IBM

IBM BASIC/VM Installation and Customization

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

Order Number:
SC26-4025-2

Program Number:
5668-996

Release Number:
2

IBM BASIC/VM Installation and Customization

Program Product

Order Number:
SC26-4025-2

Program Number:
5668-996

Release Number:
2

Industry Standards

The IBM BASIC/VM program product is designed according to the specifications of the following industry standards, as understood and interpreted by IBM as of December, 1984:

- American National Standard for Minimal BASIC, ANSI X3.60-1978
- International Organization for Standardization proposed standard ISO Minimal BASIC dp ISO-6373
- European Computer Manufacturer's Association, Standard ECMA-55 Minimal BASIC, January 1978

These standards are technically equivalent.

In addition, IBM BASIC has many capabilities not contained in the above standards.

Required Publications

The *IBM BASIC/VM Program Directory*, which is provided with the distribution package, should be used in conjunction with this manual. The Program Directory provides detailed information about product requirements and installation procedures.

If you are planning to install the IBM BASIC Processor and Library in discontiguous saved segments, you should be familiar with the information in the following publications:

- *VM/SP Planning Guide and Reference*, SC19-6201
- *VM/SP System Programmer's Guide*, SC19-6203
- *VM/SP Installation Guide*, SC24-5237

Related Publications

You may also need to be familiar with the following publications:

- *IBM BASIC Programming Guide*, SC26-4027
- *IBM BASIC Language Reference*, GC26-4026

Summary of Amendments

IBM BASIC/VM Release 2, February 1986

Virtual Machine Storage Requirements

Virtual storage requirements have been updated for Release 2.

Auxiliary Storage Requirements

Auxiliary storage requirements have been updated for Release 2.

Using IBM BASIC under VM/PC

This section has been added to describe how to use IBM BASIC under VM/PC.

Default Option Values for the Processor

New ARITHMETIC, PRFNAME, and PRFTYPE options have been added. The PREC, COLLATE, and PRTZO options have been moved to this section. DMAXSTR has been renamed to STRMAX.

Default Option Values for the IBM BASIC/VM Product

This section has been added.

Preparing for BASIC Access to SQL/DS

This section has been added to provide information on accessing SQL/DS.

Determining Which Existing Intrinsic Functions can be Replaced

This section has been reorganized and updated to include real data type.

Appendix A

This appendix has been updated to include new module names.

Appendix B

This appendix has been updated to include new intrinsic functions.

Appendix C

This appendix has been updated to include real arguments and results.

Service Changes

Minor technical and editorial changes have been made.

IBM BASIC/VM Release 1.1

New Product Name

The product IBM BASIC has been renamed IBM BASIC/VM.

New Title

The title of this manual has been changed from *IBM BASIC Installation and Customization* to *IBM BASIC/VM Installation and Customization* to reflect the change.

Service Changes

Minor technical and editorial changes have been made.

Contents

A Brief Description of IBM BASIC under VM/SP CMS 1
The Processor 1
The Library 1
The HELP Panels 2

Installation Planning 3
Requirements for Installing IBM BASIC 3
Virtual Machine Storage Requirements 4
Auxiliary Storage Requirements 4
Choosing the Processor Option: SPEED versus SPACE 4
Choosing Default Option Values 5
Default Option Values for the Processor 5
Default Option Values for the Library 8
Default Option Values for the IBM BASIC/VM Product 9

Installing BASIC 11
The Installation EXEC Procedure 12
Optional Post-Install Step: Preparing for BASIC Access to SQL/DS 13
Mode of Operation 14
Installing BASIC in Discontiguous Saved Segments 15
Defining a Saved Segment for the Processor 16
Defining a Saved Segment for the Library 17
Loading and Saving BASIC Segments 18

Using IBM BASIC under VM/PC 21
VM/PC Processing Restrictions on IBM BASIC 22
Using NUCXLOAD With IBM BASIC 22
Downloading IBM BASIC to VM/PC 23

Customization 27
Changing Options 27
Creating or Changing the Processor Options File 27
Creating or Changing the Library Options File 28
Creating or Changing the Product Options File 28
Customizing The Reserved Word Table 29
Rules for Changing Reserved Words 29
Customizing Intrinsic Functions 29
Guidelines for Customizing Intrinsic Functions 30
Coding Your Replacement Function 31
Installing Your Replacement Function 31

Appendix A. IBM BASIC Processor and Library Module Names 33
Processor Modules Used for the SPACE Option: 33
Processor Modules Used for the SPEED Option: 33

Processor Modules 34
Library Modules 34

Appendix B. Intrinsic Functions 39
Intrinsic Functions and their Module Names 39
Intrinsic Function Modules 41

Appendix C. IBM BASIC Library: Argument List Formats 45
Argument and Result types 45
Integer Arguments and Results 45
Real Single Arguments and Results 45
Real Double Arguments and Results 45
Decimal Arguments and Results 46
Character Arguments and Results 47
Function Arguments 47
Function Results 48

Index 49

Figures

1. CMS Commands to Download IBM BASIC from the Host System to VM/PC 25

A Brief Description of IBM BASIC under VM/SP CMS

IBM BASIC/VM is a program product that consists of a Processor, a Library of functions, and a set of HELP panels.

IBM BASIC/VM runs under:

- Virtual Machine/System Product — Conversational Monitor System (VM/SP CMS) Release 3 or 4.

This program product is available on a single distribution tape.

The Processor

The IBM BASIC Processor processes programs written in the BASIC language either inside or outside the BASIC environment.

Inside the BASIC environment, the Processor provides an editor for creating and modifying programs, immediate syntax checking, and online error diagnostics. The Processor produces machine-language code packets that are executed under the control of the Processor on a statement-by-statement basis. Also, through the use of the COMPILE command, all facilities outside the BASIC environment are available from inside the BASIC environment.

When outside the BASIC environment, the Processor produces object modules that are suitable for subsequent execution inside or outside the BASIC environment.

For information on using the facilities of the Processor, see *IBM BASIC Programming Guide*.

The Library

The IBM BASIC Library contains numeric functions, advanced mathematical functions, character handling functions, input/output and error handling functions, and system functions such as time and date. The Library must be available in order to use the Processor and execute programs either inside or outside the BASIC environment.

For complete information regarding the contents and use of the Library, see *IBM BASIC Language Reference*.

The HELP Panels

The HELP command displays panels of information about any BASIC command, statement, intrinsic function, message, or error code. You can also use the HELP command for interactive assistance in learning and using BASIC.

The HELP panels are contained in a single file, BASHELP MACLIB.

Installation Planning

This section describes the machine configuration, virtual storage, auxiliary storage, operating systems, and programs required to install and run BASIC. It also describes the planning you should do before actually installing the product. You should:

- Ensure that you have the machine configuration, storage, and programs required by the product.
- Decide whether to install BASIC with the SPEED or the SPACE option.
- Decide whether to install BASIC in discontiguous saved segments.
- Decide whether to change the default values for the Processor, Library, or product options.
- Decide whether to customize the IBM BASIC reserved word table and/or the intrinsic functions.

Each item in the list is discussed in one of the sections that follow or in "Customization" on page 27.

Requirements for Installing IBM BASIC

The following are required to install the product:

- An IBM processor supported by the host operating system listed below.

  The processor must have the decimal and floating-point instruction sets.

- The current version of the following operating system:

  – VM/SP CMS — Virtual Machine/System Product, Release 3 or 4, unless otherwise stated.

- The distribution tape for the IBM BASIC/VM product.
- Space available on a system minidisk, program product minidisk, or other common minidisk for the Processor and the Library (see "Auxiliary Storage Requirements" on page 4).
- The VSE/VSAM program product, if VSAM files are to be processed (program number 5746-AM2).

The auxiliary storage requirements shown below for the Processor and Library can be met with any direct-access devices supported by the host operating system. This also applies to storage of user programs and data.

Virtual Machine Storage Requirements

The amount of virtual storage required to run the IBM BASIC Processor depends on whether the SPEED or the SPACE option is selected during installation.

The minimum virtual storage required to run BASIC is approximately:

- 525K-bytes if the SPACE option is selected
- 750K-bytes if the SPEED option is selected

This required virtual storage may be provided in shared storage (DCSS).

In addition, the virtual storage required to compose and run a normal 200-line BASIC program is approximately 40K-bytes for each user.

Because of storage requirements for the host operating system, the total virtual machine size required is larger than these BASIC sizes.

For more information on the SPEED and SPACE options, see "Choosing the Processor Option: SPEED versus SPACE," below.

Auxiliary Storage Requirements

To install and run BASIC requires auxiliary storage space equivalent to 30 cylinders on an IBM 3330 Disk Storage device. If you are not installing BASIC on an A-disk, you will need an A-disk of approximately 8 cylinders on an IBM 3330 Disk Storage device. These requirements can be met on any disk medium supported by the host operating system.

To install and run BASIC using FB-512 devices, such as IBM 3370 Direct Access Storage devices, requires approximately 6500 blocks of auxiliary storage, blocked at 1024 bytes per block.

See the *IBM BASIC/VM Program Directory* for a chart giving the equivalent storage requirements for other disk devices.

Choosing the Processor Option: SPEED versus SPACE

Before installation, you should decide whether you want a SPACE version of the BASIC Processor or a SPEED version.

- The SPACE option optimizes the amount of virtual storage required to run the Processor. This is the default.

- The SPEED option optimizes the instruction path lengths, and thus the performance, of the Processor.

Your installation determines which option is more suitable for the environment in which BASIC will be used. The SPEED option is generally more suitable for installations where storage and paging constraints are not severe.

Choosing Default Option Values

During the installation EXEC procedure, you will be asked to select either the default or alternative values for a number of IBM BASIC Processor, Library, and product options. The values you select serve as the "system defaults" for the BASIC user. Unless otherwise indicated, the defaults may be overridden by the user while programming.

The following section describes these options and indicates the distributed default values (referred to as the IBM-supplied default values in other manuals). To facilitate the actual installation process, your preinstallation planning should include a review of these options and a determination of the values most suitable for your organization. For more information, see "The Installation EXEC Procedure" on page 12.

You can also modify the default values after installation. For more information, see "Changing Options" on page 27.

Default Option Values for the Processor

The following list shows the Processor options and their meanings. The underlined value represents the default setting that will be used if you do not specify any alternative.

These options may be set either when the product is installed or after installation. For more information on how to do this, see "Creating or Changing the Processor Options File" on page 27.

ARITH = <u>DECIMAL</u> | NATIVE
determines the default numeric data type. When set to DECIMAL, the default numeric data type is decimal, and, when set to NATIVE, the default numeric data type is real.

If your installation runs a large number of mathematical applications, consider setting the ARITH option to NATIVE.

AUTOINC = <u>10</u> | n
specifies the default line number increment to be used when in AUTO line number prompting mode. n must be an integer between 1 and 9999998, inclusive.

AUTOSTR = <u>100</u> | n
specifies the default starting line number when in AUTO line number prompting mode. n must be an integer between 1 and 9999999, inclusive.

BASE = 0 | 1
specifies whether the lower bound for a dimension of an array will be zero or one.

COLL = NATIVE | STANDARD
specifies the character set to be used in processing the CHR$ and ORD functions, and in character comparison. NATIVE uses the EBCDIC character set; STANDARD uses ASCII.

COPYINC = 1 | n
specifies the default increment value to be used for lines copied to another position in the workspace (using the COPY command). **n** must be an integer between 1 and 9999998, inclusive.

FLAG = I | W | E | S
specifies the level of diagnostic messages to be written:

FLAG = I specifies that all diagnostic messages are to be written.

FLAG = W specifies that only warning, error, and severe error messages are to be written.

FLAG = E specifies that only error and severe error messages are to be written.

FLAG = S specifies that only severe error messages are to be written.

FLGFIPS = FIPS | NOFIPS
specifies whether or not to produce a diagnostic informational message for any statement that does not conform to the Federal Information Processing Standard for BASIC syntax. If FIPS is specified, the messages are printed whether or not FLAG=I is in effect.

LSTLINE = 133 | n
specifies the record length for LISTING files produced by the Processor. **n** must be between 81 and 32756, inclusive.

LSTPAGE = 60 | n
specifies the number of lines per page for LISTING files produced by the Processor. **n** must be less than 32767. This option may not be overridden by the user.

MSGE = TEXT | ALL
specifies whether to print only the text portion or the entire message (including message identifier) for E-level messages.

MSGI = TEXT | ALL
specifies whether to print only the text portion or the entire message (including the message identifier) for I-level messages.

MSGS = TEXT | ALL
specifies whether to print only the text portion or the entire message (including message identifier) for S-level messages.

MSGW = TEXT | ALL
specifies whether to print only the text portion or the entire message (including message identifier) for W-level messages.

OBJLIST = LIST | NOLIST
specifies whether or not to produce a listing of the generated machine instructions, consisting of assembler instruction mnemonics and symbols, while compiling a BASIC program.

OBJPROG = OBJECT | NOOBJECT
specifies whether or not to produce an object module when compiling a BASIC program.

PREC = SPREC | LPREC
specifies the maximum number of significant digits to be displayed when printing unformatted numeric values, and the precision for real data.

SPREC specifies up to 6 significant digits when printed by an unformatted PRINT statement. It also specifies that real data, which is defined by REAL statements or by the option ARITHMETIC NATIVE, will be real single.

LPREC specifies up to 12 significant digits when printed by an unformatted PRINT statement. It also specifies that real data, which is defined by REAL statements or by the option ARITHMETIC NATIVE, will be real double.

PRFNAME = BASPROF | name
specifies the default profile filename. The **name** is from 1 to 8 characters consisting of letters, digits, and the national characters $, #, and @.

PRFTYPE = BASPROF | type
specifies the default profile filetype. The **type** is from 1 to 8 characters consisting of letters, digits, and the national characters $, #, and @.

PRTZO = 20 | n
specifies the default print zone size for unformatted printing. The following rules apply:

- The maximum value of **n** you can specify is 255. The option is device dependent; you will receive an error message if you specify a value of **n** greater than the maximum character line length of the device to which you are formatting.

- The minimum value you can specify is determined by the default value of the PREC option, above. If the default PREC=SPREC is selected, the minimum value for PRTZO is 13. If PREC=LPREC is selected, PRTZO must be at least 19.

RENUINC = 10 | n
specifies the default line number increment to be used by the RENUMBER command. **n** must be an integer between 1 and 9999998, inclusive.

RENUSTR = 100 | n
specifies the default starting line number to be used by the RENUMBER command. **n** must be an integer between 1 and 9999999, inclusive.

SORLIST = SOURCE | NOSOURCE
specifies whether or not to produce a source listing when compiling a BASIC program.

SORXREF = XREF | NOXREF
specifies whether or not to produce a cross-reference table of variables, statement numbers, and statement labels when compiling a BASIC program.

STORMAP = MAP | NOMAP
specifies whether or not to produce, when compiling a BASIC program, a storage allocation map listing common allocation, locations of variables, and references to subprograms, IBM BASIC Library routines, and user-defined functions.

STRMAX = 18 | n
specifies the default maximum string length for string variables. **n** must be an integer between 1 and 32767, inclusive.

TERM = SCROLL | LINE
controls whether display terminal I/O is to be in scrolling mode or line mode.

Default Option Values for the Library

The following list shows the Library options and their meanings. The underlined value represents the default setting that will be used if no alternative is specified.

These options may be set either when the product is installed or after installation. For more information on how to do this, see "Creating or Changing the Library Options File" on page 28.

DRECSZE = 133 | n
specifies the default record size for display type files. **n** must be an integer between 1 and 32756, inclusive.

INVP = OFF | ON
specifies whether the USA-style digit separator (comma) and decimal (period) are to be used, or whether to "invert" (European-style) the meaning of these characters.

IRECSZE = 255 | n
specifies the default record size for internal type files. **n** must be an integer between 1 and 32756, inclusive.

MARGBOT = 60 | n
specifies the default number of lines between page breaks for display files (produced by the PRINT File statement). The value of MARGBOT is the default value for the MARGIN File statement. (The value of MARGBOT does not affect the default value for the PRINT statement, which displays

data at the terminal.) **n** must be an integer between 0 and 32767, inclusive. A value of 0 indicates that no page breaks are to occur.

NRECSZE = 255 | n
specifies the default record size for native type files. **n** must be an integer between 1 and 32756, inclusive.

RECORDS = V | F
specifies the default record format for nonrelative files produced by BASIC programs.

V specifies variable record format.

F specifies fixed record format.

Default Option Values for the IBM BASIC/VM Product

The following list shows the IBM BASIC/VM product options and their meanings. The underlined value represents the default setting that will be used if no alternative is specified.

These options may be set when the product is installed. All of these options (except for PRODOPT) may be changed after the product is installed by creating a Product Options file. For information on how to do this, see "Creating or Changing the Product Options File" on page 28. To change the PRODOPT option, the product must be reinstalled.

RESOPT = file-spec | 'BASRESW BASOPTS'
specifies the name of the reserved word file. The **file-spec** must be enclosed in apostrophes and include both a filename and a filetype. An optional filemode may be specified. If the filemode is omitted, the default, an asterisk (*), is used. This option may not be overridden by the user.

PROCOPT = file-spec | 'BASPROC BASOPTS'
specifies the name of the Processor options file. The **file-spec** must be enclosed in apostrophes and include both a filename and filetype. An optional filemode may be specified. If the filemode is omitted, the default, an asterisk (*), is used. This option may not be overridden by the user.

LIBOPT = file-spec | 'BASLIB BASOPTS'
specifies the name of the Library options file. The **file-spec** must be enclosed in apostrophes and include both a filename and a filetype. An optional filemode may be specified. If the filemode is omitted, the default, an asterisk (*), is used. This option may not be overridden by the user.

PRODOPT = file-spec | 'BASPROD BASOPTS'
specifies the name of the product options file. The **file-spec** must be enclosed in apostrophes and include both a filename and a filetype. An optional filemode may be specified. If the filemode is omitted, the default, an asterisk (*), is used. This option may not be overridden by the user and may not be specified in the product options file.

LANGS = (ENGLISH, file-spec | 'BASHELP MACLIB')

specifies the name of the MACLIB containing the HELP panels. The **file-spec** must be enclosed in apostrophes and include both a filename and a filetype. An optional filemode may be specified. If the filemode is omitted, the default, an asterisk (*), is used. This option may not be overridden by the user.

Installing BASIC

This section describes, in general, the BASIC installation process under VM/SP CMS. For more specific information, refer to the Program Directory shipped with the product.

For best results in installing BASIC, this manual and the Program Directory should be used together. In particular:

- Use this manual for early installation planning, preliminary auxiliary storage allocation, and for deciding on the appropriate installation options.
- Use the Program Directory at actual installation time for details such as distribution tape format, using the installation EXEC procedure, and related tasks.

To install BASIC, you must have a read/write "A" minidisk and adequate product work space (either on the A-disk or on a separate minidisk). Make sure that SET LDRTBLS 5 is in effect.

1. Log on to VM/SP CMS, then link to and access (in read/write mode) the minidisk that is to hold the product.
2. Attach a tape drive to your virtual machine (or have the operator do so) at virtual address 181, and mount the distribution tape.
3. Use the VMFPLC2 command to load the first file on the distribution tape onto the work minidisk. This file contains the installation EXEC, named I5668996, and a product identification file, I5668996 012008. (The installation tape is in VMFPLC2 format.)
4. Execute the installation EXEC procedure, which begins product installation.
5. Respond to the prompts from the EXEC procedure.

If you have IBM BASIC/VM Release 1 installed, be sure to install Release 2 on a different minidisk. If you have Release 1 help panels on a disk that is accessed when you use BASIC, erase them. (If you do not erase them, BASIC will display the Release 1 help panels instead of the Release 2 help panels.)

The Installation EXEC Procedure

The installation EXEC procedure for BASIC performs all the installation and customization functions described in this section and in the section "Customization" on page 27.

Because you may not want to perform all the installation and customization functions when first installing BASIC, you should retain the installation EXEC on disk so that optional functions such as creating saved segments and changing the name of the product option file may be performed (or repeated) at a later date.

When you invoke the installation exec, it will put up four prompts:

1. Do you want to print or look at the Memo to Users?
2. Can the installation exec use 5668 as a prefix for temporary files? If not, specify the prefix it should use.
3. What is the filemode of the BASIC disk?
4. Do you want to do a full installation, a partial installation, or a prebuilt installation?

The **Full** installation consists of an additional series of prompts asking if you want to do a particular step. This is useful if you wish to customize many options or if you do not like to use menus. You can also use this to see all the prompts—just answer "No" to all the questions.

The **Partial** installation consists of a "Partial installation panel" and three submenus:

1. Select space or speed or both
2. Customize (change options, reserved words or intrinsic functions)
3. Build BASIC and BASICRUN modules

These cover the same ground as in the full installation. The big difference here, is that you can just choose the steps you want to do and do them, instead of being asked if you want to do each step. As in the full installation, you can look at all the panels—just enter the number associated with "Exit" to get out of a particular panel.

The **Prebuilt** installation is the shortest and simplest. It loads the product from the tape onto your BASIC disk. On the tape there are "prebuilt" BASIC and BASICRUN modules. Use this option *only* if you do not want to change any options during this invocation of the installation exec.

In both the full and partial installations, the first question you will see is "Do you want to load the installation tape?" Answer "Yes" (or just press enter) if you have not loaded the tape (or run the "prebuilt" installation) to this disk before. Otherwise, answer "No."

If you select both space and speed options for BASIC, the space version of the module is called BASIC and the speed version is called BASICF. You *cannot* specify both space and speed if you wish to load BASIC in a shared segment (DCSS).

You can customize BASIC while you are installing it the first time, or you can do it later. To do it later, reexecute the installation exec, perform your customizations and build a new BASIC and/or BASICRUN module. You may find it easier to use the partial installation option to selectively choose which steps you want to perform.

You can also change the Processor, Library or product options at any time by creating or changing the file associated with each option. See "Changing Options" on page 25 for details on how to do this. You can change the reserved words table by editing the file associated with it.

When you build your BASIC and/or BASICRUN modules you have three choices:

1. Build complete BASIC and BASICRUN modules
2. Build BASIC and BASICRUN modules for dynamic loading of the Library
3. Load the Processor and Library in shared segments (DCSS)

These are explained in detail in "Mode of Operation" on page 14.

Remember that nothing is permanent until you build a BASIC and/or BASICRUN module. (If you edit the reserved word file, however, the changes are immediate. You do not have to build a new BASIC module to get them.) All the record keeping is done on temporary files, so that if you want to stop the installation, answering "Quit" to a question (or entering the option number on the panels associated with Exit) won't leave your product in an undetermined state.

Optional Post-Install Step: Preparing for BASIC Access to SQL/DS

The SQL/DS Database Administrator must:

1. Link and access the SQL/DS production disk.
2. Preprocess the BLIOC06 ASMSQL file supplied with BASIC using the SQL/DS SQLPREP EXEC.

   The purpose of this is to identify the program BLIOC06 to SQL/DS. BASIC does not need any output from SQLPREP because the BLIOC06 TEXT deck IBM supplies has already been preprocessed; therefore, there is no output to be assembled. The use of the SQLPREP EXEC and the SQL/DS EXECs mentioned here are documented in the *SQL/DS Application Programmer's Guide* (SH24-5018). An example invocation of SQLPREP for the BLIOC06 preprocess is:

```
SQLPREP ASM PP(NOPUNCH,NOPRINT,PREP=BLIOC06,
   USER=SQLDBA/SQLDBAPW) IN(BLIOC06 ASMSQL fm)
```

Use the SQLDBA connect id. The correct password for that connect id must be substituted for *SQLDBAPW*. You also must substitute the file mode letter for *fm*.

3. Authorize the users to run the program BLIOC06. The GRANT RUN command can be issued through the EXEC SQLDBSU or using ISQL. For example:

```
GRANT RUN ON BLIOC06 TO PUBLIC
```

4. Create DBSPACES and grant authority for their use as necessary.

Individual users who want to access SQL/DS must:

1. Have the authority to run BLIOC06 (see step 3 above), have at least CONNECT authority to the database you will be using, and have the authority to use any DBSPACES and tables you will be accessing.

2. Link and access the SQL/DS production disk (normally 195). If you will be using BASIC and SQL/DS together frequently, you may want to add this step to your PROFILE EXEC.

3. Run the SQLINIT EXEC to create the access module ARISRMBT on your A-disk. This module identifies the database to be accessed by your userid.

Mode of Operation

BASIC can run in one of several ways. You can build:

1. A complete BASIC module

2. A smaller BASIC module that dynamically loads its Library text files from the BASIC product disk

3. An even smaller BASIC module that loads its Library and Processor from Discontiguous Shared Segments (DCSS)

The **complete BASIC module** has all the Processor and Library parts included. All of this will be loaded as a unit in virtual storage when BASIC is invoked. If you build this type of BASIC module and you also plan to use BASICRUN, you will probably want to also build a complete BASICRUN module.

If you build a BASIC module for **dynamic Library loading**, those Library text files will be dynamically loaded from the BASIC product disk into virtual storage on an as-needed basis when BASIC is invoked. The module contains a small set of BASIC Library components and the complete Processor. If you build this kind of BASIC module and you plan to use BASICRUN, you will probably want to also build the complementary BASICRUN module that allows dynamic loading of the Library. Note that this arrangement may actually require more virtual storage than the complete BASIC and/or BASICRUN modules because of the manner in which BASIC allocates storage for its workspace. This is especially true when the virtual machine size is small (less than 1 megabyte, for example).

If you select the options to load the Processor and Library in **shared segments**, the installation exec will build minimal BASIC and BASICRUN modules and two shared segments, BASSEG (for the Processor) and BLISEG (for the Library). The option for the Processor builds the BASIC module and the option for the Library builds the BASICRUN module. The shared segment approach permits sharing of BASIC components among multiple simultaneous users. Before choosing these options, you should familiarize yourself with the section "Installing BASIC in Discontiguous Shared Segments" to assist you in predefining the shared segments in your VM/SP system configuration.

Installing BASIC in Discontiguous Saved Segments

This section discusses how you can generate the optional discontiguous saved segments for the IBM BASIC Processor and the Library. It assumes that you are already familiar with the concept of discontiguous saved segments in general, and with the NAMESYS macro and the system name table (module DMKSNT) in particular. These are explained in *VM/SP Planning Guide and Reference*.

Before attempting to install BASIC in discontiguous saved segments, you should make sure that you have read/write access to the CP-owned volume on which the segments will be written, and that your userid is authorized for privileged class E. You should also be familiar with the load requirements presented in *VM/SP System Programmers' Guide*.

The saved segments associated with BASIC are essentially similar to the other commonly used saved segments under VM/SP, such as CMSDOS and CMSBAM, in that you must define permanent space on a CP-owned volume where they may be saved. This space must be allocated as PERM with the CP format-allocate program (using the ALLOCATE function), and must be formatted with 4K-byte pages (the FORMAT function). Also, your installation must determine the virtual addresses at which these segments will be loaded. This determination will depend upon several factors peculiar to your installation:

1. The average maximum virtual machine size in which BASIC will be run.

2. The starting addresses and sizes of other saved segments that will be loaded simultaneously with BASIC (for example, the VSAM segments).

When planning your saved segments, keep in mind that all the Processor and Library components are fully reentrant, and therefore sharable by multiple, simultaneous users. However, in order to be used in "shared" mode, the virtual address at which the segments are loaded must not fall within the defined size of the virtual machine that is to share them, nor may they overlap any of the other segments that the virtual machine is concurrently sharing (such as CMSDOS). Refer to *VM/SP Planning Guide and Reference* and *VM/SP Installation Guide* for guidelines. If these guidelines are not followed, some or all of the performance and overhead improvements associated with the use of saved/shared segments will be lost. For a more detailed discussion of discontiguous saved segments, see *VM/SP System Programmers' Guide*.

Note: It is imperative that you review all the saved segments in use at your installation and choose the virtual storage addresses accordingly.

Defining a Saved Segment for the Processor

The following is the format of a NAMESYS macro definition for the BASSEG segment. The name BASSEG is fixed and must be used if a Processor segment is to be generated.

Note: If the SPACE option has been selected, the BASSEG segment requires 384K-bytes (six 64K-byte segments, or 96 pages) of virtual storage. If the SPEED option was selected, the BASSEG segment requires 512K-bytes (eight 64K-byte segments, or 128 pages) of virtual storage .

In the following sample NAMESYS macro, it has been assumed that the SPEED option is in effect:

label	NAMESYS	SYSSIZE=512K, SYSNAME=BASSEG, VSYSRES=, VSYSADR=IGNORE, SYSVOL=VMSRES, SYSCYL=nnn, SYSSTRT=(cc,p), SYSPGCT=128, SYSPGNM=(nn,nn,nn-nn), SYSHRSG=(s,s,...)

where:

label is any desired user label

SYSSIZE=512K
is the minimum amount of storage you must have to IPL a saved system. K must be specified.

Note: Although you must code the operand, it is ignored for saved segments.

SYSNAME=BASSEG
is the name of the IBM BASIC Processor segment. This name must be coded as is.

VSYSRES=
is a required keyword that is ignored for saved segments (that is, VSYSADR=IGNORE).

VSYSADR=IGNORE
indicates that this NAMESYS macro is defining a saved segment that does not require a virtual system residence volume.

SYSVOL=VMSRES
defines the CP-owned volume on which the saved segment is to be saved.

SYSCYL=nnn
is the cylinder address of the minidisk for the system to be saved. SYSBLOK=nnn is used instead of SYSCYL when using FB-512 devices.

SYSSTRT=(cc,p)
identifies the starting cylinder and page number on the above-named CP-owned volume at which the saved segment is to start. The specification shown is for a CKD device. For FB-512 devices, use SYSSTRT=**pppppp**, where **pppppp** defines the starting page.

SYSPGCT=128
is the total number of pages to be saved (that is, the total number of pages identified in the SYSPGNM operand). If the SPACE option is in effect, specify 96 pages.

SYSPGNM=(nn,nn,nn-nn)
are the numbers (in virtual storage) of the pages to be saved. Pages may be specified singly or in groups. For example, if pages 0, 4, and 10 through 13 are to be saved, specify SYSPGNM=(0,4,10-13). The total must be equal to SVSPGCT.

SYSHRSG=(s,s,...)
are the segment numbers designated as shared (numbered from zero up).

Defining a Saved Segment for the Library

The following is the format of a NAMESYS macro definition for the BLISEG saved segment. The name BLISEG is fixed and must be used if a saved segment for the Library is to be generated.

Note: The BLISEG saved segment is 384K-bytes (six 64K-byte segments, or 96 pages) in length. It is not affected by the choice of SPEED or SPACE.

label	NAMESYS	SYSSIZE=256K, SYSNAME=BLISEG, VSYSRES=, VSYSADR=IGNORE, SYSVOL=VMSRES, SYSCYL=nnn, SYSSTRT=(cc,p), SYSPGCT=96, SYSPGNM=(nn,nn,nn-nn), SYSHRSG=(s,s,...)

where:

label is any desired user label (but different from that chosen for other NAMESYS macros).

SYSNAME=BLISEG
is the name of the IBM BASIC Library segment. This name may not be changed.

All other parameters are as described for the BASSEG saved segment.

Loading and Saving BASIC Segments

Before either of the BASIC saved segments can be used, the routines comprising each segment must be loaded into virtual storage at the addresses chosen (when coding the NAMESYS macro) and then written out to the CP-owned volume on which they are to be saved. It is assumed at this point that the necessary changes have been made to your DMKSNT module, and that a new CP nucleus has been generated and loaded containing the NAMESYS entries.

Note: Because no checks are made for overflow, and adjacent minidisks to the saved-system area could be overwritten without warning, it is advisable to double check the saved-system area on the designated CP-owned volume to ensure that sufficient space for the segments is available. Remember that, when you allocate space on the CP-owned volume, you must allow one page more than you are actually saving by using the SYSPGCT parameter. CP uses this page for system information.

After you and/or your system programmer have generated the new CP system containing the appropriate NAMESYS entries, you are ready to save the segment(s).

1. Define your virtual machine size to be larger than the highest address occupied by either segment. Precise calculation is not necessary; you can use 6, 8, or even 16 megabytes to be sure.

2. Link and access all minidisks containing TEXT files.

3. Execute the installation EXEC, selecting the option "Load the BASIC Processor and/or Library to shared segments" (or select the shared segment option during the full install procedure).

   First, you will be asked to enter the starting address for the BASSEG (Processor) segment. Enter the virtual storage address, in hexadecimal, at which the segment is to begin. You can calculate this address by multiplying the starting page number (the first value coded in the SYSPGNM operand of the NAMESYS macro for BASSEG) by 4096, and converting the result to hexadecimal.

   Next, the procedure checks that the address specified is greater than or equal to X'20000' and less than 16 megabytes. It also checks that only valid characters are specified. If you do not have an A-disk accessed, the following message:

   ```
   *** No read/write A-disk accessed
   ```

   is displayed, and processing is terminated.

   All the TEXT files needed to create the Processor segment are loaded, starting at the address specified. If there are any unresolved external references, the EXEC terminates with return code 66 and an appropriate message.

If the TEXT files are loaded without errors, then the storage keys of all the pages to be saved are set to 13 (X'D'). If errors are encountered, the following message:

```
*** BASGEN failed due to SETKEY error.
```

is displayed, and processing is terminated with error code 100.

Otherwise, the virtual storage pages specified in the NAMESYS macro are written to disk. If errors are encountered, the following message:

```
*** BASGEN failed due to SAVESYS error.
```

is displayed and processing is terminated. Otherwise, the segment has been successfully saved, the load map is saved on the BASIC disk, and the completion message:

```
*** BASGEN complete -- 'BASSEG LOADMAP A' created.
```

is displayed.

4. To create a saved segment for the IBM BASIC Library, first make sure you have completed items 1 through 3 above. Then execute the installation EXEC procedure, selecting the "Load the BASIC Library to a shared segment (BLISEG DCSS)" option from the "Loading Shared Segments" panel.

   During the BLISEG generation process, the EXEC procedure will perform the same validation, loading, and saving of the IBM BASIC Library components as described above for the BASSEG generation procedure.

5. If you intend to run BASIC using the shared segments, define your virtual machine to be smaller than the lowest address occupied by either segment.

Using IBM BASIC under VM/PC

IBM Virtual Machine/Personal Computer (VM/PC) is an IBM licensed program that runs on the IBM Personal Computer XT/370 or AT/370. VM/PC gives you an interactive system that has the characteristics of a VM/SP system.

There are three different methods you can follow to use IBM BASIC under VM/PC. (You may want your system administrator to do these tasks for you.)

- Create the BASIC module on the VM/PC. Then load the module into a nucleus extension with the NUCXLOAD command. Creating the module is necessary only when you first access BASIC, or after a new release has been installed on the host system. However, you must issue the NUCXLOAD command after every Initial Program Load (IPL) of CMS.

- Copy (download) the BASIC module onto local disk files and then invoke BASIC in local sessions. You need to download only when you first access BASIC, or when a new release has been installed on the host system.

- Link to the host-system minidisk containing BASIC and then access BASIC from the local session as a remote minidisk. You must do this after every Initial Program Load (IPL) of CMS, and whenever the link to the host system fails.

  Depending on your link with the system and on the system load, this often is not an efficient way to operate. Therefore, it is not further described in this section.

You can then invoke the Processor either inside or outside the BASIC environment.

Note: If you frequently get a message telling you that the loader table has overflowed, you should add the following command to your PROFILE EXEC procedure:

```
SET LDRTBLS 5
```

This command should correct the problem.

VM/PC Processing Restrictions on IBM BASIC

The following processing capabilities are ***not*** available when you are executing an object program in a local session:

- VSAM file processing
- Magnetic tape processing
- The Graphical Data Display Manager (GDDM)
- SQL/DS

Using NUCXLOAD With IBM BASIC

You can decrease the processing time needed to access BASIC repeatedly by installing BASIC as a nucleus extension.

You must first create the BASIC and/or BASICRUN modules on your VM/PC system. The BASIC module contains all the Processor and Library components for interactive and batch program development and execution. The BASICRUN module contains the components needed to execute a compiled BASIC program outside the BASIC environment.

After you have created these modules, you can upload them into the host system; they can then be downloaded to other VM/PC stations.

To create the modules, you should follow these steps (you may be able to have your system administrator do this for you):

1. Make sure you have the BASIC text files available on the host system. You may need to read them from the installation tape and copy them onto a CMS minidisk.

2. Modify the following line from the BGEN EXEC on the installation tape. Add the RLDSAVE option at the end of the following LOAD command:

   ```
   LOAD BLIOCRT (CLEAR NOAUTO NOLIBE RESET BLIOCRT
   ```

   as follows:

   ```
   LOAD BLIOCRT (CLEAR NOAUTO NOLIBE RESET BLIOCRT RLDSAVE
   ```

3. Create the modules during your VM/PC local session by entering one or both of the following:

   ```
   BGEN (ALL
   ```

   This command creates BASIC MODULE A.

   or

   ```
   BGEN L (ALL
   ```

   This command creates BASICRUN MODULE A.

For more information, see documentation on the LOAD, INCLUDE, NUCXLOAD, NUCXDROP, and NUCXMAP commands in the *VM/PC User's Guide*.

After you have created these modules, you can upload them into the host system; they can then be downloaded to other VM/PC stations. You may want to rename these modules first; this prevents inadvertent overwriting of the host system disks.

After the BASIC and/or BASICRUN module has been created, you can load them into your nucleus by entering the following commands:

```
NUCXLOAD BASIC
```

or

```
NUCXLOAD BASICRUN
```

You must issue these commands each time you Initial Program Load (IPL) CMS. You can put these commands into your PROFILE EXEC, which will issue them for you.

After this procedure is completed, you can invoke BASIC by entering:

```
BASIC
```

and the BASIC interactive Processor is ready for your use.

Downloading IBM BASIC to VM/PC

First, check with your system administrator to find out how the IBM BASIC Processor is stored on the host system:

1. If the Processor is a single load module, you can immediately download the Processor as described in the following paragraphs.

   For VM/PC usage, it is usually most efficient to use this module, which is created with the SPACE installation option. If BASIC was installed using the SPEED option, see the next step.

2. If the Processor is a group of text files, or is a combination of text files and load modules, or was installed with the SPEED option, ask your system

administrator to create a Processor in a single load module with the SPACE option.

3. You can then download BASIC, using the commands shown in Figure 1 on page 25. The procedure is as follows:

   a. Link (if necessary) and access the local minidisk that is the target minidisk for the copy operation. If the target minidisk is your own minidisk, the link is not required.

   b. Link and access the host minidisk that contains the BASIC modules.

   c. Copy the BASIC modules from the host minidisk to the local minidisk. (This is known as downloading.) If you are using the BASIC reserved word file, be sure to copy it. The default name is BASRESW BASOPTS. Be sure to copy any other BASIC options file you may be using.

   d. Release the host BASIC minidisk; it is no longer required.

Downloading is necessary only when you first want to access BASIC, or after a new release has been installed on the host system.

The BASIC module must contain all of the Processor and Library text files.

If you only want to execute BASIC programs from previously compiled text, you only need to copy the BASICRUN module. However, if you want to develop and process BASIC programs completely, you should copy both the BASIC and BASICRUN modules.

If space is available, you can also download the BASIC HELP panels into local storage. If you want to download them, copy the file BASHELP MACLIB.

There are approximately 1200 HELP panels, and you can choose to download any or all of them. For example, you might want to download only those files associated with BASIC messages. (The filenames of BASIC message HELP panels all begin with the prefix BAS or BLI, followed by digits. For example, BAS00034 BASHELP.) Use the CMS MOVEFILE command to move a panel from the maclib. You can either create a new maclib and download that or create individual files with filetype BASHELP. See *VM/SP CMS Command and Macro Reference* for information on the CMS commands MACLIB and MOVEFILE.

HELP panels containing syntax information corresponding to that in the *IBM BASIC Language Reference* manual are identified with filenames of the statements and commands. (For example, the OPEN statement file is BASHOPEN BASHELP.)

If you don't have the space available to download the HELP panels, you can access them for your BASIC session by linking to the appropriate host system disk.

Virtual Storage Requirements: 525K-bytes

Minidisk Storage Requirements: BASIC requires the following minidisk storage:

- BASIC module: approximately 525 1024-byte blocks

- BASICRUN module: approximately 300 1024-byte blocks
- BASHELP MACLIB: approximately 1940 1024-byte blocks

Note: These storage requirements are for the BASIC modules only; additional storage is needed for the source and/or object program files.

1) Link and access the target minidisk.

```
CP LINK vm/pc-id ttt aaa W write-password
ACCESS aaa filemode1
```

2) Link and access the host minidisk that contains the BASIC modules.

```
CP LINK host-id hhh bbb RR read-password REMOTE
ACCESS bbb filemode2
```

3) Copy the files you want (one or more of the following).

```
COPYFILE BASIC MODULE filemode2 = = filemode1
COPYFILE BASICRUN MODULE filemode2 = = filemode1
COPYFILE BASHELP MACLIB filemode2 = = filemode1
```

(If you are using the reserved word file, be sure to copy it. The default name, shown below, is BASRESW BASOPTS.)

```
COPYFILE BASRESW BASOPTS filemode2 = = filemode1
```

4) Release the host BASIC minidisk.

```
RELEASE filemode2 (DET
```

Where:

ttt	is the virtual address of the local target minidisk that will store the BASIC modules.
aaa	is an unused virtual address on the local VM/PC machine.
hhh	is the virtual address of the host minidisk that contains the BASIC modules.
bbb	is the virtual disk address you use to refer to the host disk.
filemode1	is the filemode of the local VM machine.
filemode2	is the filemode of the host minidisk that contains the BASIC modules.

Figure 1. CMS Commands to Download IBM BASIC from the Host System to VM/PC

Customization

This section describes how to customize the BASIC features to make the product more suitable for your installation. The features you can customize are:

- The Processor, Library, and product options
- The reserved word table, which you can customize by adding and replacing entries
- Some of the intrinsic functions, which you can customize by replacing their processing algorithms

You can change most options and the reserved word table after installing the product. The intrinsic functions can be replaced only during the installation procedure.

Changing Options

All the Processor options, all the Library options, and most of the product options can be changed easily by editing the options files.

Creating or Changing the Processor Options File

You can use the Processor options file to specify different values for the Processor options after linking the product. The options file is not supplied as part of the product and is not needed if the default options set at installation are satisfactory.

The file should be a sequential file. Each record in the file will be the option keyword (in column 1) followed by spaces (it may have an optional equal sign) then the option value. Only one option per record is allowed. A record beginning with an asterisk (*) in column 1 is ignored and may be used for comments. If the same option keyword is used more than once, the value for the last option keyword will be used. An error will occur if an unrecognized option keyword or an invalid value is found. The message includes the record number of the record in error.

The IBM-supplied default file specification for the Processor options file is BASPROC BASOPTS *.

For a list of the options that you may set, see "Default Option Values for the Processor" on page 5.

Creating or Changing the Library Options File

You can use the Library options file to specify different values for the Library options after linking the product. The options file is not supplied as part of the product and is not needed if the default options set at installation are satisfactory.

The file should be a sequential file. Each record in the file will be the option keyword (in column 1), followed by spaces (the equal sign is optional), then the option value. Only one option per record is allowed. A record beginning with an asterisk (*) in column one is ignored and may be used for comments. If the same option keyword is used more than once, the value for the last option keyword will be used, unless otherwise noted. An error will occur if an unrecognized option keyword or an invalid value is found. The message includes the record number of the record in error.

The IBM-supplied default file specification for the library options file is BASLIB BASOPTS *.

For a list of the options, see "Default Option Values for the Library" on page 8.

Creating or Changing the Product Options File

You can use the product options file to specify different values for the product options after linking the product. The options file is not supplied as part of the product and is not needed if the default options are satisfactory.

The options file should be a sequential file. Each record in the file will be the option keyword (in column 1), followed by spaces (the equal sign is optional), then the option value. Only one option per record is allowed. A record beginning with an asterisk (*) in column one is ignored and may be used for comments. If the same option keyword is used more than once, the value for the last option keyword will be used, unless otherwise noted. An error will occur if an unrecognized option keyword or an invalid value is found. The message includes the record number of the record in error.

The default file specification of the product options file is BASPROD BASOPTS *.

For a list of the options, see "Default Option Values for the IBM BASIC/VM Product" on page 9. All of those options may be specified in the file except for PRODOPT and LANGS.

The following option may be specified in the product options file to override the LANGS option:

```
LANGUAGE [=] ENGLISH,file-spec
```

This specifies the file-spec for the MACLIB containing the HELP panels. The **file-spec** must be enclosed in quotation marks or apostrophes and must include both a filename and filetype. An optional filemode may be specified. If the filemode is omitted, the default, an asterisk (*), is used.

Customizing The Reserved Word Table

In the BASIC product as distributed, all the language keywords (such as FOR, LET, and PRINT), as well as the supplied function names (such as SIN, COS, TIME, and DATE), are reserved words. As such, these words may not be used as variable names, user-defined function names, statement labels, or subprogram names. Any such use of a reserved word will cause a Processor diagnostic message to be issued.

Although this feature provides valuable assistance to the user in producing syntactically correct programs, there may be instances (for example, converting BASIC programs written on other systems) when you may want to permit certain words to be used as user-defined variables, subprogram names, or statement labels. If this is the case, the word(s) in question may be removed from the Processor's reserved word table. Thereafter, the IBM BASIC Processor will continue to recognize the word in its language-keyword context (based upon where it appears in a statement), and will also permit its use (without producing a diagnostic message) as a variable, label, function, or subprogram name.

Similarly, your installation may want to prevent the use of certain words in the BASIC language. This can be accomplished by adding these words to the reserved word table.

For more information on reserved words, see *IBM BASIC Language Reference.*

Rules for Changing Reserved Words

You can make the necessary changes to the reserved word table yourself by editing the reserved word file. A default reserved word table is supplied as part of the product. The file should be a sequential file. Reserved words may be in any order, with as many on a record as desired, separated by spaces. A word may appear more than once. To make a comment line, place an asterisk (*) in column one. A reserved word begins with a letter followed by letters, digits, and underscores with the last character optionally a percent sign (%), a number sign (#), or a dollar sign ($). To change the reserved word list, you can add or delete words. Sequence numbers beginning with a digit in columns 73 through 80 are ignored (column 72 must be blank). If the reserved word file is deleted or inaccessible, BASIC will assume there are no reserved words (there is no message in this case).

The IBM-supplied default file specification is:

```
BASRESW BASOPTS *
```

Customizing Intrinsic Functions

BASIC provides a comprehensive set of numeric and character intrinsic functions to assist the programmer. It is, however, possible that your installation may need or want to replace an existing function's processing algorithm or definition.

You must be extremely careful in doing this, however, and follow all the guidelines and restrictions outlined in this section. Failure to do so could compromise the operational integrity of the entire BASIC product.

For a list of the intrinsic functions supplied with the product, see Appendix B, "Intrinsic Functions" on page 39.

See *IBM BASIC Language Reference* for an explanation of each function.

Guidelines for Customizing Intrinsic Functions

The following is a list of guidelines for customizing BASIC intrinsic functions:

1. Not all functions can be replaced; only those designated in Appendix B, "Intrinsic Functions" on page 39, may be replaced.

2. All functions must be coded in Assembler language. IBM-supplied function routines conform to all standard IBM coding conventions regarding parameter lists, register usage, and save areas; your routine must do the same. For detailed information regarding argument list formats for intrinsic functions, see Appendix C, "IBM BASIC Library: Argument List Formats" on page 45.

3. The number and type of the arguments passed to and from the function may not be altered. The allowable input argument and result types are given in Appendix B, "Intrinsic Functions" on page 39.

4. Existing supplied functions in BASIC may not be deleted, nor may their names be changed. You may replace the processing algorithm for an existing function only.

5. The function processing routine must have no "side effects". That is, it must not modify any input arguments.

6. If the standard BASIC shared segments are being used, all function routines must be fully reentrant. They must be assembled with the RENT option, and may not modify instructions or data within themselves or in other parts of the Processor or Library.

7. You must name the module for your user-written function with the same name as the module it replaces.

8. Your user-written module must have the same entry point(s), and the corresponding code for each entry point, as the intrinsic function it replaces. For example, if you want to replace the integer version of the function FP, you must also replace the integer version of the function IP because they are in the same module.

9. If your user-written module has more than one entry point, the module must begin with a table of fullword addresses. These are the addresses of the entry points in the module. The addresses must be in the sequence given in "Intrinsic Function Modules" on page 41.

For example, the module BLICCAS should have the following format:

```
         ENTRY BLICLWR
         ENTRY BLICUPC
BLICCAS  CSECT
         DC    A(BLICLWR)
         DC    A(BLICUPC)
BLICLWR  DS    0H
(code for LWRC$ function)
BLICUPC  DS    0H
(code for UPRC$ function)
         END
```

10. If your replacement module has a single entry point (that is, the name of the module is the same as the entry point), the entry point must be at the first address in the module. For example, the module BLINORD should have the following format:

```
         ENTRY  BLINORD
BLINORD  CSECT
(code for ORD function)
         END
```

Coding Your Replacement Function

First, determine the filename of the functional routine you want to replace. The filename has the form BLI*xyyy*, where *x* identifies the general class of the function (that is, N for numeric functions or C for character) and *yyy* is a descriptive mnemonic for the particular function. The filetype is TEXT. For a complete list of IBM BASIC Library routine names, see Appendix B, "Intrinsic Functions" on page 39.

Next, code your new routine with the same name as the original, referring to Appendix C, "IBM BASIC Library: Argument List Formats" on page 45, for details on the format of the parameters. When tested and ready to install, refer to "Installing Your Replacement Function," following.

Installing Your Replacement Function

When your function routine is ready to install, execute the installation EXEC procedure, selecting the "Replace an Intrinsic Function" option, or by replying YES to the prompt during the full installation procedure. The installation EXEC will copy your routine's TEXT file from your work disk to the BASIC disk and rename it if necessary.

Then, if you are using the BLISEG shared segment, resave the segment, using the procedure described under "Installing BASIC in Discontiguous Saved Segments" on page 15 or, if you are using a module that contains the Library function, it should be rebuilt to include the new routine.

Appendix A. IBM BASIC Processor and Library Module Names

Processor Modules Used for the SPACE Option:

Module Name	Description
BASBBGN	Object Code Instruction Builder
BASBEGN	Expression Code Generator
BASBESN	Expression Scanner & Syntax Checker
BASBGNP	Code Generation Prelude
BASBPOP	Pop Interpreter
BASBSGN	Statement Code Generator
BASBSSN	Statement Scanner
BASBSUM	Summary Lister
BASBXRF	Cross Reference Routine
BASICMD	Command Processor
BASIEXM	Interactive Execution Monitor
BASIHLP	Help Processor
BASIWKS	Workspace Editor

Processor Modules Used for the SPEED Option:

Module Name	Description
BASFBGN	Object Code Instruction Builder
BASFEGN	Expression Code Generator
BASFESN	Expression Scanner & Syntax Checker
BASFGNP	Code Generation Prelude
BASFPOP	Pop Interpreter
BASFSGN	Statement Code Generator
BASFSSN	Statement Scanner
BASFSUM	Summary Lister
BASFXRF	Cross Reference Routine
BASFCMD	Command Processor
BASFEXM	Interactive Execution Monitor
BASFHLP	Help Processor
BASFWKS	Workspace Editor

Processor Modules

Module Name	Description
BASBDIR	Processor Module Directory
BASBEMT	Error Message Tables
BASBFRL	Fixed Roll Tables
BASBKWT	Keyword Table
BASBML	Pops Language Utilities
BASBRCD	Roll Code Tables
BASBSYS	Processor Options
BASICKT	Command Keyword Table
BASIHML	Help Script
BASIHMT	Help Mapping Table

Library Modules

Module Name	Description
BLIAASN	Array Assignment
BLIACAT	Concatenate Character Arrays
BLIACIX	Character Array Index Sort
BLIACSC	Concatenate Scalar and Character Array
BLIACSS	Character Array Sort
BLIADAA	Decimal Array Addition and Subtraction
BLIADCN	Decimal Array Constant Function
BLIADDO	Decimal Dot Product Function
BLIADID	Decimal Identity Array
BLIADIN	Inverse and/or Determinant of a Matrix
BLIADIX	Decimal Array Index Sort
BLIADML	Decimal Array Multiplication
BLIADPS	Decimal Array Product and Sum
BLIADSS	Decimal Array Sort
BLIADZR	Decimal Array Zero
BLIAICN	Integer Array Constant
BLIAIID	Integer Identity Array
BLIAIIX	Integer Array Index Sort
BLIAIPS	Integer Array Product and Sum
BLIAISS	Integer Array Sort
BLIAIZR	Integer Array Zero
BLIAMFN	MAT Scalar Functions
BLIANUL	Null Array
BLIARCN	Decimal Array Constant
BLIARDM	Copy and Redimension Array
BLIARID	Real Identity Array
BLIARIX	Real Array Index Sort
BLIARPS	Real Array Product and Sum
BLIARSS	Real Array Sort
BLIARZR	Real Array Zero
BLIASAM	Decimal and Integer Array Multiplication by Scalar

BLIASRC	Array Search
BLIATRN	Array Transpose
BLIAUSZ	Array Size and Upper Limit
BLICCAS	Lowercase and Uppercase Routines
BLICCHR	Convert to Character (Reverse Ord)
BLICNST	Convert Numeric to Character
BLICPAD	Pad Left and Pad Right Routines
BLICPRM	PARM$ Function
BLICRPT	Repeat Character String M Times
BLICSRE	String Replacement
BLICTRM	Remove Leading Spaces and Remove Following Spaces
BLIDIRA	Array Routines Directory
BLIDIRC	Character Routines Directory
BLIDIRI	I/O Routines Directory
BLIDIRM	Math Routines Directory
BLIDIRN	Numeric Routines Directory
BLIDIRO	Submonitor Routines Directory
BLIDIRS	Misc Routines Directory
BLIDIRX	Library Directory
BLIDIRZ	Root Directory
BLIDIR2	Release 2 Routines Directory
BLIFLGC	Compiler Flag
BLIFLGI	Interactive Flag
BLIICL	Close File
BLIICNV	Form, Image, and PIC Processor
BLIIDEL	Delete File
BLIIEND	End I/O List Processor
BLIIFNC	File Information
BLIIGPT	GET and PUT Statements
BLIIIN	Input from File or Terminal
BLIIINT	Preview Scalar Input List
BLIIIOL	Transfer Elements
BLIIMAR	MARGIN Statement
BLIIOPN	OPEN Statement
BLIIPOS	File Positioning Operations
BLIIPR	Print Operations (File or Terminal)
BLIIRD	Read External File
BLIIRER	REREAD Statement
BLIIREW	REWRITE Statement
BLIIRIF	Read Internal File
BLIIRS	RESET Statement
BLIISCR	SCRATCH Statement
BLIITAB	Tab Processing
BLIIVAL	File I/O Utility
BLIIWRT	WRITE Statement
BLIMCDI	Decimal to Integer Conversion
BLIMCDR	Decimal to Real Conversion
BLIMCID	Integer to Decimal Conversion
BLIMCIR	Integer to Real Conversion
BLIMCRD	Real to Decimal Conversion
BLIMCRI	Real to Integer Conversion
BLIMDAS	Decimal Addition and Subtraction
BLIMDC	Decimal Comparison
BLIMDD	Decimal Division
BLIMDDP	Decimal to Decimal Involution

BLIMDIP	Decimal to Integer Involution
BLIMDM	Decimal Multiplication
BLIMDST	Decimal Round and Store
BLIMRPI	Real to Integer Involution
BLIMRPR	Real to Real Involution
BLINDAA	Arccos and Arcsin for Decimal
BLINDAB	ABS for Decimal
BLINDAT	Angle and Arctan for Decimal
BLINDBL	Convert to Double
BLINDCS	Decimal Constants: EPS, INF, and PI
BLINDDG	Centigrade and Fahrenheit Conversion for Decimal
BLINDEC	Convert to Decimal
BLINDFI	FP and IP for Decimal
BLINDHH	Hyperbolic Cos and Hyperbolic Sin
BLINDII	Ceiling and Integer Functions for Decimal
BLINDRR	Degrees and Radians
BLINDRT	Round and Truncate
BLINDSG	Sign of X, Decimal
BLINDSS	Cos, Sin, Csc, and Sec
BLINDTC	Cot and Tan
BLINEXP	EXP
BLINFIX	Rounded Integer Value
BLINIAB	ABS for Integer
BLINIFI	FP and IP for Integer
BLINIII	Ceiling and Integer Functions for Integer
BLINISG	Sign of X, Integer
BLINLEN	Length of Character String
BLINLG2	LOG2
BLINLNE	Natural Log Base E (LOG)
BLINLOG	LOG10
BLINMRM	Modulo and Remainder
BLINMXN	Maximum and Minimum
BLINORD	Convert Char Variable to Number (ORD)
BLINPOS	Position of Y in X, after Position M
BLINRAB	ABS for Real
BLINRFI	FP and IP for Real
BLINRII	CEIL and INT for Real
BLINRSG	SGN for Real
BLINSNG	Convert to Single
BLINSQR	Square Root of X, Decimal
BLINTNH	TANH Function
BLINVAL	VAL Function
BLIOCOP	Product Options
BLIOCRT	Processor Root Module
BLIOC00	BASIC Initialization
BLIOC01	Exit - RETURN to Operating System
BLIOC02	Scan Parameter List
BLIOC03	Load an Object Program
BLIOC04	Remove Object Program from Storage
BLIOC05	Set Language
BLIOC06	SQL Support
BLIOC07	Submonitor Utility Routines for CALL SQL
BLIOC10	Open a BASIC File
BLIOC11	Read a BASIC File
BLIOC12	Write a BASIC File

BLIOC13	Rewrite a BASIC File
BLIOC14	Print a Line in BASIC File
BLIOC15	Delete a Record
BLIOC16	Set Read or Write Pointer
BLIOC17	Close a BASIC File
BLIOC18	Empty a BASIC File
BLIOC19	Erase a BASIC File
BLIOC20	Terminal/CRT Prompt
BLIOC21	Terminal/CRT Display
BLIOC22	Restore Saved CRT Buffer
BLIOC23	Obtain Virtual Storage
BLIOC24	Release Virtual Storage
BLIOC25	Obtain Large Block of Storage
BLIOC26	Return Date
BLIOC27	Return Time of Day
BLIOC28	Set Program Interrupt Exit
BLIOC29	Terminal Type and Features
BLIOC30	Issue Operating System Command
BLIOC33	Save Current CRT Buffer
BLIOC34	Query Files Routine
BLIOC35	Enter CMS Subset
BLIOC36	Read Specified CRT Field
BLIOC37	Write Specified CRT Field
BLIOC51	Open a CMS File
BLIOC52	Open an OS File
BLIOC53	Open a VSAM File
BLIOC54	Close a CMS File
BLIOC55	Close an OS File
BLIOC56	Close a VSAM File
BLIOC57	Read a CMS File
BLIOC58	Read an OS File
BLIOC59	Read a VSAM File
BLIOC60	Write a CMS File
BLIOC61	Write an OS File
BLIOC62	Write a VSAM File
BLIOC63	Rewrite a CMS File
BLIOC65	Rewrite a VSAM File
BLIOC66	Point a CMS File
BLIOC67	Point a tape File
BLIOC68	Point a VSAM File
BLIOC69	Delete A CMS Record
BLIOC71	Delete A VSAM Record
BLIOC72	Locate a File's Control Block
BLIOC74	Process File Specification
BLIOC76	Purge CMS File
BLIOC79	Scratch CMS File
BLIOC81	Scratch VSAM File
BLIOC86	Set Terminal Characteristics
BLIOC93	SQL/DS Error Message Generator
BLISBAS	CALL BASIC Statement
BLISBRK	BREAK Statement
BLISCLK	Date and Time Utility
BLISCRC	CONTINUE and RETRY Statements
BLISCSY	CALL SYSTEM
BLISCTL	GOTO, GOSUB, Function and Subroutine Entry/Exit

BLISEMC	Library Message Codes
BLISEMT	Library Messages
BLISERT	Error Processor
BLISGDD	GDDM Interface
BLISILC	Inter-Language Call Interface
BLISINT	Initialize Main Program and Subprogram
BLISPAU	PAUSE Statement
BLISRND	Generate Pseudo-random Number and Seed
BLISSCR	Compute Array Element Address
BLISSQF	Call SQL FETCH
BLISSQL	CALL SQL Statement
BLISSQM	Call SQL Memory Management
BLISSQO	CALL SQL Operations
BLISSQP	Call SQL PREPARE
BLISSQS	CALL SQL Statement Block Handling
BLISSQT	Call SQL Trace Modules
BLISSQU	CALL SQL Utility Routines
BLISSQV	CALL SQL Data Conversion
BLISSTR	String Concatenation, Compare, Replace, Extract, Copy
BLISSYS	Library Options
BLISTMM	Get and Return Free Space
BLISTRC	Trace and Debug Control
BLISTRM	CHAIN, END, and STOP Statements

Appendix B. Intrinsic Functions

Some of the intrinsic functions shipped with the IBM BASIC product can be replaced with user-written functions.

In order to replace a function, you must follow the rules given in "Guidelines for Customizing Intrinsic Functions" on page 30.

This appendix contains the following two lists:

- Intrinsic functions and their module names
- Module names, with their entry points, argument types, and result types

To use these lists, look up the function you are interested in replacing, in the first list. If the function can be replaced, then look up the module(s) for the function in the second list.

Intrinsic Functions and their Module Names

Functions that may ***not*** be replaced are marked with an asterisk (*).

Note: Where more than one module name is shown, the fifth character indicates the data type of the function:

D for decimal

I for integer

R for real (single and double)

	Function	Module Name(s)
	ABS	BLINIAB, BLINDAB, BLINRAB
	ACOS	BLINDAA
*	ANGLE	BLINDAT
	ASIN	BLINDAA
*	ATN	BLINDAT
*	CEIL	BLINIII, BLINDII, BLINRII
	CEN	BLINDDG
	CHR$	BLICCHR
*	CNT	BLIIFNC
*	CODE	BLISERT
	COS	BLINDSS

COSH	BLINDHH
COT	BLINDTC
CSC	BLINDSS
DAT$	BLISCLK
DATE	BLISCLK
DATE$	BLISCLK
* DBL	BLINDBL
* DEC	BLINDEC
DEG	BLINDRR
* DET	BLIADIN
* DOT	BLIADDO
* EPS	BLINDCS
* ERR	BLISERT
* EXP	BLINEXP
FAH	BLINDDG
* FILE	BLIIFNC
* FILENUM	BLIIFNC
* FILE$	BLIIFNC
* FP	BLINIFI, BLINDFI, BLINRFI
* IFIX	BLINFIX
* INF	BLINDCS
* INT	BLINIII, BLINDII, BLINRII
* IP	BLINIFI, BLINDFI, BLINRFI
JDY	BLISCLK
* KEYNUM	BLISERT
* KLN	BLIIFNC
* KPS	BLIIFNC
LEN	BLINLEN
* LINE	BLISERT
* LOG	BLINLNE
LOG10	BLINLOG
LOG2	BLINLG2
LPAD$	BLICPAD
LTRM$	BLICTRM
LWRC$	BLICCAS
* MAX	BLINMXN
* MIN	BLINMXN
* MOD	BLINMRM
ORD	BLINORD
PARM$	BLICPRM
* PI	BLINDCS
POS	BLINPOS
* PRD	BLIAIPS, BLIADPS, BLIARPS
RAD	BLINDRR
* REAL	BLINDBL, BLINSNG
* REC	BLIIFNC
* REM	BLINMRM
* RLN	BLIIFNC
* RND	BLISRND
ROUND	BLINDRT
RPAD$	BLICPAD
RPT$	BLICRPT
RTRM$	BLICTRM
SEC	BLINDSS
SGN	BLINISG, BLINDSG, BLINRSG

	SIN	BLINDSS
	SINH	BLINDHH
*	SIZE	BLIAUSZ
*	SNG	BLINSNG
*	SQR	BLINSQR
*	SRCH	BLIASRC
	SREP$	BLICSRE
	STR$	BLICNST
*	SUM	BLIAIPS, BLIADPS, BLIARPS
	TAN	BLINDTC
	TANH	BLINTNH
	TIME	BLISCLK
	TIME$	BLISCLK
	TRUNCATE	BLINDRT
*	UDIM	BLIAUSZ
	UPRC$	BLICCAS
	VAL	BLINVAL

Intrinsic Function Modules

The following list gives:

- The name of each routine that processes an intrinsic function (only modules that can be replaced are listed)
- The entry point(s) within each routine
- The intrinsic function processed through each entry point
- The argument type(s)
- The result type(s)

Argument and result types are indicated as:

C for character

D for decimal

I for integer

L for real double

S for real single

Optional arguments and optional argument lists are shown in brackets ([ ]). The argument count in the parameter list (see "Function Arguments" on page 47) indicates the actual number of arguments for a specific call to the function.

Routine	Entry Points Within	Function of Entry Point	Result Type
BLICCAS	BLICLWR	LWRC$ (C)	C
	BLICUPC	UPRC$ (C)	C
BLICCHR	BLICCHR	CHR$ (C)	C
BLICNST	BLICIST	STR$ (I)	C
	BLICSST	STR$ (S)	C
	BLICLST	STR$ (L)	C
	BLICDST	STR$ (D)	C
BLICPAD	BLICLPD	LPAD$ (C,I)	C
	BLICRPD	RPAD$ (C,I)	C
BLICPRM	BLICPRM	PARM$	C
BLICRPT	BLICRPT	RPT$ (C,I)	C
BLICSRE	BLICSRE	SREP$ (C,I,C,C)	C
BLICTRM	BLICLTR	LTRM$ (C)	C
	BLICRTR	RTRM$ (C)	C
BLINDAA	BLINSAC	ACOS (S)	S
	BLICLAC	ACOS (L)	L
	BLICDAC	ACOS (D)	D
	BLICSAS	ASIN (S)	S
	BLINLAS	ASIN (L)	L
	BLINDAS	ASIN (D)	D
BLINDAB	BLINDAB	ABS (D)	D
BLINDDG	BLINSCE	CEN (S)	S
	BLICLCE	CEN (L)	L
	BLICDCE	CEN (D)	D
	BLICSFA	FAH (S)	S
	BLINLFA	FAH (L)	L
	BLINDFA	FAH (D)	D
BLINDHH	BLINSHC	COSH (S)	S
	BLINSHS	SINH (S)	S
	BLINLHC	COSH (L)	L
	BLINLHS	SINH (L)	L
	BLINDHC	COSH (D)	D
	BLINDHS	SINH (D)	D
BLINDRR	BLINSDE	DEG (S)	S
	BLINSRD	RAD (S)	S
	BLINLDE	DEG (L)	L
	BLINLRD	RAD (L)	L
	BLINDDE	DEG (D)	D
	BLINDRD	RAD (D)	D

Routine	Entry Points Within	Function of Entry Point	Result Type
BLINDRT	BLINSRO	ROUND (S,I)	S
	BLINSTR	TRUNCATE (S,I)	S
	BLINLRO	ROUND (L,I)	L
	BLINLTR	TRUNCATE (L,I)	L
	BLINDRO	ROUND (D,I)	D
	BLINDTR	TRUNCATE (D,I)	D
BLINDSG	BLINDSG	SGN (D)	D
BLINDSS	BLINSCO	CSC (S)	S
	BLINSSE	SEC (S)	S
	BLINSCN	COS (S)	S
	BLINSSN	SIN (S)	S
	BLINLCO	CSC (L)	L
	BLINLSE	SEC (L)	L
	BLINLCN	COS (L)	L
	BLINLSN	SIN (L)	L
	BLINDCO	CSC (D)	D
	BLINDSE	SEC (D)	D
	BLINDCN	COS (D)	D
	BLINDSN	SIN (D)	D
BLINDTC	BLINSCT	COT (S)	S
	BLINSTN	TAN (S)	S
	BLINLCT	COT (L)	D
	BLINLTN	TAN (L)	D
	BLINDCT	COT (D)	L
	BLINDTN	TAN (D)	L
BLINIAB	BLINIAB	ABS (I)	I
BLINISG	BLINISG	SGN (I)	I
BLINLEN	BLINLEN	LEN (C)	I
BLINLG2	BLINSL2	LOG2 (S)	S
	BLINLL2	LOG2 (L)	L
	BLINDL2	LOG2 (D)	D
BLINLOG	BLINSLT	LOG10 (S)	S
	BLINLLT	LOG10 (L)	L
	BLINDLT	LOG10 (D)	D
BLINORD	BLINORD	ORD (C)	I
BLINPOS	BLINPOS	POS (C, C[,I])	I
BLINRAB	BLINSAB	ABS (S)	S
	BLINLAB	ABS (L)	L
BLINRSG	BLINSSG	SGN (S)	S
	BLINLSG	SGN (L)	L

Routine	Entry Points Within	Function of Entry Point	Result Type
BLINTNH	BLINSHT	TANH (S)	S
	BLINLHT	TANH (L)	L
	BLINDHT	TANH (D)	D
BLINVAL	BLINSVL	VAL (C)	S
	BLINLVL	VAL (C)	L
	BLINDVL	VAL (C)	D
BLISCLK	BLICDTI	DAT$ (I)	C
	BLICDTS	DAT$ (S)	C
	BLICDTL	DAT$ (L)	C
	BLICDTD	DAT$ (D)	C
	BLICDT2	DATE$	C
	BLICTIM	TIME$	C
	BLINJDT	DATE	I
	BLINJDY	JDY [(C)]	I
	BLINTIM	TIME	I

Appendix C. IBM BASIC Library: Argument List Formats

Argument and Result types

Five types of arguments and results are valid for user-modified library functions in BASIC: integer, real single, real double, decimal, and character. The format of each type as it appears in storage is discussed below.

Integer Arguments and Results

All integer arguments are 4-byte signed binary integers. An argument to the function routine occupies a fullword (aligned) in virtual storage. The address of this fullword will appear in the appropriate position in the argument list.

An integer function result is placed into general register 4. See "Function Results" on page 48 for details.

Real Single Arguments and Results

All real single arguments are 4-byte numbers in System/370 floating-point format. An argument to the function routine occupies a fullword (aligned) in virtual storage. The address of this fullword will appear in the appropriate position in the argument list.

A real single result is placed into floating-point register 0. See "Function Results" on page 48 for details.

Real Double Arguments and Results

All real double arguments are 8-byte numbers in System/370 floating-point format. An argument to the function routine occupies two fullwords (aligned to a fullword) in virtual storage. The address of this argument will appear in the appropriate position in the argument list.

A real double result is placed into floating-point register pair 0 and 1. See "Function Results" on page 48 for details.

Decimal Arguments and Results

All decimal arguments occupy 3 fullwords (aligned) in virtual storage. Of these 12 bytes, the low-order 2 bytes contain a signed binary exponent value. The low-order 4 bits of the 10th byte contain the sign, and the remaining high-order 19 hexadecimal digits contain the left-normalized, packed-decimal argument value. The address of this argument will appear in the appropriate position in the argument list.

Example 1: The following illustrates the format, in hexadecimal, of a decimal argument value of 476.3 (spaces have been added for readability):

```
X'4763000000000000000 C 0003'
```

where:

4763000... is the left-normalized argument value. Note that left-normalization means that the first significant (nonzero) digit of the argument value must be left-adjusted in the field.

C is the sign value. It may be any valid decimal sign code:

```
B'1010' - positive
B'1011' - negative
B'1100' - positive (preferred)
B'1101' - negative (preferred)
B'1110' - positive
B'1111' - negative
```

Note that X'C' is the preferred positive sign code; X'D' is the preferred negative sign code.

0003 is the binary exponent (power of 10) value.

Thus the decimal point is assumed to be to the left of the high-order digit in the field, which should be the leftmost nonzero digit of the number.

Example 2: The following illustrates the BASIC decimal representation of the value -0.0348:

```
X'3480000000000000000 D FFFF'
```

where:

34800... are the significant digits of the number.

D is the sign of the number (negative).

FFFF is the exponent (-1).

Note that a decimal zero may also be represented as 3 fullwords of zeros, including binary zeros in the sign field.

A decimal result is placed in a special location, see "Function Results" on page 48 for more information.

Character Arguments and Results

Character arguments appear in virtual storage as byte-aligned sequences of EBCDIC characters preceded by two 2-byte length fields. For arguments to a function routine, the first 2 bytes are ignored, and the second 2 bytes contain the length of the actual character data (excluding the 4-byte prefix). Such a character argument might be coded in Assembler language as follows:

```
STRING    DC    0C
          DS    XL2               IGNORED ON INPUT
          DC    XL2'000A'         STRING LENGTH
          DC    C'STRING ARG'     DATA
```

The address of this argument will appear in the appropriate position in the argument list.

The character result from a function routine is built as shown above, except that the length information must be stored in *both* 2-byte prefix fields. A character result might be coded as follows:

```
RESULT    DC    0C
          DC    XL2'000D'         RESULT LENGTH
          DC    XL2'000D'         RESULT LENGTH
          DC    C'RESULT STRING'  DATA
```

The address of this result is placed into general register 7. For more information, see "Function Results" on page 48.

Function Arguments

Argument lists to IBM BASIC Library function routines follow the IBM convention for Assembler language subroutines. That is, general register 1 contains the address of an argument list. In all cases, the argument list is an area of sequential fullwords in storage, the first of which contains a binary integer that indicates the number of fullword arguments that follow. The first fullword containing the argument count will be present in all cases, even when the number of arguments is zero, in which case it will contain binary zeros. When the number of arguments is nonzero, the first fullword following the argument count fullword will contain the address of the first argument, the second word following the argument count will contain the address of the second argument, and so on.

The arguments will have the types as shown for the entry point in "Intrinsic Function Modules" on page 41.

Function Results

The value your function routine returns to the calling BASIC program is referred to as the function result. This value will be one of the five types just discussed: integer, real single, real double, decimal, or character. The result will have the type as shown for the entry point in "Intrinsic Function Modules" on page 41. The result type determines how the result value is passed back to the caller of your function routine:

Integer result: Load the signed binary number into general register 4.

Real Single Result: Load the single precision result into floating-point register 0.

Real Double Result: Load the double precision result into floating-point register pair 0 and 1.

Decimal result: Move the 3 fullwords containing the result value to the address in general register 13 upon entry to your routine plus 128 (X'80').

For example, assuming register 13 has not been modified (or has been restored), and the result is at location ANSWER:

```
MVC    128(12,R13),ANSWER
```

Even though the internal format of decimal values allows up to nineteen digits, the result value will be rounded to seventeen digits.

Character result: Load the address of the first byte of the character length fields into general register 7.

Note: As stated under "Guidelines for Customizing Intrinsic Functions" on page 30, arguments *must not be modified* under any circumstances.

Index

A

about this manual iii
arguments
  character, format for 47
  decimal, format of 46
  function 47
  integer, format of 45
  real double, format of 45
  real single, format of 45
  result 48
  types of 45
ARITH option 5
ASCII character set 6
AUTOINC option 5
AUTOSTR option 5
auxiliary storage
  requirements 3, 4

B

BASE option 6
BASGEN EXEC procedure, messages from 18
BASIC
  See also IBM BASIC
BASIC environment
  description 1
  inside the 1
  outside the 1
BASIC MODULE 23, 24
BASICRUN MODULE 23, 24
BASSEG saved segment
  See discontiguous saved segments
BLISEG saved segment
  See discontiguous saved segments

C

character arguments, format of 47
character results, format of 47
code packets 1
coding 30
coding replacement function 31
COLL option 6
collating sequence, NATIVE versus
 STANDARD 6
COMPILE command 1
COPYINC option 6
customization
  description 27
  intrinsic functions 29
  reserved word table 29

D

data base, access from IBM BASIC 13
decimal arguments, format of 46
decimal point, USA versus European style 8
decimal results, format of 46
default option values 5, 8, 9
defining a saved segment for the Processor 16
description of operating system 1
digit separator, USA versus European style 8
discontiguous saved segments
  allocating space for 15, 18
  BLISEG segment 17, 31
  defining 16, 17
  installing IBM BASIC in 15
  loading and saving 18
  NAMESYS macro definition 16
  overview 15
  planning for 15
disk storage requirements 4
DMKSNT module 15, 18
downloading IBM BASIC to VM/PC 23
DRECSZE option 8

E

EBCDIC character set 6
EXEC procedure 4, 12-13

F

FLAG option 6
FLGFIPS option 6
function arguments 47
function result 48
functions
  See intrinsic functions

H

hardware requirements 3
HELP panels 2, 24

I

IBM BASIC
- customizing 27
- default option values 9
- downloading to VM/PC 23
- HELP panels 2
- in the BASIC environment 1
- installation planning 3
- installing 3, 11
- Library 1, 8, 17
- Library argument list formats 45
- Library modules 33, 34
- options, changing 27
- outside the BASIC environment 1
- Processor modules 33, 34
- Processor, description 1
- product options
  - LANGS 9
  - LIBOPT 9
  - PROCOPT 9
  - PRODOPT 9
  - RESOPT 9
- product options file 28
- segments, loading and saving 18
- using NUCXLOAD with 22
- using under VM/PC 21

industry standard statement iv
installation
- access the data base from IBM BASIC 13
- distribution tape format 11
- EXEC procedure 4, 12-13
- in discontiguous saved segments 15
- minimum programming requirements 3
- planning for 3
- procedure 11
- requirements
  - auxiliary storage 3
  - hardware 3
  - machine/hardware 3
  - operating system 3
  - planning 3
  - virtual storage 3

installation EXEC 11
installing your replacement function 31
integer arguments, format of 45
integer results, format of 45
intrinsic function replacement 39
intrinsic functions 39
- customizing 29
- installing replacements for 31
- listing of 41
- module names 41, 42
  - for intrinsic 39
  - listing of 41, 42
- replacement of 39
- replacing
  - argument lists for 45-48
  - cautions 29

intrinsic functions and module names 39
INVP option 8
IRECSZE option 8

L

LANGS option 9
LIBOPT option 9
Library
- argument list formats 45
- default option values 8
- defining a saved segment for 17
- description 1
- intrinsic functions
  - argument lists for 45-48
  - customizing 29
- module names 34-38
- options
  - DRECSZE 8
  - INVP 8
  - IRECSZE 8
  - MARGBOT 8
  - NRECSZE 9
  - RECORDS 9
  - selecting 5, 8
- options file, creating or changing 28

Library modules, description of 34-38
loading IBM BASIC segments 18
LSTLINE option 6
LSTPAGE option 6

M

machine requirements 3
machine/hardware requirements 3
manual organization iii
MARGBOT option 8

mode of operation, selecting 14
module names
  Library 34-38
  Processor 33, 34
modules with multiple entry points 30
MSGE option 6
MSGI option 6
MSGS option 6
MSGW option 7

N

NAMESYS macro 15, 16, 17, 18
NRECSZE option 9
NUCXLOAD, using with IBM BASIC 22
numeric variables, setting default precision 7

O

OBJLIST option 7
OBJPROG option 7
operating system, description 1
operating system, requirements 3
option value defaults
  defaults 9
  overriding defaults 9
options
  choosing 5
  setting default values
    Library 8
    overview 5
    Processor 5
  space 4
  speed 4
options, changing 27

P

performance
  see SPEED versus SPACE option
planning
  default option values 5
  SPEED versus SPACE 4
planning for installation 3
PREC option 7
precision for numeric variables, selecting default 7
preinstallation planning 5
PRFNAME option, description 7
PRFTYPE option, description 7
Processor
  code packets 1
  COMPILE command 1
  default option values 5
  defining a saved segment for 16
  description 1
  inside the BASIC environment 1
  module names 33, 34
  NAMESYS definition for BASSEG segment 16
  options
    ARITH 5
    AUTOINC 5
    AUTOSTR 5
    BASE 6
    COLL 6
    COPYINC 6
    FLAG 6
    FLGFIPS 6
    LSTLINE 6
    LSTPAGE 6
    MSGE 6
    MSGI 6
    MSGS 6
    MSGW 7
    OBJLIST 7
    OBJPROG 7
    PREC 7
    PRFNAME 7
    PRFTYPE 7
    PRTZO 7
    RENUINC 7
    RENUSTR 8
    SORLIST 8
    SORXREF 8
    SPACE 4
    SPEED 4
    STORMAP 8
    STRMAX 8
    TERM 8
  options file, creating or changing 27
  selecting default options 5
Processor modules, description of 33, 34
processor requirements 3
PROCOPT option 9
PRODOPT option 9
program directory 11
programming requirements 3
PRTZO option
  description 7
  relationship to PREC option 7
publications
  related iv
  required iv

R

real double arguments, format of 45
real double results, format of 45
real single arguments, format of 45
real single results, format of 45
RECORDS option 9
related publications iv
RENUINC option 7
RENUSTR option 8
replacement function
  coding 31
  installing 31
replacing intrinsic functions 29
required publications iv
requirements
  auxiliary storage 3, 4
  machine/hardware 3
  overview 3
  programming (operating system) 3
  universal instruction set 3
  virtual machine storage 4
  VSAM file processing 3
reserved word table
  BASRESW 29
  changing 29
  description 29
  format 29
  rules for changing 29
RESOPT option 9
results
  character, format for 47
  decimal, format of 46
  integer, format of 45
  real double, format of 45
  real single, format of 45
  types of 45

S

saved segments
  See discontiguous saved segments
saving IBM BASIC segments 18
setting default options, overview 5
shared segments
  See discontiguous saved segments
significant digits, selecting default precision 7
SORLIST option 8
SORXREF option 8
SPACE option 4, 17, 24, 33
  NAMESYS definition for BLISEG segment 17
SPEED option 5, 17, 23, 33
SPEED versus SPACE option
SQL/DS, access from IBM BASIC 13
standards iv
storage requirements
  auxiliary 4
  virtual machine 4
STORMAP option 8
STRMAX option 8
system name table 15, 18

T

tape format 11
TERM option 8

U

universal instruction set required 3

V

virtual machine storage requirements 4
virtual storage requirements 3
VM/PC
  downloading IBM BASIC to 23
  NUCXLOAD 22
  processing restrictions 22
  using IBM BASIC under 21
VMFPLC2 11
VMFPLC2 format 11
VSAM, when required 3

IBM BASIC/VM
Installation and Customization
SC26-4025-2

Reader's Comment Form

This manual is part of a library that serves as a reference source for system 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: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should 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.

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

Last 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.

SC26-4025-2

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

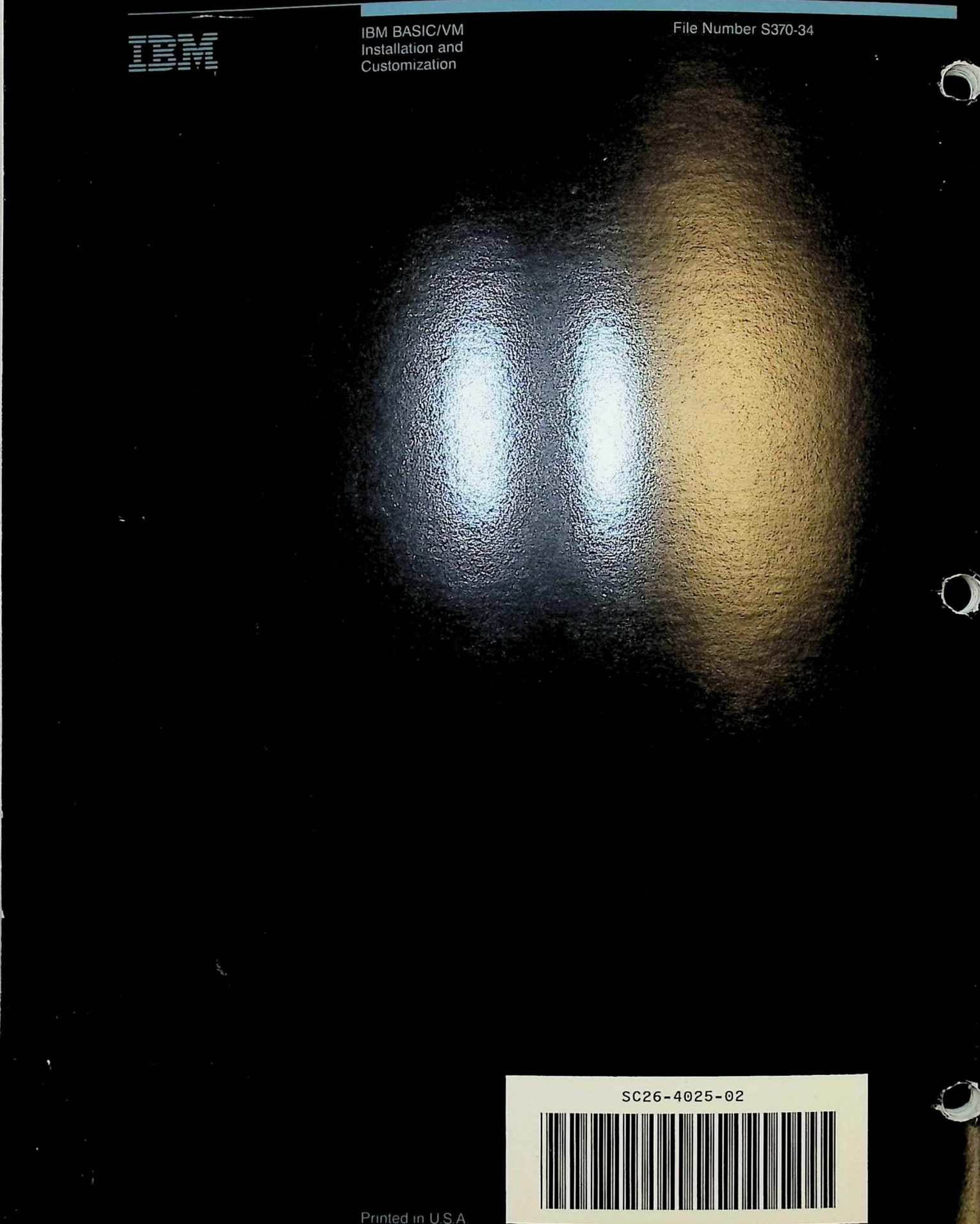
Fold and tape | Please do not staple | Fold and tape

IBM

IBM BASIC/VM Installation and Customization (File No. S370-34) Printed in U.S.A. SC26-4025-2

IBM BASIC/VM
Installation and
Customization

File Number S370-34



SC26-4025-02

Printed in U.S.A.