURN
1000 PRINT CHR$(27); "Q"; :PLOT XS,YS,X(P),Y(P)
1010 PRINT CHR$(27); "Q"; :PLOT X(P+1),Y(P+1),X(P+4),Y(P+4)
1020 PLOT X(P+4),Y(P+4),XN,YN
1030 PLOT XN,YN,X(P+1),Y(P+1):PRINT CHR$(30);
1040 IF FI$="F" THEN XM#=(X(P)+XN)/2:YM#=(Y(P)+YN)/2:GOSUB 2090

```



```

1050 RETURN
1060 PRINT CHR$(27); "Q"; :PLOT X(P+2),Y(P+2),X(P+1),Y(P+1)
1070 PLOT X(P+1),Y(P+1),XN,YN
1080 PLOT XN,YN,X(P+4),Y(P+4)
1090 PLOT X(P+4),Y(P+4),X(P+3),Y(P+3)
1100 PLOT X(P+3),Y(P+3),X(P+6),Y(P+6)
1110 PLOT X(P+6),Y(P+6),X(P+5),Y(P+5)
1120 PLOT X(P+5),Y(P+5),X(P+2),Y(P+2):PRINT CHR$(30);
1130 RETURN
1140 XN=X(P+3):YN=Y(P+3)
1150 XS=X(P+6):YS=Y(P+6)
1160 J=1
1170 XB=X(P+6):YB=Y(P+6)
1180 X(J)=X(P+6):Y(J)=Y(P+6)
1190 RETURN
1200 X=XN-XS
1210 Y=YN-YS
1220 L3=SQR((XN-XS)^2+(YN-YS)^2)
1230 W1=W/2
1240 SI#=Y/L3
1250 CS#=X/L3
1260 IF AH$="FIX" AND W1<=1 THEN GOSUB 1540:GOSUB 2010:GOSUB 1700:RETURN
1270 IF AH$="FIX" AND W1>1 THEN GOSUB 1540:GOTO 1440
1280 P=2
1290 X(P)=XS+9/10*X
1300 Y(P)=YS+9/10*Y
1310 X(P+1)=X(P)-Y/10
1320 Y(P+1)=Y(P)+X/10
1330 X(P+4)=X(P)+Y/10
1340 Y(P+4)=Y(P)-X/10
1350 X(P+5)=XS+X/10
1360 Y(P+5)=YS+Y/10
1370 X(P+6)=X(P+5)-Y/10
1380 Y(P+6)=Y(P+5)+X/10
1390 X(P+9)=X(P+5)+Y/10
1400 Y(P+9)=Y(P+5)-X/10
1410 IF DO$="D" AND W1<=1 THEN GOSUB 2010:GOSUB 1710:GOTO 1920
1420 IF DO$="D" AND W1>=1 THEN 1440
1430 IF W1<=1 THEN GOSUB 2010:GOSUB 1700:RETURN
1440 X(P+2)=X(P)-W1*SI#
1450 Y(P+2)=Y(P)+W1*CS#
1460 X(P+3)=X(P)+W1*SI#
1470 Y(P+3)=Y(P)-W1*CS#
1480 X(P+7)=X(P+5)-W1*SI#
1490 Y(P+7)=Y(P+5)+W1*CS#
1500 X(P+8)=X(P+5)+W1*SI#
1510 Y(P+8)=Y(P+5)-W1*CS#
1520 IF DO$="D" THEN GOSUB 2010:GOSUB 1710:GOSUB 1920:GOSUB 280:RETURN
1530 GOSUB 2010:GOSUB 1800:RETURN
1540 H=12
1550 P=2
1560 X(P)=XS+(L3-H)*CS#
1570 Y(P)=YS+(L3-H)*SI#
1580 X(P+1)=X(P)-H*SI#

```

```

1590 Y(P+1)=Y(P)+H*CS#
1600 X(P+4)=X(P)+H*SI#
1610 Y(P+4)=Y(P)-H*CS#
1620 X(P+5)=XS+H*CS#
1630 Y(P+5)=YS+H*SI#
1640 X(P+6)=X(P+5)-H*SI#
1650 Y(P+6)=Y(P+5)+H*CS#
1660 X(P+9)=X(P+5)+H*SI#
1670 Y(P+9)=Y(P+5)-H*CS#
1680 IF DO$="D" THEN GOSUB 2010:GOSUB 1710:GOSUB 1440
1690 RETURN
1700 PRINT CHR$(27); "Q"; :PLOT X(P+5),Y(P+5),X(P),Y(P)
1710 PRINT CHR$(27); "Q"; :PLOT X(P+1),Y(P+1),XN,YN
1720 PLOT XN,YN,X(P+4),Y(P+4)
1730 PLOT X(P+4),Y(P+4),X(P+1),Y(P+1)
1740 PLOT X(P+6),Y(P+6),X(P+9),Y(P+9)
1750 PLOT X(P+9),Y(P+9),XS,YS
1760 PLOT XS,YS,X(P+6),Y(P+6):PRINT CHR$(30);
1770 IF FI$="F" THEN XM#=(X(P)+XN)/2:YM#=(Y(P)+YN)/2:GOSUB 2090
1780 IF FI$="F" THEN XM#=(X(P+5)+XS)/2:YM#=(Y(P+5)+YS)/2:GOSUB 2090
1790 RETURN
1800 PRINT CHR$(27); "Q"; :PLOT X(P+2),Y(P+2),X(P+1),Y(P+1)
1810 PLOT X(P+1),Y(P+1),XN,YN
1820 PLOT XN,YN,X(P+4),Y(P+4)
1830 PLOT X(P+4),Y(P+4),X(P+3),Y(P+3)
1840 PLOT X(P+3),Y(P+3),X(P+8),Y(P+8)
1850 PLOT X(P+8),Y(P+8),X(P+9),Y(P+9)
1860 PLOT X(P+9),Y(P+9),XS,YS
1870 PLOT XS,YS,X(P+6),Y(P+6)
1880 PLOT X(P+6),Y(P+6),X(P+7),Y(P+7)
1890 PLOT X(P+7),Y(P+7),X(P+2),Y(P+2):PRINT CHR$(30);
1900 IF FI$="F" THEN XM#=(X(P)+XN)/2:YM#=(Y(P)+YN)/2:GOSUB 2090
1910 RETURN
1920 IF W1<=1 THEN 1980
1930 XS=X(P+8):YS=Y(P+8)
1940 XN=X(P+3):YN=Y(P+3)
1950 XB=X(P+8):YB=Y(P+8)
1960 J=1:X(J)=X(P+8):Y(J)=Y(P+8)
1970 RETURN
1980 XS=X(P+5):YS=Y(P+5)
1990 XN=X(P):YN=Y(P)
2000 GOSUB 310:RETURN
2010 PRINT CHR$(27); "Q~M";
2020 PRINT CHR$(1); "C";CHR$(48+BC);
2030 PRINT "~N";
2040 PRINT "~G";
2050 PRINT "'";CHR$(30);
2060 RETURN
2070 PRINT CHR$(27); "Q~G'";CHR$(30);
2080 RETURN
2090 PRINT CHR$(27); "Q~U"; :PLOT XM#,YM#
2100 PRINT "~J";
2110 PRINT "~M";
2120 PRINT CHR$(1); "C";CHR$(48+BC);
2130 PRINT "~N";
2140 PRINT ">";CHR$(32);CHR$(30);

```

APPENDIX I

EXTENDED PART2-WITH EDITOR SHADED

```

400 '-----INTERRUPT HANDLER-----
410 ON ERROR#1 GOTO 430
420 GOTO 420
430 IF ERR=24 THEN K=INP(&H4A) ELSE ON ERROR #0 GOTO 0
432 '
435 '-----CHECK--IF IN SEGMENTOR, THEN ROUTE INPUT TO SEGMENTOR---
440 IF SF THEN GOTO 2740 ELSE GOTO 450
450 IF K=&H80 THEN PRINT CHR$(30); CHR$(27); "IA9";:STOP
455 '
460 '-----CHECK SIZE OF CREATE BUFFER-----
470 SIZE=PEEK(&H3B45)
480 IF SIZE>254 THEN GOTO 2640 'SIZE WAS 128 FOR TESTING'
490 LO=PEEK(&H3B52)
500 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 &H4) THEN PRINT CHR$(K);:GOTO 1290
510 IF NOT FL AND NC<>0 THEN IF 48<=K AND K<=57 THEN ON CT+1
    GOTO 1410,1420,1430
520 IF BF THEN IF 48<=K AND K<=55 THEN BC=(K-48):BF=0:RESUME 1740
530 IF (FL=0 AND NC<>0 AND CT=0) THEN IF CHR$(K)="." THEN
    C(I)=CURSX(0):C(I+1)=CURSY(0):I=I+2:IF I>=NC OR NC=999 THEN
    RETURN ELSE RESUME
540 IF NC=999 THEN IF K=146 OR K=149 THEN NC=0:PRINT CHR$(7);:RESUME
550 'CATCH FOR CB
560 IF K<>215 AND CR THEN PRINT CHR$(27);"Q";
570 IF FL AND K=208 THEN IF PM THEN SP$="O":PM=0:PF=0 ELSE SP$="P":PM=-1:PF=-1
580 IF K>=129 AND K<=134 THEN ON K-128 GOTO 1140,1170,1190,1210,1230,1250
585 IF (K=212 OR K=218) AND FL THEN PRINT CHR$(21);CHR$(30);CHR$(15);
    'TURN OFF TABLET BEFORE TRYING TO EDIT "; "G";CHR$(27);"Q";:RESUME
590 IF K=212 THEN GOTO 1910
600 IF K=218 THEN T1=-1:GOSUB 1910:T2=-1:GOTO 2270
610 IF K>=176 AND K<=183 THEN PRINT P$(K-128);:GOTO 1280
620 IF K=4 THEN DO$="D"
630 IF K=2 THEN AH$="FIX"
640 IF K=3 AND NC<>0 THEN RETURN
650 IF NC<>0 THEN GOTO 1280
660 IF K>=160 AND K<=167 THEN PRINT "~U";:PLOT CURSX(0),CURSY(0):
    PRINT CHR$(14);P$(K-128);CHR$(15);:GOTO 1280
670 IF K=216 THEN PRINT CHR$(7);:GOTO 1270
680 IF (LO AND &H4) AND K=34 THEN PR$=CHR$(K):PRINT CHR$(K);:GOTO 890
690 IF K>127 THEN PR$=P$(K-128):PRINT P$(K-128);CHR$(7); ELSE
    PRINT CHR$(K);:GOTO 1280
700 IF K>=193 AND K<=196 THEN ON K-192 GOTO 1610,1620,1280,1640
710 IF K=199 THEN GOTO 1630
720 IF K=146 THEN PRINT CHR$(K-116);CHR$(7);:GOTO 790
725 IF K=149 THEN PRINT CHR$(7);:GOTO 790
730 IF K>=151 AND K<=155 THEN ON K-150 GOTO 850,1280,1280,980,850
740 IF K=145 THEN PRINT CHR$(7);:GOTO 950
750 IF K=244 THEN PRINT CHR$(7);CHR$(27);"Q";:CR=-1
760 IF K=215 AND SC=1 THEN PRINT CHR$(30);"M~C0~N~C2~N~2~Q7~J~H~L~E";
    "W000000511511";CHR$(28);CHR$(15);CHR$(12);"~P~X1,~Y1,~G~L";
    CHR$(27);"W"; ELSE IF K=215 AND SC>1 THEN GOSUB 2970

```

```

770 GOTO 1280
780 '----HANDLE PRIMITIVES
790 NC=999:GOSUB 1370:IF EF AND P=149 THEN PRINT CHR$(37);
800 IF EF AND P=146 THEN PRINT CHR$(34);
810 IF K=3 THEN NC=0:GOTO 1120
820 IF CR THEN PRINT CHR$(27);"Q";
830 IF NOT FL AND NC<>0 THEN PLOT C(I-2),C(I-1):PRINT CHR$(30);
840 IF NC=999 THEN GOTO 790 ELSE IF FL THEN GOTO 410 ELSE RESUME 410
850 NC=4:GOSUB 1370:NC=0:IF EF AND P=155 THEN PRINT CHR$(43);
860 IF EF AND P=151 THEN PRINT CHR$(39);
870 IF K=3 THEN GOTO 1120
880 GOTO 1060
890 NC=4:GOSUB 1370:NC=0:IF EF THEN PRINT CHR$(34);
900 IF K=3 THEN GOTO 1120
910 IF CR THEN PRINT CHR$(27);"Q";
920 PLOT C(0),C(1),C(3)
930 IF CR THEN PRINT CHR$(30);
940 GOTO 1120
950 NC=3:GOSUB 1370:NC=0
955 IF EF AND P=145 THEN PRINT CHR$(33);
957 IF EF AND P=146 THEN PRINT CHR$(34);
960 IF K=3 THEN GOTO 1120
970 GOTO 1060
980 NC=3:GOSUB 1370:NC=0:IF EF THEN PRINT CHR$(42);
990 IF K=3 THEN GOTO 1120
1000 IF EF THEN 1060
1010 IF K>=48 AND K<=57 THEN 1060 ELSE 1020
1020 IF CR THEN PRINT CHR$(27);"Q";
1030 PLOT C(0),C(1):PLOT SQR((C(0)-C(2))^2+((C(1)-C(3))/SQR(2))^2)
1040 IF CR THEN PRINT CHR$(30);
1050 GOTO 1120
1060 FOR M=0 TO (I-1) STEP 2
1070 IF CR THEN PRINT CHR$(27);"Q";
1080 PLOT C(M),C(M+1)
1090 IF CR THEN PRINT CHR$(30);
1100 NEXT
1110 IF EF THEN RETURN
1120 IF FL THEN 410 ELSE RESUME 410
1130 '----HANDLE STATES
1140 IF CR THEN CR=0:PRINT CHR$(30); ELSE PRINT CHR$(27);"C~P~G";
      :CR=-1:IF SC<=1 THEN SC=1
1150 IF SC>1 THEN GOSUB 2840
1160 GOTO 1280
1170 IF (LO AND &H4) THEN PRINT CHR$(21); ELSE PRINT "~G";
1180 GOTO 1280
1190 IF (LO AND &H2) THEN PRINT "~P"; ELSE PRINT "~R";
1200 GOTO 1280
1210 IF (LO AND &H40) THEN BL$="XB":PRINT "~2"; ELSE BL$="B":PRINT "~1";
1220 GOTO 1280
1230 IF (LO AND &H1) THEN PRINT "~N"; ELSE PRINT "~M~2";
1240 GOTO 1280
1250 IF (LO AND &H80) THEN FI$="XF":PRINT "~L"; ELSE FI$="F":PRINT "~F";
1260 GOTO 1280
1270 IF FL THEN FL=0 ELSE FL=-1:IF TF=0 THEN GOTO 1310
1280 PRINT CHR$(30);CHR$(7);
1290 RESUME

```

```

1300 '----FINISH MAIN----
1310 '----SUB: TABLET INIT
1320 SP$="0":PM=0
1330 PRINT CHR$(30);CHR$(27);"R1C";CHR$(27);"OE5";CHR$(27);"IE5";
1340 IF CR THEN PRINT CHR$(27);"Q";
1350 TF=-1
1360 RESUME 410
1370 '----SUB: COORD
1380 CT=0:I=0:F$="0":OX=0:OY=0:OF$="0"
1390 IF FL THEN 1440 ELSE IF EF THEN GOTO 2540 ELSE 1400
1400 RESUME 420
1410 Z=100*(K-48):CT=CT+1:IF EF THEN GOTO 2580 ELSE RESUME
1420 Z=Z+10*(K-48):CT=CT+1:IF EF THEN GOTO 2580 ELSE RESUME
1430 C(I)=Z+(K-48):CT=0:I=I+1:IF EF AND I<NC THEN GOTO 2580 ELSE
    IF I>=NC OR (NC=999 AND I=2) THEN RETURN ELSE RESUME
1440 '----TABLET COORD
1450 PRINT #4;SP$;
1460 RESUME 1470
1470 IF NOT PM THEN INPUT #4;XX,YY,F$
1480 INPUT #4;XX,YY,F$
1490 PRINT #4;"S";
1500 X1=INT(XX/4.3):Y1=INT(YY/4.3)
1510 IF OX=X1 AND OY=Y1 AND OF$=F$ THEN 1580
1520 IF NC=999 AND F$="1" AND CR THEN PRINT CHR$(27);"Q";
1530 IF (NOT PF OR NOT OC) AND NC=999 THEN IF F$="1" THEN PLOT X1,Y1:
    PRINT CHR$(30);CHR$(7);:GOTO 1580
1540 IF F$="4" THEN NC=0:K=3:RETURN
1550 IF (NOT PF OR NOT OC) AND F$="1" THEN IF NOT PM THEN PRINT CHR$(30);
    "%";CHR$(27);"Q";:PLOT X1,Y
1560 IF (NOT PF OR NOT OC) AND F$="1" THEN PRINT CHR$(7);:C(I)=X1:
    C(I+1)=Y1:I=I+2:IF I>=NC THEN PRINT PR$;:RETURN ELSE 1580
1570 PRINT "~U";:PLOT X1,Y1
1580 OX=X1:OY=Y1:OF$=F$:PF=0:OC=-1
1590 PRINT #4;SP$;:GOTO 1470
1600 '----EXTENDED PRIMITIVES
1610 CMD$="ARR":NC=5:GOTO 1650
1620 CMD$="REC":NC=6:GOTO 1650
1630 CMD$="VEC":NC=5:GOTO 1650
1640 CMD$="DAR":NC=5:GOTO 1650
1650 IF FL THEN IF CMD$="REC" THEN NC=8 ELSE NC=6
1660 GOSUB 1370:NC=0:XS=C(0):YS=C(1):XN=C(2):YN=C(3):W=C(4)
1670 IF K=3 THEN RESUME 410
1680 IF NOT FL AND CHR$(K)="." AND CMD$="REC" THEN NC=2:GOSUB 1370:
    NC=0:C(6)=C(0):C(7)=C(1)
1690 IF FL OR CHR$(K)="." THEN
    W=INT(SQR((C(2)-C(4))^2+((C(3)-C(5))/SQR(2))^2))
1700 IF FL OR CHR$(K)="." THEN IF CMD$="REC" THEN
    C(5)=INT(SQR((C(4)-C(6))^2+((C(5)-C(7))/SQR(2))^2))
1710 IF CMD$="REC" THEN HT=W:W=C(5)
1720 IF FL THEN 1730 ELSE RESUME 1730
1730 IF FI$="F" THEN BF=-1:GOTO 420
1740 GOSUB 1770
1750 IF CMD$="VEC" OR CMD$="REC" THEN DOS"CHAIN PART3
1760 IF CMD$="ARR" OR CMD$="DAR" THEN DOS"CHAIN PART4

```

```

1770 '----SUB: SAVE BACKGROUND COLOR
1780 COL=PEEK(&H380D)
1790 BG=COL AND &H55
1800 IF BG=0 THEN B1=0
1810 IF BG=1 THEN B1=1
1820 IF BG=4 THEN B1=2
1830 IF BG=5 THEN B1=3
1840 IF BG=16 THEN B1=4
1850 IF BG=17 THEN B1=5
1860 IF BG=20 THEN B1=6
1870 IF BG=21 THEN B1=7
1880 RETURN
1885 '
1890 '----EDITOR-----Mitchell
1900 '
1910 CNT=1:PRINT CHR$(30);:EF=-1
1920 BTM=(PEEK(&H3B45)*256)+PEEK(&H3B44)
1930 TP=(PEEK(&H3B47)*256)+PEEK(&H3B46)
1940 IF BTM-1=TP THEN GOTO 2510
1945 '
1950 '----CHANGE FG COLOR TO BG COLOR
1960 GOSUB 1780:PRINT"C";MID$(STR$(B1),2);
1970 '---PLOT OR CHARACTER MODE
1980 IF (LO AND &H4) THEN GOTO 2270
1985 '
1990 '----CHARACTER EDITOR
2000 M=PEEK(BTM-CNT)
2010 IF BTM-CNT=TP+3 THEN GOTO 2510
2020 N=PEEK(BTM-(CNT+1))
2030 IF M=21 THEN CNT=CNT+1:PRINT "G";:GOTO 2270
2040 IF N=1 THEN CNT=CNT+2:GOTO 2000
2050 NN=PEEK(BTM-(CNT+2))
2060 IF N=67 AND NN=1 THEN A=BTM-(CNT+2):Z=BTM-CNT:CNT=CNT+3
      GOSUB 2600:GOTO 2000 ELSE GOTO 2100
2070 MN=PEEK(BTM-(CNT+3)):MM=PEEK(BTM-(CNT+4))
2080 IF MN=1 THEN IF NN=88 OR NN=89 THEN IF N>=48 AND N<=57 THEN
      IF M=44 THEN A=BTM-(CNT+3):Z=BTM-CNT:CNT=CNT+4:GOSUB 2600
      PRINT"X1,";Y1,"";:GOTO 2000 ELSE GOTO 2100
2090 IF MM=1 THEN IF MN=88 OR MN=89 THEN IF NN>=48 AND NN<=57
      THEN IF N>=48 AND N<=57 THEN IF M=44 THEN A=BTM-(CNT+4):
      Z=BTM-CNT:CNT=CNT+5:GOSUB 2600:PRINT"X1,";Y1,"";:GOTO 2000
      ELSE GOTO 2100
2100 IF M=0 OR M=7 THEN GOTO 2240
2110 IF M=5 THEN PRINT CHR$(22);:GOTO 2240
2120 IF M=8 THEN PRINT CHR$(32);:GOTO 2240
2130 IF M=6 OR M=9 OR M=12 OR M=13 OR M=23 OR M=27 OR M=28
      THEN PRINT"C";MID$(STR$(B1+1),2);CHR$(15);
      "USER NEEDS TO PRESS REDRAW "; "C";MID$(STR$(B1),2);:GOTO 2240
2140 IF M=10 THEN PRINT CHR$(11);:GOTO 2240
2150 IF M=11 THEN PRINT CHR$(10);:GOTO 2240
2160 IF M=14 THEN PRINT CHR$(15);:GOTO 2240
2170 IF M=15 THEN PRINT CHR$(14);:GOTO 2240
2180 IF M=22 THEN PRINT CHR$(5);:GOTO 2240

```



```

2190 IF M=25 THEN PRINT CHR$(31);:GOTO 2240
2200 IF M=29 OR M=32 THEN PRINT CHR$(8);:GOTO 2240
2210 IF M=31 THEN PRINT CHR$(25);:GOTO 2240
2220 IF M>=33 THEN PRINT CHR$(8);CHR$(M);CHR$(8);:GOTO 2470
      ELSE CNT=CNT+1:GOTO 2000
2230 GOTO 2470
2240 CNT=CNT+1:GOTO 2000
2250 '
2260 '---PRIMITIVE EDITOR
2270 J=BTM-CNT
2280 IF J=TP+3 THEN GOTO 2510
2290 M=PEEK(J)
2300 IF M=7 THEN POKE J,0
2310 IF M>32 AND M<44 THEN GOTO 2360
2320 IF M=1 THEN N=PEEK(J+1) ELSE 2350
2330 IF N=71 THEN CNT=CNT+1:GOSUB 2610:PRINT"C";MID$(STR$(B1),2);:GOTO 2000
2340 IF N=67 THEN A=BTM-CNT:Z=BTM-(CNT-2):GOSUB 2600 ELSE A=BTM-CNT
      Z=BTM-(CNT-1):GOSUB 2600
2350 CNT=CNT+1:GOTO 2270
2355 '
2360 '---LAST PRIMITIVE FOUND
2370 IF T2 THEN GOTO 2440
2380 IF M=33 THEN P=145:GOSUB 950:GOTO 2440
2390 IF M=34 THEN CNT=CNT+2:GOSUB 950:GOTO 2440
2400 IF M=37 THEN P=149:GOSUB 2610:GOTO 2440
2405 IF M=40 THEN GOSUB 2610:GOTO 2440
2410 IF M=42 THEN P=154:GOSUB 980:GOTO 2440
2420 IF M=43 THEN P=155:GOSUB 850:GOTO 2440
2430 IF M=39 THEN P=151:GOSUB 850:GOTO 2440
2435 '
2440 '---CHANGE POINTERS & REPLACE COLOR
2450 IF T1 THEN T1=0:CNT=CNT+1:RETURN
2460 IF T2 THEN T2=0
2470 NBTM=BTM-CNT
2480 A=NBTM/256:X%=A
2490 Y%=NBTM-(256*X%)
2500 POKE 15172,Y%:POKE 15173,X%:POKE 14349,COL
2510 PRINT CHR$(27);"Q";:EF=0
2520 RESUME 420
2525 '
2530 '---SUB: COLLECT THE COORD OF THE PRIMITIVE
2540 FOR J=BTM-CNT TO BTM-1
2550 M=PEEK(J)
2560 IF M=0 THEN GOTO 2580
2570 IF M>=48 AND M<=57 THEN K=M:ON CT+1 GOTO 1410,1420,1430
2580 NEXT
2585 '
2590 '---SUB: PUT NULLS IN CB TO REPLACE MODE CHANGES
2600 FOR J=A TO Z: POKE J,0: NEXT: RETURN
2602 '
2605 '---SUB: CHANGE COLOR AND TELL USER TO PRESS REDRAW---
2610 PRINT CHR$(1);CHR$(21);"C";MID$(STR$(B1+1),2);CHR$(15);
      " USER NEEDS TO PRESS REDRAW ";:RETURN

```

```

2620 '----SEGMENTOR---Mitchell
2630 '
2640 PRINT CHR$(30);:SF=-1
2650 IF SC<>1 THEN GOTO 2840
2660 PRINT "~2";CHR$(15);"~P~X1,~Y1,~L"; 'INITIALIZE SCREEN CONDITIONS SO
2670 PRINT CHR$(28);"~M~CO~N~C7"; 'MESSAGE WILL APPEAR AT TOP LEFT
2680 LO=PEEK(&H3B52)
2690 IF (LO AND &H4) THEN PRINT CHR$(21);
2700 PRINT "CREATE BUFFER SEGMENTING If you wish to continue,
      name your file ";
2710 FOR I=1 TO 500 :NEXT 'DELAY
2720 RESUME 420
2725 '
2730 '----COLLECTS THE FILENAME
2740 PRINT CHR$(K);
2750 IF K=13 THEN 2810
2760 IF K=8 THEN L=LEN(SN$):SN$=LEFT$(SN$,L-1):RESUME 420
2770 SN$=SN$+CHR$(K)
2780 RESUME 420
2790 '
2800 '----PUT FIRST FILE ON DISK
2810 DOS"BUFF "+SN$+".BUF"
2820 SC=SC+1:GOTO 2900
2825 '
2830 '----PUT SUBSEQUENT FILES ON DISK
2840 S$=MID$(STR$(SC),2)
2850 LO=PEEK(&H3B52)
2855 IF RD THEN DOS"KILL "+S$+SN$+".BUF":RD=0 'REMOVES PREVIOUS FILE WITH
2860 DOS"BUFF "+S$+SN$+".BUF" 'SAME NAME IF STORED WITHIN
2870 SC=SC+1 'THIS PICTURE PRODUCTION
2880 GOTO 2910
2885 '
2890 '----SET INITIAL CONDITIONS FOR FIRST FILE
2900 PRINT CHR$(12);"~M~CO~N~C2~2";CHR$(15);"~P~X1,~Y1,~G~L";
      CHR$(27);"W";
2905 '
2910 '----DETERMINE TOP OF CREATE BUFFER AND START A NEW FILE
2920 C=PEEK(&H3B46):D=PEEK(&H3B47)
2930 POKE 15172,C:POKE 15173,D
2940 PRINT CHR$(27);"Q";
2950 GOTO 3070
2955 '
2960 '----REDRAW OF SEGMENTED FILES
2970 PRINT CHR$(30);CHR$(12);
2975 IF RD THEN DOS"KILL "+S$+SN$+".BUF"
2980 S$=MID$(STR$(SC),2):DOS"BUFF "+S$+SN$+".BUF"
2985 IF NOT RD THEN RD=-1
2990 DOS"DRAW "+SN$+".BUF"
3000 PRINT CHR$(27);"W";
3010 FOR I=2 TO SC
3020 S$=MID$(STR$(I),2)
3030 DOS"DRAW "+S$+SN$+".BUF"
3040 PRINT CHR$(27);"W";
3050 NEXT
3055 '
3060 '----TURN SEGMENTOR FLAG OFF, RETURN STATES, GO TO NEW INPUT
3070 SF=0:POKE 9651,LO:RESUME 420

```


APPENDIX J

64

EXTENDED PART2-WITH SEGMENTOR SHADED

```

400 '-----INTERRUPT HANDLER-----
410 ON ERROR#1 GOTO 430
420 GOTO 420
430 IF ERR=24 THEN K=INP(&H4A) ELSE ON ERROR #0 GOTO 0
432 '
435 '-----CHECK--IF IN SEGMENTOR, THEN ROUTE INPUT TO SEGMENTOR-----
440 IF SF THEN GOTO 2740 ELSE GOTO 450
450 IF K=&H80 THEN PRINT CHR$(30); CHR$(27); "IA9";:STOP
455 '
460 '-----CHECK SIZE OF CREATE BUFFER-----
470 SIZE=PEEK(&H3B45)
480 IF SIZE>254 THEN GOTO 2640 'SIZE WAS 128 FOR TESTING
490 LO=PEEK(&H3B52)
500 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 &H4) THEN PRINT CHR$(K);:GOTO 1290
510 IF NOT FL AND NC<>0 THEN IF 48<=K AND K<=57 THEN ON CT+1
    GOTO 1410,1420,1430
520 IF BF THEN IF 48<=K AND K<=55 THEN BC=(K-48):BF=0:RESUME 1740
530 IF (FL=0 AND NC<>0 AND CT=0) THEN IF CHR$(K)="." THEN
    C(I)=CURSX(0):C(I+1)=CURSY(0):I=I+2:IF I>=NC OR NC=999 THEN
    RETURN ELSE RESUME
540 IF NC=999 THEN IF K=146 OR K=149 THEN NC=0:PRINT CHR$(7);:RESUME
550 'CATCH FOR CB
560 IF K<>215 AND CR THEN PRINT CHR$(27);"Q";
570 IF FL AND K=208 THEN IF PM THEN SP$="O":PM=0:PF=0 ELSE SP$="P":PM=-1:PF=-1
580 IF K>=129 AND K<=134 THEN ON K-128 GOTO 1140,1170,1190,1210,1230,1250
585 IF (K=212 OR K=218) AND FL THEN PRINT CHR$(21);CHR$(30);CHR$(15);
    " TURN OFF TABLET BEFORE TRYING TO EDIT "; "G";CHR$(27);"Q";:RESUME
590 IF K=212 THEN GOTO 1910
600 IF K=218 THEN T1=-1:GOSUB 1910:T2=-1:GOTO 2270
610 IF K>=176 AND K<=183 THEN PRINT P$(K-128);:GOTO 1280
620 IF K=4 THEN DO$="D"
630 IF K=2 THEN AH$="FIX"
640 IF K=3 AND NC<>0 THEN RETURN
650 IF NC<>0 THEN GOTO 1280
660 IF K>=160 AND K<=167 THEN PRINT "~U";:PLOT CURSX(0),CURSY(0):
    PRINT CHR$(14);P$(K-128);CHR$(15);:GOTO 1280
670 IF K=216 THEN PRINT CHR$(7);:GOTO 1270
680 IF (LO AND &H4) AND K=34 THEN PR$=CHR$(K):PRINT CHR$(K);:GOTO 890
690 IF K>127 THEN PR$=P$(K-128):PRINT P$(K-128);CHR$(7); ELSE
    PRINT CHR$(K);:GOTO 1280
700 IF K>=193 AND K<=196 THEN ON K-192 GOTO 1610,1620,1280,1640
710 IF K=199 THEN GOTO 1630
720 IF K=146 THEN PRINT CHR$(K-116);CHR$(7);:GOTO 790
725 IF K=149 THEN PRINT CHR$(7);:GOTO 790
730 IF K>=151 AND K<=155 THEN ON K-150 GOTO 850,1280,1280,980,850
740 IF K=145 THEN PRINT CHR$(7);:GOTO 950
750 IF K=244 THEN PRINT CHR$(7);CHR$(27);"Q";:CR=-1
760 IF K=215 AND SC=1 THEN PRINT CHR$(30); "M-CO-N-C2-N-2-Q7-J-H-L-E"
    "W000000511511";CHR$(28);CHR$(15);CHR$(12); "P-X1,-Y1,-G-L"
    CHR$(27);"W"; ELSE IF K=215 AND SC>1 THEN GOSUB 2970

```

```

770 GOTO 1280
780 '----HANDLE PRIMITIVES
790 NC=999:GOSUB 1370:IF EF AND P=149 THEN PRINT CHR$(37);
800 IF EF AND P=146 THEN PRINT CHR$(34);
810 IF K=3 THEN NC=0:GOTO 1120
820 IF CR THEN PRINT CHR$(27);"Q";
830 IF NOT FL AND NC<>0 THEN PLOT C(I-2),C(I-1):PRINT CHR$(30);
840 IF NC=999 THEN GOTO 790 ELSE IF FL THEN GOTO 410 ELSE RESUME 410
850 NC=4:GOSUB 1370:NC=0:IF EF AND P=155 THEN PRINT CHR$(43);
860 IF EF AND P=151 THEN PRINT CHR$(39);
870 IF K=3 THEN GOTO 1120
880 GOTO 1060
890 NC=4:GOSUB 1370:NC=0:IF EF THEN PRINT CHR$(34);
900 IF K=3 THEN GOTO 1120
910 IF CR THEN PRINT CHR$(27);"Q";
920 PLOT C(0),C(1),C(3)
930 IF CR THEN PRINT CHR$(30);
940 GOTO 1120
950 NC=3:GOSUB 1370:NC=0
955 IF EF AND P=145 THEN PRINT CHR$(33);
957 IF EF AND P=146 THEN PRINT CHR$(34);
960 IF K=3 THEN GOTO 1120
970 GOTO 1060
980 NC=3:GOSUB 1370:NC=0:IF EF THEN PRINT CHR$(42);
990 IF K=3 THEN GOTO 1120
1000 IF EF THEN 1060
1010 IF K>=48 AND K<=57 THEN 1060 ELSE 1020
1020 IF CR THEN PRINT CHR$(27);"Q";
1030 PLOT C(0),C(1):PLOT SQR((C(0)-C(2))^2+((C(1)-C(3))/SQR(2))^2)
1040 IF CR THEN PRINT CHR$(30);
1050 GOTO 1120
1060 FOR M=0 TO (I-1) STEP 2
1070 IF CR THEN PRINT CHR$(27);"Q";
1080 PLOT C(M),C(M+1)
1090 IF CR THEN PRINT CHR$(30);
1100 NEXT
1110 IF EF THEN RETURN
1120 IF FL THEN 410 ELSE RESUME 410
1130 '----HANDLE STATES
1140 IF CR THEN CR=0:PRINT CHR$(30); ELSE PRINT CHR$(27);"C~P~G";
      :CR=-1:IF SC<=1 THEN SC=1;
1150 IF SC>1 THEN GOSUB 2840;
1160 GOTO 1280
1170 IF (LO AND &H4) THEN PRINT CHR$(21); ELSE PRINT "~G";
1180 GOTO 1280
1190 IF (LO AND &H2) THEN PRINT "~P"; ELSE PRINT "~R";
1200 GOTO 1280
1210 IF (LO AND &H40) THEN BL$="XB":PRINT "~2"; ELSE BL$="B":PRINT "~1";
1220 GOTO 1280
1230 IF (LO AND &H1) THEN PRINT "~N"; ELSE PRINT "~M~2";
1240 GOTO 1280
1250 IF (LO AND &H80) THEN FI$="XF":PRINT "~L"; ELSE FI$="F":PRINT "~F";
1260 GOTO 1280
1270 IF FL THEN FL=0 ELSE FL=-1:IF TF=0 THEN GOTO 1310
1280 PRINT CHR$(30);CHR$(7);
1290 RESUME

```

```

1300 '----FINISH MAIN----
1310 '----SUB: TABLET INIT
1320 SP$="0":PM=0
1330 PRINT CHR$(30);CHR$(27);"R1C";CHR$(27);"OE5";CHR$(27);"IE5";
1340 IF CR THEN PRINT CHR$(27);"Q";
1350 TF=-1
1360 RESUME 410
1370 '----SUB: COORD
1380 CT=0:I=0:F$="0":OX=0:OY=0:OF$="0"
1390 IF FL THEN 1440 ELSE IF EF THEN GOTO 2540 ELSE 1400
1400 RESUME 420
1410 Z=100*(K-48):CT=CT+1:IF EF THEN GOTO 2580 ELSE RESUME
1420 Z=Z+10*(K-48):CT=CT+1:IF EF THEN GOTO 2580 ELSE RESUME
1430 C(I)=Z+(K-48):CT=0:I=I+1:IF EF AND I<NC THEN GOTO 2580 ELSE
    IF I>=NC OR (NC=999 AND I=2) THEN RETURN ELSE RESUME
1440 '----TABLET COORD
1450 PRINT #4;SP$;
1460 RESUME 1470
1470 IF NOT PM THEN INPUT #4;XX,YY,F$
1480 INPUT #4;XX,YY,F$
1490 PRINT #4;"S";
1500 X1=INT(XX/4.3):Y1=INT(YY/4.3)
1510 IF OX=X1 AND OY=Y1 AND OF$=F$ THEN 1580
1520 IF NC=999 AND F$="1" AND CR THEN PRINT CHR$(27);"Q";
1530 IF (NOT PF OR NOT OC) AND NC=999 THEN IF F$="1" THEN PLOT X1,Y1:
    PRINT CHR$(30);CHR$(7);:GOTO 1580
1540 IF F$="4" THEN NC=0:K=3:RETURN
1550 IF (NOT PF OR NOT OC) AND F$="1" THEN IF NOT PM THEN PRINT CHR$(30);
    "%";CHR$(27);"Q";:PLOT X1,Y
1560 IF (NOT PF OR NOT OC) AND F$="1" THEN PRINT CHR$(7);:C(I)=X1:
    C(I+1)=Y1:I=I+2:IF I>=NC THEN PRINT PR$;:RETURN ELSE 1580
1570 PRINT "~U";:PLOT X1,Y1
1580 OX=X1:OY=Y1:OF$=F$:PF=0:OC=-1
1590 PRINT #4;SP$;:GOTO 1470
1600 '----EXTENDED PRIMITIVES
1610 CMD$="ARR":NC=5:GOTO 1650
1620 CMD$="REC":NC=6:GOTO 1650
1630 CMD$="VEC":NC=5:GOTO 1650
1640 CMD$="DAR":NC=5:GOTO 1650
1650 IF FL THEN IF CMD$="REC" THEN NC=8 ELSE NC=6
1660 GOSUB 1370:NC=0:XS=C(0):YS=C(1):XN=C(2):YN=C(3):W=C(4)
1670 IF K=3 THEN RESUME 410
1680 IF NOT FL AND CHR$(K)="." AND CMD$="REC" THEN NC=2:GOSUB 1370:
    NC=0:C(6)=C(0):C(7)=C(1)
1690 IF FL OR CHR$(K)="." THEN
    W=INT(SQR((C(2)-C(4))^2+((C(3)-C(5))/SQR(2))^2))
1700 IF FL OR CHR$(K)="." THEN IF CMD$="REC" THEN
    C(5)=INT(SQR((C(4)-C(6))^2+((C(5)-C(7))/SQR(2))^2))
1710 IF CMD$="REC" THEN HT=W:W=C(5)
1720 IF FL THEN 1730 ELSE RESUME 1730
1730 IF FI$="F" THEN BF=-1:GOTO 420
1740 GOSUB 1770
1750 IF CMD$="VEC" OR CMD$="REC" THEN DOS"CHAIN PART3
1760 IF CMD$="ARR" OR CMD$="DAR" THEN DOS"CHAIN PART4

```

```

1770 '-----SUB: SAVE BACKGROUND COLOR
1780 COL=PEEK(&H380D)
1790 BG=COL AND &H55
1800 IF BG=0 THEN B1=0
1810 IF BG=1 THEN B1=1
1820 IF BG=4 THEN B1=2
1830 IF BG=5 THEN B1=3
1840 IF BG=16 THEN B1=4
1850 IF BG=17 THEN B1=5
1860 IF BG=20 THEN B1=6
1870 IF BG=21 THEN B1=7
1880 RETURN
1885 '
1890 '-----EDITOR-----Mitchell
1900 '
1910 CNT=1:PRINT CHR$(30);:EF=-1
1920 BTM=(PEEK(&H3B45)*256)+PEEK(&H3B44)
1930 TP=(PEEK(&H3B47)*256)+PEEK(&H3B46)
1940 IF BTM-1=TP THEN GOTO 2510
1945 '
1950 '-----CHANGE FG COLOR TO BG COLOR
1960 GOSUB 1780:PRINT"~C";MID$(STR$(B1),2);
1970 '--PLOT OR CHARACTER MODE
1980 IF (LO AND &H4) THEN GOTO 2270
1985 '
1990 '-----CHARACTER EDITOR
2000 M=PEEK(BTM-CNT)
2010 IF BTM-CNT=TP+3 THEN GOTO 2510
2020 N=PEEK(BTM-(CNT+1))
2030 IF M=21 THEN CNT=CNT+1:PRINT "~G";:GOTO 2270
2040 IF N=1 THEN CNT=CNT+2:GOTO 2000
2050 NN=PEEK(BTM-(CNT+2))
2060 IF N=67 AND NN=1 THEN A=BTM-(CNT+2):Z=BTM-CNT:CNT=CNT+3:
    GOSUB 2600:GOTO 2000 ESLE GOTO 2100
2070 MN=PEEK(BTM-(CNT+3)):MM=PEEK(BTM-(CNT+4))
2080 IF MN=1 THEN IF NN=88 OR NN=89 THEN IF N>=48 AND N<=57 THEN
    IF M=44 THEN A=BTM-(CNT+3):Z=BTM-CNT:CNT=CNT+4:GOSUB 2600:
    PRINT"~X1,~Y1,";:GOTO 2000 ELSE GOTO 2100
2090 IF MM=1 THEN IF MN=88 OR MN=89 THEN IF NN>=48 AND NN<=57
    THEN IF N>=48 AND N<=57 THEN IF M=44 THEN A=BTM-(CNT+4):
    Z=BTM-CNT:CNT=CNT+5:GOSUB 2600:PRINT"~X1,~Y1,";:GOTO 2000
    ELSE GOTO 2100
2100 IF M=0 OR M=7 THEN GOTO 2240
2110 IF M=5 THEN PRINT CHR$(22);:GOTO 2240
2120 IF M=8 THEN PRINT CHR$(32);:GOTO 2240
2130 IF M=6 OR M=9 OR M=12 OR M=13 OR M=23 OR M=27 OR M=28
    THEN PRINT"~C";MID$(STR$(B1+1),2);CHR$(15);
    "USER NEEDS TO PRESS REDRAW "; "~C";MID$(STR$(B1),2);:GOTO 2240
2140 IF M=10 THEN PRINT CHR$(11);:GOTO 2240
2150 IF M=11 THEN PRINT CHR$(10);:GOTO 2240
2160 IF M=14 THEN PRINT CHR$(15);:GOTO 2240
2170 IF M=15 THEN PRINT CHR$(14);:GOTO 2240
2180 IF M=22 THEN PRINT CHR$(5);:GOTO 2240

```

```

2190 IF M=25 THEN PRINT CHR$(31);:GOTO 2240
2200 IF M=29 OR M=32 THEN PRINT CHR$(8);:GOTO 2240
2210 IF M=31 THEN PRINT CHR$(25);:GOTO 2240
2220 IF M>=33 THEN PRINT CHR$(8);CHR$(M);CHR$(8);:GOTO 2470
      ELSE CNT=CNT+1:GOTO 2000
2230 GOTO 2470
2240 CNT=CNT+1:GOTO 2000
2250 '
2260 '----PRIMITIVE EDITOR
2270 J=BTM-CNT
2280 IF J=TP+3 THEN GOTO 2510
2290 M=PEEK(J)
2300 IF M=7 THEN POKE J,0
2310 IF M>32 AND M<44 THEN GOTO 2360
2320 IF M=1 THEN N=PEEK(J+1) ELSE 2350
2330 IF N=71 THEN CNT=CNT+1:GOSUB 2610:PRINT"~C";MID$(STR$(B1),2);:GOTO 2000
2340 IF N=67 THEN A=BTM-CNT:Z=BTM-(CNT-2):GOSUB 2600 ELSE A=BTM-CNT:
      Z=BTM-(CNT-1):GOSUB 2600
2350 CNT=CNT+1:GOTO 2270
2355 '
2360 '--LAST PRIMITIVE FOUND
2370 IF T2 THEN GOTO 2440
2380 IF M=33 THEN P=145:GOSUB 950:GOTO 2440      'X-BAR'
2390 IF M=34 THEN CNT=CNT+2:GOSUB 950:GOTO 2440  'Y BAR'
2400 IF M=37 THEN P=149:GOSUB 2610:GOTO 2440    'DOT'
2405 IF M=40 THEN GOSUB 2610:GOTO 2440          'CONC VECT'
2410 IF M=42 THEN P=154:GOSUB 980:GOTO 2440    'CIRCLE'
2420 IF M=43 THEN P=155:GOSUB 850:GOTO 2440    'RECTANGLE'
2430 IF M=39 THEN P=151:GOSUB 850:GOTO 2440    'VECTOR'
2435 '
2440 '----CHANGE POINTERS & REPLACE COLOR
2450 IF T1 THEN T1=0:CNT=CNT+1:RETURN
2460 IF T2 THEN T2=0
2470 NBTM=BTM-CNT
2480 A=NBTM/256:X%=A
2490 Y%=NBTM-(256*X%)
2500 POKE 15172,Y%:POKE 15173,X%:POKE 14349,COL
2510 PRINT CHR$(27);"Q";:EF=0
2520 RESUME 420
2525 '
2530 '----SUB: COLLECT THE COORD OF THE PRIMITIVE
2540 FOR J=BTM-CNT TO BTM-1
2550 M=PEEK(J)
2560 IF M=0 THEN GOTO 2580
2570 IF M>=48 AND M<=57 THEN K=M:ON CT+1 GOTO 1410,1420,1430
2580 NEXT
2585 '
2590 '----SUB: PUT NULLS IN CB TO REPLACE MODE CHANGES
2600 FOR J=A TO Z: POKE J,0: NEXT: RETURN
2602 '
2605 '----SUB: CHANGE COLOR AND TELL USER TO PRESS REDRAW---
2610 PRINT CHR$(1);CHR$(21);"~C";MID$(STR$(B1+1),2);CHR$(15);
      " USER NEEDS TO PRESS REDRAW ";:RETURN

```

```

2620 -----SEGMENTOR Mitchell
2630 '
2640 PRINT CHR$(30);:SF=-1
2650 IF SC<>1 THEN GOTO 2840
2660 PRINT "2";CHR$(15);"P"X1,"Y1","L"; 'INITIALIZE SCREEN CONDITIONS SO
2670 PRINT CHR$(28);"M"CO"N"C7"; 'MESSAGE WILL APPEAR AT TOP LEFT
2680 LO=PEEK(&H3B52)
2690 IF (LO AND &H4) THEN PRINT CHR$(21);
2700 PRINT "CREATE BUFFER SEGMENTING If you wish to continue,
      name your file ";
2710 FOR I=1 TO 500:NEXT 'DELAY
2720 RESUME 420
2725 '
2730 -----COLLECTS THE FILENAME
2740 PRINT CHR$(K);
2750 IF K=13 THEN 2810
2760 IF K=8 THEN L=LEN(SN$):SN$=LEFT$(SN$,L-1):RESUME 420
2770 SN$=SN$+CHR$(K)
2780 RESUME 420
2790 '
2800 -----PUT FIRST FILE ON DISK
2810 DOS"BUFF "+SN$+"".BUF
2820 SC=SC+1:GOTO 2900
2825 '
2830 -----PUT SUBSEQUENT FILES ON DISK
2840 S$=MID$(STR$(SC),2)
2850 LO=PEEK(&H3B52)
2855 IF RD THEN DOS"KILL "+S$+SN$+"".BUF":RD=0 'REMOVES PREVIOUS FILE WITH
2860 DOS"BUFF "+S$+SN$+"".BUF" 'SAME NAME IF STORED WITHIN
2870 SC=SC+1 'THIS PICTURE PRODUCTION
2880 GOTO 2910
2885 '
2890 -----SET INITIAL CONDITIONS FOR FIRST FILE
2900 PRINT CHR$(12);"M"CO"N"C2"2";CHR$(15);"P"X1,"Y1","G"1";
      CHR$(27);"W";
2905 '
2910 -----DETERMINE TOP OF CREATE BUFFER AND START A NEW FILE
2920 C=PEEK(&H3B46):D=PEEK(&H3B47)
2930 POKE 15172,C:POKE 15173,D
2940 PRINT CHR$(27);"Q";
2950 GOTO 3070
2955 '
2960 -----REDRAW OF SEGMENTED FILES
2970 PRINT CHR$(30);CHR$(12);
2975 IF RD THEN DOS"KILL "+S$+SN$+"".BUF"
2980 S$=MID$(STR$(SC),2):DOS"BUFF "+S$+SN$+"".BUF"
2985 IF NOT RD THEN RD=-1
2990 DOS"DRAW "+SN$+"".BUF"
3000 PRINT CHR$(27);"W";
3010 FOR I=2 TO SC
3020 S$=MID$(STR$(I),2)
3030 DOS"DRAW "+S$+SN$+"".BUF"
3040 PRINT CHR$(27);"W";
3050 NEXT
3055 '
3060 -----TURN SEGMENTOR FLAG OFF, RETURN STATES, GO TO NEW INPUT
3070 SF=0:POKE 9651,LO:RESUME 420

```

BIBLIOGRAPHY

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.
- Chr79 Chromatics CG BASIC Reference Manual, CG File Handling BASIC Version 3.0, Chromatics, Inc., Atlanta, Ga., January 24, 1979.
- Chr80 Chromatics, Inc., Mike Strother and Emerald Duncan, private correspondence, July, 1980.
- Dil80 Dillinger, Marilyn McCord, System Driver for Graphics Computer, 1980.
- New79 Newman, W. and Sproull, R. Principles of Interactive Computer Graphics, Second Edition, McGraw-Hill Book Co., 1979.

EDITING AND SEGMENTING
DISPLAY FILES FOR COLOR GRAPHICS

by

SHARLENE KAY MITCHELL

B.S., Kansas State University, 1968.
M.S., Kansas State University, 1971.

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
1981

ABSTRACT

This report describes two extensions of a system driver designed for the color graphics computer in the Computer Science Department. One extension allows the user to edit drawings or text as they are being developed. This extension is needed to remove misspelled or incorrect positioning of text and incorrect or unexpected geometric figures. Two keys are redefined as the EDIT and TABLET EDIT keys which are used to remove the most recently entered character(s)/figure(s). The second extension allows the user to create complex pictures which previously could not be done because of limited buffer space. This is accomplished by segmenting the display file and maintaining these segments on disk storage. The segmenting facility will not require any action on the part of the user unless the need for segmentation arises.