IBM BASIC Language Reference Summary

SX26-3736-1
File No. S370-40

Second Edition (February 1986)

This edition makes obsolete, SX26-3736-0. It applies to Release 2 of IBM BASIC/VM, Program Product 5668-996, Release 2 of IBM BASIC/MVS, Program Product 5665-948. This edition will be updated from time to time; however, the basic documentation (*IBM BASIC Language Reference*, GC26-4026) is the authoritative source and will be first to reflect changes.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Requests for copies of this and other IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. Please direct any comments on the content of this publication to the address below. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150

BASIC Commands

AUTO
Provides automatic line number prompting.

```
AUTO [start-line] [STEP increment]
```

BREAK
Sets, removes, and lists breakpoints within a program.

```
BREAK [ON] line-num [,line-num]...
BREAK OFF [line-num [,line-num]...]
BREAK?
BREAK ?
```

CHANGE
Changes character strings within the statements in your workspace.

```
CHANGE [start-line [TO end-line]] delim
  old-string delim new-string [delim [A[LL]]]
CHANGE [line-number[.segment-number]]
```

COMPILE
Translates BASIC source programs into object programs.

```
COMPILE [prog-name] [OBJ[ECT] (obj-name)]
  [OUT (list-name)] [[OPTIONS]([option-list])]
```

COPY
Duplicates one or more lines in the workspace.

```
COPY start-line [TO end-line] AT target-line
  [STEP increment]
```

DELETE
Deletes the specified line, or group of lines, from the workspace.

```
DELETE start-line [TO end-line]
  [,start-line [TO end-line]]...
```


DROP
Erases the specified identifiers.

```
DROP [identifier [,identifier]...]
```

EXTRACT
Removes all lines, other than those specified, from the workspace.

```
EXTRACT start-line [TO end-line] [,start-line
  [TO end-line]]...
```

FETCH
Restores a program and its internal form to the workspace.

```
FETCH program-name
```

FIND
Locates character strings within a program line, or block of lines, and displays the containing lines on the terminal.

```
FIND [start-line [TO end-line]] delim string
  [delim [A[LL]]]
```

GO
Restarts a program that has been temporarily halted.

```
GO [line-number|END] [STEP]
GOTO line-number
```

HELP
Displays information about BASIC.

```
HELP [request]
```

INITIALIZE
Closes all open files and clears the workspace.

```
INITIALIZE  [program-name]
```

LIST
Displays some or all of the lines of the program in your workspace.

```
LIST [XREF] [ERROR[S] [ALL]] [OUT (list-file)]
  [line-number-range [,line-number-range]...]
LIST {FOR[WARD]|BACK[WARD]} [pages]
```

LOAD
Clears the workspace and loads a program from your library.

```
LOAD program-name
```

MERGE
Inserts statements from a file into the program currently in the workspace.

```
MERGE program-name [start-line] [TO end-line]
  [AT target-line] [STEP increment]
  [(REP[LACE] [)]]
```

PROFILE
Executes a profile during a session.

```
PROFILE profile-name
```

PURGE
Removes files from your library.

```
PURGE file-name
```

QUERY
Provides information about files.

```
QUERY [file-type] [file-name]
  [OUT (list-file [)]]
```

QUIT
Ends the current BASIC session.

```
QUIT [return-code]
```


RENAME
Either assigns a name to the program in your workspace or displays the current name associated with your workspace.

```
RENAME [program-name]
```

RENUMBER
Changes some or all of the existing line numbers of the program in the workspace.

```
RENUMBER [start-line [TO end-line]]
  [AT new-number]  [STEP increment]
```

RUN
Starts the execution of a program, starting with the lowest numbered statement.

```
RUN [program-name] [([SOURCE] [SPREC|LPREC]
  [PARM "literal"])] [STEP]
RUN object-name (OBJ[ECT]
  [PARM "literal"]) [STEP]
```

SAVE
Copies the source program in the workspace to a file.

```
SAVE [program-name] [(REP[LACE] [)]]
```

SET LOG
Activates or deactivates logging of the BASIC dialog with the terminal.

```
SET LOG {[ON] [OUT (file-spec)] [APPEND]|OFF}
```

SET MSG
Controls the content of messages displayed at the terminal.

```
SET MSG ({I|W|E|S}) {ALL|TEXT}
```

SET OPTION
Defines the default values for options.

```
SET OPTION  option [,option]...
```


SET PF
Establishes PF key definitions for command input.

```
SET PFn [IMMED|DELAYED] characters
SET PFn RETRIEVE
SET PFn
```

SET PROFILE
Controls display of lines and messages while reading a profile.

```
SET PROFILE {TERM|NOTERM}
```

SET TERM
Controls terminal input and output.

```
SET TERM {SCROLL|LINE}
```

STORE
Places the program currently in the workspace into a file.

```
STORE [program-name] [(REP[LACE] [)]]
```

SYSTEM
Executes an operating system command or enters the operating system subset mode.

```
SYSTEM ["operating-system-command["]]
```

BASIC Statements

BREAK
Suspends program execution.

```
BREAK
```

CALL
Invokes subprograms.

```
CALL name [(argument [,argument]...)]
```

CALL BASIC
Enables you to execute the SET TERM command in a BASIC program.

```
CALL BASIC (set-term-string)
```

CALL COBOL, FORTRAN or PLI Statement
Allow you to link to routines written in COBOL, FORTRAN, or PL/I.

```
CALL {COBOL|FORTRAN|PLI} (program-name
  [,argumment]...)
CALL {CLINK|FLINK|PLINK} [(character-expression
  [,character-expression]...)]
```

CALL GDDM
Performs graphic functions within an IBM BASIC program using the Graphical Data Display Manager (GDDM).

```
CALL GDDM (function [,argument]...)
```

CALL SQL
Accesses or manipulates data stored in a relational data base.

```
CALL SQL (SQL-stmt, status-info(),
  message-var [,value-list])
```

CALL SYSTEM
Allows you to execute a limited set of system commands.

```
CALL SYSTEM (character-expression)
```


CASE
Immediately precedes a group of statements within a SELECT block that are executed when the value of the selection expression in the SELECT statement satisfies the criteria of the CASE statement.

```
CASE selector [,selector]...
```

CASE ELSE
Immediately precedes a group of statements within a SELECT block that are executed if none of the CASE blocks are selected. The group of statements is referred to as a CASE ELSE block.

```
CASE ELSE
```

CAUSE
Generates an exception during processing.

```
CAUSE numeric-expression
```

CHAIN
Halts processing of the program in which it appears and starts a new main program.

```
CHAIN character-expression [,FILES ] [,name]...
```

CLOSE
Deactivates the specified file.

```
CLOSE #fileref [: [err [,err]]]
```

COMMON
Provides a means of sharing values between program units of a program or between programs when a CHAIN statement is executed.

```
COM[MON]
  {numeric-com | char-com[*length]}
  [,{numeric-com | char-com[*length]}]...
```

CONTINUE
Resumes execution at the statement following the statement causing an exception.

```
CONTINUE
```


DATA
Creates internal data tables for reference by READ statements.

```
DATA [integer*] item [,[integer*]item]...
```

DEBUG
Activates debugging facilities in a program.

```
DEBUG ON [TO #fileref]
DEBUG OFF
```

DECIMAL
Specifies which identifiers are to be assigned decimal type.

```
DECIMAL [{identifier|(letter-list)}
  [,{identifier|(letter-list)}]...]
```

DEF
Defines and names a user-defined function.

```
DEF name [(parameter [,parameter]...)]
  [=expression]
```

DELETE
Deletes a record from a keyed or relative file.

```
DELETE #fileref [,] pos [: [err [,err...]]
```

DIM
Specifies the size of arrays and the length of character variables and array elements.

```
DIM {numeric-dim | char-dim[*length]}
  [,{numeric-dim | char-dim[*length]}]...
```

DO
Initiates the execution of a set of statements that may be processed zero or more times.

```
DO [{UNTIL|WHILE} logical-expression]
```

ELSE
Specifies the beginning of the ELSE block portion of an IF block.

```
ELSE
```

END
Indicates both the physical and logical end of the main program.

```
END [numeric-expression]
```

END IF
Signifies the end of an IF block.

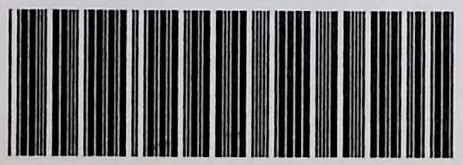
```
END IF
```

END SELECT
Signifies the end of a SELECT block.

```
END SELECT
```

IBM

SX26-3736-01



Printed in U.S.A.

END SUB
Marks the physical end of a subprogram.

```
END SUB
```

EXIT
Specifies where control is to be transferred if a particular condition occurs during the execution of an input/output statement.

```
EXIT condition line-reference
  [,condition line-reference]...
```

EXIT IF
Is used within a DO or FOR loop to transfer control to the statement immediately following the associated LOOP or NEXT statement when the EXIT IF clause is true.

```
EXIT IF logical-expression
```

FNEND
Indicates both the physical and logical end of a multistatement user-defined function.

```
FNEND
```

FOR
Initiates a FOR/NEXT count-condition loop.

```
FOR var=initial TO final [STEP increment]
```

FORM
Defines the exact appearance of both input and output data.

```
FORM item [,item]...
```

GET
Retrieves values from a stream or sequential internal file.

```
[MAT] GET #fileref : input-list [,SKIP REST]
  [err [,err]...]
```


GOSUB
Is used, together with the RETURN statement, to invoke subroutines.

```
GOSUB line-reference
```

GOTO
Unconditionally branches to the indicated line number or line label.

```
GOTO line-reference
```

IF
Evaluates a logical expression and conditionally transfers control or conditionally executes a statement or series of statements.

```
IF logical-expression THEN statement-reference
  [ELSE statement-reference]
```

IF Block
Begins an IF block.

```
IF logical-expression THEN
```

IMAGE
Specifies the format of print lines or output records for display files.

```
[IMAGE]:[character-string]
  [conversion-specification
  [character-string]]...
```

INPUT
Provides input through the terminal.

```
[MAT] INPUT [input-prompt:] input-list
  [err [,err]...]
```

INPUT FIELDS
Reads one or more data values from one or more specific fields of the terminal screen and assigns the value(s) to one or more variables.

```
INPUT [#fileref[,]] FIELDS field-definition:
  input-list [err [,err]...]
```

INPUT File
Reads either display or internal files, or from the terminal.

```
[MAT] INPUT #fileref [[,] input-prompt] :
  input-list [,SKIP REST] [err [,err]...]
[MAT] INPUT #fileref : input-list
  [,SKIP REST] [err [,err]...]
```

INTEGER
Specifies which identifiers are to be assigned integer type.

```
INTEGER [{identifier|(letter-list)}
  [,{identifier|(letter-list)}]...]
```

LET (Scalar Assignment)
Assigns values to both numeric and character variables.

```
[LET] variable [,variable]... = expression
```

LINE INPUT/LINPUT
Allows the unformatted input of character strings from a terminal.

```
[MAT] LINE INPUT [input-prompt:] input-list
  [err [,err]]
```

LINE INPUT/LINPUT File
Reads unformatted data from a display file.

```
[MAT] LINE INPUT #fileref [[,]input-prompt] :
  input-list [err [,err]...]
```


LOOP
Delimits a DO loop.

```
LOOP [{WHILE|UNTIL} logical-expression]
```

MARGIN
Specifies where output may begin and end on a terminal screen or page.

```
MARGIN numeric-expression
MARGIN [RIGHT numeric-expression]
  [LEFT numeric-expression]
  [TOP numeric-expression]
  [BOTTOM numeric-expression]
```

MARGIN File
Specifies the page margins for display type files being accessed by PRINT File statements.

```
MARGIN #fileref numeric-expression
MARGIN #fileref [RIGHT numeric-expression]
  [LEFT numeric-expression]
  [TOP numeric-expression]
  [BOTTOM numeric-expression]
```

MAT (Array Assignment)
Assigns values and dimensions to an array.

```
MAT arrayname = array-expression [(redimension)]
```

NEXT
Delimits a FOR loop.

```
NEXT variable
```

ON Condition
Indicates the action to be taken when an exception occurs.

```
ON condition action
```


ON GOTO/GOSUB
Conditionally transfers control to one of a group of statements. ON GOSUB also saves the return location.

```
ON numeric-expression {GOTO|GOSUB}
  line-reference [, line-reference]...
  [NONE line-reference | ELSE statement]
```

OPEN
Activates a file and specifies file attributes.

```
OPEN #fileref: [NAME] file-id
  [file-attributes] [err]
```

OPTION
Permits the selection of a variety of options that can be applied to a program.

```
OPTION option [,option]...
```

PAUSE
Halts execution of the program in which it appears.

```
PAUSE [pause-message]
```

PRINT
Displays data at the terminal.

```
[MAT] PRINT [output-list] [err [,err]...]
[MAT] PRINT USING image-or-form-reference
  [: [output-list] [err [,err]...]]
```

PRINT FIELDS
Displays one or more data values on a display terminal screen in the specified screen field(s).

```
PRINT [#fileref [,]] FIELDS field-definition
  [: [output-list] [;] [err [,err]...]]
```

PRINT File
Transmits data to a display type file.

```
[MAT] PRINT #fileref
  [[,] USING image-or-form-reference]
  [[,] FONT expression]
  [: [output-list] [err [,err]...]]
```

PUT
Places values in a stream file.

```
[MAT] PUT #fileref : output-list [err [,err]]
```

RANDOMIZE
Generates a new starting point for the list of pseudorandom numbers used by the RND function.

```
RANDOMIZE
```

READ
Retrieves the internal data table created by DATA statements.

```
[MAT] READ input-list [err [,err]...]
```

READ File
Retrieves a record from a native or an internal sequential file.

```
[MAT] READ #fileref [,] USING form-reference
  [[,pos] : input-list [err [,err]...]
[MAT] READ #fileref: input-list [,SKIP REST]
  [err [,err]...]
```

REAL
Specifies which identifiers are to be assigned real type.

```
REAL [SINGLE|DOUBLE] [{identifier|(letter-list)}
  [,{identifier|(letter-list)}]...]
```

REM
Inserts remarks into a program.

```
{REM|!} [remark]
```


REREAD
Makes the last accessed record in a native file available again.

```
[MAT] REREAD #fileref [,] USING form-reference:
  input-list [err [,err ]...]
```

RESET
Changes the position of a file pointer.

```
RESET #fileref [[,]pos] [: [err [,err]...]]
```

RESTORE
Lets you read the same data items more than once by restoring the data table pointer to the beginning, or other specified line, of the internal data table.

```
RESTORE [data-reference]
```

RETRY
Reprocesses a statement which caused an exception.

```
RETRY
```

RETURN
Returns control to the next executable statement following the GOSUB statement that called the subroutine.

```
RETURN
```

REWRITE
Updates an existing record stored in a native file.

```
[MAT] REWRITE #fileref [,] USING form-reference
  [[,] pos] : output-list [err [,err]...]
```

SCRATCH
Erases the contents of a file.

```
SCRATCH #fileref [: [err]]
```


SELECT
Begins a SELECT block.

```
SELECT expression
```

STOP
Terminates program execution.

```
STOP [numeric-expression]
```

SUB
Begins and names a subprogram.

```
SUB name [(parameter [,parameter]...)]
```

SUBEXIT
Stops execution of a subprogram and returns control to the subprogram's caller.

```
SUBEXIT
```

TRACE
Turns tracing ON or OFF.

```
TRACE ON [TO #fileref]
TRACE OFF
```

USE
Establishes a name correspondence in a chained program for those names passed via a CHAIN statement.

```
USE parameter [,parameter]...
```

WRITE
Adds records to native and internal files.

```
[MAT] WRITE #fileref [,] USING form-reference
  [[,] pos] : output-list [err [,err]...]
[MAT] WRITE #fileref : output-list
  [err [,err]...]
```