IBM BASIC

General Information

Program Product

B B B B B

A A A A A

S S S S S

I I I I I

C C C C C

Release Number: 2

Order Number:
GC26-4023-4

Program Numbers:
5668-996 (VM)

IBM BASIC

General Information

Program Product

Release Number: 2

Order Number:
GC26-4023-4

Program Numbers:
5668-996 (VM)
5665-948 (MVS)

Third Edition (December, 1985)

This is a major revision of, and makes obsolete, GC26-4023-3.

This edition applies to Release 2 of IBM BASIC/VM, Program Product 5668-996, Release 2 of IBM BASIC/MVS, Program Product 5665-948, and to any subsequent releases of either product until otherwise indicated in new editions or technical newsletters.

Changes are periodically made to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

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.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California, U.S.A. 95150. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Easy. Powerful. Versatile. BASIC Isn't Basic Anymore.

BASIC was introduced in the mid-1960's as a friendly, easy-to-use language, intended to introduce students to programming. BASIC (Beginners' All-purpose Symbolic Instruction Code) incorporated features that made it accessible to people who had never used a computer.

For example, BASIC statements were more similar to English than statements of other languages of the time, so it was easy for a first-time computer user to understand how BASIC worked. Statements like READ, PRINT, and SELECT made sense to a new user. Commands like RUN, SAVE, and LIST were equally comprehensible, and the syntax and punctuation were simple, straightforward, and easy to remember. BASIC also masked the inner workings of the machine, so no knowledge of such matters as registers or compilers was required.

Because BASIC was so easy to use, its popularity grew. Today, not only do more people know BASIC than any other computer language, BASIC has also become more widespread than some spoken languages. According to our best estimates, more people speak BASIC than Dutch, Swedish, or Czechoslovakian. As BASIC's popularity grew, so did its capabilities; the language has come a long way in the 20 years since its beginning.

Today's IBM BASIC expands on the original BASIC in three different ways. First, it has many features that make it even easier to use. Second, it has added many powerful capabilities that make it suitable, not only for the beginner, but for the advanced application programmer too. Last, IBM BASIC is now versatile enough to let you structure and execute your program in several different ways.

To make program development easy, IBM BASIC has introduced several new features. The new full-screen editing facility lets you make corrections to any program lines that appear on your screen, and BASIC tells you the number of lines you edited and any syntax errors you made. You also now have the option of invoking BASIC with a user profile, in which you can set many default options, including PF key settings. Full-screen editing and the user profile, combined with line-by-line syntax checking and on-line HELP, make for a friendly interactive environment that simplifies program development.

IBM BASIC has also introduced some new features that enable it to take on sophisticated applications. For example, BASIC can now access relational data bases. BASIC can also handle real data, which improves performance significantly. The new REAL data type, in conjunction with BASIC's wide range of intrinsic functions, supports complex mathematical applications.

BASIC is also flexible enough to handle your programs in several different ways. You can compile and execute your source code from either inside or outside BASIC's interactive environment. In addition, BASIC's file-handling and subprogram capabilities let you structure your applications many different ways, so you can pick the organization that best suits your needs.

IBM BASIC has become a powerful and versatile programming tool, yet has retained the simplicity and ease-of-use that earned the language its popularity. The rest of this book describes in greater detail these and other features that make IBM BASIC the choice for many different applications.

What's New With Release 2?

BASIC's even easier

Full-Screen Editing: You can now make changes to any program line that appears on your screen.

Program Function Key Support: PF Keys can be set to execute frequently used commands.

User Profile: You can specify commonly used function key settings and program options in a special PROFILE data set. The PROFILE is then processed automatically when you invoke BASIC.

BASIC's even more powerful

REAL Data: There are two additional numeric data types, REAL SINGLE and REAL DOUBLE. Mathematical computations performed using these data types execute significantly faster than those using BASIC's DECIMAL type. The use of REAL data also makes BASIC's interface to FORTRAN much more useful.

DECIMAL Data: The exponent range has been extended from ±75 to ±999, making DECIMAL data suitable for engineering and scientific users.

BASIC's even more versatile

FORTRAN, COBOL, and PL/I Interfaces: Numeric and character arrays can now be passed to subprograms written in these languages.

GDDM Interface: You can now pass GDDM function names as well as RCP codes. You can also invoke GDDM's Interactive Chart Utility (ICU) from BASIC.

Relational Data: You can now access data managed by IBM's relational data base systems (DB2 and SQL/DS) with the CALL SQL interface.

VM Tape Support: BASIC now supports multi-file tapes.

MVS/XA Support: MVS/XA support makes very large programs and data arrays possible. BASIC can reside above the 16 megabyte line, and use 31-bit addressing. BASIC can also be installed in the extended link pack area.

Editing, debugging, and HELP simplify program development.

Interactive Editing Facilities

Because the BASIC editing facilities include full-screen editing, a set of versatile editing commands, and line-by-line syntax checking, you don't need an outside editor.

Full-Screen Editing: You can use the full-screen editing facility to make changes to any BASIC program lines that appear on the screen. You can change more than one line at a time, and BASIC checks the syntax of all your changes and displays any line in which an error occurs when you press the enter key.

Editing Commands: In addition to full-screen editing, the BASIC editing commands let you make program corrections quickly and simply. Some of the most useful are:

AUTO	initializes automatic line numbering.
CHANGE	replaces character strings within a line or a block of lines.
FIND	locates strings within a line or a block of lines.
COPY	copies lines or blocks of lines.
EXTRACT	saves selected lines in a workspace and erases all others.
DELETE	erases selected lines from a workspace.

All the editing commands are listed under "Commands" on page 28.

Line-by-Line Syntax Checking: When you enter source lines, either from the terminal or from a file, they are scanned and checked, line by line. Syntax errors are immediately detected and reported.

Errors are reported by a message and a digit under the source line, along with the error number, so you can find further explanation by using HELP. If you type:

```
HELP
```

immediately after this message appears on your screen, BASIC will give you the explanation of the error number. If you don't use HELP immediately, however, you can still get information on an error message by specifying the message number. For example, to get further information on message BAS30122S, you can enter:

```
HELP BAS30122S
```

and BASIC displays HELP text explaining the message.

User input lines are typed in at the bottom of the screen. If SET TERM is set to SCROLL, as lines are displayed or new input lines are entered, previous lines are scrolled upward, with the topmost line scrolling off. When you hit the ENTER key, if SET TERM is set to LINE, input lines move to the top of the screen in the order in which they were entered and BASIC output lines are listed from the top of the screen. SET TERM SCROLL is only meaningful on display terminals; on other terminals, the command is ignored. SET TERM LINE improves performance for a remote terminal using BASIC.

Easy-to-Use Debugging Features

You can control the program execution with any of three debugging features: BASIC commands, debugging statements included in the BASIC language, or immediate execution of certain statements.

Debugging Commands: By using the BASIC commands during program execution, you can start and stop debugging actions, establish breakpoints at which execution can be suspended, begin and end program tracing, continue execution at a different line, and STEP through your program. These commands are listed under "Commands" on page 28.

Debugging Statements: With the debugging statements, you can establish breakpoints at which execution will be suspended for the inspection of data item contents, turn program tracing on and off, and activate and deactivate the execution of other debugging statements.

Immediate Statements: When you are in command mode inside the BASIC environment, you can execute selected statements immediately. Using these immediate statements, you can inspect, modify, and debug a program while execution is suspended at a breakpoint, or after the program has ended. You can also use immediate statements to start and stop debugging actions and program tracing.

HELP at the Terminal

The HELP command gives you interactive assistance in learning or using BASIC, and displays information about any BASIC command, statement, intrinsic function, message, or error code.

When you enter HELP BASIC, the topmost menu of the HELP tree is displayed. Each choice includes a *menu name* for that particular category. By entering a *menu name*, you can display a menu of *items* concerning the subject. You can then choose any *item* by entering its name.

There are three ways to use the HELP command:

1. As a diagnostic aid for explanation of BASIC messages
2. As a tutorial for assistance in learning BASIC
3. As a source for reference information about BASIC

The Diagnostic HELP: When you get a message, you can request more information about that message by entering HELP or HELP msg#. (See "Line-by-Line Syntax Checking:" on page 5.) BASIC responds with more information about that message.

The Tutorial HELP: The tutorial HELP provides explanations for such things as the BASIC environment (entering programs, debugging and running programs, immediate statements, and so forth) and programming in BASIC (program elements, data types, variables, expressions, and so forth). You can also enter any of the menu names described for tutorial HELP, and the menu for that category is displayed.

The Reference HELP: When you enter HELP followed by the name of any BASIC command, statement, or intrinsic function, reference information about that item is displayed at the terminal. For example:

```
HELP PRINT
```

retrieves information about the PRINT statement.

PF Key Support

To speed up the editing process, BASIC lets you set your PF keys to the BASIC commands you use most frequently. You can specify your PF key settings in your user profile, and they will be set automatically each time you invoke BASIC using that profile. (The user profile is also convenient for setting other defaults.)

Whether or not you use the user profile to set your PF keys, you can change them inside the BASIC environment by using the SET PF Key command.

You can also set PF keys in a program, by using the SKEY condition of the ON Condition statement. PF key settings specified in a program are in effect only while that program is running. When the program is finished, the PF key settings return to the session settings.

Comprehensive Publications

The IBM BASIC documentation (both the interactive HELP documentation and the publications) is designed to be understandable and to make information retrieval easy. *B is for BASIC* is a primer for people who have never used BASIC. The *IBM BASIC Programming Guide* is designed to provide task-oriented information for users who are still relatively new to BASIC. The *IBM BASIC Language Reference* contains an overview of the structure of the BASIC language, as well as an alphabetical reference of the statements, commands, and functions. In addition to these books, the HELP command gives you online documentation whenever it's needed. See "HELP at the Terminal."

The BASIC documentation is designed to help many different kinds of users with different levels of BASIC experience and different programming needs. A complete list of IBM BASIC manuals is given in Appendix A, "Publications" on page 34.

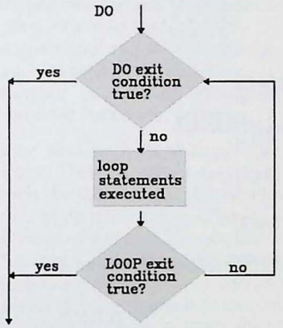
Powerful programming tools support complex applications.

Structured Programming Aids

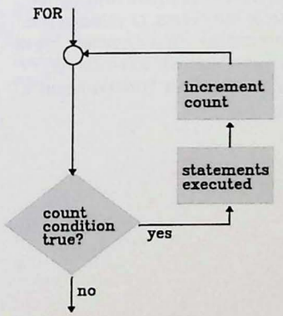
Each logical part of a structured program is a separate unit with one entry point and one exit point, and contains a separate processing construct. Each part can be developed and altered with no unforeseen consequences to other parts of the program. BASIC has several statements that make structured programs easy to design and develop: the DO/LOOP statements, the FOR/NEXT statements, the SELECT/CASE statements, and the block IF/ELSE statements.

These statements allow you to structure your programs so that coding is easy and control structures are understandable.

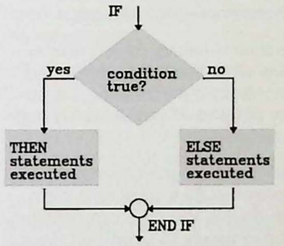
The DO/LOOP statements enable you to build condition-controlled loops. The WHILE and UNTIL clauses let you control the loops from the top of the loop, the bottom of the loop, or both.



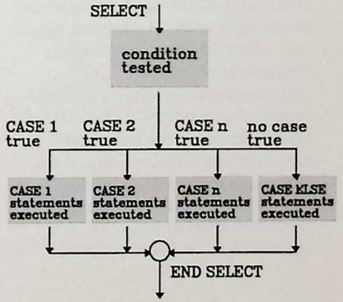
The FOR/NEXT statements let you build count-controlled loops. Such a loop is executed the number of times you specify in the FOR statement.



You can use the IF statement for two-choice conditional structures. The IF statement can include blocks of statements (including other IF blocks), for both the true and false conditions. Because the IF blocks can include many statements, you can keep your programming structures simple and easy to read.



The SELECT/CASE statements let you build conditional structures with more than two choices. A SELECT statement chooses one of a set of blocks, each identified by a CASE statement.



Versatile Data Options

Extensive Numeric Capabilities: BASIC lets you define your numeric variables as integer, decimal, or real. Decimal variables, because they allow exponents up to ±999, are useful for applications which require very large or very small numbers. If your program does a lot of number-crunching, real data will speed it up significantly. Real data is also useful for passing variables to BASIC's FORTRAN interface.

Integer variables can contain as many as 10 digits, with a maximum value of +2 147 483 647 and a minimum of -2 147 483 648.

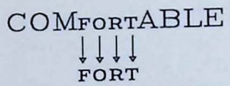
Decimal variables can contain as many as 17 significant digits, and can maintain mathematical accuracy of up to 17 places with a scale factor of ±999.

Real single variables can contain approximately 6 significant digits. Real double variables can contain approximately 15 significant digits. (The precise number of significant digits depends on the number itself.) Real single and real double both have a scale factor of -78 to +75.

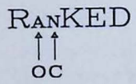
Using BASIC's arithmetic and relational operators, you can do extensive manipulation and comparison of numeric data. Arithmetic operators allow addition, subtraction, multiplication, division, and exponentiation in any valid mathematical combination. Relational operators allow the following numeric comparisons: equal to, not equal to, less than, greater than, not less than, not greater than, less than or equal to, and greater than or equal to.

Versatile Character String Handling: Variable-length character string handling is one of IBM BASIC's strongest features. It enables you to make references to substrings of character data items. You can, for example, move characters from one string to another, join strings together, and extract or delete characters from a string.

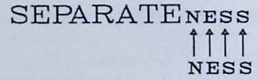
EXTRACT



REPLACE



ATTACH



By using the character intrinsic functions, you can retrieve the position of a specific character within a string, ascertain the current length of a string, replace lowercase characters with uppercase or vice versa, or retrieve the current date. (The complete list of character intrinsic functions is given in "Intrinsic Functions" on page 23.)

Comprehensive Array Capability: Arrays provide you with a convenient means of gathering similar data into tables. IBM BASIC allows arrays of up to 7 dimensions. The assignment and input/output statements have array (MAT) forms as well as the usual scalar forms, so arrays are as easy to manipulate as scalar variables.

For example, the following MAT assignment statement multiplies the arrays B and C, and assigns the resulting values to array A:

```
MAT A = B * C
```

The following MAT PRINT statement displays all the values of array A at the terminal:

```
MAT PRINT A
```

Except for specific matrix operations like inverse and multiply, these forms apply to arrays of up to 7 dimensions.

Any array may be redimensioned. You can change both the size and the number of dimensions, as long as the redimensioned array is the same size or smaller than the original.

Flexible Formatting Capabilities

You can use the IMAGE and FORM statements to format your input or output data. You can specify automatic data conversion, spacing, and character presentations to fit the application. Because the IMAGE and FORM statements allow you to control your output, you can use BASIC to generate reports and tables.

The IMAGE statement is a template for the format of your input or output data. The FORM statement provides conversion specifiers that allow the external representation of numeric data. For example, the following FORM statement:

```
100 FORM PIC ($$$$.##)
```

allows the use of a floating dollar sign in the printed output:

Printed	Data Result
123.456	$123.46
123.45	$123.45
12.34	$12.34
1.23	$1.23

The MARGIN statement lets you set top, bottom, and side margins of terminal or printed output, and the FONT specification of the PRINT statement lets you specify a particular font of the IBM 3800 Printing Subsystem.

Full-Screen Input/Output

You can control terminal input and output with the INPUT FIELDS and PRINT FIELDS statements. These statements allow the BASIC program to accept input from, and display output to, any part of the screen. This capability can be used, for example, for menu-driven data entry applications.

Extensive Intrinsic Functions

The many BASIC intrinsic (built-in) functions provide you with routines for commonly used functions that are otherwise tedious or difficult to program.

The intrinsic functions include numeric functions such as SQRT (square root), advanced mathematical functions such as COSH (hyperbolic cosine), character handling functions such as UPRC$ (uppercase), and other functions such as time, date, and error number. A complete list is given in "Intrinsic Functions" on page 23.

BASIC also has a set of array functions. These functions are used with the MAT statement, and allow you to do such things as find the inverse of a matrix (the MAT INV function), and transpose a matrix (the MAT TRN function), as well as set up an identity matrix (MAT IDN), or a constant matrix (MAT CON). The MAT functions simplify array calculations.

Program Controlled Exception Handling

Programs can take corrective action and continue processing after a mathematical or input/output exception. By using the ON condition statement, you can tell BASIC what to do if it encounters a particular type of exception. (See "Exception Handling Statements" on page 22 for a description of the ON condition statement.) Depending on what you specify, exceptions in your program can be either trapped for corrective action or ignored. Exception handling may be activated in the main program or in any subprogram.

The ERR intrinsic function returns the exception code of the last exception, and the LINE intrinsic function returns the line number of the statement that caused the exception. The CAUSE statement can simulate any exception, allowing easy checkout of exception handling and providing an extensive interrupt-handling capability.

BASIC provides versatile segmentation, compilation and execution.

Inside and Outside the BASIC Environment

You can compile and execute your programs from either inside or outside the BASIC environment. Whether you compile from inside the BASIC environment by using the COMPILE command, or outside the BASIC environment (see your IBM BASIC System Services manual for details), the object modules produced are identical.

To execute your program from inside the BASIC environment, you use the RUN command. This command first compiles the source program in your workspace, and then executes it. All the debugging features are available when you use the RUN command. This method of compiling and running your program is convenient during program development and checkout.

After you have completed and tested your program, it is more efficient to execute it as a precompiled object module. You can do this with the BASICRUN command outside the BASIC environment, and with the RUN command inside the BASIC environment.

Compilation produces machine language object code. Object modules are in IBM relocatable format, which means that they can be combined with others to create executable load modules.

Compilation is intended to be used when efficiency is the primary concern. A compilation produces output according to the options you specify, including a listing that may be directed to any appropriate output device.

Another way to compile programs from outside the BASIC environment is by using batch. If you do not have access to a terminal, you can use batch not only to compile your programs, but also to execute both object modules and load modules. Terminal input and output of the object modules are redirected to the default input and output files when you run in batch.

Program Segmentation

The program segmentation features give added program flexibility. You can break programs into separate parts in several ways:

External subprograms
Internal functions or subroutines
External program chaining

With external subprograms, using the CALL, SUB, SUBEXIT, and END SUB statements, you can split a large program into manageable program units. BASIC's subprogram facilities are also useful if you want to use prewritten routines or borrow routines from other users. You can link edit your subprograms together, or you can let BASIC do it for you. If you don't link edit, BASIC loads your subprograms dynamically at execution time. In this way, you can design BASIC object modules to save main or auxiliary storage. This concept is illustrated in Figure 1 on page 11.

The main program (MAINPRO) invokes a subprogram (SUBPRO1) for execution.

SUBPRO1 in turn invokes another subprogram (SUBPRO2) for execution.

When SUBPRO2 completes execution, control is returned to the next executable statement after the CALL SUBPRO2 statement in SUBPRO1.

If a specific condition is met, the SUBEXIT exit from SUBPRO1 is taken and control is returned to the next executable statement following the CALL SUBPRO1 statement in MAINPRO.

Otherwise, SUBPRO1 completes execution, and then control is returned to the next executable statement following the CALL SUBPRO1 statement in MAINPRO.

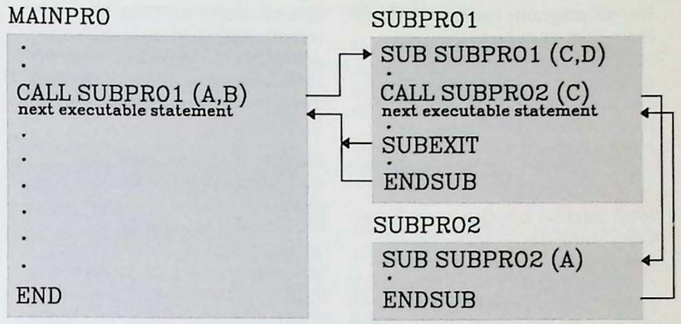


Figure 1. Calling and Called Programs

The first main program (MENUPRO), depending on the menu selection, chains either to PRO1 or to PRO2.

The new main program selected (PRO1 or PRO2) executes from beginning to end, and then chains back to the beginning of MENUPRO, where a new menu selection can be made.

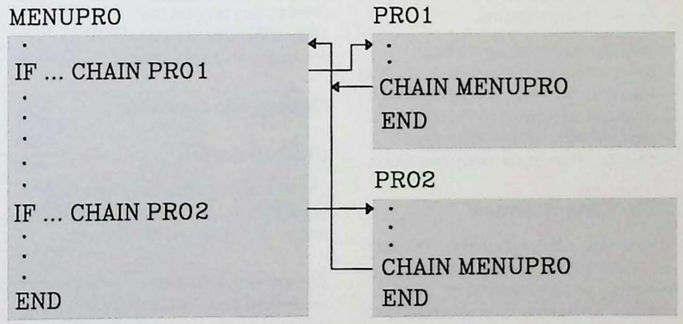


Figure 2. Program Chaining

The subprograms can be called more than once (with the CALL statement) from the calling program. A called program may call itself either directly or indirectly; recursion is allowed. For more information on the CALL statement, see "The CALL Statement."

With internal functions or subroutines, using the DEF/FNEND or GOSUB/RETURN statements, you can divide a program into its functional parts. Each internal function or subroutine can be a logical entity.

With CHAIN statements, you can reduce virtual storage requirements. By chaining programs, you can split a large program into several main programs, so that each main program (and its associated subprograms) can transfer control to the next at execution time and overlay the preceding one. This concept is illustrated in Figure 2 on page 11. Each of the main programs can, of course, call its own subprograms.

You can pass values through parameters with the CALL and CHAIN statements. Data can be shared by different programs in the COMMON area, which can survive either a CALL or CHAIN statement execution.

The CALL Statement

With the CALL statement, you can access not only subprograms written in BASIC and other computer languages, but also relational data and graphics facilities.

The CALL statement can be used to access subprograms written in COBOL, FORTRAN, and PL/I. BASIC's addition of REAL data makes the FORTRAN interface efficient because data no longer has to be converted. In addition, BASIC can now pass arrays to programs written in other languages, so it is particularly useful for engineering and scientific applications.

The CALL statement gives you access to graphics capabilities with an interface to the IBM Graphical Data Display Monitor (GDDM). GDDM is useful for generating a wide variety of business charts and graphs, including bar charts, pie charts, and line graphs. But GDDM's capabilities are not limited to charts and graphs; GDDM is sophisticated enough to generate diagrams and even illustrations, and it also makes a terrific toy.

BASIC can now access your relational database by using the CALL SQL statement. This statement provides an interface to IBM DATABASE 2 (DB2) and Structured Query Language/Data System (SQL/DS). By using CALL SQL, you can create new tables, and extract, delete, or add lines to existing tables. This capability makes BASIC a powerful tool for business applications.

Flexible File Handling

File Organization

There are four different kinds of file organization in IBM BASIC: stream, sequential, relative, and keyed.

Stream Organization: Records are organized serially and may be accessed only sequentially. Only one value is contained in a physical record, but you can treat the values as if there were no record boundaries. You might use this organization for a file in which there is no reason to group data in records. This file organization is the simplest and the easiest to use.

Sequential Organization: Records are organized serially and may be accessed only sequentially. More than one value can be contained in a record. Sequential file organization provides the ability to read records in the same sequence in which they were written. For example, you can use sequential organization for a transaction register to record all transactions completed in chronological order.

Relative Organization: Each record is assigned a record number and may be accessed sequentially or directly by record number. Records not yet written (so-called "null" records) are skipped during sequential access. You might use this organization for a "day of the week" file, where all the days are numbered consecutively, and this number can be used for record retrieval.

Keyed Organization: Each record is identified by a key, which is a character string within the record. Records can be accessed sequentially by ascending key sequence or directly through reference to the keys. You might use this organization for an employee file, in which each record contains a key (the employee number); records can then be located and processed directly by this key, or they can be located sequentially in ascending key sequence. Keyed organization can only be used with VSAM files.

File Type

There are three IBM BASIC file types:

Display Type: Each record is a sequence of characters. Carriage control characters for printers may be inserted on output. You can use this type for files to be printed or edited. You can format display files by using FORM or IMAGE statements with a PRINT statement.

Internal Type: Each record can be a sequence of numeric and/or character string values written in IBM BASIC's internal form. Each value is preceded by a designator giving the type of the value (integer, decimal, real, or character string). You might use this type for efficient file processing by BASIC programs without the need for formatting. Non-BASIC programs, however, would need to provide special processing in order to read files of this type.

Native Type: Each record is formatted by BASIC with FORM statements and can contain both character and numeric data, as well as data in various internal forms. Native type files provide the facility to specify the format of each value in a record, and you can convert values to a variety of internal and external representations. Native type can be used with sequential, relative, or keyed files.

File Access

In BASIC, the type of access determines which input/output operations are allowed on the file. The file access methods are:

Input: Only read operations are allowed.

Output: Only write operations are allowed.

Outin: Both read and write operations are allowed. Sequential files can be extended. Records can be updated in place.

The allowable combinations of file type, organization, access, and record type are shown in Figure 3.

File Type	Organization	Access	Record Type
DISPLAY	SEQUENTIAL	INPUT OUTPUT OUTIN	FIXED VARIABLE
INTERNAL	STREAM	INPUT OUTPUT	FIXED VARIABLE
INTERNAL	SEQUENTIAL	INPUT OUTPUT OUTIN	FIXED VARIABLE
NATIVE	SEQUENTIAL	INPUT OUTPUT OUTIN	FIXED VARIABLE
NATIVE	RELATIVE	INPUT OUTPUT OUTIN	FIXED
NATIVE	KEYED	INPUT OUTPUT OUTIN	FIXED VARIABLE

Figure 3. Valid File Combinations

System Access

Insofar as the access method allows, the program itself is independent of the system access operation. BASIC stream, sequential, and relative files can all be BASIC non-VSAM files. You can create and delete such files from within or without the BASIC program. Keyed files, however, must be VSAM files. You must define VSAM files before you use them in a BASIC program.

Reentrant Processor and Library

The IBM BASIC program product is made up of a Processor and Library. In order to invoke the BASIC environment, or compile programs either inside or outside the BASIC environment, you need both the Processor and Library. The Library supports input/output operations, some internal routines, and the wide variety of intrinsic functions. The Processor supports the BASIC environment, and compilation, both inside and outside the BASIC environment.

If you plan only to execute precompiled programs from outside the BASIC environment, you don't need the Processor, and can save space by installing only the Library.

Both the Processor and the Library are reentrant. Therefore, you can place the Processor and any or all Library modules (except for one "root" module in IBM BASIC/VM), in shared virtual storage, and the same copy can be used by all the BASIC programs in the system.

When the Processor and the selected Library modules are not placed in shared virtual storage, each user has a separate copy.

- Two classes of data are available: numeric (integer, decimal, real single, and real double) and character.
- Identifiers name variable data and other program elements.
- Line labels can be used in place of line numbers for reference in branching statements like GOTO and ON Condition.
- Multistatement lines and multiline statements are permitted.
- The following expressions are allowed: scalar, array, relational, and logical.
- The operators allowed in expressions are shown in the table in the next column.

Operator	Meaning
^ or ¬ or **	exponentiation (note that ^ and ¬ are the same hexadecimal code)
*	multiplication
/	division
+	addition
-	subtraction
&	concatenation (for character expressions)
= or EQ	equal
< > or > < or NE	not equal
> = or = > or GE	greater than or equal
< = or = < or LE	less than or equal
> or GT	greater than
< or LT	less than
NOT	logical negation
AND	logical conjunction
OR	logical inclusion

The advanced statements highlighted in blue make BASIC suitable for more sophisticated applications.

Declarative Statements

These statements define characteristics of program elements.

COMMON	Specifies variables to be shared by all program units.
DECIMAL	Declares one or more variables to contain decimal values.
DEFDBL	Declares one or more variables as either real double or decimal.
DEFINT	Declares one or more variables to contain integer values.
DEFSNG	Declares one or more variables as either real single or decimal.
DIM	Explicitly defines the number and extents of array dimensions and the length of character data.
INTEGER	Declares one or more variables to contain integer values.
OPTION	Sets BASIC options.
REAL	Declares one or more variables to contain real single or real double values.
REM or !	Indicates comments.

Assignment Statements

These statements assign values to variables.

LET	Assigns a value to a variable or to a substring.
MAT	In an assignment statement, assigns values to an array.
RANDOMIZE	Generates a new starting point for a series of pseudo-random numbers.

Control Statements

These statements direct the flow of execution in a program.

GOTO	Transfers control unconditionally to a specific statement.
ON ... GOTO	Transfers control conditionally to one of a group of statements.
DO	Begins a group of statements that are repetitively executed until an exit condition is satisfied.
LOOP	Ends a DO group.
FOR	Begins a group of statements that are repetitively executed until an iteration criterion is satisfied.
NEXT	Ends a FOR group.
EXIT IF	Conditionally transfers control from within a DO or FOR loop to the next statement following a DO or FOR loop.
IF	Conditionally executes an imperative statement, or branches, depending on the true or false value of a logical expression. IF is also the initial delimiter of a group of conditionally executed statements called an IF block.
ELSE	Begins the alternative part of an IF THEN ELSE statement or an IF block.
END IF	Ends an IF block.
SELECT	Begins a group of selectively executed statements known as a SELECT block. The SELECT statement includes a multichoice condition to be tested by each CASE statement.
CASE	A selection statement in a SELECT block. The statements following the CASE statement are executed if the CASE condition satisfies the SELECT condition.
CASE ELSE	An optional statement in a SELECT block. The statements following CASE ELSE are executed if no CASE condition satisfies the SELECT condition.
END SELECT	Ends a SELECT block.
PAUSE	Causes suspension of program execution until resumed by the user (ignored in batch).
STOP	Causes the end of a program execution.
END	Indicates the end of a main program.

Line-by-Line Terminal I/O Statements

These statements perform line-by-line terminal input/output (default files are used in batch).

INPUT	Accepts data from a terminal and assigns these values to specified variables.
LINE INPUT	Accepts unformatted data from a terminal and assigns these values to a character item in an I/O data list. LINPUT is the alternative spelling.
PRINT	Displays formatted or unformatted data from a program to a terminal.
MARGIN	Specifies where displayed output begins and ends on the terminal screen, and where side margins are set.

Full-Screen Terminal I/O Statements

These statements perform full-screen terminal input/output (may not be used in batch or on non-327x terminals).

INPUT FIELDS	Reads specific fields from a display terminal.
PRINT FIELDS	Writes specific fields to a display terminal.

File I/O Statements

These statements perform input/output on external files.

File Control Statements

CLOSE	Deactivates a file.
OPEN	Activates a file and specifies conditions.
RESET	Positions a file to a specified record or to the beginning or end. RESTORE is the alternative spelling.
SCRATCH	Erases the contents of a file.

Internal Stream File Statements

GET	Retrieves a stream of data items from a file into a program and assigns them to variables and arrays in an I/O data list. INPUT is the alternative spelling.
PUT	Conveys a stream of values from an I/O data list to a file. WRITE is the alternative spelling.

Display Sequential File Statements

INPUT	Retrieves a data record from a file and assigns selected fields to arrays or variables in an input data list.
LINE INPUT	Retrieves a record from a file, and assigns that record to a character variable or to a member of a character array. LINPUT is the alternative spelling.
MARGIN	Specifies where printed output may begin and end on a page or terminal screen, and where side margins are to be set.

	PRINT	Formats values from items in an output data list in the program into selected fields of a data record and transmits the formatted data to a file, which can be printed.
	Internal Sequential File Statements	
	READ	Retrieves a self-typed data record from a file and assigns self-typed data to arrays or variables in an input data list. INPUT is the alternative spelling.
	WRITE	Conveys values from items in an I/O data list in the program into self-typed data items in a record and adds the record to a file.
	Native Sequential, Relative, and Keyed File Statements	
	These statements (except DELETE) require the use of a FORM specification.	
	DELETE	Deletes a data record from a file (relative and keyed only).
	READ	Retrieves a data record from a file and assigns the selected formatted fields to arrays or variables in an input data list.
	WRITE	Formats values from items in an I/O data list in the program into defined fields of a data record and adds the record to a file.
	REREAD	Rereads a record from a file and assigns the selected formatted fields to arrays or variables in an input list.
	REWRITE	Formats values from items in an I/O data list in the program into selected fields of a data record and rewrites the record to a file.
Auxiliary I/O Statements	EXIT	Specifies the statement to transfer to in the event of an error during I/O.
	IMAGE	Specifies the format of a print line when used with a PRINT USING statement.
	FORM	Specifies the conversion of data for both input and output when used with PRINT USING, READ USING, REREAD USING, WRITE USING, or REWRITE USING.
Internal I/O Statements	DATA	Specifies data in an internal data file (contained in the program itself).
	READ	Assigns the values in an internal data file to variables or arrays.
	RESTORE	Positions an internal data file to its beginning or a specified DATA statement.

Program Segmentation Statements

IBM BASIC provides several methods for the segmentation of programs: user-defined functions (which can be more than one line) invoked by reference, internal subroutines invoked by a GOSUB statement, external subprograms with parameter lists invoked via a CALL statement, and chained execution of programs.

DEF	Indicates the beginning of a user function.
FNEND	Indicates the end of a user function.
SUB	Identifies the beginning of an external subprogram.
SUBEXIT	Terminates the execution of an external subprogram.
END SUB	Indicates the textual end and terminates the execution of an external subprogram.
CALL	Transfers control to an external subprogram, optionally passing arguments.
GOSUB	Transfers control unconditionally to an internal subroutine.
ON...GOSUB	Transfers control conditionally to an internal subroutine.
RETURN	Returns control to the first executable statement following the last active GOSUB or ON ... GOSUB statement.
CHAIN	Transfers control to a new main program, optionally passing arguments. COMMON variables are also available in the new environment.
USE	Defines the names of data items to be passed to a chained program.

Debugging Statements

Debugging statements may be included in BASIC programs to assist interactive program corrections (these statements are ignored in compiled programs).

BREAK	Specifies a breakpoint at which program execution is to be suspended.
DEBUG	Activates or deactivates the other debugging statements.
TRACE	Activates or deactivates program tracing.

Exception Handling Statements

Exception handling provides a means of regaining control within a program after an exception has occurred.

ON CONDITION Determines the action to be taken if an exception occurs.

Possible Conditions:

ATTN	Associated with the attention signal from a terminal
CONV	Error in conversion of a numeric data item
ENDPAGE or PAGEOFLOW	End of page for a print file
OFLOW	Numeric overflow
SKEY	Associated with the PF keys on a terminal
SOFLOW	String overflow
UFLOW	Numeric underflow
ZDIV	Division by zero
ERROR	Those conditions not covered by any of the above

CAUSE	Causes a specified exception to occur; useful for testing error routines.
CONTINUE	Resumes execution after the statement that caused the exception.
RETRY	Resumes execution after an exception by retrying the statement that caused the exception.

Immediate Statements

Inside the BASIC environment, some BASIC statements may be executed immediately by entering them without a line number. Immediate statements allow the user to inspect and modify the values of program variables.

The following statements may be executed immediately:

DEBUG, DECIMAL, DIM, INTEGER, LET, MAT, OPTION,
PRINT, RANDOMIZE, REAL, REM, STOP, TRACE

The full range of intrinsic functions is available with the immediate statements.

Intrinsic Functions

The advanced functions highlighted in blue make BASIC suitable for more sophisticated applications.

Numeric Functions

These include numeric conversion functions and functions performing common mathematical operations.

ABS(X)	Absolute value of X
CEIL(X)	Smallest integer not less than X
CEN(X)	Degrees Centigrade (Celsius) corresponding to X degrees Fahrenheit
DBL(X)	X converted to real double
DEC(X)	X converted to decimal
DET	Determinant of the last array inverted
DET(A)	Determinant of array A
DOT(A,B)	Dot product of two vectors
EPS	Smallest positive number representable
FAH(X)	Degrees Fahrenheit corresponding to X degrees Centigrade (Celsius)
FP(X)	Fractional part of a decimal number
IFIX(X)	Rounded integer value
INF	Largest positive number representable
INT(X)	Largest integer not greater than X
IP(X)	Integer part of X
MAX(X,Y[,...])	Larger (algebraically) of its arguments
MIN(X,Y[,...])	Smaller (algebraically) of its arguments
MOD(X,Y)	X modulo Y

PI	The constant 3.1415926535897932 (the actual value depends on the options in effect)
PRD(A)	Product of the elements of the array A
REAL(X)	X converted to real (the precision depends on whether LPREC or SPREC is in effect)
REM(X,Y)	X-Y*IP(X/Y) if Y is nonzero, and X if Y is zero
RND	Next pseudo-random number in sequence of pseudo-random numbers
RND(X)	Assigns the value of X to the seed value for pseudo-random number generator.
ROUND(X,N)	X rounded to N decimal digits from the decimal point
SGN(X)	Sign of X: -1 if X < 0, 0 if X = 0, and +1 if X > 0
SIZE(A)	Total number of elements in the array A
SIZE(A,M)	Number of elements in the Mth dimension of the array A
SNG(X)	X converted to real single
SQR(X)	Square root of X
SUM(A)	Sum of the elements of the array A
TRUNCATE(X,N)	X truncated to N decimal digits from the decimal point
UDIM(A,M)	Upper bound of the Mth dimension of array A

Advanced Mathematical Functions

These include exponentiation, logarithmic, trigonometric, and hyperbolic functions.

ACOS(X)	Arccosine of X in radians
ANGLE(X,Y)	Angle in radians between the positive X-axis and the vector joining the origin to the point with coordinates (X,Y)
ASIN(X)	Arcsine of X in radians
ATN(X)	Arctangent of X in radians
COS(X)	Cosine of X, where X is in radians
COSH(X)	Hyperbolic cosine of X
COT(X)	Cotangent of X, where X is in radians
CSC(X)	Cosecant of X, where X is in radians
DEG(X)	Converts X radians to degrees
EXP(X)	The constant e raised to the power of X
LOG(X)	Natural logarithm of X
LOG10(X)	Common logarithm of X
LOG2(X)	Base 2 logarithm of X
RAD(X)	Converts X degrees to radians
SEC(X)	Secant of X, where X is in radians
SIN(X)	Sine of X, where X is in radians
SINH(X)	Hyperbolic sine of X
TAN(X)	Tangent of X, where X is in radians
TANH(X)	Hyperbolic tangent of X

Character Functions

These functions operate on or return character strings.

CHR$(M)	Character occupying ordinal position M in current collating sequence
DAT$[(M)]	Character string of the form yyyy/mm/dd corresponding to the Julian date M
DATE$	Character expression corresponding to the current date in the form 'YY/MM/DD'
FILE$(M)	Character expression representing the file name associated with file #M
JDY[(C$)]	Julian date for the corresponding Gregorian date in C$, where C$ is expressed as 'yyyy/mm/dd'
LEN(A$)	Number of characters in A$
LPAD$(A$,M)	String of M characters produced by concatenating M-LEN(A$) spaces to the front of the value of A$
LTRM$(A$)	String A$ with all leading space characters removed
LWRC$(A$)	String of characters resulting from A$ by replacing each uppercase letter in A$ with its lowercase equivalent
ORD(A$)	Ordinal position, in the current collating sequence, of the character named by A$
POS(A$,B$)	Character position within A$, of the first character of the first occurrence of B$
POS(A$,B$,M)	Same as POS but beginning with Mth character
RPAD$(A$,M)	String of M characters produced by concatenating M-LEN(A$) spaces to the end of A$
RPT$(A$,M)	The value of A$ is repeated M times
RTRM$(A$)	String A$ with all trailing space characters removed
SREP$(A$,M,B$,C$)	Searches a character string and replaces one embedded group of characters with another
STR$(X)	String generated by the PRINT statement as the numeric representation of X
TIME$	Character expression corresponding to the time of day in 24-hour notation in the form 'HH:MM:SS'
UPRC$(A$)	String of characters resulting from replacing each lowercase letter in A$ with its uppercase equivalent
VAL(A$)	Numeric value of the character representation of a numeric constant

Other Functions

These include functions that communicate with the host operating system.

CNT	Number of data items successfully processed by the last I/O statement executed
CODE	Value returned from the host system when that system detected an error
DATE	Current date in the form YYDDD
ERR	Exception code when an exception occurs during the execution of a BASIC program
FILE(M)	Returns certain conditions regarding the status of file #M
FILENUM	File number (0 to 255) of the file in which an error occurred
KEYNUM	Number of the PF key that caused an SKEY exception
KLN(M)	Length in bytes of the embedded key for the file #M
KPS(M)	Byte position for the start of the embedded key for the file #M
LINE	Line number of the statement whose execution caused the exception
PARM$	Character string passed as an argument in the invocation of a BASIC program
REC(M)	Record number of the last record processed in file #M
RLN(M)	Length of the last record referenced for file #M
SRCH (A,X)	Subscript of array A element whose value is X
TIME	Elapsed time in seconds since the previous midnight

The advanced commands highlighted in blue make BASIC suitable for more sophisticated applications.

Editing Commands

Inside the BASIC environment, commands allow convenient program editing and debugging, and give easy access to system functions. All the commands can be abbreviated. The minimum abbreviation is two characters, but some command abbreviations must be longer in order to ensure that they are unique.

AUTO	Directs BASIC to automatically prompt with line numbers for statements entered from the terminal.
CHANGE	Performs character string replacement within a line or a block of lines.
COPY	Copies blocks of lines from one point to another within the user's workspace.
DELETE	Removes specified lines from a program.
EXTRACT	Removes all except specified lines from a program (inverse of delete).
FIND	Locates strings within a line or a block of lines.
INITIALIZE	Reinitializes (clears) the workspace.
LIST	Displays statements in the current program.
QUIT	Terminates a BASIC session.
RENAME	Changes the name of the workspace.
RENUMBER	Renumbers the lines in the current program.

Library Handling Commands

FETCH	Copies a source program with its associated internal text from the library to the workspace.
LOAD	Copies a source program from the library to the workspace and performs a syntax check of the program.
MERGE	Merges one or more portions of one program into the current workspace.
PURGE	Deletes a program or selected files from the library.
QUERY	Displays the contents of the user's library.
SAVE	Saves the current source program.
STORE	Saves the current source program with its associated internal text.

Program Execution and Debug Commands

BREAK	Sets, removes, and displays debugging breakpoints.
COMPILE	Causes the compilation of a source program.
DROP	Reinitializes values of variables in the workspace.
GO/GOTO	Continues execution after a breakpoint.
HELP	Displays instructive descriptions of BASIC commands, statements, and intrinsic functions, and additional diagnostic information.
RUN	Initiates execution of a program.
SYSTEM	Passes a command to the operating system for execution, or causes the subsystem environment (CMS) to be entered. (Requires TSO/E Release 2 under MVS.)

Interactive Session Configuration Commands

PROFILE	Executes BASIC commands and immediate statements contained in a file.
SET LOG	Starts or ends a log of the BASIC interactive session.
SET MSG	Controls the level and content of messages displayed at the terminal.
SET OPTION	Defines the default values for options.
SET PF	Establishes PF key definitions for command input.
SET PROFILE	Enables you to display your profile when it is invoked.
SET TERM	Controls whether display terminal I/O is in scrolling mode or line mode.

Compiler Options

Compiler options let you specify compilation options that differ from those established during product installation. You specify these options with the COMPILE command inside the BASIC environment, and with the BASICRUN command outside the BASIC environment.

The following compiler options are available. The default options supplied with the product are underscored. Your organization can change these defaults during product installation or later on.

ARITHMETIC DECIMAL |NATIVE
: Indicates whether the default for variables that are not typed is DECIMAL or REAL.

FIPS| NOFIPS
: Indicates whether or not a diagnostic warning for any statement not conforming to the Federal Information Processing Standard for BASIC syntax is to be produced.

FLAG (I |W|E|S)
: Indicates the level of diagnostic messages to be written.

  FLAG (I)
  : Information messages, warning messages, error messages, and severe error messages are to be written.

  FLAG (W)
  : Warning messages, error messages, and severe error messages are to be written.

  FLAG (E)
  : Only error messages and severe error messages are to be written.

  FLAG (S)
  : Only severe error messages are to be written.

LIST| NOLIST
: Indicates whether or not a listing of the generated machine instructions, consisting of assembler mnemonics and symbols is to be produced.

MAP| NOMAP
: Indicates whether or not an allocation map listing all the variables and subprograms used in the program is to be produced.

OBJECT |NOOBJECT
: Indicates whether or not the object module is to be written.

SOURCE |NOSOURCE
: Indicates whether the BASIC source program listing is to written.

XREF| NOXREF
: Indicates whether or not a cross-reference for variables, referenced statement numbers, and statement labels is to be produced.

SPREC |LPREC
: Indicates the maximum number of significant digits to be displayed by an unformatted PRINT statement:
  SPREC specifies up to 6 digits.
  LPREC specifies up to 12 digits.
  The SPREC|LPREC option also determines the default precision of REAL data (except for data whose precision is explicitly specified by a REAL SINGLE, REAL DOUBLE, DEFSNG, or DEFDBL statement). SPREC specifies that REAL data is single precision, and LPREC specifies that it is double precision.

Note: Compiler option defaults can be overridden for a BASIC session by the SET OPTION command.

OPTION Statement

The OPTION statement permits you to choose any of the following options to apply to your program (the default options are underscored):

ARITHMETIC DECIMAL |NATIVE
: This option is the same as that described under "Compiler Options" on page 30.

BASE 0|1
: Specifies the first subscript of a dimension in an array to be zero or one.

COLLATE NATIVE |STANDARD
: If NATIVE, the collating sequence is EBCDIC; if STANDARD, it is ASCII.

FIPS| NOFIPS
: This option is the same as that described under "Compiler Options" on page 30.

FLAG (I |W|E|S)
: This option is the same as that described under "Compiler Options" on page 30.

INVP [ON|OFF]
: INVP (inverted print edit facility) interchanges the usage of the decimal point and the comma to print numbers in European format.

PRTZO nn|20
: Permits the user to specify the width (nn) of print zones.

RD nn
: Permits the user to specify the number of digits (nn) at which rounding should occur when the result of an unformatted PRINT statement is displayed.

SPREC |LPREC
: This option is the same as that described under "Compiler Options" on page 30.

Installation Options

Many of the default values for IBM BASIC options can be changed to suit your needs. These options are usually changed when the product is installed, but can be changed at any time. These options apply to the entire installation. In addition to the compiler options (listed under "Compiler Options" on page 30) and those in the OPTION statement (listed under "OPTION Statement" on page 31), the options that you can change are:

Default user profile for the PROFILE option of the IBM BASIC invocation

Whether processing speed (the SPEED option) or efficient storage utilization (the SPACE option) is the primary concern

Whether terminal input/output defaults to SCROLL or LINE mode

Default starting line number and line number increment values for the AUTO and RENUMBER commands

Default line number increment value for the COPY command

Default maximum string length

INVP (OFF is U.S. style printing of numerics, for example 1,234.56; ON is European style printing of numerics, for example 1.234,56)

Default bottom margin for the MARGIN File statement

Default record type for files (can be variable or fixed)

Default record size for a display, internal, or native file

Message code suppression for all four message levels (I, W, E, S), separately

Default line and page length for LISTING files

Note: Installation option defaults can be overridden for a BASIC session by the SET OPTION command.

Appendix A. Publications

This appendix lists the IBM BASIC publications.

Planning Publications

IBM BASIC General Information, GC26-4023, provides a general description of the IBM BASIC program products.

IBM BASIC/VM Licensed Program Specifications, GC26-4024, describes the IBM BASIC/VM program product and serves as the basis for warranty.

IBM BASIC/MVS Licensed Program Specifications, GC26-4110, describes the IBM BASIC/MVS program product and serves as the basis for warranty.

Installation and Customization Publications

IBM BASIC/VM Installation and Customization, SC26-4025, describes how to install and to alter IBM BASIC/VM for the needs of individual installations.

IBM BASIC/MVS Installation and Customization, SC26-4105, describes how to install and to alter IBM BASIC/MVS for the needs of individual installations.

Programming Publications

B is for BASIC, GC26-4102, is a tutorial for the first-time BASIC user.

IBM BASIC Programming Guide, SC26-4027, gives guidance on designing, coding, executing, and debugging IBM BASIC programs, and on using the BASIC commands and immediate statements in all these tasks.

IBM BASIC Language Reference, GC26-4026, gives the rules for using IBM BASIC source statements, immediate statements, and commands.

IBM BASIC/VM System Services, SC26-4028, provides system-dependent guide and reference information on running IBM BASIC programs under VM/SP CMS.

IBM BASIC/MVS System Services, SC26-4106, provides system-dependent guide and reference information on running IBM BASIC programs under MVS and TSO.

IBM BASIC Language Reference Summary, SX26-3736, is a convenient pocket-sized quick-reference card that summarizes IBM BASIC language elements.

Diagnosis Publications

IBM BASIC/VM Diagnosis Guide, LY26-3879, is used when it is suspected that IBM BASIC/VM is not operating correctly. It gives guidance on developing a *keyword string* that describes the problem. The keyword string is then used to search a data base for similar known problems and to request assistance from an IBM Support Center.

IBM BASIC/MVS Diagnosis Guide, LY26-3885, is used when it is suspected that IBM BASIC/MVS is not operating correctly. It gives guidance on developing a *keyword string* that describes the problem. The keyword string is then used to search a data base for similar known problems and to request assistance from an IBM Support Center.

Appendix B. IBM BASIC Requirements

IBM BASIC program and machine requirements are outlined in the following sections.

Programming Requirements

The IBM BASIC Processor and Library run under:

- Virtual Machine/System Product Conversational Monitor System (VM/SP CMS) Release 3 or 4.
- Virtual Machine/System Product High Performance Option (VM/SP HPO) Release 3.
- IBM Virtual Machine/Personal Computer (VM/PC) Release 1.1.
- MVS/System Product (MVS/SP) Version 1 Release 3 (with or without the Time Sharing Option (TSO) or the Time Sharing Option/Extended (TSO/E)). TSO/E Release 2 is required for the SYSTEM command and the CALL SYSTEM statement.
- MVS/Extended Architecture (MVS/XA), which includes the following Program Products:
  - MVS/System Product (MVS/SP) Version 2 Release 1 (with or without the Time Sharing Option (TSO) or the Time Sharing Option/Extended (TSO/E)). TSO/E Release 2 is required for the SYSTEM command and the CALL SYSTEM statement.
  - MVS/Extended Architecture (MVS/XA) Data Facility Product Release 1.

For installation under VM/SP CMS, an EXEC procedure is provided as part of IBM BASIC.

For installation under MVS/SP, the functions of either the System Modification Program Release 4, or the System Modification Program/Extended are required.

If VSAM files are processed with IBM BASIC/VM, VSE/VSAM, Program Number 5746-AM2, must be installed.

Under VM/SP CMS and MVS/SP, IBM BASIC programs and subprograms can call subprograms written in the following languages and compiled with the following program products:

- IBM OS/VS COBOL Compiler and Library, Program Number 5740-CB1 (The COBOL Release 2.4 Library must be available at runtime.)
- IBM VS FORTRAN Compiler and Library, Program Number 5748-FO3 (The FORTRAN Release 4 Library must be available at runtime.)
- IBM OS PL/I Optimizing Compiler, Resident Library, and Transient Library, Program Number 5734-PL3 (The PL/I Release 5.1 Library must be available at runtime.)

Under VM/SP CMS and MVS/SP, IBM BASIC programs and subprograms can call GDDM functions if the IBM Graphical Data Display Manager (GDDM), Release 3 or 4, Program Number 5748-XXH, with or without the optional Presentation Graphics Feature (PGF), is installed.

Under VM/SP CMS, IBM BASIC programs and subprograms can access relational data bases using a subset of the SQL language if Structured Query Language/Data Systems (SQL/DS), Program Number 5748-XXJ, Release 3, is installed.

Under MVS/SP, IBM BASIC programs and subprograms can access relational data bases using a subset of the SQL language if IBM DATABASE 2 (DB2), Program Number 5740-XYR, is installed.

Machine Requirements

IBM BASIC runs in any processing unit supported by the host operating system.

IBM BASIC devices are the same as those allowed by the host operating system, provided that they are transparent to the access methods used.

The virtual storage required for the Processor and Library depends upon whether the SPEED or the SPACE option was chosen at installation time:

If the SPEED option was chosen for fast processor performance, approximately 750K bytes are required for IBM BASIC/VM, and approximately 760K bytes are required for IBM BASIC/MVS.

If the SPACE option was chosen to optimize virtual storage usage, approximately 525K bytes are required for IBM BASIC/VM, and approximately 535K bytes are required for IBM BASIC/MVS.

Space required by the user's BASIC program and its data is in addition to the sizes above.

Because of storage requirements for the host system, the virtual storage required is larger than the above IBM BASIC sizes.

The direct-access storage space required for IBM BASIC residence is approximately 5300 blocks of 1024 bytes each for either IBM BASIC/VM or IBM BASIC/MVS.

Appendix C. Migration from Other BASIC Products

Considerations for migration from other IBM BASIC products to IBM BASIC are given in this chapter.

Industry Standards

IBM BASIC is in conformance with and exceeds the following standards:

- American National Standard for Minimal BASIC, ANSI X3.60-1978
- European Computer Manufacturers Association Standard ECMA-55 Minimal BASIC, January 1978
- Federal Information Processing Standard as documented in FIPS PUB 68

VS BASIC Migration

VS BASIC has a limited range of functions, and many more restrictions than IBM BASIC. Because IBM BASIC expands on many VS BASIC functions, migration from VS BASIC is reasonably simple. Most VS BASIC programs will run on IBM BASIC after syntactic and other simple changes are made. Migration includes considerations of language, intrinsic functions, file structures, and arithmetic, as explained in the following paragraphs.

Language: Some of the same statements in both IBM BASIC and VS BASIC have different clauses and options. The syntax of these statements must be changed to conform to IBM BASIC. Some of these are: CHAIN, CLOSE, DELETE, DIM (for character variables), END, FORM, FOR/NEXT, GET, GOSUB (computed), GOTO (computed), ON, OPEN, PRINT USING, PUT, READ (FILE), REM, RESET, RETURN, REWRITE (FILE), STOP, WRITE

The INPUT FROM statement is replaced by the INPUT File statement, and the PRINT TO statement is replaced by the PRINT File statement.

If logical, relational, or concatenation operators are used, they must be changed to conform to IBM BASIC usage.

The default for the lower bound of a subscript in IBM BASIC is 0; in VS BASIC, it is 1. (In an IBM BASIC program, OPTION BASE 1 gives the same effect as in VS BASIC.)

IBM BASIC uses variable-length strings; VS BASIC uses fixed-length strings.

Intrinsic Functions: Some of the names of the intrinsic functions are different, although they perform the same functions.

File Structures: VSAM files created by IBM BASIC programs can be processed by VS BASIC programs, and vice versa. IBM BASIC and VS BASIC non-VSAM files are not usually compatible.

Arithmetic: The magnitude and precision of numeric data differ between IBM BASIC and VS BASIC. Increased accuracy of IBM BASIC may result in slightly different answers to mathematical calculations.

IBM BASIC rounding and truncation rules differ from VS BASIC rules.

Not all of the VS BASIC predefined constants exist in IBM BASIC.

VS BASIC Data Set Migration

IBM BASIC programs can read both stream-oriented and record-oriented VS BASIC files. IBM BASIC can handle VS BASIC stream-oriented files as IBM BASIC native format files, using FORM specifications.

With some restrictions, IBM BASIC programs can read VS BASIC record-oriented files either as IBM BASIC display format files or as IBM BASIC native format files.

System/23, 34, 36, and 38 BASIC Migration

The System/23, System/34, System/36, and System/38 BASIC languages are substantially compatible with the IBM BASIC language. However, IBM BASIC includes many language features not available in System/23, System/34, System/36, or System/38 BASIC.

PC BASIC Migration

Although both PC BASIC (the BASIC interpreter provided with the IBM Personal Computer) and IBM BASIC support the ANS Standard for Minimal BASIC, there are many syntactic, semantic, and environmental differences. These differences are particularly important in the I/O areas like file handling, screen formatting, and printing.

IBM BASIC supports many features, including indexed file support, structured programming constructs, interlanguage communication, and support for 3800 print fonts, that are not available in PC BASIC. Likewise, PC BASIC contains a number of features unique to the personal computer hardware, such as music and sound capability, communication and joystick support, and PEEKs and POKEs to absolute memory addresses. Because these differences are fairly extensive, most PC BASIC programs will require modification in order to run under IBM BASIC.

Index

Q

R

S

T

U

V

W

X

Z

IBM BASIC
General Information
GC26-4023-4

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Note: Staples can cause problems with automated mail sorting equipment. Please use pressure sensitive or other gummed tape to seal this form.

List TNLs here:

If you have applied any technical newsletters (TNLs) to this book, please list them here:

Last TNL _____

Previous TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.) Thank you for your cooperation.

GC26-4023-4

Reader's Comment Form

Fold and tape | Please do not staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

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

Fold and tape | Please do not staple | Fold and tape

IBM

IBM BASIC General Information (File No. S370-20) Printed in U.S.A. GC26-4023-4

IBM BASIC
General Information

Order Number:
GC26-4023-4

File No. S370-20

Printed in U.S.A.

GC26-4023-04