GC28-8302-5

Program Product

VS BASIC
General Information

GC28-8302-5

Program Product

VS BASIC
General Information

Program Number 5748-XX1

IBM

Fourth Edition (October 1976)

This edition, as amended by technical newsletter GN26-0902, applies to Release 3 of VS BASIC, program number 5748-XX1, and to any subsequent releases unless otherwise indicated in new editions or technical newsletters. Release 3 runs under the same operating-system environments that support the current version and modification level.

The changes for this edition are summarized under "Summary of Amendments" following the list of figures. Technical changes made are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Information in this publication is subject to significant change. Any such changes will be published in new editions or technical newsletters. Before using the publication, consult the latest *IBM System/370 Bibliography,* GC20-0001, and the technical newsletters that amend the bibliography, to learn which editions and technical newsletters are applicable and current.

Requests for copies of IBM publications should be made to the IBM branch office that serves you.

Forms for readers' comments are provided at the back of the publication. If the forms have been removed, comments may be addressed to IBM Corporation, P. O. Box 50020, Programming Publishing, San Jose, California 95150. All comments and suggestions become the property of IBM.

PREFACE

This publication gives general information about VS BASIC, a program product that operates in the following virtual storage environments:

- IBM VS Personal Computing (VSPC)
- CMS (Conversational Monitor System) of VM/370
- TSO (Time Sharing Option) of VS2

VS BASIC also runs under the batch environments of OS/VS1, OS/VS2, and DOS/VS.

Included are descriptions of VS BASIC language capabilities, as well as information on operating system environments and compatibility considerations.

For additional information on VS Personal Computing (VSPC), refer to *VS Personal Computing (VSPC) for OS/VS and DOS/VS: General Information,* GH20-9070.

This publication is intended as an aid in evaluation and planning; it is not meant for the BASIC terminal user or batch programmer.

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

Number 4

April 1978

Service Changes

A reference to the general information manual for the user's system has been added to the section "Operating Environments ."

Information about VSPC Release 2 has been added to the section "VSPC (VS Personal Computing)" under "Characteristics of a Time-Sharing Enviromment."

Number 3

Release 3, October 1976

Input/Output Facilities

New statements, INPUT FROM and PRINT TO, provide the ability to retrieve input-type data from a user's file and to direct output, usually printed at the terminal, to a user's data file.

Relative-Record File Extensions

VS BASIC support of VSAM files has been extended to include support of relative record-oriented files.

Error Handling During Program Execution

VS BASIC error handling facilities are extended through the addition of the ON statement and read-only internal variables to provide information relative to errors occurring at execution time.

Operators/Mnemonics

The addition of the character mnemonics to the relational, logical, and character operators extends the use of the operators to terminal keyboards without special characters. The special characters and mnemonics can be used interchangeably.

Buffered-Ahead Terminal Input

This new capability allows the VS BASIC user to enter input data to satisfy the request of more than one INPUT statement on a single input line.

New Statements and Intrinsic Function

The new language statements are; INPUT FROM, ON, OPTION, and PRINT TO. The intrinsic functions table has been updated to incorporate the CHR function.

SUMMARY OF AMENDMENTS

Number 2

Date of Publication: September 1975
Form of Publication: GC28-8302-3

VSPC support of VS BASIC

New: Program Feature

VS BASIC can now run under the VSPC (VS Personal Computing) program products in OS/VS1, OS/VS2, or DOS/VS environments, as well as under CMS (Conversational Monitor System) and TSO (Time Sharing Option).

The debugging feature available under CMS and TSO is not available under VSPC.

As of Release 3 of VM/370, CMS will support VSAM.

Editorial changes that have no technical significance are not noted here.

Specific changes to the text made as of the publishing date are indicated by a vertical bar to the left of the text. These bars will be deleted at any subsequent republication of the page affected.

SUMMARY OF AMENDMENTS

Number 1

Date of Publication: May 1974
Form of Publication: Revision GC28-8302-1

Among the changes included in this edition are the following:

- A new table containing a summary of matrix operations
- An expanded discussion of job control statements for the batch environments OS/VS, DOS/VS, and CMS
- A new reference material chapter describing the publications that support the VS BASIC product
- A restriction in the use of the RESET statement under DOS/VS
- An increase in the printed format of short-precision items from six digits to seven digits
- A number of minor changes in the description of language features and operating environments

Editorial changes that have no technical significance are not noted here.

Specific changes to the text made as of the publishing date are indicated by a vertical bar to the left of the text. These bars will be deleted at any subsequent republication of the page affected.

INTRODUCTION

During the last several years, BASIC has gained wide acceptance as a valuable programming language. Originally developed at Dartmouth College to encourage non-programmers to use computers for simple problem-solving operations, BASIC has since come out of the classroom and into other fields. Simple to learn, its elements can be mastered by the beginner in a matter of hours. Designed for interactive systems, it is ideally suited to the conversational environment of the typewriter-like terminal connected to a central computer. The occasional user—the person who sometimes finds a computer helpful for problem-solving, but who does not consider himself a programmer—will find it ideal. The professional programmer will find that its wide range of programming capability allows him to develop sophisticated programs quickly and effectively.

The ease of use, and the effectiveness, of BASIC may best be illustrated through an example. The formula to compute compound interest is:

$$A=P(1+R/100)^t$$

where P is the principal, R is the rate of interest, and t is the number of years. A represents the amount available at the end of t.

The short program below shows how the user at a terminal can enter a number for the principal and a number for the annual interest rate, compute the amount available at the end of each year for twenty years, and print the amounts.

```
100    INPUT P,R
110    PRINT 'AT END OF YEAR:     AMOUNT AVAILABLE IS:'
120    FOR T=1 to 20
130    LET A = P*(1+R/100)**T
140    PRINT USING 145,T,A
145 :  ##              $###.##
150    NEXT T
160    END
```

If the user were to specify the numbers 200 and 5.25 as the principal and interest rate respectively, the program would produce results like the following:

```
AT END OF YEAR:     AMOUNT AVAILABLE IS:
      1                   $211.00
      2                   $222.60
      3                   $234.85
      4                   $247.76
      5                   $261.39
      6                   $275.77
      7                   $290.93
      8                   $306.93
      9                   $323.82
     10                   $341.62
     11                   $360.41
     12                   $380.24
     13                   $401.15
     14                   $423.21
     15                   $446.49
     16                   $471.04
     17                   $495.95
     18                   $524.28
     19                   $553.12
     20                   $583.54
```

VS BASIC

VS BASIC provides a powerful and responsive version of this programming language. The extensive facilities offered by VS BASIC address the scientific, engineering, commercial, and educational problem-solving needs of businesses, schools, and public agencies. Among these facilities are:

- Arithmetic capability in short- and long-precision, permitting arithmetic operations on variables and arrays.
- Character handling capabilities that permit different character lengths and the ability to search character strings.
- Array handling operations for both numeric and character one- and two-dimensional arrays.
- Input/output facilities that provide great flexibility in accessing input data and in formatting output data.
- Record-oriented file facilities that allow record processing in direct or sequential fashion.
- A set of intrinsic functions performing often-needed arithmetic and character operations.
- User-defined functions that can be defined in one statement or over a group of statements.
- Program segmentation capability that allows more than one BASIC program to be executed in sequential order.
- Program control capability that enables the user to maintain control over program flow.
- Program error handling capabilities that allow the user to receive control when errors occur during program execution.

In addition, the VS BASIC processor under CMS and TSO provides an interactive debugging facility to permit the user to dynamically debug a program at the terminal. VSPC does not provide the debug facility.

VS BASIC is designed to operate in virtual storage environments. It operates in the time-sharing environments:

- IBM VS Personal Computing (VSPC)
- CMS under VM/370
- TSO under VS2

It also operates as a batch compiler under OS/VS1, OS/VS2, DOS/VS, and CMS batch. Program compatibility is thus assured for users who may decide at some later time to migrate from one system to another.

Record-oriented file facilities or VSPC stream or record facilities use VSAM (Virtual Storage Access Method). VSAM is IBM's high-performance access method for use with direct-access files.

Highlights of the Language

This section briefly describes some of the language facilities available.

Arithmetic Capability

In addition to operations that perform addition, subtraction, multiplication, division, and exponentiation, VS BASIC enables the user to perform matrix identity, inversion, and transposition.

The VS BASIC user can enter numbers in three forms: integer, for whole numbers, fixed-point, for decimal numbers, and floating-point, for extremely large or small numbers that require exponents.

Users can enter values in short-form precision, providing seven significant digits of precision, and in long-form precision, providing fifteen significant digits.

Character Handling Capabilities

To enable the user to conveniently handle character data and to perform character string operations, VS BASIC allows him to modify or refer to portions of character strings, permits different user-defined character lengths (up to a maximum of 255 characters), and supplies a concatenation facility and intrinsic functions for character data.

With the IDX intrinsic function, the user can locate a group of characters within a character string; with the STR function, he can extract or display the group.

With the ability to define different character lengths, the user is given wide latitude in structuring character strings. He can format report headings, construct data tables, annotate report entries, etc. Moreover, with the concatenation operator (the symbols | | in combination), he can join together a number of character items; this capability is useful in putting together listings containing variable information, such as dated headings or varying combinations of numbers.

In addition, the user can define and access one- and two-dimensional character arrays.

Array Handling Capabilities

Array capabilities enable the BASIC user to store, access, and control sets of related data. In addition to defining one- and two-dimensional arrays, both numeric and character, the user can redimension arrays by changing the number of dimensions in an array or the number of elements in a dimension. He can define arrays explicitly or implicitly: explicitly, by naming the array and its dimensions in a DIM statement; implicitly, by simply naming and using the array in the program. Default dimensions of 10 for a single dimensional array and 100 for a two-dimensional array are assigned to an implicitly defined array.

The user can assign a single value to all array members and can change individual array member values.

Through the ASORT and DSORT keywords in an assignment statement, the user can sort arithmetic and character arrays in ascending or descending order.

He can facilitate input/output operations by specifying arrays and scalars on the same input/output statement as, for example, shown below:

```
PRINT MAT A, 2+B-C, MAT D$
```

This statement prints the value of the arithmetic array A (identified as an array by the word MAT), the value of the scalar expression 2+B-C, and the value of the character array D$.

Figure 3 in the section "Language Statements and Debugging Summary" contains a list of matrix operations available.

Program, File, and Terminal Input/Output Facilities

VS BASIC provides extensive data handling and input/output facilities.

Input data can be accepted from external files (those that exist outside the BASIC program) with the GET or READ FILE statements, or from the user's terminal or card reader (depending upon interactive or batch environment) with the INPUT statement. The DATA statement allows the user to define data which is then stored in an internal data table; the table is accessed by the READ statement and pointers to data in the table are reset by the RESTORE statement to permit repeated access.

Input data can be retrieved from a sequential record-oriented data file by specifying the INPUT FROM statement. The INPUT FROM statement causes subsequent INPUT statements to retrieve data from the data file specified in the INPUT FROM statement.

Buffered-ahead terminal input enables the VS BASIC user to enter, on a single input line, data values required to satisfy the requests of more than one INPUT statement. Each group of data values to satisfy a given INPUT statement is separated by a semicolon. The read-only internal variable &BUFF is available to provide the user with a means of determining the number of unprocessed groups of INPUT data in the internal buffer. Using the read-only internal variable &BUFF with the RESET statement removes all the data values from the internal buffer, thereby setting the value of &BUFF to zero. Internal read-only variables are summarized in Figure 7.

Output data can be directed to the terminal or printer (depending upon interactive or batch environment) through the PRINT statement, or to external files through the PUT or WRITE FILE statements. Output can be formatted using the FORM or Image statements. The FORM statement allows the user to specify the print line position where a data item is to appear, to forward space a number of positions on a line, to skip a number of lines, and to specify such commercial print formats as asterisk protection, comma insertion, and floating plus, minus, and dollar signs.

Output data can be directed to a sequential record-oriented data file by specifying the PRINT TO statement. The PRINT TO statement causes subsequent PRINT statements to add data to the data file specified by the PRINT statement. This output can subsequently be listed at a printer.

The RESET statement allows the user to reposition a stream-oriented file to its beginning or to its end to add new records. With record-oriented files, the RESET FILE statement can reposition a file to its beginning or to a particular record within the file.

The OPEN and EXIT statements provide additional file control. The OPEN statement permits the user to specify whether a file is to be opened for input

or output operations. The EXIT statement permits him to control end-of-file and input/output error conditions.

Record-Oriented File Capability

Record-oriented file capability gives the user a method of:

- Accessing records in a sequential manner from the beginning of a file or from some point within a file.
- Accessing records in a direct manner by matching key fields in individual records.
- Accessing records in a direct manner by specifying the relative-record number in a record-oriented file.
- Mixing direct and sequential accessing for a file.

Using such input/output statements as READ FILE, REREAD FILE, WRITE FILE, REWRITE FILE, and DELETE FILE, the user can insert, retrieve, alter, replace, and erase records in a file, through a terminal or through batch job submission.

Record-oriented files can be entry-sequenced, key-sequenced or sequenced by relative-record number. In an entry-sequenced file, records are stored in the order in which they are written, that is, in sequential order. In a key-sequenced file, records are stored according to keys. A key is a unique character value that identifies the record. When the user wishes to retrieve a record from a key-sequenced file, he tells the system which key to look for and the system returns the record with the matching key. In relative-record files, records are loaded into fixed-length slots. The relative-record number identifies that slot and the record that occupies that slot. With the relative-record file direct access facility, a user can retrieve a particular record directly by specifying its relative-record number. For many applications, direct processing is faster and more efficient than sequential file processing, and allows the user to move forward and backward through a file as his needs require. The user can still process a key-sequenced file or relative-record file sequentially, either in strict sequential order or by stepping through the file, sequentially selecting records and skipping others. Moreover, sequential processing can begin from the beginning of the file or from some record within the file.

Entry-sequenced files are read sequentially from the beginning of the file.

With record-oriented file capability, the VS BASIC user is given a powerful programming capability. He is able to access records in files created by other users, is able to share with other users the files that he creates, and thus is able to communicate with users of other languages, such as COBOL and PL/I.

Note: Because VSPC record-oriented files are nonkeyed, VS BASIC record-oriented file capabilities associated with gaining access to records by key are not supported with VSPC data files. Keyed access capability under VSPC may be achieved via external files. VSPC data files may be accessed directly by specifying their relative-record number.

Intrinsic Functions and Internal Constants

VS BASIC intrinsic functions and internal constants simplify mathematical and character string operations. An intrinsic function is a routine supplied by VS BASIC that returns a particular value, such as the sine of a number. By specifying the name of the function in a BASIC statement, the user calls the routine to perform that specific action. An internal constant is an arithmetic constant having a predefined value. By specifying the name of the constant in a BASIC statement, the user causes that value to be made available to his program, thereby making it unnecessary for him to define the value himself.

Among the many mathematical functions available are SIN, COS, TAN, MAX, MIN, and SUM. Character handling functions include STR and IDX. Some functions—TIM, DAT, CLK, and CPU, to perform clock and date manipulations—are useful in accounting procedures, in printing report headings, and in obtaining computer usage times. Figure 4, in the section "Language Statements and Debugging Summary," contains a complete list of VS BASIC intrinsic functions.

Installation-written functions that are to be used in many applications can be installed in the library of intrinsic functions and can thus be made available, like IBM-supplied intrinsic functions, to all users of the library. This capability provides programming convenience and enables VS BASIC users to share such functions, thereby enhancing communication between users.

Figure 5 in "Language Statements and Debugging Summary" contains a complete list of VS BASIC internal constants. Among internal constants available are &PI for the value of π, &SQR2 for the square root of 2, and, to address the growing needs of applications performing metric operations, &INCM to convert inches to centimeters and &GALI to convert gallons to liters.

Read-Only Internal Variables

VS BASIC provides several read-only internal variables to aid the user in such operations as buffered-ahead terminal input and program error handling. An internal variable is a character or numeric value set (changed as necessary by VS BASIC) that is available to the user in certain operations. Internal variables can be used with both numeric and relational operators. Among the internal variables that aid in program error handling are &ERR, which helps to identify the error, and &FILE, which names the data file related to the error. Figure 7 lists and summarizes the use of the internal variables.

User-Defined Functions

To enable the BASIC user to define specialized functions not available as intrinsic functions, VS BASIC permits single-line and multi-line functions and multiple arguments. The user can define a function and then refer to it by name in his program, as he would an intrinsic function.

User-defined functions may be character or numeric, and may accept one or multiple arguments (character and numeric), or even no arguments.

The user can define a function in a single statement or in a group of statements. The DEF and FNEND statements are used to delimit a multi-line function, the DEF statement naming the function and marking the beginning of the group and the FNEND statement marking the end of the group. Any value to be returned by the function to the program is defined by the RETURN statement.

Program Segmentation

To enable BASIC users to execute a number of BASIC programs sequentially, the CHAIN and USE statements can be used.

By specifying the name of a program in a CHAIN statement, the user transfers control to that program when the CHAIN statement is executed. By specifying a character variable in the USE statement, he can receive a parameter passed from the "chaining" program to the "chained" program.

With the program chaining technique available to him, the VS BASIC user is able to build modular programs and intermix them at will to meet various applications. He can develop programs that perform specific operations, can store them in a library of programs, and can selectively retrieve them for each particular application, using the same program over and over again in different applications.

Program Control Statements

Program control statements enable the user to maintain control over program flow, that is, let him decide the sequence in which sections of a program are to be executed.

The IF statement can contain relational and logical operators to test expressions, and can take alternative action, such as branching to another statement if a condition is true or changing a variable's value if the condition is false.

The GOTO statement transfers control either to another statement unconditionally (simple GOTO), or to one of a set of statements depending on the value of an expression (computed GOTO). The GOSUB statement similarly can transfer control unconditionally or to one of a set of statements. The difference between these statements is that the GOSUB transfers control to a subroutine; when a RETURN statement is executed, control returns to the statement following the GOSUB statement.

The FOR and NEXT statements are used to create program loops; these statements help in coding a program section where a group of operations is to be repeated.

Program Error Handling

The VS BASIC user can control errors that occur during program execution by specifying the action to be taken when a certain error condition is encountered.

The user can specify error clauses in conjunction with both stream and record-oriented input/output statements. When errors associated with input/output operations occur, this facility transfers control to the designated statement, rather than terminating the program.

With the ON statement the user can identify specific errors to be handled by the program together with the action to be taken.

Four read-only internal variables—&CODE, &ERR, &FILE, and &LINE—provide the user with additional error handling capabilities in determining what the error was and where it occurred. Read-Only Internal Variables are summarized in Figure 7.

Interactive Debugging

An additional capability available to time-sharing VS BASIC users operating under TSO or CMS is the interactive debugging facility, which simplifies the process of debugging a program. Debugging subcommands allow the user to go through his program at his terminal and trace program execution. He can start and stop program execution dynamically, establish breakpoints, display and alter program values, control logic flow, monitor the frequency of execution of program statements, monitor arithmetic operations, and experiment with different corrections to program errors. Figure 6, in the section "Language Statements and Debugging Summary," contains a complete list of the debugging subcommands.

With interactive debugging, the VS BASIC user under CMS or TSO can isolate any area of a program that is giving trouble. Because he can control logic flow during execution, he can often check out the "fix" at the same time he changes the program—without changing the source code or recompiling until he is satisfied that he has corrected the problem. The interactive debugging facility is not available under VSPC.

LANGUAGE STATEMENTS AND DEBUGGING SUMMARY

The VS BASIC language consists of language elements and statements, intrinsic functions, internal constants and, internal variables. The interactive debugging facility under CMS and TSO consists of system-like subcommands used at program execution time.

Figure 1 briefly summarizes VS BASIC language elements. Language statements are summarized in Figure 2. Matrix operations available with the language are summarized in Figure 3.

Intrinsic functions are routines supplied by VS BASIC which can be referred to during program execution without being defined in the BASIC program. Included are mathematical functions for computing square roots, for obtaining logarithmic, trigonometric, maximum and minimum values, and for performing other operations; character functions for string handling operations; and miscellaneous functions for converting Fahrenheit to centigrade and vice versa, for counting the number of data items processed by input/output statements, for returning date and time information, and for other operations. Intrinsic functions are summarized in Figure 4.

Internal constants are arithmetic constants whose value is predefined in the VS BASIC language. Internal constants are summarized in Figure 5.

The VS BASIC interactive debugging facility is made available to the user if he specifies the TEST option in the system RUN, EDIT, or VSBASIC command (these commands are discussed in the section "Operating Environments"). Debugging subcommands are summarized in Figure 6.

Internal variables are listed and summarized in Figure 7.

Data Types	• *Arithmetic:*	Constants, variables and arrays, in integer, fixed-point, and floating-point, with precision in short and long form. Constants are specified as fixed numeric values. *Examples:* 4, 255, 25.3, 3E6 Variables are named by a single letter or a letter followed by a digit. *Examples:* A, B1 Arrays are named by a single letter. Array elements are specified by an array name followed by one or two subscripts enclosed in parentheses. *Examples:* C(4), X(5,5)	
	• *Character:*	Constants, variables and arrays, from 1 to 255 characters. Character constants are enclosed in single or double quotation marks. *Examples:* 'one', "two", "literal data 456" Variables are named by a letter followed by a dollar sign ($). *Example:* A$ Arrays are named by a letter followed by a dollar sign ($). Array elements are specified by an array name followed by one or two subscripts enclosed in parentheses. *Examples:* Q$ (10), B$ (3,5)	
Operators	• *Arithmetic:*	+	Addition and unary plus
		-	Subtraction and unary minus
		*	Multiplication
		/	Division
		↑ or **	Exponentiation
	• *Relational:*	< or .LT.	Less than
		> or .GT.	Greater than
		= or .EQ.	Equal to
		≠ or .NE. or <>	Not equal to
		≤ or .LE. or < =	Less than or equal to
		≥ or .GE. or > =	Greater than or equal to
	• *Logical:*	& or .AND.	And
		| or .OR.	Or
	• *Character:*	|| or .CAT.	Concatenation
Expressions	• *Arithmetic:*	Simple arithmetic variable Subscripted arithmetic array reference Arithmetic constant Function reference Internal constant Appropriate combinations of the above separated by arithmetic operators and parentheses *Examples:* A A+B1+C(3)-4 (X*SQR(B))/&PI	
	• *Character:*	Simple character variable Subscripted character array reference Character constant Function reference Appropriate combinations of the above separated by concatenation operators and parentheses *Examples:* Q$ (4) || Q$ (5) || ' is date required'	

Figure 1. VS BASIC Language Elements

Statement	*Use*
array assignment	Assigns values to arithmetic and character arrays; performs addition, subtraction, multiplication, and such functions as identity, transposition, inversion, and sorting. (These operations are summarized in Figure 3.)
scalar assignment	Assigns values to character or arithmetic variables and array members.
CHAIN	Terminates execution of current program and begins execution of another program.
CLOSE [FILE]	Deactivates the stream-oriented or record-oriented file(s) specified in the statement.
DATA	Defines input for an internal data table.
DEF	Defines a user function.
DELETE FILE	Deletes records from a record-oriented file.
DIM	Defines the dimensions of an array and the length of a character variable.
END	Defines the end of the program.
EXIT	Specifies action to be taken for exceptional execution conditions encountered during input/output processing.
FNEND	Marks the end of a group of statements defining a user function (used with DEF).
FOR	Initializes a loop (used with NEXT).
FORM	Controls format of printed output or record-oriented input/output.
GET[1]	Reads input from a stream-oriented file into scalar and array variables.
GOSUB (simple)	Transfers control unconditionally to a subroutine (used with RETURN).
GOSUB (computed)	Transfers control to one of several subroutines, depending on the value of a test expression (used with RETURN).
GOTO (simple)	Transfers control unconditionally to a statement.
GOTO (computed)	Transfers control to one of several statements, depending on the value of a test expression.
IF	Tests a condition and transfers control or executes a statement based on the results of the test.
Image	Controls format of printed output (used with PRINT).
INPUT[1]	Reads input from an input device into scalar and array variables.
INPUT FROM	Provides the ability to retrieve input-type data from a record-oriented data file.
NEXT	Delimits a loop (used with FOR).
ON	Defines certain types of error conditions that can be handled within a VS BASIC program at execution time.
OPEN [FILE]	Activates the stream-oriented or record-oriented file(s) specified in the statement.
OPTION	Provides for the selection of specific options that can be applied to a VS BASIC program.
PAUSE	Interrupts program execution.
PRINT[1]	Prints expression values from scalar and array variables to an output device.
PRINT TO	Provides the ability to direct output, normally printed at the terminal, to a record-oriented data file.
PRINT USING[1]	Prints values in a user-chosen format.

Figure 2 (Part 1 of 2). VS BASIC Language Statements

Statement	*Use*
PUT[1]	Writes output into a stream-oriented file.
READ[1]	Reads data from an internal data table (used with DATA).
READ FILE[1]	Reads records from a record-oriented file into scalar and array variables.
REM	Inserts remarks into a source program.
REREAD FILE[1]	Makes accessible again the last record read from a record-oriented file.
RESET [FILE]	Repositions stream-oriented files to their beginning or end (except for DOS/VS files, which cannot be positioned to their end), and repositions record-oriented files to their beginning or to a particular record.
RESTORE	Resets an internal file (used with READ and DATA).
RETURN	Returns control from a subroutine (used with GOSUB) or from a user function (used with DEF).
REWRITE FILE[1]	Alters existing records in a record-oriented file.
STOP	Terminates program execution.
USE	Receives a value passed by another program (used with CHAIN).
WRITE FILE[1]	Writes records to a record-oriented file.

[1] Matrix data in the statement is identified by the word MAT.

Figure 2 (Part 2 of 2). VS BASIC Language Statements

Operation	*Example*	*Explanation*
Scalar Assignment [1]	MAT A = (J+5) MAT A$ = ('XYZ')	The value of the expression in parentheses is assigned to all the members of the array named A (or A$).
Array Assignment [1]	MAT A = B MAT A$ = B$	The value in each member of array B (B$) is assigned to the corresponding member of array A(A$). Both arrays must have identical dimensions; for instance, A(5,3) and B(5,3).
Array Addition	MAT A = B + C	The value in each member of array C is added to the value in the corresponding member of array B and the result is assigned to the corresponding member of array A. All arrays must have identical dimensions; for instance, A(4,3), B(4,3), and C(4,3).
Array Subtraction	MAT A = B – C	The value in each member of array C is subtracted from the value in the corresponding member of array B and the result is assigned to the corresponding member of array A. All arrays must have identical dimensions; for instance, A(4,3), B(4,3), and C(4,3).
Scalar Multiplication	MAT A = (3) *B	The value in each member of array B is multiplied by the expression in parentheses and the result is assigned to the corresponding member of array A. Both arrays must have identical dimensions; for instance, A(5,3) and B(5,3).
Matrix Multiplication	MAT A = B * C	Each element of array A is the dot product of the corresponding row of the first array and corresponding column of the second array.
Identity Function	MAT A = IDN	An identity matrix is assigned to array A. Array A must be a square array; for instance, A(5,5).
Inverse Function	MAT A = INV(B)	The inverse matrix of array B is assigned to array A. Both arrays must be square and have identical dimensions; for instance, A(4,4) and B(4,4).
Transpose Function	MAT A = TRN(B)	The transpose matrix of array B is assigned to array A. Both arrays must be two-dimensional, with the number of rows in one array equal to the number of columns in the other; for instance, A(5,3) and B(3,5).
Ascending Sort [1]	MAT A=ASORT(B) MAT A$=ASORT(B$)	The values in array B (B$) are sorted in ascending order and the result is assigned to array A (A$). Both arrays must have identical dimensions; for instance, A(5,3) and B(5,3).
Descending Sort [1]	MAT A=DSORT(B) MAT A$=DSORT(B$)	The values in array B (B$) are sorted in descending order and the result is assigned to array A (A$). Both arrays must have identical dimensions; for instance, A(5,3) and B(5,3).

[1] Can be performed on both arithmetic and character arrays. All other functions are performed on arithmetic arrays only.

Figure 3. VS BASIC Matrix Operations

Functions Returning an Arithmetic Value	
Function	*Meaning*
ABS (X)	Finds the absolute value of X.
ACS (X)	Finds the arccosine (in radians) of X.
ASN (X)	Finds the arcsine (in radians) of X.
ATN (X)	Finds the arctangent (in radians) of X.
CEN (X)	Converts corresponding Fahrenheit degrees X into Centigrade degrees.
CNT	Counts the number of data items successfully processed by the last I/O statement.
COS (X)	Finds the cosine of X radians.
COT (X)	Finds the cotangent of X radians.
CPU	Finds program execution time in seconds.
CSC (X)	Finds the cosecant of X radians.
DEG (X)	Finds the number of degrees in X radians.
DET (A)	Finds the determinant of square arithmetic array A.
DOT (A, B)	Finds the dot product of arithmetic arrays A and B.
EXP (X)	Raises the value of e to power X.
FAH (X)	Converts corresponding Centigrade degrees X into Fahrenheit degrees.
HCS (X)	Finds the hyperbolic cosine of X.
HSN (X)	Finds the hyperbolic sine of X.
HTN (X)	Finds the hyperbolic tangent of X.
IDX (C, D)	Finds the position relative to 1 of the character string D within string C.
INT (X)	Returns the integer part of the real number X.
JDY [(C)]	Converts the current date or the Gregorian date C into Julian form.
KLN (C)	Finds the length, in bytes, of an embedded key in file C.
KPS (C)	Finds the byte position relative to 0 of the start of an embedded key in file C.
LEN (C)	Finds the length of the character string C, less trailing blanks.
LGT (X)	Finds the common logarithm (base 10) of X.
LOG (X)	Finds the natural logarithm (base e) of X.
LTW (X)	Finds the binary logarithm (base 2) of X.
MAX (X_1 , . . . X_n)	Finds the maximum value in the list of variables.
MIN (X_1 , . . . X_n)	Finds the minimum value in the list of variables.
NUM (C)	Returns an arithmetic value obtained by converting the character string C.
PRD (A)	Finds the product of the elements in arithmetic array A.
RAD (X)	Finds the number of radians in X degrees.
RND [(X)]	Generates random numbers using X, or a default value, as a seed.
RLN (C)	Returns the length of the last record referenced in file C.
SEC (X)	Finds the secant of X radians.
SGN (X)	Returns the sign of X (in the form +1, -1, or 0).
SIN (X)	Finds the sine of X radians.
SQR (X)	Finds the square root of X.
SUM (A)	Finds the sum of the elements in arithmetic array A.
TAN (X)	Finds the tangent of X radians.
TIM	Finds the time of day in seconds since midnight.

Functions Returning a Character Value	
Function	*Meaning*
CHR (X)	Converts the scalar numeric expression to its equivalent character form.
CLK	Produces the time of day as an eight-character string in the form hh:mm:ss, to indicate hours, minutes, seconds.
DAT [(X)]	Converts the current date or the Julian date X into Gregorian form.
STR (C,X[,Y])	Extracts the portion of the character string C beginning with position X for a length of Y or until the end of the string.

Notes: The characters X and Y represent arithmetic scalar arguments.
The characters A and B represent arithmetic array arguments.
The characters C and D represent character scalar arguments.
The STR intrinsic function can also be used as a pseudo variable; that is, it can be used in place of a variable to receive data. Thus, it may appear to the left of the equal sign in a scalar assignment statement, or in an input data list.

Figure 4. VS BASIC Intrinsic Functions

Name	*Use*	*Short-Form Value*	*Long-Form Value*
&PI	Arithmetic value of π.	3.141593	3.14159265358979
&E	Arithmetic value of the base *e* in the system of natural logarithms.	2.718282	2.71828182845905
&SQR2	Square root of 2.	1.414214	1.41421356237309
&INCM	Centimeters to the inch.	2.540005	2.54000500000000
&LBKG	Kilograms to the pound.	.4535924	.45359237000000
&GALI	Liters to the gallon.	3.785412	3.78541178400000

Figure 5. VS BASIC Internal Constants

Subcommand	*Use*
AT	Sets breakpoints in the program where control can be given to the user or to a list of subcommands.
END	Terminates the debugging session.
GO or GOTO	Resumes execution of the VS BASIC program.
HELP	Provides user information about syntax and function of debugging subcommands. (TSO only)
IF	Tests program conditions and executes either a user-specified subcommand or the HALT option to give control to the user.
LIST	Displays the value of data items.
LISTBRKS	Lists the current breakpoints.
LISTFREQ	Lists the number of times VS BASIC statements have been executed, and statement numbers of statements never executed.
NEXT	Sets a breakpoint at the next executed VS BASIC statement.
OFF	Removes breakpoints.
OFFWN	Ends monitoring of a condition specified by the WHEN subcommand.
QUALIFY	Specifies the program unit (main program or user function) to which debugging subcommands are to apply.
RUN	Resumes execution of the VS BASIC program without further debugging. At end of the program, a prompt will be given for possible program restart.
SET	Assigns values to data items.
TRACE	Traces program flow of control.
WHEN	Monitors changes of values of specified variables and array elements and interrupts program execution when a specified condition occurs.
WHERE	Produces a traceback of the program's flow of control.

Figure 6. VS BASIC Debugging Subcommands

Name	*Meaning*
&BUFF	Contains the number of unprocessed groups of input data values.
&CODE	Contains the system return code resulting from VSAM error.
&ERR	Contains the error message number for identifying the particular error that was encountered.
&FILE	Contains the name of the data file associated with the error condition.
&LINE	Contains the line number of the statement where the error occurred.
&REC	Contains the number of the last record successfully referred to in a relative record-oriented file.

Figure 7. VS BASIC Read-Only Internal Variable

OPERATING ENVIRONMENTS

The VS BASIC processor is designed to operate in a virtual storage system, in both time-sharing and batch environments. It is a single-pass compiler, enhancing compile-time performance. It accepts up to 1000 source statements and produces executable code that can be run immediately after compilation or that can be stored as an object program and run at some later time.

Under VSPC, the VS BASIC processor is reentrant in the VS1, VS2, and DOS/VS environments. Under VS1 and VS2, VS BASIC is reentrant. Under TSO, the VS BASIC processor is reentrant except for the interactive debugging module. Under DOS/VS, the processor is reentrant except for the executor module.

Where the processors are reentrant, they may be placed in the link pack area or the shared virtual area under DOS/VS so that many users can simultaneously refer to them.

Under CMS, the VS BASIC processor is not reenterable.

Because all environments in which VS BASIC operates can be virtual, real storage requirements are largely a function of performance desired. Typical program compilation and execution without debugging requires approximately 128K of virtual storage for the processor and user program. With debugging, approximately 256K of virtual storage is required.

VS BASIC runs on the processing units that are supported by the environments described below. See the general information manual of the system you are using for a complete list.

Programming Systems

In a time-sharing environment, VS BASIC operates under the following systems:

- VSPC in VS1, VS2, or DOS/VS environments
- CMS as a conversational system component of VM/370
- TSO as a subsystem of VS2

In a batch environment, VS BASIC operates under VS1, VS2, DOS/VS, and CMS.

Record-oriented file capability or VSPC file facilities require the use of VSAM (Virtual Storage Access Method).

Characteristics of a Time-Sharing Environment

The time-sharing VS BASIC user can do the following at his terminal:

- Build and execute programs.
- Modify programs freely through line insertions, deletions, and replacements.
- Save programs in private libraries or in system storage areas.
- Dynamically debug his programs (CMS and TSO only).

Although each time-sharing environment offers somewhat different operating features; they provide similar facilities for the VS BASIC user. These environments are briefly described below.

VSPC (VS Personal Computing)

VSPC is a program product that will operate in OS/VS1, OS/VS2, or DOS/VS environments.

VSPC offers users at remote conversational terminals a choice of responsive tools to meet their needs for personal computing. VSPC provides a comprehensive set of functions for data and source program manipulation that is tailored for end users without extensive data processing knowledge. With VSPC a user can also submit jobs for batch processing and retrieve job output at the terminal.

To the VS BASIC user it offers:

- Source text preparation and storage through LOAD and SAVE commands and the line entry, editing, and library facilities.
- Compilation and execution through the STORE and RUN commands. Through these commands, the user can instruct the VS BASIC compiler to store the object code; to run a previously compiled program or to compile and execute a program.
- A method of chaining to programs written in VSPC FORTRAN or VSPC PL/I or to VSPC command list files (CLIST).
- Security through the filename and LOGON passwords and the PROTECT command.
- Record processing of external VSAM files (entry-sequenced, key-sequenced, and relative-record).
- Access to VSPC files in either a direct or a sequential manner.

With VSPC Release 2, the VS BASIC user is offered:

- Full screen editing and program function key support for greater usability of the IBM 3270 Display System.
- Record processing of external VSAM files with primary or alternate indexes.

TSO (Time Sharing Option)

TSO is a general-purpose time-sharing subsystem of VS2. It provides a powerful command language through which a user at a terminal can access and share the facilities of VS2. Its range is such that it can be used by the professional programmer and the novice.

To the VS BASIC user it offers:

- Source text preparation and storage through line entry, editing, and system library facilities.
- Compilation and execution through either the VSBASIC, EDIT, or RUN command. Through these commands, the user can instruct the VS BASIC compiler whether to store a successfully compiled program, whether to execute a program immediately after compilation, and whether to make available the interactive debugging facility.

- Data file control through ALLOCATE, DELETE (stream-oriented files only), FREE (stream-oriented files only), and RENAME commands. VS BASIC files follow TSO naming conventions in the form:

  userid.name.DATA.

- Security through the LOGON password and the PROTECT command.
- Record processing of VSAM files in a direct or sequential manner.

CMS (Conversational Monitor System)

CMS is the conversational system component of VM/370. It gives the user control over a virtual computing system; that is, the user controls all system facilities at the terminal as though he were sitting at an operator's console. CMS is directed to scientific and technical problem solvers, research groups in schools, and commercial users developing programs at terminals.

To the VS BASIC user it offers:

- Source text preparation and storage through line entry, editing, and library routines.
- Compilation and execution through the VSBASIC command. Like the TSO commands, this command allows the user to instruct the VS BASIC compiler whether to store a successfully compiled program, whether to execute a program immediately after compilation, and whether to make available the interactive debugging facility.
- Data file control through ERASE and RENAME commands. VS BASIC data files are processed by specifying the filetype VSBDATA.
- Security through the LOGON command to VM/370 and the maintenance of the user's virtual machine.
- Record processing of VSAM files in a direct or sequential manner.

Characteristics of a Batch Environment

A batch environment is typified by the use of a centralized computing installation to process jobs in a batched, or serial, manner; that is, jobs are grouped and then run by the system one by one in sequential order. The user communicates with a batch system through the job control statements he submits with his program. Job control statements used to execute a VS BASIC program are the following.

Under VS1 and VS2:

1. The JOB statement, to identify the user to the system.
2. The EXEC statement, to call the VS BASIC processor.
3. The following DD statements, to specify information about the VS BASIC program:
   a. A DD statement to name the VS BASIC program and to specify its location in the system's resources.
   b. CONTROL, to describe a file of RUN commands. The RUN command is the batch counterpart of the time-sharing RUN, EDIT, or VSBASIC commands. It indicates whether the program is a source program or an object program, whether a compiled program should be stored, and whether a program should be run immediately after compilation. A RUN command is required for each VS BASIC program. The CONTROL DD statement may describe a file of many RUN commands, thereby allowing sequential execution of many VS BASIC programs.
   c. SYSPRINT, to describe the output data set that is to receive system messages and output from VS BASIC PRINT statements.
   d. Other DD statements, as required, to describe the object program being stored, and the files used by the program, such as data for INPUT

statements, files used by other input/output statements, and programs called by CHAIN statements.

Under DOS/VS:

1. The JOB statement.
2. The EXEC statement.
3. The following statements containing information about the VS BASIC program:
   a. DLBL and EXTENT statements, to describe files on direct access devices.
   b. TLBL, to describe files on magnetic tape.
   c. ASSGN statements, as required, to assign files to physical devices.

   In addition, the system logical units SYSIPT and SYSLST are required: SYSIPT, to describe the RUN command, the VS BASIC program, and input data to the program; and SYSLST, to describe the device that is to receive system messages and output from VS BASIC PRINT statements.

Under CMS:

1. The JOB statement.
2. The SET statement, to optionally specify user options controlling system timing, printing, and punching resources.
3. The /* statement, to denote the end of input to the CMS Batch Facility.

   In addition, the CP command LINK may be needed to communicate data information to the system.

Compatibility with Other IBM BASIC Products

All correct programs running under ITF: BASIC will run under VS BASIC as long as the maximum literal pool size is not exceeded.

Files created under OS and TSO ITF: BASIC can be read by VS BASIC provided the filename is specified in the form 'userid.DATA(name)'. Files created under DOS ITF: BASIC must first be converted to conventional DOS format; the ITF CONVERT command may be used.

All correct programs running under the CALL-OS BASIC processor supported by CMS will run under VS BASIC after adjustment of file input/output statements. CMS editing commands can be used to change these statements to conform to VS BASIC syntax.

Additionally, under CALL-OS BASIC, if an OPEN statement is issued for a file already open, the file is repositioned to its beginning. Under VS BASIC, an OPEN statement for a file already open is ignored. CALL-OS BASIC programs containing such OPEN statements should be converted by adding a CLOSE statement before the OPEN statement.

Files created by CALL-OS BASIC under CMS can be read by VS BASIC after being converted to VS BASIC format using the file conversion utility supplied with VS BASIC.

All correct programs running under VS BASIC (non-VSPC) will run under VS BASIC for VSPC after adjustment of file input/output statements and

program names in CHAIN statements. These changes can be made using the VSPC editing facilities.

Non-VSPC VS BASIC files can be imported from sequential data sets to VSPC by using the VSPC Service Program.

The MAT ZER and MAT CON functions, which exist in earlier BASIC processors, are replaced in VS BASIC by the facilities of the MAT assignment statement. However, as a compatibility convenience, these functions are available in VS BASIC so that existing BASIC programs containing references to them will not require change in order to run under VS BASIC.

Cross Language Data Exchanging

Data can be exchanged using VSPC files, provided the data types written are readable by the receiving language.

Exchanging Data With VSPC FORTRAN

VS BASIC record-oriented files can be read by FORTRAN formatted and unformatted record I/O statements. Similarly FORTRAN files written by formatted or unformatted I/O statements can be read by VS BASIC record-oriented I/O statements. However, it is the user's responsibility to ensure that the data types match. Figure 8 summarizes the compatibility.

VS BASIC Data Type	FORTRAN Formatted	FORTRAN Unformatted
S(short precision floating point)	A(4) format into real *4	Real *4 item
L(long precision floating point) or unconverted arithmetic variable	A(8) format into real *8	Real *8 item
PIC notation (numeric with exponential)	D or F format	—
PIC (numeric with decimal point)	F format	—
PIC (numeric)	I format	—
C (character)	A format	Any data item

Figure 8. Data Type Compatibility

Note the following rules:

1. VS BASIC cannot read FORTRAN formatted or unformatted complex or logical data.
2. FORTRAN cannot read:
   - PD (packed decimal)
   - NC (zoned decimal)
   - PIC with editing characters (*, commas, slashes, or currency signs).
3. VS BASIC will be able to read FORTRAN direct access files in sequential mode.
4. VS BASIC record files can be read using FORTRAN list-directed, statements, but the constraints required for success are strict and their use is not recommended.
5. VS BASIC writes arrays in row major order; they must, therefore, be transposed before they can be used by any FORTRAN array operator.

6. VS BASIC format control specifications X[n] or POS[n] can be used to skip over unreadable data items.
7. VS BASIC stream I/O facility is similar in many ways to FORTRAN's list-directed I/O. However, VS BASIC does not recognize record boundaries except where they are necessary for writing the file. As the position of the record boundary cannot be determined easily, it is normally only possible for FORTRAN to read a VS BASIC stream file by the use of one READ statement which causes FORTRAN to read VS BASIC records until it fills all the data items. VS BASIC can read a FORTRAN list-directed record provided that neither COMPLEX nor LOGICAL data types are used. It should be noted that VS BASIC writes arrays in row major order and FORTRAN in column major order.

Exchanging Data With APL

APL's EBCDIC files can be read by VS BASIC I/O statements. For stream GET statements, the APL file must be written so that each field is separated by a blank. For record-oriented I/O statements, each character vector written by APL will appear as a separate record.

VS BASIC's stream files can be read by APL but the character vector read must be decoded by the receiving function. For files containing only numeric data, the EXECUTE operator may be used. Each record of a VS BASIC VSPC file can be read into a character vector. Each item in the record should be in EBCDIC if it is to be interpreted by APL and the receiving function must decode it.

Compatibility with Previous Releases

VS BASIC Release 2 programs for which object programs were saved must be recompiled with the Release 3 processor.

Those few user programs which depend upon a PRINT USING statement with a reference to an IMAGE to force the preceding data to be printed, will need to be modified. One way is through the addition of a PRINT statement which will then cause the preceding data to be printed.

REFERENCE MATERIAL

The System/370 VS BASIC processor is supported by the publications described below. A summary of the available literature supporting IBM System/370 will be found in *IBM System/370 Bibliography,* GC20-0001.

B is for BASIC

These publications, SC28-8300 (for TSO) and SC28-8310 (for CMS), provide a tutorial introduction to the use of VS BASIC under a time-sharing system. They are intended for users with no prior programming experience. Elementary BASIC programming concepts and language statements are presented in a simple step-by-step approach. Language features are gradually introduced through meaningful examples. Material of a detailed nature is not described here; such material is left to the *VS BASIC Language* manual.

These publications can be used alone or in conjunction with instructor availability.

VS BASIC Language

This publication, GC28-8303, describes the statements in the VS BASIC language. The publication is divided into two main parts. The first presents the language in a tutorial manner, proceeding from simple to complex concepts. This part takes the reader through various steps in planning a program, and fits language statements into those steps. Numerous examples are included. The second part is a reference section, presenting BASIC language syntax and rules of usage. VS BASIC statements are organized into alphabetical order for easier retrievability. Implementation dependencies are noted, so that the reader is aware of how a particular operating environment affects the use of the language.

This publication is intended to be used with an accompanying Terminal User's Guide or Programmer's Guide for the appropriate environment. Conceptually, the Language manual and the accompanying Guide can be thought of as separate sections of one document that gives the user all the information needed to program VS BASIC in a particular environment. The accompanying Guide will include a master index containing Language manual entries as well as entries to material in the Guide.

Some knowledge of programming concepts is required for use of this publication. The novice programmer will find the publication *B is for BASIC* useful before proceeding to this publication.

VS BASIC TSO Terminal User's Guide

This publication, SC28-8304, is directed to the VS BASIC user programming in the TSO environment. It is intended to be used with the *VS BASIC Language* manual to provide a complete guide to using VS BASIC under TSO.

The publication introduces the structure and commands of the TSO environment and is a primary source of information for the commonly-used system command formats, the debug facility, and VS BASIC language implementation dependencies under TSO.

No previous experience with TSO is required for use of this publication.

VS BASIC (TSO) Reference Summary

This reference card, SX28-6385, provides a handy guide to using VS BASIC under TSO. It summarizes language syntax, rules of usage, VS BASIC debugging subcommands, and the most commonly-used TSO system commands.

The card is intended to supplement the Language/TSO Terminal User's Guide combination so that the user need rarely refer to either book once he's achieved a working familiarity with VS BASIC under TSO.

VS BASIC CMS Terminal User's Guide

This publication, SC28-8306, is directed to the VS BASIC user programming in the CMS environment, both interactive and batch. It is intended to be used with the *VS BASIC Language* manual to provide a complete guide to using VS BASIC under CMS.

The publication introduces the structure and commands of the CMS environment and is a primary source of information for the commonly-used system command formats, the debug facility, and VS BASIC language implementation dependencies under CMS.

No previous experience with CMS is required for use of this publication.

VS BASIC (CMS) Reference Summary

This reference card, SX28-6386, provides a handy guide to using VS BASIC under CMS. It summarizes language syntax, rules of usage, VS BASIC debugging subcommands, and the most commonly-used CMS system commands.

The card is intended to supplement the Language/CMS Terminal User's Guide combination so that the user need rarely refer to either book once he's achieved a working familiarity with VS BASIC under CMS.

VS BASIC for VSPC: Terminal User's Guide

This publication is directed to the VS BASIC user programming in the VSPC environment. It is intended to be used with the *VS BASIC Language* manual to provide a complete guide to using VS BASIC for VSPC.

The publication introduces the structure and commands of the VSPC environment and is a primary source of information for the commonly-used system command formats and VS BASIC language implementation dependencies under VSPC.

No previous experience with VSPC is required for use of this publication.

VS BASIC for VSPC: Reference Summary

This reference card provides a handy guide to using VS BASIC for VSPC. It summarizes language syntax, rules of usage, and the most commonly-used VSPC system commands.

The card is intended to supplement the Language/VSPC Terminal User's Guide combination so that the user need rarely refer to either book once he's achieved a working familiarity with VS BASIC for VSPC.

VS BASIC OS/VS and DOS/VS Programmer's Guide

This publication, SC28-8308, is directed to the VS BASIC user programming in the batch environment of VS1, VS2, or DOS/VS. It is intended to be used with the *VS BASIC Language* manual to provide a complete guide to using VS BASIC in these batch environments.

In addition to containing information about the use of the VS BASIC processor, this publication is a primary source of information for that subset of the OS/VS or DOS/VS job control language needed to compile and execute a VS BASIC program.

No previous experience with either OS/VS or DOS/VS is required for use of this publication.

VS BASIC Installation Reference Material

This publication, SC28-8309, provides installation personnel with information on how to install the VS BASIC processor under OS/VS, DOS/VS, VSPC, TSO, and CMS. Included with the step-by-step procedure for each environment are storage information and system programmer considerations.

This publication is essentially supplemental, in that it assumes the availability of or familiarity with other system publications pertaining to the use of the environment under which VS BASIC will operate (for example, system generation reference publications, storage estimates, VM/370 Planning Guide.

VS BASIC Program Logic

This publication, LY28-6422, describes the internal structure and method of operation of the VS BASIC processor, and is meant to be used for purposes of program maintenance.

Program logic manuals are available to licensees only and are provided for the use of program maintenance personnel. They are not necessary for the installation or use of the program product. Contact your IBM representative for further information.

INDEX

J

K

L

M

N

O

P

R

S

T

U

V

W

VS BASIC General Information
GC28-8302-5

Reader's Comment Form

Your comments about this publication will help us to improve it for you. Comment in the space below, giving specific page and paragraph references whenever possible. All comments become the property of IBM.

Please do not use this form to ask technical questions about IBM systems and programs or to request copies of publications. Rather, direct such questions or requests to your local IBM representative.

If you would like a reply, please provide your name and address (including ZIP code).

Fold on two lines, staple, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

Fold and Staple

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 Staple

VS BASIC General Information Printed in U.S.A. GC28-8302-5

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601

IBM® Technical Newsletter

This Newsletter No. GN26-0902
Date April 21, 1978

Base Publication No. GC28-8302-5

Prerequisite Newsletters None

VS BASIC
General Information

This technical newsletter, a part of Release 3 of VS BASIC, program number 5748-XX1, provides replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered.

Pages to be inserted and removed are:

cover, edition notice
9,10
31-32.1 (32.1 added)

Each technical change is marked by a vertical bar to the left of the change.

Summary of Amendments

Changes included in this newsletter are summarized under "Summary of Amendments" following the list of figures.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

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

Printed in U.S.A.

GC28-8302-5

VS BASIC General Information Printed in U.S.A. GC28-8302-5

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, N.Y. 10604

IBM World Trade Americas/Far East Corporation
Town of Mount Pleasant, Route 9, North Tarrytown, N.Y., U.S.A. 10591

IBM World Trade Europe/Middle East/Africa Corporation
360 Hamilton Avenue, White Plains, N.Y., U.S.A. 10601