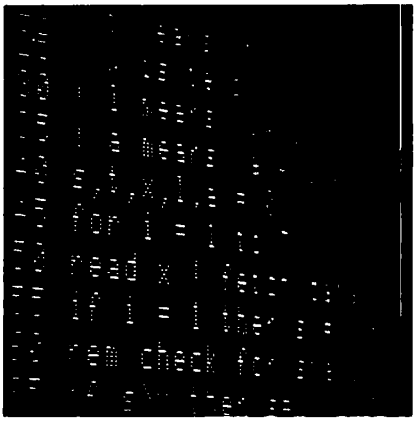
Prime Computer, Inc.

BASIC/VM
Revision 18



FDR3341

TABLE OF CONTENTS

The Programmer's Companion is a new series of pocket size quick-reference guides to Prime software products.

Published by Prime Computer, Inc.
Technical Publications, 500 Old Connecticut Path, Framingham, MA 01701

First Printing: February 1979
Second Printing, Revision: March 1980
Third Printing, Revision: April 1981

Credits

Research and copy
Laura Douros
Deborah Horte

Design and production
William Agush

Typesetting
JL Associates

TYPOGRAPHIC CONVENTIONS

abbreviation of PRIMOS commands The minimum required abbreviation of PRIMOS commands is shown in rust colored letters Only internal commands can be abbreviated

braces { }· Of a group of words or parameters contained within braces at least one must appear in command or statement

comma Where a comma appears in a BASIC/VM statement it is required

parentheses ( ) Parentheses where they appear, are a required literal part of the command or statement syntax

square brackets [ ] A word or parameter enclosed in square brackets is optional

PRIMOS CONCEPTS

binary file· A translation of source file generated by the BASICV compiler

byte: 8 bits, 1 ASCII character

directory: A PRIMOS file directory a special kind of file containing a list of files and/or other directories, along with information on their characteristics and location MFDs, UFDs, and subdirectories (sub-UFDs) are all directories (Also see **segment directory**)

file· An organized collection of information stored on a disk (or a peripheral storage medium such as tape) Each file has an identifying label called a filename

filename: A sequence of 32 or fewer characters which names a file or a directory Within any directory, each file name is unique Directory names and a filename may be combined into a pathname Most commands accept a pathname wherever a filename is required

Filenames may contain only the following characters

A Z 0-9 ___ # S + *

The first character of a filename must not be numeric On some devices underscore (___) prints as backarrow (←)

file unit A number between 1 and 63 ('77) assigned as a pseudonym to each open file by PRIMOS This number may be given in place of a filename in certain commands, such as CLOSE PRIMOS-level internal commands require octal values The maximum number of units that each user may have opened at one time is determined on a *per-installation basis Certain commands or activities use* particular unit numbers by default, e g , unit 127 reserved for COMOUTPUT files

PRIMOS assigned units	Octal	Decimal
INPUT, SLIST	1	1
LISTING	2	2
BINARY	3	3
AVAIL	5	5
COMINPUT	6	6
SEG's loadmap	13	11
COMOUTPUT	77	63
EDITOR	1,2	1,2
SORT	1-4	1-4
RUNOFF	1-3	1-3

pathname· A multi-part name which uniquely specifies a particular file (or directory) within a file system tree A pathname (also called treename) gives a path from the disk volume, through directory and subdirectories to a particular file or directory Pathnames and filenames can be used interchangeably in most PRIMOS and BASIC commands

segment directory A special form of directory used in direct access file operations Not to be confused with directory which means "file directory"

source file: A file containing programming language statements and data as entered from the terminal

subdirectory (also called **sub-UFD**) a directory that is in a UFD or another subdirectory

treename A synonym for pathname

ELEMENTS OF BASIC

array: A list or table of contiguous numeric or string values in one- or two-dimensional form. Arrays are named by singly or doubly subscripted numeric or string variables, e.g., A(1) or A(1,2). See also matrix.

characters: The following characters are accepted by the BASIC/VM subsystem:

- Upper and lowercase letters A-Z
- Digits from 0-9
- Special characters ' +*/(), $ blank (space)

commands: Directives to BASICV subsystem issued at command level in upper or lowercase ">", in response to ">" prompt. Commands do not require line numbers as do statements. Some commands may be used as statements in programs and are so indicated in the list of system commands.

comments. May be included in programs for notation and are preceded either by REM or a '. They may be in upper or lowercase and are ignored by the system.

constants: Can be either a numeric or a literal (quoted) string whose value does not change during program execution.

numeric string	positive or negative integers, decimal or exponential expressions
literal string	sequence of characters enclosed in single or double quotes. Maximum length is 160 characters

data type. BASIC/VM supports double precision floating-point numeric data and string data. Numbers have up to 13 significant figures in the mantissa and 2 significant figures in the exponent.

expressions: Various ordered combinations of constants, variables, operators, and functions that can be arithmetically or logically evaluated.

foreground file: The file currently open in the user's working directory.

functions BASIC/VM provides a set of numeric and string system functions identified by a 3 or 4 letter name plus a dollar sign ($) for string functions followed by parenthetically enclosed arguments. BASIC/VM also supports both numeric and string user defined functions. User defined numeric functions are identified by the letters FN followed by a numeric variable e.g. FNQ, FNQ8. User defined string functions are named by the letters FN followed by a string variable e.g. FNQ$.

matrix A matrix is that part of a one or two dimensional array with non zero subscripts. Example

Array A			
(0 0)	(0 1)	(0 2)	
(1 0)	(1 1)	(1 2)	matrix A
(2 0)	(2 1)	(2 2)	

The complete set of BASIC/VM matrix operations is found in the rear.

operands Elements manipulated by a program. These are constants, variables and arrays.

operators Connect operands and indicate how they are to be manipulated by the program. BASIC/VM supports three types of operators: arithmetic, logical, relational.

operators arithmetic Unary or binary. Unary operations indicate the sign (+ or -) of a number. Binary operations require two operands e.g. A+B.

Operator	Definition	Example
+	Addition (unary positive)	A + B + A
-	Subtraction (unary negative)	A - B - A
*	Multiply	A*B
/	Divide	A/B
^ or **	Exponentation	A^B, A**B
MOD	Remainder from division modulus	A MOD B
MIN	Select lesser value	A MIN B
MAX	Select greater value	A MAX B

operators, logical. Connectives for relational expressions

Operator	Meaning	Form
AND	True if both A and B are true	A AND B
OR	True if either A, B or both are true	A OR B
NOT	If A is true NOT A is false	NOT A

operators, relational: Used with conditional statements and statement modifiers. There are six relational operators

Operator	Meaning	Example
<	Less than	A<B
>	Greater than	A>B
=	Equal	A$=B
<= <=	Less than or equal	A<=C
>= =>	Greater than or equal	A=>C
<> ><	Not equal	A<>D

operators, priority of: Expressions are evaluated in order of operational priority. The priority list from highest to lowest for BASIC/VM is

( ) Parenthetical Expressions
FN System and User-defined Functions
^ (or **) Exponentiation
NOT Unary (+ -)
* / MOD
+, -
MIN, MAX
Relationals (=, > <, =>, <=, <>)
AND
OR

Within each level the evaluation order is from left to right.

operators, string: String operands take only the above relational operators plus a concatenation operator (+) for combining two strings.

statements: Statements are upper or lowercase directives included in a program and preceded by a line number. Some may be used as commands and as such are not preceded by line numbers.

statement syntax Statements must adhere to the following rules

1. Each statement must be contained on one line
2. Statements must not exceed 160 ASCII characters in length
3. Portions of the statement (i.e. string literals) which the user wishes processed verbatim must be enclosed in single or double quotes
4. Statements should be separated from their identifying line numbers with a blank space to avoid misinterpretations
5. Statements cannot be abbreviated

statement numbers Statement numbers are one to five digit numbers ranging from 1 to 99999. Successive statements are generally numbered in ascending order in increments of 10 for ease of insertion of new statements.

variables Variables are representations of data to which values are assigned. BASIC/VM supports four types of variables.

numeric scalar	Single letter (A-Z) optionally followed by a single digit (0-9). 286 may be defined per program. Initialized to zero at the start of program execution.
string scalar	Single letter (A-Z) followed by a dollar sign ($) or by an optional decimal digit and a dollar sign. Initialized to null at start of program execution.
numeric subscripted	Single numeric variable followed by one or two values enclosed in parentheses. Also called an array.
string subscripted	Single string variable followed by one or two values enclosed in parentheses. Also known as string arrays.

Legal and Illegal Variables

Type	Legal		Illegal	
numeric	A2	A	AB1	AR
scalar	X4	Z	X14	BZ
string	B$		AB$	AB3$
scalar	A2$		A21$	
numeric	A2(1)	A(1 2)	A12(1)	
subscripted	A(1)	A2(1 2)	AB(1 2)	
string	A$(1)	A$(1 2)	A12$(1 2)	
subscripted	A2$(4)	A2$(1 2)	AB$(1)	

COMMANDS

Command abbreviations are in rust

ALTER line-number

Changes any portion of specified line with parameters listed below Returns colon prompt until QUIT is typed

Parameter	Effect
A/string/	Append **string** to end of line
Bnn	Move pointer back **nn** characters (where nn is any integer)
Cc	Copy line up to but not including c (where c is any character)
Dc	Delete line up to but not including c
Enn	Erase **nn** characters
F	Copy to end of line
I/string/	Insert **string** at current position (The slash may be any delimiter not used as part of the string)
Mnn	Move **nn** characters
N	Reverse meaning of next COPY parameter (copy until character = <c or delete until character = >c)
O/string/	Overlay **string** on line from current position A ! changes a character to a space a space leaves character unchanged

Q	Exit from ALTER mode.
R/string/	Retype line with **string** from current position
S	Move pointer to start of line

ATTACH pathname

Attaches to directory specified by **pathname**

BREAK { ON / OFF } **lin-num-1 [....lin-num-n]**

Sets and unsets breakpoints at specified statement lines. Maximum of 10 may be set

CATALOG [options]

Lists all filenames under current directory

options	
DATE	Returns date and time when file was last modified
PROTECTION	Returns protection attributes on file
SIZE	Returns size of each file (in records)
TYPE	Indicates file type
ALL	Returns all of above information

CLEAR

Resets all previously defined numeric variables to zero, all string variables to null Deallocates defined arrays and closes open files

COMINP { **pathname** / CONTINUE / PAUSE / TTY }

Opens and reads commands in file specified by **pathname.** If control options are specified, command file halts at COMINP PAUSE resumes with **COMINP CONTINUE.** Reads commands in file until **COMINP TTY** is reached. Takes unquoted argument Also used as a statement

COMPILE [pathname]

Translates source file into executable binary form Displays compile-time errors Optional **pathname** specification saves binary file to disk All Rev 16 programs must be recompiled to run under Rev 17

CONTINUE

Resumes program execution after PAUSE or breakpoint

DELETE { **lin-num-1,...lin-num-n** | **lin-num-1 — lin-num-n** }

Deletes specified statement lines from program

EXECUTE [pathname]

Executes indicated file (binary or source) or foreground file if no **pathname** is specified also displays run time errors

EXTRACT { **lin-num-1,.. lin-num-n** | **lin-num-1 — lin-num-n** }

Deletes all except specified lines Statements must be in ascending order

FILE [pathname]

Saves all input and modifications to current file under original filename or to specified **pathname**

LBPS

Lists currently set breakpoints

LENGTH

Reports total number of statements in current program

LIST [NH] [**lin-num-1 . lin-num-n** | **lin-num-1 — lin-num-n**]

Displays contents of foreground file or specific lines of file NH option suppresses program header (time date etc)

LOAD pathname

Merges external program with foreground program Line numbers in the external file duplicated in the foreground file are overwritten by those in the external file If loaded file is binary, it is loaded into user memory and is not merged with foreground file

NEW [pathname]

Indicates new file is to be created with specified name

OLD [pathname]

Calls pre-existing file to foreground

PERF { ON / OFF / HIST / TABLE } { lin-num-1 - lin-num-2 / screen-size { AVG / CNT / TTL } lin-num-1 - lin-num-2 }

Turns performance measurement feature ON or OFF, measures program efficiency Must be issued prior to compilation TABLE option prints these statistics

AVG	average statement execution time
CNT	number of times each statement was executed
DEV	standard deviation of execution time
SN	statement number
SQSUM	total squared-sum of statement run-time
TTL	total running time of each statement

Times are measured in 'ticks' at 3 03 msec per tick **lin-num-1** specifies statement number at which to start display **lin-num-2** statement number at which to stop display

HIST displays statement statistics in histogram form scaled according to **screen-size** in number of characters, default is currently set margin (default margin = 80 chars) Symbols used in histogram display are

- **CNT**
- **AVG**
+ **TTL**

PURGE [pathname]

Deletes specified file from directory Default deletes foreground file File must be closed in order to PURGE

QUIT

Returns control to PRIMOS from BASICV command level Closes all files opened by BASICV and deletes temporary files created by BASICV

RENAME newname

Changes name of foreground file but does not rename original disk copy of the file two copies of the same file will exist with different names if renamed file is FILED

RESEQUENCE [new-start] [old-start] [,new-incr]

Renumbers statement in the foreground program **new-start** is the actual number with which line renumbering will begin (Default 100) **old-start** is the existing line number at which to begin renumbering (Default lowest numbered line) **new-incr** specifies increment value (Default is 10)

RUN [NH] [lin-num]

Begins compilation and execution of foreground source file (at **lin-num** if option specified) Prints program name No binary file stored

TRACE { ON / OFF }

Displays in brackets all statement numbers as they are executed until **TRACE OFF** is typed Used to examine program logic flow

TYPE pathname

Displays specified file at terminal but does not replace file in foreground

STATEMENTS

ADD #unit str-expr-1 { PRIMKEY / KEY-zero-expr / KEY } **str-expr-2 keylist**

where keylist [,KEY num-expr-1 str-expr-3]*

Adds record **str-expr-1** to MIDAS file opened on unit A primary key **PRIMKEY KEY-zero-expr** or **KEY** and its value **str-expr-2** must be supplied One or more secondary keys may be specified in keylist which contains the names **num-expr-1** and value(s) **str-expr-3** of the secondary key(s) * indicates repetition of expression as necessary

CALL subr-name (arg [arg])

Calls any declared and shared system non system or library routine from within a BASIC VM program See **SUB FORTRAN**

CHAIN pathname

Closes all open files and transfers program control to external program specified by **pathname**

CHANGE { **num-array** / **str-expr** } **TO** { **str-var** / **num-array** }

Transforms ASCII character string, **str-expr**, into a one-dimensional numeric array (**num-array**) containing the decimal value of the string, or transforms a numeric array to its ASCII equivalent, **str-var** ASCII characters and their decimal equivalents are listed in the rear

CLOSE #unit-1,...unit-n

Closes file opened on **unit**, where **n** is maximum of 12

CNAME oldname TO newname

Changes name of specified file

COMINP { **pathname** / **CONTINUE** / **PAUSE** / **TTY** }

Stops execution of current program and executes commands from command file specified by **pathname** a string expression **COMINP PAUSE** and **COMINP CONTINUE** temporarily halt and restart a program respectively Commands in file are executed until **COMINP TTY** is reached Also used as a command

DATA item-1,...item-n

Lists numeric and string constants to be accessed by a READ statement

DEFINE [READ / APPEND] **FILE #unit** filename[,type-code][,record-size]

Opens file, named by **filename** on specified **unit** Optionally assigns file type and access method, indicated by **type-code** Type-codes are listed in the following table If no type-code is given the default (ASC) is assumed Record size (default = 60 words) can be increased or decreased by specifying **record-size**, a numeric expression For MIDAS files, record-size should be set equal to the combined length of the data record and the primary key specified during CREATK Access may be restricted to read or append with the **READ** or **APPEND** options respectively A file DEFINEd as a READ file is assumed to exist

Note

The terminal can be assigned as a file unit using the (ASR ') filename

Table of Type-Codes

Type-Code	Access Method	Contents
ASC (default)	SAM	ASCII data, formatted like terminal output, using BASICV PRINT conventions, e g commas, colons and semi-colons all dictate the appropriate number of spaces to be used as data delimiters Records variable-length and easily inspected
ASCSEP	SAM	ASCII data stored with commas inserted as data delimiters Data are stored and read back exactly as entered Records fixed-length, accessed sequentially
ASCLN	SAM	ASCII data with comma delimiters and line numbers inserted in increments of 10 at the start of each record Takes 6 characters Designed to be edited as BASICV command level
ASCDA	DAM	Similar to ASCSEP Records fixed-length and blank-padded as necessary Direct access method used for quick, random access to any record in the file

BIN	SAM	Data storage transparent to user Records are fixed-length, accessed sequentially String data stored in ASCII code numeric data stored in four-word floating-point form Provide maximum precision and compactness of numeric data, but cannot be inspected by TYPE etc
BINDA	DAM	Same as BIN but direct access method is used for random record access Records not data-filled are zeroed out
SEGDIR	SPECIAL	Identifies file as a segment directory Subordinate files, identified by number, may be SAM, DAM or other SEGDIR files An additional DEFINE is required to access a subordinate file
MIDAS	SPECIAL	Multiple Index Data Access files Created by Prime-supplied MIDAS utilities

DEFINE SCRATCH FILE #unit [,file-type] [,record-size]

Opens a temporary file on specified **unit** When unit is closed, the scratch file is automatically deleted

DEF FN var [(arg-1 ...arg-n)] = expression

Defines a one line function named by var (a string or numeric variable), with no FNEND statement Arguments (arg-1 to arg-n) are numeric to string scalar variables only

DEF FN var [(arg-1 ...arg-n)]

.
.
.

FNEND

Defines a user-defined numeric or string function, of one or more lines The last line must be **FNEND var** is a simple numeric or string variable **arg-1** to **arg-n** are dummy arguments for the function may be numeric or string scalar variables

DIM array { **(num-con)** | **(num-con-1, num-con-2)** }

Defines the dimensions of a numeric or string array represented by **(num-con)** and **(num-con-1 num-con-2)**, numeric constants Default (10) or (10 10) Variables are not legal in DIM statements

DO

.
.

DOEND
ELSE DO

.
.

DOEND

Sets up a series of statements in association with IF-THEN statements, executed if a specified condition is met **DOEND** indicates the end of the series **ELSE DO** is an optional alternative to previous set of DO statements.

END

Terminates program execution Serves as messageless STOP

ENTER time-limit, time-var, var

Allows a specified number of seconds **time-limit**, range 1 to 1800 for user input of a value to numeric or string variable, **var** indicated No prompt is given **time-var**, a numeric variable, represents the actual time taken to enter value Only one value can be input from the terminal with each ENTER statement

ENTER # user-num-var [,time-limit time-var, num-var]

Sets user number assigned at LOGIN to numeric variable, **user-num-var** Remainder of options same as for ENTER above

ERROR OFF

Turns off all error traps in conjunction with ON ERROR GOTO mechanism

FOR index start TO end [STEP incr]

Specifies beginning of loop Used with NEXT statement The loop index is specified by **index**, a numeric variable the initial value of the index is set to **start**, a numeric expression, the increment value is set by **incr** and the final value of the index is represented by **end**, a numeric expression

FOR index start STEP incr { WHILE / UNTIL } **condition-expr**

Specifies the beginning of a conditional loop **condition-expr**, a conditional expression, determines how long the loop will be executed See above for other parameters

GOSUB lin-num

Unconditionally transfers program control to an internal subroutine beginning at specified **lin-num** A RETURN must be executed terminating subroutine Up to 16 GOSUB statements may be nested

GOTO lin-num

Transfers program control forward or backward to a specified **lin-num** A loop is created when the specified line-number appears prior to the GOTO statement May be used with IF

IF expr { GOTO **lin-num-1** / THEN **lin-num-1** / THEN **statement-1** } [ELSE { **lin-num-2** / **statement-2** }]

Transfers program control depending on the value of a relational, logical or numeric expression, expr **lin-num** is the statement number to which program control is transferred if the expression is true **statement-1** is executed if the preceding expression is true If the expression is not

true either **statement-2** will be executed or control will be transferred to **lin-num-2**, depending on which if any, is specified If **expr** is not true, and no alternative is provided, the next sequential statement is executed

INPUT ['prompt-string,] var-1 var-n

Prompts user for input specified by var-1 through **var-n** which are either numeric or string variables or array elements separated by commas If no prompt string is provided, the default prompt character (?) is returned

INPUT LINE ['prompt-string] str-var

Prompts user with optional prompt string, for **str-var**, a string variable or string array element Accepts entire input line, including colons commas, and leading blanks as one entry

[LET] var expr

The assignment statement optional **var** represents a numeric or string variable or array element **expr** is either a numeric value string expression or another variable

LOCAL { **var-1 . var-n** | **DIM var-1 (dim-1) ,(dim-2)** }

Declares listed variables **(var-1 -var-n)** as local to function definition in which they appear **(dim-1)** and **(dim-2)** represent dimensions in a one- or two-dimensional array or matrix Loval variables cannot be LISTed during a PAUSE or BREAK

MARGIN { **value** | **OFF** }

Sets number of characters per line to value, a numeric expression Range is 1 to 32767, the default is 80 **MARGIN OFF** turns off all margin checking

MAT mat - { **ZER** | **CON** | **IDN** | **NULL** } [**(dim-1)** | **(dim-1,dim-2)**]

Sets initial value of matrix elements to zero, one, identity or null, respectively Also used to redimension a one-dimensional matrix to **dim**, (a numeric expression) or a two-dimensional matrix to **dim-1,dim-2** NULL can only

be used for nulling string matrices IDN transforms a matrix into an identity matrix, one in which all elements, except those on main left-to-right diagonal, are 0 the main diagonal elements are 1

MAT mat-3 mat-1 { + / - } **mat-2**

Adds, subtracts or multiplies the elements of mat-1 and mat-2 to form a target matrix mat-3 In multiplication, the target matrix dimensions are the number of rows of mat-1 and the number of columns of mat-2

MAT mat-2 = (expr) mat-1

Multiplies each element of mat-1 by a specified numeric value expr and assigns results to mat-2 If mat 2 exists; its elements will be redefined and its dimensions will be changed to that of mat-1

MAT mat-1 INV (mat - 2)

Assigns the inverse values of a square matrix **mat-2** (Determinant not equal to 0) to the target matrix, **mat-1.**

MAT mat-1- TRN (mat-2)

Calculates the transpose of the values of mat-2 and assigns them to target mat-1 A matrix is transposed by rotating it along the main diagonal

MAT INPUT [prompt-string] mat-1[**mat-2**] [...{**mat (*)** / **mat-n**}]

Reads data from the terminal and assigns the values to specified matrices mat-1 throughmat-n mat (*) indicates that elements may be input until a new line is typed Matrix is automatically dimensioned to number of input elements Default prompt character is '?'.

MAT PRINT mat-1 [...mat-n]

Prints indicated matrices mat-1 to mat-n at terminal If a matrix name is followed by a colon instead of a comma, the elements will be separated by spaces instead of columns when printed

MAT READ mat-1 [...mat-n]

Reads values from a data list and assigns them to the elements of the specified matrix or matrices until matrix is filled

MAT READ [*] #unit, mat-1 [, ..mat-n]

Reads data from an external file and assigns them to elements of specified matrix or matrices Optional * indicates that all data from current record should be read before a new record is read

MAT WRITE #unit, mat-1 [,...mat-n]

Writes an entire matrix or matrices to a file on the specified unit

NEXT num-var

Defines the end of a loop beginning with a FOR statement The **num-var** matches the variable used with the companion FOR statement

ON num-expr GOSUB lin-num [.. .lin-num-n]

Transfers program control to a subroutine at a specified line number depending on value of a numeric expression **num-expr** When RETURN statement is reached, control returns to statement following ON **GOSUB** The value of **num-expr** must be less than or equal to the number of statement lines listed else error occurs If num-expr = 1, control transfers to **lin-num-1** if num expr=2, control transfers to **lin-num-2**, and so on

ON num-expr GOTO lin-num-1,...lin-num-n

Transfers program control to one of a list of line numbers (**lin-num-1** to lin-num-n) depending on the value of the numeric expression (**num-expr**) The value of num expr must be less than or equal to the number of statement lines value listed If the expression value exceeds the number of lines listed an error message is displayed

ON END #unit GOTO lin-num

Establishes a line number to which program control will transfer when an END OF FILE occurs on specified **unit**

ON ERROR GOTO lin-num

Establishes a line number to which program control transfers when a run-time error occurs Two variables, ERR and ERL, and the function ERRS (num expr) are associated with ON ERROR GOTO

ERR	Variable set to the code number of the error which activated the ON ERROR statement
ERL	Line number being executed when the error occurred
ERR$ (num-expr)	Outputs actual text of error message associated with an error code represented by a numeric expression, num-expr

ON ERROR #unit GOTO lin-num

Establishes a statement line to which program control transfers when an I/O error occurs on the specified **unit**

PAUSE

Acts as a BREAK command Suspends program process at line where PAUSE occurs To resume program type CONTINUE

POSITION #unit TO record-number

In direct access files, positions the internal record pointer to a specified **record-number** in a file on the specified **unit** Works on ASC DA and BIN DA files The error message, END OF FILE, is displayed when pointer is positioned past last record in file

POSITION #unit { SEQ / KEY [**num-expr**] - **str-expr** / SAME KEY }

Positions a file read pointer to a specified record in a MIDAS file opened on **unit** If a secondary key number, **num-expr**>0 and value **str-expr** are not indicated, pointer will position to primary key If **SEQ** is supplied in lieu of key, the next sequential record is positioned to **SAME KEY** positions to datum only if next key matches current one

PRINT [**item-1** [{ LIN / TAB / SPA } (**num**)] ... **item-n** [{ LIN / TAB / SPA } (**num**)]]

Prints formatted information at the terminal **Item-1** to **item-n** represent numeric and/or string values **LIN** forces the specified number (**num**) of carriage return — line feed combinations between items in the output if number>0

TAB forces tab to specified column number SPA forces number (num) of spaces between items in output Num specifies number of blank lines, tab positions or spaces to be printed in the output A comma in a print list causes next item to be printed in next print zone Each print zone contains 21 characters Semi-colons cause no spaces to occur between printed items Colons force one space between items

PRINT USING format-string item-1 item-n

Generates formatted output according to format characters in format-string including a dollar sign plus or minus signs decimal points and right-left justification item-1 through item-n represent string or numeric values

A **format-string** may be a string constant or a string variable

Numeric format field characters

#	Specifies number of positions in field for corresponding digits Forces rounding off if too few # s are indicated for decimal number A row of asterisks is printed if too few # s are indicated for integer
.	Forces decimal point to be included at appropriate position in number
,	Forces comma to be inserted at appropriate position in number unless all digits preceding comma are zeros
^	Forces representation of number in exponential form at indicated position Each ^ represents ^ 1 digit in the exponent field
+	Forces sign of number to be printed where indicated
-	Forces minus sign to be printed where indicated
$	Forces dollar sign to be printed where indicated
<	Left-justifies item in field
>	Right justifies item in field
#	Specifies number of positions in field for corresponding character string item

READ var-1 ...var-n

Reads numeric or string values from a DATA statement within the program **var-1** through **var-n** are string or numeric variables separated by commas. Begins accepting values with first item in lowest numbered DATA statement

READ [KEY] #unit [{ SEQ / [KEY [num-expr] str-expr], / SAME KEY }] **str-var**

Reads data from specified record in MIDAS file on **unit** Data is read into **str-var** If **READ KEY** is specified, the key value is read into **str-var** **Num-expr** and **str-expr** are the key numbers and values, respectively of the primary or secondary key **SEQ** reads next sequential record **SAME KEY** returns datum only if next key matches current one

READ LINE #unit str-var

Accepts entire line of text (including commas and colons) as one data item and puts it in **str-var**

READ #unit, var-1,...var-n

Forces program to read a new record from the file on **unit** **var-1** through **var-n** are values to be read beginning with the first value in the current record.

READ * #unit, var-1,...var-n

Forces continued reading of data in current record before new one is read **var-1** through **var-n** are values to be read from current record and next record as necessary

REM string

Indicates remark to reader Exclamation point [!] is substituted for REM when comments are added to executable statements

REMOVE #unit [, KEY[num-expr] - str-expr]+

Deletes specified key from MIDAS file If primary key, **num-expr** = 0, is specified, data associated with key are removed also Multiple keys may be deleted with one statement line + indicates that bracketed expression may be repeated as necessary

REPLACE #unit SEG x BY SEG y

Deletes files referenced by indicated segment directory (**SEG x**) on unit Pointer at **SEG y** (segment y) is moved to segment x, old pointer at SEG y is zeroed

RESTORE { # / $ }

Instructs program to reuse list of data items beginning with first item in lowest numbered DATA statement Numeric data items are reused by specifying #; string items, by $ Both numeric and string items are reused if neither symbol is specified RESTORE must precede READ # statement indicating data items to be reused

RETURN

Causes control to be returned from GOSUB subroutine

REWIND unit-1 [,unit-2,. .unit-n]

Repositions record pointer to top of file on specified **unit** or units

REWIND #unit [,KEY num-expr]

Places pointer at top of MIDAS file opened on **unit**, at column specified by **KEY num-expr** If num-expr=0 or is unspecified, pointer is positioned to primary key (default)

SUB FORTRAN subr-name (arg-format,...[,arg-format])

Declares any shared system, non system or library routine which observes the FORTRAN calling sequence inside a BASIC/VM program Routines cannot be called from BASIC/VM unless so declared

STOP

Causes termination of program execution Returns message STOP AT LINE lin num

UPDATE #unit [,str-expr]

Writes string expression **str-expr** to current MIDAS file open on **unit** Overwrites the current record Beware of changing keys with UPDATE if keys are being stored in record

WRITE #unit, item-1,. .item-n

Writes data, string or numeric, specified by **item-1** through **item-n**, (string or numeric variables), to the current record or output device opened on **unit** If no values are specified a blank line appears in the output If file is closed after WRITE # statement all subsequent records in file are truncated

WRITE #unit USING format-string, item-1,...item-n

OR

WRITE USING format-string, #unit item-1,...item-n

Formats items according to format characters in **format-string**, including tabs, spaces, and column headings Output is written to current record or output device opened on **unit item-1** through **item-n** are numeric or string variables or expressions A **format-string** may be a string constant or a string variable See PRINT USING statement for format characters

SYSTEM FUNCTIONS

BASIC/VM provides both numeric and string system functions for use in programming User defined functions are also supported

NUMERIC SYSTEM FUNCTIONS

Parameters

X Represents any numeric expression
Y,Z Represent any integers
X$ Represents string expression

ABS(X)	Computes the absolute value of X
ACS(X)	Computes the principal arccosine of X The result is in radians in the range of 0 to π 360 degrees=2π radians
ASN(X)	Computes the principal arcsine of X The result is in radians in the range of $-\pi$ 2 to $\pi/2$
ATN(X)	Computes the principal arctangent of X The result is in radians in the range of $-\pi$ 2 to $-\pi/2$

COS(X)	Computes the cosine of X The argument is in radians The result is in the range -1 to +1
COSH(X)	Computes the hyperbolic cosine of X, defined as (EXP(X)+EXP(-X/2)
DEG(X)	Computes the number of degrees in X, [(180/π/*X]
DET(X)	Computes the determinant of matrix X If DET(X) unequal to 0 matrix X has an inverse
ENT(X)	Computes the greatest integer that is less than or equal to X
ERL	Returns the statement number of the line which caused an error
ERR	Returns the error code number of the last error
EXP(X)	Computes e raised to the X power
INT(X)	If X=>0, returns the greatest integer <=X If X<0 returns the least integer >-X INT performs integer truncation
LIN#(X)	For ASC LN files, returns the statement number stripped from the last input on unit X For DA files, returns the current record positioned to in the file on unit X
LOG(X)	Computes the natural logarithm (base e) of X
NUM	Returns the actual number of entries to MAT INPUT M(*) statement Matrix M is one-dimensional
PI	Computes the value of π (3 14159)
RAD(X)	Computes the number of radians in X degrees
RND(X)	If X>0, uses X to initialize the random number generator and returns X as the function value If X<0, uses X to initialize the random number generator, and returns a value in the range zero to one If X=0 returns a random number in the range zero <=result 1

SGN(X)	Computes a value based on the sign of X as follows X<0 SGN(X)=-1 X=0 SGN(X)=0 X>0 SGN(X)=1
SIN(X)	Computes the sine of X The argument is in radians The result is in the range -1 to +1
SINH(X)	Computes the hyperbolic sine of X defined as (EXP(X)-EXP(-X))/2
SQR(X)	Computes the positive square root of X
TAN(X)	Computes the tangent of X The argument is in radians
TANH(X)	Computes the hyperbolic tangent of X defined as (EXP(X)-EXP(-X)/ EXP(X)+EXP(-X))

STRING SYSTEM FUNCTIONS

CHAR(X)	Returns the character whose ASCII code is X. X is in the range 128-255
CODE(X$)	Computes the decimal ASCII code of the first character of X$
CVT$$(X$,Y)	Reformats X$ according to the mask Y (Masks are listed in the following table.)
DATE$	Returns the date as YYMMDD
INDEX(X$,Y$,[Z])	Computes the starting position Y$ in X$ optionally beginning at character Z
LEFT(X$,Y)	Returns leftmost Y characters of X$
LEN(A$)	Returns the length (number of characters) of string A$
TIME$	Returns the time as HHMMSSFF (FFF is milliseconds)
MID(X$,Y,Z)	Returns Z characters of X$ starting at position Y
RIGHT(X$,Y)	Returns rightmost characters of X$ beginning with character number Y
STR$(X)	Returns the string representation of the number X
SUB(X$,Y,[Z])	Returns the substring composed of characters Y through Z of string X$. If Z is not specified the result is a one character substring consisting of character Y of string X$
VAL(X$,[Y])	Converts a string to the number it represents. Y returns the conversion status 0=successful 1=unsuccessful

MASKS FOR CVT$$

Masks can be combined additively

Mask	Function
1	Force parity bit off
2	Discard all spaces
4	Discard NUL, NL FF CR ESC
8	Discard leading spaces
16	Reduce multiple spaces to one space
32	Convert lower case to upper
64	Convert { to (and } to)
128	Discard trailing spaces
256	Converts upper case to lower case

USER-DEFINED FUNCTIONS

Users may define their own functions with the DEF FN statement Numeric function names are identified by the letters FN followed by a letter or a letter and a digit, as in FNA, FNA4 String functions are identified by FN followed by a string scalar variable as in FNQ$ FNQ1$ The arguments to a user-defined function must be numeric or string scalar variables If the function definition is more than one line in length, the last line should be FNEND A user-defined function is not executed until it is referenced in the program A reference consists of the name of the function followed by a parenthetically enclosed argument expression e g z=FNA(y)

RUNTIME ERROR MESSAGES

The following is a list of BASIC/VM error messages which appear at run-time (execution time).

Code number	Message
1	GOSUBS NESTED TOO DEEP
2	RETURN WITHOUT GOSUB
3	EXCESS SUBSCRIPT
4	TOO FEW SUBSCRIPTS
5	SUBSCRIPT OUT OF RANGE
6	ARRAY TOO LARGE
7	STORAGE SPACE EXCEEDED
8	BAD I-O UNIT
9	BAD FILE RECORD SIZE
10	DA RECORD SIZE ERROR
11	UNDEFINED I-O UNIT
12	WRITE ON READ ONLY FILE
13	END OF DATA
14	END OF FILE
15	FILE IN USE
16	NO UFD ATTACHED
17	DISK FULL
18	NO RIGHT TO FILE
19	ILLEGAL FILE NAME
20	FILE I-O ERROR
21	FILE NOT FOUND
22	INPUT DATA ERROR
23	VAL ARG NOT NUMERIC
24	BAD LINE NUMBER IN ASC IN FILE
25	ILLEGAL OPERATION ON SEGMENT DIRECTORY
26	READ AFTER WRITE ON SEQUENTIAL FILE
27	ILLEGAL OPERATION ON BINARY FILE
28	UNDEFINED MATRIX
29	ILLEGAL SEG DIR REFERENCE
30	ILLEGAL FILE TYPE FOR POSITION
31	ILLEGAL POSITION RECORD NUMBER

32	WRITE USING TO NON-ASCII FILE
33	PRINT USING STRING IN NUMERIC FORMAT
34	PRINT USING NUMERIC IN STRING FORMAT
35	PRINT USING FORMAT WITH NO EDIT FIELDS
36	BAD MARGIN SPECIFIER
37	MATRIX NOT SQUARE
38	MISMATCHED DIMENSIONS
39	OPERAND AND RESULT MUST BE DISTINCT
40	2 DIMENSIONAL MATRIX REQUIRED
41	INV MATRIX IS SINGULAR
42	MOD — SECOND ARGUMENT ZERO
43	EXPONENTIATION — BAD ARGUMENTS
44	SIN, COS — ARGUMENT RANGE ERROR
45	TAN — OVERFLOW
46	ASN, ACS — ARGUMENT RANGE ERROR
47	EXP — OVERFLOW
48	EXP — ARGUMENT TOO LARGE
49	LOG — ARGUMENT < = 0
50	SQRT — ARGUMENT < 0
51	EXPONENT OVERFLOW, UNDERFLOW
52	DIVISION BY ZERO
53	STORE FLOATING ERROR
54	REAL TO INTEGER CONVERSION ERROR
55	ON GOTO-GOSUB OVERRANGE ERROR
56	RECORD NOT FOUND
57	RECORD LOCKED
58	RECORD NOT LOCKED
59	KEY ALREADY EXISTS
60	SEGMENT FILE IN USE

61	INCONSISTENT RECORD LENGTH
62	RECORD FILE FULL
63	KEY FILE FULL
64	IMPROPER FILE TYPE
65	PRIMARY KEY NOT SUPPLIED
66	ILLEGAL OPERATION ON UNIT 0
67	FATAL MIDAS ERROR
68	0 RAISED TO 0 OR A NEGATIVE POWER
69	CONSTANT ON LEFT SIDE OF ASSIGNMENT STATEMENT
70	MIDAS CONCURRENCY ERROR

ASCII CHARACTER SET

Decimal Value (with parity on)	ASCII Character	Explanation
128		Null or fill character
129		Start of heading
130		Start of text
131		End of text
132		End of transmission
133		Enquiry
134		Acknowledge
135		Bell
136		Backspace
137		Horizontal tab
138		Line feed
139		Vertical tab
140		Form feed
141		Carriage return
142		Shift out
143		Shift in
144		Data link escape
145		Device control 1
146		Device control 2
147		Device control 3
148		Device control 4
149		Negative acknowledge
150		Synchronous idle
151		End of transmission block

152		Cancel
153		End of medium
154		Substitute
155		Escape
156		File separator
157		Group separator
158		Record separator
159		Unit separator
160		Space
161	!	Exclamation point
162	"	Double quotation mark
163	#	Number or pound sign
164	$	Dollar sign
165	%	Percent sign
166	&	Ampersand
167	'	Apostrophe
168	(	Open (left) parenthesis
169	)	Closing (right) parenthesis
170	*	Asterisk
171	+	Plus
172	,	Comma
173	-	Hyphen or minus
174		Period or decimal point
175	/	Forward slant
176	0	Zero
177	1	One
178	2	Two
179	3	Three
180	4	Four
181	5	Five
182	6	Six
183	7	Seven
184	8	Eight
185	9	Nine
186		Colon
187	,	Semicolon
188	<	Left angle bracket (less than)
189	=	Equal sign
190	>	Right angle bracket (greater than)
191	?	Question mark
192	@	Commercial at sign
193	A	(193 through 218 are upper case characters)

194	B	
195	C	
196	D	
197	E	
198	F	
199	G	
200	H	
201	I	
202	J	
203	K	
204	L	
205	M	
206	N	
207	O	
208	P	
209	Q	
210	R	
211	S	
212	T	
213	U	
214	V	
215	W	
216	X	
217	Y	
218	Z	
219	[	Open bracket
220	\	Backward slant
221	]	Closing bracket
222	^	Circumflex or up arrow
223	..	Underscore or backarrow
224		Grave accent
225	a	(225 through 250 are lower case characters)
226	b	
227	c	
228	d	
229	e	
230	f	
231	g	
232	h	
233	i	
234	j	
235	k	
236	l	

237	m	
238	n	
239	o	
240	p	
241	q	
242	r	
243	s	
244	t	
245	u	
246	v	
247	w	
248	x	
249	y	
250	z	
251	{	Open (left) brace
252	|	Vertical line
253	}	Closing (right) brace
254	~	Tilde
255		Delete