dimension

MICRO CRAFT CORPORATION

January 5, 1984

The Unibasic User's Guide (part number 680-0200-100) has been revised. Here are the new, revised pages that are to be placed into the manual.

The pages to be replaced are as follows:

Pages to be Replaced	New Page
3,4	3,4
15,16	15,16
17,18	17,18
21,22	21,22
23,24	23,24
25,26	25,26
27,28	27,28

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 Ø - Ø 2 Ø Ø - 1 Ø Ø A

PRELIMINARY

Ø 1 / Ø 5 / 8 4   R E V I S I O N

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.

This manual is copyrighted. All rights are reserved. This document may not, in whole or in part, be copied, photocopied, translated or reduced to any electronic medium or machine readable form without prior consent, in writing, from Micro Craft Corporation.

Copyright 1983 by Micro Craft Corporation

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

We are Micro Craft Corporation, designers and manufacturers of the the DIMENSION 68ØØØ, 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 68ØØØ.

Welcome to the realm of DIMENSION computing.

SIMILARITIES TO APPLESOFT BASIC

The UNIBASIC BASIC language interpreter, by RD Software, Inc., 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. UNIBASIC also allows "peeks" and "pokes" to absolute memory locations using the APEEK and APOKE commands.

REQUIREMENTS

UNIBASIC requires:

256K of memory minimum:

- 6ØK for UNIBASIC
- 64K for graphics and text buffers
- 32K for CP/M-68K
- Additional memory to run programs

1 diskette drive

All Dimension 68ØØØ 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 68ØØØ system is purchased. UNIBASIC resides on the "SYSTEM 1" diskette.

The Dimension 68ØØØ 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 which 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 68ØØØ 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 68ØØØ 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 68ØØØ 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.

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 68ØØØ 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# command must be followed immediately by either a TEXT command, an HGR command, or an HGR2 command. The MODE values and command sequences are shown below.

MODE#n where n is one of the following values:

```
Ø   = Initialize video to 8Ø columns by 24 lines.
      MODE#Ø


1   = Reset ERROR FLAG to OFF
      MODE#1

      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
      MODE#2


3   = Reset COLOR to OFF (Black & White = ON)
      MODE#3


4   = SET COLOR to ON
      MODE#4


5   = Mixed Graphics and Text
        TEXT     = 4Ø columns by 24 lines
        GRAPHICS = 32Ø x 24Ø pixels
      MODE#5


6   = Mixed Graphics and Text
        TEXT     = 4Ø columns by 48 lines
        GRAPHICS = 32Ø x 48Ø pixels
      MODE#6
```

```
7   = Mixed Graphics and Text
        TEXT     = 8Ø columns by 24 lines
        GRAPHICS = 64Ø x 24Ø pixels
      MODE#7


8   = Mixed Graphics and Text
        TEXT     = 8Ø columns by 48 lines
        GRAPHICS = 64Ø x 48Ø pixels
      MODE#8

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 Ø <= 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 8Ø x 24 mode, the graphics portion is from line 1 to line 2Ø, (24 - 4 = 2Ø), 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 8Ø 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 68ØØØ 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.

CALL FUNCTION with Arguments

The CALL function can use arguments to link data to an assembly language function. See the CALL function in Chapter 4 of this manual.

CHAPTER 2

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. If the FILETYPE is NOT explicitly stated, then CP/M defines the FILETYPE to the default, which is blanks. 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 long 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.

   ```
   DIM X$(3)
   PRINT CHR$(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". For a list of error numbers, see the UNIBASIC Reference Manual discussion regarding "ONERR...GOTO" on page 82.

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$(4);"CLOSE"
150 END
1985 REM *************************
1990 REM  ERR 42 = OUT OF DATA
1995 REM  ERR 6  = END OF DEVICE
1997 REM *************************
2000 IF ERR = 42 OR ERR = 5 THEN PRINT CHR$(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.

I N D E X

U N I B A S I C   U S E R ' S   G U I D E

INDEX
UNIBASIC USER'S GUIDE