Chromatics™

NEW DIMENSIONS IN COLOR GRAPHICS

DISK SOFTWARE

REFERENCE MANUAL

CG SERIES

COLOR GRAPHICS COMPUTERS

Models

1398	1598	1998
1399	1599	1999

TABLE OF CONTENTS

TABLE OF CONTENTS -continued

TABLE OF CONTENTS - continued

TABLE OF CONTENTS - continued

TABLE OF CONTENTS - continued

APPENDICES

1. INTRODUCTION

This manual provides a detailed description of the disk resident software which is optionally available from Chromatics, Incorporated when any of the floppy disk peripherals are added to one of the CG series terminals. This manual is intended as a supplement to the Operator's Manual, which should be studied before beginning the present document.

1.1 Scope of the Manual

Chapter 2 covers the Disk Operating System, under which all the other software executes. This chapter must be understood before reading the remaining chapters. The third chapter explains the use of the Text Editor, which is used to create and modify files stored on the disk system. Chapter 4 describes the use of the Z-80 Disk Assembler, which can be used to create executable code for use on the Chromatics CG series terminals. The final chapter covers the PROM Programmer, which may be used to write EPROM integrated circuits such as found in Chromatics terminals. Note that the BASIC Language Interpreter is described in a separate manual since it is a ROM resident rather that disk resident system.

1.2 Terminology and Conventions

Conventions similar to those used in the Operator's Manual, (see especially chapter 1 and section 1.4), will be used here. However, since the non-primary keys will be used only rarely, they will be indicated by underlining, (for example, BOOT). Non-underlined capitals may then be reserved to mean keys struck in sequence without intervening blanks. To indicate clearly and precisely how to enter the commands used by the software, the following conventions will be followed.

1) Each keystroke will be identified by the name given on the key. Keys with multiple character names will be underlined. Blank character input will always be given explicitly, (in the formal command definition), by "SPACE".

2) The key modifiers, CTRL and SHIFT, will immediately precede the single key which they are to modify.

3) Keys are to be struck in order from left to right. All returns and line feeds will be explicitly indicated by RETURN and LF, respectively.

4) All zeroes will be slashed (Ø), and all alphabetic O's will be unslashed.

5) Underlined, lower case words will be used to name one of a set of keys or key sequences. For example, delim is defined to be a sequence of one or more spaces and commas by

   delim ::= SPACE | , | delim SPACE | delim ,

   which is read as "delim is defined to be a SPACE or a comma or a delim followed by a space or a delim followed by a comma." The vertical bars are used to separate the alternatives.

1.3 The Chromatics Disk System

Up to six drives, numbered 1 through 6, may be attached to any Chromatics CG series terminal when Option 41, (Disk Controller with DOS software), is purchased. The drives are identified by the single digit drive number assigned when they are attached. A drive number of Ø is used to specify all the drives together, usually implying a search over all drives in ascending drive number order.

Information is stored on disks in concentric rings called *tracks*. Standard disks have 77 tracks, (numbered 0 through 76), and Minifloppy ^{R} disks have 35 tracks, (numbered 0 through 34). The highest and lowest numbered tracks are reserved for system use. Each track is divided into consecutive records called *sectors*. Standard disks have 26 sectors per track, and Minifloppy ^{R} disks have 18 sectors per track. The sectors are numbered from 1 through the highest sector number. Each sector, (for both types of disks), contains 128 bytes of data. This gives 256,256 bytes of storage per standard disk and 80,640 bytes per Minifloppy ^{R} disk. For more information on disk formatting, see subsection 2.5.6.

2. DISK OPERATING SYSTEM (DOS) (Option 41)

The Disk Operating System is provided with the purchase of Option 41, the Floppy Disk Controller. DOS establishes the environment necessary to execute all disk commands available from Chromatics and those created by the user. This chapter will describe all presently implemented Chromatics commands except for EDIT, ASMB and PROM, which are covered in the remaining chapters of this volume.

2.1 Entry into DOS

The Disk Operating System is entered by either of the following key sequences:

DISK OS | ESC D

DOS expects its commands from logical device AI and displays its output on logical device AO. (See the Operator's Manual, section 3.3.7 for information on logical device assignments.) Note that the output window assigned to AO should usually be in alphabetic, roll mode with a background color of black for maximum readability of DOS responses.

The system notifies the user that he is in DOS by displaying a green asterisk (*) as a prompt character. The asterisk indicates that DOS is ready to accept a disk command.

2.2 File Names

DOS is a file oriented system. Disk commands are the names of files residing on the disk system which may be loaded and executed. The user may thus easily add new disk commands by creating machine language code with the aid of the Text Editor and the Z-80 Assembler, and storing it on the disk. The names of disk files must satisfy the following syntax:

```
        filename  ::=  name.type
where
        name      ::=  alnm  |  alnm alnm  |  . . .  |
                       alnm alnm alnm alnm alnm alnm alnm alnm
        type      ::=  alnm  |  alnm  |  alnm alnm alnm
        alnm      ::=  letter  |  digit
        letter    ::=  A | B | C | D | . . . | Y | Z
        digit     ::=  Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

That is, a filename is a one to eight character alphanumeric string, followed by a period, followed by a one to three character alphanumeric string. Although the user is free to assign any file type that he desires, the system commands recognize only those described in the next section.

2.3 File Types

Nine file types, each consisting of three letters, are used by the various DOS commands. A brief description of each is given here.

2.3.1 ABS

ABS files are absolute, unformatted memory image binary files.

2.3.2 BAS

BAS files are program source files used by the Chromatics BASIC Language Interpreter. For further information, see the BASIC Language Manual.

2.3.3 BUF

BUF files are used to save and restore the contents of the create buffer. See disk commands APPEND, BUFF and DRAW.

2.3.4 DAT

DAT files are used for data storage by the Chromatics BASIC Language Interpreter. For further information, see tha BASIC Language Manual.

2.3.5 KIL

The KIL type designation is used to indicate a file which has been killed and is to be physically deleted by the next COMPRESS command.

2.3.6 OBJ

OBJ files are object code files consisting of records with load address, data bytes, checksums and an optional execution address. CBJ files are produced as output by the Z-80 Assembler. They may be made executable under DOS with the RENAME command by changing the type field to SYS.

2.3.7 PIC

PIC files consist of a direct memory dump of the refresh memory. This allows the screen contents to be saved and restored using the FRAME and REFRESH commands.

2.3.8 SRC

SRC files consist of source lines in ASCII produced by the Text Editor. These files are used to prepare assembly language source code for the Z-80 Assembler.

2.3.9 SYS

SYS files are object code files which may be executed under DOS. SYS files cannot be deleted by the KILL command. They are not listed in the disk directory unless specifically requested.

2.4 Patterns

When referencing disk files, it is frequently convenient to be able to specify a set of files, rather than only one. Patterns are used to select a set of files on a given drive, (or over all drives), which match a template. Disk commands using patterns generally act either on the first matching file found, or else act on all matching files. The syntax for a pattern is given by

```
          pattern  ::=  pat  |  pat/drive  |  /drive
```

where

```
          pat      ::=  namepat.typepat  |  namepat  |  .typepat
          drive    ::=  0 | 1 | 2 | 3 | 4 | 5 | 6
          namepat  ::=  char  |  char char  |  . . .  |
                        char char char char char char char char
          typepat  ::=  char  |  char char  |  char char char
          char     ::=  *  |  letter  |  digit
```

It can be seen that patterns are similar to filenames except that asterisks may be used in addition to alphanumerics, the drive field may be optionally added and other fields may be omitted. When both the namepat and typepat are given without asterisks, the pattern is called "fully specified". A fully specified pattern will always match only the first file found which is identical to the pattern. When asterisks are present, they indicate "don't care" positions in matching with file names. An asterisk in the last position of a field will match any string of zero or more characters. When the namepat or typepat is omitted, it is taken to be a single asterisk, which will match any string. For example, the pattern

"A*" is equivalent to the pattern "A*.*", both of which will match any filename beginning with the letter "A", regardless of type. Also, note that the pattern "*.*" will match all file names.

When the drive field is present, it indicates which disk is to be searched to find matches for the pattern. The special value of drive = Ø is used to indicate that all drives are to be searched in order from 1 to 6. (A value of Ø is not allowed in cases where this would not make sense.) If the drive field is omitted, the last referenced drive is implied.

2.5 Disk Commands

A disk command consists of a reference, (pattern), to a file name followed by zero or more arguments. Multiple disk commands may be entered together as long as they will fit on one line of the screen. The format for a list of commands is

```
        comlist  ::=  command RETURN  |  command : comlist
```

where

```
        command  ::=  pattern  |  pattern delim arglist
        arglist  ::=  arg  |  arglist delim arg
```

The length of the argument list and the types of the arguments depend on the indivual command. The argument lists for each command are discussed in the subsections below.

The command to be executed is determined by finding the first file matching the given pattern, as discussed in section 2.4. However, for a disk command to be valid, the file type must be either BAS or SYS.

For this reason, if the given type is not BAS or SYS, a type field of SYS is substituted. BAS commands are executed by loading the file and calling the BASIC Language interpreter. SYS commands are executed directly under DOS.

When a list of commands, (separated by colons), is given, they are executed one at a time from left to right. Note that some commands may require significant processing and interaction from the user, (e.g., the EDIT command). When a normal return to DOS is made by one command, the next waiting command in the list is executed until the list is exhausted. If an unrecoverable syntax error is found in one of the commands, (such as a MODE code), the commands following the error are ignored.

The disk commands which are presently implemented are listed in the following subsections. These commands are supplied with the system on a master disk. Note that although the command descriptions all use complete file names without drive indication, in all cases a pattern could be used.

2.5.1 DIR

DIR <u>RETURN</u> | DIR <u>delim</u> <u>pattern</u> <u>RETURN</u>

The DIR command lists a directory of all the files matching the given pattern. The omission of the pattern is equivalent to a pattern with both the name and type field omitted. When the name field is omitted from the pattern, a heading is displayed for the directory. The first line of the heading gives the name of the disk followed by three hexadecimal values: NXTK = next free track, NXSC = next free sector and AVSEC = number of available sectors. The second line of the heading gives titles to the columns of the directory: file name, beginning track, beginning sector and length in sectors.

The omission of the type field in the pattern also has a special meaning: SYS and KIL files are not listed. The various combinations of these conventions are illustrated below.

DIR RETURN	*list all but SYS and KIL with heading*
DIR , * RETURN	*list all but SYS and KIL, omit heading*
DIR , .* RETURN	*list all files with heading*
DIR , *.* RETURN	*list all files, omit heading*

When all files are listed, as in the last two examples, SYS files are listed in yellow, KIL files in red and all others in green. If the RETURN is replaced by %RETURN, then only the first 25 files will be listed before a pause will occur. Striking any key except BREAK will continue the list. The BREAK will terminate the command and return the prompt *.

2.5.2 KILL

KILL delim pattern RETURN

The KILL commands kills, (i.e., changes the file type to KIL), all files matching the pattern, except that SYS are specially protected in that they cannot be killed by this command. This is to inhibit accidental destruction of the disk commands. SYS files can be killed, however, using the RENAME command, (this should be done only with great care).

2.5.3 RENAME

RENAME delim old delim new RETURN

where

old ::= pattern

new ::= pattern

The RENAME command changes the name of each file matching the old pattern to a new name based on the new pattern. (The new pattern may not contain a drive field.) The new name will contain each non-asterisk character given in the new pattern. The positions in the new pattern which contain asterisks will be replaced by the characters in the old pattern. For example, suppose a disk has files named ABC.SRC, AEFG.SYS and AA.SYS. After executing the command:

```
          RENAME , A* , X*Y*.KIL RETURN
```

the files will be named XBY.KIL, XEYG.KIL and XAY.KIL, respectively.

Note that using the RENAME command may result in two or more commands with identical names. These can be recovered as individual files by using the RENAME command with a fully specified old pattern. This allows the files to be given unique names one at a time, beginning with the first file in the directory.

2.5.4 COMPRESS

```
          COMPRESS RETURN  |  COMPRESS /drive RETURN
```

The COMPRESS command compacts the disk on the indicated drive, (or the last drive referenced if the first format is used), by deleting all files with type KIL.

2.5.5 COPY

```
          COPY pattern delim /drive RETURN
```

The COPY command is used to copy files from one disk to another. The drive argument must be distinct from the drive implied by the pattern. All files matching the pattern are copied one at a time to the specified drive. If a file with the same name already exists on the target drive, the original file on the target disk is killed before the copy is made. A special case is made when both the name and type fields are omitted from the pattern field. In this case, the entire disk is copied directly so that the target disk becomes a duplicate of the first disk. Note that this implies that any information previously on the target disk will be overwritten.

2.5.6 FORMAT

FORMAT diskname/drive RETURN

where

```
diskname ::= alnm | alnm alnm | . . . |
             alnm alnm alnm alnm alnm alnm alnm alnm
```

Before a brand new disk can be used, it must be sectored and initialized, (formatted). This is done interactively with the user by the FORMAT command. The command first requests the number of tracks, (beginning with track Ø), to interweave with the message "INTERWEAVE TO TRACK NO.". The user responds with a one or two digit hexadecimal number followed by a RETURN. Interweaving is used to allow maximum speed in reading SYS files from the disk. Normally, it is sufficient to interweave through track no. 7. The command then displays the message: "LOAD DRIVE #x AND STRIKE 'F' ". (The "x" represents the drive number where the disk to be formatted should be loaded.) The user responds F to begin formatting. The FORMAT command sends a two digit hexadecimal code indicating the status of each track formatted. Values of "ØØ" indicate success. If any of the values returned differ from "ØØ", the disk should be reformatted.

2.5.7 BUFF

BUFF delim name RETURN

The BUFF command causes the contents of the create buffer to be saved on disk with a filename of name.BUF.

2.5.8 DRAW

DRAW delim namepat RETURN

The DRAW command causes the first BUF file matching the given namepat to be loaded into the create buffer, overlaying the previous contents.

2.5.9 APPEND

APPEND delim namepat RETURN

The APPEND command causes the first BUF file matching the given namepat to be loaded into the create buffer immediately after the file currently in the buffer. This effectively appends the named disk file to the current contents of the create buffer.

2.5.10 PICTURE

PICTURE name RETURN

The PICTURE command causes the entire contents of the refresh memory, (i.e., the screen image), to be stored onto disk with a filename of name.PIC.

2.5.11 REFRESH

REFRESH pattern RETURN

The REFRESH command causes the first file matching the pattern, (which must be of type PIC), to be loaded into the refresh memory. REFRESH restores the image as saved by a PICTURE command.

2.5.12 STORE

```
        STORE filename delim addresslist RETURN  |
        STORE filename delim addresslist@address RETURN
```

where

```
        addresslist  ::=  addresspair  |  addresspair delim addresslist
        addresspair  ::=  address delim address  |
                          address delim address+displacement  |
                          address delim address-displacement
        address      ::=  hex  |  address hex
        hex          ::=  Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
                          8 | 9 | A | B | C | D | E | F
        displacement ::=  hex  |  displacement hex
```

The STORE command creates a disk file with the given filename; the contents of the file are taken from the concatenated memory segments specified by the address pairs. (Note: if an address is given with more than four digits, only the four least significant digits are used.) The address pairs give the first and last byte of each segment. For OBJ and SYS files only, displacements and a start address (@address) may be used. Displacements allow memory segments to be reloaded at locations offset from where they were created. The start address indicates the beginning point of execution for OBJ and SYS files.

2.5.13 FETCH

```
        FETCH pattern delim address RETURN  |  FETCH pattern RETURN  |
        FETCH pattern delim +displacement RETURN  |
        FETCH pattern delim -displacement RETURN
```

The FETCH command causes the first file matching the given pattern to be loaded into memory, regardless of file type. The first format, with the address field, must be used for file types other than OBJ and SYS. In this case, the file is loaded sequentially into memory at the indicated address for as many bytes as the file is long. Care must be taken that RAM locations used by the system (hex addresses 38ØØ through 3FFF) are not accidentally overwritten, as could happen with a large file which loaded beyond the top of memory and wrapped around back past zero.

OBJ and SYS files have load addresses for each memory segment, so any of the last three command formats may be used. If a displacement is given, all memory segments are offset by the indicated amount.

2.5.14 KEYS

KEYS RETURN

The KEYS command defines the user function keys (F1 through F8) in a special and very useful way. After the KEYS command has been executed, each of the keys F2 through F8 can be defined by the user to be any sequence of up to 64 codes. (Note that some keystrokes result in more than one 8 bit code. See the Operator's Manual for the code definition of the various keys.) Since one of the keys may use any of the others in its definition, quite long sequences may be defined. To define a key, F1 is struck, followed by the key to be defined, followed by the key sequence to be stored, and finally followed by the key being defined. For example:

```
BOOT  DISK OS  KEYS  RETURN
F1  F2  A B C  F2
F1  F3  F2  X  F2  F3
```

Now the F3 key will send the sequence: ABCXABC .

2.5.15

```
DUPE Pattern delim/drive RETURN
DUPE /drive delim/drive RETURN
DUPE RETURN
```

The drive command is used to copy files from one disk to another on the same drive (i.e. copy for single drive systems). The drive argument may or may not be distinct from the drive implied by the pattern. All files matching the pattern are copied one at a time to the specified drive. If a file with the same name already exists on the target disk, the original file on the target disk is killed before the copy is made. A special case is made when both the name and type fields are omitted from the pattern field. In this case, the entire disk is copied directly so that the target disk becomes a duplicate of the first disk. NOTE that this implies that any information previously on the target disk will be overwritten.

The DUPE command will pause with one of the two following messages during its operations:

```
     MOUNT SOURCE DISK -- Strike any key! --
     MOUNT TARGET DISK -- Strike any key! --
```

At this point, the correct diskett should be inserted, door closed securely on the drive and any key (other than RESET) struck. NOTE that when individual files are being copied,the same message may occur twice in a row. This is normal, simply strike the key a second time and proceed to the next message. WARNING: Do not try to copy a file from one diskett to the same diskett. This will work but the program will not end until the file has been copied enough times to completely fill the entire empty space on the diskett.

2.6 Initializing a New System Disk

One of the common tasks in using the disk system is initializing new disks. It is usually advisable to copy some or all of the command files onto a new disk to facilitate future processing. This section will illustrate how to do this with multiple and single drive systems. The sample interactive sessions will be given with the user responses in sans serif typeface. Comments will appear to the right in script.

2.6.1 Multiple drive systems

The initialization and copying task is quite simple with a multiple disk drive system. Assume that an existing system disk is mounted on drive 1 and a new disk is mounted on drive 2. The interactive task might then appear as follows:

```
BOOT   DISK OS
*FORMAT/1 , DISKNAME/2 RETURN           new disk named 'DISKNAME'
INTERWEAVE TO TRACK NO. 7 RETURN        interweave through track 7
LOAD DRIVE #2 AND STRIKE 'F' F          execute
ØØØØØØØØ ...  ØØ                        no errors
*COPY/1 , *.SYS , /2 RETURN             copy all SYS files
*DIR/2 , .*  RETURN                     check new disk
```

2.6.2 Single drive systems

The above task is much more difficult on a single disk system since the source files cannot be mounted at the same time as the object disk. The COPY command thus cannot be used. Instead, the CPU operating system may be used. (See Chapter 5 of the Operator's Manual.) Let the drive number of the single drive be 1. Assume that all SYS files reside in the first 8Ø (hex) sectors of the original disk and that there is sufficient RAM to load all 8Ø sectors at once. Then the initialization and copying might be done as follows.

```
BOOT   DISK OS
*FORMAT , DISKNAME RETURN                now replace disk
INTERWEAVE TO TRACK NO. 7 RETURN
LOAD DRIVE #1 AND STRIKE 'F' F           execute
ØØØØØØØØ ... ØØ
*CPU OS                                  reload original disk
#R1  Ø 1 8Ø 4ØØØ                         read 8Ø sectors into 4ØØØ
                                         reload new disk
#W1  Ø 1 8Ø 4ØØØ                         write back 8Ø sectors
#DISK OS                                 return to DOS
*DIR , .* RETURN                         check new disk
```

Note that, since the directory is copied directly to the new disk, the diskname given by the FORMAT command will be overwritten, (i.e., the new disk will have the same name as the original). Also, if more information was on the original disk beyond the first 6Ø tracks, the directory will be in error. This can be corrected by KILLing the extra files and compressing the disk. Assume that all the additional files are non-SYS files. Then this could be accomplished as follows, (continuing the previous example):

```
*KILL , *.* RETURN         Kill non-SYS files
*COMPRESS   RETURN         delete them
*DIR , .*   RETURN         check
```

If more than 6Ø sectors must be copied, several reads and write could be done using CPUOS and following the above example. If less RAM is available, shorter segments must be copied.

2.7 Error Messages

Many errors may occur while using the disk operating system. The system notifies the user of the error by the message "ERROR #-" in yellow, followed by a two digit error code in red. An explanation of these codes is given in this section.

2.7.1 ERROR # 10 - CRC ERROR CODE

This error indicates that a cyclic redundancy check has failed, implying that there may be an error in the file last read. The system retries several times on a error of this type, so there is probably bad data on the disk. Try the disk on a alternate drive to check for an error in the drive hardware.

2.7.2 ERROR # 11 - SEEK ERROR CODE

The track sought cannot be found. This error may be caused by an unformatted disk, bad seek logic or a disk with bad data. Try the disk on an alternate drive.

2.7.3 ERROR # 12 - RECORD NOT FOUND

The sector sought cannot be found on the designated track. The causes of this error are similar to those for ERROR # 11.

2.7.4 ERROR # 13 - LOST DATA

The occurrence of this error indicates a hardware related problem.

2.7.5 ERROR # 14 - NOT READY

The addressed disk drive is not ready. Make sure that the drive is on and that the disk is properly loaded.

2.7.6 ERROR # 15 - WRITE PROTECT

An attempt has been made to write on a disk with the file protection slot open. First verify that the disk should be written. If so, close the write protect slot with opaque tape and repeat the operation.

2.7.7 ERROR # 20 - FILE NOT FOUND

The named file could not be found in the directory of the indicated disk.

2.7.8 ERROR # 21 - EMPTY SLOT FOUND

This is an internal system message indicating that space exists for a new file. If it occurs in normal operations of DOS, then it carries the same meaning as ERROR # 20.

2.7.9 ERROR # 22 - NO HEX EOF

The referenced command file is in error in that no end of file can be found.

2.7.10 ERROR # 30 - CREATE BUFFER OVERFLOW

The end of the create buffer has been exceeded when loading a file with the DRAW or APPEND command.

2.7.11 ERROR # 31 - COMMAND ERROR

The last disk command given did not properly reference a system file. This error frequently occurs when a RETURN is given in response to the disk system prompt.

2.7.12 ERROR # 32 - NO RUN ADDRESS

The system file referenced as a command does not have a start address stored and thus cannot be executed.

2.7.13 ERROR # 33 - ILLEGAL CHARACTERS

Illegal characters detected in a command. This error oftern occurs when control codes are inadvertantly placed in a command, (e.g., using the color keys within a command).

2.7.14 ERROR # 34 - NO ARGUMENTS

One or more of the arguments required by the command are not present.

2.7.15 ERROR # 40 - ILLEGAL FILE NAME

The referenced file name is not legal in the context given. For example, this message will result from a KILL *.SYS command.

2.7.16 ERROR # 41 - BAD DISK DRIVE NUMBER

The given drive number is illegal in the context given. The message may be given by an attempt to copy a file back onto the sending drive.

2.7.17 ERROR # 42 - FATAL DISK ERROR

An unrecoverable error has been detected on the disk.

2.7.18 ERROR # 43 - DUPLICATE FILE NAME

This error will occur if the BUFF command is used to try to store a file with the name of an existing file.

2.7.19 ERROR # 44 - DATA ERROR

A data error check has occurred on the disk.

2.7.20 ERROR # 45 - DISK OR DIRECTORY OVERFLOW

Insufficient space has been found to store a file, either because the disk file storage area or the directory has become full.

3. TEXT EDITOR (Option 62)

The Chromatics Text Editor facilitates the creation and modification of ASCII source files, which are primarily used as input to the Z-80 Assembler. Both line editting and screen editting capabilities are available. This chapter is organized by logical groupings of features of the editor. The final section provides a brief, alphabetical list of all commands.

The Text Editor accepts commands on logical device AI and displays information to the user on logical device AO. (Logical device BO is also used by one command, see subsection 3.6.2.) For convenience, the display is automatically placed in roll mode upon entering the Editor. The background is set by the user, usually to black.

3.1 Entering the Text Editor

The Text Editor operates under the disk operating system. It may be entered either by using the special key provided or from DOS.

```
TEXT EDIT  |  DISK OS  EDIT  RETURN   |
DISK OS  EDIT  delim  pattern  RETURN
```

Using the special key is equivalent to using the second form shown above. The third form shown enters the editor and opens the file specified by pattern for input.

The Text Editor may also be re-entered, (once it is resident in memory), by the following key sequence:

RESET ESC E

This re-entry technique is sometimes useful in terminating a "run-away" command, (e.g., listing a very long file). This command does not affect the file in the workspace.

3.2 Edit Commands

When the Text Editor is entered, the following display is made

```
CHROMATICS DISK TEXT EDITOR  VER 2.Ø

COMMAND:
```

The "COMMAND:" message appears as a prompt whenever the Editor expects another command. Every command begins with an alphabetic command name, followed by zero or more arguments and terminated by a RETURN. All command names may be abbreviated to two characters.

3.3 Current Line Controls

Each command line prepared for the Text Editor is built up in a 75 character line buffer. Until the RETURN key is struck, causing the command to be executed, the command may be modified as necessary. The skip and backspace keys, (→ and ←), may be used to move the cursor to the point at which a modification is to be made. (Note: the ↑ and ↓ keys have no affect.) Five additional special control codes are also recognized. These are explained below.

3.3.1 Delete current line

CTRL X

This causes the entire line to be deleted, (replaced by blanks), and moves the cursor to the leftmost position of the line.

3.3.2 Delete to end of line

CTRL Z

This causes the character under the cursor and those to its right to be replaced by blanks. The cursor is not moved.

3.3.3 Recall last line

CTRL R

This causes the current line to be replaced by the contents of the previously executed line. This feature is often useful when a command must be repeated several times, perhaps with small alterations. (Note: the two most recent lines are saved in a stack for retrieval.)

3.3.4 Delete current character

CTRL F

The character under the cursor is deleted and all characters to the right of the cursor are shifted left one character postion.

3.3.5 Insert space

CTRL W

Each character of the line beginning with the character under the cursor and to its right is shifted one character to the right. A space is inserted at the cursor position. The cursor is not moved.

3.4 Files

The Text Editor makes use of three files: an input file, a working file and an output file. These files are described in this section. The following section describes the file handling capabilities of the Editor.

3.4.1 Input file

The input file serves as the source of previously stored text lines to be editted. Lines are brought in from the input file using the GET command. Several input files may be used in an edit, but only one may be open at a time.

3.4.2 Working file

The working file resides in memory. All editting changes are made to the working file. No line numbers are stored, but the lines of the working file are implicitly numbered beginning with ØØØØ. Note that deleting and inserting lines immediately changes the implicit numbering for all successive lines. The area of memory used by the working file is called the "workspace".

3.4.3 Output file

The editted text produced in the working file is eventually sent to the output file, (if it is to be saved). A new output file is automatically created when the first lines are sent, (an existing file may not be overwritten). The output file is not entered into the directory until a CLOSE command is given. Unclosed output files are implicitly deleted.

3.5 File Handling

The Text Editor works with the input, output and working files. Since the workspace is limited in size, provision is made to edit files by segments. Groups of lines may be brought in from the input file, editted in the working file and sent to the output file. By progressing sequentially, a file of any size may be editted. The commands necessary to control the handling of files are explained in this section. An illustration of the use of these commands is given in Figure 3-1.

3.5.1 Opening an input file

OPEN delim pattern RETURN

The first file matching the pattern is opened for input. The file pointer for this file, (which must be of type SRC), is set to the beginning of the file. Re-opening an already open input file is equivalent to rewinding it.

3.5.2 Getting lines from the input file

GET delim lines RETURN

where

lines ::= number

number ::= digit | number digit

Beginning after the last line in the working file, the number of lines specified by lines are read from the input file. The file pointer for the input file is moved to a point following the last line read. If there are fewer than lines lines remaining in the input file, the input file is closed, and a message is sent to the operator. If there is insufficient space in the workspace for all of the lines requested, only as many lines as will fit are read, and a "WORKSPACE FULL" message is sent to the operator.

AN EXAMPLE

Let A be a file of 2ØØØ lines on drive 1. The following key sequence is one way to create three new files: B and C on drive 1, and D on drive 2. File B is a copy of the first 5ØØ lines of file A. File C is a copy of the next 1ØØØ lines. File D is a copy of the last 5ØØ lines of file A. It is assumed that the workspace is large enough to hold at least 1ØØØ lines of text.

```
TEXT EDIT
OP  A/1  RETURN
GET 1ØØØ  RETURN
PUT Ø+5ØØ  RETURN
CL  B  RETURN
GET 5ØØ  RETURN
PUT  Ø,9ØØ  RETURN
CLOSE  C  RETURN
DR  2  RETURN
GET  5ØØ  RETURN
PUT  Ø,9999  RETURN
CL   D  RETURN
```

Figure 3-1

3.5.3 Assigning the output drive

DRIVE delim drive RETURN

The drive number specified by drive is assigned for the use of the output file. Before the first DRIVE command is given, the drive of the last referenced file is used as an implicit drive number. The output drive number may be reassigned any number of times during a run. WARNING: if the DRIVE command is given while an output is unclosed, the file will be lost.

3.5.4 Sending lines to the output file

PUT delim range RETURN

where

```
range  ::=  first delim last   |   first+lines
first  ::=  number
last   ::=  number
```

The PUT command appends the specified lines to the output file. If no output file is currently open, then a new one is opened on the drive set by the last DRIVE command.

The range field in the PUT command indicates a range of lines to be output. The first form of range specifies the first and last line number of the lines to be sent. The second form specifies the first line and the number of lines to be sent. All lines of the working file can always be sent by the command: PUT Ø,9999 RETURN. In normal operation, the first field will be Ø, since this is the first line of the working file. However, it is possible to select subsequences of lines for output; this technique can be used to effectively move blocks of source lines. After the lines are sent to the output file, they are deleted from the working file.

3.5.5 Closing an output file

```
CLOSE delim name RETURN   |   CLOSE RETURN
```

The current output file is closed and entered into the disk directory with the name name.SRC. When the second form is used, the name of the last opened input file is used for name, (if no input file has been opened, a blank name results). If a file with the same name as the closed file already exists in the directory, the older file is killed. This allows a file to be updated easily without changing names. Note that once a file has been closed, it can be immediately re-opened for input.

3.5.6 Returning to DOS

```
EXIT RETURN
```

The EXIT command causes a direct return to the disk operating system.

WARNING: an unclosed output file will be lost as a result of the EXIT command.

3.5.7 Examining the state of the workspace

```
BYTES RETURN
```

The BYTES command displays the decimal number of bytes in use and still available in the workspace.

3.6 Displaying Text

The text in the workspace can be displayed either with or without line numbers. Both commands are explained in this section.

3.6.1 Listing with line numbers

LIST delim range RETURN | LIST RETURN

The lines in the indicated range are displayed on the screen, (logical device AO), prefixed by the line numbers in green. If the range field is omitted, it is taken to be Ø,9999. Note: each line number displayed is begun with a 'set foreground to green' mode code. This may have an effect on the appearance of the listing if color changes are used.

The facilitate reading of the listing, a pause is inserted after each page is listed. (The number of lines in a page can be changed by the PAGE command, see subsection 3.6.3.) To continue the listing after the pause, type RETURN. To terminate the listing early, type RESET ESC E, which will cause a return to the command mode.

3.6.2 Printing without line numbers

PRINT delim range RETURN | PRINT RETURN

The PRINT command is similar to LIST except that no line numbers are inserted. Omission of the range causes the entire file to be printed. The PRINT command has the special feature that the output goes to both logical devices AO and BO. BO is normally assigned to SIO #Ø. Presumably, SIO #Ø will be attached to a hardcopy device so that a permanent printed listing can be made.

3.6.3 Changing page size

PAGE delim lines RETURN

The PAGE command resets the length of the page as used by the LIST command. The default page length is 4Ø lines.

3.7 Line Editting

The Chromatics Text Editor provides features for adding deleting, modifying and searching for lines of text. All of these features make use of the implicit line numbers of the working file. The current line numbering can be determined with the aid of the LIST command, (see subsection 3.6.1).

3.7.1 Inserting new lines of text

```
          INSERT delim linenum RETURN  |  INSERT RETURN
```

where

```
          linenum  ::=  number
```

The INSERT command puts the Editor into insert mode. All successive lines are inserted sequentially into the working file preceding the indicated line number. If the linenum field is omitted, it is taken to be Ø, causing the new lines to be inserted at the beginning of the working file. If linenum is larger than the highest line number in the file, the new lines are appended to the end of the file. The Editor remains in the insert mode until an ERASE PAGE or a BREAK is given, either of which causes a return to the command mode.

All of the line editting features described in section 3.3 apply to lines created in insert mode. Three control codes, (TAB, MODE and RETURN), may be used in text. TAB is displayed on the input line as "¦", and MODE is displayed as "~". When RETURN is struck, (regardless of the location of the cursor), the following things happen: a RETURN character is appended to the input line, the line is inserted in the working file and the line is displayed with its assigned line number. When text lines are displayed, the TAB's and MODE's are executed according to their definitions, and a line feed (LF) is inserted after each RETURN. The compressed form for text, (with TAB's and MODE's shown as printing characters), is used only on input and in modify mode, (see section 3.8).

3.7.2 Deleting lines from the text

DELETE delim range RETURN

The DELETE command deletes the lines in the indicated range and compacts the workspace. The range field is required.

3.7.3 Finding lines in the text

FIND delim range \ string \ RETURN

wherer

string ::= txtchar | txtchar string

txtchar ::= *any character that may legally appear in text*

The FIND command lists all lines in the given range, (with line numbers), which contain a substring matching string.

3.7.4 Changing lines of the text

SUBSTITUTE delim range \ oldstring \ newstring \ global

where

oldstring ::= string

newstring ::= string

global ::= G RETURN | RETURN

If global = G RETURN, then every occurrence of oldstring in the lines in the given range is replaced by newstring. If global = RETURN, only the first occurence on each line is replaced.

3.8 Screen Editting

MODIFY delim linenum RETURN | MODIFY RETURN

The MODIFY command places the Editor in modify mode. The indicated line is displayed in the center of the screen in compressed form, (see subsection 3.7.1). (Note: if the screen center is not within the output window, the line will be displayed at the top or the bottom of the window.) The line to be modified is loaded into the current line buffer and may be modified in place exactly as if it were a newly input line, using all the facilities described in section 3.3. Once the line has been corrected to satisfaction, a RETURN must be given to cause the modified line to replace the original line in the text.

Additional lines may be modified while in modify mode by scanning up and down throught the text using the cursor controls, (↑ and ↓). As the cursor moves through the text lines, the line ready for modification appears in magenta. The remaining lines which have been scanned are listed in green.

To return to the command mode, either an ERASE PAGE or a BREAK may be given. Also, if the cursor is on the last line of text in the working file, a RETURN causes the Editor to go into insert mode, to allow additional lines to be added.

3.9 Summary of Editting Commands

The following alphabetical list may serve as a convenient reminder of the available editting commands. Remember that all commands may be abbreviated to two characters.

```
BYTES  RETURN
CLOSE delim name  RETURN
DELETE delim range  RETURN
DRIVE delim drive  RETURN
EXIT  RETURN
FIND delim range \string\  RETURN
GET delim lines  RETURN
INSERT delim linenum  RETURN  |  INSERT RETURN
LIST delim range  RETURN      |  LIST  RETURN
MODIFY delim linenum  RETURN  |  MODIFY  RETURN
OPEN delim pattern  RETURN
PAGE delim lines  RETURN
PRINT delim range  RETURN     |  PRINT  RETURN
PUT delim range  RETURN
SUBSTITUTE delim range \ oldstring \ newstring \  RETURN
```

4. Z-80 ASSEMBLER (Option 63)

The Chromatics Z-80 Assembler is a two pass assembler for translating Z-80 assembly language disk source files into machine executable object code. There are no explicit program size restrictions, but the internal symbol table is limited to 500 or 2500 six character symbols, depending on the size of available RAM.

4.1 Entering the Assembler

The Z-80 Assembler operates under the disk operating system. It may be entered either by using the special key provided or from DOS.

ASMB | DISK OS ASMB RETURN |
DISK OS ASMB delim patternlist RETURN

where

patternlist ::= pattern | patternlist delim pattern

The first two forms shown are equivalent. The third form enters the assembler and begins execution of an assembly with the files specified by the pattern list, (see subsection 4.2.9).

4.2 Assembler Commands

Assembler commands control the various options of the Assembler and direct the file handling. The command formats follow those of the Text Editor: a command name, (which may be abbreviated to two characters), followed by an argument list depending on the command and terminated by a RETURN. All assembler commands are listed in this section.

4.2.1 Set output mode to absolute

ABSOLUTE RETURN

The ABSOLUTE command directs the Assembler to produce output in AbsoluteBinary form. The resulting file is made up of contiguous bytes of code and cannot be loaded into separate areas of memory. This type of object file is the most efficient in disk space and load time. Absolute output should only be used on programs with a single origin. The output file produced is type ABS.

4.2.2 Set output mode to binary

BINARY RETURN

The BINARY command directs the Assembler to produce output in Load Module Binary form. This type of file is designed to be executed within the Chromatics CG series. Any number of origin statements are allowed. The output file is of type OBJ.

4.2.3 Inhibit assembly listing

NOLIST RETURN

The NOLIST command turns off the list switch. No text listing will be produced by the Assembler.

4.2.4 Turn on assembly listing

LIST RETURN

The LIST command turns on the list switch, which causes a listing of the text to be produced during assembly. The pause switch is set off.

4.2.5 Set pause control

PAUSE RETURN

The PAUSE command sets the pause switch on. This causes a pause after every 45 lines of listed text during assembly. A RETURN restarts the listing.

4.2.6 List symbol table

SYMBOL RETURN

The symbol command causes the symbol table to be listed immediately. This command is only effective after an assembly has been executed.

4.2.7 Direct output to alternate device

TTY RETURN

The TTY command causes subsequent output from the Assembler to go to output device BO. Normally, BO is assigned to SIO #0, which is presumably attached to a hardcopy printing device, such as a teletype.

4.2.8 Direct output to standard device

CRT RETURN

The CRT command reverses the effect of the TTY command and causes subsequent Assembler outputs to go to logical device AO, presumably attached to a window.

4.2.9 Execute assembly

ASSEMBLE delim patternlist RETURN

The ASSEMBLE command causes the assembly of the files specified by the pattern list. Each pattern in the list specifies one file. The files are concatenated in the order given and assembled as a single program, (all labels are global). All of the input files must be of type SRC. The assembly creates an object code file on the same drive as the last file specified by the pattern list.

4.2.10 Close output file

CLOSE delim name RETURN

The CLOSE command closes the last output file created by the Assembler and enters it in the directory with the file name name.OBJ or name.ABS. The output need not be closed if errors occurred during assembly; in this case, the file will effectively deleted.

4.2.11 Leaving the assembler

EXIT RETURN

The EXIT command causes a return to the disk operating system. WARNING: any unclosed output file will be lost if it is not closed before executing this command.

4.3 Syntax of Assembly Language Statements

The Chromatics Z-80 Assembler allows free format input of assembly language statements. The syntax of a statement is given by

statement ::= stat RETURN | stat ; comment RETURN |
; comment RETURN

where

comment ::= *any character sequence excluding* RETURN*'s*

stat ::= label sp instruction | sp instruction

label ::= letter | letter alph | letter alph alph |
... | letter alph alph alph alph alph

sp ::= SPACE | TAB | SPACE sp | TAB sp

instruction ::= stdinstr | pseudoinstr

More informally, a statement consists of an optional label, followed by an instruction, followed by an optional comment. It is also allowable to have a comment only line. Labels may be from one to six alphanumeric characters in length, beginning with a letter. Two types of intructions are available - standard instructions, which are standard Z-80 mnemonics, and pseudo-instructions, which are defined by Chromatics. These two instructions types are described in the next two sections.

4.4 Chromatics Pseudo-instructions

Chromatics has defined six pseudo-instructions for controlling assembly, reserving space and initializing constants. The syntax for each and its meaning are described in the following subsections. Most of the operations require operands, which may be expressions of the form:

exp ::= constant | constant + exp | constant - exp

where

constant ::= $ | number | hexnum | label | literal

hexnum ::= hex H | hex hexnum

literal ::= " symbol "

symbol ::= *any character except* "

The "$" is evaluated as the current value of the location counter. Number and hexnum are evaluated as their obvious numerical value. Literals are evaluated as the numeric value of the ASCII code of the symbol. Labels are evaluated as the address associated with the label. For all pseudo-instructions, forward referencing of labels is not allowed.

4.4.1 Origin

ORG sp exp

The internal assembly program counter is set to the value of the expression, exp, which is evaluated as a 16 bit quantity. Subsequent instructions will be assembled to load from the point specified as the origin. Multiple origins are allowed for assemblies done in BINARY mode.

4.4.2 Equate

EQU sp exp

The equate instruction is used to define symbolic names as labels or constants. Note that this instruction must have a label to be meaningful. Previously defined labels are allowed in the expression.

4.4.3 Define storage

DFS sp exp

The define storage instruction reserves the number of bytes specified by exp, which must evaluate to less than or equal to 65535. A label used with this instruction will be equated to the first byte reserved.

4.4.4 Define bytes

DFB sp explist

where

explist ::= exp | explist , exp

The define bytes instruction reserves one byte for each element in the expression list. The byte is initialized to the value of the associated expression, evaluated as an eight bit quantity. If the statement is labeled, the label is equated to the first byte reserved. NOTE: strings of symbols are allowed as well as single symbols in literals for this instruction. In this case, a string is treated as a list of single symbol literals. For example, the following two instructions are equivalent:

```
DFB  "ABC"
DFB  "A","B","C"
```

4.4.5 Define word

```
DFW  sp  exp
```

The define word instruction is used to reserve two bytes of storage, which are set to the value of exp, evaluated as a 16 bit quantity. Because of the method of using word quantities in the Z-80, the low order byte of the quantity is stored in the first byte reserved, and the high order byte in the second byte. For example, the statements:

```
ORG  Ø  RETURN
DFW  Ø1Ø2H  RETURN
```

would store the value Ø2 in byte ØØØØ and the value Ø1 in byte ØØØ1.

4.4.6 End

```
END    |   END delim exp
```

The END instruction is used to tell the assembler to terminate the pass. If the second form of the END statement is used, the value of the exp is used as the start address for the object element produced. The END instruction is optional unless several source files are to be assembled together.

4.5 Standard Z-80 Instructions

No attempt will be made to describe the Z-80 standard instructions in this manual, although a brief summary is given in Appendix D. For a more detailed explanation, see the Z-80 CPU manual published by Zilog or Mostek. This section will explain the syntax expected by the Chromatics Assembler.

The Z-80 has 18 eight bit and 4 sixteen bit registers. Those which are explictly addressable have reserved names recognized by the Chromatics Assembler.

Eight Bit Registers

A	accumulator
F	flags
B,C,D,E,H,L	general purpose
I	interrupt vector
R	memory refresh

Sixteen Bit Registers

IX,IY	index registers
SP	stack pointer
PC	program counter
BC,DE,HL	general purpose
AF	A concatenated with F

Single Bit Flags (in F)

C	carry
NC	no carry
Z	zero
NZ	not zero
P	sign positive
M	sign negative
PO	parity odd
PE	parity even

The 16 bit register BC, DE and HL reference pairs of 8 bits registers. For further information on the meaning and use of these registers, see the Z-80 CPU manual.

The syntax of the standard Z-80 instructions is given below. The number and types of the operands depends on the opcode.

```
stdinstr ::=  opcode | opcode sp operand |
              opcode sp operand , operand
```

where

```
operand  ::=  reg8 | reg16 | const | indirect | index | flag
reg8     ::=  A | F | B | C | D | E | H | L | I | R
reg16    ::=  IX | IY | PC | BC | DE | HL | AF | SP
indirect ::=  ( reg16 ) | ( const ) | ( C )
index    ::=  ( IX + const ) | ( IY + const )
flag     ::=  C | NC | Z | NZ | P | M | PO | PE
const    ::=  exp
```

Not all types of operands are accepted by all opcodes, but they generally have a consistent meaning. The register operands obviously correspond to the appropriate registers. Constant operands, which may be expressions with forward references, are used to provide immediate data, such as an offset for a jump relative instruction. Indirect operands reference memory locations indirectly; the value stored in the indicated register or the value of the constant is used to point to an address. The special operand (C) is used only in I/O instructions. Index operands also reference memory indirectly with the value of an index register offset by a displacement.

4.6 Assembler Error Messages

The Chromatics Z-80 Assembler makes two passes through the source file to create object code. The first pass makes address assignments and builds the symbol table, while the second pass produces the actual code. Each pass may detect errors during assembly. Errors are indicated as single character codes which appear to the right of the machine code in the assembly listing. The meaning of these codes is given below.

4.6.1 A - Argument error

An illegal type of form of argument or a missing argument has been detected. This error can occur only in pass 2.

4.6.2 D - Duplicate label

A label has been found which already appears in the symbol table. This is a pass 1 error.

4.6.3 L - Label error

A syntactically incorrect label has been detected in pass 1. The label may begin with a digit, contain an illegal character or be too long.

4.6.4 M - Missing label

A label has not been found where expected. This error usually occurs when a forward reference is made by an EQU.

4.6.5 O - Opcode error

The operation code given is unknown. This is a pass 2 error.

4.6.6 P - Paging error

An attempt has been made to reference an address outside of the range of +129 to -126 in a jump relative instruction. This is a pass 2 error.

4.6.7 S - Syntax error

The statement is syntactically incorrect. This is a pass 2 error.

4.6.8 U - Undefined symbol

The referenced symbol is not in the symbol table and is therefore undefined. This error may occur on either pass.

4.6.9 V - Value error

The evaluation of the constant expression has led to a value which is outside the range that can be stored. This error may occur on either pass.

4.7 Using CRTOS I/O with Assembly Language

The CRT Operating System provides convenient and flexible input and output to all devices, including the pseudo-devices called windows, (see the Operator's Manual). The user is advised to make use of these facilities by using the standard system I/O routines.

4.7.1 Input

There are five logical devices available on the system. The standard addresses for the routines to reference these devices can be defined for ease of use as follows:

```
AI      EQU     17DFH          RETURN
BI      EQU     17E2H          RETURN
CI      EQU     17E5H          RETURN
DI      EQU     17E8H          RETURN
EI      EQU     17EBH          RETURN
```

Input characters are received one character at a time in the accumulator, (register A). If the eight bit input character is not ready, the input routine returns immediately with the Z flag set to 1. The normal input technique, illustrated below, is therefore to loop until the character has been received.

```
INPUT   CALL    AI     ; GET CHAR FROM DEVICE AI   RETURN
        JR      Z,INPUT-$                          RETURN
```

4.7.2 Output

There are also five logical output devices.

```
AO      EQU     17EEH          RETURN
BO      EQU     17F1H          RETURN
CO      EQU     17F4H          RETURN
DO      EQU     17F7H          RETURN
EO      EQU     17FAH          RETURN
```

No looping is required on output since the output routine will not return until the operation is complete. The character to be sent must be loaded into the accumulator before the routine is called. The example sends a "Z" to device AO.

```
        LD      A,"Z"          RETURN
        CALL    AO             RETURN
```

5. PROM PROGRAMMER (Option 52)

The Chromatics PROM Programmer provides the capability to produce customized ROM's by writing into Erasable PROM IC's. Option 52 provides the hardware programmer, the interface and the software required. This chapter describes the software.

5.1 Entering the PROM Programmer

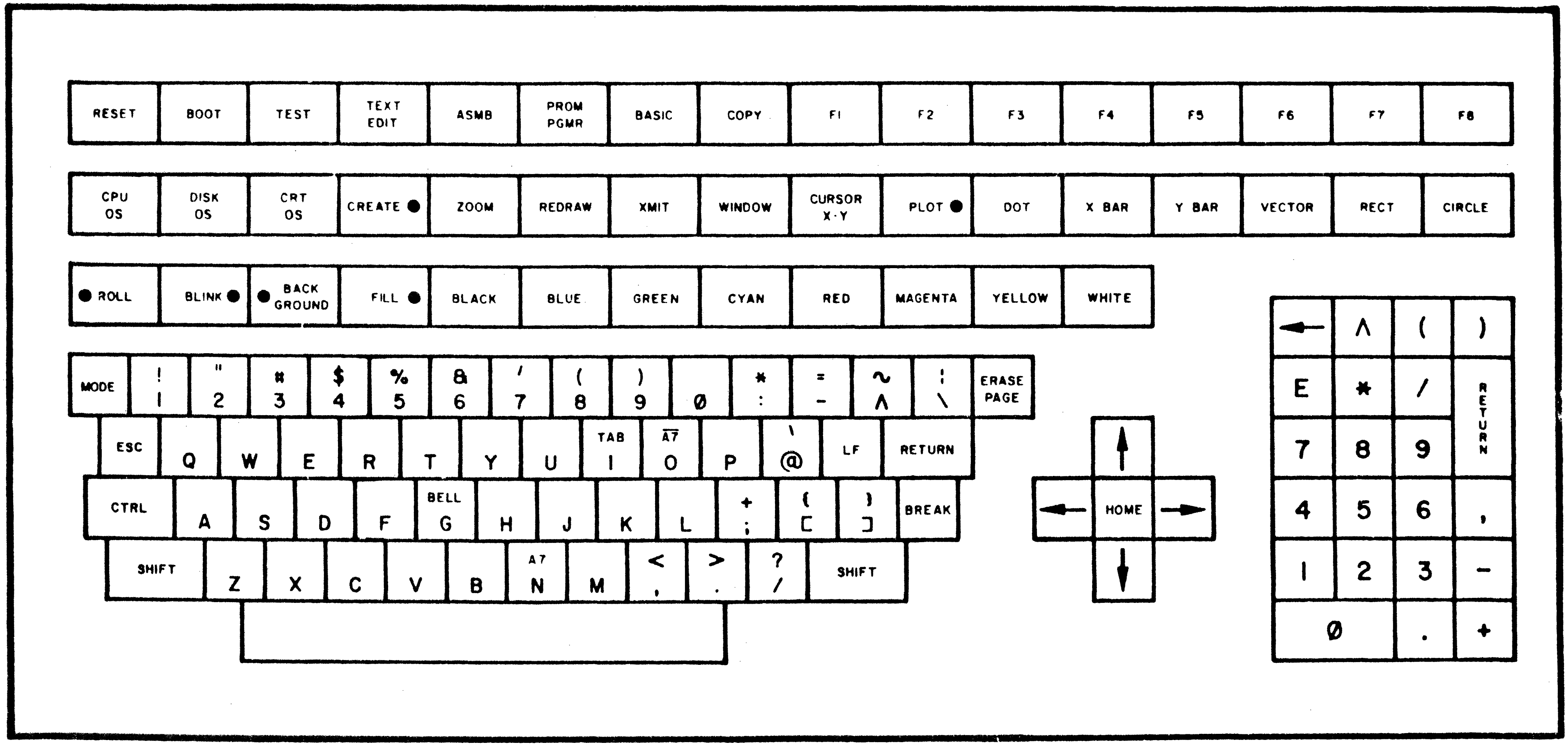
The PROM Programmer may be entered either by using the special key provided or from the disk operating system.

PROM PGMR | DISK OS PROM RETURN

Note: the PROM Programmer software is currently undergoing development, and the user interface has not been finalized.

APPENDICES

KEYBOARD LAYOUT



NOTE: ● INDICATES ILLUMINATED KEY.

ASCII CODE ASSIGNMENT

APPENDIX B. ASCII CODE ASSIGNMENT

				HEX	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
				A7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
				A6	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				A5	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				A4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
HEX	A3	A2	A1	A0	CONTROL @ TO O	CONTROL P TO _	SHIFT Ø TO ;	SHIFT , TO \			SHIFT @ TO O	SHIFT P TO _	CONTROL @ TO O	CONTROL P TO _						
0	0	0	0	0	NULL 0	✕	SPACE 32	0 48	@ 64	P 80	` 96	p 112			SPACE 32	┌ 48	┼ 64	80	96	⊥ 112
1	0	0	0	1	MODE 1	✕	! 33	1 49	A 65	Q 81	a 97	q 113			█ 33	┘ 49	┴ 65	81	97	T 113
2	0	0	1	0	✕	✕	" 34	2 50	B 66	R 82	b 98	r 114			↓ 34	┐ 50	┬ 66	82	98	┤ 114
3	0	0	1	1	✕	✕	# 35	3 51	C 67	S 83	c 99	s 115			↑ 35	└ 51	┤ 67	83	99	├ 115
4	0	1	0	0	✕	✕	$ 36	4 52	D 68	T 84	d 100	t 116			← 36	┌ 52	├ 68	84	100	┼ 116
5	0	1	0	1	ONE DOT UP 5	MODE CANCEL 21	% 37	5 53	E 69	U 85	e 101	u 117			→ 37	┐ 53	┤ 69	85	101	┼ 117
6	0	1	1	0	DELETE CHARACTER 6	ONE DOT DOWN 22	& 38	6 54	F 70	V 86	f 102	v 118			↓ 38	└ 54	├ 70	86	102	118
7	0	1	1	1	BELL 7	INSERT CHARACTER 23	' 39	7 55	G 71	W 87	g 103	w 119			↑ 39	┘ 55	┬ 71	87	103	⏚ 119
8	1	0	0	0	BS 8	✕	(40	8 56	H 72	X 88	h 104	x 120			40	┘ 56	┴ 72	88	104	120
9	1	0	0	1	TAB 9	ONE DOT LEFT 25	) 41	9 57	I 73	Y 89	i 105	y 121			41	┐ 57	═ 73	89	105	121
A	1	0	1	0	LF 10	✕	* 42	: 58	J 74	Z 90	j 106	z 122			│ 42	┘ 58	‖ 74	90	106	122
B	1	0	1	1	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123			─ 43	└ 59	I 75	91	107	123
C	1	1	0	0	ERASE PAGE 12	HOME 28	, 44	< 60	L 76	\ 92	l 108	¦ 124			│ 44	┌ 60	H 76	92	108	124
D	1	1	0	1	CR 13	CURSOR RIGHT 29	- 45	= 61	M 77	] 93	m 109	} 125			45	┐ 61	□ 77	93	109	125
E	1	1	1	0	A7 ON 14	EOF 30	46	> 62	N 78	^ 94	n 110	~ 126			─ 46	┌ 62	◀ 78	94	110	126
F	1	1	1	1	A7 OFF 15	ONE DOT RIGHT 31	/ 47	? 63	O 79	- 95	o 111	▮ 127			─ 47	└ 63	▶ 79	95	┼ 111	✓ 127

NOTE: ENTRIES WITH ✕ INDICATE UNUSED ANSI ASCII CODES.

SAME AS 0 THRU 31

A7 ON IMPLIES SPECIAL GRAPHIC CHARACTERS.

Appendix C. ERROR CODES

This appendix lists the error codes which may occur when using the disk operating system or the assembler.

C.1 Disk Error Codes

See section 2.7 for more information.

ERROR # 10	-	CRC ERROR CODE
ERROR # 11	-	SEEK ERROR CODE
ERROR # 12	-	RECORD NOT FOUND
ERROR # 13	-	LOST DATA
ERROR # 14	-	NOT READY
ERROR # 15	-	WRITE PROTECT
ERROR # 20	-	FILE NOT FOUND
ERROR # 21	-	EMPTY SLOT FOUND
ERROR # 22	-	NO HEX EOF
ERROR # 30	-	CREATE BUFFER OVERFLOW
ERROR # 31	-	COMMAND ERROR
ERROR # 32	-	NO RUN ADDRESS
ERROR # 33	-	ILLEGAL CHARACTERS
ERROR # 34	-	NO ARGUMENTS
ERROR # 40	-	ILLEGAL FILE NAME
ERROR # 41	-	BAD DISK DRIVE NUMBER
ERROR # 42	-	FATAL DISK ERROR
ERROR # 43	-	DUPLICATE FILE NAME
ERROR # 44	-	DATA ERROR
ERROR # 45	-	DISK OR DIRECTORY OVERFLOW

C.2 Assembly Error Codes

See section 4.6 for more information.

Error	Pass	Meaning
A	2	Argument error
D	1	Duplicate label
L	1	Label error
M	1,2	Missing label
O	2	Opcode error
P	2	Paging error
S	2	Syntax error
U	1,2	Undefined label
V	1,2	Value error

Appendix D. Z-80 OPCODES

The following is a list of the standard mnemonics for the Z-80 opcodes. For information on their use, see a Z-80 CPU manual.

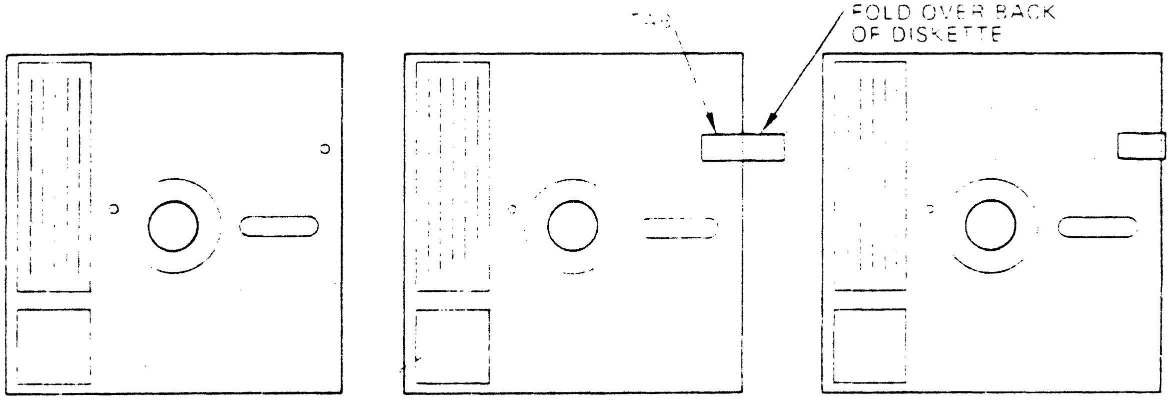
ADC	add with carry
ADD	add
AND	logical and
BIT	test bit
CALL	subroutine call
CCF	complement carry flag
CP	compare
CPD	compare, decrement
CPDR	compare, decrement, repeat
CPI	compare, increment
CPIR	compare, increment, repeat
CPL	compare logical
DAA	decimal adjust accumulator
DEC	decrement
DI	disable interrupt
DJNZ	decrement B, jump if non zero
EI	enable interrupt
EX	exchange
EXX	exchange gpu
HALT	halt
IM	set internal mode
IN	input
INC	increment
IND	input, decrement
INDR	input, decrement, repeat
INI	input, increment
INIR	input, increment, repeat

JP	jump
JR	jump relative
LD	load
LDD	load, decrement
LDDR	load, decrement, repeat
LDI	load, increment
LDIR	load, increment, repeat
NEG	negate accumulator
NOP	no operation
OR	logical or
OTDR	output, decrement, repeat
OTIR	output, increment, repeat
OUT	output
OUTD	output, decrement
OUTI	output, increment
POP	pop stack
PUSH	push stack
RES	reset bit
RET	return
RETI	return from interrupt
RETN	return form nonmaskable interrupt
RL	rotate left
RLA	rotate left accumulator
RLC	rotate left circular
RLCA	rotate left circular accumulator
RLD	rotate digit left
RR	rotate right
RRA	rotate right accumulator
RRC	rotate right circular
RRCA	rotate right circular accumulator
RRD	rotate digit right
RST	restart
SBC	subtract with carry
SCF	set carry flag

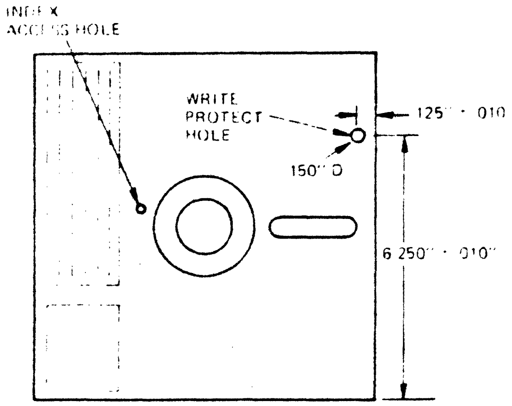
SET	set bit
SLA	shift left arithmetic
SRA	shift right arithmetic
SRL	shift right logical
SUB	subtract
XOR	exclusive or

Appendix E. DISK WRITE PROTECT

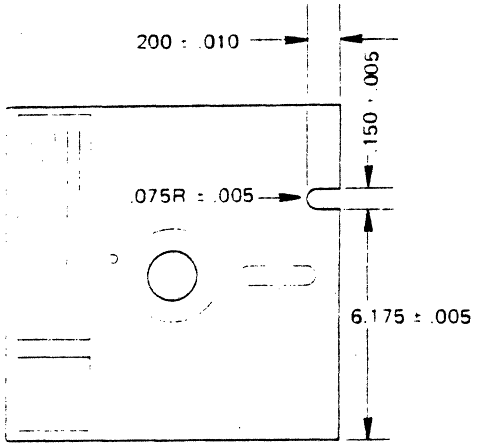
Both standard and Minfloppy^{R} disks have write protection slots which prevent writing on the disk when open. The Minifloppy^{R} disks come with a prepunched rectangular slot, but the user must punch the hole in standard disks. The figure below indicates the position that the hole must be in. In order to write on slotted disks, the hole can be covered with an opaque, peelable label.



Diskette Write Protected



Write Protect Hold Specifications



Write Inhibit Notch (Optional)

I. INDEX

Both section number and page number are given for each item in this index. When an item is referenced over several pages, only the first page number is given.

ADDENDUM TO DISK SOFTWARE REFERENCE MANUAL

The Chromatics Z-80 Disk Assembler Version 3.Ø has been enhanced over Version 2.Ø covered in the Disk Software Reference Manual. One new operator command, seven new pseudo operations (including conditional assembly), and four new arithmetic and logical operators have been added. The following sheets will describe these new functions and provide the proper syntax for implementation.

New Commands

4.2.12 Select Output Drive

DRIVE sp delim sp drive

The DRIVE command directs the assembler to select the specified drive as the current object output device. Source input drives may be specified using the DOS/drive syntax with the source filenames to be assembled by the ASSEMBLE command (see Section 2.60). By default, if the DRIVE command is not used, the object will go to the drive that contained the assembler.

New Pseudo Operations

4.4.7 Let

Label sp LET sp exp

The Label is assigned the value of the expression. Unlike an EQUATE statement, the LET statement may redefine the same label as many times as necessary. The expression may not use forward referencing, but may contain the same label as the one it is defining if it has previously been assigned a value.

Example:

```
          .
          .
          .
Q Let 6+3-5  ; Q=4
Q Let Q+1    ; Q=5
Q Let Q*Q    ; Q=25
          .
          .
          .
```

If a Let statement defines a label, the label cannot be originally defined by an EQU statement or different values of the label will be assigned on pass two by the LET statements.

4.4.8 End If

EIF

Conditional assembly is accomplished by using IF statements to select or deselect sections of assembly language code. They may be nested up to 8 levels deep. The End If pseudo operation delimits the last encountered If statement.

4.4.9 If True

IFT sp expression

If the expression evaluates to be non zero, then the instructions from the If True statement to the next End If statement will be included in the assembly, otherwise they will be ignored.

Example: - Program -

```
DOG          EQU          2
CAT          EQU          DOG-1
RAT          EQU          CAT-1
             IFT          CAT
             IFT          DOG
             LD           A,B
             EIF
             IFT          RAT
             LD           A,C
             EIF
             EIF
```

-Code Assembled -

```
             LD           A,B
```

4.4.10 If False

IFF sp expression

If the expression evaluates to be zero, then the instructions from the If False statement will be included in the assembly, otherwise, they will be ignored.

Example

- Program -

```
DOG          LET          01
             IFT          DOG
             LD           A,B
             EIF
             IFF          DOG
             LD           A,C
             EIF
DOG          LET          DOG-1
             IFT          DOG
             LD           A,D
             EIF
             IFF          DOG
             LD           A,E
             EIF
```

- Assembled Code -

```
LD          A,B
LD          A,E
```

4.4.11 List On

LON

The list switch may be toggled by the List On pseudo-op to turn on the listing from inside the assembly language program. The List/ No List operator commands have a higher priority; therefore, the listing may be suppressed by the operator regardless of the LON pseudo-op.

4.4.12 List Off

LOF

The List Off pseudo-op turns off the listing from inside the assembly language code.

4.4.13 Eject

EJT

The Eject pseudo-op performs a form feed to an external list device (eg. line printer) or executes an ERASE PAGE to the CRT when encountered during assembly.

New Arithmetic Operators

In version 2.Ø of the Chromatics Disk Assembler, expressions could only contain the arithmetic operators + and - for addition and subtraction. The following operators have been added to this set.

* multiplication (16 bit integer)

/ division (16 bit integer)

! logical OR (16 bits)

& logical AND (16 bits)

The operators evaluate from left to right only with no priority over type.

New Assembler Errors

With the addition of the Arithmetic Operator / (integer division) a new assembler error X has been added to indicate division by Ø.

Example:

```
                          .
                          .
                          .

               DOG        EQU       5
               CAT        EQU       6325
                          LD        A,DOG
                          LD        HL,CAT
               RAT        LET       DOG-5
          X               LD        DE,CAT/RAT
                          LD        BC,RAT
                          .
                          .
                          .
```

New Assembler Constant Base

The version 2.Ø Chromatics Disk Assembler allowed both decimal and hexidecimal arithmetic. The version 3.Ø assembler adds the octal base as well. Octal numbers may be indicated by placing the letter "O" or "Q" immediately after the digits. The range of octal digits are from Ø to 7; therefore, valid octal constants would be:

```
               DOG        EQU       230
               CAT        EQU       635Ø0
```

Illegal octal constants would be:

```
               DOG        EQU       930
               CAT        EQU       3AØ0
```

PROCEDURE FOR DUPLICATING SYSTEM REFERENCE DISK WITH A SINGLE DRIVE CONFIGURATION

1. Insert System Reference Disk.

2. Press " DISK OS" key.

3. Type "FORMAT/1" (Return).

4. Insert new disk (do not format System Reference Disk).

5. Type "3" (space) as answer to first question.

6. Type "F" to start formatting process.

7. If any number other than "ØØ" was printed by formatting routine, go back to #1 and start over.

8. Insert System Disk.

9. Press "CPUOS" key.

10. Type "R1,Ø,1,40,4000,".

11. CPUOS should add a "ØØ" to end of above line; if not, go back to step 9.

12. Insert newly formatted disk.

13. Type "W1,Ø,1,40,4000,".

14. CPUOS should add a "ØØ" to end of above line; if not, go back to step 13.

15. Insert System Disk

16. Type "R1,2,C,40,4000,"

17. CPUOS should add a " ØØ" to end of above line; if not, go back to step 15.

18. Insert newly formatted disk.

19. Type "W1,2,C,40,4000,"

20. CPUOS should add a "ØØ" to end of above line; if not, go back to step 18.

21. Now the newly formatted disk will be identical to System Reference Disk and may be used in it's place.