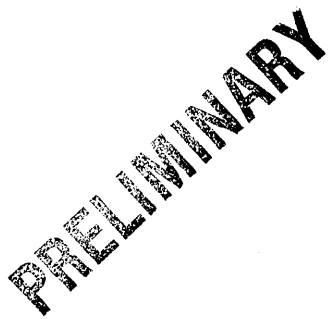
U N I B A S I C

U S E R ' S G U I D E

M i c r o C r a f t C o r p o r a t i o n

A S V E R S I O N

6 8 0 - 0 2 0 0 - 1 0 0



NOTICE

Micro Craft Corporation reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

MICRO CRAFT CORPORATION MAKES NO WARRANTIES, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL OR WITH RESPECT TO THE SOFTWARE DESCRIBED IN THIS MANUAL, ITS QUALITY, PERFORMANCE, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. MICRO CRAFT CORPORATION SOFTWARE IS SOLD OR LICENSED "AS IS." THE ENTIRE RISK AS TO ITS QUALITY AND PERFORMANCE IS WITH THE BUYER. SHOULD THE PROGRAMS PROVE DEFECTIVE FOLLOWING THEIR PURCHASE, THE BUYER (AND NOT MICRO CRAFT CORPORATION ITS DISTRIBUTOR, OR ITS RETAILER) ASSUMES THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION AND ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES. IN NO EVENT WILL MICRO CRAFT CORPORATION BE LIABLE FOR DIRECT, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT IN THE SOFTWARE, EVEN IF MICRO CRAFT CORPORATION HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATION OR EXCLUSTION MAY NOT APPLY TO YOU.

Micro Craft Corporation
4747 Irving Blvd.
Dallas, Texas 75247
(214)630-2562

Additional copies of this manual

may be ordered from your DEALER

by using the MICRO CRAFT part number 680-0200-100.

Ask your DEALER also for a

free brochure with a complete list of all

Micro Craft manuals and products

MICRO CRAFT CORPORATION
Customer Support Department
4747 Irving Blvd.
Dallas, Texas 75247

Table of Contents

I N T R O D U C T I O N

INTRODUCTION

We are Micro Craft Corporation, designers and manufacturers of the the DIMENSION 68000, the first and only universal microcomputer available today. To go with this powerful machine, we have commissioned the design of a UNIVERSAL BASIC, UNIBASIC (TM). The version that has been delivered with your machine is the AS Version, which has been designed to be source code compatible with programs written in APPLESOFT (TM) BASIC. UNIBASIC, AS Version, will run most APPLESOFT programs without change, however UNIBASIC has some very powerful extensions. The purpose of this manual is to explain the use of those extensions, and how to make the most of them to unleash the power of your DIMENSION 68000.

Welcome to the realm of DIMENSION computing.

SIMILARITIES TO APPLESOFT BASIC

The UNIBASIC BASIC language interpreter, by Micro Craft Corporation, is very similar to APPLESOFT (TM) BASIC, a product of Apple Computer, Inc. UNIBASIC also includes most of the standard APPLESOFT peeks and pokes, and it has some powerful extensions beyond the standard APPLESOFT. UNI-UNIBASIC also allows peeks and pokes to absolute memory locations using the APEEK and APOKE commands.

REQUIREMENTS

UNIBASIC requires:

256K of memory minimum:

- 60K for UNIBASIC
- 64K for graphics and text buffers
- 32K for CP/M-68K
- Additional memory to run programs

1 diskette drive

All Dimension 68000 systems are shipped from the factory with a minimum of 256K bytes of memory and 2 diskette drives, which meets the above requirements.

HOW UNIBASIC IS SHIPPED

UNIBASIC is shipped as a standard offering from Micro Craft Corporation, bundled at no additional charge when a Dimension 68000 system is purchased. UNIBASIC resides on the "SYSTEM 1" diskette.

The Dimension 68000 system is shipped with a "SYSTEM 1" diskette and a "SYSTEM 2" diskette. Micro Craft Corporation strongly advises the customer to copy the "SYSTEM 1" and the "SYSTEM 2" diskettes onto formatted blank diskettes, and then to operate off of the copies and not the originals that were shipped with the system. The process of making copies of valuable information on diskettes, etc., so as to safeguard the original information is called "backing-up". For a detailed discussion on making "back-ups", see "BACK-UP PROCEDURE" in the appendix.

PLEASE, if you have not already made working copies of your distribution diskettes, DO IT NOW !!

HOW TO USE THIS MANUAL

The Micro Craft Corporation UNIBASIC USER'S GUIDE contains information about UNIBASIC for the Dimension 68000 system. Also provided are chapters on converting previously written programs to UNIBASIC, handling disk files, and calling assembly language subroutines. Briefly:

This "Introduction" tells you about UNIBASIC and its special features, your system requirements, the diskettes that you receive with your Dimension 68000 system, and the conventions used in syntax notation. It also lists additional sources of information about programming in BASIC.

Chapter 1, UNIBASIC ON THE DIMENSION 68000 SYSTEM, tells you how to use UNIBASIC and explains some of the features of UNIBASIC.

Chapter 2, CONVERTING PROGRAMS TO UNIBASIC, describes the minor adjustments necessary to run BASIC programs in UNIBASIC.

Chapter 3, DISK FILE HANDLING, explains disk file handling procedures. This chapter can be read as an overview or used for reference for disk related operations while running UNIBASIC.

Chapter 4, UNIBASIC ASSEMBLY LANGUAGE SUBROUTINES, provides information about calling assembly language subroutines.

SYNTAX NOTATION

The following notation is used throughout this manual in descriptions of command and statement syntax:

[ ]	Square brackets indicate that the enclosed entry is optional.
< >	Angle brackets indicate user-entered data. When the angle brackets enclose lowercase text, the user must type in an entry defined by the text; for example, <filename>. When the angle brackets enclose uppercase text, the user must press the key named by the text; for example, <CR> for either the key marked 'Retrn' or the key marked 'Enter'.
{ }	Braces indicate that the user has a choice between two or more entries. At least one of the entries enclosed in braces must be chosen unless the entries are also enclosed in square brackets.
|	A vertical bar means "or": The material on the left or right of the bar may be specified.
. . .	Ellipses indicate that an entry can be repeated as many times as needed or desired.
CAPS	Capital letters indicate portions of statements or commands that must be entered, exactly as shown.

All other punctuation, such as commas, colons, slash marks, and equal signs, must be entered, exactly as shown. Spaces, or blanks are ignored, except as noted.

LEARNING MORE ABOUT BASIC

The manuals in this package provide complete reference information for UNIBASIC. they do not, however, teach you to write programs in BASIC. If you are new to BASIC or need help in learning to program, we suggest that you read one of the following books:

Apple Computer, Inc. "APPLESOFT BASIC Programming Reference Manual". Cupertino, California: Apple Computer, Inc., 1978.

Lien, David A. "The BASIC Handbook, 2nd Edition". San Diego, California: Compusoft Publishing, 1981.

Billings, Karen and Moursand, David. "Are You Computer Literate ?". Beaverton, Oregon: Dilithium Press, 1979.

Albrecht, Robert 1., Finkel, LeRoy, and Brown, Jerry. "BASIC". New York: Wiley Interscience, 2nd ed., 1978.

Coan, James. "Basic BASIC". Rochelle Park, N. J.: Hayden Book Company, 1978.

Dwyer, Thomas A. and Critchfield, Margot. "BASIC and the Personal Computer". Reading, Mass.: Addison Wesley Publishing Company, 1978.

Simon, David E. "BASIC From the Ground Up". Rochelle Park, N.J.: Hayden Book Company, 1978.

C H A P T E R 1

U N I B A S I C
O N T H E D I M E N S I O N 6 8 Ø Ø Ø S Y S T E M

CHAPTER 1

HOW TO RUN UNIBASIC

To run UNIBASIC on your Dimension system, enter the following command:

```
A> BASIC
```

The system will reply:

UniBasic Version AS-X.X (mm/dd/yy) Copyrighted YYYY by RD Software

Then the system will display the UNIBASIC prompt character

:

You can also specify a set of options in the command line. The syntax is:

```
A> BASIC [<filename>]
```

where <filename> follows the file naming convention described later in this chapter.

Notes: -The file naming convention for UNIBASIC is not the same as the file naming convention for CP/M-68K.

-At least one single space, or blank, is required between BASIC and "filename", as per the CP/M-68K commands convention. (See the CP/M-68K (TM) USER'S GUIDE.)

If <filename> is present, UNIBASIC procedes as if a RUN <filename> command were typed after initialization is complete. this allows UNIBASIC programs to be executed in batch mode using the CP/M-68K SUBMIT facility.

HOW TO EXIT UNIBASIC

To exit UNIBASIC and return to the CP/M-68K operating system level, enter the command:

```
:QUIT
```

which closes all files and then reloads CP/M-68K into memory without deleting any existing programs or data (i.e. performs a "warm start"). Note that a <Control-C>, when executed within UNIBASIC, returns to the UNIBASIC command level, not to CP/M-68K.

FILE NAMING CONVENTIONS

Filenames are a combination of the CP/M-68K and the APPLESOFT (TM) naming conventions. All UNIBASIC filenames consist of three parts:

- The FILENAME

- The FILETYPE

- The DRIVE SPECIFICATION

The FILENAME consists of from one to eight characters. The first character must be alphabetic. All of the rest of the characters may be either alphabetic or numeric.

The FILETYPE consists of a period (.) followed by from one to three characters. The characters may be either alphabetic or numeric.

The DRIVE SPECIFICATION consists of a comma (,), followed by a D, followed by either a 1, a 2, a 3, or a 4. The numbers 1, 2, 3, and 4 correspond to the drives A:, B:, C:, and D:. If no DRIVE SPECIFICATION is included, the system will use the CP/M default drive.

As an example, the standard CP/M-68K filename B:TEST.DAT would be TEST.DAT,D2 for UNIBASIC.

Under normal CP/M-68K conventions for compilers and interpreters, if no period (.) appears in the filename and if the filename is less than 9 characters long, a default filetype extension would be appended to the filename. If the filename is 9 characters long or more than 9 characters long, then the ninth and subsequent characters, up to a total of three characters, become the filetype extension. UNIBASIC does NOT append a default extension of .BAS to the LOAD <filename> command, to the the SAVE <filename> command, and to the RUN <filename> command.

For more information on the CP/M conventions for assigning filenames and assigning filetype extensions, see the CP/M-68K (TM) USER'S GUIDE.

CATALOG

UNIBASIC includes a CATALOG command, which displays the names of files on the CP/M default disk or disk specified. A detailed discussion of this command is described in the UNIBASIC REFERENCE MANUAL.

Syntax CATALOG [,Dx]

where x is the drive number desired (1, 2, 3, or 4), or IF NO drive is specified, then the CP/M-68K default disk drive will be used.

HIGH RESOLUTION GRAPHICS

The high resolution graphics shape table starts at POKE address 4000 decimal. To load the shape table requires either using the SHLOAD command to load a shape table that already exists, or using the POKE command and POKEing the values into the table area in memory, one at a time, until the table is loaded. To POKE in a shape table, start the POKEs at address 4000 decimal. To save the table, use the SHSAVE command. The default shape table size is 500 bytes. To set a different shape table size, use the SHSIZE command.

CHAPTER 2

CONVERTING PROGRAMS TO UNIBASIC

CHAPTER 2

This section in the manual is intended to show the differences between APPLESOFT (TM) BASIC and UNIBASIC. To obtain information about the differences between APPLESOFT (TM) BASIC and other BASICs, The reader is advised to refer to the "APPLESOFT BASIC Programming Reference Manual", published by APPLE COMPUTER, Inc.

MODE COMMAND

The Dimension 68000 system has some significant differences from the APPLE in the area of the video display. The APPLE, in the HIRES graphics mode has a total of 6 colors, while the Dimension has a total of 16 colors. The MODE values and command sequences are:

0 = Initialize video to 80 columns by 24 lines.
MODE#0: TEXT:

1 = Reset ERROR FLAG to OFF

Note: If the ERROR FLAG is ON, then when an attempt is made to plot a point outside of the screen window, an OUT OF RANGE ERROR message is given and execution is terminated.

2 = Set ERROR FLAG to ON

3 = Reset COLOR to OFF (Black & White = ON)
LSB of COLOR Byte = 0 for BLACK & WHITE
MODE#3 :

4 = SET COLOR to ON
LSB of COLOR Byte = 1 for COLOR
MODE#4 :

5 = Mixed Graphics and Text
TEXT = 40 columns by 24 lines
MODE#5: TEXT
GRAPHICS = 320 x 240 pixels
MODE#5: HGR:

6 = Mixed Graphics and Text
TEXT = 40 columns by 48 lines
MODE#6: TEXT:
GRAPHICS = 320 x 480 pixels
MODE#6: HGR:

```
7    = Mixed Graphics and Text
       TEXT     = 80 columns by 24 lines
     MODE#7: TEXT:
       GRAPHICS = 640 x 240 pixels
     MODE#7: HGR:


8    = Mixed Graphics and Text
       TEXT     = 80 columns by 48 lines
     MODE#8: TEXT:
       GRAPHICS = 640 x 480 pixels
     MODE#8: HGR:

9    = INTERNAL USE ONLY


1xx = Mixed Graphics and Text
      GRAPHICS = as chosen on the preselected Mixed page with
                 xx lines of text on the preselected Mixed page
                 where xx is 0 <= xx <= maximum number of lines
                       on the Mixed page.
```

The graphics area is defined as the equivalent space from the top of the screen to the text line "n" (where "n" is defined to be the value of "(maximum-lines - xx)". In other words, "n" is defaulted to 4 and therefore in the 80 x 24 mode, the graphics portion is from line 1 to line 20, (24 - 4 = 20), and text is lines 21 through 24.

Text can be PRINTed any where on the screen using the HTAB and VTAB commands to define the starting point of the text to be printed. The significance of the mixed mode print is the following:

1 - If the text is printed on a line inside of the graphics area, then the inverse cursor will not be shown and the PRINTed text only will show on the screen.

2 - If the text is PRINTed on a line inside of the text area, then the normal inverse cursor will be shown.

3 - When the text PRINTed exceeds the bottom of the screen, then the bottom "n" lines of text will be scrolled upward on the screen.

4 - Graphics can be plotted anywhere on the screen, even in the "text" area.

Using the last fact and setting the mode value xx to the number of lines in the text mode (i.e. xx=24 in the 80 x 24 mode) allows the graphics screen to scroll if a carriage return is printed on the last line.

PAGE COMMAND

To change high resolution graphics pages on the DIMENSION, use the PAGE command. By issuing either a PAGE#1 or a PAGE#2 command, the user can select either page 1 of the high resolution graphics or page 2.

NF FUNCTION

The NF function is an extension to the standard APPLESOFT that allows the determination of whether or not a file existed prior to the issuance of an OPEN command. This can be very helpful as the system duplicates APPLESOFT in that if the file does not exist, the file is then created.

VARPTR FUNCTION

The DIMENSION 68000 has some significant extensions to the standard APPLESOFT (TM) BASIC. The VARPTR function returns an integer whose value is the location, in memory, of the variable whose name was given as the argument in the call to the VARPTR function. The VARPTR function is discussed in Chapter 4 of this manual and in detail in the UNIBASIC REFERENCE MANUAL.

CHAPTER 2

C H A P T E R 3

D I S K F I L E H A N D L I N G

CHAPTER 3

FILENAMES

UNIBASIC filenames are made up of a combination of the CP/M-68K and the APPLESOFT (TM) conventions. The filename consists of three parts;

- The FILENAME
- The FILETYPE
- The DRIVE SPECIFICATION

The FILENAME consists of from one to eight characters. The first character must be alphabetic. All of the rest of the characters may be either alphabetic or numeric.

The FILETYPE consists of a period (.) followed by from one to three characters. The characters may be either alphabetic or numeric.

The DRIVE SPECIFICATION consists of a comma (,), followed by a D, followed by either a 1, a 2, a 3, or a 4. The numbers 1, 2, 3, and 4 correspond to the drives A:, B:, C:, and D:. If no DRIVE SPECIFICATION is provided, then the CP/M-68K default disk drive will be used.

As an example, the standard CP/M-68K filename B:TEST.DAT would be TEST.DAT,D2 for UNIBASIC.

UNIBASIC operates under the CP/M-68K operating system. CP/M forces all FILENAMES to be 8 characters internally. If the FILENAME is less than 8 characters, then CP/M pads the FILENAME out to 8 characters with blanks. If the FILENAME is greater than 8 characters, then CP/M assumes that the first 8 characters are the FILENAME. CP/M then inserts a period (.) after the first 8 characters, and then treats the next characters, up to 3 characters, as the FILETYPE.

CP/M assumes that there is always a FILETYPE. CP/M also assumes that the FILETYPE is always 3 characters long. If the FILETYPE is less than 3 characters long, then CP/M pads the FILETYPE out to 3 characters with blanks.

PROGRAM FILE COMMANDS

The following commands are used to manipulate program files. Each of these commands is discussed in detail in the UNIBASIC REFERENCE MANUAL.

SAVE <filename>
Writes to disk the program that currently resides in memory.

LOAD <filename>
Loads the program from disk into memory. LOAD always deletes the current contents of memory and closes all files before LOADing.

RUN <filename>
Loads the program from disk into memory and runs it. RUN deletes the current contents of memory and closes all files before loading the program.

ALOAD <filename>
Loads an ASCII text file as the program from disk into memory. ALOAD always deletes the current contents of memory and closes all files before loading the program.

ASAVE <filename>
Writes to disk, in ASCII text file format, the program that currently resides in memory.

BLOAD <filename>[,A<addr>][,D<drive-number>]
Loads a binary file into memory from the disk <filename> specified. The file is loaded at address <addr>. If <addr> is not specified, then the address saved in the disk file that is the location that the file was saved from is used.

BRUN <filename>[,A<addr>][,D<drive-number>]
Loads a binary file into the same memory locations from which the file was saved, or if specified, into the address <addr>. Then jumps to the file's first memory address and begins to attempt to execute.

BSAVE <filename>,A<addr>,L<length>,[D<drive-number>]
Writes to disk, in binary file format, the contents of memory at address <addr>, the length of memory written <length> bytes, to the disk file <filename>.

DISK DATA FILES
SEQUENTIAL AND RANDOM ACCESS

Two types of disk data files can be created and accessed by a UNIBASIC program; sequential access files and random access files. Both types of files are described in the following sections.

SEQUENTIAL ACCESS

Sequential access data files are easier to create than are random access data files, but they are limited in flexibility and speed when it comes to accessing data. Data is written to a sequential file as ASCII characters. These characters are stored, one after another (sequentially), in the order that the characters are sent to the disk. They are read back from the disk in the same way.

The statements and functions that are used with sequential files are:

OPEN
READ
WRITE
POSITION
PRINT
APPEND
CLOSE

See the UNIBASIC REFERENCE MANUAL for a more detailed discussion of these commands.

CREATING A SEQUENTIAL ACCESS FILE

The following program steps are required to create a sequential file and access the data in the file:

1. OPEN the file.

```
PRINT CHR$(4);"OPEN DATA,D1"
```

2. WRITE data to the file.

```
PRINT CHR$(4);"WRITE DATA"
PRINT INFO1
PRINT INFO2
PRINT INFO3
```

3. To access the data in the file, you must CLOSE the file and reOPEN it to READ the data.

```
PRINT CHR$(4);"CLOSE DATA"
PRINT CHR$(4);"OPEN DATA"
```

4. Use the INPUT statement to read data from the sequential file into the program.

```
PRINT CHR$(4);"READ DATA"
FOR I = 1 TO 3
  INPUT X$(I)
NEXT I
```

Program 1 creates a sequential file, named "DATA," from information you input at the keyboard.

PROGRAM 1 - CREATE A SEQUENTIAL DATA FILE (UNTESTED, REF. ONLY)

```
10 PRINT CHR$(4);"OPEN DATA.DAT,D1": REM  CREATES & OPENS FILE
20 INPUT "NAME?";N$
30 IF N$="DONE" GOTO 90: REM              USED TO END INPUT
40 INPUT "DEPARTMENT?";D$
50 INPUT "DATE HIRED?";H$
60 PRINT CHR$(4);"WRITE DATA.DAT": REM    WRITE DATA TO FILE
70 PRINT N$,D$,H$
80 PRINT:GOTO 20
90 PRINT CHR$(4);"CLOSE":END
RUN
NAME?MICKEY MOUSE
DEPARTMENT? AUDIO-VISUAL AIDS
DATE HIRED? 01/12/72

NAME?SHERLOCK HOLMES
DEPARTMENT?RESEARCH
DATE HIRED? 12/03/78

NAME?EBENEEZER SCROOGE
DEPARTMENT?ACCOUNTING
DATE HIRED?04/27/78

NAME?SUPER MAN
DEPARTMENT?MAINTENANCE
DATE HIRED?08/16/78

NAME?etc.
```

Program 2 accesses and files "DATA" that was created in Program 1 and displays the name of everyone hired in 1978.

PROGRAM 2 - ACCESSING A SEQUENTIAL FILE (UNTESTED, REF. ONLY)

```
10 PRINT CHR$(4);"OPEN DATA.DAT,D2": REM    OPENS FILE
20 PRINT CHR$(4);"READ DATA.DAT": REM       READS
30 INPUT N$,D$,H$: REM                        FILE
40 IF RIGHT$(H$,2)="78" THEN PRINT N$: REM  TESTS DATE HIRED
50 GOTO 20
RUN
EBENEEZER SCROOGE
SUPER MAN
Input past end in 20
Ok
```

Program 2 reads, sequentially, every item in the file. when all the data has been read, line 20 causes an "Input past end" error. To avoid getting this error, use the ONERR GOTO approach.

ADDING DATA TO A SEQUENTIAL FILE

Data can be added to an existing sequential access data file. It is important, however, to follow carefully the procedure given below.

WARNING

If you have a sequential access data file residing on disk and later want to add more data to the end of it, you must use the APPEND command instead of the WRITE command.

The following procedure will add data to an existing sequential access data file called "NAMES.DAT"

1. OPEN "NAMES.DAT"

2. APPEND the new information to the end of "NAMES.DAT"

3. Now the file, on the disk, called "NAMES.DAT" includes all the previous data plus the data you just added.

Program 3 illustrates this technique. It can be used to create or add onto a file called "NAMES.DAT".

PROGRAM 3 - ADDING DATA TO A SEQUENTIAL FILE (UNTESTED, REF. ONLY)

```
10 ON ERR GOTO 2000
20 PRINT CHR$(4);"OPEN NAMES.DAT"
30 REM ADD NEW ENTRIES TO FILE
40 INPUT "NAME?";N$
50 IF N$="" GOTO 140
60 REM CARRIAGE RETURN EXITS INPUT LOOP
70 INPUT "ADDRESS?";A$
80 INPUT "BIRTHDAY?";B$
90 PRINT CHR$(4);"APPEND NAMES.DAT"
100 PRINT N$
110 PRINT A$
120 PRINT B$
130 PRINT: GOTO 40
140 PRINT CHR$(4);"CLOSE"
150 END
2000  IF ERR = 53 AND ERL = 20 THEN PRINT CHR$(4);"OPEN NAMES.DAT":GOTO
      40
2020 ON ERR GOTO 0
```

The error handling routine in line 2000 traps a "File not found" error in line 20. If this happens, the statements that copy the file are skipped, and "NAMES.DAT" is created as if it were a new file.

RANDOM ACCESS

Creating and accessing random access data files requires more program steps than for sequential access files. However, there are advantages too in using random access data files. The biggest advantage of using random access data files is that data can be accessed randomly, i.e., anywhere on the disk - it is not necessary to read through all the information, as with sequential access files. This is possible because the information is stored and accessed in distinct units, called records, and each record is numbered.

The statements and functions that are used with random access files are:

OPEN
READ
WRITE
PRINT
CLOSE

See the UNIBASIC REFERENCE MANUAL for a detailed discussion of these statements and functions.

CREATING A RANDOM ACCESS FILE

The following program steps are required to create a random access file.

1. OPEN the file for random access. This example specifies a record length of 32 bytes. If the record length is omitted, the file will not be opened as a random access data file.

```
PRINT CHR$(4);"OPEN FILE.DAT,L32"
```

2. WRITE the data to the file.

```
FOR I = 1 TO 41
  PRINT CHR$(4);"WRITE FILE.DAT,R";I
  PRINT DATA
NEXT I
```

In this example, I is used as the record number.

Program 4 writes information that is input at the terminal to a random access data file.

PROGRAM 4 - CREATE A RANDOM ACCESS FILE (UNTESTED, REF. ONLY)

```
10 PRINT CHR$(4);"OPEN FILE.DAT,L32"
20 REM N$ = 20 CHAR, A$ = 4 CHAR, P$ = 8 CHAR
30 INPUT "2-DIGIT CODE";CODE%
40 INPUT "NAME?";N$
50 INPUT "AMOUNT";AMT
60 INPUT "PHONE";TEL$: PRINT
70 REM DO CONVERTS
75 N$ = LEFT$(N$+"                    ",20)
80 A$ = RIGHT$("0000"+STR$(AMT),4)
90 P$ = LEFT$(TEL$+"        ",8)
100 PRINT CHR$(4);"WRITE FILE.DAT,R";CODE%
105 PRINT N$;A$;P$
110 GOTO 30
```

Each time lines 100 and 105 are executed, a record is written to the file. The two-digit code that is input in line 30 becomes the record number.

ACCESSING A RANDOM ACCESS FILE

The following steps are required to access a random access data file:

1. OPEN the file in random access mode.

```
PRINT CHR$(4);"OPEN FILE.DAT,L32"
```

2. READ the record.

```
PRINT CHR$(4);"READ FILE.DAT,R";CODE%
INPUT XX$
```

3. SEPARATE the string that was input into the various data fields.

```
N$ = LEFT$(XX$,20)
A$ = MID$(XX$,21,4)
P$ = MID$(XX$,25,8)
```

Program 5 accesses the random access data file "FILE.DAT" that was created in Program 4. When the two-digit code set up in Program 4 is input, the information associated with that code is read from the file and displayed, hence "RANDOM ACCESS".

PROGRAM 5 - ACCESS A RANDOM FILE (UNTESTED, REF. ONLY)

```
10 PRINT CHR$(4);"OPEN FILE.DAT,L32"
20 REM N$ = 20 CHAR, A$ = 4 CHAR, P$ = 8 CHAR
30 INPUT "2-DIGIT CODE?";CODE%
40 PRINT CHR$(4);"READ FILE.DAT,R";CODE%:INPUT XX$
45 N$ = LEFT$(XX$,20):A$ = MID$(XX$,21,4):P$ = MID$(XX$,25,8)
50 PRINT N$
55 AMT = INT(VAL(A$)): ANT = (VAL(A$)-AMT)*100
56 AN$ = RIGHT$("00"+STR$(ANT),2):AM$ = RIGHT$("0000"+STR$(AMT),4)
60 PRINT "$";AM$;".";AN$
70 PRINT P$:PRINT
80 GOTO 30
```

Program 6 is an inventory program that illustrates random file access. In this program, the record number is used as the part number. It is assumed that the inventory will contain no more than 100 different part numbers.

Lines 900-960 initialize the data file by writing CHR$(255) as the first character of each record. This is used later (line 270and line 500) to determine whether an entry already exists for that part number.

Lines 140-210 display the different inventory functions that the program performs. When you type in the desired function number, line 230 branches to the appropriate subroutine.

PROGRAM 6 - INVENTORY (UNTESTED, REF. ONLY)

```
120 PRINT CHR$(4);"OPEN INVEN.DAT, L39"
130 REM F$ - 1, D$ - 30, Q$ - 5, R$ - 5, P$ - 6
140 PRINT:PRINT"FUNCTIONS:":PRINT
150 PRINT"1 INITIALIZE FILE"
160 PRINT"2 CREATE A NEW ENTRY"
170 PRINT"3 DISPLAY INVENTORY FOR ONE PART"
180 PRINT"4 ADD TO STOCK"
190 PRINT"5 SUBTRACT FROM STOCK"
200 PRINT"6 DISPLAY ALL ITEMS BELOW REORDER LEVEL"
210 PRINT:PRINT:INPUT"FUNCTION?";FUNCTION
220 IF (FUNCTION<1) OR (FUNCTION>6) THEN PRINT "BAD FUNCTION NUMBER":
    GOTO 140
230 ON FUNCTION GOSUB 900,250,390,480,560,680
240 GOTO 210
250 REM BUILD NEW ENTRY
260 GOSUB 840
270 IF ASC(F$)<>255 THEN INPUT "OVERWRITE?";A$:IF A$<>"Y" THEN RETURN
280 F$ = CHR$(0)
290 INPUT "DESCRIPTION?";DESC$
300 D$ = DESC$
310 INPUT "QUANTITY IN STOCK?";Q%
320 Q$ = LEFT$("     "+STR$(Q%),5)
330 INPUT "REORDER LEVEL?";R%
340 R$ = LEFT$("     "+STR$(R%),5)
350 INPUT "UNIT PRICE?";P
355 PA = INT(P + .5): PB = INT((P - PA) * 100 + .5)
360 P$ = RIGHT$("     "+STR$(PA),3)+"."+RIGHT$("00"+STR$(PB),2)
370 GOSUB 970: REM PUT IT IN FILE
380 RETURN
```

```
390 REM DISPLAY ENTRY
400 GOSUB 840
410 IF ASC(F$) = 255 THEN PRINT "NULL ENTRY": RETURN
420 PRINT "PART NUMBER ";P$
430 PRINT D$
440 PRINT "QUANTITY ON HAND ";Q$
450 PRINT "REORDER LEVEL ";R$
460 PRINT "UNIT PRICE $";P$
470 RETURN
480 REM ADD TO STOCK
490 GOSUB 840
500 IF ASC(F$)=255 THEN PRINT "NULL ENTRY": RETURN
510 PRINT D$: INPUT "QUANTITY TO ADD?";A%
520 Q% = INT(VAL(Q$)) + A%
530 Q$ = LEFT$("      "+STR$(Q%),5)
540 GOSUB 970: REM PUT IT IN THE FILE
550 RETURN
560 REM REMOVE FROM STOCK
570 GOSUB 840
580 IF ASC(F$)=255 THEN PRINT "NULL ENTRY": RETURN
590 PRINT D$
600 INPUT "QUANTITY TO SUBTRACT?";S%
610 Q% = INT(VAL(Q$))
620 IF (Q%-S%)<0 THEN PRINT "ONLY ";Q%;" IN STOCK": GOTO 600
630 Q% = Q% - S%
635 R% = INT(VAL(R$))
640 IF Q%<R% THEN PRINT "QUANTITY NOW ";Q%;"  REORDER LEVEL ";R%
650 Q$ = LEFT$("      "+STR$(Q%),5)
660 GOSUN 970: REM PUT IT IN FILE
670 RETURN
680 REM DISPLAY ITEMS BELOW REORDER LEVEL
690 FOR I = 1 TO 100
710   GOSUB 1040: REM GET IT FROM FILE
715   Q% = INT( VAL( Q$)): R% = INT( VAL( R$))
720   IF Q%<R% THEN PRINT D$;" QUANTITY ";Q%;"    REORDER LEVEL ";R%
730 NEXT I
740 RETURN
840 INPUT "PART NUMBER?";PART%
850 IF (PART%<1) OR (PART%>100) THEN PRINT "BAD PART NUMBER":GOTO 840
855 GOSUB 1040: RETURN
890 END
900 REM INITIALIZE FILE
910 INPUT "ARE YOU SURE?";B$:IF B$<>"Y" THEN RETURN
920 F$ = CHR$(255)
930 FOR I = 1 TO 100
940   GOSUB 970: REM PUT IT IN FILE
950 NEXT I
960 RETURN
```

```
970 REM WRITE TO FILE
980 F$=LEFT$(F$+" ",1)
990 D$=LEFT$(D$+"                                ",30)
1000 Q$ = LEFT$(Q$+"  ",5): R$ = LEFT$(R$+"  ",5)
1010 P$ = LEFT$(P$+"    ",6)
1020 PRINT CHR$(4);"WRITE INVEN.DAT,R";PART%:PRINT F$;D$;Q$;R$;P$
1030 RETURN
1040 REM READ FROM FILE
1050 PRINT CHR$(4);"READ INVEN.DAT,R";PART%:INPUT XX$
1060 F$ = LEFT$(XX$,1): D$ = MID$(XX$,2,30)
1070 Q$ = MID$(XX$,32,5): R$ = MID$(XX$,34,5)
1080 P$ = MID$(XX$,36,6)
1090 RETURN
1100 END
```

C H A P T E R 4

U N I B A S I C
A S S E M B L Y L A N G U A G E S U B R O U T I N E S

CHAPTER 4

UNIBASIC provides interfacing with assembly language programs with the CALL statement and the CALL function.

CALL STATEMENT

Assembly language subroutine calls can be made with the call statement. The syntax is:

```
CALL <address> [( <argument list> )]
```

where <address> = an arithmetic expression whose value is the location in memory of the assembly language subroutine that is to be accessed.
<argument list> = a list of arguments, or variables, whose values are to be available to the assembly language sub-routine.

The argument list, if used, can contain up to 15 arguments. Each argument is a 32 bit double integer. Addresses, data, and arguments can be arithmetic expressions. Arguments are written into the DØ - D7 and AØ - A6 registers respectively.

When CALL is used as a function, the value shall be returned in the DØ register.

VARPTR

UNIBASIC includes a VARPTR function, which returns a signed integer whose value is the address of the first byte of the data identified by the named <variable>. A detailed discussion of the function is described in the UNIBASIC REFERENCE MANUAL.

Syntax VARPTR (<variable>)

where <variable> is the name of a numeric variable, a string variable, or an array variable.

A P P E N D I X A

T E R M I N O L O G Y

TERMINOLOGY

TERMINOLOGY

ALPHANUMERIC. Characters which consist of letters and/or digits.

APPLEDOS. Apple Disc Operating System: The disk operating system used in Apple computers.

APPLICATIONS PROGRAM. Programs, or software, designed for word-processing, games, education, home-finance, and other practical uses.

ASCII. A contraction for the "American Standard Code for Information Interchange. This standard defines the codes for a character set to be used for information interchange. It is used to store characters in memory and to transmit them to peripheral devices such as printers and other computers.

BACKUP. A copy of a file that can be used in the event that the original is lost or damaged, or used instead of the original to protect the original.

BASIC. A contraction for the "Beginner's All-purpose Instruction Code. It is a computer language that is easy to learn and use. BASIC is widely used with microcomputers. BASIC was developed at Dartmouth College with the assistance of General Electric.

BINARY. A characteristic, property, or condition in which there are but two possible alternatives. The binary number system using 2 as its base or radix, uses only the digits zero (Ø) and (1). Most computers store numbers in binary format.

BIT. A binary digit, either Ø or 1. The most basic unit of memory in a binary computer.

BIT MAPPED I/O. A technique whereby bits in memory are used to control the Input/Output.

BOOT. To ready a computer for use by loading the disk operating system into the computer's temporary memory, or RAM. The term derives from the idea that the "bootable" program loads itself into the system by it's own bootstraps.

BYTE. A group of eight adjacent bits that are treated as a single entity. A byte may be used to store a single character or a binary number.

CHAINING. The process where one program causes another program to execute when it finishes. The first program is said to "chain" to the second if it transfers control to the next program and it keeps the variables from the first program intact.

CHARACTER. A string of bits (a byte) which represents a symbol that can be displayed on a screen or printed.

CHARACTER COORDINATES. The position on the screen denoted by a line number and a character position within that line. The standard Dimension screen consists of 8Ø columns of characters by 24 lines of characters. See SCREEN COORDINATES.

CHARACTER SET. All the characters that can be used with a particular computer. The Dimension character set consists of 256 characters. Characters Ø-127 are the ASCII character set. The other 128 are special symbols.

CHIP. An integrated circuit made by etching myriads of transistors and other electronic components onto a wafer of Silicon a fraction of an inch on a side.

COMMAND. An order to the computer to execute a task.

COMPILER. A computer program that translates a computer language such as BASIC to a form known as machine language, which is a form that can be interpreted or executed directly by a computer.

CONTINUOUS FORMS. Sheets of perforated paper with sprocket holes on the side that can be fed into a printer continuously rather than one sheet at a time. (Usually Fan-Folded)

CONTROL KEY. Key that executes commands, in conjunction with other keys pressed simultaneously.

COPY. To duplicate a file or program in order to retain the original and work on the duplicate. Usually refers to duplicating one disk to another. Also see BACKUP.

COPY PROTECT. A technique which prevents a diskette from being copied.

CP/M. Control Program for Microprocessors, developed by Gary Kildall of the Digital Research Corp. The disk operating system that has become an industry standard for business-oriented personal computers.

CPU (Central Processing Unit). The chip that directs the flow of information within the computer and does the actual computing. Also frequently used to refer to the physical part of the computer that contains the CPU chip and other ancillary hardware.

CRASH. Abrupt computer failure.

CRT. The Cathode Ray Tube in a television set or video display monitor.

CURSOR. A small rectangle of light which marks the input position on the screen.

DATA. Information that a computer processes.

DATABASE. A collection of related data, such as in inventory or a collection of names on a mailing list.

DEFAULT. A preset system parameter value that will be used unless it is changed.

DISK DRIVE. A device that uses a rotating platter or disk to store data and programs.

DISK OPERATING SYSTEM (DOS). The program that instructs the computer's CPU how to transfer information to and from a disk.

DISKETTE. A low-cost sheet of magnetic material enclosed in an envelope. A diskette can be put into a disk drive and used to store data.

DISPLAY. The information on a video screen.

DOCUMENTATION. Written instructions that tell you how to use computer hardware or software.

DOT MATRIX. A technique whereby characters are defined as a two-dimensional array of dots.

DOUBLE DENSITY. A way of putting information on a disk that allows the disk to store twice as much data as a single-density disk.

EDITOR. A computer program that can be used to enter and change data on the screen.

ENHANCEMENT. Improvement.

EXCLUSIVE-OR. A Boolean operation that is true(1) if either, but not both, of its inputs are true (1). Otherwise, the result is false (Ø).

FILE. A set of records stored on a device such as a diskette or tape.

FIRMWARE. A program stored in the computer's permanent memory, or ROM. Since such a program doesn't have to be re-entered every time the computer is turned on, it is "harder" than software.

FLOPPY DISK. A small, flexible sheet of magnetic media used to store data.

FONT. A set of characters.

FORMATTED DISK. A diskette that has been initialized with timing information so that it can be read and written by a computer.

FRIENDLINESS. How easy a program or computer is to work with. A "user friendly" program is one that takes little time to learn, or that offers on-screen prompts, or that protects the user from making disastrous mistakes.

GRAPHICS. Visual information constructed using objects such as lines, circles, and rectangles.

GRAPHICS LANGUAGE. A set of commands that are used to describe how graphics images are to be drawn.

GRAPHICS PRINTER. A printer capable of transferring graphics data to the printed page. Most graphics printers print dots to represent the pixel elements.

HALF ADDER. A circuit that sums two binary (Ø or 1) inputs.

HARD COPY. Text or other work printed on paper by a printer. Same as print-out.

HARD DISK. A rigid disk used to store information. Hard disks can store far more information than floppy disks and can write and read information more quickly.

HARDWARE. The physical parts of a computer system as opposed to the programs, or software.

HIGH-LEVEL LANGUAGE. A programming language such as BASIC, written in a kind of English shorthand rather than in numbers and symbols.

IMAGE FILE. A file on a diskette or other media that contains the bits that comprise a graphics image. If this file is read into the area of memory that is mapped to the screen, the image is displayed.

INITIALIZE. To reset the computer and its peripherals to a starting state before beginning a task. Done automatically by the disk operating system.

INTERFACE. A communication path between a computer and peripheral devices such as printers and disk drives.

INTERFACE CARD. A printed circuit card providing the control logic needed for communication between the computer and an external device.

INPUT/OUTPUT (I/O). An input device such as a keyboard feeds information into the computer. An output device such as a printer or monitor takes information from the computer and turns it into usable form. Modems, cassettes, and disks work in both directions, so they are I/O devices. Input and output are also used as verbs: You input data from the keyboard.

I/O SLOT. The location where an interface card plugs into the computer.

K. One kilobyte, or 1,Ø24 bytes of memory.

LINKAGE. The establishing of a communication path between programs or parts of programs.

LITERAL. A string of characters within quotes, i.e., "LITERAL".

LOAD. To enter a program into the computer from cartridge, cassette, or disk.

MEMORY. An area inside the computer where data such as numbers, characters, and program instructions are stored. A computer's memory capacty is measured in units known as K's. One K is equal to 1Ø24 bytes of memory.

MENU. A list of options displayed on the screen. The options can usually be selected by typing a single letter or number.

MICROPROCESSOR. Another name for the CPU chip.

MODEM. Short for modulator-demodulator--a piece of equipment that links two computers over a telephone line.

MONITOR. A supervisory program that controls the sequencing of other activities. Video device; quality of display is better than that of a television set's.

MS-DOS. A disk operating system developed by MicroSoft. Used in modified form by the IBM Personal Computer, under the designation PC-DOS, and now used in a number of other computers as well.

ON-LINE. An I/O device is on-line if it is attached to the computer via an active interface. Otherwise, it is off-line.

OPEN (FILE). Before a file can be read or written, the program must locate the file and open it.

OPERATING SYSTEM. Programs, such as monitors and compilers, that enable you to use a computer.

OVERLAY. A technique whereby a program that is too large to fit in memory is divided into segments that are loaded only as they are needed.

PAGE. The basic unit of a file. Each page is one screen of data--either text or graphics.

PARALLEL INTERFACE. A port that sends or receives the eight bits in each byte all at one time. Many printers likely to be used in homes use a parallel interface to connect to the computer.

PARSE. A procedure or technique used to separate a group or groups of characters (i.e. letters, words, or numbers) from a line of text so that the groups or phrases may be used in later processing.

PASCAL. A general-purpose computer language that is easy to understand and to use.

PC-DOS. IBM's name for the disk operating system used in the IBM Personal Computer. Similar to MS-DOS.

PERIPHERALS. Accessory parts of a computer system not considered essential to its operation. Printers and modems are peripherals.

PIXEL. A picture element. Each pixel defines one dot on the screen.

PORT. The gateway that connects the computer to its outside world.

POWERFUL. Usually refers either to a computer with a lot of memory or a lot of processing speed (a DIMENSION 68000 computer with 256K RAM is "powerful") or to a program with unusual versatility (a spreadsheet is a "powerful" business tool).

PRINT CONTROL CHARACTERS. Character codes that are not printed on paper. Instead, they are used to cause a printer action such as move to the top of the next page or to skip a line.

PRINTER. Transforms computer's output into hard copy.

PRINTOUT. See HARD COPY.

PROGRAM. A sequence of instructions written in a computer language such as BASIC that controls what a computer does.

PROGRAMMABLE KEY. Another term for user or program defined key.

PROMPT. An on-screen hint to the user about what to do next.

RAM. Random Access Memory: "Temporary" memory on chips. You can store data in RAM or take data from RAM at very high rates of speed. It's temporary, or volatile, because information stored in it disappears when the computer is switched off.

READ. To extract data from a computer's memory or from a tape or disk.

RESET. See INITIALIZE.

ROM. Read Only Memory: "Permanent" memory on chips. You can read permanently stored programs from ROM but cannot store information in it. It's permanent memory because the information stored in ROM remains there when you turn the computer off. (Also called firmware)

SAVE. A command to the computer to store completed work on tape or disk.

SCREEN COORDINATES. The x,y location of pixel elements on the screen. The Dimension high-resolution screen consists of 640 rows. Each row contains 480 pixels.

SCROLL. To move a video display up or down, line by line, or row by row, character by character.

SEGMENTATION. The process of dividing a program into pieces that can be overlayed in memory.

SERIAL INTERFACE. A port that sends or receives the eight bits in each byte one by one, much like beads on a string. Printers that will be located far from the computer usually require a serial interface.

SOFT-FUNCTION KEY. See USER-DEFINED KEY.

SOFTWARE. Another name for programs.

SPECIAL-FUNCTION KEY. Usually understood to mean the CONTROL, SHIFT, ESCAPE, ALTERNATE, or PRINT SCREEN keys.

STORAGE. Usually refers to long-term storage, such as storage of files on tape or disk.

SUPPORT. Help available from computer and software merchants. Also used as a verb to describe what things are compatible with each other, as in: "with a Z-80 card, the DIMENSION 68000 will support CP/M-80 and TRS-80 software."

TRSDOS. TRS Disk Operating System: The disk operating system used in Tandy Radio Shack's personal computers.

TYPE-AHEAD BUFFER. A set of memory locations that is used to store characters as they are typed. The program may accept these characters from the buffer at a slower rate than they are typed. A type-ahead buffer is used so that if characters are being typed faster than the program can accept them, they are not lost.

USER-DEFINED KEY. A key whose function you or a program can change, so that a command or sequence of commands can be exected with a single keystroke. Same as programmable key and soft-function key. Unlike a special-function key, a user-defined key may have a predefined purpose.

UTILILTY PROGRAM. A program that can be used for basic file operations such as formattiing and copying diskettes and printing files.

VOLUME. A device capable of storing one or more files. Each diskette has a volume name that identifies it. Devices such as printers and disk drives sometimes are specified by a volume number.

WINCHESTER DRIVE. A form of hard disk permanently sealed into a case.

WRITE. To enter information into memory or onto a tape or disk.

WRITE-PROTECT. Any technique that prevents a diskette or tape from being written on. The write-protect notch is located on the right side of a 5 and 1/4 inch diskette. If this notch is covered with a piece of tape, data on the diskette cannot be written over because the write electronics are prevented from doing so by a sensor that senses the absense of an open notch.

A P P E N D I X B

B A C K - U P P R O C E D U R E

BACK-UP PROCEDURE

The DIMENSION 68000 System is shipped with two diskettes, the diskettes are labeled "SYSTEM 1" and "SYSTEM 2". It is STRONGLY recommended that you make copies of these diskettes, and then operate off of the copies. This protects the originals. If anything should happen to the copies, new copies can be made, as the originals are intact. The process of making copies of any important diskettes, so as to protect them from damage, etc., is called "making back-ups" or "backing up".

To BACK-UP the "SYSTEM 1" and "SYSTEM 2" diskettes, perform the following steps:

1 - TURN ON the POWER

2 - INSERT the "SYSTEM 1" diskette into DISK DRIVE A:

3 - INSERT a BLANK, UNFORMATTED DISKETTE into DISK DRIVE B:

4 - When the CP/M prompt (A>) appears at the left side of the screen, type in the following command:

A>format<CR>

where <CR> means the "Retrn" key or the "Enter" key. Both of these keys cause the ASCII carriage return code to be generated.

5 - The format program will then display the DIMENSION 68000 FORMAT program select menu, which looks something like the following:

```
Micro Craft DIMENSION 68000 Disk Formatting Program
******  5 1/4 Inch Drives  ******
A = Micro Craft Standard 40 track
B = Micro Craft Standard 80 track
C = IBM-PC Single and Double Sided
D = TRS-80 Model III
E = KayPro
F = Cromeco Single Density
G = Osborne Single Density
******  8 Inch Drives  ******
H = 8 Inch 3740 Format, Single Density, Single Sided
I = 8 Inch TRS-16, Double Density, Double Sided
Select Type
```

6 - PRESS the A Key.

The format program will then ask the following :

```
Which drive to use? (a-h)
```

7 - PRESS the B Key.

The format program will next ask the following:

Do you wish (F)ormat, (T)est, (D)ump, (P)rint

8 - PRESS the F Key.

The format program will then display the following message.

Starting format

After the above message is displayed, the red indicator light on disk drive B will turn on and disk drive B will make noise as the disk head is positioned. The disk drive will make noises every time it repositions the disk head for another track on the disk. Formatting the disk takes about 62 seconds. When the disk has been formatted, the disk is then tested. The format program will display the following message:

Starting test

After the above message is displayed, the format program tests the formatted diskette by attempting to read what was written on each track of the diskette. In this fashion, each track of the diskette is verified. If the format program cannot verify any part of the diskette, an error message is displayed. The error message will identify specifically the disk head, the disk track, and the disk sector where the error occurred.

Do not attempt to use a disk that fails this test.

9 - When the format program has finished, the format program will then ask the following:

Another function (y) or return to cpm (n)

PRESS the Y key.

10- REMOVE the diskette that has just been formatted from disk drive B and put it aside to be used later.

INSERT another BLANK, UNFORMATTED DISKETTE into DISK DRIVE B:

11- The format program will again display the DIMENSION 68000 FORMAT program select menu, as in step 5.

12- PRESS the A Key, as in Step 6.

13- PRESS the B Key, as in Step 7.

14- PRESS the F Key, as in Step 8.

15- When this diskette has been formatted, the format program will again ask:

Another function (y) or return to cpm (n)

PRESS the N key

16- The format program will display the CP/M prompt (A>). ENTER the following command:

A>copy all a b [v]

This command will load the CP/M-68K DISK COPY program and instruct the copy program to copy the contents of the diskette in disk drive A onto the diskette in disk drive B and to verify that the information is copied correctly.

17- After the above command has been entered, the format program will display the following message:

(^C to ABORT)
RETURN to copy ALL from A to B

18- PRESS the <CR> Key

This will start the copying process. The format program will then display the following message:

```
*** COPYING TRACKS ***

     Ø
```

As each diskette track is copied, the format program will display the number of the track that it is copying on the next line. So, that when the format program is copying track 5, the format program will be displaying the following message:

```
*** COPYING TRACKS ***

     Ø
     1
     2
     3
     4
     5
```

When the last track has been copied, the format program will display the following message:

```
     21
     22
     23
     24
     25
     26
     27
     28
     29
     3Ø
     31
     32
     33
     34
     35
     36
     37
     38
     39
disk full
A>
```

19- MAKE a diskette LABEL for the diskette that has just been copied. Write the label BEFORE it is put on the diskette. DO NOT ever put a label on a diskette and then write on the label with a hard writing instrument, such as a ball-point pen. If this is done, the diskette may be permanently damaged, and the diskette will NOT be usable.

    If it is necessary to mark on a label that is already on a diskette, then use a felt-tip pen.

    REMOVE the diskette from disk drive B and PUT the LABEL on the diskette that has just been copied.

20- PUT the diskette that was formatted earlier and set aside (in Step 10) into disk drive B.

21- ENTER the following command:

```
A>copy all a b [v]
```

22- The copy program will display the following message:

```
(^C to ABORT)
RETURN to copy ALL from A to B
```

    REMOVE the "SYSTEM 1" diskette from disk drive A: and PUT the diskette in a safe place for safe keeping.

    Diskettes should NOT be left in direct sunlight, they should not be exposed to magnetic fields, they should NOT be stapled, paper-clipped, or folded. The magnetic surface should NOT be touched. Nor should any liquid be spilled on the diskette. Also, diskettes should not be exposed to heat above about 120 degrees F., nor should they be exposed to cold below about 32 degrees F. (Do NOT leave diskettes in a locked automobile in the summer!)

23- INSERT the "SYSTEM 2" diskette into disk drive A:

24- PRESS the <CR> Key.

    This will start the copying of the "SYSTEM 2" diskette.

25- When the copying is complete, REMOVE the "SYSTEM 2" diskette from disk drive A and PUT the diskette with the "SYSTEM 1" diskette in a safe place.

26- RE-INSERT the copied "SYSTEM 1" diskette and CONFIGURE the CP/M operating system for the amount of memory on the system.

If the system has 128K bytes of Random Access Memory (RAM) installed, then ENTER the following command:

```
A>sys128
```

If the system has 256K bytes of Random Access Memory (RAM) installed, then ENTER the following command:

```
A>sys256
```

If the system has 384K bytes of Random Access Memory (RAM) installed, then ENTER the following command:

```
A>sys384
```

If the system has 512K bytes of Random Access Memory (RAM) installed, then ENTER the following command:

```
A>sys512
```

The execution of the "SYS" command will cause the CP/M-68K operating system to be configured to the memory size specified in the "SYS" command.

It is a good idea to copy the configured diskette so that there is a back-up of the configured "SYSTEM 1" diskette. The steps to take are similar to the steps taken above.

I N D E X

D I M E N S I O N 6 8 Ø Ø Ø

S Y S T E M U S E R ' S G U I D E

INDEX

INDEX
DIMENSION 68000
SYSTEM USER'S GUIDE