U N I B A S I C

R E F E R E N C E M A N U A L

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 - 2 0 0

PRELIMINARY

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 680-0200-200.

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

T A B L E O F C O N T E N T S

TABLE OF CONTENTS

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

INTRODUCTION

UNIBASIC is a powerful BASIC language interpreter for the Dimension 68000 system. UNIBASIC is designed to run under the CP/M-68K operating system. UNIBASIC is supplied "bundled" with the Dimension 68000 system.

HOW TO USE THIS MANUAL

This manual is a reference for the UNIBASIC interpreter.

This manual is divided into three chapters plus appendices. Chapter 1 covers a variety of topics, largely pertaining to data representation in UNIBASIC. Chapter 2 describes the syntax and semantics of every command and statement in UNIBASIC, ordered alphabetically. Chapter 3 describes all UNIBASIC intrinsic functions, also ordered alphabetically. The appendices contain, among other things, a list of error messages and codes, a list of mathematical functions, a list of ASCII character codes, and a list of UNIBASIC reserved words.

Additional information about programming UNIBASIC is covered in the UNIBASIC USER'S GUIDE. the USER'S GUIDE describes the features of UNIBASIC. It also contains information relevant to your operating system, CP/M-68K, and helpful hints about such matters as data I/O and assembly language subroutines.

SYNTAX NOTATION

Unibasic commands and statements are described using the following conventions and definitions:

CAPITALS	Items in capital letters must be input as shown.
Lower-case words	Lower-case words shown in the syntax definition should be replaced with their value when entered. For example, in RUN [line-number], the lower-case name line-number, if specified, should be replaced with an actual line number. Example: RUN 1Ø.
[]	Square brackets indicate that the enclosed material is optional. Example: RUN [line-number]. The brackets indicate that line-number need not be specified.
{}	Braces indicate that the enclosed material may be repeated.
|	A vertical bar means "or" : The material on the left or right of the bar may be specified.

All punctuation except angle brackets and square brackets (i.e., commas, parentheses, semicolons, hyphens, equal signs) must be included where shown. Blanks or spaces are generally ignored.

C H A P T E R 1

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

CHAPTER 1

GETTING STARTED

The Dimension 68000 system is shipped with a "SYSTEM 1" diskette and a "SYSTEM 2" diskette. Before doing ANYTHING else, make a copy of the "system diskettes that were shipped with your DImension 68000 system. A step by step procedure for making these copies, or "BACKING-UP" these diskettes is included in the appendices, BACKING-UP.

Always make a back-up of any diskettes received from Micro Craft, Inc.

If you should damage the "SYSTEM" diskette or the "LANGUAGES UTILITIES" diskette, additional diskettes may be purchased from Micro Craft, Inc., for $350.00 plus shipping and handling fees.

RUNNING UNIBASIC

To use the Micro Craft, Inc., UNIBASIC interpreter on the Dimension 68000 system, insert the "SYSTEM 1" diskette into the "A" diskette drive. Then, either type

BASIC

or

BASIC filename

where filename = the name of the file that contains the basic program to be run.

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 up 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 disk drives A:, B:, C:, AND D:. If no DRIVE SPECIFICATION is included, the system will use the system default disk drive.

MODES OF OPERATION

When UNIBASIC is initialized it displays the prompt character ":". The prompt character indicates that UNIBASIC is at the command level; that is, it is ready to accept commands. At this point, UNIBASIC may be used in either of two modes: direct mode or indirect mode.

In direct mode, UNIBASIC statements and commands are not preceded by line numbers. They are executed as they are entered. Results of arithmetic and logical operations may be displayed immediately and stored for later use but the instructions themselves are lost after execution. Direct mode is useful for debugging and for using UNIBASIC as a "calculator" for quick computations that do not require a complete program.

Indirect mode is used for entering programs. Program lines are preceded by line numbers and are stored in memory. The program stored in memory is executed by entering the RUN command.

LINE FORMAT

UNIBASIC program lines have the following format (square brackets indicate optional input):

```
nnnnn UNIBASIC-STATEMENT[:UNIBASIC-STATEMENT...] <CR>
```

More than one UNIBASIC statement may be placed on a line, but each must be separated from the last by a colon.

A UNIBASIC program line always begins with a line number and ends with a carriage return. A line may contain a maximum of 255 characters.

It is possible to extend a logical line over more than one physical line by using the <LINE-FEED> key. <LINE-FEED> lets you continue typing a logical line on the next physical line without entering a <CR> (carraige return).

LINE NUMBERS

Every UNIBASIC program line begins with a line number. Line numbers indicate the order in which the program lines are stored in memory. Line numbers are also used as references in branching and editing. line numbers must be in the range of Ø to 63999.

A period (.) may be used in EDIT, LIST, AUTO, and DELETE commands to refer to the current line.

CHARACTER SET

The UNIBASIC character set is comprised of the alphabetic characters, numeric characters, and special characters.

The alphabetic characters in UNIBASIC are the upper-case letters of the alphabet.

The UNIBASIC numeric characters include the digits Ø through 9.

In addition, the following special characters and terminal keys are recognized by UNIBASIC:

CHARACTER	ACTION
	Blank or Space
=	Equals sign or assignment symbol
+	Plus sign
-	Minus sign
*	Asterisk or multiplication symbol
/	Slash or division symbol
^	Up arrow or exponentiation symbol
(	Left or open parenthesis
)	Right or close parenthesis
%	Percent
#	Number or pound sign
$	Dollar sign
!	Exclamation point or "bang"
[	Left or open bracket
]	Right or close bracket
,	Comma
.	Period
;	Semicolon
:	Colon
&	Ampersand or and sign
'	Single quotation mark (apostrophe)
?	Question mark
<	Less than
>	Greater than
@	At sign
"	Quotation mark
<BS>	Back Space key - deletes the last character typed
<ESC>	Escape key
<BREAK>	Break key
<CR>	Carriage Return keys (marked "Retrn" and "Enter")
<LINE_FEED>	Line feed key (Ctrl-L)

CONTROL CHARACTERS

UNIBASIC supports the following control characters:

CONTROL CHARACTER	ACTION
CTRL-A	Enters edit mode on the line being typed.
CTRL-C	Interrupts program execution and returns to UNIBASIC command level.
CTRL-G	Rings the bell at the terminal.
CTRL-H	Backspaces. Deletes the last character typed.
CTRL-I	Tabs to the next tab stop. Tab stops are set every eight columns.
CTRL-L	Line Feed. Moves the cursor down one line.
CTRL-M	Carriage Return. Moves the cursor to the left side of the screen
CTRL-O	Halts program output while execution continues. A second CTRL-O resumes output.
CTRL-R	Lists the line that is currently being typed.
CTRL-S	Suspends program execution.
CTRL-Q	Resumes program execution after a CTRL-S.
CTRL-U	Deletes the line that is currently being typed.

CONSTANTS

Constants are the values UNIBASIC uses during execution. There are two types of constants: string and numeric.

A string constant is a sequence of up to 255 alphanumeric characters enclosed in quotation marks(").

Examples

```
"HELLO"
"$25,000.00"
"Number of Employees"
```

Numeric constants are positive or negative numbers. UNIBASIC numeric constants cannot contain commas. There are five types of numeric constants:

1. Integer constants

   Whole numbers between -32,767 and 32,767. Integer constants do not contain decimal points.

2. Fixed-point constants

   Positive or negative real numbers, i.e., numbers that contain decimal points.

3. Floating-point constants

   Positive or negative numbers represented in exponential form (similar to scientific notation). A floating-point constant consists of an optionally signed integer or fixed-point number (the mantissa) followed by the letter E and an optionally signed integer (the exponent). The allowable range for floating-point constants is -1E38 to +1E38.

   Examples

   ```
   235.988E-7 = .0000235988
   2359E6     = 2359000000
   ```

   (Double precision floating-point constants are denoted by the letter D instead of E).

PRECISION FORM FOR NUMERIC CONSTANTS

Numeric constants are stored with up to 10 digits of precision. The tenth digit is rounded off.

Examples

46.8
-1.09E-06
3489.0
22.5

VARIABLES

Variables are names used to represent values used in a UNIBASIC program. The value of a variable may be assigned explicitly by the programmer, or it may be assigned as the result of calculations in the program. Before a variable is assigned a value, its value is assumed to be zero.

VARIABLE NAMES AND DECLARATION CHARACTERS

UNIBASIC variable names may be any length up to 238 characters. Only the first 8 characters are used by UNIBASIC. Variable names can contain letters and digits. However, the first character must be a letter. No special characters or reserved words may be used in a variable name.

Reserved words include all UNIBASIC commands, statements, function names, and operator names. If a variable begins with FN, it is assumed to be a call to a user defined function.

Variables may represent either a numeric value or a string. String variable names are written with a dollar sign ($) as the last character. For example: A$ = "SALES REPORT." The dollar sign is a variable type declaration character; that is, it "declares" that the variable will represent a string.

Numeric variable names may declare integer or single precision names and are denoted as follows:

% Integer variable

Single precision variable

Examples of UNIBASIC variable names:

MINIMUM	Declares a single precision value.
LIMIT%	Declares an integer value.
N$	Declares a string value.

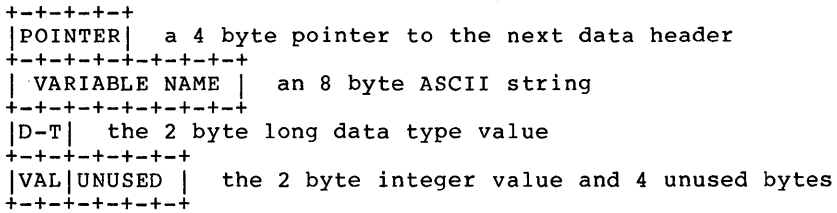
ARRAY VARIABLES

An array is a group or table of values referenced by the same variable name. Each element in an array is referenced by an array variable that is subscripted with an integer or an integer expression. An array variable name has many subscripts as there are dimensions in the array. For example, V(1Ø) would reference a value in a one-dimension array, T(1,4) would reference a value in a two-dimension array, and so on. The maximum number of elements per dimension is 32,767. The maximum number of dimensions is 88.

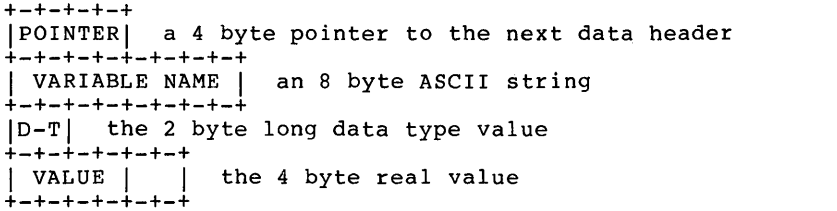
SPACE REQUIREMENTS

All UNIBASIC variables and arrays have a data header. The data headers are shown below for each data type:

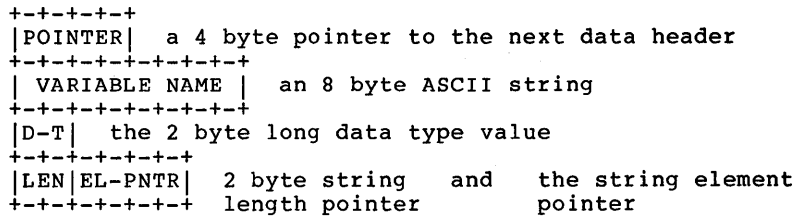
INTEGER



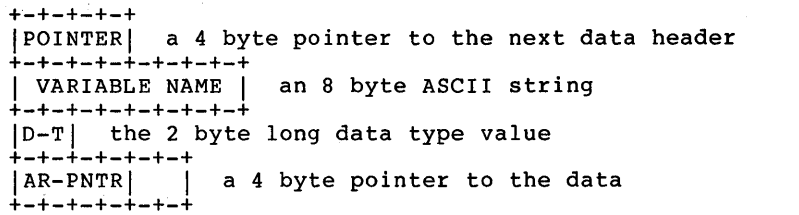
REAL



STRING

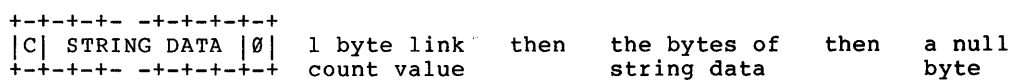


ARRAY



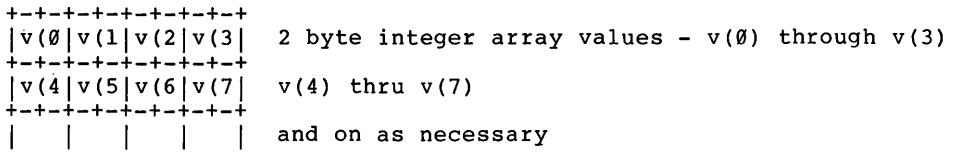
The string element has the following layout in memory:

STRING ELEMENT

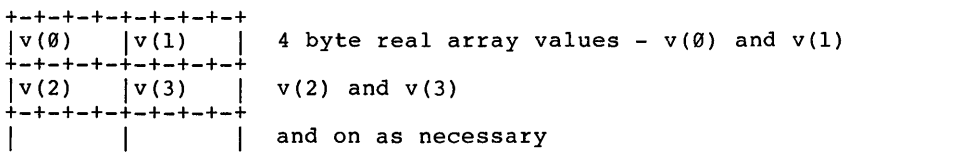


Array data is stored in the same sequence as APPLESOFT BASIC. The array data is stored as follows:

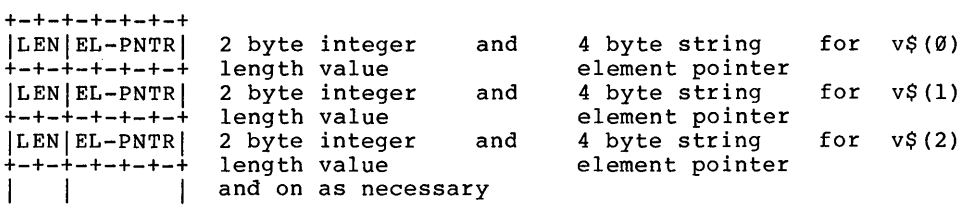
INTEGER ARRAY



REAL ARRAY



STRING ARRAY



TYPE CONVERSION

When necessary, UNIBASIC will convert a numeric constant from one type to another. The following rules and examples should be kept in mind.

1. If a numeric constant of one type is set equal to a numeric variable of a different type, the number will be stored as the type declared in the variable name. (If a string variable is set equal to a numeric value or vice versa, a "Type mismatch" error occurs.)

Example

```
10 A% = 23.42
20 PRINT A%
RUN
 23
```

2. When a floating-point value is converted to an integer, the fractional portion is rounded.

Example

```
10 C% = 55.88
20 PRINT C%
RUN
 56
```

3. Logical operators (see later in this section) convert their operands to integers and return an integer result. Operands must be in the range -32767 to 32767 or an "Overflow" error occurs.

EXPRESSIONS AND OPERATORS

An expression may be a string or numeric constant, a variable, or a combination of constants and variables with operators which produces a single value.

Operators perform mathematical or logical operations on values. The UNIBASIC operators may be divided into four categories:

1. Arithmetic

2. Relational

3. Logical

4. Functional

Each category is described in the following sections.

ARITHMETIC OPERATORS

The arithmetic operators, in order of precedence, are:

Operator	Operation	Expression
^	Exponentiation	X^Y
-	Negation	-X
*, /	Multiplication, Division	X*Y, X/Y
+, -	Addition, Subtraction	X+Y, X-Y

To change the order in which the operations are performed, use parentheses. Operations within parentheses are performed first. Inside parentheses, the usual order of operations is maintained.

OVERFLOW AND DIVISION BY ZERO

If, during the evaluation of an expression, division by zero is encountered, the following things occur:

- the "Division by zero" error message is displayed.
- the result is machine infinity with the sign of the numerator.
- execution continues.

If, during the evaluation of an exponentiation expression, zero being raised to a negative power is encountered, the following things occur:

- the "Division by zero" error message is displayed.
- the result is positive machine infinity.
- execution continues.

If overflow occurs, the following things occur:

- the "Overflow" error message is displayed.
- the result is machine infinity with the algebraically correct sign.
- execution continues.

RELATIONAL OPERATORS

Relational operators are used to compare two values. The result of the comparison is either "true" (-1) or "false" (Ø). This result may then be used to make a decision regarding program flow. (See "IF" statements in a later section.)

The relational operators are:

Operator	Relation Tested	Example
=	Equality	X=Y
<>	Inequality	X<>Y
<	Less than	X<Y
>	Greater than	X>Y
<=	Less than or equal to	X<=Y
>=	Greater than or equal to	X>=Y

(The equal sign is also used to assign a value to a variable. See "LET" statements in a later section.)

When arithmetic and relational operators are combined in one expression, the arithmetic is always performed first. For example, the expression

```
X+Y<(T-1)/Z
```

is true if the value of X plus Y is less than the value of T minus 1 divided by Z.

Examples

```
IF SIN(X)<Ø GOTO 1ØØØ
IF I**2<>Ø THEN K=K+1
```

LOGICAL OPERATORS

Logical operators perform tests on multiple relations, bit manipulation, or Boolean operations. The logical operator returns a bitwise result which is either "true" (not zero) or "false" (zero). In an expression, logical operations are performed after arithmetic and relational operations. The outcome of a logical operation is determined as shown in the table below. the operators are listed in order of precedence.

UNIBASIC
RELATIONAL OPERATORS TRUTH TABLE

NOT

X	NOT X
Ø	1
1	Ø

AND

X	Y	X AND Y
Ø	Ø	Ø
Ø	1	Ø
1	Ø	Ø
1	1	1

OR

X	Y	X OR Y
Ø	Ø	Ø
Ø	1	1
1	Ø	1
1	1	1

Just as the relational operators can be used to make decisions regarding program flow, logical operators can connect two or more relations and return a true or false value to be used in a decision (see "IF" statements in a later section).

Examples

```
IF C<250 AND F<5 THEN 80
IF I>10 OR K<0 THEN 50
IF NOT P THEN 100
```

Logical operators work by converting their operands to 16-bit, signed, two's complement integers in the range -32767 to 32767. (if the operands are not in this range, an error results.) If both operands are supplied as 0 or -1, logical operators return 0 or -1. The given operation is performed on these integers in bitwise fashion, i.e., each bit of the result is determined by the corresponding bits in the two operands.

Thus, it is possible to use logical operators to test bytes for a particular bit pattern. For instance, the AND operator may be used to "mask" all but one of the bits of a status byte at a machine I/O port. The OR operator may be used to merge two bytes to create a particular binary value. The following examples will help demonstrate how the logical operators work.

63 AND 16 = 16	63 = binary 111111 and 16 = binary 010000, so 63 AND 16 = 16 (binary 0010000)
15 AND 14 = 14	15 = binary 1111 and 14 = binary 1110, so 15 AND 14 = 14
-1 AND 8 = 8	-1 = binary 1111111111111111 and 8 = binary 1000, so -1 AND 8 = 8
4 OR 2 = 6	4 = binary 0100 and 2 = binary 0010, so 4 OR 2 = 6 (binary 0110)
10 OR 10 = 10	10 = binary 1010, so 1010 OR 1010 = 1010 (decimal 10)
-1 OR -2 = -1	-1 = binary 1111111111111111 and -2 = binary 1111111111111110, so -1 OR -2 = -1. The bit complement of sixteen zeroes is sixteen ones, which is the two's complement representation of -1
NOT X=-(X+1)	The two's complement of any integer is the bit complement plus one.

FUNCTIONAL OPERATORS

A function is used in an expression to call a predetermined operation that is to be preformed on an operand. UNIBASIC has "intrinsic" functions that reside in the system, such as SQR (square root) or SIN (sine). All UNIBASIC intrinsic functions are described in Chapter 3.

UNIBASIC also allows "user-defined" functions that are written by the programmer. See "DEF FN" in a later section.

STRING OPERATORS

Strings may be concatenated by using +.

Example

```
10 A$ = "FILE": B$ = "NAME"
20 PRINT A$+B$
30 PRINT "NEW"+A$+B$
RUN
FILENAME
NEWFILENAME
```

Strings may be compared using the same relational operators that are used with numbers:

```
=   <>   <   >   <=   >=
```

String comparisons are made by taking one character at a time from each string and comparing the ASCII codes. If all the ASCII codes are the same, the strings are equal. If the ASCII codes differ, the lower code number precedes the higher. If during string comparison, the end of one string is reached, the shorter string is said to be smaller. Leading and trailing blanks ARE significant.

Examples

```
"AA"<"AB"
"FILENAME"="FILENAME"
"X&">"X#"
"CL">"CL"
"kg">"KG"
"SMYTH"<"SMYTHE"
B$<"9/12/78" where
B$="8/12/78"
```

Thus, string comparisons can be used to test string values or to alphabetize strings. All string constants used in comparison expressions must be enclosed in quotation (") marks.

HIGH RESOLUTION GRAPHICS

There are two pages of high resolution graphics. The user selects the page desired by issuing either a PAGE#1 command or a PAGE#2 command.

SHAPE TABLE

The shape table begins at address 4000 decimal. The shape table has a default size of 500 bytes. The shape table size can be changed by using the SHSIZE command. The shape table is loaded either from a disk file by using the SHLOAD command or by POKEing the values in starting at address 4000 decimal. The shape table can be saved into a disk file by using the SHSAVE command.

INPUT EDITING

If an incorrect character is entered as a line is being typed, it can be deleted with the <Back Space> (<BS>) key or with CONTROL-H. Both the <BS> key and CONTROL-H have the effect of backspacing over a character and erasing it. Once a character(s) has been deleted, simply continue typing the line as desired.

To delete a line that is in the process of being typed, type CONTROL-U. A carriage return is executed automatically after the line is deleted.

To correct program lines for a program that is currently in memory, simply retype the line using the same line number. UNIBASIC will automatically replace the old line with the new line.

More sophisticated editing capabilities are provided. See the "EDIT" statement in a later section.

To delete the entire program currently residing in memory, enter the NEW command. (See the "NEW" command in a later section.) NEW is usually used to clear memory prior to entering a new program.

ERROR MESSAGES

If an error causes program execution to terminate, an error message is printed. For a complete list of UNIBASIC error codes and error messages, see the APPENDICES.

C H A P T E R 2

U N I B A S I C C O M M A N D S

CHAPTER 2

EDITING - alt arrow

Syntax

```
alt ->
alt <-
alt  V
alt  ^
```

Description

These commands do not affect characters moved over by the cursor: the characters remain both on the TV screen and in memory. By themselves, these commands do not affect the program line being typed.

```
alt ->      moves the cursor one space to the right
alt <-      moves the cursor one space to the left
alt  V      moves the cursor one space down
alt  ^      moves the cursor one space up
```

Parameters

None.

Notes

- To change a program line: LIST the line on the screen and use the alt arrow commands to place the cursor over the first character of the line. Use the right-arrow key to move across the line, stopping at characters you wish to change and entering the desired character. When you are finished changing the line, press RETURN to store or execute the corrected line. If you did not use LIST to display the line, do not copy the prompt character (:).

- The alt arrow commands may be used in the immediate execution mode only.

Error Messages

Examples

Caveat

<BREAK> key

Syntax — break

Description — break interrupts the current process immediately after the statement that is currently being executed.

Parameters — None.

Notes

- break may be entered to interrupt an INPUT or GET but must be the first character entered. The interruption occurs when return is pressed for INPUT and immediately for GET.
- BREAK IN line-number is displayed a program is executing.
- break may be used in the deferred execution mode only.

Error Messages

Examples

Caveat

ALOAD

Syntax — ALOAD filename

Description — This command loads a UNIBASIC program from an ASCII file, and converts the ASCII program to UNIBASIC internal format.

Parameters — Filename is the name of the file in UNIBASIC format as described in the UNIBASIC USER'S GUIDE.

Notes

Error Messages

Examples

Caveat

ASAVE

Syntax	ASAVE filename
Description	This command saves a UNIBASIC program on diskette as an ASCII file.
Parameters	Filename is the name of the file in UNIBASIC format as described in the UNIBASIC USER'S GUIDE.
Notes	
Error Messages	
Examples	
Caveat	

APEEK

Syntax	APEEK (address)
Description	This command reads a byte out of the 68000 memory at the absolute address.
Parameters	Address is an absoulute address in the 68000 memory. It is a 32 bit integer value.
Notes	
Error Messages	
Examples	
Caveat	

APOKE

Syntax — APOKE address, data

Description — This command is used to place a byte of information, data, into the 68000 memory at the absolute location specified by address.

Parameters — Address is the absolute location in 68000 memory that is desired to have information placed into. It is a 32 bit integer.

Data is the byte of information that is to be placed into 68000 memory.

Notes

Error Messages

Examples

Caveat

AT

Syntax

```
DRAW arithexpl AT arithexp2, arithexp3
HLIN arithexpl, arithexp2 AT arithexp3
VLIN arithexpl, arithexp2 AT arithexp3
XDRAW arithexpl AT arithexp2, arithexp3
```

Description

See description of the commands for DRAW, HLIN, VLIN, and XDRAW

Parameters

Notes

Error Messages

Examples

Caveat

CALL

Syntax

CALL address[(arg1,,,arg14)]

Description

This command allows UNIBASIC to "call" an assembly language subroutine.

Parameters

Address is the 32 bit integer absolute location in 68000 memory where the assembly language subroutine is residing.

Arg1 is the 32 bit real number used to pass data to the assembly language subroutine.

Notes

- There can be up to 14 arguments
- Address & arguments can be arithmetic-expressions.
- When CALL is used as a function, the value is returned in the D0 register.

Error Messages

Examples

Caveat

CATALOG

Syntax

CATALOG[,Dn]

Description

This command causes a list of the contents of the directory of the disk drive specified to be displayed on the screen.

Parameters

n is the number of the disk drive that the directory is to be displayed for. The following is a correlation of disk drive numbers and CP/M-68K drive specifiers:

DRIVE NUMBER	CP/M-68K DRIVE
1	A:
2	B:
3	C:
4	D:

If no disk drive is specified, then the disk drive that was most recently accessed will be used.

Notes

Error Messages

Examples

Caveat

COLOR

Syntax

COLOR = arithexpr

Description

Sets the color for plotting in low resolution graphics mode.

Parameters

The range of values for arithexpr is from Ø through 255.

Color numbers and their associated names are:

Ø black	4 dark green	8 brown	12 green
1 magenta	5 grey	9 orange	13 yellow
2 dark blue	6 medium blue	1Ø grey	14 aqua
3 purple	7 light blue	11 pink	15 white

COLOR evaluates arithexpr modulo 16 to return a value in the range of Ø to 15.

Notes

- In high-resolution graphics mode COLOR has no meaning.

- See SCRN and PLOT for more information.

Error Messages

Examples

Caveat

When used in TEXT mode with PLOT, COLOR will affect which character the PLOT instruction places in the text window.

CLEAR

Syntax

CLEAR

Description

CLEAR zeros (clears) all variables, strings, and arrays.

Parameters

None.

Notes

- CLEAR places zeros in numeric variables and nulls (nothing) in string variables.
- CLEAR may be used in either the immediate or deferred execution mode.

Error Messages

Examples

Caveat

CONT

Syntax

CONT

Description

If a STOP, END, or break command halts a program, CONT will resume program execution at the next instruction, but not necessarily the next line number. Memory is not cleared or changed.

Parameters

None.

Notes

- If no program has been halted, then CONT has no effect.

- CONT may only be used in the immediate execution mode.

Error Messages

If the user changes or deletes a program line or causes an error message when a program has halted, the ?CAN'T CONTINUE ERROR message is displayed.

When the DEL command is used in deferred execution mode, the specified lines are deleted and program execution stops. If CONT is entered, the ?CAN'T CONTINUE MESSAGE will be displayed.

Examples

Caveat

DATA

Syntax

DATA [datavalue] [{,[datavalue]}]
note: datavalue = [literal string real integer]

Description

Creates a list of elements which can be used by READ statements.

Parameters

DATA elements may be any mixture of reals, integers, strings and literals.

"Non-existent" (zero or null string) data elements occur when any of the following are true:

1) There is no non-space character between DATA and return.
2) Comma is the first non-space character following DATA.
3) There is no non-space character between 2 commas.
4) Comma is the last non-space character before return.

Notes

- Each DATA statement adds data elements to the list of elements built up by the program's previous (lower line number) DATA statements.

- DATA statements may appear anywhere throughout a program. They do not have to precede the READ statements.

- The rules for DATA elements READ into variables are the same as the rules for INPUT responses assigned to variables with 1 exception. The colon cannot be included as a character in a numeric DATA element.

- See INPUT for more details.

- DATA may be used in deferred execution mode only.

Error Messages

?SYNTAX ERROR
For any quotation mark appearing within a string. Or for an attempt to assign a string or literal DATA element to a numeric variable.

DATA (cont'd)

Examples

```
100 DATA,,
```

When READ, this statement may return up to 3 elements consisting of zeros or null strings.

```
10 DATA ,4,100,99
20 READ A,B,C
30 PRINT A,B,C
RUN

 0             4             100

120 DATA "Commas , may appear , in  quoted  strings"
```

Caveat

When used in immediate execution mode, DATA does not cause a SYNTAX ERROR, but the data elements are not available to READ statements.

DEF FN

Syntax

DEF FN name (dummyvariable) = arithexprl
FN name (arithexpr2)

Description

Defines functions in a program. Functions may be used wherever arithmetic expressions may be used. After the execution of a program line containing DEF, the DEFined function may be used in the form

FN name (argument)

where the argument may be any arithmetic expression.

Parameters

The rules for using arithmetic variables apply to function names (the first 8 characters must be unique).

Arithexprl may be only 1 program line in length.

Dummyvariable must be a real number arithmetic variable.

FN substitutes the argument for dummyvariable wherever dummyvariable appears in the DEFinition. Arithexprl may contain any number of variables. At most 1 of those variables corrresponds to dummyvariable.

Notes

- The DEFinition's dummyvariable need not appear in arithexprl. In that case, when the function is used, the function's argument is ignored in evaluating arithexprl. The function's argument must always be legal.

- Functions may be redefined during the course of a program.

- When a new function is defined by a DEF statement, 6 bytes in memory are used to store the pointer to the definition.

- DEF may be used in deferred execution mode only. FN may be used in deferred or immediate execution mode.

DEF FN (cont'd)

Error Messages

?UNDEFN'D FUNCTION ERROR
If a deferred execution DEF FN name statement is not executed prior to using FN name.

Examples

```
100 DEF FN A(W) = 2 * W + W
110 PRINT FN A(23)
120 DEF FN B(X = 4 + 3
130 G = FN B(23)
140 PRINT G
150 DEF FN A(Y) = FN B(Z) + Y
160 PRINT FN A(G)

RUN
69                      [ FN A(23) = 2 * 23 + 23 ]
7                       [ FN B(anything) = 7     ]
14                      [ new FN A(7) = 7 + 7    ]

10 DEF FN ABC(I) = COS(I)
20 DEF FN ABC(I) = TAN(I)
```

The function AB is defined in line 10 and then redefined in line 20.

Caveat

User-defined string functions are not allowed.

Functions defined using an integer name (name%) for the function name or for dummyvariable are not allowed.

If CLEAR, NEW, DEL, or RUN destroys or skips a DEFined function in memory, the function may not be defined.

DISK OPERATIONS

Syntax

PRINT D$;"operation [,filenm] [,specification]"

Description

Disk commands are given from print statements. The first character of the print statement must have a decimal value of 4 (CHR$(4) or CNTRL D).

Parameters

Various literal strings, string variables, or arithmetic variables are included as PRINT parameters to form the desired command.

D$ must equal CHR$(4) (CONTROL D).

Operation is one of the operations described below.

Filenm is a filename in CPM format.

Specification is a capital letter followed by a decimal number. Dn is an optional diskdrive specification. If Dn is omitted, the default is the last used drive.
n = 1 is drive A
n = 2 is drive B
n = 3 is drive C
n = 4 is drive D

Notes

The following disk operations are available:

OPEN Opens the specified file if it exists, or creates the file if it does not exist. A maximum of 16 files may be open at any one time. The function NF# may be called to determine if the file was created.
EXAMPLE:

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

CLOSE Closes the named disk file, or all disk files.
EXAMPLE:

```
PRINT D$;"CLOSE FILENM"
```
Closes FILENM.

```
PRINT D$;"CLOSE"
```
Closes all disk files.

DISK OPERATIONS (cont'd)

READ Causes INPUT and GET statements to read statements from the named file. If the file was previously being written, then it is closed and re-opened. READing starts at the beginning of the file.
EXAMPLE:

```
PRINT D$;"READ FILENM"
```

WRITE Causes PRINT statements to write to the specified file. If the previous command for the file was APPEND, the writing starts at the end of the file. If not, writing starts at the beginning of the file. If file was being read, then it is closed and writing starts at the beginning.
EXAMPLE:

```
PRINT D$;"WRITE FILENM"
```

APPEND Used in place of the OPEN command. Subsequent write statements will start at the end of the file. Dn is the optional disk drive specification.
EXAMPLE:

```
PRINT D$;"APPEND FILENM,D2"
```

RENAME Changes the name of a disk file. Dn is the optional disk drive specification.
EXAMPLE:

```
PRINT D$;"RENAME OLDNM,NEWNM,D4"
```

DELETE Removes a file from disk. Dn is the optional disk drive specification.
EXAMPLE:

```
PRINT D$;"DELETE FILENM,D3"
```

BSAVE Saves binary data from memory, such as a display screen, on disk. Dn is the optional disk drive specification. An is the required decimal data address. Ln is the required decimal data length. The order of the options is insignificant.
EXAMPLE:

```
PRINT D$;"BSAVE FILENM,A4000,L1024,D2"
```

DISK OPERATIONS (cont'd)

BLOAD Reloads binary data from disk to memory. Specifications are the same as BSAVE. If An and Ln are omitted, then BLOAD uses values from BSAVE on the file.
EXAMPLE:

```
PRINT D$;"BLOAD FILENM,D2"
```

Error Messages

Examples See Above

Caveat

DEL

Syntax — DEL line-number-1, line-number-2

Description — DEL deletes program lines line-number-1 to line-number-2, inclusive.

Parameters — line-number-1 is a program line number and specifies the first line to be deleted.

line-number-2 is a program line number and specified the last line to be deleted.

Notes

- Both line-number-1 and line-number-2 are required.
- If line-number-1 does not exist in the program, the next greater line number in the program is used instead of line-number-1; if line-number-2 does not exist in the program, the next smaller program line number is used.
- When used in deferred execution, DEL works as described above, then halts execution. CONT will not work in this situation.
- DEL may be used in either immediate or deferred execution mode.

Error Messages

Examples — DEL 7,9 deletes program line numbers 7 through 9.

Caveat

DIM

Syntax

DIM variable (subscript [{, subscript}])

Description

The DIM statement declares and reserves memory space for an array and clears all elements.

Parameters

variable is an array name.

subscript specifies the largest numbered element of a dimension. Note that subscript numbering begins with Ø.

Notes

- DIM reserves 8 bytes for the array's variable name, 4 bytes for linkage, 4 for data pointer, 2 for the number of dimensions, and 2 for each dimension.

- To determine the number of elements in an array, add 1 to each subscript parameter and multiply the resulting numbers.

- The maximum number of dimensions for an array may not exceed 88, regardless of the number of elements in each dimension.

- The amount of memory available when DIM is executed will limit the size of arrays. An integer array element will occupy 2 bytes in memory; a real array element will occupy 8 bytes. A string array element will occupy 6 bytes for each element including 2 for the length and 4 for a location pointer. As characters are stored they occupy one byte.

- If an array element is referenced before the array is DIMensioned, Unibasic assigns a maximum subscript of 1Ø for each dimension referenced by the statement.

- String arrays are allocated memory dynamically, as the strings are created and assigned by the program. The arrays will become larger or smaller depending on the string assigned.

- The RUN and CLEAR commands reset numeric array elements to Ø, and string array elements to null.

- Do not attempt to DIMension an array more than once.

- DIM may be used in either the immediate or deferred execution mode.

DIM (cont'd)

Error Messages

Referencing a subscript larger than the maximum declared for a variable will cause the ?BAD SUBSCRIPT ERROR to be displayed.

If an array variable is used which requires a different number of dimensions than reserved by the DIM statement, the ?BAD SUBSCRIPT ERROR is displayed.

An attempt to DIMension an array variable which already exists causes the ?REDIM'D ARRAY ERROR, even if the first array was dimensioned by default.

Examples

DIM NEW (2,3,4) reserves 3*4*5 (60) elements for the array named NEW.

DIM STARRAY$(ROW,COLUMN) reserves ROW+1 * COLUMN+1 array elements for the string variable named STARRAY$.

Caveat

DRAW

Syntax

```
DRAW shapenumber AT X, Y
DRAW shapenumber
```

Description

This command will draw the shape specified, in the high-resolution GRaphics mode, on the screen. If no X and Y co-ordinates are specified by the AT clause, then the shape will be drawn at the last point plotted.

Parameters

Shapenumber is the number of the shape, in the shape table, to be drawn. Shapenumber must be in the range of Ø through n, where n is the number of shape definitions in byte Ø of the shape table (max = 278).

X is the horizontal co-ordinate arithmetic expression.

Y is the vertical co-ordinate arithmetic expression.

Notes

Error Messages

Examples

Caveat

END

Syntax	END
Description	END stops program execution and returns control to the user.
Parameters	None.
Notes	- No BREAK IN message is displayed. - END may be used in either the immediate or deferred execution mode.
Error Messages	
Examples	
Caveat	

FLASH

Syntax

FLASH

Description

FLASH sets the output video mode so that the computer's output is black letters on a white background. This the same as reverse video.

Parameters

None.

Notes

- FLASH does not effect the display of characters as you type them into the computer nor characters already on the screen.

- FLASH may be used in either the immediate or deferred execution mode.

Error Messages

Examples

Caveat

FN

Syntax	FN name (arithexpr2)
Description	See DEF FN name (arithexpr1)
Parameters	Arithexpr2 may be any valid arithmetic expression.
Notes	- FN may be used in immediate or deferred execution mode.
Error Messages	?UNDEF'D FUNCTION ERROR If the function has not been DEFined yet.
Examples	
Caveat	

FOR

Syntax

FOR realvar = aexpr1 TO aexpr2 [STEP aexpr3]

Description

Controls looping in a program. The lines of the program between the FOR statement and the corresponding NEXT statement comprise the body of the loop.

Realvar is incremented by aexpr3 and compared to aexpr2 at the bottom of the FOR...NEXT loop. Aexpr3 is optional. It defaults to 1.

The portion of the program inside the loop executes at least once.

Realvar is set to aexpr1, and the statements following the FOR are executed until a statement

NEXT realvar

is encountered.

Realvar is incremented by aexpr3 and compared to aexpr2. If realvar is greater than aexpr2, then execution proceeds with the statement following the NEXT. If realvar is less than or equal to aexpr2, execution proceeds from the statement following the FOR.

If aexpr3 is less than Ø then operation is different after it is added to realvar. If realvar is less than aexpr2, execution proceeds with the statement following the NEXT. If realvar is greater than or equal to aexpr2, the loop is repeated.

Parameters

The arithmetic expressions (aexpr) which form the parameters of the FOR loop may be reals, real variables, integers, or integer variables. Realvar must be a real variable.

Notes

- Each active FOR...NEXT loop uses 52 bytes in memory.

- FOR...NEXT loops may be used in immediate or deferred execution mode.

- To run a FOR...NEXT loop in immediate-execution mode, the FOR statement and the NEXT statement should be included in the same line (a line is up to 239 characters long).

FOR (cont'd)

Error Messages

?SYNTAX ERROR
For attempting to use an integer variable for realvar.

?NEXT WITHOUT FOR ERROR
If FOR...NEXT loops cross each other.

See NEXT for more information.

Examples

Caveat

If the letter A is used immediately prior to TO, do not allow a space between the T and O.

FOR I=BETA TO 56

is fine, but

FOR I=BETA T O 56

parses as

FOR I=BET AT Ø56

and generates a

?SYNTAX ERROR

on execution.

GET

Syntax — GET var

Description — Fetches a single character from the keyboard. The user is not required to press the return key. The character is not displayed on the screen.

Parameters — Accepts string or numeric variables.

Notes

- Ctrl @ returns the null character (ASCII Ø).
- Break interrupts program execution.
- If var is a numeric variable, then +, -, ctrl @, E, space, colon, comma, and the period return zero as the typed value.
- GET may be used in deferred execution mode only.

Error Messages

?EXTRA IGNORED
For issuing a GET which receives a colon or comma for a numeric variable.

?SYNTAX ERROR
For typing a return or non-numeric input for a numeric variable.

Examples

Caveat — Because of the limitations on numeric variables, it is recommened that programmers GET numbers using string variables (GET stringvar). Convert the resulting string to a number using the VAL function.

GOSUB

Syntax

GOSUB linenumber

Description

The program branches to the indicated line. When a RETURN statement is executed, the program branches to the statement immediately following the most recently executed GOSUB.

Parameters

Linenumber must correspond to a line in the program.

Notes

- Each active GOSUB (that has not RETURNed yet) uses 14 bytes of memory. Each time a GOSUB is executed, the address of the following statement is stored on top of a "stack" of return addresses. The program may return to the correct statement by looking on the top of the stack. A RETURN or a POP removes the top address of the RETURN stack.

- GOSUB may be used in immediate or deferred execution mode.

- If used in immediate mode, the program branches to linenumber. Variables are kept. On RETURN the program halts like END.

- See RETURN and POP for more information.

Error Messages

?UNDEFINED STATEMENT ERROR IN linenumber
If the linenumber of GOSUB does not correspond to an existing program line. IN linenumber indicates the program line containing the GOSUB statement.

Examples

```
10Ø DATA 12
11Ø READ A
12Ø GOSUB 5ØØ
13Ø PRINT A
14Ø END
5ØØ A = A*5
51Ø RETURN
RUN
6Ø
```

Caveat

GOTO

Syntax

GOTO linenumber

Description

Branches to the line whose number is linenumber.

Parameters

Linenumber must be the number of a line in the program.

Notes

- GOTO may be executed in immediate or deferred execution mode.

Error Messages

?UNDEF'D STATEMENT ERROR IN number
If there is no such line with linenumber, or if linenumber is absent from the GOTO statement. Number indicates the number of the line containing the GOTO statement.

Examples

```
10 A = 0
20 GOTO 40
30 A = 99
40 PRINT A
RUN
0
```

Caveat

GR

Syntax

GR

Description

Sets the screen for low-resolution GRaphics mode (80 by 40).

Leaves a 4 line text window at the bottom of the screen.

Clears the screen to black and moves the cursor to the text window.

Sets COLOR to zero.

Parameters

None.

Notes

- See MODE#, HLIN, VLIN, COLOR and TEXT for more information.

Error Messages

Examples

Caveat

HCOLOR

Syntax — HCOLOR = arithmetic-expression

Description — This command sets the high-resolution GRaphics color to that specified by the value of arithmetic-expression.

Parameters — The range of values for arithmetic-expression is from Ø through 255.

Notes — - In the low-resolution graphics mode, HCOLOR has no meaning.

Error Messages

Examples

Caveat

HGR

Syntax

HGR

Description

Sets the screen for High-resolution GRaphics mode (180 by 160).

Displays the bottom 4 lines of the text window below the graphics.

Clears the screen to black and displays page 1 of memory.

Parameters

None.

Notes

- HCOLOR is not changed.

- Text screen memory is not affected.

- Leaves the text "window" at full screen, but only the bottom 4 text lines are visible below the graphics. The cursor will still be in the text "window", but may not be visible unless moved to 1 of the bottom 4 lines.

- See MODE#, PAGE#, GR, HGR2, TEXT and COLOR for more information.

Error Messages

Examples

Caveat

If the reserved word HGR is used as the first characters of a variable name, the HGR may be executed before the

?SYNTAX ERROR

appears. Executing the statement

HGRIP=4

sets high-resolution graphics mode, which may erase the program.

HGR2

Syntax

HGR2

Description

Sets the screen to full-screen High-resolution GRaphics mode (280 by 192).

Clears the screen to black and displays page 2 of memory.

Parameters

None.

Notes

- No portion of the text screen memory is displayed.

- HCOLOR is not changed.

- Text screen memory is not affected.

- See MODE#, PAGE#, GR, HGR, TEXT and HCOLOR for related information.

Error Messages

?SYNTAX ERROR
If the reserved word HGR2 appears as the first characters of a variable name.

Examples

Caveat

If the reserved word HGR2 is used as the first characters of a variable name, the HGR2 may be executed before the

?SYNTAX ERROR

appears. Executing the statement

140 IF X > 150 THEN HGR2PIECES = 12
clears the screen, and may erase part of the program.

HLIN

Syntax

HLIN X1, X2 AT Y

Description

In low-resolution GRaphics mode, draws a Horizontal LINe from (X1, Y) to (X2, Y).

Parameters

X1 and X2 are arithmetic expressions in the range 0 to 79. Y is an arithmetic expression in the range 0 to 47.

Notes

- HLIN has no visible effect when used in high-resolution graphics mode.

- The "H" in "HLIN" refers to horizontal, not high-resolution. Except for HLIN and HTAB, the prefix "H" refers to a high-resolution instruction.

- See GR, MODE# and COLOR.

Error Messages

?ILLEGAL QUANTITY ERROR
If X1 or X2 are less than 0 or greater than 79, or if Y is less than 0 or greater than 47.

Examples

Caveat

If HLIN is used on a TEXT window, a line of characters is placed where the line of graphic dots would have been plotted. (A character occupies the space of 2 low-resolution dots stacked vertically.)

HOME

Syntax — HOME

Description — HOME moves the cursor to the upper left screen position within the scrolling window and clears all text within the window.

Parameters — None.

Notes — - HOME may be used in either the immediate or deferred execution mode.

Error Messages

Examples

Caveat

HPLOT

Syntax

```
HPLOT X1, Y1
HPLOT TO X2, Y2
HPLOT X1, Y1 TO X2, Y2
```

Description

This command will draw a high-resolution dot or line. If only X1 and Y1 are specified, then a dot will be drawn. If only X2 and Y2 are specified, then a line will be drawn from the last point plotted to the co-ordinates specified. If both the X1, Y1 and X2, Y2 co-ordinates are specified, then a line will be plotted from the X1, Y1 co-ordinates to the X2, Y2 co-ordinates.

Notes

Error Messages

Examples

Caveat

HTAB

Syntax — HTAB arithmetic-expression

Description — HTAB moves the cursor horizontally from the extreme left of the current screen line.

Parameters — arithmetic-expression is a cursor screen position and must be in the range 1 through 255.

Notes

- Assume the line in which the cursor is located has 255 positions, 1 through 255. Regardless of the text window width you may have set, positions 1 through 8Ø are on the current line, positions 81 through 16Ø are on the next line down, and so on.
- HTAB's moves are relative to the left margin of the text window, but independent of the line width.
- HTAB can move the cursor outside the text window, but only long enough to PRINT one character.
- To place the cursor in the leftmost position of the current line, use HTAB 1.
- The effects of HTAB and VTAB are not parallel. HTABs beyond the right edge of the screen do not cause an error message. The cursor jumps to the lower line and tabs to ((arithmetic-expression) MOD 8Ø) + 1.
- HTAB may be used in either immediate or deferred execution mode.

Error Messages — If arithmetic-expression is out of range, the ?ILLEGAL QUANTITY ERROR message is displayed.

Examples — HTAB NUM moves the cursor to the horizontal position specified by the variable called NUM.

Caveat

IF

Syntax

```
IF expr THEN instruction [{: instruction }]
IF expr THEN [GOTO] linenumber
IF expr [THEN] GOTO linenumber
```

Description

Compares an expression. If expr is true, then statements following the THEN are executed. If expr is false execution passes on to the instruction in the next numbered line of the program.

Parameters

If expr is an arithmetic expression whose value is not zero (and whose absolute value is greater than about 2.93873E-39), expr is considered to be true.

If expr is an arithmetic expression whose value is zero (or whose absolute value is less than about 2.93873E-39), expr is considered to be false.

If expr is an expression involving string expressions and string logical operators, expr is evaluated by comparing the alphabetic ranking of the string expressions as determined by the ASCII codes for the characters involved (see Appendix K).

UNIBASIC compares strings character by character. If 2 strings contain the same characters, and they are the same length, then the 2 strings are equal. If string1 is equal to string2 when compared character by character, but string1 has more characters than string2, then string1 is greater than string2. The first mismatched character determines which string is greater.

Notes

- IF may be used in deferred execution mode only.

Error Messages

?SYNTAX ERROR
Causes by a THEN without a corresponding IF or an IF without a corresponding THEN. Caused by trying to execute IF in immediate execution mode.

Examples

This statement

```
IF A THEN A=B
```

compares A to Ø (zero).

IF (cont'd)

These are equivalent:

```
IF A=3 THEN 16Ø
IF A=3 GOTO 16Ø
IF A=3 THEN GOTO 16Ø
```

Caveat

Before THEN, the letter A causes parsing problems:

```
IF BETA THEN 23Ø
```

parses to

```
IF BET AT HEN23Ø
```

which generates a

```
?SYNTAX ERROR
```

message on execution.

IN#

Syntax — IN# arithexpr

Description — Specifies which peripheral will provide input for subsequent INPUT statements.

Parameters — IN# Ø
Indicates that subsequent input will be from the keyboard instead of the peripheral. Slot Ø is not addressable from UniBasic for use with a peripheral device. All values other than Ø are illegal.

Notes
- IN# may be used in immediate or deferred execution mode.
- See MODE#.

Error Messages — ?ILLEGAL QUANTITY ERROR
If arithexpr is not equal to zero.

Examples

Caveat

INPUT

Syntax

INPUT [promptstring ;]var [{, var,...,var}]

Description

This command reads values from the keyboard. It waits for the user to type a number (if var is an arithmetic variable) or characters (if var is a string variable). The value of the number or the string is placed into var.

Parameters

Promptstring must be a literal, if it is present. It must appear directly after "INPUT" and be followed by a semi-colon. Promptstring prints exactly as specified. No question mark, no spaces or blanks, nor other punctuation is printed after promptstring. If promptstring is used, only one promptstring may be used.

If promptstring is left out, then a question mark is printed.

Successive variables get successively typed values.

String variables and arithmetic variables may be mixed in the same INPUT statement, but the user's responses must be of the appropriate types, that is the user must respond with a number for an arithmetic variable and a character string for a string input.

Notes

Responses must be separated by colons or commas. A colon or a comma as the first INPUT response evaluates as a zero or a null string. Break will interrupt an INPUT statement only if it is the first key typed.

NUMERIC VARIABLES

INPUT accepts only real or integer as numeric input. Arithmetic expressions are invalid. The characters +, -, space, E, and period are legitimate parts of numeric input. INPUT accepts any combination of these characters in acceptable form (e.g. +E- is acceptable, +- is not). Such input, by itself, evaluates as Ø. Spaces in any position are ignored. If a colon or a comma is the first character, the response evaluates to zero.

INPUT (cont'd)

STRING VARIABLES

A response assigned to a string variable must be a single string or literal, not a string expression. Spaces, or blanks, preceding the first character are ignored. Within a string, all characters, except the quotation mark, are accepted as input, except the first non-blank character. Spaces following the last character are accepted as part of the literal. The comma and the colon are not accepted as characters in the literal. If the return key <CR> alone is pressed, the response is interpreted as a null string.

INPUT may be used in deferred execution mode only.

Error Messages

If a carriage return is encountered before all the var's have been assigned, then the system will display
??

If the response contains more fields than the statement expected, or if a colon exists in the final expected response (but not within a string), the system will display
?EXTRA IGNORED

For numeric input which is not a real, an integer, a comma, or a colon, and for string input containing a quotation mark, the system will display
?REENTER

If attempting to CONTinue execution after the program has been halted by a 'break', the system will display
?SYNTAX ERROR

Examples

```
1ØØ INPUT "WHAT IS YOUR NAME? ";NAME$
11Ø PRINT "HELLO "; NAME$; " !"
RUN
WHAT IS YOUR NAME? MIKE
HELLO MIKE !
```

Caveat

INVERSE

Syntax — INVERSE

Description — This command sets the video output mode so that the display shows as black letters on a white background, instead of the normal white letters on a black background.

Parameters — None.

Notes

- INVERSE does not affect the display of characters as you type them into the computer, nor does it effect characters already on the screen.
- INVERSE will NOT inverse lower-case letters.
- INVERSE may be used in either the immediate or the deferred execution mode.

Error Messages

Examples

Caveat

LET

Syntax

```
[LET] arithvariable[subscript] = arithexpr
[LET] stringvariable[subscript] = stringexpr
```

Description

This command assigns the value of the expression on the right of the equals sign to the variable named on the left.

Parameters

Arithmetic values may only be assigned to arithmetic variables. String values may only be assigned to string variables.

Notes

LET may be used in either immediate or deferred execution mode.

Error Messages

```
?TYPE MISMATCH ERROR
for assigning:
1) an arithmetic expression to a string variable name.
2) a string variable name to a literal ("DOG" = A$).
3) a string expression to an arithmetic variable name.
```

Examples

```
LET A=2
and
A=2
are equivalent.
```

Caveat

If a literal assigned to an arithmetic variable name, it is parsed as an arithmetic expression.

LIST

Syntax

```
LIST [line-number-1] [- line-number-2]
LIST [line-number-1] [, line-number-2]
```

Description

This command displays lines in a program on the screen.

Parameter

Line-number-1 is a program line number and it specifies the first program line to be displayed.

Line-number-2 is a program line number and it specifies the last program line to be displayed.

Notes

- If no parameters are specified, the entire program is displayed.
- If line-number-1 is a Ø and line-number-2 is not specified, the entire program is displayed.
- If line-number-1 is specified without a delimiter, or if line-number-1 = line-number-2, then just the line numbered line-number-1 is displayed.
- If line-number-1 and a delimiter are specified, then the program is listed from line-number-1 through the end.
- If a delimiter and line-number-2 are specified, then the program is listed from the beginning through the line numbered line-number-2.
- If line-number-1, a delimiter, and line-number-2 are all present, then the program is listed from the line numbered line-number-1 through the line numbered line-numbered-2, inclusive.
- If more than one line is listed, and line-number-1 in the LIST command does not exist in the actual program, then the LIST command will use the next greater line that does exist.
- If line-number-2 in the LIST command does not exist in the program, then the LIST command will use the next smaller line number that does exist.

LIST (cont'd)

- Since UNIBASIC "tokenizes" your program lines before storing them, thus removing unnecessary spaces in the process, the LIST command "reconstitutes" the tokenized program lines, adding spaces according to its own rules. For example

```
10 C=+5/-6:B=-5
```

  becomes

```
10 C = + 5 / - 6:B = - 5
```

  when listed.

- LIST may be used in either immediate or deferred execution mode.

Error Messages

Examples

LIST displays the entire program.

LIST 10-50 displays those program lines numbered 10 through 50.

LIST ,45 displays all program lines from the begining of the program through line number 45.

Caveat

LOAD

Syntax	LOAD filename
Description	This command causes UNIBASIC to attempt to "load" into memory the filename specified from disk as a UNIBASIC program.
Parameters	Filename is the name of a disk file in the form specified in the UNIBASIC USER'S GUIDE.
Notes	
Error Messages	
Examples	
Caveat	

MODE#

Syntax MODE# modenumber

Description This command selects various graphics and text screen options, based on the value of modenumber. This statement is executed instead of PEEKing and POKEing.

Parameters Modenumber has the following options and values:

Mode#	Option
Ø	Initializes video to 8Ø columns by 24 lines MODE#Ø : TEXT :
1	Reset the 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 the ERROR FLAG to ON
3	Set COLOR to OFF LSB of COLOR BYTE = Ø for BLACK & WHITE MODE#3 :
4	Set COLOR to ON LSB of COLOR BYTE = 1 for COLOR MODE#4 :
5	Mixed Graphics and Text for TEXT of 4Ø columns by 24 lines MODE#5 : TEXT : for GRAPHICS of 32Ø x 24Ø pixels MODE#5 : HGR :

MODE# (cont'd)

6 Mixed Graphics and Text

for TEXT of 40 columns by 48 lines

MODE#6 : TEXT :

for GRAPHICS of 320 x 480 pixels

MODE#6 : HGR :

7 Mixed Graphics and Text

for TEXT of 80 columns by 24 lines

MODE#7 : TEXT :

for GRAPHICS of 640 x 240 pixels

MODE#7 : HGR :

8 Mixed Graphics and Text

for TEXT of 80 columns by 48 lines

MODE#8 : TEXT :

for GRAPHICS of 640 x 480 pixels

MODE#8 : HGR :

9 INTERNAL USE ONLY

1xx Mixed Graphics and Text

the graphics are as chosen on the preselected page with xx lines of text on the preselected MIXED page
where xx = the number of lines of text in the range of from 0 to the maximum number of lines on the MIXED page.

Example

```
MODE# 100
sets 0 lines of text, all graphics
MODE# 106
```

sets 6 lines of text, rest of screen = graphics

Notes — See GR, HGR, HGR2, and TEXT.

Error Messages

Caveat

NEW

Syntax	NEW
Description	This command clears UNIBASIC program memory and resets UNIBASIC so that it can accept a "new" program.
Parameters	None.
Notes	
Error Messages	
Examples	
Caveat	

NEXT

Syntax	NEXT [realvar]
Description	This command is used with the FOR command to control looping. The NEXT command terminates, or marks the logical end point, in a program, of a loop.
Notes	See the FOR command.
Error Messages	
Examples	
Caveat	

NORMAL

Syntax	NORMAL
Description	This command sets the video output to the mode of white letters on a black background, which is the "normal" video mode.
Parameters	None.
Notes	NORMAL may be used in either the immediate or the deferred execution mode.
Error Messages	
Examples	
Caveat	

NOTRACE

Syntax NOTRACE

Description This command turns off the TRACE debugging option.

Parameters None.

Notes
- See the TRACE command.
- NOTRACE may be used in either the immediate or the deferred execution mode.

Error Messages

Examples

Caveat

ON ... GOTO
ON ... GOSUB

Syntax

ON arithexpr GOTO linenumber [{,linenumber}]
ON arithexpr GOSUB linenumber [{,linenumber}]

Descriptions

ON...GOTO branches to the line number whose position in the list corresponds to arithexpr.

ON...GOSUB operates in a similar fashion, but as a subroutine call rather than a branch.

If arithexpr is equal to Ø or greater than the number of listed linenumbers, then execution proceeds to the next statement.

Parameters

Arithexpr must be in the range of from Ø to 255.

Notes

See the GOTO and GOSUB commands.

Error Messages

If arithexpr is less than Ø or greater than 255, then the system will display
?ILLEGAL QUANTITY ERROR

Examples

```
1ØØ INPUT "TYPE A NUMBER (1, 2, OR 3) >";NUM
11Ø ON NUM GOTO 15Ø, 2ØØ, 25Ø
12Ø PRINT "SORRY, "NUM" IS NOT A VALID CHOICE."
13Ø GOTO 1ØØ
15Ø PRINT "NUM IS 1": GOTO 1ØØ
2ØØ PRINT "NUM IS 2": GOTO 1ØØ
25Ø PRINT "NUM IS 3": GOTO 1ØØ
RUN
TYPE A NUMBER (1, 2, OR 3) >5
SORRY, 5 IS NOT A VALID CHOICE.
TYPE A NUMBER (1, 2, OR 3) >3
NUM IS 3
TYPE A NUMBER (1, 2, OR 3) >5ØØ
?ILLEGAL QUANTITY ERROR
```

Caveat

ONERR GOTO

Syntax

ONERR GOTO linenumber

Description

This command causes an unconditional branch to the program line indicated by linenumber, when an error occurs in the program. This command must be placed in the sequence of execution ahead of the statement that an error is expected to occur in.

Error messages are suppressed. Execution of the program is NOT halted. Error codes are available to indicate the type of error that occurred. The error codes are:

Ø NEXT WITHOUT FOR	9Ø UNDEFINED STATEMENT
5 END OF DEVICE	1Ø7 BAD SUBSCRIPT
6 FILE NOT FOUND	12Ø REDIMENSIONED ARRAY
8 INPUT/OUTPUT ERROR	133 DIVISION BY ZERO
9 DISK FULL	163 TYPE MISMATCH
16 SYNTAX	176 STRING TOO LONG
22 RETURN WITHOUT GOSUB	191 FORMULA TOO COMPLEX
42 OUT OF DATA	224 UNDEFINED FUNCTION
53 ILLEGAL QUANTITY	254 BAD RESPONSE TO INPUT STATEMENT
69 OVERFLOW	
77 OUT OF MEMORY	

Parameters

Linenumber must be the number of a statement in the program.

Notes

The ONERR GOTO statement must be executed before the occurance of an error to avoid program interruption.

See the RESUME command

Error Messages

?NEXT WITHOUT FOR ERROR
?RETURN WITHOUT GOSUB ERROR
For a return to a NEXT or RETURN after error handling.

Examples

Caveat

Care must be taken when handling errors that occur within FOR...NEXT loops or between GOSUB and RETURN. The pointers and RETURN stacks are disturbed after errors if ONERR...GOTO is in effect. The error handling routine must return to the FOR or GOSUB statement, NOT the NEXT or RETURN statement.

PAGE#

Syntax

PAGE#1
PAGE#2

Description

This command selects a high resolution page for plotting.

Parameters

None.

Notes

- Permits drawing in the background while the previous drawing is displayed.

- Does not change the page that is displayed on the screen.

Error Messages

Examples

Caveat

PEEK

Syntax	PEEK (address)
Description	This command is to "read" a location in memory.
Parameters	Address is the arithmetic expression whose value is the location to be PEEKed.
Notes	A list of PEEKs and POKEs that are implemented in UNIBASIC is included in the appendices.
Error Messages	
Examples	
Caveat	

PLOT

Syntax

PLOT x, y

Description

In low resolution graphics mode, this command places a dot on the screen at screen location (x, y).

Parameters

X and y must be arithmetic expressions.

X must be in the range of Ø to 79.

Y must be in the range of Ø to 47.

Notes

The origin (Ø,Ø) is the upper left corner of the screen.

The most recently executed COLOR statement determines the color of the dot.

PLOT has no visible effect when used in HGR2 mode. This is true even if GR precedes PLOT, because the screen is not "looking at" the low resolution graphics page (page one) of memory.

See the GR and the TEXT commands.

Error Messages

?ILLEGAL QUANTITY ERROR
If the arithmetic expression for X is not in the range of Ø to 79 or if the arithmetic expression for Y is not in the range of Ø to 47.

Examples

PLOT Ø,Ø

places a dot in the upper left corner of the screen.

Caveat

Attempting to PLOT to a TEXT window results in a character being placed where the dot would have appeared. (A character occupies the space of 2 low resolution graphics characters stacked vertically.)

POKE

Syntax	POKE address, arithexpr
Description	This command places the eight bit value of arithexpr in the location address.
Parameters	Address is a 32 bit real number. Arithexpr is an arithmetic expression whose value is in the range of Ø to 255.
Notes	
Error Messages	
Examples	
Caveat	

POP

Syntax	POP
Description	This command has the effect of a RETURN without the GOSUB. The next RETURN encountered will branch to the statement after the second most recently executed GOSUB.
Parameters	None.
Notes	This command is called "POP" because it POPs 1 address off of the "stack" of RETURN addresses. See the GOSUB and the RETURN commands.
Error Messages	?RETURN WITHOUT GOSUB ERROR If POP is executed before a GOSUB has been encountered.
Examples	
Caveat	

POS

Syntax — POS (expression)

Description — This command returns the cursor's current horizontal position on the screen, counting from the text window's leftmost position (which is Ø).

Parameters — Expression is a dummy parameter which is used to separate the parenthesis. It must be a legal number, a legal string, a legal literal, or a legal variable. If expression is not a legal variable name, then it must be enclosed in quotation marks (unless it is a number).

Notes

- Positions are numbered from the left, beginning with Ø (for TAB and HTAB, the positions are numbered from the left, beginning with 1).
- POS may be used in either the immediate or the deferred execution mode.

Error Messages

Examples

Caveat

PR#

Syntax PR# arithexpr

Description This command transfers output to the slot whose value is arithexpr. This command specifies which peripheral will receive output from subsequent PRINT statements.

Parameters Arithexpr is an arithmetic expression whose value is in the range of Ø to 1.

Notes PR# Ø indicates subsequent output will be to the display, not slot Ø. Slot Ø is not addressable from UNIBASIC for use with a peripheral device.

PR# 1 directs output to the parallel printer port, on the Dimension 68ØØØ system, as well as the display.

See the IN# command.

Error Messages ?ILLEGAL QUANTITY ERROR
If arithexpr is less than Ø or greater than 1.

Examples

Caveat

PRINT

Syntax

```
PRINT [{expr}[{,|; [{expr}]}]] [,|;]
PRINT {;}
PRINT {,}
```

Description

This command, when executed without options, causes a line feed and a carriage return to be executed on the screen or designated output device.

When this command is executed with options, the values of the list of the specified expressions are printed. If neither a comma nor a semi-colon ends the list, then a line feed and a carriage return are executed following the last item printed. If an item on the list is followed by a comma, then the first character of the next item to be printed will appear in the first position of the next available tab field.

The first tab field comprises the leftmost 16 printing positions in the text window (HTAB positions 1 through 16). The second tab field occupies the next 16 positions (17 through 32), and it is available for tab field printing only if nothing is printed in position 16. The third tab field consists of the next 16 positions (33 through 48), and is available only if nothing is printed in position 32. The remaining tab positions follow the same rules.

Parameters

If an item on the list is followed by a semi-colon, then the next item is concatenated, it is printed directly afterward with no intervening spaces or blanks.

Items listed without intervening commas or semi-colons are concatenated if the items can be parsed without syntax problems.

If a period cannot be treated as a decimal point, then it is assumed to be the number zero.

PRINT followed by a list of semi-colons does nothing more than PRINT alone. Print followed by a list of commas spaces across 1 tab field per comma, up to a limit of 239 characters per instruction.

Notes

The question mark (?) may be used as an abbreviation for the PRINT command, it LISTs as PRINT.

PRINT may be used in immediate or deferred execution mode.

PRINT (cont'd)

Error
Messages

Examples

```
A=1 : B=2 : C=3 : C(4)=5 : C5=7

PRINT 1/3(2*4)51, : PRINT 1(A)2(B)3C(4)C5
.333333333851                 1122357

PRINT 3.4.5.6. , : PRINT A."B."C.4
3.4.5.6Ø                      1ØB.3.4
```

Caveat

QUIT

Syntax	QUIT
Description	This command returns control to CP/M-68K.
Parameters	None.
Notes	
Error Messages	
Examples	
Caveat	

READ

Syntax

READ var [{var}]

Description

This command transfers values from a DATA list to the variable(s) specified in this READ command. When the first READ statement is executed in a program, then its first variable takes on the value of the first element in the DATA list (the DATA list consists of all the elements from all the DATA statements in the stored program).

The second variable (if there is one) takes on the value of the second element in the DATA list, and so on.

When the READ statement finishes execution, it leaves a data list pointer after the last element of data used. The next READ statement executed (if any) begins using the data list from the position of the pointer.

Either RUN or RESTORE resets the pointer to the first element in the DATA list.

Parameters

Successive variables get successively typed values.

String variables and arithmetic variables may be mixed in the same READ statement, but the DATA list must contain values of the appropriate types.

Notes

Extra data left unread is alright.

In immediate mode, you can only READ elements from DATA statements which exist as lines in a currently stored program. The elements of DATA in a stored program can be READ even if the stored program has not been RUN. Executing a program in the immediate mode does not set the data list pointer to the first element in the DATA list.

Error Messages

?OUT OF DATA ERROR IN linenumber
Attempting to READ more data than the DATA list contains. Linenumber is the line number of the READ statement which asked for the additional DATA.

?OUT OF DATA ERROR
If no DATA statement has been stored while executing READ in the immediate mode.

Examples

Caveat

REM

Syntax

REM [character(s)]

Parameters

All characters, including statement separators, may be included. Their usual meanings are ignored.

Notes

- A REM is terminated only by a <CR> (carraige return).

- When REMs are LISTed, UNIBASIC inserts one space, or blank, after REM, no matter how many spaces were typed after REM by the user.

- REM may be used in either the immediate of the deferred execution mode.

Error Messages

Examples

```
10 REM THIS IS A REMARK
```

creates a comment or remark.
The comment is "THIS IS A REMARK".

Caveat

Reset Button

Syntax — None, just push the button.

Description — This action immediately stops all UNIBASIC program processing. The program that was executing is destroyed.

Parameters — None.

Notes — - Pressing the reset button puts the Dimension 68000 system under the control of the monitor program. The monitor program prompt character (:) is displayed.

Error Messages

Examples

Caveat

RESTORE

Syntax	RESTORE
Description	This command resets the DATA list pointer back to the beginning of the DATA list.
Parameters	None.
Notes	RESTORE may be used in the immediate or the deferred execution mode.
Error Messages	
Examples	
Caveat	

RESUME

Syntax RESUME

Description Used at the end of the error handling routines. RESUME causes the program to resume execution at the beginning of the statement in which the error occured.

Parameters None.

Notes If an error occurs in an error handling routine, the use of RESUME will place the program in an infinite loop.

See the ONERR GOTO command.

Error Messages

Examples

Caveat

RETURN

Syntax	RETURN
Description	This command causes the program to branch to the statement following the most recently executed GOSUB. The address of the statement branched to is on top of the RETURN "stack".
Parameters	None.
Notes	See the GOSUB and the POP commands.
Error Messages	?RETURN WITHOUT GOSUB ERROR If a program encounters RETURN (and/or POP) statements once more than it has encountered GOSUB statements.
Examples	
Caveat	

ROT

Syntax — ROT = arithexpr

Description — This command causes the shape plotted on the high resolution display to be rotated angularly on the display screen by arithexpr amount.

Parameters — Arithexpr is the amount of angular rotation which is in the range of from Ø to 255.

Notes — See the DRAW, the XDRAW, and the SCALE commands.

ROT can be used in the immediate and the deferred execution modes.

Error Messages

Examples

Caveat

RUN

Syntax

RUN [filename]
RUN [linenumber]

Description

This command clears all variables and starts execution of the program currently in memory. If this command is given with a filename, then the system attempt to load the specified file as a UNIBASIC program and then execute the loaded program. If this command is given with a linenumber, then the system attempts to execute the program in memory at the linenumber specified.

Parameters

Filename is the name of a CP/M-68K file that contains a UNIBASIC program. Filename is in the file naming convention given in the UNIBASIC USER'S GUIDE.

Linenumber is the program line number.

Notes

- When no program is currently in memory, RUN returns control to the user.

- RUN may be used in either the immediate or the deferred execution mode.

Error Messages

?UNDEF'D STATEMENT ERROR
When a line number is specified, but it does not exist

Examples

RUN 4Ø
starts executing the current program at line number 4Ø

Caveat

SAVE

Syntax

SAVE [filename]

Description

This command saves a program that is residing in memory onto the disk as a program type file.

Parameters

Filename is a name for the disk file in the format described in the UNIBASIC USER'S GUIDE under file naming conventions.

Notes

- SAVE will not display a prompt.
- After the SAVE command has executed, the current program continues executing.
- SAVE may be interrupted by pressing the reset button only.
- SAVE may be used in either the immediate or the deferred execution mode.

Error Messages

?SYNTAX ERROR
When the first 4 characters of a variable name are "SAVE", an actual SAVE command must be entered or the error message is displayed.

Examples

SAVE NEWFILE.BAS,D3
saves the program as NEWFILE.BAS on disk drive 3.

Caveat

SCALE

Syntax — SCALE = arithexpr

Description — This command sets the high resolution scale size for a shape to be drawn.

Parameters — Arithexpr is the size factor and is in the range of 1 to 255. A value of 1 is for a point for point reproduction of the shape table definition. A value of 255 extends each vector by 255.

Notes — SCALE=Ø is MAXIMUM size and not a single point.

Error Messages

Examples

Caveat

SHLOAD

Syntax

SHLOAD filename

Description

This command loads a shape table into memory from the disk file specified.

Parameters

Filename is the name of the file on the disk that contains the shape table desired. The filename follows the FILE NAMING CONVENTIONS given earlier in this manual.

Notes

- The shape table starts at location 4000 decimal in UNIBASIC.

Error Messages

Examples

Caveat

SHSAVE

Syntax — SHSAVE filename

Description — This command saves the shape table from memory to the disk file specified.

Parameters — Filename is the name of the file where the shape table is to be saved. The filename follows the FILE NAMING CONVENTIONS given earlier in this manual.

Notes — - The shape table starts at location 4000 decimal in UNIBASIC. Data must be POKEd into the shape table.

Error Messages

Examples

Caveat

SHSIZE

Syntax — SHSIZE (size)

Description — This command is used to set the size of the shape table to the value desired.

Parameters — Size is an arithmetic expression that is equal to the desired size of the shape table.

Notes

- The default size of the shape table is 500 bytes.
- Data is inserted into the shape table by using the POKE command. The shape table starts at the location 4000 decimal.

Error Messages

Examples

Caveat

SPC

Syntax — SPC (arithexpr)

Description — This command prints spaces, or blanks, from the current cursor position.

Parameters — Arithexpr is an arithmetic expression that specifies the number of spaces, or blanks, to be printed. It must be enclosed in parenthesis and be in the range of from Ø to 255.

Notes

- SPC may only be used in a print statement.
- SPC (Ø) will not print a space.
- A large number of spaces may be printed by repeating the SPC command, for example, PRINT SPC(5Ø)SPC(255).
- The difference between HTAB and SPC is that SPC print a specified number of spaces, while HTAB moves the cursor to a specified position.
- When spacing goes beyond the rightmost edge of the text window, it continues at the left edge of the next line.
- When printing in tab fields, spacing may be within a tab field or across into another tab field, or it may occupy a tab field of its own.
- If arithexpr is a real number, it will be converted to an integer.
- SPC will be interpreted as a reserved word if the next non-blank character is a left parenthesis.

Error Message — ?ILLEGAL QUANTITY ERROR
If arithexpr is out of range

Examples

```
N=1Ø : PRINT 1SPC(N)2
1          2
```

the above printed the digit 1 followed by 1Ø spaces and then the digit 2.

Caveat

STEP

Syntax

FOR realavar = aexpr1 TO aexpr2 STEP aexpr3

Description

This command is used with the FOR...NEXT command as a modifier to specify the looping increment amount.

Parameters

Aexpr3 set the looping increment amount, and it must be in the range of -32767 to 32767. Aexpr3 is added to realvar at each loop iterration.

Notes

See the FOR...NEXT command.

Error Messages

Examples

```
100 FOR IJ = 1 TO 100 STEP 5
.
.
.
200 NEXT IJ
```

Caveat

STOP

Syntax

STOP

Description

This command causes the program execution to halt and returns control to the user.

Parameters

None.

Notes

- STOP displays the message
  BREAK IN linenumber
  where linenumber is line number of the statement that contains the STOP statement.

- STOP may be used in either the immediate or the deferred execution mode.

Error Message

Examples

Caveat

TAB

Syntax — TAB (arithmetic-expression)

Description — This command positions the cursor to the position specified by arithmetic-expression.

Parameters — Arithmetic-expression must be enclosed in parenthesis.

Notes

- TAB may only be used in a PRINT statement.
- TAB will not move the cursor to the left (use HTAB). If the value of arithmetic expression is less than the value of the current cursor position, the cursor is not moved.
- When TAB causes the cursor to move beyond the current text window line, the cursor goes to the beginning of the next line and continues spacing.
- TAB will be interpreted as a reserved word only if the next non-blank character is a left parenthesis.
- TAB may be used in either the immediate or the deferred execution mode.

Error Messages — ?ILLEGAL QUANTITY ERROR
If arithmetic-expression is out of range.

Examples

Caveat

TEXT

Syntax — TEXT

Description — This command sets the screen to non-graphics text mode which is 8Ø characters per line and 24 lines on the screen.

Parameters — None.

Notes — The prompt and the cursor are moved to the last line of the screen, which is equivalent to a VTAB 24 in the TEXT mode.

TEXT always resets to a full screen.

TEXT does not clear the screen. Low resolution graphics will be distorted.

TEXT should be executed before switching from HGR2 to GR.

See the MODE# command.

Error Messages

Examples

Caveat

TRACE

Syntax

TRACE

Description

This command activates the debugging feature of UNIBASIC which causes each line number in the program to be displayed as it is executed.

Parameters

None.

Notes

- When a program is displaying output, TRACE output may be changed or destroyed.
- RUN, CLEAR, NEW, and DEL will NOT turn off TRACE.
- TRACE may be used in either the immediate or the deferred execution mode.

Error Messages

Examples

Caveat

VLIN

Syntax — VLIN Y1, Y2 AT X

Description — This command draws, in low resolution GRaphics mode, a vertical line from the co-ordinates (X,Y1) to (X,Y2).

Parameters — Y1 and Y2 are arithmetic expressions in the range of from Ø to 47.

X is an arithmetic expression in the range of from Ø to 79.

Notes — The most recently executed COLOR statement determines the color of the line.

VLIN has no visible effect when used in the high resolution graphics mode.

See the GR, the HLIN, and the MODE# commands.

Error Messages — ?ILLEGAL QUANTITY ERROR
If Y1, Y2 or X is out of range.

Examples

Caveat — If VLIN is used on a TEXT window, a line of characters is placed where the line of graphic dots would have been plotted. (A character occupies the space of 2 low resolution dots stacked vertically.)

VTAB

Syntax — VTAB arithmetic-expression

Description — This command moves the cursor vertically on the screen. The top line is line number 1 and the bottom line is line number 24.

Parameters — Arithmetic-expression indicates the screen line number that the cursor is moved to and must be in the range of from 1 to 24.

Notes

- VTAB move the cursor only vertically (up or down) and will not move it horizontally (right or left).
- VTAB makes absolute moves, relative only to the top or bottom of the screen. the text window is ignored.
- In the graphics mode, VTAB will move the cursor into the graphics area of the screen.
- VTAB may be used in either the immediate or the deferred execution mode.

Error Messages — ?ILLEGAL QUANTITY ERROR
If the arithmetic-expression is out of the range of from 1 to 24.

Examples

Caveat

WAIT

Syntax

WAIT address,and-mask[,xor-mask]

Description

This command causes the program that is executing to conditionally pause.

Parameters

Address is an arithmetic expression (in the range of -65535 to +65535) that gives the location in memory of the word that is to be tested to determine when to end the program pause.

And-mask is an arithmetic expression that is equivalent to an 8-bit mask. The mask is ANDed with the contents of <address>. For each bit, this ANDing gives a Ø unless both of the corresponding bits are high (1). If the results of this process are eight zeros, then the test is repeated. Only when the result is non-zero (which means at least one high (1) bit is and-mask was matched by a corresponding high (1) bit at location <address>), is the WAIT completed and the program resumes execution at the next instruction.

Xor-mask is an arithmetic expression that is Equivalent to an 8-bit mask. The mask is XORed with the contents of location <address> first, and then the result of this XORing is compared against the <and-mask> as described above. The XOR process specifies that high (1) bit in the <xor-mask> gives a result that is the REVERSE of the corresponding bit at location <address> (a 1 becomes a Ø; a Ø becomes a 1). A low (Ø) bit in <xor-mask> give a result that is the same as the corresponding bit in location <address>. If <xor-mask> is all zeros, the XOR portion does nothing.

Notes

- Only pressing the Reset Button can interrupt a WAIT.

Error Messages

Examples

Caveat

XDRAW

Syntax — XDRAW shapenumber [AT X, Y]

Description — This command is used to draw the shape specified, from the shape table, by shapenumber. If the optional X and Y co-ordinates are NOT specified, then the shape is drawn at the point that the last point on the screen was plotted. If the X and Y co-ordinates are specified, then the shape is drawn at those co-ordinates. The color that the shape is plotted in is the complement of the COLOR existing at each point being drawn over.

Parameters — Shapenumber is the number of the shape definition in the shape definition table that was previously loaded using the SHLOAD command. Shape number is an arithmetic expression.

X and Y are arithmetic expressions. X must be in the range of from Ø to 278. Y must be in the range of from Ø to 191.

Notes

Error Messages — ?ILLEGAL QUANTITY ERROR
If any of the parameters are out of range.

Examples

Caveat

C H A P T E R 3

U N I B A S I C F U N C T I O N S

ABS

Syntax	ABS (arithexpr)
Description	Returns the absolute value of arithexpr. Returns arithexpr if arithexpr is greater than or equal to Ø. Returns -(arithexpr) if arithexpr is less than Ø.
Parameters	Arithexpr may be any arithmetic expression.
Notes	- This function may be used wherever an expression the same type may be used.
Error Messages	
Examples	
Caveat	

ASC

Syntax — ASC (string-expression)

Description — The ASC function returns an ASCII code for the first character of string-expression.

Parameters — String-expression is the string examined. If it is a literal string, it must be enclosed in quotation marks, and quotation marks must not be used within the string.

Notes — - ASC may be used in either the immediate or deferred execution mode.

Error Messages — If string-expression is a null string, the ?ILLEGAL QUANTITY ERROR message is displayed.

Examples — If the string variable STRING$ has the value "ALL", then PRINT ASC(STRING$) will print the ASCII code 65.

100 CODE=ASC("C") will assign the ASCII code 67 to the variable named CODE.

Caveat

ATN

Syntax	ATN (arithexpr)
Description	Returns the arctangent in radians, of arithexpr. The angle returned is in the range -pi/2 through +pi/2.
Parameters	Arithexpr may be any arithmetic expression.
Notes	- This function may be used wherever an expression of the same type may be used.
Error Messages	
Examples	
Caveat	

CALL

Syntax CALL address[(arg1,,,arg14)

Description See the description of the CALL command in a previous chapter.

COS

Syntax	COS (arithexpr)
Description	Returns the cosine of arithexpr radians.
Parameters	Arithexpr may be any arithmetic expression.
Notes	- This function may be used wherever an expression of the same type may be used.
Error Messages	
Examples	To derive the function secant: SEC (X) = 1/COS(x)
Caveat	

CHR$

Syntax	CHR$ (arithmetic-expression)
Description	CHR$ is a function which evaluates arithmetic-expression and returns the ASCII character which corresponds to it.
Parameters	arithmetic-expression must be in the range Ø through 255.
Notes	- Real numbers are converted to integers. - CHR$ may be used in either the immediate or deferred execution mode.
Error Messages	If arithmetic-expression is out of range, the ?ILLEGAL QUANTITY ERROR is displayed.
Examples	1ØØ A$=CHR$(N) will return the ASCII code for the contents of the variable N.

Caveat

DEF FN

Syntax

DEF FN name (dummyvariable) = arithexpr1
FN name (arithexpr2)

Description

Defines functions in a program. Functions may be used wherever arithmetic expressions may be used. After the execution of a program line containing DEF, the DEFined function may be used in the form

FN name (argument)

where the argument may be any arithmetic expression.

Parameters

The rules for using arithmetic variables apply to function names (the first 8 characters must be unique).

Arithexpr1 may be only 1 program line in length.

Dummyvariable must be a real number arithmetic variable.

FN substitutes the argument for dummyvariable wherever dummyvariable appears in the DEFinition. Arithexpr1 may contain any number of variables. At most 1 of those variables corresponds to dummyvariable.

Notes

- The DEFinition's dummyvariable need not appear in arithexpr1. In that case, when the function is used, the function's argument is ignored in evaluating arithexpr1. The function's argument must always be legal.

- Functions may be redefined during the course of a program.

- When a new function is defined by a DEF statement, 6 bytes in memory are used to store the pointer to the definition.

- DEF may be used in deferred execution mode only. FN may be used in deferred or immediate execution mode.

DEF FN (cont'd)

Error Messages

?UNDEFN'D FUNCTION ERROR
If a deferred execution DEF FN name statement is not executed prior to using FN name.

Examples

```
100 DEF FN A(W) = 2 * W + W
110 PRINT FN A(23)
120 DEF FN B(X = 4 + 3
130 G = FN B(23)
140 PRINT G
150 DEF FN A(Y) = FN B(Z) + Y
160 PRINT FN A(G)

RUN
69                      [ FN A(23) = 2 * 23 + 23 ]
7                       [ FN B(anything) = 7     ]
14                      [ new FN A(7) = 7 + 7    ]

10 DEF FN ABC(I) = COS(I)
20 DEF FN ABC(I) = TAN(I)
```

The function AB is defined in line 10 and then redefined in line 20.

Caveat

User-defined string functions are not allowed.

Functions defined using an integer name (name%) for the function name or for dummyvariable are not allowed.

If CLEAR, NEW, DEL, or RUN destroys or skips a DEFined function in memory, the function may not be defined.

EXP

Syntax EXP (arithmetic-expression)

Description This function returns the value of e (the natural logrithm base - e=2.718289 to 6 places) raised to the power which is the value of arithmetic-expression.

Parameters Arithmetic-expression mat be any valid arithmetic expression.

Notes

Error Messages

Caveat

FN

Syntax	FN name (arithexpr2)
Description	See DEF FN name (arithexpr1)
Parameters	Arithexpr2 may be any valid arithmetic expression.
Notes	- FN may be used in immediate or deferred execution mode.
Error Messages	?UNDEF'D FUNCTION ERROR If the function has not been DEFined yet.
Examples	
Caveat	

INT

Syntax	INT (arithmetic-expression)
Description	This function returns the integer value that is less than or equal to the value of arithmetic-expression.
Parameters	Arithmetic-expression may be any valid arithmetic expression.
Notes	This function may be used wherever an expression of the same type may be used.
Error Messages	
Examples	
Caveat	

LEFT$

Syntax — LEFT$ (string-expression, arithmetic-expression)

Description — This function returns the first arithmetic-expression characters out of string-expression.

Parameters — String-expression is the string examined.

Arithmetic-expression is converted to an integer and its value, after conversion to integer, must be greater than or equal to 1 and it must be less than or equal to 255.

Notes

- If arithmetic-expression is greater than the length of string expression, only the characters in the string-expression are returned.

- LEFT$ may be used in either the immediate or the deferred execution mode.

Error — If the arithmetic-expression is out of range, the "?ILLEGAL QUANTITY ERROR" message is displayed.

Examples — PRINT LEFT$("MICRO CRAFT",5) selects the first 5 characters of the string "MICRO CRAFT" and returns the string "MICRO".

Caveat

LEN

Syntax	LEN (string-expression)
Description	This function returns the number of characters in string-expression.
Parameters	String-expression is the string examined.
Notes	- LEN may be used in either immediate or deferred execution mode.
Error Messages	
Examples	LEN("ASTRING") returns a value for the string length of 7.
Caveat	

LOG

Syntax	LOG (arithmetic-expression)
Description	This function returns the natural logrithm value of the arithmetic-expression.
Parameters	Arithmetic-expression is any valid arithmetic expression.
Notes	
Error Messages	
Examples	
Caveat	

MID$

Syntax

MID$ (string-expression, arithmetic-expression-1, [arithmetic-expression-2])

Description

This function returns a sub-string, or portion of a string.

Parameters

String-expression is the string examined.

Arithmetic-expression-1 is the first position within the string from which characters are extracted. It must be in the range of from 1 to 255.

Arithmetic-expression-2 is the number of characters to be extracted from the string. It must be in the range of from 1 to 255.

Notes

- If arithmetic-expression-2 is not specified, the entire string is returned, beginning with the position specified by arithmetic-expression-1.
- If arithmetic-expression-1 is greater than the length of string-expression, then a null string is returned.
- If the sum of arithmetic-expression-1 and arithmetic-expression-2 is greater than 255 or the length of string-expression, only the sub-string is returned.
- MID$(STR$,255,255) will return 1 character if the length of STR$ is equal to 255, otherwise a null string is returned.
- MID$ may be used in either the immediate or deferred execution mode.

Error Messages

If arithmetic-expression-1 or arithmetic-expression-2 are out of range, the "?ILLEGAL QUANTITY ERROR" message is displayed.

The "$" must not be omitted from MID$ or UNIBASIC will interpret it as an arithmetic variable and the "?TYPE MISMATCH ERROR" message will be displayed.

Examples

MID$("TESTSTRING",3,4) extracts 4 characters from the string "TESTSTRING" begining in position 3 and returns the string "STST".

MID$("TESTSTRING",5) extracts 6 characters from the string "TESTSTRING" begining in position 5 and returns the string "STRING".

MID$ (cont'd)

Caveat

NF#

Syntax

NF#

Description

This function, NF# (No File) returns information on the most recently opened file, as follows:

NF# = Ø the file already existed.
NF# = 1 the file did not exist and was created.

Parameters

Notes

See the commands for DISK OPERATIONS in a previous section.

Error Messages

Examples

```
1Ø CREATED = NF#
2Ø IF CREATED THEN GOTO 4Ø
or the equivalent,
1Ø IF NF# THEN GOTO 4Ø
```

If the file was created, then the program branches to line 4Ø.

Caveat

PDL

Syntax	PDL (paddlenumber)
Description	This function returns the current value of the game control (paddle) specified by paddlenumber.
Parameters	The arithmetic-expression paddlenumber must be in the range of Ø to 3.
Notes	See the appendices for information about other game switches.
Error Messages	If paddlenumber is less than Ø or greater than 3, then a "?ILLEGAL QUANTITY ERROR" message is displayed.
Examples	1Ø PPOS = PDL(1) Sets the variable PPOS to the value of game control number one.
Caveat	

RIGHT$

Syntax

RIGHT$ (string-expressiom, arithmetic-expression)

Description

This function returns the last arithmetic-expression characters of string-expression.

Parameters

String-expression is the string examined.

Arithmetic-expression is any valid arithmetic expression. It must be in the range of from 1 to 255.

Notes

- If arithmetic-expression is greater than or equal to the length of the string, then the entire string is returned.

- RIGHT$ may be used in either the immediate or the deferred execution mode.

Error Messages

If arithmetic-expression is out of range, the "?ILLEGAL QUANTITY ERROR" message is displayed.

The "$" must not be omitted from RIGHT$ or UNIBASIC will interpret it as an arithmetic variable, and the "?TYPE MISMATCH ERROR" will be displayed.

Examples

PRINT RIGHT$("UNIBASIC",5) selects the last 5 characters of the string "UNIBASIC" and returns the string "BASIC".

Caveat

RND

Syntax — RND (arithmetic-expression)

Description — This function returns the a random real number that is greater than or equal to Ø and is less than or equal to 1.

Parameters — If arithmetic-expression is greater than Ø, then RND returns a new random number each time it is used.

If arithmetic-expression is less than Ø, then RND generates the same random number each time it is used with the same arithmetic-expression as if from a permanent random number generator table.

If a particular negative argument is used to generate a random number, then subsequent random numbers generated with positive arguments will follow the same sequence each time. A different random sequence is initialized by each different negative argument.

The reason for using a negative argument for RND is to initialize a repeatable sequence of random numbers.

If arithmetic-expression is zero, RND returns the most recent previous random number generated.

Notes

Error Messages

Examples

Caveat

SCRN

Syntax SCRN (X, Y)

Description This function only operates in the low-resolution GRaphics mode. This function returns the color code of the dot whose co-ordinate is (X, Y).

Parameters

Notes

Error Messages

Examples

Caveat

SGN

Syntax — SGN (arithmetic-expression)

Description — This function returns the following values:

Returns -1 If arithmetic-expression < Ø
Returns Ø If arithmetic-expression = Ø
Returns +1 If arithmetic-expression > Ø

Parameters — Arithmetic-expression may be any arithmetic expression.

Notes — This function may be used wherever an expression of the same type may be used.

Error Messages

Examples

Caveat

SIN

Syntax	SIN (arithmetic-expression)
Description	This function returns the sine of arithmetic-expression in radians.
Parameters	Arithmetic-expression may be any arithmetic expression.
Notes	This function may be used wherever an expression of the same type may be used.
Error Messages	
Examples	
Caveat	

SQR

Syntax — SQR (arithmetic-expression)

Description — This function returns the positive square root of the arithmetic-expression.

Parameters — Arithmetic-expression may be any arithmetic expression.

Notes

Error Messages

Examples

Caveat

STR$

Syntax	STR$ (arithmetic-expression)
Description	This function converts the value of arithmetic-expression into a string.
Parameters	Arithmetic-expression is evaluated before conversion.
Notes	- STR$ may be used in either the immediate or the deferred execution mode.
Error Messages	If arithmetic-expression is outside the limits for real numbers, then the "?OVERFLOW ERROR" message is displayed.
Examples	40 N = -3.5 50 NN$ = STR$(N) NN$ = "-3.5"
Caveat	

TAN

Syntax	TAN (arithmetic-expression)
Description	This function returns the tangent of the arithmetic expression.
Parameters	Arithmetic-expression may be any arithmetic expression.
Notes	This function may be used wherever an expression of the same type may be used.
Error Messages	
Examples	
Caveat	

USR

Syntax USR (arith-expr)

Description This function is NOT implemented in UNIBASIC. See the description of the CALL command in a previous chapter.

Parameters

Notes

Error Messages

Examples

Caveat

VAL

Syntax

VAL (string-expression)

Description

This function attempts to convert a string into an integer or a real number, and then return the resulting number.

Parameters

The first character of the string-expression must be a legal numeric character or a space, otherwise a Ø will be returned.

Notes

- VAL looks at each character of string-expression. If a non-numeric character is found, the search ends and VAL interprets the string up to that point as an integer or real number.

- Legal numeric characters are the digits Ø through 9, spaces, the letter E, a decimal point, and the + and - signs.

Error Messages

If the absolute value of the number returned is greater than 1E38, less than -1E38, or the number contains more than 38 digits, including zeros, then the "?OVERFLOW ERROR" message is displayed.

Examples

```
4Ø A$ = "-1.58E-1Ø.3"
5Ø AA = VAL(A$)

AA = -1.58E-1Ø
```

Caveat

VARPTR

Syntax

VARPTR (name)

Description

This function returns an integer whose value is the location, in memory, of the variable whose name is given as the argument (name). This function always returns the address, in memory, of the data for the argument (name).

Notes

- A value must be assigned to the varaible given as the argument (name) for this function prior to execution of VARPTR. Otherwise an ILLEGAL FUNCTION CALL ERROR message results.

- A function call of the form VARPTR(A(Ø)) is usually specified when passing an array, so that the lowest-addressed element of the array is returned.

- The address returned will be a signed integer in the range of from -32767 to +32768. If a negative address is returned, add it to 65536 to obtain the actual address.

Error Messages

?ILLEGAL FUNCTION CALL
A value was not assigned to (name) prior to execution of VARPTR function.

Examples

1ØØ CALL (VARPTR(YY))

Caveat

The UNIBASIC interpreter may assign more than one string variable (name) to the same string data in memory. The byte that immediately precedes the string data contains an integer value that is the number of string variables that are "linked" or are using that data string in memory. All strings terminate with a zero or "null" byte. Because of the way that multiple string variables can share the same string data, the user is strongly discouraged from writing string data to memory using the VARPTR function.

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 80 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 0-127 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 1024 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.

A P P E N D I X C

A S C I I C O D E S

American Standard Code for Information Interchange

A S C I I C H A R A C T E R S E T

(7 - B I T C O D E)

MSD / LSD	Ø ØØØ	1 ØØ1	2 Ø1Ø	3 Ø11	4 1ØØ	5 1Ø1	6 11Ø	7 111
Ø ØØØØ	NUL	DLE	SP	Ø	@	P	`	p
1 ØØØ1	SOH	DC1	!	1	A	Q	a	q
2 ØØ1Ø	STX	DC2	"	2	B	R	b	r
3 ØØ11	ETX	DC3	#	3	C	S	c	s
4 Ø1ØØ	EOT	DC4	$	4	D	T	d	t
5 Ø1Ø1	ENQ	NAK	%	5	E	U	e	u
6 Ø11Ø	ACK	SYN	&	6	F	V	f	v
7 Ø111	BEL	ETB	'	7	G	W	g	w
8 1ØØØ	BS	CAN	(	8	H	X	h	x
9 1ØØ1	HT	EM	)	9	I	Y	i	y
A 1Ø1Ø	LF	SUB	*	:	J	Z	j	z
B 1Ø11	VT	ESC	+	;	K	[	k	{
C 11ØØ	FF	FS	,	<	L	\	l	|
D 11Ø1	CR	GS	-	=	M	]	m	}
E 111Ø	SO	RS	.	>	N	^	n	~
F 1111	SI	US	/	?	O	_	o	DEL

A P P E N D I X D

M A T H M A T I C A L F U N C T I O N S

D E R I V E D F U N C T I O N S

M A T H E M A T I C A L F U N C T I O N S

D E R I V E D F U N C T I O N S

```
SEC(X)      = 1/COS(X)
CSC(X)      = 1/SIN(X)
COT(X)      = 1/TAN(X)
ARCSIN(X)   = ATN(X/SQR(-X*X+1))
ARCCOS(X)   = -ATN(X/SQR(-X*X+1))+1.5708
ARCSEC(X)   = ATN(X/SQR(X*X-1))+SGN(SGN(X)-1)*1.5708
ARCCSC(X)   = ATN(X/SQR(X*X-1))+(SGN(X)-1)*1.5708
ARCCOT(X)   = ATN(X)+1.5708
SINH(X)     = (EXP(X)-EXP(-X))/2
COSH(X)     = (EXP(X)+EXP(-X))/2
TANH(X)     = (EXP(-X)/EXP(X)+EXP(-X))*2+1
SECH(X)     = 2/(EXP(X)+EXP(-X))
CSCH(X)     = 2/(EXP(X)-EXP(-X))
COTH(X)     = EXP(-X)/(EXP(X)-EXP(-X))*2+1
ARCSINH(X)  = LOG(X+SQR(X*X+1))
ARCCOSH(X)  = LOG(X+SQR(X*X-1))
ARCTANH(X)  = LOG((1+X)/(1-X))/2
ARCSECH(X)  = LOG((SQR(-X*X+1)+1)/X)
ARCCSCH(X)  = LOG((SGN(X)*SQR(X*X+1)+1)/X)
ARCCOTH(X)  = LOG((X+1)/(X-1))/2
```

A P P E N D I X E

R E S E R V E D W O R D S

RESERVED WORDS

RESERVED WORDS

ABS
ASC
ATN
CALL
CHR$
CLEAR
COLOR
CONT
DATA
DEF FN
DEL
DIM
DRAW
END
EXP
FOR
FLASH
GET
GOSUB
GOTO
GR
HCOLOR
HGR
HGR2
HLIN
HOME
HPLOT
HTAB
IF
INPUT
INT
INVERSE
IN#
LEFT$
LEN
LET
LIST
LOAD
LOG
MID$
NEW
NEXT
NORMAL
NOTRACE
ON
ONERR
PDL
PEEK
PLOT
POKE
POP
POS
PRINT
PR#
READ
RECALL
REM
RESTORE
RESUME
RETURN
RIGHT$
ROT
RND
RUN
SAVE
SCALE
SCRN
SGN
SHLOAD
SIN
SPC
SQR
STEP
STOP
STORE
STR$
TAB
TAN
TEXT
TRACE
VAL
VARPTR
VLIN
VTAB
XDRAW

A P P E N D I X F

E R R O R M E S S A G E S

ERROR MESSGAES

CAN'T CONTINUE
ILLEGAL DIRECT

Ø	NEXT WITHOUT FOR
5	END OF DEVICE
6	FILE NOT FOUND
8	INPUT/OUTPUT ERROR
9	DISK FULL
16	SYNTAX ERROR
22	RETURN WITHOUT GOSUB
42	OUT OF DATA
53	ILLEGAL QUANTITY
69	OVERFLOW
77	OUT OF MEMORY
9Ø	UNDEFINED STATEMENT
1Ø7	BAD SUBSCRIPT
12Ø	REDIMENSIONED ARRAY
133	DIVISION BY ZERO
163	TYPE MISMATCH
176	STRING TOO LONG
191	FORMULA TOO COMPLEX
224	UNDEFINED FUNCTION
254	BAD RESPONSE TO INPUT STATEMENT

A P P E N D I X G

P E E K s a n d P O K E s

PEEKs and POKEs

P O K E S

POKE 216 - Clears ERROR FLAG

POKE -16368,Ø - Clear Keyboard Ready

POKE -16304,Ø - GRaphics ON

POKE -16303,Ø - TEXT ON

POKE -16302,Ø - No Mixed TEXT and GRaphics

POKE -16301,Ø - Mixed TEXT and GRaphics

POKE -16300,Ø - Ignored but accepted

POKE -16299,Ø - Ignored but accepted

POKE -16298,Ø - Sets LO RES

POKE -16297,Ø - Sets HI RES

P E E K S

PEEK(216)	Error Flag
PEEK(218)	Reads LSB of Error Line Number
PEEK(219)	Reads MSB of Error Line Number
PEEK(222)	Error Code
PEEK(-16384)	KEYBOARD
PEEK(-16336)	TOGGLES SPEAKER ONCE
PEEK(-16287)	READS GAME CONTROL #Ø PUSHBUTTON
PEEK(-16286)	READS GAME CONTROL #1 PUSHBUTTON
PEEK(-16285)	READS GAME CONTROL #2 PUSHBUTTON

I N D E X

S Y S T E M R E F E R E N C E M A N U A L

INDEX

INDEX

dimension

MICRO CRAFT CORPORATION

January 5, 1984

The Unibasic Reference Manual (part number 680-0200-200) 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:

Page to be Replaced	New Page
7,8	7,8
9,10	9,10
13,14	13,14
21,22	21,22
25,26	25,26
33,34	33,34
57,58	57,58
61,62	61,62
73,74	73,74
75,76	75,76
85,86	85,86
135,136	135,136

4747 Irving Blvd., Suite 241, Dallas, Texas 75247, (214) 630-2562

U N I B A S I C

R E F E R E N C E M A N U A L

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 - 2 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.

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

GETTING STARTED

The Dimension 68000 system is shipped with a "SYSTEM 1" diskette and a "SYSTEM 2" diskette. Before doing ANYTHING else, make a copy of the "system diskettes that were shipped with your DImension 68000 system. A step by step procedure for making these copies, or "BACKING-UP" these diskettes is included in the "BACKING-UP" APPENDIX.

Always make a back-up of any diskettes received from Micro Craft, Inc.

If you should damage the "SYSTEM" diskette or the "LANGUAGES UTILITIES" diskette, additional diskettes may be purchased from Micro Craft, Inc., for $350.00 plus shipping and handling fees.

RUNNING UNIBASIC

To use the Micro Craft, Inc., UNIBASIC interpreter on the Dimension 68000 system, insert the "SYSTEM 1" diskette into the "A" diskette drive. Then, either type

BASIC

or

BASIC filename

where filename = the name of the file that contains the basic program to be run.

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 up to three characters. The characters may be either alphabetic or numeric. CP/M normally reserves certain FILETYPES (such as .BAS for BASIC programs or .$$$ for temporary files). UNIBASIC does not require that it's programs have a specific FILETYPE.

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 disk drives A:, B:, C:, AND D:. If no DRIVE SPECIFICATION is included, the system will use the system default disk drive.

MODES OF OPERATION

When UNIBASIC is initialized it displays the prompt character ":". The prompt character indicates that UNIBASIC is at the command level; that is, it is ready to accept commands. At this point, UNIBASIC may be used in either of two modes: direct mode or indirect mode.

In direct mode, UNIBASIC statements and commands are not preceded by line numbers. They are executed as they are entered. Results of arithmetic and logical operations may be displayed immediately and stored for later use but the instructions themselves are lost after execution. Direct mode is useful for debugging and for using UNIBASIC as a "calculator" for quick computations that do not require a complete program.

Indirect mode is used for entering programs. Program lines are preceded by line numbers and are stored in memory. The program stored in memory is executed by entering the RUN command.

LINE FORMAT

UNIBASIC program lines have the following format (square brackets indicate optional input):

```
nnnnn UNIBASIC-STATEMENT[:UNIBASIC-STATEMENT...] <CR>
```

More than one UNIBASIC statement may be placed on a line, but each must be separated from the last by a colon.

A UNIBASIC program line always begins with a line number and ends with a carriage return. A line may contain a maximum of 255 characters.

A line may contain up to 256 characters. When the line displayed requires more characters than a physical CRT line contains, UNIBASIC automatically continues displaying the line on the next physical line of the CRT.

LINE NUMBERS

Every UNIBASIC program line begins with a line number. Line numbers indicate the order in which the program lines are stored in memory. Line numbers are also used as references in branching and editing. line numbers must be in the range of Ø to 63999.

A period (.) may be used in the LIST, and the DELETE commands to refer to the current line.

CHARACTER SET

The UNIBASIC character set is comprised of the alphabetic characters, numeric characters, and special characters.

The alphabetic characters in UNIBASIC are the upper-case letters of the alphabet.

The UNIBASIC numeric characters include the digits Ø through 9.

In addition, the following special characters and terminal keys are recognized by UNIBASIC:

CHARACTER	ACTION
	Blank or Space
=	Equals sign or assignment symbol
+	Plus sign
-	Minus sign
*	Asterisk or multiplication symbol
/	Slash or division symbol
^	Up arrow or exponentiation symbol
(	Left or open parenthesis
)	Right or close parenthesis
%	Percent
#	Number or pound sign
$	Dollar sign
!	Exclamation point or "bang"
[	Left or open bracket
]	Right or close bracket
,	Comma
.	Period
;	Semicolon
:	Colon
&	Ampersand or and sign
'	Single quotation mark (apostrophe)
?	Question mark
<	Less than
>	Greater than
@	At sign
"	Quotation mark
<BS>	Back Space key - deletes the last character typed
<ESC>	Escape key
<BREAK>	Break key
<CR>	Carriage Return keys (marked "Retrn" and "Enter")
<LINE_FEED>	Line feed key (Ctrl-L)

CONTROL CHARACTERS

UNIBASIC supports the following control characters:

CONTROL CHARACTER	ACTION
<break>	Interrupts program execution and returns to UNIBASIC command level when at an INPUT statement.
CTRL-C	Interrupts program execution and returns to CP/M command levelwhen at an INPUT statement.
CTRL-G	Rings the bell at the terminal.
CTRL-H	Backspaces. Deletes the last character typed.
CTRL-I	Tabs to the next tab stop. Tab stops are set every eight columns.
CTRL-L	Line Feed. Moves the cursor down one line.
CTRL-M	Carriage Return. Moves the cursor to the left side of the screen
CTRL-S	Suspends output to the CRT.
CTRL-Q	Resumes output to the CRT after a CTRL-S.

CONSTANTS

Constants are the values UNIBASIC uses during execution. There are two types of constants: string and numeric.

A string constant is a sequence of up to 255 alphanumeric characters enclosed in quotation marks(").

Examples

"HELLO"
"$25,000.00"
"Number of Employees"

ARRAY VARIABLES

An array is a group or table of values referenced by the same variable name. Each element in an array is referenced by an array variable that is subscripted with an integer or an integer expression. An array variable name has many subscripts as there are dimensions in the array. For example, V(10) would reference a value in a one-dimension array, T(1,4) would reference a value in a two-dimension array, and so on. The maximum number of elements per dimension is 32,767. The maximum number of dimensions is 88.

SPACE REQUIREMENTS

All UNIBASIC variables and arrays have a data header. The data headers are located in the UNIBASIC's data area. The data area is located between the location of the interpreter in memory and the location of the interpreter's stack in memory. The interpreter's stack is located just below the CP/M kernal in memory. The CP/M kernal is located in the top approximately 8100 (hex) of RAM. The spaces that are occupied by the interpreter, by the data area, and by the interpreter's stack are allocated dynamically. The data headers are shown below for each data type:

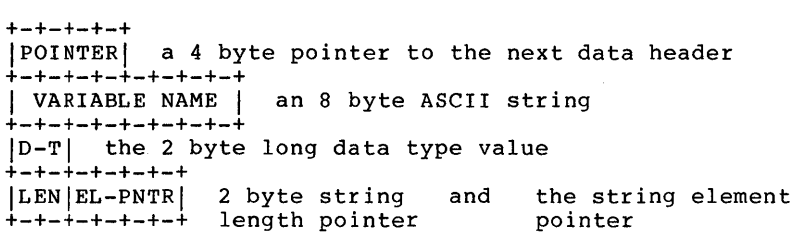
INTEGER

```
+-+-+-+-+
|POINTER|  a 4 byte pointer to the next data header
+-+-+-+-+-+-+-+-+
| VARIABLE NAME |  an 8 byte ASCII string
+-+-+-+-+-+-+-+-+
|D-T|  the 2 byte long data type value
+-+-+-+-+-+-+
|VAL|UNUSED |  the 2 byte integer value and 4 unused bytes
+-+-+-+-+-+-+
```

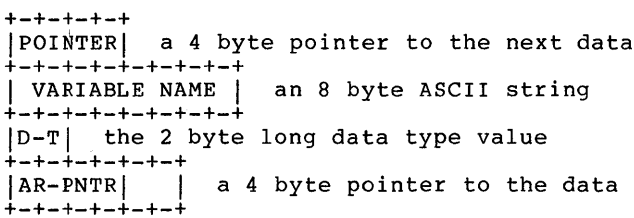
REAL

```
+-+-+-+-+
|POINTER|  a 4 byte pointer to the next data header
+-+-+-+-+-+-+-+-+
| VARIABLE NAME |  an 8 byte ASCII string
+-+-+-+-+-+-+-+-+
|D-T|  the 2 byte long data type value
+-+-+-+-+-+-+
| VALUE |   |  the 4 byte real value
+-+-+-+-+-+-+
```

STRING

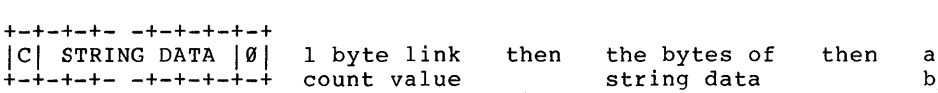


ARRAY



The string element has the following layout in memory:

STRING ELEMENT



FUNCTIONAL OPERATORS

A function is used in an expression to call a predetermined operation that is to be preformed on an operand. UNIBASIC has "intrinsic" functions that reside in the system, such as SQR (square root) or SIN (sine). All UNIBASIC intrinsic functions are described in Chapter 3.

UNIBASIC also allows "user-defined" functions that are written by the programmer. See "DEF FN" in a later section.

STRING OPERATORS

Strings may be concatenated by using +.

Example

```
10 A$ = "FILE": B$ = "NAME"
20 PRINT A$+B$
30 PRINT "NEW"+A$+B$
RUN
FILENAME
NEWFILENAME
```

Strings may be compared using the same relational operators that are used with numbers:

= <> < > <= >=

String comparisons are made by taking one character at a time from each string and comparing the ASCII codes. If all the ASCII codes are the same, the strings are equal. If the ASCII codes differ, the lower code number precedes the higher. If during string comparison, the end of one string is reached, the shorter string is said to be smaller. Leading and trailing blanks ARE significant.

Examples

```
"AA"<"AB"
"FILENAME"="FILENAME"
"X&">"X#"
"CL">"CL"
"kg">"KG"
"SMYTH"<"SMYTHE"
B$<"9/12/78" where
B$="8/12/78"
```

Thus, string comparisons can be used to test string values or to alphabetize strings. All string constants used in comparison expressions must be enclosed in quotation (") marks.

HIGH RESOLUTION GRAPHICS

There are two pages of high resolution graphics. The user selects the page desired by issuing either a PAGE#1 command or a PAGE#2 command.

SHAPE TABLE

The shape table begins at address 4000 decimal. The shape table has a default size of 500 bytes. The shape table size can be changed by using the SHSIZE command. The shape table is loaded either from a disk file by using the SHLOAD command or by POKEing the values in starting at address 4000 decimal. The shape table can be saved into a disk file by using the SHSAVE command.

INPUT EDITING

If an incorrect character is entered as a line is being typed, it can be deleted with the <Back Space> (<BS>) key or with CONTROL-H. Both the <BS> key and CONTROL-H have the effect of backspacing over a character and erasing it. Once a character(s) has been deleted, simply continue typing the line as desired.

To delete a line that is in the process of being typed, type CONTROL-U. A carriage return is executed automatically after the line is deleted.

To correct program lines for a program that is currently in memory, simply retype the line using the same line number. UNIBASIC will automatically replace the old line with the new line.

More sophisticated editing capabilities are provided. See the alternate arrow commands in a later section.

To delete the entire program currently residing in memory, enter the NEW command. (See the "NEW" command in a later section.) NEW is usually used to clear memory prior to entering a new program.

ERROR MESSAGES

If an error causes program execution to terminate, an error message is printed. For a complete list of UNIBASIC error codes and error messages, see the APPENDICES.

EDITING - alt arrow

Syntax

alt →
alt ←
alt ↓
alt ↑

Description

These commands do not affect characters moved over by the cursor: the characters remain both on the CRT screen and in memory. By themselves, these commands do not affect the program line being typed.

alt →	moves the cursor one space to the right
alt ←	moves the cursor one space to the left
alt ↓	moves the cursor one space down
alt ↑	moves the cursor one space up

Parameters

None.

Notes

- To change a program line: LIST the line on the CRT screen and use the alt. arrow commands to place the cursor over the first character of the line. Use the right-arrow key to move across the line, stopping at characters you wish to change and entering the desired character. When you are finished changing the line, press RETURN to store or execute the corrected line. If you did not use LIST to display the line, do not copy the prompt character (:).

- The alt arrow commands may be used in the immediate execution mode only.

Error Messages

Examples

Caveat

<BREAK> key

Syntax — break

Description — break interrupts the current process immediately after the statement that is currently being executed.

Parameters — None.

Notes

- break may be entered to interrupt an INPUT or GET but must be the first character entered. The interruption occurs when return is pressed for INPUT and immediately for GET.
- BREAK IN line-number is displayed a program is executing.
- break may be used in the deferred execution mode only.

Error Messages

Examples

Caveat

CATALOG

Syntax

CATALOG[,Dn]

Description

This command causes a list of the contents of the directory of the disk drive specified to be displayed on the screen.

Parameters

n is the number of the disk drive that the directory is to be displayed for. The following is a correlation of disk drive numbers and CP/M-68K drive specifiers:

DRIVE NUMBER	CP/M-68K DRIVE
1	A:
2	B:
3	C:
4	D:

If no disk drive is specified, then the disk drive that was most recently accessed will be used.

Notes

Error Messages

Examples

Caveat

COLOR

Syntax

COLOR = arithexpr

Description

Sets the color for plotting in low resolution graphics mode.

Parameters

The range of values for arithexpr is from Ø through 255.

Color numbers and their associated names are:

Ø black	4 dark green	8 dark aqua	12 green
1 dark blue	5 grey	9 bright blue	13 aqua
2 red	6 orange	1Ø grey	14 yellow
3 magenta	7 pink	11 light blue	15 white

COLOR evaluates arithexpr modulo 16 to return a value in the range of Ø to 15.

Notes

- In high-resolution graphics mode COLOR has no meaning.

- See SCRN and PLOT for more information.

Error Messages

Examples

Caveat

When used in TEXT mode with PLOT, COLOR will affect which character the PLOT instruction places in the text window.

HCOLOR

Syntax

HCOLOR = arithmetic-expression

Description

This command sets the high-resolution GRaphics color to that specified by the value of arithmetic-expression.

Parameters

The range of values for arithmetic-expression is from Ø through 15 if COLOR is ON. And, a range of Ø to 1 if COLOR is OFF.

The colors are as follows:

COLOR = ON	COLOR = OFF
Ø = Black	Ø = Black
1 = Dark Blue	1 = White
2 = Red	
3 = Magenta	
4 = Dark Green	
5 = Grey	
6 = Orange	
7 = Pink	
8 = Dark Aqua	
9 = Bright Blue	
1Ø = Grey	
11 = Light Blue	
12 = Green	
13 = Aqua	
14 = Yellow	
15 = White	

Notes

- In the low-resolution graphics mode, HCOLOR has no meaning.

Error Messages

Examples

Caveat

HGR

Syntax

HGR

Description

Sets the screen for High-resolution GRaphics mode. The resolution depends on the MODE# command.

Displays the bottom N lines of the text window below the graphics.

Clears the screen to black and displays page 1 of memory.

Parameters

None.

Notes

- HCOLOR is not changed.

- Text screen memory is not affected.

- Leaves the text "window" at full screen, but only the bottom N text lines are visible below the graphics. The cursor will still be in the text "window", but may not be visible unless moved to one of the bottom N lines.

- See MODE#, PAGE#, GR, HGR2, TEXT, COLOR, and HCOLOR for more information.

Error Messages

?SYNTAX ERROR
If the reserved word HGR appears as the first three characters of a variable name.

Examples

Caveat

If the reserved word HGR is used as the first characters of a variable name, the HGR may be executed before the

?SYNTAX ERROR

appears. Executing the statement

HGRIP=4

sets the high-resolution graphics mode.

HOME

Syntax	HOME
Description	HOME moves the cursor to the upper left screen position within the scrolling window and clears all text within the window.
Parameters	None.
Notes	- HOME may be used in either the immediate or deferred execution mode.
Error Messages	
Examples	
Caveat	

HPLOT

Syntax

```
HPLOT X1, Y1
HPLOT TO X2, Y2
HPLOT X1, Y1 TO X2, Y2 [TO Xm, Ym ... TO Xn, Yn]
```

Description

This command will draw a high-resolution dot or line. If only X1 and Y1 are specified, then a dot will be drawn. If only X2 and Y2 are specified, then a line will be drawn from the last point plotted to the co-ordinates specified. If both the X1, Y1 and X2, Y2 co-ordinates are specified, then a line will be plotted from the X1, Y1 co-ordinates to the X2, Y2 co-ordinates.

Notes

Error Messages

Examples

Caveat

LOAD

Syntax	LOAD filename
Description	This command causes UNIBASIC to attempt to "load" into memory the filename specified from disk as a UNIBASIC program.
Parameters	Filename is the name of a disk file in the form specified in the UNIBASIC USER'S GUIDE.
Notes	
Error Messages	
Examples	
Caveat	

MODE#

Syntax MODE# modenumber

Description This command selects various graphics and text screen options, based on the value of modenumber. This statement is executed instead of PEEKing and POKEing. This statement must be immediately followed by either a TEXT command, an HGR command, or by an HGR2 command.

Parameters Modenumber has the following options and values:

Mode#	Option
Ø	Initializes video to 8Ø columns by 24 lines MODE#Ø
1	Reset the 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 the ERROR FLAG to ON MODE#2
3	Set COLOR to OFF MODE#3
4	Set COLOR to ON MODE#4
5	Mixed Graphics and Text for TEXT of 4Ø columns by 24 lines or for GRAPHICS of 32Ø x 24Ø pixels MODE#5

MODE# (cont'd)

6 Mixed Graphics and Text
for TEXT of 40 columns by 48 lines
or
for GRAPHICS of 320 x 480 pixels

MODE#6

7 Mixed Graphics and Text
for TEXT of 80 columns by 24 lines
or
for GRAPHICS of 640 x 240 pixels

MODE#7

8 Mixed Graphics and Text
for TEXT of 80 columns by 48 lines
or
for GRAPHICS of 640 x 480 pixels

MODE#8

9 INTERNAL USE ONLY

1xx Mixed Graphics and Text

the graphics are as chosen on the preselected page with xx lines of text on the preselected MIXED page
where xx = the number of lines of text in the range of from 0 to the maximum number of lines on the MIXED page.

Example

MODE#100
sets 0 lines of text, all graphics

MODE# 106
sets 6 lines of text, rest of screen = graphics

MODE# (cont'd)

Notes See GR, HGR, HGR2, and TEXT.

Error
Messages

Caveat

PLOT

Syntax — PLOT x, y

Description — In low resolution graphics mode, this command places a dot on the screen at screen location (x, y).

Parameters — X and y must be arithmetic expressions.

X must be in the range of Ø to 79.

Y must be in the range of Ø to 47.

Notes — The origin (Ø,Ø) is the upper left corner of the screen.

The most recently executed COLOR statement determines the color of the dot.

PLOT has no visible effect when used in HGR2 mode. This is true even if GR precedes PLOT, because the screen is not "looking at" the low resolution graphics page (page one) of memory.

See the GR and the TEXT commands.

Error Messages — ?ILLEGAL QUANTITY ERROR
If the arithmetic expression for X is not in the range of Ø to 79, or if the arithmetic expression for Y is not in the range of Ø to 47.

Examples — PLOT Ø,Ø

places a dot in the upper left corner of the screen.

Caveat — Attempting to PLOT to a TEXT window results in a character being placed where the dot would have appeared. (A character occupies the space of 2 low resolution graphics characters stacked vertically.)

POKE

Syntax — POKE address, arithexpr

Description — This command places the eight bit value of arithexpr in the location address.

Parameters — Address is a 32 bit real number.

Arithexpr is an arithmetic expression whose value is in the range of Ø to 255.

Notes

Error Messages

Examples

Caveat

NF#

Syntax NF#

Description This function, NF# (No File) returns information on the most recently opened file, as follows:

NF# = Ø the file already existed.
NF# = 1 the file did not exist and was created.

Parameters

Notes See the commands for DISK OPERATIONS in a previous section.

Error Messages

Example 1Ø IF NF# = 1 GOTO 4Ø

or the equivalent

1Ø IF NF# GOTO 4Ø

If the file was created, then the program branches to line 4Ø.

Caveat

PDL

Syntax	PDL (paddlenumber)
Description	This function returns the current value of the game control (paddle) specified by paddlenumber.
Parameters	The arithmetic-expression paddlenumber must be in the range of Ø to 3.
Notes	See the appendices for information about other game switches.
Error Messages	?ILLEGAL QUANTITY ERROR If paddlenumber is less than Ø or greater than 3.
Examples	1Ø PPOS = PDL(1) Sets the variable PPOS to the value of game control number one.
Caveat	

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

4747 Irving Blvd., Suite 241, Dallas, Texas 75247, (214) 630-2562

UNIBASIC
USER'S GUIDE
Micro Craft Corporation

AS VERSION
680-0200-100A

PRELIMINARY

01/05/84 REVISION

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

I N D E X

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

INDEX

INDEX
UNIBASIC USER'S GUIDE

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 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:

- 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 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 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.

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# 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:

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

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 = 40 columns by 24 lines
GRAPHICS = 320 x 240 pixels
MODE#5

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

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


8    = Mixed Graphics and Text
         TEXT     = 80 columns by 48 lines
         GRAPHICS = 640 x 480 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 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.

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.

dimension

MICRO CRAFT CORPORATION

January 5, 1984

The Dimension 68000 User's Guide, (part number 680-0001-100) 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:

Page to be Replaced	New Page
9,10	9,10
27,28	27,28
31,32	31,32
35,36	35,36

4747 Irving Blvd., Suite 241, Dallas, Texas 75247, (214) 630-2562

D I M E N S I O N 6 8 0 0 0

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

Micro Craft Corporation

6 8 0 - 0 0 0 1 - 1 0 0 A

PRELIMINARY

0 1 / 0 5 / 8 4 R E V I S I O N

N O T I C E

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

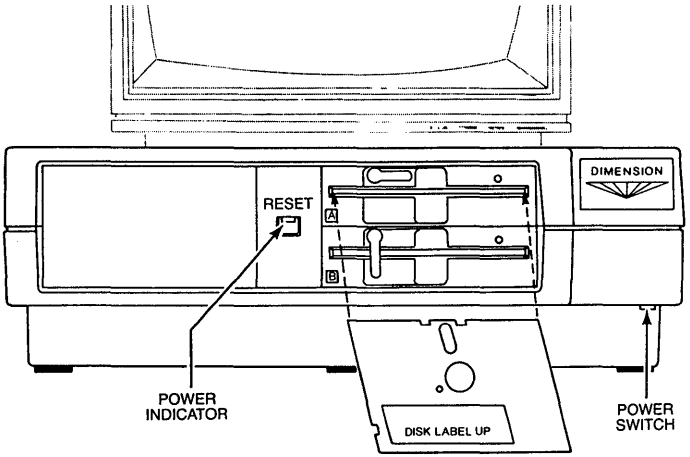
DISCLAIMER OF ALL WARANTIES AND LIABILITY

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, MERCHANTIBILITY, 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 EXCLUSION MAY NOT APPLY TO YOU.

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

NOTE: The lever on the diskette drives can NOT be moved unless either a protector card or a diskette is inserted into the drive.

Forcing the lever will DAMAGE the drive!



Inserting A Diskette Into The DIMENSION 68000

- OPTIONALLY, if there is a printer, it should be connected to the connector on the rear of the System Unit that is labeled "PARALLEL CENTRONICS PRINTER".
- OPTIONALLY, if there is a modem, it should be connected to the connector on the rear of the System Unit that is labeled "RS232C". This connection may be used for any device that uses the EIA (Electronics Industry Association) RS-232C interface.

*** CAUTION ***

BE SURE THAT EACH CONNECTION HAS BEEN FIRMLY SEATED PRIOR TO "POWERING UP" THE SYSTEM.

To "POWER UP" the system, use the following steps:

1 - TURN ON the CRT. Allow time for the CRT to warm up.

2 - TURN ON the DIMENSION 68000. The power switch is under the right front edge of the System Unit. When the power is turned on, the LED in the RESET Switch will come ON. For the first few seconds, the computer will perform a self test routine. After the self test routine, the computer will display the message below, then the LED on Disk Drive A will light up. The lighting of the LED on Disk Drive A indicates that the system is ready for a "bootable" diskette to be inserted into the disk drive. The system should display, on the CRT, the following:

Welcome to the Realm
of
Dimension Computing
by
Micro Craft Corporation

There will be a character sized block, displayed in reversed video (light in color, instead of dark) displayed on the left side of the screen under the above message. This block is called the cursor. Since the display on the screen is white letters on a dark screen, the cursor will be a block of light.

3 - INSERT a "SYSTEM 1" diskette into Disk Drive A. The "SYSTEM 1" diskette that you use should be a COPY of the "SYSTEM 1" diskette that was shipped with the DIMENSION 68000 System, NOT THE ORIGINAL. If you have not made copies of the "SYSTEM" diskettes, STOP! You need to make copies NOW! The section on BACKING UP later in this chapter tells how to make copies of your "SYSTEM" diskettes. So does APPENDIX B which is titled "THE BACK UP PROCEDURE."

KEYBOARD

The keyboard for the DIMENSION 68000 is a microprocessor controlled, 30 character per second (300 BPS), ASCII coded, TTL output level device. The keyboard has 10 function keys and a combination numeric pad/cursor control pad .

CRT INTERFACE

The CRT interface is an EIA RS-170 compatible interface. The voltage output is adjusted to be 1 volt pk-pk nominal. The interface supplies composite sync. The interface can supply either an interlace or a non-interlace output signal. The DIMENSION 68000 sets the mode to interlace or to non-interlace as follows:

NON-INTERLACE	INTERLACE
20x20 TEXT	80x50 TEXT
40x24 TEXT	100x50 TEXT
80x24 TEXT	HI-RES GRAPHICS
50x25 TEXT	
LO-RES and MEDIUM RES GRAPHICS	

Interlace mode has two times the resolution as non-interlace mode. This is because interlace mode has 525 horizontal lines on the screen, while non-interlace mode has 262 1/2 horizontal lines.

REAL TIME CLOCK

There is an internal, interrupt driven, Real-Time Clock and event timer that has programmable interval rates between 10 microseconds and 250 milliseconds.

PROCESSOR

The microprocessor used in the DIMENSION 68000 is an 8 MHz 68000 type microprocessor. The 68000 microprocessor has 16 bit wide external data paths. The internal architecture of the 68000 microprocessor is 32 bits wide. The 68000 has the following registers:

- 8 DATA REGISTERS that are 32 BITS wide
- 7 ADDRESS REGISTERS that are 32 BITS wide
- 2 STACK POINTER REGISTERS that are 32 BITS wide
  - 1 for the USER
  - 1 for the SUPERVISOR
- 1 PROGRAM COUNTER REGISTER that is 32 BITS wide
- 1 STATUS REGISTER that is 16 BITS wide

Because of the internal architecture, the 68000 is properly described as a 32 bit micro-processor.

Some of the features of the 68000 microprocessor are:

- 5 DATA TYPES
  - Bit
  - BCD Digits (4 bits)
  - Bytes (8 bits)
  - Words (16 bits)
  - Long Words (32 bits)
- 16M byte direct addressing range
- 14 addressing modes on 61 basic instructions for over 1000 total instruction types

DISK DRIVES

The standard disk drives used for diskettes in the DIMENSION 68000 system are half height, 5 1/4 inch, double sided, double density, half stepable, 40 track units. They are capable of storing up to 400K bytes. Optionally, the DIMENSION 68000 system can be supplied with the following types of drives:

- 80 track, 817K byte, 5 1/4 inch diskette drives
- 8 inch diskette drives
- 3 1/2 inch diskette drives
- 3 1/4 inch diskette drives
- Winchester-type Hard disk drives

Space is provided on the rear panel of the system unit for a 34 pin connector and for a 50 pin connector. These connectors can be used for connections to any externally mounted disk drives.

Micro Craft Corporation manufactures a disk drive expansion unit that can contain two 8 inch diskette drives, or a mix of 3 1/2 inch diskette drives, 3 1/4 inch diskette drives, and 8 inch diskette drives. These diskette drives are packaged in an expansion chassis. The expansion chassis also includes a power supply to supply the necessary voltages and currents to operate the drives.

EXPANSION SLOTS

The six expansion slots in the DIMENSION 68000 system may be used for additional memory, co-processors, additional input or output (I/O) ports, etc. as desired by the user. A description of the pinouts used by the expansion slots is available in the DIMENSION 68000 System Reference Manual.

ADDITIONAL MEMORY

The DIMENSION 68000 can support up to 16M bytes of memory. If the user desires to expand the RAM on the DIMENSION beyond 512K bytes, the additional memory may be added by installing cards that contain the extra memory.

CO-PROCESSORS

The DIMENSION 68000 can support other microprocessors co-resident with the 68000 type processor that resides in the system. The DIMENSION, by using co-processors, can emulate other personal computers. By using the 6512 processor as a co-processor, the DIMENSION is able to emulate the APPLE II. By using an 8086 processor, the DIMENSION is able to emulate the IBM-PC and the IBM-PC look-alikes. And, by using a Z-80, the DIMENSION can emulate most of the CP/M-80 machines and the TRS-80 units.

Micro Craft can provide the user with an APPLE II emulator card, an IBM emulator card, and a Z-80 emulator card. The emulator cards from Micro Craft do not have to be plugged into any particular slot. They are slot independent.

HARD DISK

The DIMENSION 68000 Winchester-type hard disk controller will plug into any expansion slot. It is a ST506 type interface capable of handling 2 Winchester-type disks with a total of 300M bytes of storage.

PROTOTYPING KIT

Micro Craft Corporation has a prototyping kit which contains a prototyping board and all of the necessary documentation for anyone to be able to build a card that will properly plug in to and properly operate in the expansion slots of the DIMENSION 68000 system.

CHAPTER 3

ARCHITECTURE

The DIMENSION 68000 is designed to have the hardware functions that are inside the system to be software configurable. This is accomplished by means of software controlled hardware latches that are placed between the system bus and the various device controllers. The system bus carries the address signals, the data signals, and the control signals for the whole DIMENSION 68000 system. The device controllers handle the following devices for the system:

- memory
- video display
- speaker
- disk drives
- keyboard
- RS-232 interface
- parallel interface
- game controller interface
- real time clock

MEMORY USAGE

The organization of the DIMENSION 68000 memory is detailed in the DIMENSION 68000 SYSTEM REFERENCE MANUAL. The overall memory usage is as follows:

ADDRESSES	USAGE
000000 - 0000FF	INTERRUPT VECTORS
000100 - 0001FF	SYSTEM RAM AREA
000200 - 0011FF	VIDEO SCREEN TEXT AREA
001200 - 0012FF	SYSTEM FUNCTIONS
001300 - FEFFFF	CP/M TRANSIENT AREA (Depending on Memory Size)
010000 - 01FFFF	CO-PROCESSOR AREA DURING EMULATION (MIN.)
020000 - 07FFFF	CO-PROCESSOR EXPANSION AREA
080000 - FEFFFF	RAM EXPANSION AREA
FF0000 - FF1FFF	ROMBIOS
FF2000 - FF7FFF	RESERVED GRAPHICS RAM
FF8000 - FFFFFF	PERIPHERAL CONTROL AREA

The DIMENSION 68000 has 8K of Read Only Memory (ROM) which is located in memory between FF0000 and FF1FFF. This ROM is known as the ROMBIOS. ROMBIOS stands for Read Only Memory Built-in Input / Output System. The overall ROMBIOS usage is as a group of 68000 machine language routines that handle the I/O requirements of the DIMENSION 68000.

I/O

The I/O requirements of the DIMENSION 68000 are handled by the machine language routines in the ROMBIOS. The ROMBIOS functions are used to handle the following I/O device requirements:

- the CRT Controller
- the Keyboard
- the Disk Drives
- the RS-232 Interface
- the Parallel Printer Interface
- the Real Time Clock

A detailed description of the ROMBIOS functions is contained in the DIMENSION 68000 SYSTEM REFERENCE MANUAL.

The DIMENSION 68000

The DIMENSION 68000 system is designed to allow software configuration of the hardware controllers that are connected to the system bus. This feature and the memory utilization design lend themselves to the easy implementation of co-processor emulation of other microprocessor systems. Even the memory controllers are software configurable.

CO-PROCESSORS

The DIMENSION 68000 system was designed to allow co-processor emulation of other microprocessor systems. The Micro Craft Corporation can supply for the DIMENSION 68000 system, as options, three co-processor boards. These co-processor boards plug into the expansion slots on the main board inside the DIMENSION 68000 system unit. The co-processor boards do not have to be plugged into any particular expansion slot. They are slot independent. The three co-processor boards are the 6512 board, the 8086 board, and the Z-80 board. The system can have one board, two boards, or all three boards installed at the same time. However, only one co-processor can be in operation at a time.

6512

The Micro Craft 6512 co-processor board is supplied with the necessary software to allow the DIMENSION 68000 to be able to emulate the Apple II (TM), the APPLE II+ (TM), and the Apple IIe (TM) personal computers.