This manual was produced on a Cromemco System Three computer utilizing a Cromemco HDN-22 Hard Disk Storage System running under the Cromemco Cromix™ Operating System. The text was edited with the Cromemco Cromix Screen Editor. The edited text was formatted using the Cromemco Word Processing System Formatter II. Final camera-ready copy was printed on a Cromemco 3355A printer.
# Table of Contents

**INTRODUCTION**

<table>
<thead>
<tr>
<th>Chapter 1: BEGINNER’S GUIDE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Information About Diskettes</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Some Technical Terms Explained</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Utilities and Intrinsic Commands</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Control Characters</td>
<td>12</td>
</tr>
<tr>
<td>1.5 Safeguarding Your Data</td>
<td>12</td>
</tr>
<tr>
<td>1.6 The Reset Switch</td>
<td>13</td>
</tr>
</tbody>
</table>

**Chapter 2: SYSTEM STRUCTURE**

<table>
<thead>
<tr>
<th>2.1 Memory Allocation</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2 Disk Organization</td>
<td>17</td>
</tr>
<tr>
<td>2.2.1 Disk Specifications</td>
<td>18</td>
</tr>
<tr>
<td>2.2.2 Disk Type Specifiers</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3 Write-Protecting Diskettes</td>
<td>20</td>
</tr>
<tr>
<td>2.2.4 Precautions Concerning Diskettes</td>
<td>20</td>
</tr>
<tr>
<td>2.3 Data Files</td>
<td>21</td>
</tr>
<tr>
<td>2.3.1 Device Names</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2 Disk File References</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2.1 Single File Reference</td>
<td>23</td>
</tr>
<tr>
<td>2.3.2.2 Ambiguous File Reference</td>
<td>25</td>
</tr>
</tbody>
</table>

**Chapter 3: CDOSGEN**

<table>
<thead>
<tr>
<th>3.1 Introduction and Features</th>
<th>27</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 Generating a New CDOS</td>
<td>27</td>
</tr>
<tr>
<td>3.2.1 Memory Size</td>
<td>27</td>
</tr>
<tr>
<td>3.2.2 Disk Drive Configuration</td>
<td>28</td>
</tr>
<tr>
<td>3.2.3 Function Key Decoding</td>
<td>29</td>
</tr>
<tr>
<td>3.2.3.1 Standard Function Key Decoding</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3.2 No Function Key Decoding</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3.3 User Defined Function Key Decoding</td>
<td>30</td>
</tr>
<tr>
<td>3.2.3.4 File-Defined Function Key Decoding</td>
<td>31</td>
</tr>
<tr>
<td>3.2.4 Addresses</td>
<td>32</td>
</tr>
<tr>
<td>3.2.5 Command File</td>
<td>33</td>
</tr>
<tr>
<td>3.2.6 Boot File</td>
<td>33</td>
</tr>
<tr>
<td>Chapter 8:</td>
<td>ERROR MESSAGES</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>8.1</td>
<td>Floppy Disk Access</td>
</tr>
<tr>
<td></td>
<td>Error Messages</td>
</tr>
<tr>
<td>8.2</td>
<td>Hard Disk Error</td>
</tr>
<tr>
<td></td>
<td>Messages</td>
</tr>
<tr>
<td>8.3</td>
<td>System Error</td>
</tr>
<tr>
<td></td>
<td>Messages</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Appendix A:</td>
<td>GLOSSARY OF TERMS</td>
</tr>
<tr>
<td></td>
<td>AND SYMBOLS</td>
</tr>
<tr>
<td>Appendix B:</td>
<td>SWITCH SETTINGs</td>
</tr>
<tr>
<td>Appendix C:</td>
<td>I/O DRIVERS UNASSEMBLED SOURCE LISTINGS</td>
</tr>
<tr>
<td>Appendix D:</td>
<td>I/O DRIVERS ASSEMBLED SOURCE LISTINGS</td>
</tr>
</tbody>
</table>
CDOS is an acronym for the Cromemco Disk Operating System.

The primary use of CDOS is to control input from and output to mass storage devices such as floppy and hard disks. It is designed to allow users of Cromemco microcomputer systems to create and manipulate both random and sequential disk files using symbolic names.

CDOSGEN stands for the Cromemco Disk Operating System GENERator. It is designed to allow CDOS to be tailored to the needs of the user and hardware configuration at hand. It allows standard or custom functions to be called by the function keys of Cromemco terminals.

Most Cromemco software packages are provided with a 64K version of CDOS which may be directly booted up as shipped. CDOSGEN is also provided with most Cromemco software packages.

This manual is designed as both a reference and an instructional manual. Chapter 1 gives an overview of CDOS to the user who is new to operating systems. Chapter 2 describes the structure of CDOS, its memory allocation, disk layout, and file structure. Chapter 3 covers CDOSGEN including the various parameters necessary to use this program. CDOS operation, startup, and command structure are described in Chapter 4. Intrinsic commands and Utility programs are covered in Chapter 5. Chapter 6 is the CDOS Programmer's Manual. This section is designed for the advanced user who wants to gain a deeper understanding of CDOS and its file structure. Chapter 7 contains a list and explanation of the CDOS error messages. Finally, Chapter 8 contains a glossary of terms and symbols as they are used throughout this manual.

The Cromemco Disk Operating System (CDOS*) is an original product designed and written in Z-80 machine code by Cromemco, Inc. for its own line of microcomputers. However, due to the large number of programs currently available to run under the CP/M** operating system, CDOS was designed to be upwards CP/M compatible. This means that many programs written

* CDOS is a Trademark of Cromemco, Inc.
Mountain View, California

** CP/M is a Trademark of Digital Research, Inc.
Pacific Grove, California
for CP/M (versions up to and including 1.3) will run without modification under CDOS. This also means that programs written for CDOS will not generally run under CP/M.

Cromemco is licensed by Digital Research, the originator of CP/M, for use of the CP/M data structures and user interface.

There are several advantages to end users which result from this compatibility. First, users of Cromemco machines are able to draw on the large library of existing CP/M and CP/M compatible programs available on the market. Second, users familiar with CP/M can easily move up to CDOS taking advantage of the many additional features available with CDOS.

The enhancements contained in CDOS, but not CP/M, are primarily visible in the system calls. CDOS has added a number of new system calls to allow the user even more flexible means of device and disk I/O. CDOS includes all twenty-seven of the system calls of CP/M version 1.3.
Chapter 1
BEGINNER'S GUIDE

IMPORTANT NOTE
All commands to CDOS must be terminated by pressing the
RETURN key. If you enter a command and nothing happens,
check that you have properly terminated the command
(with a RETURN).

1.1 INFORMATION ABOUT DISKETTES

There are five significant parts of the diskette that
you need to know about.

1. The label on the plastic casing of the diskette
which can be used to describe the general contents.

2. The write protect notch on the plastic casing that
enables or disables the ability to write to the
diskette.

3. The oblong window in the plastic casing through
which the disk drive reads from and writes to the
inside circular diskette.

4. The circular window in the middle of the diskette.
The disk drive clamps onto the inner portion of the
circular diskette here and spins it.

5. The index holes which indicate to the operating
system if the diskette is single or double sided.

There are several precautions that you need to take with
diskettes.

1. Whenever a diskette is not in the computer, make
sure that it is in its protective envelope.

2. Never bend a diskette.

3. Never touch the surface of the inner disk of the
diskette.

4. Never place a diskette near a source of magnetism.

5. Diskettes cannot tolerate temperature or humidity
extremes. As a general rule, if you are hot or cold, the diskette is too.

Diskettes are inserted into a drive with the edge nearest the oblong window going in first and with the label on the left. If the drive slot on your computer is horizontal, the label will face up.

If you have a System Three, the drives can be identified by the letters on the white eject buttons beneath each drive slot.

On a System Two or a 22-H, the drives can be identified by the painted letter below each drive.

1.2 SOME TECHNICAL TERMS EXPLAINED

The cursor is the small white rectangle on the screen of your terminal. It indicates the position where text will appear when you type on the keyboard.

An operating system is a program which gets information, whether in the form of text or other programs, from your disks, sends printing to your printer, creates places on disk to store information, and also manages that space. This operating system is called CDOS, which stands for the Cromemco Disk Operating System.

A CDOS prompt is an indication to the user that the operating system is ready to receive an instruction. The prompt will be in the form of a capital letter followed by a period, e.g., A., D., H., etc. The instruction given in response to the prompt can be an intrinsic operating system function, a program, or one of certain control functions.

The current drive is the drive that you are working from. The letter of the CDOS prompt will specify which is the current drive.

A file is a collection of related data. A file can be a program, a letter to your mother, an inventory list, or any other group of data that is stored on disk.

Filename is the term for the name of a file with the format that CDOS will accept. There are two parts of a filename that uniquely identify it on a disk. The fundamental name of the file can be up to eight characters long. After this name can be a three letter extension which is generally used to classify what type of file it is. This extension is connected to the name
with a period, e.g., cdos.com, payables.bas, primes.280.

A disk specifier, when used by itself, can change the current drive. When it prefaces a filename, it further identifies that file. The disk specifier is composed of a drive letter followed by a colon. When you log on, A. is displayed as the CDOS prompt. That means that the drive that you are working on is drive A. If you want to work on drive B, type B: and the CDOS prompt B. will be displayed on the screen. The current drive is now drive B. It is also useful in accessing a file on another disk drive. If you are doing something on drive A and need to refer to the file recvabsl.led on drive B, you can specify the file on drive B as

b:\recvabsl.led.

Memory refers to the random access memory in your computer, probably a 64K board. It is in the "work area" of your computer.

Storage refers to the devices which house your programs and data when not in use. These are usually diskettes or hard disks.

RETURN refers to the RETURN key of the terminal.

1.3 UTILITIES AND INTRINSIC COMMANDS

A utility is a program that is related to the operating system and which performs a useful function, but is not a part of the operating system. Utilities are separate programs found in the disk directory, and must be on either the current disk or the master disk (a:) to be executed. DUMP, STATUS, and XFER are examples of utility programs. When entering a utility program name, do not type the extension ".com".

An intrinsic command (hereafter referred to as an intrinsic) is a command that is part of the operating system and may be executed wherever the CDOS prompt is displayed. Examples of inrinsics are ATTR, DIR, ERA, and TYPE.

When entering a utility program name or an intrinsic, enter only the portion in capital letters. For instance, if you want to use the STATUS utility, type only STAT.
Directory

DIR is the intrinsic that allows you to see what files are on a disk. It is like a table of contents for the disk. DIR is short for directory.

There are several different ways that dir can be used. It can be used by itself, dir, to display the filenames and file space used on the current disk. It can be followed by a disk specifier to display the filenames and file space used on a disk in another drive:

dir b:

You can use it with a single filename to verify the existence or size of that file:

dir c:\photom.z80

Type

TYPE is used to quickly look at files that are composed of alphabetic, numeric, and punctuation characters.

The contents of a file can be displayed by typing type followed by a text filename:

    type thesis.txt

TYPE should only be used with text files. Attempting to TYPE nontext files will produce unpredictable results.

Erase

ERA, short for erase, enables you to erase files from the disk. It is also an intrinsic command.

A file can be erased from a disk by typing era followed by its filename:

    era chromatg.rel

Disk specifiers can be used with the filename to erase a file which is on a disk in a different drive:
era b:chromatg.rel

Attribute

**ATTR** is used to change the security attributes of a file. With this intrinsic, files can be protected from read, write, or erase operations. ATTR is short for attributes.

There are three different types of protection available for files. They are **R**, which prevents the file from being erased; **W**, which prevents the file from being read; and **E**, which prevents the file from being written to.

A file can be assigned attributes by typing `attr` followed by the name of the file, and the letter(s) corresponding to the desired protections. The file called `letter.mom` can be erase and write protected by typing:

```
attr letter.mom ew
```

Attributes can be removed by typing `attr`, followed by the filename, followed by no attributes.

Rename

**REN** is the intrinsic that enables you to change the name of a file.

You can change the name of a file by typing `ren`, which is short for rename, followed by the new filename, an equal sign (=), and then the current filename:

```
ren newname.txt=oldname.txt
```

Renaming a file does not change the data in the file or move the file on the disk. It only changes the name of the file.
Initialize

INIT prepares a disk so that information can be stored on it. This process destroys any data that is already on the disk.

This program should only be run when 1) the disk is new, 2) the disk is unreadable, i.e., the data and formatting of the disk have been magnetically or electrically destroyed, or 3) if you want to store data in double density or single sided format.

All 8" diskettes supplied by Cromemco have already been initialized as double sided disks and must be reinitialized if they are to be used as single sided diskettes.

To initialize a diskette first type init and you will be asked several questions concerning the diskette. The characters that appear between the brackets are the default values that can be entered by just pressing the RETURN key. After a diskette has been initialized, STAT/L should be run to label the diskette. The diskette is now ready for use.

Transfer

XFER enables you to copy files to other disks, to the printer, and to your terminal.

A file can be copied to another disk by typing xfer followed by the disk specifier of the destination disk, an equal sign (=), and the name of the file:

```plaintext
xfer b:=a:source.txt
```

There are four significant options. They are:

- `/v` Verify the copy.
- `/a` Delete the end of file marker (text files only).
- `/t` Expand tabs in source file into spaces in destination file.
- `/c` Compare two files without transfer.
If you want to use one or more of the options, put them immediately after `xfer` with no intervening spaces:

```
 xfer/v a:=b:fibonacci.z80
```

copies the file `fibonacci.z80` from drive B to drive A and verifies the copy.

```
 xfer/t prt:=phi.txt
```

copies the file `phi.txt`, expanding tabs, from the current drive to the printer.

The `/t` option should be used when copying a file which contains tabs. If it is not used, tabs will not be displayed on devices incapable of expanding them, such as most printers.

The `/v` option verifies that the file has been copied correctly.

The `/a` option is very useful for removing the end of file markers when concatenating files:

```
 xfer/a book.txt=chapter1.txt,chapter2.txt,appendix.txt
```

In this example, each successive file is appended to the end of the previous one. This example uses a filename as a destination instead of a disk specifier. Also notice that since no disk specifiers were used all files are on the current drive. Disk specifiers can be used for any of the filenames if they are applicable. The `/a` option in this example deletes the end of file marker from `chapter1.txt` and `chapter2.txt` and leaves the end of file marker from the last file, `appendix.txt`.

The `/c` option is used to compare two files. If you suspect that you have two duplicate files when only one is desired, you can resolve your suspicions with the `/c` option:

```
 xfer/c file1.lis=file2.lis
```

No copying is done with this option.
Status

STAT allows you to check and modify various aspects of your system. Following are several of the available options.

/a Displays an alphabetical directory of the files on a disk along with how much space each one takes.
/b Displays a brief description of the space available on a disk.
/d Sets the current date.
/e Allows you to selectively erase files on a disk. These are displayed in alphabetical order.
/l Labels a disk with name, date, and description of the disk.
/t Sets the time of day.

This program is called by typing STAT immediately followed by the desired option and pressing the RETURN key. You can execute several of STAT's options at one time. The time and date can be set by typing STAT/dt. STAT with no options displays a comprehensive status description of the current disk and memory.

Batch

@, called Batch, enables you to type a group of commands and have them execute sequentially.

Batch jobs can be run two different ways. If the sequence of commands to be executed is not one that is to be run frequently, type @. After a few seconds, an exclamation point will appear on the next line. Here, you will enter the first in the sequence of commands. Press the RETURN key and the cursor will move to the beginning of the next line and you can enter the second command. This procedure is repeated for each successive command. When you have entered the entire sequence of commands and are on the beginning of a new line following the last command, press RETURN once more. The commands will begin executing in the order in which you entered them.

If there is a sequence of commands that you want to run frequently, you can create a file containing these
commands with one of the Cromemco text editors. This file must contain one command per line. The name of this file must have the extension .cmd.

`compile.cmd`

Enter `@ filename` to execute your BATCH file:

`@ compile`

1.4 CONTROL CHARACTERS

Control characters perform console and printer functions. Some useful control characters are:

- **CNTRL-S** Stops printing to the console or the printer. Pressing any key will restart the printing.
- **CNTRL-V** Deletes the current line on the console.
- **CNTRL-P** Sends printing that normally goes to the `console only` to the printer as well. Pressing CNTRL-P again will resume printing to the console only.

Control characters are used by holding down the CNTRL key and pressing another key. CNTRL-V is entered by holding down the CNTRL key and pressing the V key. Users having Cromemco 3102 terminals may use the CE function key (clear entry) for CNTRL-V, the PRINT function key for CNTRL-P, and the PAUSE function key for CNTRL-S. The PAUSE key is located between the EOL and PRINT keys and may not be marked.

1.5 SAFEGUARDING YOUR DATA

It is a wise investment of time and effort to make frequent copies of your work. It is recommended that you make backups at least twice per day, e.g., before lunch and before going home.

Backups are made in different ways depending upon what you are doing. If you are working with the Screen Editor, exiting and updating your file will create a
backup. If you are in BASIC, listing or saving your program will create a backup. You should also make a backup copy of your disk using the xfer utility. This should be done daily, or more often depending on the nature of your work.

1.6 THE RESET SWITCH

The reset switch is used to put your computer in a state such that CDOS can be rebooted. The reset switch is used when you don't like what your computer is doing, i.e., looping forever in a program. Pressing or turning the reset switch will enable you to escape from your program, boot CDOS, and reenter your program to make the necessary changes.

The reset switch on Cromemco computers is found on the back of the computer. On System Three computers, the key switch on the front is also a reset switch. If you do not have a System Three, there is a jack on the back of your computer that will accommodate a remote reset switch.

Pressing reset while the disk is being written to will result in a file that cannot be read.
2.1 MEMORY ALLOCATION

Under CDOS, memory is divided into two major parts.

The first part is that area of RAM which is reserved for CDOS itself. CDOS occupies memory from locations 0 through 100H (Low Memory) as well as approximately the top 11K to 18K of RAM.

The second part is the User Area of RAM. The user area occupies memory from 100H up to the bottom of CDOS. The size of the user area is determined when CDOSGEN is run and is limited by the amount of memory in the system. It is usually about 48K.

![Memory Use Map Diagram](image-url)
The system is described by the total number of bytes it occupies. Most Cromemco software packages are supplied with a CDOS configured for a 64K system.

CDOS is loaded from the System Area of the disk into memory by a bootstrap routine.

By special use of low memory, all user programs call CDOS through a standard sequence which is transparent to the size of CDOS.

Referring to the CDOS Memory Use Map, we see that RAM is divided into the following areas:

**High Memory**

CDOS contains the basic input/output functions for the console, printer, punch, and reader as well as the disk I/O drivers.

CDOS contains the file management functions which are responsible for managing, creating, opening, reading, and writing disk files. It also is in charge of calling user programs and editing console input.

CDOS also has some internal functions called intrinsic commands.

**User Area**

This is where programs actually run. The User Area begins at 100H (256 decimal) and extends to the bottom of CDOS. All programs which are not intrinsic to CDOS are run in this area. Intrinsic programs do not run in this area and therefore do not alter it.

The external functions are the utility and user Command files which are located on the disk. These files can be identified by the COM filename extension. They are executed by typing the filename without the filename extension (COM is assumed) in response to the CDOS prompt.

**Low Memory**

Memory below the User Area is reserved by CDOS for the following special purposes:
2. System Structure

0-2H  System warm start vector
3H   I/O byte
5-7H  System call vector for user requests
8H   Specifies running under CDOS if FFH
     and under CROMIX Operating System if C3H
30-32H Breakpoints for DEBUG
38-3AH Jump to invalid jump message
40-55H Reserved for system
5C-78H Standard user file control blocks
80-FFH Standard user I/O buffer (disk & command line)

The reader is referred to the CDOS Programmer's Guide for a more detailed discussion on the use of Low Memory.

2.2 DISK ORGANIZATION

Each disk used under CDOS is divided into two general areas. The first area is the System Area. It may be accessed by the user only through the WRTSYS utility program or when creating a boot file with CDOSGEN. The contents of this area are not listed by the DIRECtory intrinsic command. The System Area occupies the outer tracks of the disk.

The second area is the File Area. This is the section where user files (e.g., programs, data, etc.) and the disk directory are stored.

<table>
<thead>
<tr>
<th>Disk</th>
<th>Tracks in System Area</th>
<th>Approximate File Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>5&quot;SS SD</td>
<td>3</td>
<td>61K</td>
</tr>
<tr>
<td>5&quot;DS SD</td>
<td>3</td>
<td>171K</td>
</tr>
<tr>
<td>5&quot;SS DD</td>
<td>2</td>
<td>188K</td>
</tr>
<tr>
<td>5&quot;DS DD</td>
<td>2</td>
<td>386K</td>
</tr>
<tr>
<td>8&quot;SS SD</td>
<td>2</td>
<td>241K</td>
</tr>
<tr>
<td>8&quot;DS SD</td>
<td>2</td>
<td>490K</td>
</tr>
<tr>
<td>8&quot;SS DD</td>
<td>2</td>
<td>596K</td>
</tr>
<tr>
<td>8&quot;DS DD</td>
<td>2</td>
<td>1,208K</td>
</tr>
<tr>
<td>Hard-ll</td>
<td>1</td>
<td>10,490K</td>
</tr>
</tbody>
</table>

(SS=Single Sided; DS=Double Sided; SD=Single Density; DD=Double Density)

The use of the two areas previously described is not related. Even if the DIRECtory command indicates a full disk, a copy of the CDOS boot file may still be written to the System Area using WRTSYS or CDOSGEN. The
DIRECTORY intrinsic indicates only the user file portion of the File Area which is occupied on the disk. This has no bearing on the System Area.

2.2.1 Disk Specifications

This table shows the number of tracks per disk surface, surfaces per track, and the sector size for CDOS disks. Numbers not within parentheses are decimal. Numbers within parentheses are hexadecimal.

<table>
<thead>
<tr>
<th>Disk</th>
<th>Cylinders</th>
<th>Surfaces</th>
<th>Sectors/Track</th>
<th>Sector Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>8&quot;SD</td>
<td>77(0-4CH)</td>
<td>2</td>
<td>26(1-1AH)</td>
<td>128 bytes</td>
</tr>
<tr>
<td>8&quot;DD</td>
<td>77(0-4CH)</td>
<td>2</td>
<td>16(1-10H)</td>
<td>512 bytes</td>
</tr>
<tr>
<td>5&quot;SD</td>
<td>40(0-27H)</td>
<td>2</td>
<td>18(1-12H)</td>
<td>128 bytes</td>
</tr>
<tr>
<td>5&quot;DD</td>
<td>40(0-27H)</td>
<td>2</td>
<td>10(1-0AH)</td>
<td>512 bytes</td>
</tr>
<tr>
<td>HARD</td>
<td>350(0-15DH)</td>
<td>3</td>
<td>20(0-14H)</td>
<td>512 bytes</td>
</tr>
</tbody>
</table>

Note:
The first track (cylinder 0, side 0) of all floppy diskettes is initialized as single density with 128-byte sectors by the INIT program to allow the disk to be booted with 16PDC and 4PDC versions of RDOS.

On hard disks, there are four additional cylinders which are reserved as alternates to be used if other tracks develop hard errors.

2.2.2 Disk Type Specifiers

CDOS determines what type of disk is being used from a special disk type specifier stored in the first sector of the disk (sector 1, cylinder 0, side 0 of floppy disks and sector 0, cylinder 0, surface 0 of hard disks). The disk type specifier consists of bytes 121 through 128 of this sector. The specifier is composed of four groups of two bytes each which contain the ASCII values of the characters listed in the following table.
Bytes | Characters | Meaning
------|------------|-------------
121 - 122 | LG | CDOS large floppy
   | SM | CDOS small floppy
   | HD | CDOS hard disk
123 - 124 | SS | single sided floppy
   | DS | double sided floppy
   | 11 | 11-Mbyte hard disk
125 - 126 | SD | single density
   | DD | double density
127 - 128 | reserved for future use

The System Area of the disk includes all or part of the first 1, 2, or 3 tracks of the disk, depending on the disk type. The space reserved the System Area is always at least 6.5K. On double density floppy disks, part of the system area may be stored on a single density track (cylinder 0, side 0) and part on a double density track (cylinder 0, track 1).

The File Area starts at the beginning of the track following the system area. (CDOS accesses disks by alternating sides or surfaces as it works its way into the disk by increasing cylinder numbers, so the next track may be a different surface of the same cylinder.) The directory always begins at the beginning of the file area (i.e., the first 1K of directory space is always on the first track of the file area), but other parts of the directory may be elsewhere on the disk. This information is summarized for each of the various types of CDOS disks in the following table.

<table>
<thead>
<tr>
<th>Disk Type</th>
<th>System Area</th>
<th>Start of File Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>LG SS SD</td>
<td>c0,0; c1,0; c1,0</td>
<td>c2,0</td>
</tr>
<tr>
<td>LG SS DD</td>
<td>c0,0; c1,0</td>
<td>c2,0</td>
</tr>
<tr>
<td>LG DD SD</td>
<td>c0,0; c0,1</td>
<td>c1,0</td>
</tr>
<tr>
<td>LG DD DD</td>
<td>c0,0; c0,1</td>
<td>c1,0</td>
</tr>
<tr>
<td>SM SS SD</td>
<td>c0,0; c1,0; c2,0</td>
<td>c3,0</td>
</tr>
<tr>
<td>SM SS DD</td>
<td>c0,0; c1,0</td>
<td>c2,0</td>
</tr>
<tr>
<td>SM DD SD</td>
<td>c0,0; c0,1, c1,0</td>
<td>c1,1</td>
</tr>
<tr>
<td>SM DD DD</td>
<td>c0,0; c0,1</td>
<td>c1,0</td>
</tr>
<tr>
<td>HD 11</td>
<td>c0,0</td>
<td>c0,1</td>
</tr>
</tbody>
</table>
2.2.3 Write-Protecting Diskettes

8" Diskettes

The 8" (large) diskettes are write-protected by a notch on the bottom right side (as the label faces you) of the plastic disk cover. To be able to write on the disk, cover the notch with a silver sticker or a piece of masking tape.

5.25" Diskettes

The 5.25" (small) diskettes are write-protected by the presence of the silver write-protect sticker covering the notch. Remove this sticker if you want to write on the disk.

Important Distinction

It is important to note that large disks are write-protected by removing the silver sticker, and small disks are write-protected by placing the silver sticker over the notch.

Files may be write-protected as well as, or instead of, diskettes. This can be done with the ATTR intrinsic. ATTR is a software write-protect only.

2.2.4 Precautions Concerning Diskettes

The following precautions are suggested. They are designed to minimize the chance of damage to files stored on floppy diskettes.

1. While in a program, do not exchange diskettes unless the program provides for it. Terminating execution of the program with CTRL-C will not close files. Diskettes may be exchanged while in BASIC if the DISK-8 command is used.

2. Execute the STATUs Utility program occasionally in order to verify the directory.

3. Diskettes are magnetic media. The following care and attention should be given to them:
   a. Keep them away from all sources of magnetic fields such as power transformers and
solenoids.

b. Store a diskette in its dust covers and **never** lay the bare disk down on a dusty surface.

c. Keep them out of direct sunlight as the black plastic heats up rapidly. Normal storage temperature is 50 to 125 degrees Fahrenheit (10 to 52 degrees Celsius).

d. Do not write on the plastic disk jacket with anything but a soft felt tip pen.

e. Do not touch or try to clean the disk surface. Abrasions may cause loss of data.

f. Never bend, fold, or staple the disk.

g. It is suggested that the disk **not be loaded** (i.e., inserted in the drive with the door closed) while powering up or down. Under these conditions random data may be written to the disk. In case of power failure it is wise to check the disk for errors following the return of power.

4. As an additional safety precaution, maintain adequate archives of backup disks. Data may occasionally be lost and the additional cost of backup disks is well worth the valuable programs, data, and time which may be saved.

2.3 DATA FILES

Data is information. Some examples of data are: a list of names and addresses, a FORTRAN program, the text of a letter or a manual, etc.

A file is a group of related individual items of information. Some examples of files are: a telephone or address book, a filing cabinet, the paper on which a grocery list is written, etc.

A computer data file (or simply file) is accessed by describing:

1. the storage medium (floppy disk, hard disk, paper tape, etc.),

2. the method of accessing the data (sequential or random), and
3. the code by which the data is translated for storage (ASCII or internal machine representation).

When a file is created, it is given an identifier so that it may be referenced at a later time. This identifier is the filename and optionally the filename extension.

Files may be stored in the same format as data is stored inside the computer. This is referred to as an Internal Machine Representation. Files also may be coded, or formatted, according to the American Standard Code for Information Interchange which is usually called ASCII. An ASCII file contains only characters from the ASCII table. On output, each of these numbers is translated into the character it represents. An ASCII file may be TYPED while a file stored in internal machine representation must be PUMPED.

Files may be read from or written to a number of devices. The standard devices available under CDOS are:

<table>
<thead>
<tr>
<th>Device</th>
<th>Data Transfer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Console</td>
<td>Input &amp; Output</td>
</tr>
<tr>
<td>Printer</td>
<td>Output</td>
</tr>
<tr>
<td>Disk Drive</td>
<td>Input &amp; Output</td>
</tr>
<tr>
<td>Paper Tape Reader</td>
<td>Input</td>
</tr>
<tr>
<td>Paper Tape Punch</td>
<td>Output</td>
</tr>
</tbody>
</table>

As normally delivered, only the console, printer, and disk are active. The paper tape reader and punch drivers are implemented using the same port assignments as the console. These may be changed by modifying the I/O device drivers.

The primary use of CDOS is to perform I/O with the disk. Any combination of up to four floppy disk drives and up to seven hard disk drives for a total of eight drives may be connected to a Cromemco floppy disk controller and WD1 hard disk controller. Unlike some large computer systems, all disk files under CDOS may be accessed in either random or sequential order.

Devices are predefined by CDOS, but disk files are dynamically created, extended, or deleted as required.
2.3.1 Device Names

The following symbolic names may be used when referring to devices accessible by CDOS.

Format: xxx:#

where:

xxx represents a three character name and # is an optional number from the following table:

<table>
<thead>
<tr>
<th>Device</th>
<th>Name</th>
<th>Number Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Console</td>
<td>CON</td>
<td>0...7</td>
</tr>
<tr>
<td>Card Reader</td>
<td>RDR</td>
<td>0...3</td>
</tr>
<tr>
<td>Paper tape Punch</td>
<td>PUN</td>
<td>0,1</td>
</tr>
<tr>
<td>Line Printer</td>
<td>PRT</td>
<td>0...3</td>
</tr>
<tr>
<td>Dummy Device</td>
<td>DUM</td>
<td>--- (bit bucket/EOF)</td>
</tr>
</tbody>
</table>

2.3.2 Disk File References

The term

file-ref or file reference

is used throughout this manual to describe:

1. a single file reference including a file name and optionally a disk drive specifier and filename extension,

   or

2. an ambiguous file reference if it is specifically stated that the file-ref may include the * and ? replacement characters.

2.3.2.1 Single File Reference

A Single File Reference is a unique reference to a unique file stored on a disk and accessible by CDOS. By default or by specification this type of reference addresses a particular file (filename plus an optional
filename extension) on a particular disk drive.

Format:  [X:]filename[.ext]

where:

X is an optional disk drive specifier indicating the location of the file being referenced. Appropriate values are the letters A through H.

filename is a filename composed of up to eight printable ASCII characters except as specified in Note 1 below.

ext is an optional 1 to 3 character extension to the filename. See Notes 1 and 3.

Notes:

1. A filename or extension may include any printable ASCII character except the following:

$ * ? = / . , : space

2. Although lower case characters are accepted without modification by most programs, all system functions convert lower case input of filenames to upper case.

3. There are several standard types of filename extensions expected by Cromemco system programs. These are listed below:
BAK  Editor backup file
BAS  BASIC LISTed source file (optional)
CMD  Batch command file
COB  COBOL source file
COM  Executable command program
FOR  FORTRAN source file
HEX  Hex format object file (8080 file)
LIS  BASIC LISTed source file (optional)
PRN  Printer or listing file
REL  Relocatable module (object file)
SAV  BASIC SAVED source file (optional)
SYS  System image file
TXT  Text Formaster input file (optional)
Z80  Assembler source file

4. When an executable COMmand file is referred to without the optional disk drive specifier, the system will search the current drive for the file. If this search fails, and the current drive is not the master drive, the master drive is then searched for the file. The default master drive is drive A. This procedure is followed only for COM files.

Examples:

A:PROGRAM1.FOR refers to a FORTRAN source file on the disk in drive A named PROGRAM1 with a filename extension of FOR.

C:BASIC.COM refers to an executable COMMAND file on the disk in drive C. The filename is BASIC and the filename extension is COM.

PROG.REL refers to a relocatable object file on the disk in the current drive named PROG with a filename extension of REL.

2.3.2.2 Ambiguous File Reference Using Replacement Characters

The asterisk (*), question mark (?), and characters within brackets ([]) may be used as replacement characters in a filename or filename extension to create an ambiguous file reference. The format of the ambiguous file reference is the same as that of the single file reference.
The asterisk replaces any character(s) from the position
it occupies, to the right, up to the next delimiter
(i.e., period (.), question mark (?), or carriage
RETURN).

PROG.* will match PROGRAM.FOR
PROGTEST.8-88
PROG.BAS
PROG123.REL

The question mark replaces any single character in the
exact position it occupies.

?OOK.TXT will match GOOK.TXT
OOK.TXT
LOOK.TXT
NOOK.TXT

Brackets may be used to indicate that several single
characters are to be substituted for that single
character position. Brackets may be used only in the
utility programs Xfer and Stat.

TEST[XYA-D].REL will match TESTX.REL
TESTY.REL
TESTA.REL
TESTB.REL
TESTC.REL
TESTD.REL

Notes:

1. These replacement characters in no way alter the
original file reference. They do not become part
of the filename or filename extension. The
asterisk and question mark serve only to refer to
several files at once by creating an ambiguous file
reference.

2. These replacement characters may be used only in
commands and programs as specified in this manual.
3.1 **INTRODUCTION AND FEATURES**

CDOSGEN is a very powerful feature of the Cromemco Disk Operating System. It allows CDOS to be built around the user's particular hardware configuration and software needs. As needs and equipment change, CDOS can be reconfigured in a matter of minutes to conform to a new hardware environment.

The ability to program twenty individual console function keys gives CDOS, and all programs run under CDOS, a new flexibility. These programmable keys can be used to facilitate user interaction with programs, any of the many languages offered by Cromemco, and CDOS itself.

CDOS supports up to 64 kilobytes of memory. CDOSGEN will design an operating system around any combination of up to eight disk drives. CDOS can support up to four floppy disk drives and up to seven hard disk drives with drive A being a floppy disk drive.

3.2 **GENERATING A NEW CDOS**

CDOSGEN is executed by responding to the CDOS prompt by typing CDOSGEN. The file CDOSGEN.COM must be located on the current drive or the master drive if a disk drive specifier is not used.

The program will prompt the user with questions concerning the desired system.

3.2.1 **Memory Size**

After the header, the first prompt CDOSGEN will display is:

```
Memory Size (3FFF through FFFF or 16K through 64) [n] ?
```

where n is the actual amount of memory available. There are three ways in which the user can respond to this.
3.2.2 Disk Drive Configuration

The following table shows the drive configurations which CDOS will allow.

<table>
<thead>
<tr>
<th>Drive</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>floppy</td>
</tr>
<tr>
<td>B-D</td>
<td>floppy or hard</td>
</tr>
<tr>
<td>E-H</td>
<td>hard</td>
</tr>
</tbody>
</table>

After establishing the system size, CDOSGEN will begin querying the user about the disk drive configuration with the prompt:

```
Drive A Type (S=Small, L=Large) ?
```

Enter S if drive A is a 5 inch floppy drive or L for an 8 inch floppy drive. If the drive is a 5 inch drive, you will be asked:

```
Fast or slow seek [S] ?
```

Enter S or a RETURN if the 5 inch drive is the older style having a full width front door; otherwise, enter F. For both 5 and 8 inch drives you will be asked:

```
Single or Double Sided [S] ?
```

If the drive is double sided, then type D and press
RETURN. If the drive is single sided, press RETURN or type S and press RETURN.

Single or Dual Density [S] ?

If the drive is dual density, capable of handling either single density or double density disks, type D and press RETURN. If the drive is single density, press RETURN or type S and press RETURN.

If drive A is designated as a large drive, CDOSGEN will make the assumption that drive B is also a large drive since Cromemco 8 inch floppy disk drives are always adjacent pairs. If drive A is a 5 inch drive and drive B is a large drive, CDOSGEN will assume that drive C is also a large drive.

The next prompt will be:

Drive X Type (S=Small, L=Large, H=Hard, N=None, E=End) ?

where X is a letter from A to H.

If you do not have a drive X and there are no more drives in your system, enter E for "end of drive specification." If you do not have a drive X and there are more drives in your system, enter N for "no drive assigned to this letter." If drive X is a hard disk, enter H.

3.2.3 Function Key Decoding

The user is then asked to specify the type of function key decoding desired:

Function Key Decoding
(S=Standard, N=None, U=User, F=File) [S] ?

These options are covered in the next sections.

The function key decoding options are supported by Cromemco 3102 and 3101 terminals. Users who have not incorporated either of these terminals into their system should respond to this prompt with an N.
3.2.3.1 Standard Function Key Decoding

Responding to the function key decoding prompt with an
S will cause each of the function keys to issue a
predefined standard command. These standard commands
are:

F1 A: <RETURN>       F11 SCREEN <space>
F2 B: <RETURN>       F12 XFER/V <space>
F3 C: <RETURN>       F13 DEBUG <RETURN>
F4 D: <RETURN>       F14 C <RETURN>
F5 E: <RETURN>       F15 L$ <RETURN>
F6 F: <RETURN>       F16 G/r$0 <RETURN>
F7 STAT/<space>      F17 STAT/DT <RETURN>
F8 *<space>          F18 BASIC <RETURN>
F9 STAT <RETURN>     F19 XFER/C <space>
F10 STAT/B <RETURN>  F20 XFER/AT PRT: = <space>

All function keys, except F13 to F16, are designed to be
used in response to the CDOS prompt. The commands which
are terminated with a carriage RETURN (<RETURN>) are
stand-alone functions and will cause CDOS to respond.
Those terminated with a <space> will wait for the user
to input a file reference followed by a carriage RETURN.
Functions 13 through 16 are designed to be used with the
Debug program.

3.2.3.2 No Function Key Decoding

Responding to the function key decoding prompt with an
N will disable the function keys. This will also free
some additional space in CDOS for drivers and allow CDOS
to occupy less memory after booting.

3.2.3.3 User Defined Function Key Decoding

Responding to the function key decoding prompt with a U
will cause CDOSGEN to prompt the user for the desired
decoding of each function key. In response to each
prompt (F1:, F2:, etc.) the user may enter any series
of characters not including the ESCape character. In
most applications, CTRL-Z may be substituted for the
ESCAPE character. The ESCape character terminates the
current function key definition.

Any command, response, or instruction may be entered as
a function. Then, when the function key is depressed,
it will repeat the characters which were entered during the definition of the function. Functions keys may be defined for use while in CDOS, the Screen Editor, or any program using CDOS System Calls for console I/O.

Function sequences may contain or be terminated with a carriage RETURN character which, in CDOS, will cause execution of the command. Function sequences may also be terminated with a blank, allowing the user to supply additional information as well as a terminating carriage RETURN.

Function keys may be programmed with a command line which includes carriage RETURNS. Thus F1 may be programmed with the sequence:

```
DIR A:<RETURN>
DIR B:<RETURN>
<ESC>
```

When the F1 key is then depressed, the directory of the disk in drive A will be listed followed by the directory of the disk in drive B.

3.2.3.4 File-Defined Function Key Decoding

The file referred to in response to this query must be an assembled file which defines each of 20 functions. Each function definition contains the ASCII equivalent of the (command) line to be displayed when the function key is depressed and must be terminated by a -1 (FFM). There must be 20 terminators in the file.

Example:

The following file was assembled with the Cromemco Macro Assembler, linked with the Cromemco Linker (link/p:100, filename, filename/n/e), which saves the file on the disk as a COM file to give the standard CDOS function key decoding:

31
3.2.4 Addresses

Several important addresses will be displayed.

Starting address of CDOS - This is the bottom of CDOS. The bottom of CDOS will always fall on an even 256 (1024) byte or page boundary.

Starting address of I/O drivers - This is the first location of the CDOS I/O drivers.

Last address of CDOS - This is the highest address used by CDOS. Memory between this address and the highest address in the system may be allocated by the user for a particular configuration of CDOS. This is not generally recommended.

Top of memory - This is the amount of memory that the user specified was in the system.
Size of CDOS - This is the Last address minus the Starting address.

Size of the Boot Loader - This is the size of the system area used.

3.2.5 Command File

You will be prompted for the command filename:

Enter command filename [n:CDOS] -

where n is the current drive. There are two options here. Either a RETURN can be entered, so that CDOS.COM will be generated on the current drive, or another filename may be entered. The filename can have a different drive specifier only such as B:CDOS or a completely different name such as C:\HARDOS. The extension COM will be automatically appended to the filename entered. Note that only the name CDOS.COM will boot the system from RDOS. However, a name such as HARDOS may be used to boot one CDOS from another.

3.2.6 Boot File

You will be prompted as to whether the boot file should be written to the disk:

Write system boot to drive n: (Y = Yes, N = No) [Y] ?

where drive n is the same as that of the COM file.

If Y is entered in response to the prompt for a boot file, the file will be written to the System Area of the same disk specified in the previous question and will not appear in the directory.

In order to bring up the system which was just created, the disk upon which the system was written must be placed in the A drive and then booted up. The user will not be running under the new CDOS until it is brought into memory and this is not done until CDOS is reloaded (booted up).
4.1 SYSTEM STARTUP

4.1.1 Loading CDOS

With all the circuit boards installed, the terminal connected, and the switches set as described in the appendix, the following procedure will load CDOS:

1. Turn on the power to the computer, terminal, and disk if an external disk storage device is used.
2. Place the CDOS system diskette in disk drive A.
3. Press the carriage RETURN key up to four times to set the console baud rate. Carriage RETURNs do not need to be sent from a Cromemco 3102 terminal since these characters are automatically sent. If switch 3 of the disk controller board is set to the ON position, CDOS will automatically boot up at this point. If switch 3 is set OFF, ROOS will respond with a ";" prompt to which the user must respond with b and a RETURN to boot up CDOS.

The system is now up and running.

Either of the above procedures is known as a cold bootstrap which includes reading CDOS and the I/O routines from disk. All of CDOS is contained in the file CDOS.COM.

Note:

It is advisable to insert the disks after powering-up and remove them before powering-down the machine. The disks may be left in the drives when resetting the machine.
4.1.2 Warm Start and Drive Selection

When a command is issued, the current disk drive is always referred to unless another drive is specified in the command. The current drive can be changed by entering the disk specifier followed by a colon and a carriage RETURN to terminate.

If drive A is the current drive and it is desired to make drive B the current drive, the user should type:

B:<RETURN>

and the console will display B, indicating that drive B is now the current drive.

If an attempt is made to access a file without entering a disk specifier, CDOS will search the current disk and if it is not found will then search the master disk. If a disk specifier is entered, only the specified disk is searched.

Before a program is executed, the system logs off all drives by clearing the bitmaps. This is called a warm start. After a warm start when a drive is accessed a new bitmap will be obtained. See the Stat utility program for a method of determining whether or not a disk has been written to improperly.

4.2 CONTROL FUNCTIONS

Certain nonprinting characters, called control characters, serve to control specific console and printer operations. These characters are described and summarized in the following sections.

4.2.1 Console Control Characters

While typing a command, the standard buffer input mode is active and certain control characters may be used. To type a control character, press the CNTRL key first and hold it in a depressed position while typing the letter. Since a control character is nonprinting, in some applications it will be displayed on the console as the character preceded by an up-arrow (e.g. "I"). Following is a list of control characters and their functions:
"E  Physical carriage return and line feed, go to
the next line without terminating.

Backspace
Underscore
RUBout
DELe te
any of these will delete the last character
entered without echo. These will backspace
the cursor on a CRT terminal.

RETURN
"M  Either of these will terminate a command line.
"R  Retype current line (after many corrections).

PAUSE (3102 only)
"S  Pause during device I/O. This is primarily
used to stop and restart a listing on the
console. Any key may be typed to resume
processing, but only "S can be used to pause.

"U  Delete the current line. Used primarily with
hard copy terminals.

CE (3102 only)
"V  Erase the current line.

"X  Delete the last character with echo. This
deletes and echoes the character following
three backslashes; three forward slashes are
generated by resuming typing. Used with hard
copy terminals.

4.2.2 Printer Control Characters

There are three control characters which are used to
control output to the printer. They are:

"L  CNTRL-L sends a formfeed to the printer.

"N  This character is only for use with Cromemco
Printer model 3103. When this character is
included in a line which is sent to the
printer, it will cause the entire line to be
printed in double width characters. A line
printed in double width characters may contain
only half as many characters as a normal line
because each double width character takes up
twice as much room as a normal character.
PRINT (3102 terminals only)

"P Send all console output to the printer as well as to the terminal. This is a toggle action switch. By entering CNTRL-P output to the console will also be sent to the printer. Output to the printer in this mode can be terminated by entering another CNTRL-P. If a CNTRL-P is inadvertently sent while a printer is either not connected to the system or not enabled, another CNTRL-P will cancel the previous one. CNTRL-P automatically selects 3703 printers.

"T Turn off all output to the printer. This control character can be output by a user program but will have no effect if issued from the console.

"W Send all output to the printer as well as to the console. This control character can be output by a user program but will have no effect if issued from the console.

4.3 AUTOMATIC STARTUP AND PROGRAM EXECUTION

A very powerful feature of CDOS is the ability to enter directly into an application program when powering up the computer. This is done with the Batch file STARTUP.CMD which is accessed after booting up the computer or reentering CDOS. The contents of this Batch file will execute automatically. This is especially useful for the inexperienced user as there is no need to deal with any of the commands which are used to load and execute a program.

The following procedure will cause the BASIC user program MULTIPLY.SAV to automatically begin execution when CDOS is entered.

1. Make sure that there is a copy of the batch command file STARTUP.CMD on disk A.

2. Save the BASIC program you want to RUN in a file (in this example we are using MULTIPLY.SAV). The program must be SAVED (not LISTed) in order for this to work.

Our program for this example is:
100 Rem This is my application program
110 First = 5
120 Second = 10
130 Print "The answer is "; First*Second
140 End

3. Using the Cromemco Screen Editor, create a file named STARTUP.CMD on disk A. This file must be named STARTUP.CMD since this is the filename that CDOS and @ (batch) look for.

In this example the command file should contain the line:

BASIC MULTIPLY.SAV

When CDOS is entered, the batch command will call BASIC which will RUN the saved program MULTIPLY.SAV.

4. When the computer is turned on and CDOS is entered (you must depress the carriage return several times if you do not have a Cromemco 3102 terminal), our example will output the following:

A.@ STARTUP
@ (Batch) version #.##

A.BASIC MULTIPLY.SAV

CROMEMCO 32K STRUCTURED BASIC version #.##
Copyright (c) 1977, 1979 Cromemco, Inc.

The answer is 50
***140 End***

>>

Note:

while the STARTUP.CMD file is controlling the operation of the system, the RETURN key, which is used to terminate a batch command, is disabled. After the STARTUP.CMD file has finished, this function will be returned to its normal mode of operation. The disabling of this function during the startup procedure can be useful in preventing a novice or unskilled user from
inadvertently gaining control of the machine.

See the @ (Batch) command for further information.

4.4 COMMAND STRUCTURE AND SYNTAX

When a user enters a command on the console, CDOS processes the command to determine if it is one of the intrinsic commands (those commands which are internal to CDOS and are not saved as disk files). If the command is intrinsic, it is executed. If the command is not recognized as intrinsic, it is assumed to be a COMMAND file on the disk and CDOS attempts to locate the file with the COM extension. If no disk is specified, the current disk is searched first, and if the file is not located, the master disk. If the program is found, it is loaded into memory starting at 100H, the remainder of the command line is passed to it as control information and execution is started at 100H. If it is not found, a message to that effect is displayed on the console.

The command line starts with an optional disk drive specifier. If this is omitted, the current disk drive is assumed except as noted previously. This is followed by the command with no extension (COM is assumed). The rest of the line is determined by the function being called. The following conventions are observed:

1. All options are preceded by a slash (/).
2. An assignment command generally follows this format:

   Destination-file-ref=Source-file-ref

3. A comma, blank, or equal sign acts as a delimiter to separate filenames.

4. All letters in command lines are translated into upper case upon entry. All filenames appear in upper case only, but may be referenced by any combination of upper and lower case characters.

5. A blank will be ignored except as a delimiter separating filenames.
4.5 **RESET SWITCH**

Pressing or turning the *reset* switch on your Cromemco computer causes a hardware reset. This causes control to be transferred to the power on jump address selected on the ZPU card. With the switches on the ZPU and disk controller cards set as suggested in the appendix, resetting the computer will cause control to be transferred to RDOS and, if switch 1 on the disk controller is ON, causes CDOS to automatically be reloaded into memory (cold bootstrap).

RESET will interrupt any disk operations in progress, so it is recommended that you not press RESET during a disk write operation.

**Note:**

If your terminal is not a Cromemco 3102, the RETURN key must be depressed several times after resetting the computer to reestablish the terminal baud rate.
5.1 CROMEMCO PRINTER DRIVERS

CDOS is supplied with a printer driver designed for use with Cromemco dot matrix printers.

If a Cromemco typewriter quality character printer is to be used as the system printer, the special driver which is supplied with the Cromemco model 3355A printer must be used.

After CDOS has been loaded, place the disk containing the file 3355A.COM in the current drive or in the master drive. Type 3355A followed by a RETURN and a message will be displayed when the driver has been properly loaded. The driver will remain loaded as long as the system is not rebooted.

If the typewriter quality character printer is to be used with the Cromemco Formatter II, the @ty command must be used at the beginning of the file which is to be formatted to specify this. This will cause the Formatter program to use an internal 3355A driver which incorporates microspacing to achieve margin justification. Refer to the Cromemco Formatter II Instruction Manual, part number 023-4027, for further information on this command.

5.2 ADDING NEW I/O DEVICE DRIVERS TO CDOS

Device drivers can be changed or added by modifying the source file to the CDOS I/O drivers which is called DRIVERS.Z-80. This may be used in conjunction with the Batch file, DRIVERS.CMD, to easily modify drivers for devices connected to CDOS. These files are available on the Cromemco Z-80 Macro Assembler diskette, model numbers FDA-L or FDA-S.

The ability to change the CDOS I/O drivers has several uses. First, it is a convenient way to remove portions of CDOS in order to make it occupy less machine memory. Second, it allows you to write custom drivers for nonstandard I/O devices and be able to access these through CDOS. Third, it is possible to have the I/O drivers make a decision on which of several devices to access according to the condition of the CDOS I/O Byte.
A programmer attempting to modify the drivers must be familiar with Z-80 assembly language programming, conditional assembly, the Cromemco Z-80 Macro Assembler, and the design of I/O drivers.

The file containing the CDOS I/O drivers is called DRIVERS.Z-80. This file contains switches for conditional assembly and EQU's for port assignments followed by the routines for the various devices.

The following guidelines should be observed when modifying the drivers:

1. The programmer must follow the instructions and notes in the source listing.

2. Tables must not be moved or changed. This applies to those tables which CDOS needs and expects in certain locations.

3. All routines are preceded by a header which specifies entry and/or exit parameters, register contents, etc. These specifications must be observed as CDOS is dependent upon them.

4. If the programmer uses any of the prime registers or the IX or IY registers their value must be preserved (typically on the stack). The nonprime registers need only be preserved to the extent which they are used.

5. The CDOS stack should not be used to a depth greater than ten (approximately).

The following procedure will create a CDOS with the modified I/O drivers as specified in the file MYDRIVER.Z-80. Notice that although the procedure must be followed step by step, the names of the files may be changed as desired. The commands in boldface are given in response to the CDOS prompt and the subsequent text explains the purpose of each.

XFER/V MYDRIVER.Z-80=DRIVERS.Z-80 makes a copy of the file DRIVERS.Z-80 called MYDRIVER.Z-80. This is done so that the original source file will be saved as a reference and backup.

SCREEN MYDRIVER.Z-80 loads the Screen editor and the file MYDRIVER.Z-80 so that the drivers can be changed. Many changes may be performed by merely changing the EQU's at the beginning of the source. For example, if the console to which CDOS is connected is a Model 3101 rather than a Model 3102, the I/O drivers can be changed.
to reflect this by changing the definition of C3102 in the source to FALSE and C3101 to TRUE. Model 3100 terminals may be selected by changing both C3102 and C3101 as for a Model 3101 terminal, as well as changing FUR.KEYS to FALSE.

ASM MPYDRIVER.@% HEX=0 assembles the drivers in HEX format with an ORG of OH. The filename extension of @@ will instruct the Assembler that the source file is on the current disk, the object file is to be placed on the current disk, and that no print file is to be produced. The address of OH must be used.

REN MYDO.HEX=MPYDRIVER.HEX renames the resultant HEX file.

ASM MYDRIVER.@% HEX=100 assembles the drivers in HEX format with an ORG of 100H. The address of 100H must be used.

REN MYD100.HEX=MYDRIVER.HEX renames the assembled HEX file. The original source file, MYDRIVER.1-80, remains unchanged on the current disk.

CDOSGEN MYD8.HEX MYD100.HEX generates a version of CDOS which includes the modified drivers. The two HEX files are used to relocate the drivers to their final location in CDOS. They must appear in the order shown for CDOSGEN to work correctly. All questions in CDOSGEN must be answered as usual. When CDOSGEN has finished writing the CDOS file to the disk, CDOS must be rebooted up again. To add these drivers to any copies of CDOS you make from now on, simply type this last command:

CDOSGEN Myd8.hex Myd100.hex

An example of using the I/O Byte to select a device is contained in the file DRIVERS.1-86. Two printers, both one serial and one parallel may be connected to CDOS by specifying both the labels C3703 and S.PRINTER as TRUE, and the label NO.LST as 2; then reassembling and relocating the drivers as already described.

The program STAT (version 02.16 or higher) may then be used to select one of these two printers by one of the following commands:

STAT PRT:=0 (or STAT PRT:=PAR:)
STAT PRT:=1 (or STAT PRT:=SER:)

45
If the 3355A driver has been loaded, one of the previous two commands will select another printer in the system. If you wish to access the 3355A again, type:

```
STAT PRT:=2 (or STAT PRT:=TYP:)
```

Other multiple devices may be accessed through CDOS by first changing the the I/O Byte. Note that the standard I/O drivers have the code necessary to access two printers only. Other configurations of multiple devices must be designed and implemented by the user.

The configurations allowed by STAT are as follows:

```
STAT dev:=n:
```

where dev = CON:, RDR:, PUN:, or PRT: and n = 0-7, 0-3, 0-1, or 0-3, respectively. The actual bit format of the CDOS I/O Byte is:

Bits 0,1,2 are assigned to CONsoles 0 through 7; Bits 3,4 are assigned to ReaDeRs 0 through 3; Bit 5 is assigned to FUNches 0 and 1; Bits 6,7 are assigned to PRinTers 0 through 3.
6.1 INTRINSIC COMMANDS

The intrinsic commands reside in the High Memory that is occupied by CDOS after the system has been loaded. Because these commands are intrinsic to CDOS, their execution does not alter the User Area of memory. All files referred to by intrinsic commands are disk files.
6.1.1 **Attributes**

ATTR establishes or changes allowable file access modes.

Format: `ATTR file-ref [+] [p...]`

where:

- `file-ref` is a file reference which may include the * and ? replacement characters.
- `+` is an optional parameter which indicates that the following ATTRIBUTES are to be added to those already describing the file.
- `p...` are optional ATTRIBUTE parameters. They are abbreviated by one or more of the following letters:
  - `E` Erase protect. This file cannot be erased or renamed.
  - `R` Read protect. The system cannot read from this file. The file may be erased or executed.
  - `W` Write protect. The system cannot write to this file. The file may be erased or executed.
  - `S` System file.
  - `U` User file.

Attributes may be deleted by assigning a new set of ATTRIBUTES or by giving the ATTR command with only a file reference and no optional parameters. This will cause all user assignable (erase, read, and write protect) ATTRIBUTES to be deleted. ATTRIBUTES may be added to those already existing by use of the `+` symbol.

**Note:**

ATTR is a software protection only against writing, reading, or erasing disk files. If more positive write protection is desired, the use of a write protect sticker is recommended.
The ATTR intrinsic can also be executed by typing ATRIB instead of ATTR.

Examples:

These examples assume that the following directory is on the current disk:

```
PROGRAM1 FOR 7K
PROG 2K
PROGRAM2 REL 5K
*** 5 Files, 6 Entries, 34 K Displayed, 207 K Left ***
```

This directory indicates that none of the files have limited access modes (i.e., none of the allowable access modes have been altered by ATTR). If the command:

```
ATTR *.FOR R
```

is given, then the directory will appear as follows:

```
PROGRAM1 FOR 7K R
PROG 2K
PROGRAM2 REL 5K
*** 5 Files, 6 Entries, 34 K Displayed, 207 K Left ***
```

The command used an ambiguous file reference to refer to all files on the current disk with the extension FOR (*.FOR). The command instructed the ATTR utility to make all the referenced files Read protected (by means of the R parameter). The R following each of two directory entries indicates that PROGRAM1.FOR and PROGRAM2.FOR have been given a Read protect status. If, following this, the command:

```
ATTR PROGRAM1.FOR +EW
```

is given, then the directory will appear as:

```
PROGRAM1 FOR 7K EWR
PROG 2K
PROGRAM2 REL 5K
*** 5 Files, 6 Entries, 34 K Displayed, 207 K Left ***
```
This time ATTR used a single file reference (PROGRAM1.POR). The command added (by means of the plus sign) categories of protection to the already existing category. The EWR following the file entry in the resulting directory indicates that the file PROGRAM1.POR is now Write and Erase protected in addition to its previous status of being Read protected. If the plus sign had been omitted from the parameters specified for this command, the file would no longer be Read protected as the Write and Erase protect would have replaced, not have been added to, this status.
6.1.2 DIRECTORY

DIR lists disk filenames and sizes followed by a summary of the total disk space used by the files which were listed.

Format:  \texttt{DIR \{y\} \{file-ref\}}

where:

\texttt{y}  

is an optional disk drive specifier. When included in the command line, this parameter will specify the drive whose disk directory is to be examined. When omitted, the DIR command will default to the disk in the current drive. Values acceptable to CDOS are the letters A through H.

\texttt{file-ref}  

is an optional file reference which may include the * and ? replacement characters. When this parameter is included, only filename(s) which match the file reference will be listed.

Each line of the directory listing (except for the last line) includes:

1. filename,
2. filename extension (if one exists),
3. length of the file in kilobytes,
4. \texttt{ATTR}ibute protection of the file.

The last line of the directory is a summary of the listing. This is not always the same as a summary all of the files on the disk. The summary line includes the total number of files, kilobytes, and entries which were listed, as well as the file space remaining on that disk.

For an alphabetized list of filenames and their sizes use Stat/A. An alphabetized list of filenames only is available from Stat/W.

Examples:

Assume that the DIR command, given without any of the optional parameters, will yield the following directory:

51
This is a listing of the names of all of the files on the current disk. If the current drive is not drive C, the command:

```
DIR C:
```

might yield the following directory:

```
FILENAME BAS 5K BASIC COM 19K
*** 2 Files, 3 Entries, 24 K Displayed, 217 K Left ***
```

This is a listing of the names of all the files on the disk in drive C.

The following command would give the user the names of all of the REL files on the current disk:

```
DIR *.REL
```

The directory would appear as:

```
PROGRAM1 REL 2K PROGRAM2 REL 5K
*** 2 Files, 2 Entries, 7 K Displayed, 207 K Left ***
```
6.1.3 **ERAse**

ERA delete(s) file(s) from a disk directory.

**Format:** `ERA file-ref`

where:

`file-ref` is a file reference which may include the * and ? replacement characters. All file(s) which match the file reference will be deleted from the disk directory. The space on the disk which the erased files had occupied will then be available for other use. Files may also be selectively erased with Stat/E which prompts the user with each filename in alphabetical order.

It is possible to delete a great many files at one time using an ambiguous file reference. Caution is recommended when using replacement characters in the ERAse command file reference. Prior to issuing the ERA command, the DIR command may be given with the same file reference in order to obtain a list of the files which will be deleted by the ERA command. If a file has erase attribute protection, the attribute must be removed before the file can be erased.

**Example:**

If the current disk drive directory is:

```
PROGRAM1 FOR  7K   PROGRAM2 FOR 18K
  PROG    2K     PROGRAM1 REL  2K
  PROGRAM2 REL  5K
*** 5 Files, 6 Entries, 34 K Displayed, 207 K Left ***
```

then the command:

```
ERA PROGRAM1.*
```

would erase the two files referred to by the ambiguous file reference. The resulting directory would appear as:

```

53
```
PROGRAM2 FOR 18K
PROGRAM2 REL 5K
*** 5 Files, 6 Entries, 34 K Displayed, 207 K Left ***
6.1.4 RENAME

RENAME changes the filename and/or filename extension of an existing file.

Format: REN new file-ref=old file-ref

where:

new file-ref is a file reference which may include the * and ? replacement characters. This is the file reference which will exist in the disk directory after the execution of the command. Note: If replacement characters are used in the new file-ref, they will be replaced by characters from the filename and filename extension referred to by the old file-ref. Replacement characters never appear in an actual filename or filename extension.

old file-ref is a file reference which may include the * and ? replacement characters. This is the file reference which existed in the disk directory before the execution of the command.

Initially, this command verifies that no file exists on the disk which satisfies the new file-ref. If the new file-ref includes a replacement character, any existing file which satisfies the ambiguous file reference will cause the message 'file already exists' to appear and command execution will be aborted. After this initial check, no further file reference checking takes place. It is possible, in a multiple RENAME command, to create more than one file with the same file reference. It is up to the user to ensure that this does not happen.

Note:

The ambiguous file reference will work only if there is no existing file that matches that reference. For example, if there is a file PROG.REL, then REN *.REL=*HEX won't work. It will work if PROG.REL isn't there.
Examples:

Assume the directory on the current disk drive appears as follows:

PROGRAM1 FOR 7K  PROGRAM2 FOR 18K
PROG 2K  PROGRAM1 REL 2K
PROGRAM2 REL 5K

*** 5 Files, 6 Entries, 34 K Displayed, 207 K Left ***

If the files PROGRAM1.FOR and PROGRAM2.FOR are to be used as text files and the user wants to have their extensions reflect this, the following command will change each filename extension of FOR to TXT on the current disk.

REN *.TXT=*.FOR

If, in addition, the user desired to change the name of the file PROG to PROGRAM.FOR, the following command line would be entered:

REN PROGRAM.FOR=PROG

After giving these two commands, the directory would appear as:

PROGRAM1 TXT 7K  PROGRAM2 TXT 18K
PROGRAM FOR 2K  PROGRAM1 REL 2K
PROGRAM2 REL 5K

*** 5 Files, 6 Entries, 34 K Displayed, 207 K Left ***
6.1.5 **SAVE**

SAVE causes part of the User Area to be saved on disk.

*Format:*  **SAVE** file-ref n

*where:*

- **file-ref** will become the name of the SAVED disk file.
- **n** is the decimal number of 256 byte pages to be saved.

The SAVE command may be used to save a portion of the User Area, beginning at 100H, in a disk file. For example, if a FORTRAN, COBOL, or Assembler program was linked without the /N option, before beginning execution the SAVE command may be issued to create a COMMAND file. A COMMAND file may have any filename and must have the filename extension COM.

The number of pages to be saved is displayed by the linker as the last of a series of three exit parameters enclosed in a set of brackets.

It may also be computed by converting the high byte of the highest address to be saved to decimal (e.g., if the user area is to be saved through address 0BFF0H, convert 0B to decimal (11) and save 11 pages).

Remember that the user area starts at 100H and that the SAVE command saves from this address on.
6.1.6 **TYPE**

TYPE causes an ASCII file to be output to the console (and optionally to the printer).

**Format:** TYPE file-ref

**where:**

file-ref is the file to be TYPEd.

**Note:** that only ASCII files may be TYPEd and that an attempt to TYPE a binary (i.e., relocatable or REL or COM) file will yield unpredictable results.

During the execution of this command all of the applicable console control characters will be in effect. CNTRL-S (PAUSE on a 3102) will cause the listing to pause, CNTRL-P (PRINT on a 3102) will cause the listing to go to the printer, and any other character will abort an active listing. Entering any character will restart a listing which has paused in response to a CNTRL-S.

If a CNTRL-W is included in the file to be TYPEd, all output following this character will be sent to the printer as well as the console. Output to the printer may be stopped by using the CNTRL-T character in the file being TYPEd.
6.2 UTILITY PROGRAMS

Utility programs are not part of CDOS but are supplied with most software packages. They reside on the disk as command files which can be called into the user area as desired. As opposed to intrinsic commands, execution of utility programs does alter the user area.
6.2.1 @ (Batch)

The Batch (@) utility allows the user to automatically execute a sequential list of commands from CDOS. In addition, in the immediate mode it allows the user to create a file of commands for one time execution.

Format (one time mode):
[x:]@[/y] <RETURN>

Format (file mode):
[x:]@[/y] [file-ref] [pl p2...p9]

where:

x is an optional disk drive specifier indicating the location of the batch COM file (@.COM). This parameter is required only if the COM file is not located on either the master drive or the current drive. Applicable values are the letters A through H.

y is an optional disk drive specifier indicating the location of the Batch work file, $$$$CMD.

pl... are optional parameters to be passed to the CMD file.

In file mode, Batch takes its commands sequentially from a file containing all of the commands which are to be executed. In one time mode, Batch will prompt the user with an exclamation mark (!). Valid responses include all legal responses to the CDOS prompt. Execution of the batch command file will commence when a carriage return is entered in response to the prompt. During execution, Batch makes use of its own temporary file, $$$$CMD.

When used in the file mode, the Batch command references an ASCII file containing a list of CDOS commands. This file must have a filename extension of CMD.

The parameters pl through p9 are inserted wherever '1,...,9' appear(s) in the CMD file.
Note:

The file-ref (name of the Batch CMD file) may be referenced by using "@. These are not control characters, but rather are the two separate characters, up-arrow (') followed by a number.

Parameter 0 stands for the command file reference and with it you may refer to the CMD file reference itself. Parameters 1 through 9 are those in the command line. These parameter numbers may be repeated in a file. The up-arrow itself is represented in the command line by two successive up-arrow characters, only one of which is transmitted.

When the Batch command line is given, each word after the filename is treated as a parameter. More complex parameters may be enclosed in single quotation marks. If too many or too few parameters are given, Batch ignores either the extra parameters or the extra commands, respectively.

Examples:

The one-time mode can be used to issue a long string of commands which are to be executed without user intervention. The user might issue the following sequence at the console (the A. is the CDOS prompt while the ! is the Batch one-time mode prompt):

```
A.@<RETURN>
!DIR<RETURN>
!TYPE PROGRAM1.FOR<RETURN>
!REN TEMP=PROGRAM1.FOR<RETURN>
!<RETURN>
```

(Batch - one-time mode)  (types the Directory)  (types the file)  (renames the file)  (begins execution)

Following the null line, Batch immediately begins execution of the three commands issued, giving the command line for each one just prior to execution.

In the file mode Batch allows the user to create a file containing the desired command stream and to execute this file as often as desired. As the following example demonstrates, this can be useful for making a backup CDOS disk. The file used by Batch may be created using the Screen editor and must have an extension of CMD to be found by Batch. In this example, the file used by Batch is called COPY.CMD and contains:

61
XPER/V B:=A:^.*COM
DIR B:

The user inserts a blank diskette containing only the
CDOS resident image into drive B while the master copy
of the CDOS.COM files is in drive A and then types the
Batch command:

@ COPY

The system then copies all files with the filename
extension COM from the disk in drive A to the disk in
drive B. The copy routines are followed by a directory
of disk B so the user may verify that all the desired
files have been copied.

Suppose the user creates a file called EXAMPL.CMD
containing the following:

DIR "1
REN NEWFILE"2

The user then types

@ EXAML OLDFILE '=OLDFILE'

which will call the Batch file EXAML.CMD and pass it
the parameters OLDFILE (for "1) and '=OLDFILE' (for "2).

DIR OLDFILE1
REN NEWFILE=OLDFILE

The system will then type the directory listing OLDFILE
and its size followed by renaming OLDFILE. The equal
sign (=) was included in the single quotation marks so
that it could be passed as part of the second parameter.

The filename "startup.cmd" has special meaning when it
is present on the disk that the system is booted from.
After CDOS is loaded, it checks the master disk for the
file Startup.cmd. If it is present, CDOS will execute
it first before displaying the CDOS prompt.

62
6.2.2 DUMP

DUMP is used to display the contents of a file by 128 byte records.

Format: [x:]DUMP file-ref

where:

x

is an optional disk drive specifier indicating the location of the DUMP command file. This parameter is required only if the COM file is not located on either the master drive or the current drive. Applicable values are the letters A through H.

file-ref

is the file to be DUMPed.

The file is DUMPed in hexadecimal with the first address of a line displayed along the left margin and the ASCII characters corresponding to the hex displayed as characters on the right margin.

Unlike the TYPE intrinsic, both ASCII and binary files may be DUMPed. The records are numbered starting with 0.

63
6.2.3 INITIALIZE

INIT is used to initialize large and small floppy diskettes and hard disks. This process records the track, sector, and surface information on the disk to enable the disk controller hardware to address and retrieve data.

Format: [x:]INIT

where:

x is an optional disk drive specifier indicating the location of the INIT COM file. This parameter is required only if the COM file is not located on either the master drive or the current drive. Values acceptable to CDOS are the letters A through H.

All types of disks require initialization at some point after they are manufactured. Many floppy diskettes supplied by Cromemco have already been initialized and contain data. Cromemco hard disks are always initialized at the factory during testing. Therefore, INIT is a program which you may use infrequently or perhaps not at all.

Cromemco 8 inch floppy disks as supplied have been initialized for double sided use according to the IBM 3740 diskette format. It is recommended that the user not reinitialize these disks when new. Diskettes not supplied by Cromemco or diskettes that are to be used in single sided drives must be initialized. Blank 5 inch floppy disks require initialization before use. Occasionally any disk may require reinitialization due to magnetic damage.

Some of its uses are to initialize new, blank floppy diskettes, to reinitialize floppy disks which have developed soft errors through use with a misaligned drive, and to declare alternate tracks on a hard disk.

INIT is executed by typing its name in response to the CDOS prompt. INIT requires a number of parameters which must be supplied by the user in response to questions the program asks.

The first question asks which drive is to be initialized. INIT determines the allowable responses to this question from CDOS; therefore, it is important that
CDOS has been GENERATED correctly for the computer system it is currently operating.

The user should supply the correct drive letter in response to this question.

INIT will then prompt the user for the format of the disk. You will be asked whether the disk is single sided or double sided and is single density or double density. Bracketed quantities following these questions are default values which can be entered by pressing the RETURN key. These values are derived from your configuration of CDOS.

The next two questions ask for the first and last cylinders to be initialized. If the entire disk is to be initialized, the RETURN key may be pressed twice to enter the default values. INIT is also capable of initializing any single track or any range of tracks.

The last question asks for the surfaces to be initialized. This question also has a default for all the surfaces on that type of drive (press RETURN to select the default). INIT is capable of initializing any single surface as well.

Following the termination of this question by the RETURN key, the program will begin initializing the appropriate disk according to your instructions. It is possible to abort the initialization in an emergency by pressing the ESCape key at this point.

When initialization is finished and control has returned to CDOS, the disk may be labeled using the program SAT2/L.

INITIALIZING a disk will destroy any information which may have been present on the disk.

Switch 4 on the 16PDC or 4PDC board must be off for initialization to take place. Double density initialization is not possible with the 4PDC.
6.2.3.1 Hard disk Alternate Tracks

The INIT program will not return to CDOS immediately following initialization when INITing hard disks. Instead, it will ask one or two further questions about alternate track declaration. The user should be familiar with the track and sector structure of Cromemco hard disks before attempting to answer these questions.

These two questions ask whether you wish to redeclare the existing alternate tracks and whether you wish to add any new alternate tracks to the table. The usual procedure is to answer no to both these questions.

If you answer yes to either of these questions, you will be further prompted for the hard error track to be declared an alternate. These will automatically be assigned a number from 1 to 12 by the program. The program prohibits any illegal or unreasonable responses during this part, and also inhibits a CMTRL-C program abort. This is because the current alternate track declaration is being held in memory and has not yet been written back to the disk. It is strongly recommended that you not reset your computer or otherwise prevent the normal operation of INIT in this section of the program.

Alternate tracks which have been declared at the factory (discovered during testing) should under no circumstances be removed from the alternate track table. Doing so voids any warranties Cromemco makes for that hard disk drive. Cromemco keeps a record of the alternate tracks declared for each drive shipped.
6.2.4 **STATUS**

The program STAT is used to display and change a variety of parameters used by the operating system. Its simplest use is to provide a printout on the console which is a complete summary of all aspects of the computer system. Here is an example of a STAT display:

```
STAT (System Status) version 02.16  9:29:01

SYSTEM MEMORY:
Operating system version 02.36
Total system memory  64 K
Operating system size  14 K
User memory size     49 K

DISK MEMORY:
Disk label     SYSDISK
Date on disk   03-24-81
Total disk space 494 K
Disk space used by directory  4 K
Disk space used by files   426 K
Disk space left      64 K

DEVICE CONFIGURATION:
CON: = Console 0
PRT: = Printer 0 (PAR:)
RDI: = Reader 0
FUS: = Punch 0

DISK CONFIGURATION:
Master disk drive     A
Cluster size          2 K
Sector size           128
Total directory entries 128
Directory entries used 55
Directory entries left 73

DRIVE:  Double sided, Single density
DISKETTE: Double sided, Single density
```

STAT displays with the following information when applicable:

- **Time and Date:** Printed on heading line if previously stored in CDOS.
- **System Memory:** Description of amount and configuration of machine memory.
- **Device Configuration:** Description of device assignment.
- **Disk Memory:** Description of total, used, and available disk space (in kilobytes).
- **Disk Configuration:** Description of total, used, and available disk space (in directory entries). Errors in the directory will be displayed.
Drive: Description of the selected drive.

Diskette: Description of floppy diskette mounted in the selected drive.

STAT, in the /B, /L, or /S modes, runs a validation of the disk directory to see if any cross-linked files have been created or if any clusters have not been allocated. These errors are caused by exchanging diskettes while executing a program that does not provide for this operation.

The general format of the command line for STAT includes a way to request information on any of the disk drives of the system:

\texttt{STAT[/o1]/o2][/on.][\{d:\}][parameters]}

where the on represent one or more of the options described next, \texttt{d:} represents one of the disk drive specifiers (A-H), and parameters represents any of a number of other parameters which may be required. If the drive specifier is omitted, STAT will default to the current drive. Also note that multiple options may be specified; e.g., STAT/D/T and STAT/DT are both legal expressions.

If there is both a Cromemco 3703 (or 3779) and a 3355A printer in your system, you may use STAT to select the printer to be used. After the 3355A driver has been loaded, the 3355A printer will be selected. To access the dot matrix printer, type:

\texttt{STAT PRT:=0 (or STAT PRT:=PAR:1)}

The 3355A printer may be reselected by typing:

\texttt{STAT PRT:=2 (or STAT PRT:=TYP:1)}

Other devices may be accessed through CDOS by first changing the the I/O Byte. Note that the standard I/O drivers have the code necessary to access two printers only. Other configurations of multiple devices may be designed and implemented by the user.
A Option (Alphabetical directory listing)

This option will produce an alphabetical directory of filenames on the selected disk, along with the space allocated to each one and its system attributes. The format of the command is:

```plaintext
STAT/A [x:][file-ref]
```

where `x:` represents a disk specifier (A-H) and `file-ref` represents any single or ambiguous filename on that disk. Normal system status information is not displayed with this option unless the S option is invoked simultaneously. The format of this utility function exactly parallels that of the DIR command.

B Option (Brief system status)

This option allows the user to obtain a quick summary of available disk and machine memory if the normal full system status report is not desired. Upon typing `STAT/B` to select this option, the user is prompted with a display similar to the following:

```
User memory size 49K
Total disk space 243K
Disk space left 34K
Directory entries left 24
```

D Option (set system Date)

This option allows the user to store the current date in CDOS. This date may then be accessed by system or user programs through the Read Date system call (no. 144). The appropriate values will be returned in the A, B, and C registers in binary. Upon typing `STAT/D` to request this option, the user is prompted with

```plaintext
(mm/dd/yy)
```

and is expected to respond with the current month, date, and year. STAT will respond by printing the full date along with the day of the week. Subsequent executions of STAT will display the date on the header line if it has been previously set using the D option.

69
If CDOS is rebooted, the date stored is reset to 00/00/00. The normal printing of system status information is suppressed when the D option is specified. Also note that the date option may be used in conjunction with the time option by typing **STAT/DT**.

Pressing the RETURN key only in response to the date prompt above leaves alone the stored values for date in CDOS. This can be used if the user requested to set the date by means of **STAT/D** and then found it had been set previously.

**E** Option (Erase files)

The **E** option allows the user to erase files from a disk. **STAT/E** differs from the **ERA** intrinsic in that the user does not need to type in the filenames which are to be erased. Another difference is that **STAT/E** displays filenames in alphabetical order whereas **ERA** does not list filenames at all. Ambiguous file references can be made with **STAT/E**. When **STAT/E** is entered

file erase, Query mode (Y=Yes, N=No) [Y] ?

will be displayed. If **N** is entered, all files on the disk will be erased. If **Y** or RETURN is pressed, the filenames will be displayed alphabetically and you will be asked if each file should be deleted:

x:filename extension (Y/N) ?

If **N** is entered,

x:filename extension (Y/N) ? No

the file will not be erased and the next filename will be displayed. If **Y** is entered,

x:filename extension (Y/N) ? Yes, deleted

the file will be erased and you will then be asked about the next file.
If the file is erase protected,

```
x:filename extension (Y/N) ? erase-protected
```

will be displayed and the user will be prompted for the next file.

After the query for the last file,

```
n files erased
```

will be displayed.

**L Option (set Label)**

This option is used to label a disk. Disk labels are a feature of Series-2 CDOS, which both allows users to assign a name and a date to their disk, and enables CDOS to obtain certain important information about that disk for file access. All system disks, including hard disks, should be labeled using the L option. A disk must be labeled before any files or data have been stored on it.

The label option is invoked by typing `STAT/L`. STAT/LS is very useful because it displays information about that disk both before and after labeling. Following the normal printout of system status, the user will be prompted for either three or four items of information which comprise the disk label: 1) whether the disk is single- or double sided, 2) the disk name, 3) the date, and 4) the number of directory entries.

All of these questions are supplied with a default quantity printed in brackets, which the user may specify by pressing the RETURN key only. If the disk has been previously labeled, the defaults will be the values stored in the existing label on the disk. If the disk has no label, the defaults will be those supplied by the STAT program; e.g., "Harddisk" and "Userdisk" are the built-in default names for hard disks and floppy disks, respectively. If a user has previously specified a date using the D option and no date is currently stored on the disk, the default date will be the current date.

The label option may be used to change the number of directory entries of a particular disk. The default values are 64 entries for all floppies except double
sided 8" disks for which the default is 128, and 512 entries for a hard disk. It is frequently desirable to have more than 64 entries on a floppy disk if a large number of short files are being stored.

There is, however, a trade-off: increasing the allowed number of entries above 64 uses additional disk space for the directory. STAT will allow you to enter any value between 64 and 512 for the number of directory entries, but it will round the entered quantity to the next lower number evenly divisible by 4 (thus, 67 would be rounded to 64). In general, to make most efficient use of the disk, the number you enter for directory entries should be a multiple of 32 times the cluster size.

For example, hard disks have a cluster size of 2 Kbytes and thus should have \( n \times (32 \times 2) \) directory entries, where \( n = 1, 2, 3, \ldots, 8 \). You can determine the cluster size for a particular disk from the normal system status display under DISK CONFIGURATION.

If adding or changing a label on a disk necessitates destroying a portion of the present disk directory, STAT will automatically ask whether or not it's OK to do so. Responding N to this question cancels the label request and no label is written. Responding Y to this question clears the present directory and writes the label. Be aware that this effectively creates a blank disk because, even though data may still be stored on the disk, there will be no way to retrieve that information once the directory is cleared.

**M Option (select Master drive)**

The M option allows the user to select a drive to be searched other than drive A if the file cannot be found on the current disk. This can be done by entering

```
STAT/M drive:
```

**N Option (display filenames)**

The N option will display the filenames on a disk in alphabetical order without their sizes. This is the fastest, most compact way to obtain an alphabetical list of the filenames in the directory.
S Option (force Status printout)

The S option is used in conjunction with other options to cause the normal system status display to be performed in addition to the other function(s) requested.

Any of the options described in this section may be specified together; e.g., STAT/A/S and STAT/DTS are both legal expressions.

T Option (set system Time)

This option is similar to the date option except that it allows the user to enter the time. This will also be stored in CDOS, and may be used to set the time of a hardware clock device if the CDOS I/O drivers have been appropriately changed. Users of Series-2 CDOS with 3102 terminals will find that the T option sets the internal clock of the terminal. This may be displayed at any time by pressing CTRL-1 to view the status line.

The time may be accessed by system or user programs through the Read Time system call (146). Refer to the section on CDOS system calls.

If CDOS is rebooted with the system power on, the time will not be changed. If the system power is turned off, the time stored is reset to 00:00:00. The normal printing of system status information is suppressed when the T option is specified. Also note that the time option may be used in conjunction with the date option by typing STAT/DT.

Pressing the RETURN key only in response to the time prompt printed by the T option leaves alone the stored values for time in CDOS. This can be used if the user requested to set the time by means of STAT/T and then found it had been set previously.

% Option (delete all files on a disk)

The % option, which must be used in conjunction with the E option, is similar to the E option without the query. The advantage of the % option is that it may be used in batch mode. Ambiguous file references can be used.

STAT/EZ C:

73
will list all of the files in alphabetical order as they are being erased from the disk in drive C.
6.2.5 WRTSYS

WRTSYS is used to write to or read from the CDOS resident image in the system area of a disk.

Format: [x:]WRTSYS[/s] {d: \{ file-ref-1 \} = \{ f: \{ file-ref-2 \} }

where:

x is an optional disk drive specifier indicating the location of the WRTSYS COM file. This parameter is required only if the COM file is not located on either the master drive or the current drive. Applicable values are the letters A through E.

s is an optional switch indicating that the system is to be written from one disk to another disk, but that only one disk drive is to be used. The program will prompt the user for insertion of the second disk. This is useful for computers having only one drive.

d is a disk drive specifier indicating the disk upon which the CDOS resident image is to be written. Using this specifier with a filename in the described format indicates that CDOS is to be written to the system area of the disk.

f is a disk drive specifier indicating the disk from which the CDOS resident image is to be copied. Using this specifier with a filename in the described format indicates that CDOS is to be copied from the system area of the disk.

file-ref-1 & file-ref-2 are each file references indicating the source and destination files respectively. Using a file reference indicates that CDOS is to be copied to or from the file area of the Disk.
The following conventions apply to both the left (destination) and right (source) sides of the equal sign. If only a disk drive specifier is used in the described format, the CDOS resident image is copied to or from the system area of that disk. If a file reference is used, it must have a filename extension of SYS. In this case the system will be written to or from a user file on the disk.

Note:

Using the WRTSYS program to copy any system files does not change the CDOS which is resident in the computer. To change the operating system in use, CDOS must be rebooted.

WRTSYS also preserves the eight byte label for a particular disk. Thus, one can WRTSYS from a double sided disk to a single sided disk, etc.

Examples:

The command

WRTSYS B:=A:

will copy CDOS from the system area of the disk in drive A to the system area of the disk in drive B. The WRTSYS program will be read from the current disk or, if there is no WRTSYS program on the current disk, from the disk in the master drive.

The command

D:WRTSYS A:=B:BOOT.SYS

will copy BOOT.SYS from the file area of the disk in drive B to the system area of the disk in drive A. The WRTSYS program will be read from the disk in drive D.

The command:

WRTSYS A:SPECIAL.SYS=A:

will copy CDOS from the system area of the disk in drive
A to a file called SPECIAL.SYS in the file area of the same disk. The WRTSYS program will be read from the current disk or, if there is no WRTSYS program on the current disk, from the disk in the master drive.
6.2.6 XFER

The XFER program transfers files from a disk or other device to another disk or device. It can be used in one of two modes. The repeat mode:

Format: [x:]XFER<RETURN>

will repeatedly prompt the user with an exclamation mark (!). Valid responses to this prompt are the same as the portion of the command line following the switches when XFER is used in the one-time mode. To exit to CDOS, press RETURN.

The one time mode will complete one (set of) transfer(s) per command and can be used with the optional switch(es).

Format: [x:]XFER[/s1/s2...] \{d:\} \{file-ref-1 \= file-ref-2[,file-ref-3...]

where:

x

is an optional disk drive specifier indicating the location of the XFER COM file. This parameter is required only if the COM file is not located on either the master drive or the current drive. Applicable values are the letters A through H.

s1,s2...

are any number of the following optional switches (each must be preceded by a slash):

A

transfer ASCII file. Eliminates end of file marker in all but the last of a group of concatenated files and prints a count of the lines copied.

C

Compare files without transfer. This operation is driven by the source (file-ref-2) file. If file-ref-2 is shorter than file-ref-1, and the two files are identical for the length of file-ref-2, then the two files will compare as the same.
F Filter out illegal ASCII characters (ASCII files only).
R transfer Read protected file.
S Strip all rubouts and nulls from file (ASCII files only).
T expand Tabs (ASCII files only).
V Verify files after transfer.
Z Do not print size statistics at completion of XFER.

d is the destination specifier. If a disk specifier alone is used, the original names and extensions of any files transferred will be preserved. Device specifiers can also be used here, e.g., prt:

file-ref-1 is the destination file reference which may include the * and ? replacement characters. If replacement characters are used, the portion of the destination file reference which is ambiguous will match the source file.

file-ref-2... is (are) the source file reference(s). If only one file reference is used, it may include the * and ? replacement characters. If more than one source file is entered, they will be concatenated.

If more than one single file reference is given as the source, the files will be concatenated. If ASCII files are concatenated, the /A switch must be used to remove the end of file markers from between the files.

An ambiguous transfer with verification will be terminated by a verification error.

Note:

The XFER utility will transfer files only to and from the file area of the disk. The WRITSYS utility must be used to write system files to and from the system area of the disk.
XFER will not transfer random access files. Users who must copy random access or ISAM files will need to write a simple program (in the language that created the file) to transfer these files.

Examples:

The command

```
XFER/V B:=PROGRAM1.FOR
```

will copy and verify PROGRAM1.FOR from the current disk to disk B. The copied file will have the same filename and filename extension as the source file. The XFER program will be read from the current drive or the master drive.

The command

```
XFER B:=?.*.FOR
```

will copy all files with the filename extension FOR from drive A to drive B. Each of the copied files will have the same filename and filename extension as each of the source files. The XFER program will be read from the current drive or the master drive.

The command

```
XFER D:=*.*.TXT=A:=*.*.TXT
```

will copy all files with the filename extension TXT from drive A to drive D. Each of the copied files will have the same filename as each of the source files, but will have the filename extension TXT. The XFER program will be read from the current drive or the master drive.

Sending an ASCII file to the printer can be done in the following manner:

```
XFER/T PRT:=E:SOURCE.COB
```

This will copy the COBOL program SOURCE.COB on drive E to the printer. When sending text files to the printer
it is good practice to use the T option so that tabs will be expanded into spaces.

The following command will copy all files from drive A to drive B and then verify these copies:

```
XFER/V B:=A;*:.*
```

The XFER program will be read from the current drive or the master drive.

6.3 EDITORS

6.3.1 Cromemco Screen Editor

The Cromemco Screen Editor enables the user to create, edit, and save ASCII text or program files. The user who is not familiar with the CDOS Text Editor is referred to the Cromemco Screen Editor Instruction Manual (part number 023-0881). In particular, Chapter 2 will aid the novice user by means of an example of an actual Screen session.

The Cromemco Screen editor displays an entire screen of information during the editing process. A cursor in the display can be readily moved around the screen to add, delete, or change information. Special features of Cromemco CRT terminals such as cursor positioning, blinking fields, and programmable function keys are used to simplify operation to the fullest.

One important feature of the Screen editor is that it prompts the user automatically. This is done by using the top line of the screen display as a "menu" of command choices. By referring to this menu there is less need to refer back to the instruction manual during the routine operation of the editor. Another feature of the editor is that the user is politely notified by a beeping tone if an illegal command has been entered.
6.3.2 **Cromemco Text Editor**

The Cromemco CDOS Text Editor, also known as EDIT, enables the user to create, edit, and save ASCII text or program files. The Text Editor is versatile in that it can be used to manipulate and edit text on a line, word, or character basis. Characters and words can be inserted in, deleted from, or changed within a line of text. The point of change can be chosen to be between any two characters. Insertions and deletions can be made that cover more than one line of text. The Text Editor is not encumbered by line numbers or other extraneous information, and operates using only the text itself as a guideline to changes.

The user who is not familiar with the CDOS Text Editor is referred to the **Cromemco Text Editor Instruction Manual**, part number 023-0040.
Chapter 7
PROGRAMMER'S GUIDE

7.1 INTRODUCTION TO CDOS SYSTEM CALLS

To a programmer, system calls are the single most important feature of CDOS. The user who is writing assembly language programs to run under CDOS should become familiar with their use.

A system call is a call to the operating system which initiates a function, usually involving one of the I/O devices. The most important system calls perform I/O with the disk drives. CDOS also has system calls to perform device I/O with CRTs, printers, punches, and readers. System calls are available to perform such special purpose functions as storing and reading the date or time of day and multiplying and dividing integers.

A system call is executed by loading the C register with the number of the call and loading any entry parameters into the specified registers. Upon execution of a 2-80 CALL 5 instruction, CDOS will perform the desired function. When CDOS has finished, it will return to the user program with a RET (return) instruction.

All 2-80 registers will be preserved by system calls except the P (Flag) register and those containing Return Parameters. Programs may safely use the 2-80 set of Primed Registers for temporary storage because system calls which use these registers restore their former values. Entry Parameters are preserved by system calls unless otherwise noted.

All device and disk input and output should be done through the CDOS system calls. This allows user programs to be independent of physical devices or port assignments and assures that the program will be able to run on other Cromemco machines regardless of how I/O devices are connected to those machines. If a change needs to be made in a device driver, it has only to be done once in the system drivers and this change becomes effective in all programs which access that driver through the system calls.

To use one of these routines, the C register must be set to the function number given with the title of each system call. The other registers are set up as the system call requires (for example, the L or DE registers...
usually contain the entry parameter passed). A CALL 5 instruction is then executed to carry out the function. Remember that CDOS initializes location 5 with a jump instruction. This is done so that the location of CDOS in memory is transparent to a user program. A program using the CDOS system functions does not therefore need to (nor should it) perform a CALL to a particular address in High Memory.

7.2 CDOS MEMORY ALLOCATION

CDOS resides in High Memory. It reserves memory below 100H for its own use. The user is left all memory from 100H to the beginning of CDOS, usually about 48K.

A program with the three-letter filename extension COM can be loaded and executed by typing the program name. The program must have its origin at 100H because that is where CDOS loads and executes it. (Note that when saving files that have been linked using the CROMEMCO Linker, they can be LINKed anywhere using the /P option. This is because LINK automatically puts the correct jump instruction at 100H.) After it is loaded, the program can use any memory at all. Note however that if it alters the CDOS areas, it will have no way of communicating with the disk or returning to CDOS. (CDOS will have to be reloaded by resetting the computer.)

When loaded, CDOS places a jump instruction at bytes 0, 1 and 2. If a jump is made to location 0, the CDOS warm start, control will be returned with the prompt for the current drive (e.g., A:). This is the proper method for exiting from a program. Command lines may then be entered from the console keyboard. CDOS places another jump instruction at locations 5, 6 and 7. The normal way to make system requests of CDOS is to call location 5. The address stored at locations 6 and 7 is the address of the beginning of CDOS and thus marks the upper limit of user memory.

The following address map describes the memory area from 0 to 0FFH. All addresses are in hex.
CDOS reentry
3 I/O byte
4 reserved
5....7 system jump call
8 FPP if running under CDOS, C3H if running
under the Cronix CDOS Simulator
30...32 breakpoints for DEBUG
3B...3A jump to "invalid jump" message
40...59 reserved
5A flag
5B flag
5C...6B default File Control Block 1 (FCB-1)
6C...7B default File Control Block 2 (FCB-2)
7C...7F reserved
80...FF default command line buffer

When a COM program is run by typing the program name on
the console, the default command line buffer and default
file control blocks are used as follows. FCB-1 will
contain the first filename, if any, which was typed
after the program name. FCB-2 will contain the second
filename, if any. These filenames will be converted to
FCB format names, i.e., spaces added. The default
buffer will contain the entire command line following
the program name. For example, if this command line is
typed:

PROG FILE1.280 FILE2.COM

CDOS will place "FILE1.Z80" in FCB-1, "FILE2.COM"
in FCB-2, "FILE1.280 FILE2.COM" in the command line
buffer, and load and execute PROG.COM at 100H. Note
that the second FCB starts before the end of the first
FCB (FCB-1 is 33 bytes long and there are 16 bytes
allotted for it if there is an FCB-2). Before using
FCB-1, FCB-2 should be moved. If it is not moved, part
of FCB-2 will be destroyed.

The command line which is placed in the default buffer
can be used to send more than two filenames to a
program, or to start execution of a program with various
options specified. For the following command line:

PROG FILE1.280 FILE2.COM OPTION1 OPTION2

the string of ASCII characters "FILE1.280 FILE2.COM
OPTION1 OPTION2" will be stored beginning at location
81H. The byte at location 80H will contain the length

85
of the string. The byte following the string will contain a null (00). PROG.COM can then look at the command line stored in the default buffer to determine which options were specified.

When a program is loaded, the disk buffer is set to 80H, which is the default command buffer. If the disk is then read to or written from, this buffer will be altered. The program must either reset the disk buffer to another area or move the command line before accessing the disk, if it is desired to save the command line.
7.3 FILE CONTROL BLOCKS

CDOS divides the disk into regions called files. Files are referenced through file control blocks (FCBs). FCBs are 33 bytes long and have the following format:

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disk descriptor before an open (0=current disk, 1 - 8 for drives A - H; the disk number is stored in bits 0 - 3)</td>
</tr>
<tr>
<td></td>
<td><strong>Attribute byte</strong> after an open (attributes are stored in bits 4 - 7)</td>
</tr>
<tr>
<td></td>
<td>bit 7 - write protect</td>
</tr>
<tr>
<td></td>
<td>6 - read protect</td>
</tr>
<tr>
<td></td>
<td>5 - system file</td>
</tr>
<tr>
<td></td>
<td>4 - user file</td>
</tr>
<tr>
<td>1 - 8</td>
<td><strong>filename</strong> (right-filled with blanks)</td>
</tr>
<tr>
<td>9 - 11</td>
<td><strong>File type(extension)</strong> (right-filled with blanks)</td>
</tr>
<tr>
<td>12</td>
<td><strong>File entry or extent</strong> (initially 0; is incremented by one in every new entry of 16 Kbytes)</td>
</tr>
<tr>
<td>13 - 14</td>
<td><strong>Reserved</strong></td>
</tr>
<tr>
<td>15</td>
<td><strong>Record count</strong> (total number of records in this entry)</td>
</tr>
<tr>
<td>16 - 31</td>
<td><strong>Cluster allocation map</strong> (clusters allocated to this entry)</td>
</tr>
<tr>
<td>32</td>
<td><strong>Next record</strong> (next record to be read or written; has the value 0 through 127)</td>
</tr>
</tbody>
</table>
7.4 DIRECTORY ENTRIES

A directory entry is a description of usage of an extent. It describes the attributes, name, and location of the file, or portion of file, in that extent. The structure of directory entries is similar to that of an FCB.

<table>
<thead>
<tr>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>special - bit</td>
</tr>
<tr>
<td>1</td>
<td>filename</td>
</tr>
<tr>
<td>2</td>
<td>extent number</td>
</tr>
<tr>
<td>3</td>
<td>not used</td>
</tr>
<tr>
<td>4</td>
<td>record count in last extent (for hard disks only)</td>
</tr>
<tr>
<td>5</td>
<td>record count</td>
</tr>
<tr>
<td>6</td>
<td>erase protected</td>
</tr>
<tr>
<td>7</td>
<td>write protected</td>
</tr>
<tr>
<td>8</td>
<td>system file attribute</td>
</tr>
<tr>
<td>9</td>
<td>user file attribute</td>
</tr>
<tr>
<td>10</td>
<td>extended file format</td>
</tr>
<tr>
<td>11</td>
<td>not used</td>
</tr>
<tr>
<td>12</td>
<td>either</td>
</tr>
<tr>
<td>13</td>
<td>erased file if the byte value is E5H or disk label if the byte value is 81H</td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16-31</td>
<td>cluster numbers</td>
</tr>
</tbody>
</table>

**Extent number** indicates the number of the directory entry for files larger than 16K. The first directory entry number is zero.

**Record count** indicates how many 128 byte records there are in the entry.

**Cluster numbers** are either one or two byte pointers as defined in the disk label. One byte pointers allow a range of cluster numbers from 0 to 255 and are used on floppy disks. Two byte pointers are used on hard disks and have a range of 0 to 65535. The cluster itself is either 1K or 2K depending upon the disk format, i.e.,
Double sided single density, double sided double density, hard disk, etc.

If the extended file format bit is set in the directory entry this indicates to CDOS that the cluster pointers point to a 2K cluster of directory entries instead of a 2K cluster of file. This is used only on hard disks for files larger than 16K (1 extent).
7.5 DISK LABEL STRUCTURE

The first directory entry is the disk label and its structure is different than that of other directory entries. It includes the name of the disk, the date that the disk was labeled, and disk format information.

Byte   Contents
0      Label flag
       This byte is always 81H
1 - 8  Label name
       (right-filled with blanks)
9 - 11 Date
       Byte 9 = month
       10 = day
       11 = year (relative to 1900)
12     Number of records per cluster
       CDOS records are 128 bytes long. Since cluster size is either 1K or 2K, this
       value is either 8 or 16 (10H).
13     Flags
       Bit 7 = 2-byte cluster pointers
       6 = extended file format (hard disk only)
       5 = bitmap on disk (hard disk only)
       4 through 0 are not used
14     Reserved
15     Record count of directory
       (total number of 128 byte records)
16 - 31 Cluster numbers of the directory

The extended file format bit in the disk label of a hard disk indicates to CDOS that it is necessary to check directory entries to determine if the file is larger than 16K (1 extent).
7.6 INTERRUPTS

During disk I/O operations interrupts are disabled. When a system call is made, interrupts may also be disabled. Registers should be saved on a user stack before an interrupt so that they may be restored after the interrupt and have the desired contents.
7.7 CDOS SYSTEM CALLS

System call:  program abort
Purpose:  This call will abort the current program and return control to CDOS.
Calling parameters:  None
Return parameters:  None

This call has the same effect as jumping to location 0. This is the normal method for exiting from a program.

This call is implemented in the Cromix CDOS Simulator.
System call: read console (with echo) 1 (01H)

Purpose: This call is used to retrieve a single character (one byte) from the console keyboard and echo it to the screen.

Calling parameters: None

Return parameters: A will contain the byte with the parity bit (Bit 7) reset.

CDOS does not return control to the user program until a character has been read and echoed back to the CRT.

Note that a CNTRL-Z ('^Z') character is usually to be considered by a user program as an end of file mark. Also, most other Control characters will not be echoed back to the CRT and some have special meanings for the operating system. For example, CNTRL-J (LF), CNTRL-M (CR), and CNTRL-G (BEL) are echoed directly, CNTRL-I (TAB) is echoed as expanded spaces (see write console), and CNTRL-P will toggle the printer on and off and is not echoed.

This call is implemented in the Cromix CDOS Simulator.
System call: write console
2 (02H)

Purpose: This call is used to write a single
ASCII character (one byte) to the
CRT.

Calling parameters: E contains the byte to be written.

Return parameters: None

CDOS will wait until the console is ready to receive the
caracter and then print it.

After CNTRL-P ('P) is typed while CDOS is outputting
characters with this system call, all subsequent
characters are sent to both the console and the printer
until CNTRL-P is depressed a second time (thus CNTRL-P
acts as a toggle switch).

CNTRL-W ('W) also causes subsequent characters to be
sent to both the console and the printer but must be
encountered in a file to do so. CNTRL-T ('T) in a file
cancels the effect of either the CNTRL-W or the CNTRL-P
and causes characters to be sent only to the console.
CNTRL-W and CNTRL-T may be edited into a file so when
that file is being typed out on the console, it can stop
and start the printer at the appropriate places.

CNTRL-I is the tab character and is converted to spaces
as it is typed out so that the cursor is positioned at
one of the standard tab stops: column 1, 9, 17, 25, 33,
41, 49, 57, 65, or 73. However, the tab is still stored
internally in a file as a single ASCII character (098).

This call is implemented in the Cromix CDOS Simulator.

94
System call: read reader
3 (03H)

Purpose: This call will read one character from a paper tape or card reader or any device connected in its location in the CDOS I/O drivers.

Calling parameters: None

Return parameters: A contains the 8 bits which were read (the parity bit is not stripped).

Since no card or paper tape reader is connected to a standard Cromemco computer system, the port assignments and method of interface (default is serial) for this system call are set up initially with the console as a dummy reader.

Also note that console status is checked during the read for the CNTRL-S (^S) toggle, enabling the user to stop/start the reading process at will. This is useful for pausing during a paper tape jam, for example.

This call is implemented in the Cromix CDOS Simulator,
System call: write punch
4 (04H)

Purpose: This call will punch one character on a paper tape punch or any device connected in its location in the CDOS I/O drivers. All 8 bits are punched (including the parity bit).

Calling parameters: E contains the byte to be punched.

Return parameters: None

The character is placed in the E register. The system will wait until the punch is turned on and is ready to receive the character.

Since no paper tape punch is connected to a standard Cromemco computer system, the port assignments and method of interface (default is serial) for this system call are set up initially with the console as a dummy punch.

Also note that console status is checked during the read for CNTRL-S (^S), enabling the user to stop/start the punching process. This is useful for pausing during a paper tape jam.

This call is implemented in the Cromix CDOS Simulator.
System call: write list
5 (05H)

Purpose: This call will print a single
character (one byte) on the printer.

Calling parameters: E contains the byte to be printed.

Return parameters: None

The character is placed in the E register. The system
will wait until the printer is ready to receive the
character.

Tabs are not expanded, and control characters which do
not have meaning to the printer will be transmitted
anyway. Cromemco printers will ignore such control
characters. A useful control character for the Cromemco
Model 3703 Printer is CNTRL-N ("N"), which, when present
in a line of printer output, will cause that line to be
printed in double width characters.

Also note that console status is checked during the
printing for the CNTRL-S ("S") character, enabling the
user to stop/start the listing. This is useful for
pausing to start a new box of line printer paper.

This call is implemented in the Cromix CDOS Simulator.
System call: get I/O byte
7 (07H)

Purpose: Allows for CDOS to interact with additional or different I/O devices.

Calling parameters: None

Return parameters: A will contain the I0BYTE.

The format of the I0BYTE is:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PRT</td>
</tr>
<tr>
<td>6</td>
<td>Punch</td>
</tr>
<tr>
<td>5</td>
<td>Reader</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Console</td>
</tr>
</tbody>
</table>

I/O Byte

Up to eight devices can be designated, three of which are for paper tape punch and reader, and two for printers. This byte is not used by the standard CDOS I/O drivers. It is, however, used by the 3355A printer driver. The program STAT can modify this byte.

The I0BYTE is stored at location 03H.

This call is implemented in the Cromix CDOS Simulator.
System call: set I/O byte
8 (08H)

Purpose: This call allows the user program to set the IOBYTE.

Calling parameters: E contains the IOBYTE.

Return parameters: None

The format of the IOBYTE is shown in the description of the previous system call.

Up to eight devices can be designated, three of which are for paper tape punch and reader, and two for printers. This byte is not used by the standard CDOS I/O drivers. It is, however, used by the 3355A printer driver. The program STAT can modify this byte.

The IOBYTE is stored at location 03H.

This call is implemented in the Cromix CDOS Simulator.
System call: print buffered line
9 (09H)

Purpose: This call will print a string of ASCII characters which has been
terminated with the dollar sign ($) character.

Calling parameters: DE contains the address of the
beginning of the string.

Return parameters: None

When the line is being output, the following characters
will have special meaning:

CTRL-P ('P) Toggle printer/console link. When
this character is first typed, the
link is toggled on. All characters
will then be sent to the console and
the printer. The next time the
character is typed, the toggle will
be turned off. All characters will
then be sent only to the console.

CTRL-W ('W) Send all output to the printer as
well as to the console.

CTRL-T ('T) Turn off all output to the printer.

This call is implemented in the Crosix CDOS Simulator.
System call: input buffered line
10 (OAH)

Purpose: This call will read an input line from the console.

Calling parameters: DE contains the address of an available buffer.

Return parameters: None

The first byte of the buffer must contain the maximum length of the buffer. On return from this call the second byte of the buffer will contain the actual length entered. The line that is input will be stored beginning at the third byte. If the buffer is not full, the byte at the end of the line will contain a zero.

When the line is being entered, the following characters will have special meaning:

CNTRL-C (**C) Abort. Warm boot back to CDOS.

CNTRL-E (**E) Physical CR-LF. The line is not terminated and nothing is entered into the buffer. This character is used to enter a line longer than can be entered on the console.

CNTRL-P (**P) Toggle printer/console link. When this character is first typed, the link is toggled on. All characters will then be sent to the console and the printer. The next time the character is typed, the toggle will be turned off. All characters will then be sent only to the console.

CNTRL-R (**R) Repeat what has been typed so far on the line.

CNTRL-U (**U) Delete the entered line and go back to beginning of buffer for new line.

CNTRL-V (**V) Delete all previous characters on the current line and back up the cursor (used for CRT terminals).

CNTRL-X (**X) Delete the previous character and
echo the deleted character (used for hard copy terminals).

RUBout Delete the previous character and back up the cursor (used for CRT terminals).

DEL Same as RUBout.

Underscore Same as RUBout.

Backspace ("\^H") Same as RUBout.

This call is implemented in the Cromix CDOS Simulator.
System call: test for console ready
11 (0BB)

Purpose: The console is tested to see if a character has been typed.

Calling parameters: None

Return parameters: A contains -1 (0FFH) if a character was typed.
A contains 0 if no character was typed.

This call may be used during the running of a program to check the console keyboard to see whether a key has been depressed (i.e., CNTRL-C, ESCape, etc.) without causing a noticeable break in the program.

This call is implemented in the Cromix CDOS Simulator.
System call: \texttt{deselect current disk} \\
Purpose: Deselects the current disk. \\
Calling parameters: None \\
Return parameters: None

When a program finishes executing, CDOS logs off the bitmap of all diskettes. This system call logs off the bitmap of the current disk.

Disks should not be changed during program execution unless this call is used because data could be written to an allocated cluster as the bitmap of the old disk is still in memory. The Cromemco Screen Editor uses this call when a disk overflows.

This call is ignored in the Cromix CDOS Simulator.
System call:    | reset CDOS parameter area & select master drive  
| 13 (0DH)        |

Purpose:        | CDOS parameters are initialized and the master drive is selected as the current drive. |

Calling parameters:  | None |

Return parameters: | None |

This call resets CDOS by a jump to location 0, logs off all disks, sets the current drive to A, and sets the disk I/O buffer at 808. Disks will be logged on as soon as they are accessed.

This call is implemented in the Cromix CDOS Simulator.
System call: select current disk drive 14 (0EH)

Purpose: The specified disk drive is selected as the current disk.

Calling parameters: E contains a number corresponding to a drive (0 - 7 for drives A - H).

Return parameters: None

This call should be used in conjunction with search directory for filename (11H) and find next directory entry (12H).

This call is used to change the current disk. CDOS uses this call when you type a disk specifier to change the current disk. BASIC uses this call with the DSK command.

This call is implemented in the Cromix CDOS Simulator.
System Call: open disk file
15 (0FH)

Purpose: This call opens a file to allow reading or writing to that file.

Calling parameters: DE contains the address of the FCB which specifies the filename.

Return parameters: A contains the record number if the file is found.
A contains -1 (0FFH) if the file is not found.

CDOS call 86H may be used before this call to set up a valid FCB from a string.

When this call is made the cluster map in the directory entry is loaded into the FCA.

A file does not need to be opened with this call if it has just been created with create file (16H).

This call is implemented in the Cromix CDOS Simulator.
System call: close disk file
16 (10H)

Purpose: The disk file is closed and the disk
directory is updated (i.e., the FCB
containing updated cluster
information is written to the disk).

Calling
parameters: DX contains the address of the FCB
describing the file to be closed.

Return
parameters: A contains the directory block
number if the file is found.
A contains -1 (0FFH) if the file is
not found.

The file described by the FCB should have been
previously opened or created. A file to which bytes
have just been written must be closed using this
function or the entire last entry (or extent) will be
unable to be read (i.e., no cluster information will be
present for this entry in the directory).

This call is implemented in the Cromix CDOS Simulator.
System call: search directory for filename
Purpose: The directory is searched for the first occurrence of the file specified in the PCB.
Calling parameters: DR contains the address of the PCB.
Return parameters: A contains the block number if the file is found.
A contains -1 (OFFH) if the file is not found.
BL contains the address of the directory entry.

ASCII question mark (?) - 3FH in the PCB matches any character. The current drive will be designated if 3FH appears in the first byte of the PCB and deleted entries will be found as well as valid entries.

An important point to note about this call and the one following (12H) is that they will get the directory entry whether it has been erased or not; i.e., these calls do not check to see if a file has been erased. Files are erased by placing a 0E5H in the first byte of the PCB; the remaining bytes are left unchanged.

This call is implemented in the Cromix CDOS Simulator.
System call: find next directory entry 16 (12H)

Purpose: This call is the same as 11H (17) described previously except that it finds the next occurrence of the filename in the directory.

Calling parameters: DE contains the address of the PCB.

Return parameters: A contains the block number if found (see description of directory block numbers in 0FH - Open Disk File described previously). A contains -1 (OFFH) if the filename is not found.

HL contains the address of the directory entry.

This may be either the next entry of a file occupying several entries (extents), or another filename if the question mark match character (?) is used in the FCB. This call is made after system call 17 and no other disk system function can be executed between these calls.

This call is implemented in the Cromix CDOS Simulator.
System call: delete file 19 (13H)

Purpose: The ambiguous file specified by the FCB is deleted from the disk directory.

Calling parameters: DE contains the address of the FCB.

Return parameters: A contains the number of deleted directory entries.

ASCII question marks (3FH) which appear in the FCB will match any character in the corresponding position of filenames in the directory. A series of eight question marks in the filename portion of the FCB corresponds to an asterisk (*) which is a CDOS ambiguous filename replacement character.

This call is implemented in the Cromix CDOS Simulator.
System call: read next record
20 (14H)

Purpose: The next record (128 bytes) is read into the current disk buffer.

Calling parameters: DE contains the address of the FCB.

Return parameters: A will contain one of the following:

0 - read completed
1 - end of file
2 - read attempted on unwritten cluster (random access files only)

The last byte of the FCB is incremented to read the next record.

The default disk buffer at 80H will be used unless CDOS call 26H is made.

This call is implemented in the Cromix CDOS Simulator.
System call: write next record
21 (15H)

Purpose: The next record (128 bytes) is written into the file from the current disk buffer.

Calling parameters: DB contains the address of the PCB.

Return parameters: A contains one of the following:

0 - write completed
1 - entry error (attempted to close an unopened entry)
2 - out of disk space
-1 - (or FFH) out of directory space

The last byte of the PCB is incremented to be ready to write the next record.

The default disk buffer at 80H will be used unless CDOS call 26H is made.

This call is implemented in the Cromix CDOS Simulator.
System call: 
create file
22 (16H)

Purpose: 
The file specified in the PCB is created on the disk.

Calling parameters: 
DE contains the address of the PCB.

Return parameters: 
A contains the block number of the directory entry (see 0FH - open disk file).
A contains -1 (OFFH) if there is no more directory space or the file already exists.

This call is implemented in the Cromix CDOS Simulator.
System call: rename file
23 (178)

Purpose: This call will rename a disk file.

Calling parameters: DE contains the address of the PCB.

Return parameters: A contains the number of renamed directory entries.

The old filename and file type are in the first 16 bytes and the new filename and file type are in the second 16 bytes of the PCB. ASCII question mark (?) in the PCB will match with any character.

This call is implemented in the Cromix CDOS Simulator.
System call: get disk log-in vector
24 (18H)

Purpose: This call is used to determine which disks are logged in.

Calling parameters: None

Return parameters: A contains a byte specifying which disks are logged in.

Each bit represents one disk drive logged in. If the bit is a one, then it is logged in; else it is off-line. The least significant bit is the A drive, next most significant (Bit 1) is drive B, etc.

CDOS call 18H may be used to determine which drives were used in the program up to the time this call was made.

This call is not implemented in the Cromix CDOS Simulator.
**System call:**
get current disk
25 (19H)

**Purpose:**
The number of the current disk drive is returned.

**Calling parameters:**
None.

**Return parameters:**
A contains a number (0 - 7) corresponding to a drive (A - H).

CDOS uses this call to display the correct CDOS prompt.

CDOS call 19H may be used to get the value of the current drive. This value can be stored so that if the program selects another current drive the program may return to the old current drive.

This call is implemented in the Cromix CDOS Simulator.
System call: set disk buffer
Purpose: This call sets an existing buffer to be used for disk I/O.
Calling parameters: DR contains the address of the disk buffer.
Return parameters: None

This call sets a disk buffer 128 bytes long.

The default disk buffer at location 80H is used if this call is not made. The user should take care not to overwrite the system area from 0H to 100F and CDOS. The bottom of CDOS can be determined with CDOS call 97H.

This call is implemented in the Cromix CDOS Simulator.
System call: get disk cluster allocation map
27 (IBR)

Purpose: Returns information about disk storage.

Calling parameters: None

Return parameters:
- BC contains the address of a bitmap which corresponds to the allocated clusters on the disk.
- DK contains the number of clusters on the current disk.
- HL contains last address in CDOS.
- A contains the number records per cluster.

This call may be used to determine how much free space there is on a disk. This is done by multiplying the number of bits not set in the bitmap by the number of records on the current disk. The number of bits in the bitmap is the same as the number of clusters on the current disk.

This call is not implemented in the Cromix CDOS Simulator.
System call: read console (without echo)
128 (80H)

Purpose: This call is the same as read console (with echo) except that it does not echo the character after it is read.

Calling parameters: None

Return parameters: A contains the byte read.

CDOS does not return control to the user program until a character has been read.

Note that a CNTRL-Z ("Z") character is usually to be considered by a user program as an end of file mark. CNTRL-P will toggle a printer on and off.

This call is implemented in the Cromix CDOS Simulator.
System call: get user-register pointer
129 (81H)

Purpose: This call is provided for expansion of CDOS to a multiprogramming system.

Calling parameters: None

Return parameters: BC contains the address of the user register pointers.

This call may be used to access the Standard Device Driver Table.

Example:

```
LD C,81H
CALL 5
LD HL,3
ADD HL,BC
LD E,(HL)
INC HL
LD E,(HL)
```

DE will now be pointing to the Standard Device Driver Table.

This call is not implemented in the Cromix CDOS Simulator.
System call: set user CONTROL-C abort
130 (82H)

Purpose: When CNTRL-C ("C") is typed, the system normally aborts and returns control to CDOS. This call allows the programmer to change the address to which control is transferred when CNTRL-C is typed (i.e., a user may assign a new function to CNTRL-C).

Calling Parameters: DE contains the address.
If DE contains 0, the system abort is reset.
If DE contains -1 (OFFH), CNTRL-C will be disabled.

Return parameters: None

Jumping to location 0 at any time causes a return to CDOS as well as restoring CNTRL-C to its original function unless DE contained -1. In which case CNTRL-C will be disabled.

If CNTRL-C is disabled, CMD files cannot be aborted by pressing the RETURN key.

This call is implemented in the Cromix CDOS Simulator.
System call: read logical record 131 (83H)

Purpose: This system call will read a logical record from the disk without any attention to the files it may contain (i.e., no FCB is specified). A record is defined to be one record of 128 bytes.

Calling parameters: B contains the disk number (0 for current drive, 1 - 8 for A - H).

If bit 6 of register B is set to 1,
HLDE should contain the record number.
If bit 6 of register B is set to 0,
DE should contain the record number.

If bit 7 of register B is set to 1,
the read is interleaved.
If bit 7 of register B is set to 0,
the read is noninterleaved.

Return parameters: A contains the read status corresponding to one of the following:

0 - OK
1 - I/O error
2 - illegal request
3 - illegal record

Interleaved means the record which is read is found in the order CDOS stores it. Noninterleaved means the record which is read is found in sequential order, the order it is physically stored on the disk.

An example will help to illustrate the use of these parameters. CDOS makes use of 716 sectors on the small single sided single density floppy disks. The record numbers which can legally be loaded into the DE register are 0 through 715 decimal, or 0 through 2CBH. Suppose that DE is loaded with the value 2 and the B register with 0 (current disk, noninterleaved read). Thus, since the sectors are numbered beginning with 1, sector 3 would be read into memory in the disk buffer (located at 80H if it has not been changed). The same read with the B register loaded with 80H (current disk, interleaved read) would read sector 0B (the third sector when they
are read every fifth one).

This call is not implemented in the Cromix CDOS Simulator.
System call: write logical record
132 (84H)

Purpose: This system call will write a logical record or sector to the disk without any attention to the file there (no PCB is specified).

Calling parameters: B contains the disk number (0 for current drive, 1 - 8 for A - H).
If bit 6 of register B is set to 1, HLDE should contain the record number.
If bit 6 of register B is set to 0, DE should contain the record number.
If bit 7 of register B is set to 1, the read is interleaved.
If bit 7 of register B is set to 0, the read is noninterleaved.

Return parameters: A contains the read status corresponding to one of the following:
0 - OK
1 - I/O error
2 - Illegal request
3 - Illegal record

This call is not implemented in the Cromix CDOS Simulator.

125
System call: format name to file control block

Purpose: This system call will build the filename portion of a File Control Block from an input string.

Calling parameters:
- HL contains the address of the start of the input line.
- DE contains the address where the FCB is to be built.

Return parameters:
- HL contains the address of the terminator that ended the build operation.

The input line is of the format:

d:filename.ext

where d: represents an optional disk specifier, one of A-H, the filename is up to 8 letters with a 3 letter extension. If a disk specifier is not included, the current drive will be accessed. The FCB is then built from this input line, converting lower case to upper case. The input line is terminated by an ASCII slash (/), equals (=), comma (,), or any character with an ASCII value less than 21H (such as a space or carriage return).

This call formats only the filename portion of the FCB. System call OFH, open disk file, will complete construction of a valid FCB.

The ambiguous replacement character * will be expanded to question marks to fill out the appropriate portion of the input line.

This call is implemented in the Cromix CDOS Simulator.
System call: update directory entry
135 (87B)

Purpose: The last disk I/O function called must have been system call 17 or 18, Search Directory or Find Next Entry. The directory entry is then updated on the disk; this means that the entry is written back to the disk without the user having to specify a block.

Calling parameters: DE contains the FCB used in the system call 17 or 18.

Return parameters: None

The user merely specifies a filename when calling 17 or 18. This is useful if it is desired to change a directory entry and write it back to the disk.

This call is not implemented in the Cromix CDOS Simulator.
System call: link to new program
136 (888)

Purpose: This enables one command program to call another.

Calling parameters: DR contains the address of the FCB of the new program (which must have an extension of COM).

Return parameters: If the new program is not found, A contains -1 (0FFH). In this case the first 80H bytes (from 100H to 17FH) will be destroyed because this is used in reading the directory.

If the program is found execution begins at 100H, no return is made to the original program.

The default command line buffer and default FCBs for the new program must be set up prior to this call if that program expects to be able to use them.

This call is not implemented in the Cromix CDOS Simulator.
System call: multiply integers

Purpose: This system call provides a 16 bit multiply.

Calling parameters: HL and DE contain the two 16-bit factors.

Return parameters: DE contains the result (i.e., DE = DE*HL).

This call is implemented in the Cromix CDOS Simulator.
System call: divide integers
138 (8AB)

Purpose: This system call provides a 16-bit divide.

Calling parameters:
HL contains the dividend.
DE contains the divisor.

Return parameters:
HL contains the quotient (i.e., HL = HL/DE).
DE contains the remainder (i.e., DE = remainder).

This call is implemented in the Cromix CDOS Simulator.
System call: home drive head
139 (88H)

Purpose: The disk drive specified is sent a command to home the head. The disk drive head will return to track 0.

Calling parameters: B contains the number corresponding to the drive to be homed (0 for current drive and 1 - 8 for drives A - H).

Return parameters: None

This call should be used before using read logical record or write logical record for the first time.

This call is not implemented in the Cromix CDOS Simulator.
System call: eject diskette
140 (8CH)

Purpose: This call will eject a diskette an 8" floppy disk drive.

Calling parameters: \( K \) contains the number corresponding to the drive with the disk to be ejected (0 for current drive and 1-8 for drives A - H).

Return parameters: None

This call will eject a diskette from a Cromemco 8" floppy disk drive with the eject option. Otherwise, the call will have no effect.

This call is not implemented in the Cromix CDOS Simulator.
System call: get CDOS version and release numbers

141 (BDH)

Purpose: This call will return the version and release numbers of CDOS.

Calling parameters: None.

Return parameters: B contains the CDOS version number

Binary Coded Decimal.

C contains the release number in

BCD.

A contains a number corresponding to

the operating system being used:

0 - CDOS

1 - Multi-User BASIC Operating System

2 - Cromix Operating System

The user's program can make this call and check the

version number of CDOS to verify that the operating

system is current enough to include all of the necessary

system calls for the program to function correctly.

This call is implemented in the Cromix CDOS Simulator.

The simulator will return the current version of CDOS.
System call: **set special CRT function**

142 (0EH)

**Purpose:**
This call is used to perform special functions on CRT terminals. The call is designed to be very broad and include as many of the special features available in present-day intelligent terminals as possible. In particular it allows the programmer to take full advantage of the features available in Cromemco Model 3102, 3101, and 3100 CRT terminals.

**Calling parameters:**
DE contains parameters as defined in the following chart:

<table>
<thead>
<tr>
<th>Function</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>address cursor on screen</td>
<td>1-80</td>
<td>1-24</td>
</tr>
<tr>
<td>clear CRT screen</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>home cursor without clearing</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>cursor left one character position</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>cursor right one character position</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>cursor up one line</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>cursor down one line</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>clear to end of line from cursor position</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>clear to end of screen from cursor position</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>intensity set to high light</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>intensity set to low-light</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>intensity set to normal-light</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>keyboard enable</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>keyboard disable</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>dynamic function keys</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>static function keys</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>protected field end</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>blinking characters begin</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>blinking characters end</td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>send from cursor position to end of line</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>send from cursor position to end of screen</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>transmit screen out auxiliary port</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>insert character at present cursor position</td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>delete character at present cursor position</td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>delete line at present cursor position</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>insert line at present cursor position</td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>formatted screen on</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>formatted screen off</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>reverse background field begin</td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>reverse background field end</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>underlining characters begin</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>
underlining characters end 31 0
display message on 32 0
display message off 33 0
CPU message deposit 34 0
HL points to the message which is terminated by 00H.
insert character off 35 0
graphics mode on 36 0
graphics mode off 37 0
cursor on (3102 toggle) 38 0
cursor off (3102 toggle) 39 0
memory lock on 40 0
memory lock off 41 0
line lock 42 0
A contains the line number.
line unlock 43 0
A contains the line number.
read character at cursor 44 0
alarm on 45 0
alarm off 46 0

Return parameters: None except read character at cursor returns the character read in the A register.

Those features marked with an asterisk (*) above are all standard features of a Cromemco Model 3101 terminal. The E register is always loaded with 0 to select any special CRT function except cursor addressing.

For cursor addressing the D register should contain the column address (1 through 80 for Cromemco CRTs) and the E register should contain the row address (1 through 24 for Cromemco CRTs) of the desired cursor position. The system call will generate no error if these values are exceeded. Addressing the cursor at a nonexistent location may cause it to disappear from the screen. The location (1,1) is considered to be the upper left-hand corner and the location (80,24) the lower right-hand corner of the screen.

Dynamic function keys enables the preset function key coding. Static function keys disables those preset functions and each function key sends a unique control character sequence.

This call is implemented in the Cromix CDOS Simulator.
System call: set calendar date
143 (8FH)

Purpose: This call is used to store the date
(day/month/year) in CDOS.

Calling parameters:
B contains the day.
D contains the month.
E contains the year minus 1900.

Return parameters: None

The values entered into the registers will be stored in
locations in CDOS where they may be accessed by user
programs (through system call 144) and thus added to
listings or other output.

The operating system makes no check for the correctness
or plausibility of the incoming values; thus, it is up
to the user to supply this error-checking. Also, the
date is not stored on the disk and is thus volatile
(will be lost if the user reboots or turns off the
power).

The program STAT uses this call to set the current date.

This call is implemented in the Cromix CDOS Simulator.

136
<table>
<thead>
<tr>
<th>System call:</th>
<th>read calendar date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose:</td>
<td>This call is used to retrieve the date (day/mo/yr) stored in CDOS by system call 143.</td>
</tr>
<tr>
<td>Calling parameters:</td>
<td>None</td>
</tr>
<tr>
<td>Return parameters:</td>
<td>A contains the day.</td>
</tr>
<tr>
<td></td>
<td>B contains the month.</td>
</tr>
<tr>
<td></td>
<td>C contains the year minus 1900.</td>
</tr>
</tbody>
</table>

No entry parameters are required other than the value in the C register. Note that the C register is changed by this call unlike most other system calls which preserve C.

This is the function which should be used by a program to recover the last previously stored date from the operating system. Note that if set date has not yet been used, read date will return the values 00/00/00.

The program STAT uses this call to read the current date.

This call is implemented in the Cromix CDOS Simulator.
System call: set time of day
145 (91H)

Purpose: This call is used to store the time of day (sec/min/hr) in CDOS for use by a hardware clock or user program.

Calling parameters:
- B contains the seconds.
- D contains the minutes.
- E contains the hours in 24-hour time.

Return parameters: None

The values in these registers will be stored in locations in CDOS where they may either be accessed and updated by user programs or may in turn be stored in registers of an electronic clock.

The operating system makes no check for the correctness or plausibility of the incoming values. It is up to the user to supply this error checking. Note in the I/O device drivers that a dummy routine is supplied to start clock. This dummy routine is called by the operating system during the set time function; thus, users may substitute their own routine in the drivers to initialize a hardware clock.

The program STAT uses this call to set the current time. If there is a Cromecmo 3102 terminal in the user's system, its clock can be set with STAT/T.

This call is implemented in the Cromix CDOS Simulator.
System call: read time of day
146 (92H)

Purpose: This call is used to retrieve the
time of day (sec/min/hr) stored in
CDOS by system call 145.

Calling parameters: None

Return parameters: A contains the seconds.
B contains the minutes.
C contains the hours in 24-hour
time.

Note that the C register is changed by this call unlike
most other system calls which preserve C.

This is the function which should be used by a program
to recover the last previously stored time from the
operating system. Note that if Set Time has not yet
been used, Read Time will return the values 00/00/00.

The I/O Device Drivers contain a dummy routine to Read
Clock. This dummy routine is called by CDOS during the
Read Time system call. Thus, users may substitute their
own routine in the drivers to read the time from a
hardware clock and store it in the time registers also
supplied in the drivers.

The program STAT uses this call to display the time.

This call is implemented in the Cromix CDOS Simulator.
System call: set program return code
147 (93H)

Purpose: Sets return code for the next program.

Calling parameters: A contains the return code for the next program.

Return parameters: None

The currently running program can use this call as a flag for subsequent programs. When the next program is loaded, CDOS will load the program return code in the A register. The A register should be checked as the first operation in the new program, as CDOS will not retain the value of the return code.

The value of the return code is assigned by the user program and has no meaning for CDOS.

This call is implemented in the Cromix CDOS Simulator.
System call: set file attributes
Purpose: This call is used to set and/or add file protection flags.
Calling parameters: DE contains the FCB address.
B contains a byte the bits of which correspond to file attributes.
Return parameters: None

If the following bits are set to 1 the attributes will be enabled:

<table>
<thead>
<tr>
<th>Bit set</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Erase protect</td>
</tr>
<tr>
<td>6</td>
<td>Write protect</td>
</tr>
<tr>
<td>5</td>
<td>Read protect</td>
</tr>
<tr>
<td>4</td>
<td>Not currently used</td>
</tr>
<tr>
<td>3</td>
<td>Not currently used</td>
</tr>
<tr>
<td>2</td>
<td>Not currently used</td>
</tr>
<tr>
<td>1</td>
<td>Not currently used</td>
</tr>
<tr>
<td>0</td>
<td>Add to current attributes</td>
</tr>
</tbody>
</table>

This call is ignored in the Cromix CDOS Simulator.
System call: read disk label 149 (95H)

Purpose: This call is used to read the label stored at the beginning of a disk directory for all CDOS disks.

Calling parameters: DE contains the address of the FCB entry.

Return parameters: A is 0 if there was no error. A is not 0 if an error occurred.

For hard disks and floppies the label becomes the first entry in the directory. It has roughly the same format as a file FCB, containing both the label name in byted 2-9 and the cluster numbers allocated to the directory in bytes 16-31. The first byte of the entry will be 81H, which indicates that this is a label.

Be aware that since the label always occupies the first entry of a disk, a disk allowing a total of n directory entries will have only n-1 entries available to files. It is also important to note that directory entries of a hard disk represent the space assigned to that file through secondary directories which are transparent to the user. This means that the number of declared directory entries (minus one for the label) is the actual maximum number of files which may be stored on that hard disk. For floppy disks, however, each directory entry represents a maximum of 16 Kbytes of file space. This means that individual files which are allocated more than 16 Kbytes of disk space will be assigned another directory entry for each additional 16K used.

There is a second part to the CDOS disk label which is written to the last eight bytes of the first sector on the disk (in double sided drives this is cylinder 0, side 0, sector 1). The format of these bytes is:

142
bytes 1,2:  The ASCII characters LG for large
diskettes; SM for small diskettes; HD for
hard disks.

bytes 3,4:  The ASCII characters SS for single sided
diskettes; DS for double sided diskettes;
11 for 11 megabyte hard disks.

bytes 5,6:  The ASCII characters SD for single
density; DD for double density.

bytes 7,8:  Reserved for future expansion.

If any of bytes 3 through 6 are missing from a diskette
(e.g., if all 8 bytes are E5H as on a new diskette),
C DOS assumes single sided and/or single density.

Finally, some programmers may find it useful to read and
check the disk label from programs to determine whether
or not the user has inserted the proper diskette. This
may be done through the Read Disk Label system call (no.
149) with the DE register pointing to 32 bytes of free
memory where the label name and other information can be
stored. The byte pointed to by DE should contain a 0 to
read the label of the current disk, and 1-8 to read the
label of drives A-H, respectively.

The desired label name will be read into the 8 bytes
beginning with the memory location pointed to by DE+1.
This will be followed by the last disk date, the cluster
numbers assigned to the directory, and other information
used by C DOS. Disk labels, unlike filenames, may be
both upper and lower case so user programs checking for
a particular label should typically translate all
characters in the label name to upper case. A label
name which is returned as all ASCII periods (2EH)
indicates that that disk has not yet been logged on. A
label name which is returned as all ASCII spaces (20H)
indicates that that disk does not have a label (single
sided, single density floppy).

This call is not implemented in the C ronix C DOS
Simulator.

143
System call:  \textbf{turn drive motors off}  
155 (96H)

Purpose: This call is used to turn off the disk drive motors.

Calling parameters: None

Return parameters: None

No parameters are required on entry or given on return from this call other than the value in the C register.

This call may be used by any program which will perform its primary function in memory over a long period of time during which there will be few disk accesses (e.g., an editor or interpreter).

Note that there is no corollary call to turn the motors on. This will be performed automatically by the operating system the next time any disk operation is attempted. CDOS will also pause for approximately 1 second after turning on the motors and before accessing the disk only if the motor off call has been issued. This is to allow the motors to come up to speed before the disk is accessed. This call has no affect on hard disks.

This call is ignored in the Cromix CDOS Simulator.
System call: set bottom of CDOSt in RAM
151 (97H)

Purpose: This call is used to set the bottom address of CDOSt to a lower value than the one at which CDOSt was originally loaded when it was booted up.

Calling parameters: E contains the high byte of the address of the new bottom of CDOSt.

Return parameters: None

The high byte of the address of the new bottom is placed into the E register prior to executing the call. The low byte is assumed 0; thus, the bottom of CDOSt can never be located on any address other than a 256 byte boundary. If the value is -1 (OFFH) or any other value greater than the high byte of the original bottom when booting up, CDOSt will restore this original bottom address.

This function will change the system call jump at locations 5, 6, and 7. Programs using the address at locations 6 and 7 to determine the size of the present User Area will find this area to be reduced in size. A second set of jumps (9 bytes) will be loaded at the new bottom of CDOSt which points to the old bottom so that system calls will still execute correctly. Note that CDOSt is in no way relocated by this function and will reside in the same memory space as it did previously. The purpose of the call is to make it possible to attach a permanent patch space to CDOSt for programs which are to become a permanent part of the operating system for as long as it resides in memory. The only way the patch space may be removed is by a second set bottom call.

This call is not implemented in the Cromix CDOSt Simulator.

145
System call: read current record
152 (98H)

Purpose: The current record is read into the current disk buffer.

Calling parameters: DE contains the PFB address.

Return parameters: A will contain one of the following:
0 if OK;
1 if end of file;
2 if tried to read an unwritten record.

This call is the same as read next record except that it does not update to the next record. This is useful for random access applications.

The default disk buffer at 80H will be used unless CDOS call 26H is made.

This call is implemented in the Cromix CDOS Simulator.
System call: write current record
153 (99H)

Purpose: The current record is written into the file from the current disk buffer.

Calling parameters: DE contains the PCB address.

Return parameters: A will contain:
0 if OK;
1 if entry error;
2 if out of disk space;
-1 if out of directory space.

This call is the same as write next record except that it does not update to the next record. This is useful for random access applications.

This call is implemented in the Cromix CDOS Simulator.
System call:       check if allocated
                  154 (9AH)

Purpose:          Determines if a record is written.

Calling parameters: DE contains the PCB address.

Return parameters: A is 0 if allocated. A is -1 (0FFH) if not allocated.

This call may be used in conjunction with random files to determine if a record is unwritten.

This call is implemented in the Cromix CDOS Simulator, but always returns 0 in the A register.
System call: list directory
156 (9CH)

Purpose: This call lists the directory of a disk.

Calling parameters: DE contains the FCB address of the filename.

Return parameters: None

Call 86H should be used prior to this call to ensure a valid FCB.

This call is implemented in the Cromix CDOS Simulator.
System call: set options
157 (9DH)

Purpose: This call sets I/O and verify options.

Calling parameters: D contains the desired options.
E contains the mask.

Return parameters: A will contain the old options.

If the following bits are set to 1 the options will be enabled:
The mask should contain a 1 in every bit position to be changed.

0 - CNTRL-P flag
1 - read after write
2 - ESCape key use as carriage RETURN
3 - do not echo carriage RETURN
6 - do not echo

Upon exit from the program options 2, 3, and 6 will be restored to their normal state of 0 and option 1 will be restored to its normal state of 1. Option 0 will not change state upon exit. It is recommended that the user not set read after write because valuable error checking will be lost. Data integrity cannot be assured if there is not a verifying read after the write.

This call is implemented in the Cromix CDOS Simulator.

150
System call: delete extents
158 (9EH)
Purpose: Reduces size of file.
Calling parameters: DE contains the FCB address.
Return parameters: A is 0 if not found. A is 1 if found and erased.

This call is not implemented in the Cromix CDOS Simulator.
System call: \text{get master drive} 159 (9FH)

Purpose: Determines which drive is the master drive.

Calling parameters: None.

Return parameters: A will contain the master drive number. B will contain the number of the last drive used in the batch command (0).

The master drive is the drive which is searched if a file cannot be found on the current drive. If the master drive is the current drive it will be searched only once.

The master drive is set with the M option of the STAT utility.

This call is not implemented in the Cromix CDOS Simulator.
### Summary of CDOS System Calls

The following is a summary table listing all of the system calls implemented in CDOS version 02.17 along with their entry and return parameters. The system calls are listed in numerical order, i.e., by order of the number which is loaded into the C register to achieve the desired function.

<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Entry Parameters</th>
<th>Return Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PROGRAM ABORT</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>1</td>
<td>READ CONSOLE (with echo)</td>
<td>none</td>
<td>A = character (parity bit reset)</td>
</tr>
<tr>
<td>2</td>
<td>WRITE CONSOLE</td>
<td>E = character</td>
<td>none</td>
</tr>
<tr>
<td>3</td>
<td>READ READER</td>
<td>none</td>
<td>A = character</td>
</tr>
<tr>
<td>4</td>
<td>WRITE PUNCH</td>
<td>E = character</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>WRITE LIST</td>
<td>E = character</td>
<td>none</td>
</tr>
<tr>
<td>6</td>
<td>not in use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GET I/O BYTE</td>
<td>none</td>
<td>A = I/O byte</td>
</tr>
<tr>
<td>8</td>
<td>SET I/O BYTE</td>
<td>E = 1/O byte</td>
<td>none</td>
</tr>
<tr>
<td>9</td>
<td>PRINT BUFFERED LINE</td>
<td>DE = buffer address</td>
<td>none</td>
</tr>
<tr>
<td>10</td>
<td>(0AH) INPUT BUFFERED LINE</td>
<td>DE = buffer address</td>
<td>none</td>
</tr>
<tr>
<td>11</td>
<td>(0BH) TEST CONSOLE READY</td>
<td>none</td>
<td>A = -1 (FFH) if ready</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A = 0 if not ready</td>
</tr>
<tr>
<td>12</td>
<td>(0CH) DESELECT CURRENT DISK</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>13</td>
<td>(0DH) RESET CDOS AND SELECT DRIVE A</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>14</td>
<td>(0EH) SELECT CURRENT DISK</td>
<td>E = disk drive no.</td>
<td>none</td>
</tr>
<tr>
<td>15</td>
<td>(0FH) OPEN DISK FILE</td>
<td>DE = FCB address</td>
<td>A = directory block</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A = -1 (FFH) if not found</td>
</tr>
<tr>
<td>16</td>
<td>(10H) CLOSE DISK FILE</td>
<td>DE = FCB address</td>
<td>A = directory block</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A = -1 (FFH) if not found</td>
</tr>
</tbody>
</table>

153
<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Entry Parameters</th>
<th>Return Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 (11H)</td>
<td>SEARCH DIRECTORY</td>
<td>DE = FCB address</td>
<td>A = directory block&lt;br&gt;A = -1 (FFH) if not found</td>
</tr>
<tr>
<td>18 (12H)</td>
<td>FIND NEXT ENTRY</td>
<td>DE = FCB address</td>
<td>A = directory block&lt;br&gt;A = -1 (FFH) if not found</td>
</tr>
<tr>
<td>19 (13H)</td>
<td>DELETE FILE</td>
<td>DE = FCB address</td>
<td>A = number of entries deleted</td>
</tr>
<tr>
<td>20 (14H)</td>
<td>READ NEXT RECORD</td>
<td>DE = FCB address</td>
<td>A = 0 if OK&lt;br&gt;A = 1 if end of file&lt;br&gt;A = 2 if tried to read unwritten records</td>
</tr>
<tr>
<td>21 (15H)</td>
<td>WRITE NEXT RECORD</td>
<td>DE = FCB address</td>
<td>A = 0 if OK&lt;br&gt;A = 1 if entry error&lt;br&gt;A = 2 if out of disk space&lt;br&gt;A = -1 (FFH) if out of directory space</td>
</tr>
<tr>
<td>22 (16H)</td>
<td>CREATE FILE</td>
<td>DE = FCB address</td>
<td>A = directory block&lt;br&gt;A = -1 (FFH) if out of directory space</td>
</tr>
<tr>
<td>23 (17H)</td>
<td>RENAME FILE</td>
<td>DE = FCB address</td>
<td>A = number of entries renamed</td>
</tr>
<tr>
<td>24 (18H)</td>
<td>GET DISK LOG IN VECTOR</td>
<td>none</td>
<td>A = those disks currently logged in</td>
</tr>
<tr>
<td>25 (19H)</td>
<td>CURRENT DISK</td>
<td>none</td>
<td>A = disk drive number</td>
</tr>
<tr>
<td>26 (1AH)</td>
<td>SET DISK BUFFER</td>
<td>DE = buffer address</td>
<td>none</td>
</tr>
<tr>
<td>27 (18H)</td>
<td>DISK CLUSTER ALLOCATION MAP</td>
<td>none</td>
<td>BC = address of bitmap&lt;br&gt;DE = number of clusters&lt;br&gt;HE = last address of CDOS&lt;br&gt;A = records/cluster</td>
</tr>
<tr>
<td>128 (80H)</td>
<td>READ CONSOLE (with no echo)</td>
<td>none</td>
<td>A = character</td>
</tr>
<tr>
<td>129 (81H)</td>
<td>GET USER REGISTER POINTERS</td>
<td>none</td>
<td>BC = pointer to user register pointers</td>
</tr>
<tr>
<td>130 (82H)</td>
<td>SET USER CNTRL-C ABORT</td>
<td>DE = address of <code>C</code> handler (0 to reset; -1 to disable)</td>
<td>none</td>
</tr>
<tr>
<td>Number</td>
<td>Function</td>
<td>Entry Parameters</td>
<td>Return Parameters</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>131 (83H)</td>
<td>READ LOGICAL RECORD</td>
<td>DE = block number</td>
<td>A = 0 if OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B = drive number</td>
<td>A = 1 if I/O error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B top bit = 1 if interleaved</td>
<td>A = 2 if illegal request</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A = 3 if illegal block</td>
</tr>
<tr>
<td>132 (84H)</td>
<td>WAIT LOGICAL RECORD</td>
<td>DE = block number</td>
<td>A = 0 if OK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B = drive number</td>
<td>A = 1 if I/O error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B top bit = 1 if interleaved</td>
<td>A = 2 if illegal request</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A = 3 if illegal block</td>
</tr>
<tr>
<td>133 (85H)</td>
<td>not in use</td>
<td></td>
<td></td>
</tr>
<tr>
<td>134 (86H)</td>
<td>FORMAT NAME TO FILE CONTROL BLOCK</td>
<td>HL = address of string</td>
<td>HL = address of terminator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE = PCB address</td>
<td></td>
</tr>
<tr>
<td>135 (87H)</td>
<td>UPDATE DIRECTORY ENTRY</td>
<td>DE = PCB address</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>136 (88H)</td>
<td>LINK TO PROGRAM</td>
<td>DE = PCB address</td>
<td>A = -1 (FFH) if error; else execute at 100H</td>
</tr>
<tr>
<td>137 (89H)</td>
<td>MULTIPLY INTEGERS</td>
<td>DE = factor 1</td>
<td>DE = product</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HL = factor 2</td>
<td></td>
</tr>
<tr>
<td>138 (8AH)</td>
<td>DIVIDE INTEGERS</td>
<td>HL = dividend</td>
<td>HL = quotient</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DE = divisor</td>
<td>DE = remainder</td>
</tr>
<tr>
<td>139 (8BH)</td>
<td>HOME DRIVE</td>
<td>B = drive number</td>
<td></td>
</tr>
<tr>
<td>140 (8CH)</td>
<td>EJECT DISKETTE</td>
<td>E = drive number</td>
<td></td>
</tr>
<tr>
<td>141 (8DH)</td>
<td>GET VERSION OF OPERATING SYSTEM</td>
<td>none</td>
<td>A = operating system</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B = version-number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C = release-number</td>
</tr>
<tr>
<td>142 (8EH)</td>
<td>SET SPECIAL CKT FUNCTION</td>
<td>D = column address/ special function</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>E = row address/0</td>
<td></td>
</tr>
<tr>
<td>143 (8FH)</td>
<td>SET DATE</td>
<td>B = day</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D = month</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>E = year-1900</td>
<td></td>
</tr>
<tr>
<td>144 (90H)</td>
<td>READ DATE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A = day
B = month
C = year-1900

155
<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Entry Parameters</th>
<th>Return Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>145 (91H)</td>
<td>SET TIME OF DAY</td>
<td>B = seconds \ D = minutes \ E = hours (24 hr. time)</td>
<td>none</td>
</tr>
<tr>
<td>146 (92H)</td>
<td>READ TIME OF DAY</td>
<td>none</td>
<td>A = seconds \ B = minutes \ C = hours (24 hr. time)</td>
</tr>
<tr>
<td>147 (93H)</td>
<td>SET PROGRAM RETURN CODE</td>
<td>A = return code for next program</td>
<td>A = none</td>
</tr>
<tr>
<td>148 (94H)</td>
<td>SET FILE ATTRIBUTES</td>
<td>DE = PCB address \ B = new attributes</td>
<td>none</td>
</tr>
<tr>
<td>149 (95H)</td>
<td>READ DISK LABEL</td>
<td>DE = PCB address</td>
<td>none</td>
</tr>
<tr>
<td>150 (96H)</td>
<td>TURN MOTORS OFF</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>151 (97H)</td>
<td>SET BOTTOM OF CDOS IN RAM</td>
<td>E = high byte of address of bottom of CDOS</td>
<td>none</td>
</tr>
<tr>
<td>152 (98H)</td>
<td>READ CURRENT RECORD</td>
<td>DE = PCB address</td>
<td>A = 0 if OK \ A = 1 if end of file \ A = 2 if tried to read unwritten records</td>
</tr>
<tr>
<td>153 (99H)</td>
<td>WRITE CURRENT RECORD</td>
<td>DE = PCB address</td>
<td>A = 0 if OK \ A = 1 if entry error \ A = 2 if out of disk space \ A = -1 (FFH) if out of directory space</td>
</tr>
<tr>
<td>154 (9AH)</td>
<td>CHECK IF ALLOCATED</td>
<td>DE = PCB address</td>
<td>A = 0 if allocated \ A = -1 if not allocated</td>
</tr>
<tr>
<td>155 (9BH)</td>
<td>not in use</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>156 (9CH)</td>
<td>LIST DIRECTORY</td>
<td>DE = PCB address</td>
<td>none</td>
</tr>
<tr>
<td>157 (9DH)</td>
<td>SET OPTIONS</td>
<td>D = desired option \ E = mask</td>
<td>A = old options \ bit 0 = CMTRLP-P flag \ bit 1 = read after write Options bit 2 = ESCape key use as carriage return \ bit 3 = do not echo carriage return \ bit 6 = do not echo</td>
</tr>
</tbody>
</table>

156
<table>
<thead>
<tr>
<th>Number</th>
<th>Function</th>
<th>Entry Parameters</th>
<th>Return Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>158 (9EH)</td>
<td>DELETE EXTENTS</td>
<td>DE = PCB address</td>
<td>A = 0 if not found</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A = 1 if found and erased</td>
</tr>
<tr>
<td>159 (9FH)</td>
<td>GET MASTER DRIVE</td>
<td>none</td>
<td>A = master drive</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>B = last drive used in batch (0)</td>
</tr>
</tbody>
</table>
In the event of a system malfunction, CDOS displays a complete error message to the aid in the diagnosis and correction of the problem. The following section describes these messages and their interpretation.

8.1 FLOPPY DISK ACCESS ERROR MESSAGES

When the operating system cannot successfully access a diskette an error message is displayed.

Format:

mode Error, Drive x, Cylinder cc, Sector ss, Status=ee

where:

mode stands for one of the following words:

Seek Error occurred in seeking a track on the disk.
Read Error occurred during a read from the disk.
Write Error occurred during a write to the disk.
Home Error occurred in seeking track 0 on the disk.
Read-after-Write Error occurred during the Cyclic Redundancy Check.

x is a letter from A to H which represents the disk drive with the error.

cc is the cylinder number (in hexadecimal) where the error occurred.

ss is the sector number (in hexadecimal) where the error occurred.

ee is the 8 bit status byte displayed in hexadecimal which describes the error and the conditions at the time the error occurred.
The status byte will be a hexadecimal number that will either be one of the hex values in the above table or the combination of two or more of those hex values. The bits which correspond to those hex values will describe the reasons or the error.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Hex value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

If the status byte was 0A, the bits set would be 3, 1, and 0 because the only combination of corresponding hexadecimal values that add up to 0A are the ones which correspond to bits 3, 1, and 0.

The following table describes the malfunctions corresponding to the bits set in the status byte.

<table>
<thead>
<tr>
<th>Status Bits Set</th>
<th>Status Bits</th>
<th>Seek</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7 - not ready</td>
<td>not ready</td>
<td>not ready</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6 - write protect*</td>
<td>record type*</td>
<td>write protect</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5 - head engaged*</td>
<td>record type*</td>
<td>write fault</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4 - seek error</td>
<td>record not found</td>
<td>record not found</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3 - crc error</td>
<td>crc error</td>
<td>crc error</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2 - track 0*</td>
<td>lost data</td>
<td>lost data</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1 - index*</td>
<td>data request*</td>
<td>data request*</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 - busy*</td>
<td>busy*</td>
<td>busy*</td>
<td></td>
</tr>
</tbody>
</table>

The asterisk (*) in the above table indicates that the condition is not the cause of the error message, but

160
that it was present when the error occurred. For example, if the status byte was 30H during a Seek error, this means that bits 4 and 5 are set (=1). This is a Seek error and the head is engaged. The head is supposed to be engaged during a seek and therefore this condition is not an error and is marked with an asterisk. CRC stands for Cyclic Redundancy Check. It is a verification that is done after a Read or Read-after-Write operation. A CRC error indicates that data was not transferred without error.

There are four possible responses to the error message:

R
This will cause the system to retry the disk access which caused the error.

Note:
The error message does not appear until after the disk access instruction has been repeated ten times.

I
This will cause the system to Ignore the error message and continue. The function which caused the error message is not completed and no error code is returned to the calling program.

C
This will cause the system to Continue. The function which caused the error message is not completed and an error code is returned to the calling program.

CNTRL-C
This will abort the program and return control to the CDOS monitor.

Examples:
The following examples use some of the more common status codes:

Seek Error, Drive A, Track 17, Sector 1A, Status=36

During a Seek operation, status code 36 or B6 indicates that the system expected to find a mini disk drive when there was actually a maxi drive (or vice versa) at the location (specified by A above). CDOSGEN may be run to correct this problem. Be sure that the disk drives are
correctly specified as small and large during the system
generation.

Read Error, Drive B, Track 1C, Sector 10, Status=10

During a Read operation, status code 10 or 08 indicate
that the data is not readable. This may be caused by
bringing the disk close to a magnetic source or by
scratching or otherwise mishandling the disk.

8.2 HARD DISK ERROR MESSAGES

If CDOS should encounter an error when accessing a hard
disk drive, it will display the error in the following
format:

mode Drive d Cylinder cc Surface hh Sector ss Status ffss

where:

mode is either Read error, Write error,
Read-after-Write error, Home error, or
Seek error.
d is the letter of the drive.
c is number of the cylinder in hexadecimal.
h is head number.
ss is the sector number in hex.
ffss is the error number. The first two
digits indicate the fatal error number
and the second two digits indicate the
system error number.

Hard Disk Fatal Errors

The following error codes are displayed when a fatal
disk error occurs:
00 Failed to Seek & Read Header during R/W
An error occurred during an attempt to seek & read
header preceding a read/write operation.

01 Failed to Seek - Timeout
The seek did not complete within a specified time.
Check the drive electronics.

02 Fault Occurred during Seek
During the seek, a fault error occurred within the
drive, as reported by the drive. This may be any
of several errors.

03 Failed to Seek to Correct Track
The sector header as read off the disk is not what
the drivers expected, thus the current disk
location is incorrect.

04 Failed to Read CRC of Header
The CRC for the header as read from the disk is
incorrect; it is different than what was expected.
Most likely the current disk location is incorrect
or the media surface is damaged.

05 Failed to Rezero - Timeout
A rezero command did not complete within a
specified time. Check the drive electronics.

06 Fault Occurred after Rezeroing
A fault error occurred within the drive after a
rezero command was executed. This may be any of
several errors.

07 Drive not Ready
The ready signal from the drive is not active.
Make sure the drive is connected properly.
08 Failed to Write - Fault Error
During the write, a fault error occurred within the drive, as reported by the drive. This may be any of several errors.

09 Failed to Verify after Write
After data is written to the disk, it is read back and verified. This error occurs if the data cannot be properly verified.

0A Failed to Read - Fault Error
During the read, a fault error occurred within the drive, as reported by the drive. This may be any of several errors.

0B Failed to Read - CRC Error
The CRC just read from the disk is incorrect; it is different from the expected CRC. This error usually means that the data just read is incorrect.

0C Failed to Read - Cannot Locate Sector
The sector being looked for cannot be found on the current track. This error can occur if the media surface is damaged or if the controller electronics are not functioning properly.

0D Surface is Write Protected
The surface selected for the current write command is write protected and cannot be written to.

Hard Disk System Errors
The following error codes are displayed when a system disk error occurs:

00 No Acknowledge Received from Drive
The drive did not acknowledge a command sent to it. Make sure the drive is connected properly.
01 Drive Remains BUSY - Acknowledge Stuck Low
   The acknowledge signal from the drive did not go high again after the command strobe went inactive.

02 Timeout Occurred during Rezeroing
   A rezero command did not complete within a specified time. Check the drive electronics.

03 Fault Condition Reported by Drive
   A fault condition occurred within the drive, as reported by the drive. This may be any of several errors.

04 Failed to Read - CRC Error
   The CRC just read from the disk is incorrect; it is different from the expected CRC. This error usually means that the data just read is incorrect.

05 Header Off the Disk Does Not Compare with Expected Header
   The sector header as read off the disk is not what the drivers expected, thus the current disk location is incorrect.

06 Failed to Verify after Write Operation
   After data is written to the disk, it is read back and verified. This error occurs if the data cannot be properly verified.

8.3 SYSTEM ERROR MESSAGES

Bad directory block dddd
   An attempt was made to read the directory block at location dddd which was overwritten with inappropriate data.
Bad disk block overwritten
A response of C was entered in response to an error which occurred while attempting to SAVE a file.

Cannot read double density diskettes
An attempt was made to access double density diskettes via a CDOS that was configured for single density drives only.

Cannot read double sided diskettes
An attempt was made to access double sided diskettes via a CDOS that was configured for single sided drives only.

CDOS.COM not found
An attempt was made to boot and there was no CDOS.COM file on either the current drive or the master drive.

Drive x write-protected
Diskette in drive x write-protected
The first message will appear if an attempt was made to write to a hard disk that was write protected with the key lock on its rear panel. The second message will appear if an attempt was made to write to either an 8" diskette without a write-enable sticker or a 5" diskette with a write-protect sticker.

Drive not found
An attempt was made to access a drive which was not included in the current CDOS configuration.

Drive not ready
An attempt was made to access a drive which did not have a diskette in it.
File already exists
An attempt was made to rename a file using a name that already exists.

File not found
An attempt was made to access a file which was not on the current disk or the master disk, e.g., REN OLDNAME.TXT=NEWNAME.TXT when OLDNAME.TXT does not exist.

file-ref program too big
An attempt was made to load a program, file-ref, which was too big to fit into memory.

Illegal system call cccH at aaaaH
An attempt was made to access a CDOS call ccc which does not exist. The call was made at location aaaaH.

Invalid jump to location xxxx
where xxxx is the hexadecimal address to which control was transferred. An instruction was executed which caused control to be transferred to a nonexistent memory location or any memory location containing 0FFH (Restart 38H).

Logical disk error
An attempt was made to access a sector which was not on the disk. This is usually due to an error in the disk directory.

Program not found
An attempt was made to run a program with an extension of COM which was not on the current disk or the master disk.
Appendix A

GLOSSARY OF TERMS AND SYMBOLS

{} Braces are used to indicate a choice of items. One of the items enclosed in the braces must be used in the position indicated. An optional choice of items is indicated by braces enclosed in square brackets.

[] Square brackets are used to indicate an optional quantity. The item enclosed in square brackets may be used, in the position indicated, at the user's discretion.

Ambiguous file Reference
This is a file reference which may refer to more than one file by using a replacement character(s).

ASCII
American Standard Code for Information Interchange.

Attribute
The type of protection assigned to a disk file.

Bitmap
A bitmap is a record of the allocation of clusters on a disk. On floppy disks the bitmap is derived from the directory. On hard disks the bitmap is stored on the disk itself.
Cluster
A group of bytes on a disk. CDOS accesses the disk by clusters. A cluster may be 1024 or 2048 bytes depending upon the disk format (single or double density).

Device driver
A program which controls the operation of a peripheral device, such as the console, printer, or disk.

Directory
A list of the user files contained on the disk.

Disk Specifier
A disk specifier is one of the letters from A through H followed by a colon. This letter references a disk drive and allows the user to refer to a disk located in the drive. The disk specifier is an optional part of a file reference.

Extent
An area on the disk occupied by a file or a portion of a file, up to 16K bytes long. There is one disk directory entry for each extent occupied by a file.

File Area (disk)
User files are stored on this part of the disk. The contents of this part of the disk are listed by the DIRectory command.

File Control Block (PCB)
One of two areas starting at addresses 5Ch and 6Ch used by CDOS. The PCB contains the information CDOS needs to manipulate a disk file.

170
Filename

This is a one to eight character label which is used to refer to a file. Several files may have the same filename. These files may be uniquely identified by the use of a disk specifier and/or a filename extension. A filename is a necessary part of a file reference.

Filename Extension

This is a one to three character label which is frequently used to indicate how a file is to be used. A filename extension is an optional part of a file reference.

File or Data File

A file is a collection of bytes containing related information. This information is addressed by means of a file reference and usually resides on a floppy diskette.

File Reference

A file reference identifies and locates a file.

Format: [x:]filename[.ext]

where:

x is an optional disk drive specifier.

filename is a filename up to 8 characters long.

ext is an optional filename extension up to 3 characters long.

A file reference is a single file reference unless it is specifically stated that it may incorporate replacement characters to form an ambiguous file reference.
Intrinsic
A command in CDOS that is executed from the console, such as DIR or ATTR.

Label
The first entry in each disk directory used by CDOS to identify the disk and to keep information about the directory.

Replacement Character
A replacement character is an asterisk (*) or a question mark (?). These characters may be used where specifically indicated in order to create an ambiguous file reference.

Single File Reference
This is a label specifying a unique file. This file reference may not include replacement characters.

System Area (disk)
The boot loader of CDOS is stored on this part of the disk. This section is normally accessed only by CDOS. It does not appear in the user area DIRECTory.

System Call
A CDOS subroutine that may be accessed by a user program by placing the system call number in the C register, setting up all other registers as required by the call, and executing a CALL 5 instruction.
Text file

A file that consists only of printable ASCII encoded characters and ASCII print control characters.

User Area (RAM)

The User Area is RAM which is available to user programs. This is the part of memory from 100H up to the bottom of CDOS. The size of this area may be determined by executing STAT.

Utility

A program that performs a useful function; specifically one of the program supplied with CDOS, such as STAT or XFER.
Appendix B

SWITCH SETTINGS

16FDC

A brief description of the function of each of the 16FDC switches and their recommended settings follows. For further information on the 16FDC switch settings please refer to the Cromemco 16FDC Disk Controller Manual (part number 023-2004). Switch settings for the 4FDC are identical with those of 16FDC listed here.

Switch 1 is the RDOS (PROM Resident Disk Operating System) DISABLE switch. When ON, the PROM containing RDOS cannot be accessed. When OFF, the PROM resides from C000H to C3FFH in memory during startup. This switch should be OFF for initial system operation.

Switch 2 is the RDOS DISABLE AFTER BOOT switch. When ON, RDOS will automatically be disabled from address space following CDOS boot. When OFF, RDOS remains in memory at C000H following CDOS boot. This switch should be ON for initial system operation.

Switch 3 is the BOOT ENABLE switch. When ON, CDOS boot strap is executed from power-on or a computer reset. When OFF, RDOS comes up when power is applied to the system or when the computer is reset. This switch should be ON for initial system operation.

Switch 4 is the INITIALIZATION INHIBIT switch. When ON, diskettes cannot be initialized under software control. When OFF, disks may be initialized. This switch may be ON or OFF for initial system operation.

Note:

When configuring a system with 64 kilobytes of memory, it is important that switch 2 be ON. This will disable RDOS after CDOS is booted up so that RDOS and system memory do not overlap at locations C000H to C3FFH.

With switch 2 ON the only way RDOS can be reentered after booting CDOS is by resetting the machine. If switch 3 is also ON, the user will never be able to

175
access RDOS because CDOS will automatically be booted up
any time RDOS is called.

ZPU

The power-on jump should initially be set to C000H, the
location of RDOS. To do this, the DIP switch should be
set as follows:

#15 = 1 (off)
#14 = 1 (off)
#13 = 0 (on)
#12 = 0 (on)

The clock switch should be set to 4MHz.
Cromesco CDS User's Manual
C. Unassembled Source Listings

TRUE EQU -1
FALSE EQU 0

; At least one of the following three names must be TRUE to prevent errors:
C3102 EQU TRUE ; Cromesco Model-3102 Terminal
C3101 EQU FALSE ; Cromesco Model-3101 Terminal
ADM3A EQU FALSE ; TRUE to include ADM-3A CRT driver

; The state of the following name should match that of C3102 or C3101:
FUN.KEYS EQU TRUE ; TRUE to assemble function key decoding routines

; The following two names may be either TRUE or FALSE:
S.READER EQU FALSE ; TRUE for serial reader connected to TUART/
S.PUNCH EQU FALSE ; FALSE for reader driver same as CIN

; At least one of the following three names must be TRUE to prevent errors:
; (C3703 and C3779 both TRUE counts as only 1 of the printers of NO, LIST)
C3703 EQU TRUE ; Cromesco Model-3703 Printer
C3779 EQU FALSE ; Cromesco Model-3779 Printer

; Numbers of devices to be accessed by CDS:
NO.CORG EQU 1 ; Number of consoles to be accessed ($F maximum)
NO.RDR EQU 0 ; Number of readers to be accessed ($4 maximum)
NO.PUN EQU 0 ; Number of punches to be accessed ($2 maximum)
NO.LIST EQU 1 ; Number of printers to be accessed ($4 maximum)

; I/O byte defined values:
I0.B0 EQU 0 ; I/O byte - used by multiple-device routines
I0.B1 EQU 1 ; I/O byte bit 0 (Console bit 0)
I0.B2 EQU 2 ; I/O byte bit 1 (Console bit 1)
I0.B3 EQU 3 ; I/O byte bit 2 (Console bit 2)
I0.B4 EQU 4 ; I/O byte bit 3 (Reader bit 0)
I0.B5 EQU 5 ; I/O byte bit 4 (Reader bit 1)
I0.B6 EQU 6 ; I/O byte bit 5 (Printer bit 0)
I0.B7 EQU 7 ; I/O byte bit 7 (Printer bit 1)

; Miscellaneous defined values:
NULLS EQU 6 ; Number of nulls transmitted after line feeds
PAGE.SII EQU 66 ; Number of lines of text per page for printer

177
SUBTITLE ASCII Character Definitions

; ASCII characters
CTRLA EQU 2          ; ASCII control-A character
BACK EQU 8           ; ASCII back space
LF EQU 0AH           ; ASCII line feed
VT EQU 0AH           ; ASCII vertical tab
FORM EQU 0CR         ; ASCII form feed
CR EQU 0DH           ; ASCII carriage return
CTRLM EQU 0EH        ; ASCII control-M character
CTRLO EQU 0FH        ; ASCII control-O character
CTRLP EQU 10H        ; ASCII control-P character
CTRLQ EQU 11H        ; ASCII control-Q character
CTRLS EQU 13H        ; ASCII control-S character
CTRLV EQU 16H        ; ASCII control-V character
CTRLW EQU 17H        ; ASCII control-W character
CTRLX EQU 1AH        ; ASCII control-X character
ESC EQU 1BH          ; ASCII escape character
CTRLZ EQU 1DH        ; ASCII control-Z character
CTRLSP EQU 1BH       ; ASCII control-SP character
SPC EQU 20H          ; ASCII space character
AUXTYL: Device Port Assignments, Status Bits, and Baud Rates

I/O device port assignments and status bits

<table>
<thead>
<tr>
<th>EQU</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSTATTP</td>
<td>0</td>
<td>Console status port (input)</td>
</tr>
<tr>
<td>CDATA</td>
<td>CSTATTP+1</td>
<td>Console data port (input/output)</td>
</tr>
<tr>
<td>CRDA</td>
<td>40H</td>
<td>Console Receiver-Busy Available mask</td>
</tr>
<tr>
<td>CTRB</td>
<td>80H</td>
<td>Console Transmitter-Buffer-Empty mask</td>
</tr>
<tr>
<td>RSTATTP</td>
<td>20H</td>
<td>Serial reader status port (input)</td>
</tr>
<tr>
<td>RBAUD</td>
<td>RSTATTP</td>
<td>Serial reader baud rate port (output)</td>
</tr>
<tr>
<td>RDATA</td>
<td>RSTATTP+1</td>
<td>Serial reader data port (input)</td>
</tr>
<tr>
<td>RRD A</td>
<td>40H</td>
<td>Serial reader RD bit mask</td>
</tr>
<tr>
<td>PSTAT TP</td>
<td>20H</td>
<td>Serial punch status port (input)</td>
</tr>
<tr>
<td>PBAUD</td>
<td>PSTAT TP</td>
<td>Serial punch baud rate port (output)</td>
</tr>
<tr>
<td>PDATA</td>
<td>PSTAT TP+1</td>
<td>Serial punch data port (output)</td>
</tr>
<tr>
<td>PTEBS</td>
<td>80H</td>
<td>Serial punch TBE bit mask</td>
</tr>
<tr>
<td>LSTAT TP</td>
<td>54H</td>
<td>List device status port (input)</td>
</tr>
<tr>
<td>LDATA</td>
<td>LSTAT TP</td>
<td>List device data port (output)</td>
</tr>
<tr>
<td>LRTSTP</td>
<td>20H</td>
<td>List device Ready-To-Print bit mask</td>
</tr>
<tr>
<td>LSTROB</td>
<td>7</td>
<td>List device strobe bit</td>
</tr>
<tr>
<td>SSTAT TP</td>
<td>50H</td>
<td>Serial printer status port (input)</td>
</tr>
<tr>
<td>SBAUD</td>
<td>SSTAT TP</td>
<td>Serial printer baud rate port (output)</td>
</tr>
<tr>
<td>SDATA</td>
<td>SSTAT TP+1</td>
<td>Serial printer data port (output)</td>
</tr>
<tr>
<td>STBE</td>
<td>80H</td>
<td>Serial printer TBE bit mask</td>
</tr>
</tbody>
</table>

I/O device baud rate assignment table for TURNT

- 01H = 110 baud / 2 stop bits
- 02H = 110 baud / 1 stop bit
- 04H = 300 baud / 2 stop bit
- 08H = 1200 baud / 1 stop bit
- 09H = 2400 baud / 1 stop bit
- 0A H = 4800 baud / 1 stop bit
- 0BH = 9600 baud / 1 stop bit

(Refer to TURNT manual for other rate or stop bit configurations)

The following baud rates were chosen from the table above:

- RSB.RD.RT EQU 01H  | Baud rate of serial reader
- PUS.BD.RT EQU 01H  | Baud rate of serial punch
- SER.RD.RT EQU 04H  | Baud rate of serial printer

179
SUBTTYL: Device Driver Address Table

; The following is a table of addresses needed by CDOS
; to find the starting locations of each of the I/O device
; routines. The address values are filled in by CDOSREG.
; therefore, this table MUST NOT be removed from the drivers.

CONSOLE: DW CINIT   ; Console initialize
         DW CSTAT   ; Console input-status
         IF FUN,KEYS
         DW C鸪ECIN ; Console input a byte or function key
         ENDIF
         IF NOT FUN,KEYS
         DW Cтрм   ; Condition #2
         ENDIF
         DW CIN     ; Console input a byte
         DW CRDY    ; End conditional #2
         DW COUT    ; Console output a byte
         DW CSРТ    ; Console set special command

READER: DW RINIT    ; Reader initialize
         DW RSTAT    ; Reader input-status
         DW RIN      ; Reader input a byte

PUNCH:  DW PINIT     ; Punch initialize
         DW PSTAT    ; Punch output-ready
         DW POUT     ; Punch output a byte

PRINTER: DW LINIT    ; List initialize
          DW LSTAT    ; List output-ready
          DW LOUT     ; List output a byte

CLOCK:  DW STRTCLOE ; Start clock
         DW READCLOE ; Read clock

YEAR:   DB 0        ; Year (-1900) binary storage
MONTH:  DB 0        ; Month binary storage
DATE:   DB 0        ; Date binary storage
HOURS:  DB 0        ; Hour binary storage
MIN:    DB 0        ; Minute binary storage
SEC:    DB 0        ; Second binary storage
SUBTL: Function Key Address Table and Dummy Return Routine

; The following is a table of addresses needed by CDOS to
; locate the pre-programmed value of each of the function
; keys. The first 20 address values are filled in by CDOSGEN
; and MUST NOT be removed from the drivers.

FUNCADDR:

DM 0 ; Function key F1 (3102 and 3103)
DM 0 ; Function key F2
DM 0 ; Function key F3
DM 0 ; Function key F4
DM 0 ; Function key F5
DM 0 ; Function key F6
DM 0 ; Function key F7
DM 0 ; Function key F8
DM 0 ; Function key F9
DM 0 ; Function key F10
DM 0 ; Function key F11
DM 0 ; Function key F12
DM 0 ; Function key F13
DM 0 ; Function key F14
DM 0 ; Function key F15
DM 0 ; Function key F16
DM 0 ; Function key F17 (3102 only)
DM 0 ; Function key F18
DM 0 ; Function key F19
DM 0 ; Function key F20
IF FUN KEYS and C3103 ; Conditional #3
DM DBNLNE ; CE (Clear Entry) function key
SW PAUSE ; PAUSE function key
SW PRINT ; PRINT function key
SW HELP ; HELP function key
ENDIF ; End conditional #3

; Dummy routine to use when returning to caller with no changes
DUMMY: RET ; Return to caller with no changes

181
Cromemco CDS User's Manual
C. Unassembled Source Listings

SUBTL  Console Routines
    IF C3102  ; Conditional #4
    ; Console Initialization Routine for 3102 Terminal
CINIT:  LD  B,'9'
        JP  SEND,ESC
ENDIF  ; End conditional #4
        IF NOT C3102
            ; Conditional #5
        ENDIF
    ; [Dummy] Console Initialization Routine
CINIT  EQU  DUMMY  ; (Console baud rate already set before CDS booted)
ENDIF  ; End conditional #5

; Get Console Input Status
; Upon Exit:  A = -1 (PPM) and Z-flag is reset if char. is ready
;            A = 0 and Z-flag is set if character is not ready
;            C-flag is set if function key transmission is in progress
CSTAT:  IN  A,CSTATP
        AND  CEDA
        IF NOT FUN.KEYS
            ; Conditional #6
            LD  A,-1
            ; Character ready
            RET
        ENDIF
        IF FUN.KEYS
            ; Conditional #7
            JR  Z,CSTAT50
            ; Skip to check further if char. not ready
            LD  A,-1
            ; Character ready
            RET
CSTAT50: LD  A,(PPFLAG)
        ; Check whether or not in midst of
        AND  A
        ; Function key transmission to CDS
        RET  I
        ; Return if not with 1 and C-flags cleared
        SUB  A
        ; Clear A-reg, set Z-flag for char., not ready
        SCP  A
        ; Return C-flag set to indicate to CDS that
        RET
        ; Function key transmission is in progress
        ENDIF  ; End conditional #7

; Console Input Routine
; Upon Exit:  A contains the character read
;            Z-flag is reset to prevent indicating end of file
;           (Change routine to return Z-flag set ON! if you wish
;            to have a particular character indicate end of file.)
CIN:    CALL  CSTAT
        ; Get console-in status
        IN  A,CDATA
        ; Read the character
        AND  TPS
        ; Strip off parity bit
        IF NOT C3703
            ; Conditional #8
            RET
            ; Return with Z-flag reset
        ENDIF  ; End conditional #8

102
IF C3703
CP  CTELP
RST  HZ
PUSH  XP
LD  A,CTEQ
CALL  LIQT
POP  AF
AND  A
RET
ENDIF

; Conditional #9
; Check for control-P
; Return if any other character
; Save control-P for a moment,
; get select char, etc.; end
; output it to select the printer
; Restore the original control-P for return
; Reset Z-flag to avoid indicating EOF
; End conditional #9
IF FUN.KEYS
; Conditional #10
EJECT

; Special Console Input Routine Including Function Key Decoding
; Upon Exit: A contains the character read, either from the
; console or as a character in a function key string

CSPECIM: CALL CSTAT
JR EX,CSIN20
LD A,(FFFLAG) ; Check whether or not in midst of
AND A ; function key transmission to COM
JR EX,CSIN10 ; Skip if so to finish the transmission
CSIN20: CALL GETFUNC
JR EX,CSIN40 ; Return if it's a single byte
RET

CSIN40: LD HL,(FFPTR) ; Point to next byte to be passed to COM
CSIN45: LD A,-1 ; Non-zero means function-in-progress
LD A,(HL) ; Get the character being transmitted
PUSH AF ; Save character for a moment
INC HL ; Increment to point to next character
LD (FFPTR),HL ; Store pointer back
LD A,(HL) ; Get subsequent character and check
SUB -1 ; whether it's the end-of-transmission
JR EX,CSIN50 ; Return with character if not
LD (FFFLAG),A ; If end-of-transmission, zero progress flag
POP AF ; Restore the character and return

; Get either a function key or a single byte from the console
; Upon Exit: for a function keys:
; - Z-flag is set and HL points to start of definition
; for a single byte:
; - Z-flag is reset and A contains the character read

GETFUNC: CALL CIN
CP CTRLB
RET NZ ; Return if any other character
LD (FEBUFF),A ; Save the control-B in sequence buffer
LD (FEBUFF+1),A ; in first and second positions
CALL GETBYTE ; Get first byte of function key sequence
JR EX,GETPC30 ; Skip to get other chars, if 3101 function key
LD A,EX ; Set last byte of 4-byte sequence to make
LD (FEBUFF+1),A ; 3101 func. key look like 3101 func. key
CAL CALL ; Get second byte of 3102 func. key sequence
JR GETPC20 ; Skip to return if timeout
CALL CTRLB ; Check for control-B as second character
JR EX,GETPC40 ; Skip to do as block-send (don't echo CTRL-B)
LD A,CTRLB ; Prepare to echo control-B since function key
CALL GETPC ; Echo control-B as required by 3102 protocol
JR GETPC40 ; Skip to decode the function key
GETPC20: LD A,CTRLB
AND A ; Reset Z-flag to indicate single byte
RET
EJECT
GTFC30: CP CTRLB
RET R0
CALL CIN
LD (PERUFF+2),A
CALL CIN
LD (PERUFF+3),A
LD R1
CALL WAIT30RE
LD A,(PERUFF+2)
LD R0,A
IF C110Q
LD HL,BLEND
CP CTRLB
LD R0
RET R1
ENDIF
LD HL,FUNPV
LD A,(CPTFLAG)
AND R0
RET R1
LD HL,FUNVAL
LD DE,FUNADDR
LD A,(HL)
AND R0
RET R1
LD (CPTFLAG)
AND R0
JR L,GETFUNC
JR Z,GTFC70
INC HL
INC DE
INC DE
JR GTFC60
LD A,(HL)
JR DE,HL
DEC DE
LD A,(HL)
JR DE,HL
LD A,B
JR DE,HL
LD B,A
JR E,GTFC70
INC B
JR C,GETFUNC
CD
JR Z,GTFC60
RET
: Variables needed for function key routines
CPTFLAG: DB 0
: Function-transmission-in-progress flag
PPPTR: DB 2
: Pointer to current byte of pre-programmed function key
PERUFF: DB 3,0,0,0,0,-1
: Buffer for function key sequence

185
EJECT

; Table of function key values transmitted
; Note: When assembled, the number of entries in this table
; MUST NOT exceed the number in the FUNCAD0R table.

FUNCVAL:DB 76H ; Function key F1 (3102 and 3101)
        DB 71H ; Function key F2
        DB 72H ; Function key F3
        DB 73H ; Function key F4
        DB 74H ; Function key F5
        DB 75H ; Function key F6
        DB 76H ; Function key F7
        DB 77H ; Function key F8
        DB 78H ; Function key F9
        DB 79H ; Function key F10
        DB 7AH ; Function key F11
        DB 7BH ; Function key F12
        DB 7CH ; Function key F13
        DB 7DH ; Function key F14
        DB 7EH ; Function key F15
        DB 7FH ; Function key F16
        DB 6FH ; Function key F17 (3102 only)
        DB 6EH ; Function key F18
        DB 6DH ; Function key F19
        DB 6CH ; Function key F20
        IF NOT C3102 ; Conditional $10B
           DB 0 ; End of table
        ENDIF
        IF C3102 ; Conditional $12C
           DB 5EH ; CE (Clear Entry) function key (3102 only)
           DB 5FH ; PAGE function key (3102 only)
           DB 6AH ; PRINT function key (3102 only)
           DB 6BH ; HELP function key (3102 only)
           DB 0 ; End of table
        ENDIF

; Character sequences transmitted for special-purpose function keys

DELLINE:DB CTRLV,-1 ; Delete line (control-V)
PAGES: DB CTRLV,-1 ; Page down (control-D)
PRINT: DB CTRLS,-1 ; Print Console output (control-P)
HELP: DB CTRLU,-1 ; Help key (control-U)
BDELETE: DB CTRLB,CTRLA,-1 ; Block and end sequence
        ENDIF
        IF NOT C310C ; Conditional $10C
        ENDIF

186
IF C3102 or FUN.Keys ; Conditional #11 EJECT

; Ask terminal for a function key byte by sending a control-B (3102 only)
; Upon Exit: Z-flag is reset if function key was pressed
; Z-flag is set if timeout occurred before subsequent char.

AJEKBYTE:
LD A,CCTRL
CALL COUT
; Output a control-B to console
; to request a function key byte
; Fall through to get function key byte;

; Get a function key byte
; Upon Exit: Z-flag is reset if function key was pressed
; Z-flag is set if timeout occurred before subsequent char.

GETBYTE:
LD EL,FUNCTIME
; Get counter for time between characters
; Get console-in status
JP NL,CIN
; Non-zero means char. is ready; get it and
; return with Z-flag reset (CIN returns
; flag this way) to indicate function key
DEC L
; If still no character, count down
JR NS,GTFB20
DEC H
;/
JR NS,GTFB20
;/
RET
; Return with Z-flag set to indicate
; no character within timeout

; Delay routine to wait for approx. 30 msec.
; Registers: EL registers are not preserved

WAITMNS:
LD EL,8000 ; Load counter for time of 30 msec.
WAIT20: DEC L ; Total time approx. = (no. in H) \times 1 msec.
JR NS,WAIT20
;/
DEC H
;/
JR NS,WAIT20
;/

; Equate needed for GETBYTE

FUNCTIME EQU 1400 ; Maximum time allowable between characters
; of function key sequence (total time in
; approx. 2; usec. times value shown)

ENDIF ; End condition of #11

187
; EJECT
; Get Console Output Status
; Upon Exit: A = -1 (FPR) and 1-Flag is reset if ready for char.
; A = 0 and 2-Flag is set if not ready for character

CRDT: IN A, CRTATP ; Get console-out status
AND CTBE ; Check console TBE flag
RET I ; Console ready not for character
LD A,-1 ; Console ready for character
RET

; Console Output Routine
; Upon Entry: A contains the character to be output

COUT: PUSH AF ; Save character for a moment
COUT30: CALL CRDT ; Get console-out status
JR I, COUT30 ; Zero means console busy
POP AF ; Restore character
OUT CDATA, A ; Output the character
IF NULLS=0 ; Conditional #12
RET ; End of procedure
ENDIF ; End of conditional #12
IF NULLE=0 ; Conditional #13
CP LF ; Check for end of line
RET NE ; Return if not line feed character
LD A, NULLE+1 ; If LF, get number of nulls
COUT50: DSE A ; Check for 0 nulls at top of loop
RET I ; Return if all nulls output
PUSH AF ; Save nulls counter
SUB A ; Print a single null
CALL COUT ; Character (recursive)
PUSH AF ; Restore nulls counter
JR COUT50 ; Loop to print next null
ENDIF ; End of conditional #13

188
EJECT

; Set special console command including Cursor Addressing
; Upon Entry: for cursor addressing
; E contains cursor row in the range 1-24
; D contains cursor column in the range 1-80
; for special console commands
; S = 0
; D contains the special command number
; SL contains pointer to string for some commands
; A contains additional information for some commands

CSET:  LD CA ; Save the additional information
        LD A,E ; Check whether it's a special
        AND A ; or cursor-address command
        JR D,COSCOM ; Skip to do special command
        IF C3102 or C3103 ; Conditional #14
            LD B,'P' ; Second special character is "P"
        ENDIF ; End conditional #14
        IF ADM3A ; Conditional #15
            LD B,'=' ; Second special character is "=
        ENDIF ; End conditional #15
        CALL SENDESC ; Send escape-sequence for cursor addressing
        ADD E ; Add incoming row number to the offset
        CALL COST ; Output so-created character
        ADD D ; Add incoming column number to the offset
        JP COST ; Output so-created character & return

; Print escape-sequence on console
; Upon Entry: B contains command character

SENDESC:  LD A,ESC ; Send an escape character to
        CALL COST ; console to start sequence
        LD A,E ; Retrieve the command character
        JP COST ; Print the command char. & return
        IF C3102 ; Conditional #16

; Print escape-dot sequence on console
; Upon Entry: B contains command character

SEND,ESC:  LD A,ESC ; Send an escape character to
        CALL COST ; console to start sequence
        LD A,'.' ; Send a dot character to console
        CALL COST ; an second specifier of sequence
        LD A,E ; Retrieve the command character
        JP COST ; Print the command char. & return
        ENDIF ; End conditional #16

189
A/VECT

; Set special console command (part of CSEF)
; Upon Entry: D contains the special command number
; HL contains pointer to string for some commands
; C contains additional information for some commands

CCOMMAND LD A,D ; Get number of special command
CP SC,MAX ; check for illegal special
RET BC ; command and return if so
PUSH HL ; Save address pointer
LD HL,SC,TBL ; Point to table of special command values
ADD L ; Add offset in A to table address in HL
LD L,A ;
JR RC,CSCMD30 ;
INC B ;

CSCMD30:LD A,(HL) ; Get the command from the table
POP HL ; Restore address pointer
AND A ; 0 means command not implemented
RET E ; Return if command not implemented
JR ADWA ; Conditional 18
JP COUT ; Output the special character
ENDIF ; End conditional 18
IF C3102 or C3101 ; Conditional 18
LD B,A ; Save the special character
JP P,SERDESC ; Send escape-sequence to console & return
AND 7FH ; Strip off top bit
LD B,A ; Multiply by 5
ADD B ;
ADD B ;
PUSH HL ; Save address pointer
LD HL,ROUTTBBL ; Point to routine table
ADD L ; Add displacement to HL
LD L,A ;
JR RC,CSCMD50 ;
INC B ;

CSCMD50:LD E,(HL) ; Get routine address into DE-reg.
INC HL ;
LD D,(HL) ;
INC HL ;
LD A,(HL) ; Get mask into A-reg.
PUSH HL ; Get address pointer
PUSH DE ; Put routine address on stack
RET ; Execute routine

CPFLAGS:EN 1 ; Cursor pad enable/disable special command flag
; (1 = CDSU pre-programmed function keys;
; 0 = terminal's actual function key sequence)
ENDIF ; End conditional 18

190
### IF C3102 or C3101

<table>
<thead>
<tr>
<th>SC.TAB:</th>
<th></th>
<th>0</th>
<th>Clear screen</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 'E'</td>
<td></td>
<td>1</td>
<td>Home cursor</td>
</tr>
<tr>
<td>DB 'D'</td>
<td></td>
<td>2</td>
<td>Back space</td>
</tr>
<tr>
<td>DB 'C'</td>
<td></td>
<td>3</td>
<td>Forward space</td>
</tr>
<tr>
<td>DB 'A'</td>
<td></td>
<td>4</td>
<td>Move cursor up</td>
</tr>
<tr>
<td>DB 'B'</td>
<td></td>
<td>5</td>
<td>Move cursor down</td>
</tr>
<tr>
<td>DB 'E'</td>
<td></td>
<td>6</td>
<td>Clear to EOL</td>
</tr>
<tr>
<td>DB 'Q'</td>
<td></td>
<td>7</td>
<td>Clear to BOB</td>
</tr>
</tbody>
</table>

#### Conditional #19A

<table>
<thead>
<tr>
<th>IF C3102</th>
<th></th>
<th>8</th>
<th>High light</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 84H</td>
<td></td>
<td>9</td>
<td>Low light</td>
</tr>
<tr>
<td>DB 86H</td>
<td></td>
<td>10</td>
<td>Medium light</td>
</tr>
</tbody>
</table>

#### End conditional #19A

<table>
<thead>
<tr>
<th>IF C3101</th>
<th></th>
<th>Condition</th>
<th>Low light</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 0</td>
<td></td>
<td>8</td>
<td>High light</td>
</tr>
<tr>
<td>DB 0</td>
<td></td>
<td>9</td>
<td>Low light</td>
</tr>
<tr>
<td>DB 0</td>
<td></td>
<td>10</td>
<td>Medium light</td>
</tr>
</tbody>
</table>

#### Conditional #19B

<table>
<thead>
<tr>
<th>ENDIF</th>
<th></th>
<th>End conditional #19B</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 'E'</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>DB 'C'</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>DB 80H</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>DB 81H</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>DB 'F'</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>DB 'T'</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>DB 82H</td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>DB 83H</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>DB 'I'</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>DB 'T'</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>DB 'O'</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>DB 'D'</td>
<td></td>
<td>22</td>
</tr>
</tbody>
</table>

#### Conditional #19C

<table>
<thead>
<tr>
<th>IF C3102</th>
<th></th>
<th>Condition</th>
<th>Insert character on</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 'O'</td>
<td></td>
<td>23</td>
<td>Insert character</td>
</tr>
<tr>
<td>DB 'M'</td>
<td></td>
<td>24</td>
<td>Delete line</td>
</tr>
<tr>
<td>DB 'L'</td>
<td></td>
<td>25</td>
<td>Insert line</td>
</tr>
</tbody>
</table>

#### Conditional #19D

<table>
<thead>
<tr>
<th>ENDIF</th>
<th></th>
<th>End conditional #19D</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 0</td>
<td></td>
<td>23</td>
</tr>
<tr>
<td>DB 0</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>DB 0</td>
<td></td>
<td>25</td>
</tr>
</tbody>
</table>

#### Conditional #19E

<table>
<thead>
<tr>
<th>IF C3103</th>
<th></th>
<th>Condition</th>
<th>Format on</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 'X'</td>
<td></td>
<td>26</td>
<td>Format on</td>
</tr>
<tr>
<td>DB 'E'</td>
<td></td>
<td>27</td>
<td>Format off</td>
</tr>
</tbody>
</table>

#### Conditional #19F

<table>
<thead>
<tr>
<th>IF C3102</th>
<th></th>
<th>Condition</th>
<th>Reverse on</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB 87H</td>
<td></td>
<td>28</td>
<td>Reverse on</td>
</tr>
<tr>
<td>DB 88H</td>
<td></td>
<td>29</td>
<td>Reverse off</td>
</tr>
<tr>
<td>DB 89H</td>
<td></td>
<td>30</td>
<td>Underline on</td>
</tr>
<tr>
<td>DB 8AH</td>
<td></td>
<td>31</td>
<td>Underline off</td>
</tr>
<tr>
<td>DB 'L'</td>
<td></td>
<td>32</td>
<td>Display message on</td>
</tr>
<tr>
<td>DB 'P'</td>
<td></td>
<td>33</td>
<td>Display message off</td>
</tr>
<tr>
<td>DB 88H</td>
<td></td>
<td>34</td>
<td>CPU message deposit</td>
</tr>
<tr>
<td>DB 'L'</td>
<td></td>
<td>35</td>
<td>Insert character off</td>
</tr>
<tr>
<td>DB 'B'</td>
<td></td>
<td>36</td>
<td>Graphics mode on</td>
</tr>
<tr>
<td>DB 'D'</td>
<td></td>
<td>37</td>
<td>Graphics mode off</td>
</tr>
</tbody>
</table>

---

191
DB 'z' ; 38 - Cursor on (toggle in 3102)
DB 'z' ; 39 - Cursor off (toggle in 3102)
DB 'q' ; 40 - Memory lock on
DB 'h' ; 41 - Memory lock off
DB 0CH ; 42 - Line lock
DB $DH ; 43 - Line unlock
DB $DF ; 44 - Read character at cursor
DB '8' ; 45 - Alarm on
DB '9' ; 46 - Alarm off
ENDIF ; End conditional $19E
SC.MAN EQU 0-SC.TBL ; Length of table
ENDIF ; End conditional $19
IF ADMJA ; Conditional $20
EJECT

; Special command table for ADM-3A terminals
SC.TBL: DB CTRLI ; 0 - Clear screen
      DB CTRL.UP  ; 1 - Home cursor
      DB BACK ; 2 - Back space
      DB FORMF ; 3 - Forward space
      DB VT ; 4 - Move cursor up
      DB LF ; 5 - Move cursor down
      DB 0 ; 6 - Clear to EOL
      DB 0 ; 7 - Clear to POS
      DB 0 ; 8 - High light
      DB 0 ; 9 - Low light
      DB 0 ; 10 - Medium light
      DB CTRLN ; 11 - Enable keyboard
      DB CTRLG ; 12 - Disable keyboard
SC.MAX EQU $-SC.TBL ; Length of table
ENDIF ; End conditional $20

193
IF C3102 or C3101 ; Conditional #21

EJECT

; Routine address table for special console commands
; Note: When assembled, the number of entries in this table
; MUST equal the number of entries in SCVMH with bit 7 set.

ROUTELOW CURSEPAD ; 80H - Enable cursor pad
DB 1
DB CURSEPAD
DB 0
DB SETATTR ; 82H - Begin blinking
DB RESATTR ; 83H - End blinking
DB BLINK
DB BLINK

IF C3102 ; Conditional #21A
DB RESATTR ; 84H - High light (normal)
DB HALFINTS
DB SETATTR ; 85H - Low light
DB RESATTR ; 86H - Medium light
DB SETATTR ; 87H - Reverse on
DB REVERSE
DB RESATTR ; 88H - Reverse off
DB REVERSE
DB UNDERLINE
DB RESATTR ; 89H - Underline on
DB UNDERLINE
DB RESATTR ; 8AH - Underline off
DB CFUNCG ; 8BH - CPU message deposit
DB 0
DB LINELock
DB LINELock ; 8CH - Line lock
DB "c"
DB UNDRLINE
DB "a"
DB ROCURS ; 8EH - Read character at cursor
DB "G"
ENDP ; End conditional #21A

; Equates and variable needed for 3102 and 3101 special command routines
HALFINTS EQU "0" ; Half-intensity attribute bit mask
BLINK EQU "1" ; Blinking-fade attribute bit mask
REVERSE EQU "4" ; Reverse-video attribute bit mask
UNDERLINE EQU "5" ; Underline attribute bit mask

AFLAG: DB 0 ; Attribute-set flag byte

194
CROMEMCO CDOS User's Manual
C. Unassembled Source Listings

; Enable/disable function key transmit-through (cursor pad on/off)
; Upon Entry: A contains 0 to transmit actual function key sequence and
; non-zero to transmit CDOS pre-programmed function keys
CUSEFPAD:LD (CFPFLAG),A ; Store value in cursor pad flag & return

; Set terminal attribute at present cursor position
; Upon Entry: A contains the bit mask for the attribute to be set
; (blinking field - 3102 or 3101 terminal)
; (half intensity, reverse video, & underline - 3102 only)
SEATTR:LD HL,ATFLAG ; Point to attributes-set flag byte
OR (HL) ; Combine old attributes with new in A-reg.
JR SENDATTR ; Send attributes to the terminal

; Reset terminal attribute at present cursor position (3102 only)
; Upon Entry: A contains the bit mask for the attribute to be set
; (blinking field - 3102 or 3101 terminal)
; (half intensity, reverse video, & underline - 3102 only)
RESATTR: CPL ; Invert all incoming bits
LD HL,ATFLAG ; Point to attributes-set flag byte
AND (HL) ; Use mask in A-reg. to turn off old attribute
JR Send through to send attributes to terminal:

; Send sequence to terminal to finish setting/resetting attributes
; Upon Entry: A contains byte with appropriate attribute bits set/reset
SENDATTR:LD (HL),A ; Save byte specifying attributes set
AND A ; Normal-video (3102) or end-blinking (3101)
JP N7,SENDESC ; Check whether all attributes are reset
LD B,17 ; Start-blinking special command to 3102 & 3101
IF NOT C1105 ; Conditional #21B
; Start-blinking special command to 3102 & 3101
JP SENDESC ; Conditional #21B
; Send escape-sequence to console & return
SENDIP ; End conditional #21B
IF C1102 ; Conditional #21C
CP BLINK ; Check for blinking-field attribute bit mask
JP Z,SENDESC ; Skip if so to send special command & return
LD B,51 ; Set-visual-attributes special command to 3102
CALL SENDESC ; Send escape-sequence to console
LD A,(ATFLAG) ; Get flag byte specifying attributes set
ADD #4 ; Convert attribute to appropriate ASCII
JP COUT ; Output so-created character & return

195
EJECT
; Send message to terminal buffer (CPU message deposit for 3102 only)
; Upon Entry: HL points to message to be printed terminated in a 0 or a CR

CPUMSG: LD B,'r' ; CPU-message-deposit special command to 3102
CALL SENDESC ; Send escape-sequence to console
AND A ; Get a character of the message
JR L:CPUM50 ; Skip if = 0, end of line indicator
CP ; Check for CR, end of line indicator
JR L:CPUM50 ; Skip if = 0, end of line indicator
CALL COUT ; Print the message character
INC HL ; Point to next message character
JR CPUM50 ; Skip to process next character
CPUM50: LD A,CTRS.RB ; Get terminating character for
JR COUT ; CPU-message-deposit & output it

; Lock/unlock a display line on terminal (3102 only)
; Upon Entry: A contains the command byte to lock/unlock the line
; C contains line number to be locked/unlocked (in range 1-24)
; 05
C contains number > 24 to unlock all display lines

LIMELOCK:
LD B,'A' ; Line-lock/unlock special command to 3102
LD A,C ; Get line number in C-reg.
CP 21 ; Check it for outside the range 1-24
JR NCL,LIMEL50 ; Skip if so to unlock all lines
CALL SENDESC ; Send escape-sequence to console.
LD A,IFN ; Load A-reg, with offset to generate line
AND C ; Add incoming line number to the offset
JP COUT ; Output so-created character & return

LIMEL50: LD B,'y' ; Unlock-all-lines special command to 3102
JP SENDESC ; Send escape-sequence to console & return

; Read character at present cursor position (3102 only)
; Upon Entry: A contains the command byte to read cursor character
; Upon Exit: A contains the character on the screen at the cursor position

ROCURS: LD CALL SENDESC ; Read-cursor-character special command to 3102
JP CIN ; Send escape-sequence to console
ENDIF ; Get the character to be returned
ENDIF ; End conditional #21

196
Cromemco CADOS User's Manual
C. Unassembled Source Listings

SUBTL Paper Tape or Card Reader Routines
IF S.READER or (NO,ROB=0) ; Conditional #22

; Reader Initialization Routine

INIT:  LD A,BRD,RT          ; Get reader baud rate and
        RRA,VA              ; output to baud rate port
        RET

; Get Reader Input Status
; Upon Exit: A = -1 (FYF) and 2-flag is reset if char. is ready
; A = 0 and 2-flag is set if character is not ready

RSTAT: LD HL,(RD.CTR)       ; Get timeout counter,
        DEC HL             ; decrement it,
        LD (RD.CTR),HL    ; and store it back
        LD A,R             ; Check to see whether reader timed
        OR L               ; out (zero means timeout)
        JR IX, RSTA50     ; Return as though character were received
        IN A               ; Get reader-in status
        AND RDA            ; Check reader. RDA flag
        RET                ; Character not ready
        X                   ; Character ready
        A                   ; 2-flag reset to show char. ready
        RET

; Reader Input Routine
; Upon Exit: A contains the character read
; 2-flag is reset if a character was read
; 2-flag is set if 20 sec. timeout occurred before
; character was read (indicating end of file)

RIN:   CALL RSTAT          ; Get reader-in status
        JR Z, BNR          ; Zero means reader busy
        LD HL,(RD.CTR)     ; Get timeout counter
        LD A,R             ; Check to see whether reader timed
        OR L               ; out (zero means timeout)
        LD A,CTRLS        ; Return the end-of-file character and
        RET                ; with 2-Flag set to indicate timeout
        HL,READTIME       ; Get value for timeout counter
        LD (RD.CTR),HL     ; Re-initialize the counter since no timeout
        IN A               ; Read the character
        RET                ; Return with 2-flag reset to indicate char.

READTIME EQU 65536         ; Timeout value for reader (total time is
                          ; approx. 300 usec. times value shown)
RD.CTR: DW READTIME        ; Timeout counter storage
ELSE
        BNL                     ; Else conditional #22

RINIT EQU DUMMY            ; If no reader is present, use console
EZSTK EQU CTRST           ; routine and consider it the case of a
EZTIP EQU CIN             ; teletype with paper tape reader connected
; End conditional #22

197
SUBTL Paper Tape Punch Routines  
IF B.PUNCH or (NO,PUNI>0) ; Conditional #23  

; Punch Initialization Routine  

Initializ  
LD  
A, PUNI, B, RT  
; Get punch baud rate and output to baud rate port  
OUT  
PBAUD, A  

; Get Punch Output Status  
; Upon Exit: A = -1 (FFH) and Z-flag is reset if ready for char.  
; A = 0 and Z-flag is set if not ready for character  
PRT:  
IN  
A, PSTATP  
AND  
PTE  
RET  
; Get punch-out status  
; Check punch TIE flag  
; Punch not ready for character  
; Punch ready for character  

; Punch Output Routine  
; Upon Entry: A contains the character to be output  

OUTPUT:  
PUSH  
AF  
; Save character for a moment  
CALL  
PDRY  
; Get punch-out status  
; Zero means punch busy  
POP  
AF  
; Restore character  
OUT  
PDA2A, A  
; Output the character  
RET  
ELSE  
; Else conditional #23  

FINIT EQU  
DUMMY  
; If no punch is present, use console  
PDRY EQU  
CRDY  
; routines and consider it the case of a teletype with paper tape punch connected  
POUT EQU  
COUT  
; End conditional #23  

198
Cromemco CDOS User's Manual
C. Unassembled Source Listings

SUBTL: List Device Routines
IF C3703 or C3779 ; Conditional #24
ENDC

; [Dummy] List Device Initialization Routine
LINIT RQU DUMMY ; (TUART is already initialized by CDOS upon booting)

; Parallel Printer (List Device) Output Status
; Upon Exit: A = 1 (FRF) and Z-flag is reset if ready for char.
; A = 0 and Z-flag is set if not ready for character
LIBOT: IN A,LIOTFP ; Get list-out status
CLL LIOTFP ; Check for negative-logic
LHO LHF ; printer-ready flag
RET ; Printer ready for character
LD A,-1 ; Printer ready for character

; Parallel Printer (List Device) Output Routine
; Upon Entry: A contains the character to be output
LIOUT: CP CTRLQ ; Check for printer-select character
JR Z,LIOUT40 ; if yes, skip & don't check for ready
PUSH AF ; Save character for a moment
LIOUT40: CALL LIOUT ; Get list-out status
JR Z,LIOUT30 ; Zero means printer busy
POP AF ; Restore character
IF C3779 ; Conditional #24A
AND TFM ; Skip off parity bit for comparison
CP FORM ; Check for form feed character
LD H,LIOUTC ; Point to line feeds counter before skipping
JR Z,LIOUT50 ; Skip to process form feed
ENDIF
LIOU50: SET LSTROB,A ; Data must be presented with strobe
OUT LDATA,A ; bit high prior to printing
RES LSTROB,A ; Low-to-high transition of strobe
OUT LDATA,A ; bit prints the character
SET LSTROB,A ; Strobe is set high upon this
OUT LDATA,A ; Instruction and character is printed
ENDIF ; End conditional #24A
IF NOT C3779 ; Conditional #25
RET
ENDIF

IF C3779 ; Conditional #26
CP LF or '7' ; Check for line feed characters
RET NZ ; Return if not line feed character
LD A,(HL) ; If LF, get number of lines already done
INC A ; Increment counter and
LD (HL),A ; store it back
CP PAGE,B12 ; Check for having reached maximum
RET NZ ; Return if still less than a full page
LD A A ; Zero out the line feeds counter
LD (HL),A ; if a full page of text has been reached
199
EJECT
LIOT50: LD A,PAGE.S18+1 ; Get number of lines to a page
SUB (86) ; Subtract number of lines already done
LIOT60: DEC A ; Check for 0 line feeds first
RET Z ; Return if all line feeds output
PUSH AF ; Save line feeds counter
LD A,LF ; Print a single line feed
CALL LIOUT ; character (recursive)
POP AF ; Restore line feeds counter
JR LIOT60 ; Loop to print next line feed
LF,CTR: DB 0 ; Counter of number of line feeds done
ENDIF ; End conditional $26
IF S.PRINTER ; Conditional $27
   JREXIT

; Serial Printer Initialization Routine
L2INIT: LD A,SER,BD,W ; Get serial printer baud rate
   OUT S&AUD,A ; and output to baud rate port

; Get Serial Printer Output Status
; Upon Exit: A = -1 (DRF) and I-flag is reset if ready for char.
; A = 0 and I-flag is set if not ready for character
L2EOY: IN A,STATP ; Get list-out status
   AND S&BE ; Check printer BUSY flag
   RET $ ; Printer not ready for character
   LD A,-1 ; Printer ready for character
   RET

; Serial Printer Output Routine
; Upon Entry: A contains the character to be output
L2OUT: PUSH AF ; Save character for a moment
   CALL LJEOY ; Get list-out status
   POP AF ; Store meaning printer busy
   OUT S&DATA,A ; Output the character
   RET
   RNDIP ; End conditional $27
IF (C3703 or C3779) and S.PRINTER and (NO.LST)?1 ; Conditional #28

EJECT

; Determine List Device Initialization Routine When Two Printers Used

LIMIT: LD A,(LOBYTE) ; Get I/O byte to determine which printer
AND "70.87 or "70.86 ; Check for bit combination 00 in high 2 bits
JP $2,LIMIT ; if found, use printer-1
CP "70.86 ; Check for bit combination 01 in high 2 bits
JR $2,LIMIT ; if found, use printer-2
RET ; All other combinations are ignored

; Determine List Device Ready Routine When Two Printers Used

LDRT: LD A,(LOBYTE) ; Get I/O byte to determine which printer
AND "70.87 or "70.86 ; Check for bit combination 00 in high 2 bits
JR $2,LDROY ; if found, use printer-1
CP "70.86 ; Check for bit combination 01 in high 2 bits
JR $2,LDROX ; if found, use printer-2
LD A,-1 ; No printer means always ready (S-flag reset)
RET ; All other combinations are ignored

; Determine List Device Output Routine When Two Printers Used

LOUT: LD B,A ; Save character to be output
LD A,(LOBYTE) ; Get I/O byte to determine which printer
AND "70.87 or "70.86 ; Check for bit combination 00 in high 2 bits
LD C,A ; Save 1/0 byte value for a moment
LD A,A ; Restore character to be output
JR $2,LOUT ; if 00 combination, use printer-1
LD A,C ; Retrieve 1/0 byte value
CP "70.86 ; Check for bit combination 01 in high 2 bits
LD A,A ; Restore character to be output
JR $2,LOUT ; if found, use printer-2
RET ; All other combinations are ignored
EJECT ; End conditional #28

IF (C3703 or C3779) and (NO.LST)?1 ; Conditional #29
EJECT

LIMRT EQU LIMIT ; Parallel printer initialize
LORIT EQU LOUT ; Parallel printer output a byte
EJECT ; End conditional #29

IF S.PRINTER and (NO.LST)?1 ; Conditional #30
EJECT

LIMRT EQU LIMIT ; Serial printer initialize
LORT EQU LOUT ; Serial printer output-ready
LOUT EQU LOUT ; Serial printer output a byte
EJECT ; End conditional #30

202
SUBTTL Clock Routines
IF C3102 ; Conditional #31

Start-Time Routine for Clock in 3102 Terminal

STRTCICL:LD B,SPEC ; Set-clock special command to 3102
    CALL SEDESC ; Send escape-sequence to console
    LD A,(HOUR) ; Get the hours value
    CALL PRTRSC ; Print hours to console in ASCII
    LD A,(MIN) ; Get the minutes value
    CALL PRTRSC ; Print minutes to console in ASCII
    LD A,(SEC) ; Get the seconds value
    JP PRTRSC ; Print seconds to console in ASCII

Read-Time Routine for Clock in 3102 Terminal

READICL:LD B,'O' ; Read-status-line special command to 3102
    CALL SEDESC ; Send escape-sequence to console
    CALL WAITTONS ; Give 3102 time to process special function
    CALL GETFSBYT ; Read first control-B and/or clear UART buffer
    RET ; Return if timeout; this terminal not a 3102
    CP CTRLB ; Check for control-B as second character
    RET NS ; Return if any other character
    LD B,27 ; Prepare to skip the next 27 characters
    CALL ADEFSBYT ; Request a function byte by sending a CTRL-B
    RET Z ; Return if timeout; unable to read the time
    GNE RCL30 ; Loop to bit-bucket the next 27 characters
    CALL GETWDO ; Read 2 hours digits
    RET Z ; Return if timeout; unable to read hours
    LD (HOUR),A ; Store the binary value for hours
    CALL ADEFSBYT ; Request and bit-bucket the "$" character
    RET Z ; Return if timeout
    CALL GETWDO ; Read 2 minutes digits
    RET Z ; Return if timeout; unable to read minutes
    LD (MIN),A ; Store the binary value for minutes
    CALL ADEFSBYT ; Request and bit-bucket the "$" character
    RET Z ; Return if timeout
    CALL GETWDO ; Read 2 seconds digits
    RET Z ; Return if timeout; unable to read seconds
    LD (SEC),A ; Store the binary value for seconds
    LD A,CTRLB ; Acknowledge the last character with
    JP COUT ; final CTRL-B as required by protocol

Get two ASCII characters from terminal
and combine then into a binary number returned in A-reg.
Upon Exit: A contains the binary byte
2-flag in set if timeout occurs before char.

GETWDO: CALL ADEFSBYT ; Request a function byte by sending CTRL-B
    RET Z ; Return if timeout occurred before byte
    AND DPR ; Strip to value between 0 and 9
    LD B,A ; Multiply first digit by 10
    ADD A ;
    ADD B ;
    ADD A ;

203
LD B, A    ; Save first digit for a moment
CALL AERBYTE    ; Request a second special function byte
RET    ; Return if timeout occurred before byte
AND 0FH    ; Strip to value between 0 and 9
ADD B    ; Combine first digit with second digit
LD B, A    ; and hold binary value in B-reg.
INC A    ; Reset I-flag to indicate no timeout
LD A, B    ; Retrieve binary value to be returned
EJECT

; Print binary number on console in ASCII
; Upon Entry: A contains the binary number to be sent to 3102 terminal
PRTASC: LD B,'0'-1 ; B-reg. will contain most sig. printable digit
PRTASC: INC B ; Increment to next printable digit
PRTASC: SUB 10 ; Compare value in A-reg. to 10
PRTASC: JR NC,PRTASC ; Loop to increment most sig. digit if A >= 10
PRTASC: ADD '0'+10 ; Convert remainder to ASCII if A < 10
PRTASC: LD C,A ; Save second digit for a moment
PRTASC: LD A,B ; Retrieve first digit
PRTASC: CALL COST ; and print it on console
PRTASC: LD A,C ; Retrieve second digit
PRTASC: CALL COST ; and print it also
PRTASC: ELSE ; Else conditional #31
PRTASC: ENDIF ; If no clock is present, use
PRTASC: DUMMY ; dummy routine to return
PRTASC: END

SUBTL Notes

; Note: The last assembled byte of this module MUST NOT be a Define
; Storage (OS or DEF3) pseudo-up to assure proper operation with CODGEN

205
### ASCII Characters

<table>
<thead>
<tr>
<th>ASCII Control Character</th>
<th>ASCII Character Code</th>
<th>ASCII Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ctrl-A</td>
<td>0041</td>
<td><code>A</code></td>
</tr>
<tr>
<td>Ctrl-B</td>
<td>0042</td>
<td><code>B</code></td>
</tr>
<tr>
<td>Ctrl-C</td>
<td>0043</td>
<td><code>C</code></td>
</tr>
<tr>
<td>Ctrl-D</td>
<td>0044</td>
<td><code>D</code></td>
</tr>
<tr>
<td>Ctrl-E</td>
<td>0045</td>
<td><code>E</code></td>
</tr>
<tr>
<td>Ctrl-F</td>
<td>0046</td>
<td><code>F</code></td>
</tr>
<tr>
<td>Ctrl-G</td>
<td>0047</td>
<td><code>G</code></td>
</tr>
<tr>
<td>Ctrl-H</td>
<td>0048</td>
<td><code>H</code></td>
</tr>
<tr>
<td>Ctrl-I</td>
<td>0049</td>
<td><code>I</code></td>
</tr>
<tr>
<td>Ctrl-J</td>
<td>004A</td>
<td><code>J</code></td>
</tr>
<tr>
<td>Ctrl-K</td>
<td>004B</td>
<td><code>K</code></td>
</tr>
<tr>
<td>Ctrl-L</td>
<td>004C</td>
<td><code>L</code></td>
</tr>
<tr>
<td>Ctrl-M</td>
<td>004D</td>
<td><code>M</code></td>
</tr>
<tr>
<td>Ctrl-N</td>
<td>004E</td>
<td><code>N</code></td>
</tr>
<tr>
<td>Ctrl-O</td>
<td>004F</td>
<td><code>O</code></td>
</tr>
<tr>
<td>Ctrl-P</td>
<td>0050</td>
<td><code>P</code></td>
</tr>
<tr>
<td>Ctrl-Q</td>
<td>0051</td>
<td><code>Q</code></td>
</tr>
<tr>
<td>Ctrl-R</td>
<td>0052</td>
<td><code>R</code></td>
</tr>
<tr>
<td>Ctrl-S</td>
<td>0053</td>
<td><code>S</code></td>
</tr>
<tr>
<td>Ctrl-T</td>
<td>0054</td>
<td><code>T</code></td>
</tr>
<tr>
<td>Ctrl-U</td>
<td>0055</td>
<td><code>U</code></td>
</tr>
<tr>
<td>Ctrl-V</td>
<td>0056</td>
<td><code>V</code></td>
</tr>
<tr>
<td>Ctrl-W</td>
<td>0057</td>
<td><code>W</code></td>
</tr>
<tr>
<td>Ctrl-X</td>
<td>0058</td>
<td><code>X</code></td>
</tr>
<tr>
<td>Ctrl-Y</td>
<td>0059</td>
<td><code>Y</code></td>
</tr>
<tr>
<td>Ctrl-Z</td>
<td>005A</td>
<td><code>Z</code></td>
</tr>
</tbody>
</table>

### Code Listing for DOS

```
0000: 0053 LDA #0000
0002: 0054 ORL #0000
0004: 0055 PLA
0006: 0056 SDA
0008: 0057 SAA
000A: 0058 STA #0000
000C: 0059 LDA #0000
000E: 005A ORL #0000
0010: 005B PLA
0012: 005C SDA
0014: 005D SAA
0016: 005E STA #0000
```

---

**Note:**
- The code listing is for demonstrating ASCII characters and their control codes in a micro assembler context for DOS.
- This listing is an example of how to represent and manipulate ASCII characters using assembly language instructions.
I/O Device Drivers for Codos

Device Port Assignments, Status Bits, and Baud Rates

```assembly
0076  ; I/O device port assignments and status bits
0080
(0000) 0081 CSTAPTP EQU 0 ; Console status port (input)
(0001) 0082 CUAYA EQU CSTAPTP+1 ; Console data port (input/output)
(0040) 0083 CRDA EQU 40H ; Console Receiver-data-available mask
(0080) 0084 CTTP EQU 80H ; Console Transmitter-buffer-empty mask
(0085)
(0090) 0086 REAMP EQU 20H ; Serial reader status port (input)
(0095)
(0100) 0090 PEASH EQU REAMP ; Serial reader baud rate port (output)
(0105)
(0110) 0091 PSTAPTP EQU 20H ; Serial punch status port (input)
(0115)
(0120) 0092 PRAUD EQU PSTAPTP ; Serial punch baud rate port (output)
(0125)
(0130) 0093 PDAAYA EQU PSTAPTP+1 ; Serial punch data port (output)
(0135)
(0140) 0094 PDBE EQU 80H ; Serial punch DBE bit mask
(0145)
(0150) 0095 LEAMP EQU 14H ; List device status port (input)
(0155)
(0160) 0096 LAATA EQU LEAMP ; List device data port (output)
(0165)
(0170) 0097 LDPB EQU 20H ; List device Ready-to-print bit mask
(0175)
(0180) 0098 LSTRB EQU 7 ; List device strobe bit
(0185)
(0190) 0099 SSTAPTP EQU 56H ; Serial printer status port (input)
(0195)
(0200) 0100 SRAAD EQU SSTAPTP ; Serial printer baud rate port (output)
(0205)
(0210) 0101 SDAAYA EQU SSTAPTP+1 ; Serial printer data port (output)
(0215)
(0220) 0102 STBE EQU 80H ; Serial printer DBE bit mask
(0225)
(0230)
(0235)
(0240)
(0245)
(0250)
(0255)
(0260)
(0265)
(0270)
(0275)
(0280)
(0285)
(0290)
(0295)
(0300)
(0305)
(0310)
(0315)
(0320)
(0325)
(0330)
(0335)
(0340)
(0345)
(0350)
(0355)
(0360)
(0365)
(0370)
(0375)
(0380)
(0385)
(0390)
(0395)
(0400)
(0405)
(0410)
(0415)
(0420)
(0425)
(0430)
(0435)
(0440)
(0445)
(0450)
(0455)
(0460)
(0465)
(0470)
(0475)
(0480)
(0485)
(0490)
(0495)
(0500)
(0505)
(0510)
(0515)
(0520)
(0525)
(0530)
(0535)
(0540)
(0545)
(0550)
(0555)
(0560)
(0565)
(0570)
(0575)
(0580)
(0585)
(0590)
(0595)
(0600)
(0605)
(0610)
(0615)
(0620)
(0625)
(0630)
(0635)
(0640)
(0645)
(0650)
(0655)
(0660)
(0665)
(0670)
(0675)
(0680)
(0685)
(0690)
(0695)
(0700)
(0705)
(0710)
(0715)
(0720)
(0725)
(0730)
(0735)
(0740)
(0745)
(0750)
(0755)
(0760)
(0765)
(0770)
(0775)
(0780)
(0785)
(0790)
(0795)
(0800)
(0805)
(0810)
(0815)
(0820)
(0825)
(0830)
(0835)
(0840)
(0845)
(0850)
(0855)
(0860)
(0865)
(0870)
(0875)
(0880)
(0885)
(0890)
(0895)
(0900)
(0905)
(0910)
(0915)
(0920)
(0925)
(0930)
(0935)
(0940)
(0945)
(0950)
(0955)
(0960)
(0965)
(0970)
(0975)
(0980)
(0985)
(0990)
(0995)
(1000)
(1005)
(1010)
(1015)
(1020)
(1025)
(1030)
(1035)
(1040)
(1045)
(1050)
(1055)
(1060)
(1065)
(1070)
(1075)
(1080)
(1085)
(1090)
(1095)
(1100)
(1105)
(1110)
(1115)
(1120)
(1125)
(1130)
(1135)
(1140)
(1145)
(1150)
(1155)
(1160)
(1165)
(1170)
(1175)
(1180)
(1185)
(1190)
(1195)
(1200)
(1205)
(1210)
(1215)
(1220)
(1225)

01H = 110 baud / 2 stop bits
02H = 150 baud / 1 stop bit
04H = 300 baud / 1 stop bit
08H = 1200 baud / 1 stop bit
10H = 2400 baud / 1 stop bit
20H = 4800 baud / 1 stop bit
40H = 9600 baud / 1 stop bit

Refer to TUART manual for other rate or stop bit configurations

The following baud rates were chosen from the table above:

(0001) 0120 RND.BD.RT EQU 01H ; Baud rate of serial reader
(0001) 0121 FUN.BD.RT EQU 01H ; Baud rate of serial punch
(0004) 0122 SEN.BD.RT EQU 84H ; Baud rate of serial printer
```
The following is a table of addresses used by C64 systems. Each item in the table is a memory location that may be read or written by the programmer. The first column lists the address of the memory location, the second column lists the name of the object located at that address, and the third column lists the value stored at that address. The fourth column lists the size of the object located at that address.
Reset &-flag to indicate single byte

A

AND

BYTE

214
<table>
<thead>
<tr>
<th>Variable needed for function key routines</th>
<th>Function-Transmission-in-progress flag</th>
<th>Function-Transmission-target type for CDS assembled</th>
<th>Buffet for function key sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$01A: 00$</td>
<td>$0132: 0000$</td>
<td>$0132: 0000$</td>
<td>$0132: 0000$</td>
</tr>
<tr>
<td>$01B: 00$</td>
<td>$0232: 0000$</td>
<td>$0232: 0000$</td>
<td>$0232: 0000$</td>
</tr>
<tr>
<td>$01C: 00$</td>
<td>$013J: 0000$</td>
<td>$013J: 0000$</td>
<td>$013J: 0000$</td>
</tr>
<tr>
<td>$01D: 00$</td>
<td>$023J: 0000$</td>
<td>$023J: 0000$</td>
<td>$023J: 0000$</td>
</tr>
<tr>
<td>$01E: 00$</td>
<td>$0134: 0000$</td>
<td>$0134: 0000$</td>
<td>$0134: 0000$</td>
</tr>
<tr>
<td>$01F: 00$</td>
<td>$0234: 0000$</td>
<td>$0234: 0000$</td>
<td>$0234: 0000$</td>
</tr>
</tbody>
</table>
0376 0377: Table of function key values transmitted
0378
0379: Note: When assembled, the number of entries in this table
0380: MUST NOT exceed the number of entries in the FUNCADR table.
0381
0382 0383 0384 0385 0386 0387 0388 0389 0390 0391 0392 0393 0394 0395 0396 0397 0398 0399 0400 0401 0402 0403 0404 0405 0406 0407 0408 0409 0410 0411 0412 0413 0414 0415 0416 0417 0418 0419
0122' 70 0123' 71 0124' 72 0125' 73 0126' 74 0127' 75 0128' 76 0129' 77 0130' 78 0131' 79 0132' 7A 0133' 7B 0134' 7C 0135' 7D 0136' 7E 0137' 7F 0138' 80 0139' 81 013A' 82 013B' 83 013C' 84 013D' 85 013E' 86 013F' 87 0140' 88 0141' 89 0142' 8A 0143' 8B 0144' 8C 0145' 8D 0146' 8E 0147' 8F 0148' 90 0149' 91 014A' 92 014B' 93 014C' 94 014D' 95 014E' 96 014F' 97 0150' 98 0151' 99 0152' A0 0153' A1 0154' A2 0155' A3 0156' A4 0157' A5 0158' A6 0159' A7 015A' A8 015B' A9 015C' AA 015D' AB 015E' AC 015F' AD 0160' AE 0161' AF 0162' BF 0163' CF 0164' DF 0165' EF 0166' FF
0382 FUNCVAL:DB 70H ; Function key F1 (3102 and 3261)
0383 DB 71H ; Function key F2
0384 DB 72H ; Function key F3
0385 DB 73H ; Function key F4
0386 DB 74H ; Function key F5
0387 DB 75H ; Function key F6
0388 DB 76H ; Function key F7
0389 DB 77H ; Function key F8
0390 DB 78H ; Function key F9
0391 DB 79H ; Function key F10
0392 DB 7AH ; Function key F11
0393 DB 7BH ; Function key F12
0394 DB 7CH ; Function key F13
0395 DB 7DH ; Function key F14
0396 DB 7EH ; Function key F15
0397 DB 7FH ; Function key F16 (3102 only)
0398 DB 6FH ; Function key F17 (3102 only)
0399 DB 6BH ; Function key F18
0400 DB 6CH ; Function key F19
0401 DB 5BH ; CE (Clear Entry) function key (3102 only)
0402 DB 5FH ; PAUSE function key (3102 only)
0403 DB 6AH ; PRINT function key (3102 only)
0404 DB 6BH ; HELP function key (3102 only)
0405 DB 0
0412
0413: Character sequences transmitted for special-purpose function keys
0414
0415 DELSINE:DB CTRLV,-1 ; Delete line (control-V)
0416 PASE:DB CTRL-,1 ; Pause console output (control-5)
0417 PRINTS:DB CTRL-,0 ; Print console output (control-P)
0418 HELPS:DB CTRL-,0 ; Help key (control-?)
0419 BLKSEND:DB CHRLA,CTRL-,1 ; Block-send sequence
I/O Device Drivers for DOS

Console Routines

0472  ; Get Console Output Status
0473  ; Open Exit: A = -1 (FPB) and Z-flag is set if ready for char.
0474  ; A = 0 and Z-flag is set if not ready for character
0475  ;
0476

0165' D800
0167' 680
0169' CB
016A' 3FF
016C' C9
0477  CRDT: IN A,CSTATP  ; Get console-out status
0478  AND CTBE  ; Check console TBE flag
0479  RET Z  ; Console not ready for character
0480  LD A,-1  ; Console ready for character
0481  RET

0482
0483
0484  ; Console Output Routine
0485  ; Upon Entry: A contains the character to be output
0486
0487  ; Save character for a moment
0487  COUT: PUSH AF  ; Save character for a moment
0488  COUT30: CALL CRDT  ; Get console-out status
0489  JR S,COUT30  ; Zero means console busy
0490  POP AF  ; Restore character
0491  OUT CDATA,A  ; Output the character
CROMEMCO 800 Macro Assembler Version 0.07
May 22, 1981 11:23:16
Page 001

I/O Device Drivers for CDOES

Console Routines

0560
0561 ; Set special console command (part of CSEF)
0562 ; Upon Entry: $ contains the special command number
0563 ; $ contains pointer to string for some commands
0564 ; C contains additional information for some commands
0565
01A4' 7A 0566 CSCOMM:LD A,D ; Get number of special command
01A5' F32F 0567 CP SC,MAX ; Check for illegal special
01A7' 00 0568 RET NC ; command and return if so
01A8' 65 0569 PUSH HL ; Save address pointer
01A9' 21B001' 0570 LD HL,SC.YBL ; Point to table of special command values
01AC' 85 0571 ADD L ; Add offset in A to table address in HL
01AE' 6F 0572 LD L,A ;
01AF' 3001 0573 JR NC,CSYMD30 ;
01B0' 24 0574 INC H ;
01B1' 7E 0575 CSYMD30:LD A,(HL) ; Get the command from the table
01B2' E1 0576 POP HL ; Restore address pointer
01B3' A7 0577 ADD A ; Zero means command not implemented
01B4' 08 0578 RET I ; Return if command not implemented
01B5' 47 0583 ; Save the special character
01B6' F28001' 0584 JP P,SENDESC ; Send escape-sequence to console & return
01B8' 267F 0585 ANI ZF ; Strip off top bit
01B8' 47 0586 LD B,A ; Multiply by 3
01C3' 00 0587 ADD H ;
01C3' 80 0588 ADD H ;
01C6' 65 0589 PUSH HL ; Save address pointer
01C8' 21FF01' 0590 LD HL,ROUTYBL ; Point to routine table
01C9' 95 0591 ADD L ; Add displacement to HL
01CA' 6F 0592 LD L,A ;
01CE' 3001 0593 JR NC,CSYMD50 ;
01C7' 24 0594 INC H ;
01C7' 0E 0595 CSYMD50:LD E,(HL) ; Get routine address into DE-reg.
01CA' 23 0596 INC HL ;
01CB' 56 0597 LD D,(HL) ;
01CA' 23 0598 INC HL ;
01CB' 7E 0599 LD A,(HL) ; Get mask into A-reg.
01CC' E1 0600 POP HL ; Get address pointer
01CD' D5 0601 POP DE ; Put routine address on stack
01CE' C9 0602 RET ; Execute routine
0603
0604
0605
01CF' 01 0606 CPFLAG: DB 1 ; Cursor pad enable/disable special command flag
0607 ; (1 = CDOES pre-programmed function key;
0608 ; 0 = terminal's actual function key sequence)
CROMEMCO $80 Macro Assembler version 03.07
I/O Device Drivers for CCGS

Console Routine:

0702 0703 0704 0705 0706 0707
0708 0709 0710 0711 0712 0713
0714 0715 0716 0717 0718 0719
0720 0721 0722 0723 0724 0725
0726 0727 0728 0729 0730 0731
0732 0733 0734 0735 0736 0737
0738 0739 0740 0741 0742 0743
0744 0745 0746 0747 0748 0749
0750 0751 0752 0753 0754 0755

01FF 2002
0201 01 0202 2002
0204 00 0205 3102
0207 02 0208 3702
020A 02 020B 3702
020D 01 020E 3102
0210 02 0211 3702
0213 01 0214 3102
0216 10 0217 3702
0219 10 021A 3102
021C 20 021D 3702
021F 20 0220 5702
0222 00 0223 6F02
0225 3C 0226 6F02
0228 35 022B 8302
022B 47 0720

0710
0711
0712
0713
0714
0715
0716
0717
0718
0719
0720
0721
0722
0723
0724
0725
0726
0727
0728
0729
0730
0731
0732
0733
0734
0735
0736
0737
0738
0739

0740
0741
0742

0743
0744
0745
0746
0747
0748
0749
0750

(0001) 0744
(0002) 0745
(0010) 0746
(0020) 0747

HALFINTS EQU '0'
BLINK EQU '1'
REVERSE EQU '2'
UNDERLINE EQU '3'

022C 00 0750

ATFLAG; DB 0

; Routine address table for special console commands
; Note: When assembled, the number of entries in this table
; Must equal the number of entries in SC_TRL with bit 7 set.

; 80H - Enable cursor pad
; 81H - Disable cursor pad
; 82H - Begin blinking
; 83H - End blinking
; 84H - High light (normal)
; 85H - Low light
; 86H - Medium light
; 87H - Reverse on
; 88H - Reverse off
; 89H - Underline on
; 8AH - Underline off
; 8BH - CPU message deposit
; 8CH - Line lock
; 8DH - Line unlock
; 8EH - Read character at cursor

; Equates and variable needed for 3102 and 3101 special command routines

; Half-intensity attribute bit mask
; Blinking-field attribute bit mask
; Reverse-video attribute bit mask
; Underline attribute bit mask

; Attributer-set flag byte
Cromemco CDOS User's Manual
D. Assembled Source Listings
<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0936</td>
<td>(0058')</td>
<td>PINIT EQU DUMMY</td>
</tr>
<tr>
<td>0937</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0938</td>
<td>(0165')</td>
<td>PROY EQU CEDY</td>
</tr>
<tr>
<td>0939</td>
<td>(018D')</td>
<td>POUT EQU COUT</td>
</tr>
</tbody>
</table>

; If no punch is present, use console routine and consider it the case of a teletype with paper tape punch connected
D. Assembled Source Listings

Cromemco CDOS User's Manual
CROMEMCO IRS Macro Assembler version 03.07
I/O Device Drivers for CROS
List Device Routines

May 22, 1981 11:23:16

Page 0022

02

03

05

{0058'} 1088 LIMIT EQU LINIT ; Parallel printer initialize
{028A'} 1089 LCDY EQU LDIRY ; Parallel printer output-ready
{0289'} 1090 LCDX EQU LDIRX ; Parallel printer output 8 byte
<table>
<thead>
<tr>
<th>Address</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0308h 87</td>
<td>ADD A</td>
<td>1</td>
</tr>
<tr>
<td>0309h 87</td>
<td>ADD A</td>
<td>;</td>
</tr>
<tr>
<td>030Ah 80</td>
<td>ADD B</td>
<td>;</td>
</tr>
<tr>
<td>030Bh 87</td>
<td>ADD A</td>
<td>;</td>
</tr>
<tr>
<td>030Ch 47</td>
<td>LD B,A</td>
<td>;</td>
</tr>
<tr>
<td>0310h 1160</td>
<td>CALL ASEBYTE</td>
<td>;</td>
</tr>
<tr>
<td>0311h 1161</td>
<td>RET E</td>
<td>;</td>
</tr>
<tr>
<td>0312h 1162</td>
<td>AND OFF</td>
<td>;</td>
</tr>
</tbody>
</table>

5/0 Device Drivers for CDOS Clock Routines

May 22, 1981 11:23:16
CROMERCO 180 Macro Assembler version 03.07
I/O Device Drivers for CROS
Clock Routines

1169
1170 ; Print binary number on console in ASCII
1171 ; Upon Entry: A contains the binary number to be sent to 3102 terminal
1172
0318' 062F
031A' 04
0318' 660A
031D' 30F9
031F' 663A
0321' 4F
0322' 78
0323' C6D01'
0326' 79
0327' C36D01'
1173 PRTASC; LD B, '0'-1 ; B-reg will contain most sig. printable digit
1174 PRTA30; INC B ; Increment to next printable digit
1175 SUB 10 ; Compare value in A-reg. to 10
1176 JR NC,PRTA30 ; Loop to increment most. digit if A > 10
1177 ADD '0'+10 ; Convert remainder to ASCII if A < 10
1178 LD C,A ; Save second digit for a moment
1179 LD A,B ; Retrieve first digit
1180 CALL COUT ; and print it on console
1181 LD A,C ; Retrieve second digit
1182 JP COUT ; and print it also
Cromemco CDOS User's Manual
D. Assembled Source Listings

Cromemco Macro Assembler version 0.97
CROMEO.com (C) Copyright 1976 Cromemco

I/O Device Drivers for CDOS

Notes

1130 " BEGIN "

1131 " STORAGE (OS or NOS) preproc-up to assure proper operation with COBOL

032-1W (0000)

1135 " END "

Error Count 0

Program Length 032-1W (010)

Page 015
Cromemco CDOS User's Manual
D. Assembled Source Listings
Cromemco CDOS User’s Manual

Index

16FDC, 18, 65

4FDC, 18, 65

@ program, 11, 60

Abort, 92, 122
Adding different I/O device drivers to CDOS, 44
Addresses, 32
Alternate tracks, 66
Ambiguous file reference, 25, 169
ASCII definition, 22
ATTR intrinsic, 8, 20, 48
ATTRIBUTE, 51, 87
Attribute protection of files, 8, 48
Automatic startup and program execution, 38

Backup of disks, 12
Batch (@) utility, 11, 60
Bitmap, 104, 169
Buffer, 85, 105

CDOS, 1
CDOS prompt – definition, 5
CDOS simulator, 85
CDOSGEN, 1, 27
Check if allocated system call, 148
Clock Switch, 176
Close disk file, 108
Close disk file system call, 108
CNTRL-I, 73
CNTRL-C, 20, 66, 101, 103, 122, 161
CNTRL-E, 37, 101
CNTRL-G, 93
CNTRL-H, 102
CNTRL-I, 93
CNTRL-J, 93
CNTRL-L, 37
CNTRL-M, 37, 93
CNTRL-N, 37, 97
CNTRL-P, 12, 38, 58, 93, 100, 120, 150
CNTRL-R, 101
CNTRL-S, 12, 37, 58, 95, 96
CNTRL-T, 38, 58, 100
CNTRL-U, 37, 101
CNTRL-V, 12, 37, 101
CNTRL-W, 38, 58, 100
CNTRL-X, 37, 101

237
CROMEMCO CDOS User's Manual

Index

CNTRL-E, 93, 120
Cold bootstrap, 35
Command line buffer, 85
Command structure & syntax, 40
Compare files, 78
Concatenate files, 78
Console ready, 103
Control character usage, 12
Control characters, 12, 36
Control characters - console, 36
Control characters - printer, 37
CP/M - CDOS differences, 2
CP/M compatibility, 1
Create file system call, 114
CRT functions, 134
Current disk, 104, 106, 117
Current disk system call, 117
Current drive, 5, 36, 125
Current record, 146
Cursor - definition, 5

Data file, 171
Data-definition, 21
Date, 69, 83, 90, 136, 137
Date, setting of - STAT/D, 69
Default, 85
DEL, 102
Delete extents system call, 151
Delete file system call, 111
Deselect current disk system call, 104
Device drivers, 43
Device I/O, 83
Device names, 23
DIR, 7
DIREcory, 51
DIREcory command, 17
Directory entries, 71
Directory entry structure, 88
Directory listing alphabetical - STAT/A, 69
Directory of a disk, 7, 51, 88, 149
Disk, 105
Disk buffer, 86, 112, 113, 118
Disk cluster allocation map system call, 119
Disk drive configuration for CDOS, 28
Disk label, 90, 142
Disk label, writing of - STAT/L, 71
Disk log-in vector system call, 116
Disk organization, 17
Disk precautions, 3, 20
Disk specifications, 18
Disk specifier, 8, 170
Disk type specifiers, 18

238
Diskette - 3740, 64
Diskettes, 3
Display filenames - STAT/B, 72
Divide, 130
Divide integers system call, 130
Double width characters, 97
Drive selection, 36
Drivers - adding to CDOS, 44
Drivers - I/O device, 43
DUMP, 63
Dump file contents, 63

Editor, Screen, 81
Editor, Text, 82
Editors, 81
Eject disk system call, 132
ERA, 7
ERAsm, 53
Erase a file, 7
Erase all files on a disk - STAT/Z, 73
Erase files alphabetically - STAT/E, 70
Erase files from a disk, 53
Error messages, 159, 162, 165
ESC, 65, 103, 150
Extended file format, 89
Extents, 88, 151

PCB, 85, 107, 112
File, 171
File Area of a disk, 17, 19, 79, 170
File attributes, 141
File concatenation, 79
File control block, 85, 87, 126
File definition, 5, 21
File reference, 23, 24, 25, 171
Filename, 5, 171
Filename extension, 171
Find next entry system call, 110
Floppy disk access error messages, 159
Format disk, 64
Format name to PCB system call, 126
Function keys, 29, 30, 31, 135

Generating a new CDOS, 27
Get I/O byte system call, 98
Get master drive system call, 152
Get user-register pointer system call, 121
Get version number system call, 133
Glossary of terms and symbols, 169

239
Index

Hard disk, 28, 64, 66, 142
Hard disk access error messages, 162
Hard disk alternate tracks, 66
High Memory, 15, 16
Home drive system call, 131

I/O Byte, 45, 98, 99
I/O device drivers, 43, 44
INITialize, 9, 64
Initialize a disk, 9, 64
Input buffered line system call, 101
Interrupts, 91
Intrinsic commands, 6, 40, 47

Label, 71
Labeling a disk after initialization, 65
Link to program system call, 128
Linker, 84
List, 97
List directory system call, 149
Loading CDOS, 35
Logical record, 123, 125, 131
Low Memory, 15, 16

Master disk, 40
Master drive, 72, 105, 152
Master drive, setting of - STAT/M, 72
Memory, 6, 15, 27, 84
Modification of I/O device drivers, 44
Motors off system call, 144
Multiply, 129
Multiply integers system call, 129

Open disk file, 101, 126
Open disk file system call, 107
Operating system, 5

Port assignments, 83
Power-on Jump, 176
Primed registers, 83
Print buffered line system call, 100
Print text file, 80
Printer - 3355A, 43, 68, 98
Printer - 3703, 97
Printer drivers, 43
Program abort system call, 92
Punch, 96

240
Random access files, 80, 148
Read console, 93, 120
Read console with echo system call, 93
Read console without echo system call, 120
Read current record system call, 146
Read date system call, 137
Read disk label system call, 142
Read logical record system call, 123
Read next record system call, 112
Read reader system call, 95
Read time system call, 139
Reader, 95
REName, 8, 55
Rename file, 8, 115
Rename file system call, 115
Replacement characters, 25, 172
Reset CDOS & select master drive system call, 105
Reset switch, 13, 41
RETURN - definition, 6
RUBOut, 37, 102

Safeguarding your data, 12
SAVE, 57
Save memory contents on disk, 57
Screen editor, 81
Search directory system call, 109
Select current disk drive system call, 106
Set bottom of CDOS system call, 145
Set date system call, 136
Set disk buffer system call, 118
Set file attributes system call, 141
Set I/O byte system call, 99
Set options system call, 150
Set program return code system call, 140
Set special CRT function system call, 134
Set time system call, 138
Set user CNTXL-C abort system call, 122
Single file reference, 23, 172
Special CRT function, 134
Startup.cmd, 38, 62
STATUS, 11, 67
Status of system printout - STAT/S, 73
Status of system, brief - STAT/B, 69
Status of the system, 11, 67
Storage - definition, 6
Summary of CDOS system calls, 153, 154, 155, 156, 157
Switch settings, 175, 176
System Area of a disk, 16, 17, 19, 75, 79, 172
System call - check if allocated, 148
System call - close disk file, 108
System call - create file, 114
System call - current disk, 117

241
System call - delete extents, 151
System call - delete file, 111
System call - deselect current disk, 104
System call - disk cluster allocation map, 119
System call - disk log-in vector, 116
System call - divide integers, 130
System call - eject disk, 132
System call - find next entry, 110
System call - format name to PCB, 126
System call - get I/O byte, 98
System call - get master drive, 152
System call - get user-register pointer, 121
System call - get version number, 133
System call - home drive, 131
System call - input buffered line, 101
System call - link to program, 128
System call - list directory, 149
System call - motors off, 144
System call - multiply integers, 129
System call - open disk file, 107
System call - print buffered line, 100
System call - program abort, 92
System call - read console with echo, 93
System call - read console without echo, 120
System call - read current record, 146
System call - read date, 137
System call - read disk label, 142
System call - read logical record, 123
System call - read next record, 112
System call - read reader, 95
System call - read time, 139
System call - rename file, 115
System call - reset CDOS & select master drive, 105
System call - search directory, 109
System call - select current disk drive, 106
System call - set bottom of CDOS, 145
System call - set date, 136
System call - set disk buffer, 118
System call - set file attributes, 141
System call - set I/O byte, 99
System call - set options, 150
System call - set program return code, 140
System call - set special CRT function, 134
System call - set time, 138
System call - set user CTRL-C abort, 122
System call - test for console ready, 103
System call - update directory entry, 127
System call - write console, 94
System call - write current record, 147
System call - write list, 97
System call - write logical record, 125
System call - write next record, 113
System call - write punch, 96

242
System calls, 83, 92, 153, 154, 155, 156, 157
System error messages, 165
System startup, 35

Terminal - 3101, 135
Terminal - 3102, 12, 58, 138
Test for console ready system call, 103
Text editor, 82
Time, 73, 83, 138, 139
Time, setting of - STAT/T, 73
Transfer a file, 9
Transfer files and expand tabs, 79
Transfer files and strip rubouts, nulls, 79
Transfer files and verify, 79
Transfer read protected files, 79
Transferring files, 78
TYPE, 7, 58
Type a file, 7
Type-out of a file, 58

Underscore, 37, 102
Update directory entry system call, 127
User Area of memory, 15, 16, 173
Utility programs, 59

Warm start, 36, 84
Write console system call, 94
Write current record system call, 147
Write list system call, 97
Write logical record system call, 125
Write next record system call, 113
Write punch system call, 96
Write-protecting diskettes, 20
WRTSYS utility program, 17, 75

XFER, 9, 78

%80 registers, 83

[ ], 169

{ }, 169