SC28-8309-0

Program Product

System/370 VS BASIC Installation Reference Material

Program Number: 5748-XX1

IBM

PREFACE

This publication is intended for system programmers and planners, who will be responsible for the installation and maintenance of the IBM VS BASIC Processor in one of the following systems:

- OS/VS2(TSO)
- VM/370(CMS)
- OS/VS1
- OS/VS2
- DOS/VS

This publication describes the requirements and procedures for installing and running the VS BASIC Processor. It is organized as follows:

- The "Introduction" presents a broad overview of the VS BASIC Processor and discusses some preliminary information that is necessary before the product can be installed.
- The "Installation Procedures" section is divided into four parts. If you intend to use VS BASIC in an OS/VS2(TSO) environment see the part "Installing VS BASIC as an Interactive and Batch Processor under OS/VS2(TSO)". This section can also be used to optionally install VS BASIC as a batch processor in the batch environment of OS/VS2(TSO). If you intend to install VS BASIC as a batch processor only under OS/VS1 or OS/VS2 see the part "Installing VS BASIC as a Batch Processor under OS/VS1 or OS/VS2". If you intend to install VS BASIC under CMS see the part "Installing VS BASIC as an Interactive and Batch Processor under VM/370(CMS)." If you intend to install VS BASIC under DOS/VS see the part "Installing VS BASIC as a Batch Processor under DOS/VS."
- The "Storage Estimates" section contains the real and virtual storage requirements of the VS BASIC Processor for each system under which it operates.
- The "Diagnostic Messages" section describes how to obtain diagnostic messages for each system.
- The "System Programming" section contains information of interest to system programmers for changing the installation procedures described in this book or for adapting it to special conditions that exist in a particular computer installation. Also, information that is not strictly classified as installation or operating procedures but which is necessary before the VS BASIC Processor can be used by application programmers is described.
- The "Appendixes" section contains the installation tape procedures for each system under which VS BASIC can be installed. It also lists the VS BASIC Processor modules and contains a copy of the sample program that is produced during the installation.

First Edition (May 1974)

This edition corresponds to Release 1.0 of the IBM System/370 VS BASIC Processor.

Changes are periodically made to the specifications herein; any such changes will be reported in subsequent revisions or Technical Newsletters. Before using this publication in conjunction with the operation of any IBM system, refer to the latest IBM System/360 and System/370 Bibliography, Order No. GA22-6822 for editions that are applicable and current.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your area.

A form for reader's comments is provided at the back of this publication. If the form has beem removed, comments may be addressed to IBM Corporation, Programming Publications, 1271 Avenue of the Americas, New York, New York, 10020.

As this book is revised, a summary of amendments will be included with the TNL or complete revision. It will be inserted immediately following the cover page and will highlight the changes made. As the book changes over a period of time, these summaries of amendments will enable you to build, as the first part of your book, a permanent section that will trace, in reverse chronological order, the development of this publication. Any revision of the complete book will include a reprinting of all previous summaries of amendments.

REFERENCE PUBLICATIONS

It is assumed that all readers installing VS BASIC under OS/VS are familiar with the contents of the following publications:

OS/VS Linkage Editor and Loader, Order No. GC26-3813

OS/VS JCL Services, Order No. GC28-0617

OS/VS JFC Reference, Order No. GC28-0618

OS/VS Utilities, Order No. GC35-0005

OS/VS Message Library: Routing and Descriptor Codes, Order No. GC38-1004

OS/VS Message Library: Utilities Messages, Order No. GC38-1005

OS/VS Message Library: Linkage Editor and Loader Messages, Order No. GC38-1007

Those readers who are using OS/VS1 should also be familiar with the following publications:

OS/VS1 System Generation Reference, Order No. GC26-3791

Operator's Library: OS/VS1 Reference, Order No. GC38-0110

OS/VS Message Library: VS1 System Messages, Order No. GC38-1001

OS/VS Message Library: VS1 System Codes, Order No. GC38-1003

Those readers who are using OS/VS2 should also be familiar with the following publications:

OS/VS2 System Generation Reference, Order No. GC26-3792

Operator's Library: VS2 Reference, Order No. GC38-0210

OS/VS Message Library: VS2 System Messages, Order No. GC38-1002

OS/VS Message Library: VS2 System Codes, Order No. GC38-1008

Readers who are TSO under OS/VS2 should be familiar with these additional publications:

OS/VS2 TSO Guide, Order No. GC28-0644

OS/VS2 TSO Guide to Writing a Terminal Monitor Program or a Command Processor, Order No. GC28-0648

Operator's Library: OS/VS2 TSO, Order No. GC38-0220

OS/VS Message Library: VS2 TSO Messages, Order No. GC38-1009

Readers who are installation VS BASIC under VM/370(CMS) should be familiar with the contents of the following publications:

IBM VM/370 Planning and System Generation Guide, Order No. GC20-1801

IBM VM/370 Command Language Guide for General User, Order No. GC20-1804

IBM VM/370 Operator's Guide, Order No. GC20-1806

Readers who are installing VS BASIC under DOS/VS should be familiar with the contents of the following publications:

DOS/VS System Control Statements, Order No. GC33-5376

DOS/VS System Generation, Order No. GC33-5377

Operator's Library DOS/VS Operating Procedures, Order No. GC33-5378

DOS/VS Messages, Order No. GC33-5379

DOS/VS Utilities, Order No. GC33-5381

The VS BASIC Processor is designed to operate in virtual storage systems, both time-sharing and batch. The processor is a problem program that runs under the following systems:

Time-sharing

- OS/VS2(TSO)
- VM/370(CMS)

Batch

- OS/VS1
- OS/VS2
- VM/370(CMS)
- DOS/VS

The VS BASIC Processor can be logically divided into four parts: an executor, a compiler, a library, and a debug processor. The executor serves as an interface between the system, under which VS BASIC is running, and the other three parts of the processor. It insulates the processor from the system and permits it to operate without any dependence on the host system. The executor intercepts and relays any processor requests for system services.

The compiler is a fast, one-pass language translator that accepts source programs written in the VS BASIC language and produces object code that is suitable for execution on a System/370 machine. Optionally, the compiler will process programs in long or short precision, permit the compilation to proceed into execution, store the object code produced, or produce object code that has been tailored to meet the needs of the debug processor.

The library contains run-time routines that assist in the execution of VS BASIC programs. In addition, it also contains routines that execute intrinsic library functions.

Since the compiler, library and OS/VS and TSO executors are reentrant, they can be installed under OS/VS in the link pack area making them available to a number of users simultaneously.

The debug processor permits the user to set breakpoints in his program as it is executing, display the contents of his program variables, and to trace the flow of control through the program. It is available only under TSO and CMS (interactive only).

The VS BASIC Processor is distributed on one tape. This tape contains all the processor components required plus procedures for link editing them into any system under which VS BASIC is designed to operate.

FORMAT OF THE VS BASIC PROCESSOR DISTRIBUTION TAPE

All the modules that are required to install VS BASIC on any system, under which it is designed to run, are contained on one installation tape. If you are a DOS/VS user, you have the option of receiving a disk instead of a tape. The format of the installation tape or disk is shown in Table 1.

Table 1. Files on the VS BASIC Distribution Tape or Disk

File No.	Contents	Record Characteristics		
		RECFM	LRECEL	BLKSIZE
1	DOS JCL and object code for the compiler, library, and executor and source macros and modules for SLF	FB	80	3440
2	OS/VS Installation JCL Procedure	FB	80	80
3	OS/VS(TSO) Installation JCL Procedure	FB	80	80
4	OS/VS Executor Module	FB	80	3200
5	TSO Executor Module	FB	80	3200
6	TSO RENUM Modules	FB	80	3200
7	TSO HELP Command Messages	FB	80	3200
8	CMS Installation EXEC Procedure	FB	80	3200
9	VS BASIC Compiler Module	FB	80	3200
10	VS BASIC Library Module	FB	80	3200
11	VS BASIC Debug Module	FB	80	3200
12	CMS Executor Module	FB	80	3200
13	CMS Utility Conversion Module	FB	80	3200
14	CMS HELP Module	FB	80	3200
15	HELP Error Message File	FB	80	3200
16	Sample Program	FB	80	3200
17	Separable Library Function (SLF) Macro Source	FB	80	3200
18	ICDKBFTB Source Module (for SLF)	FB	80	3200

INSTALLING VS BASIC AS AN INTERACTIVE AND BATCH PROCESSOR UNDER OS/VS2(TSO)

This section describes installing VS BASIC as an interactive processor under TSO. It optionally permits you to install, simultaneously, VS BASIC as a batch processor in your OS/VS2 system. For information on installing the VS BASIC as a batch processor only under OS/VS1, OS/VS2 (without TSO), or DOS/VS, or as an interactive and batch processor under VM/370(CMS), see the appropriate section of this book.

Note: If you are using private libraries to install VS BASIC, the private libraries can be transferred to an OS/VS1 system and VS BASIC will execute. Any system dependencies (that is, link list procedures) must be repeated for OS/VS1. If you are using the system libraries you will have to install VS BASIC following the instructions for OS/VS1.

REQUIREMENTS FOR INSTALLATION UNDER OS/VS2(TSO)

Equipment Configuration for OS/VS2(TSO)

- A System/370 machine configuration that can support the OS/VS2(TSO) environment (Model 145 or equivalent).
- At least one magnetic tape device. For OS/VS2(TSO), the VS BASIC processor is distributed only on magnetic tape.

OS/VS2(TSO) System Generation Requirements

- An installed Release 1.6 or a subsequent release of OS/VS2.
- The Time Sharing Option.
- TSO Utilities Maintenance Release VIM3 must be applied to TSO.
- The TSO EDIT, HELP information (EDIT member). (This is optional; however, if your users intend to use the HELP command with VS BASIC, it must be available.)
- The Floating-point Instruction Set.
- The Extended-precision, Floating-point Instruction Set. (This feature is optional; however, if your users intent to make use of the VS BASIC DOT, PRD, and SUM functions in extended-precision, it must be available.)
- The following access methods:

  TCAM
  QSAM
  BSAM
  VSAM (optional)
- The Level F Linkage Editor (alias IEWL).

- The following OS/VS utilities:

  IEBGENER
  IEBUPDTE
  IEHLIST
  IEHPROGM

- The line printer must be output class A.
- The card punch must be output class B.
- SYSDA must be available.

Release 1.6 also requires the following:

- TSO Enhancement Package #2 must be applied (Release 1.6 of OS/VS2 only).
- VS BASIC ICR must be applied (Release 1.6 of OS/VS2 only). VS BASIC ICR is available with the following feature numbers:

5036	800bpi
5037	1600bpi

  In addition, the optional source is available with these feature numbers:

5425	800bpi
5426	1600bpi

  VS BASIC ICR must be installed after the TSO Enhancement Package #2.

OS/VS2(TSO) Installation Requirements

- The distribution tape for VS BASIC
- A minimum region size of 128K.
- Space available on SYS1.LINKLIB or a private library for the VS BASIC TSO interactive and batch executors, compiler, library, debug processor, and RENUM facility. (See Table 3 in the "Storage Estimates" section for the storage requirements.)
- Optionally, space available on SYS1.HELP or a private library for the VS BASIC extensions to the HELP facility. (See Table 3 in the "Storage Estimates" for the storage requirements.)
- Optionally, space available on SYS1.LINKLIB or a private library for the VS BASIC batch executor, only if you wish to install the batch executor in separate library. (See Table 3 in the "Storage Requirements" section for the storage requirements.)
- Space available on SYS1.PROCLIB for the installation JCL procedure, VSBDEF, which you will write to define the target libraries for the installation procedure. (See Table 3 in the "Storage Estimates" section for the storage requirements.)

OVERVIEW OF THE INSTALLATION PROCEDURE UNDER OS/VS2(TSO)

To help you understand and select the information required for the installation of VS BASIC under OS/VS2(TSO), the following sequence of events is given:

- Ensure that your system conforms to the installation requirements of the VS BASIC Processor.
- Determine the target libraries that you will use. The VS BASIC Processor under OS/VS2(TSO) can use up to three libraries. One library is required for the interactive and batch versions of the executor, the compiler, the debug processor, the RENUM facility, and the run-time library. A second library is optionally required if you are installing the VS BASIC modifications to the TSO EDIT HELP facility. A third library is optionally required if you are installing the batch version of the executor in a library different from the library containing the interactive version.
- Prepare and run a JCL procedure that will be placed on SYS1.PROCLIB and that will define the the target libraries to your system.
- Prepare and run a JCL procedure that will ensure that any new private libraries that you may be creating for the VS BASIC Processor do not already exist in your system.
- Allocate and catalog any new private libraries, if used.
- Mount the distribution tape and start the reader to the third file on the tape.
- Decide whether you will concatenate any private libraries with SYS1.LINKLIB or whether you will identify tham with STEPLIB DD statements.
- Prepare a TSO LOGON procedure for your TSO users.
- Test the success of the installation procedure using the sample program, card deck that is provided.

INSTALLATION PROCEDURE FOR VS BASIC UNDER OS/VS2(TSO)

This procedure is designed to install VS BASIC as an interactive and optionally a batch processor under OS/VS2(TSO) only. If you wish to install VS BASIC as a batch processor only under OS/VS1 or OS/VS2, refer to the section "Installing VS BASIC as a Batch Processor under OS/VS1 and OS/VS2".

1 Prepare and run the following JCL procedure that will add to SYS1.PROCLIB, VSBDEF, a JCL procedure that defines the libraries that will contain the VS BASIC compiler, library, executors, HELP facility, debug processor, and RENUM modules:

```
A   //DEFINE     JOB      accounting-information,MSGLEVEL=(1,1)
    //           EXEC     PGM=IEBUPDTE,PARM=NEW
    //SYSPRINT   DD       SYSOUT=A
    //SYSUT2     DD       DSN=SYS1.PROCLIB,DISP=OLD
    //SYSIN      DD       DATA
    ./           ADD      NAME=VSBDEF,LIST=ALL
    ./           NUMBER   NEW1=10,INCR=10
    //VSB        EXEC     PGM=IEHLIST
    //SYSPRINT   DD       DUMMY
    //SYSIN      DD       DUMMY
B   //TLNK       DD       DSN=library-name, DISP=(OLD,PASS)
C   //VLNK       DD       DSN=library-name,DISP=(OLD,PASS)
D   //SYSHELP    DD       DSN=library-name(EDIT),DISP=(OLD,PASS)
E   //HELP       DD       DSN=library-name,DISP=(OLD,PASS)
F   //TAPE       DD       LABEL=(,NL).UNIT=(2400,,DEFER),
    //                    DCB=DEN=density,VOL=(,RETAIN,SER=VSBAS),
    //                    DISP=(OLD,PASS)
    ./           ENDUP
    /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B This statement defines the library that will contain the VS BASIC interactive version of the executor, compiler, library, debug processor, and RENUM facility. You must supply the following information:

library-name - is the name of the library to be used. You may specify either SYS1.LINKLIB or a private library name. The private library name may refer to a library that already exists or indicate the name of a new library that will be created later in the installation procedure.

C This statement defines the library that will contain the batch version of the VS BASIC executor. Installing the batch executor is optional. If you want the batch version of the VS BASIC executor installed in a separate library, you must supply the following information:

library-name - is the name of the library to be used. You may specify either SYS1.LINKLIB or a private library name (it may be the same private library name specified in the TLNK statement). The private library name may refer to a library that already exists or indicate the name of a new library that will be created later in the installation procedure.

If you do not want the batch version, replace this statement with the following statement:

```
//VLNK   DD   DSN=&&any-name,UNIT=SYSDA
//            DISP=(NEW,PASS),SPACE=(CYL(1,1,3))
```

where:

any-name - is any name that you choose for a temporary data set.

D This statement is optional and defines the library that contains the EDIT member of SYS1.HELP. Note that this data set must be accessed sequentially name(EDIT). If you want the VS BASIC modifications for the HELP facility, you must supply the following information.

library-name - is the name of the library that contains the EDIT member of SYS1.HELP. You may specify either SYS1.HELP or the private library name in which the EDIT member resides.

If you do not want the HELP facility updated, replace this statement with the following statement:

```
// SYSHELP DD   DUMMY
```

E This statement defines the library that will contain the VS BASIC modifications to the TSO HELP facility. Installing these modifications is optional. If you want this additional capability, you must supply the following information:

library-name - is the name of the library to be used. You may specify either SYS1.HELP or a private library name. The private library name may refer to a library that already exists or indicate the name of a new library that will be created later in the installation procedure.

If you do not want them, replace this statement with the following statement:

```
// HELP    DD   DUMMY
```

F This statement defines the magnetic tape unit on which the distribution tape is mounted. You must supply the following information:

density - indicates the density of the distribution tape. Specify 2 if the tape is 800 BPI or 3 if the tape is 1600 BPI.

2 If, in step 1 , you specified private, library names for the libraries that you plan to create, make sure that those names do not already exist in your system. The following JCL procedure can be used if you are not sure and it will also delete a data set that may have the same name:

```
A  //DELETE     JOB      accounting-information,MSGLEVEL=(1,1)
   //           EXEC     PGM=IEHPROGM
   //SYSPRINT   DD       SYSOUT=A
B  //TARGET     DD       VOL=(PRIVATE,RETAIN,SER=serial-number)
   //                    UNIT=unit,DISP=OLD
   //SYSIN      DD       *
C               SCRATCH  DSN=library-name,VOL=unit=serial-number
D               UNCATLG  DSN=library-name
   /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B This statement locates the volume that is to be searched for an old data set with the same name as the new library that is to be created. Supply the following information:

unit - indicates the direct access unit on which the volume is mounted.

serial-number - indicates the volume serial number of the volume to be searched.

Note: If you need to search more than one volume, you must insert a similar statement for each volume to be searched. You must, however, use a different ddname on each statement (for example, TARGET, TARGET1, TARGET2)

C This statement scratches the old data set. Supply the following information.

library-name - must be the same as the library name that you specified in the TLNK, VLNK, or HELP DD statements in step 1 . to be scratched.

unit - indicates the direct access unit on which the library resides.

serial-number - indicates the volume serial number of the volume containing the old data set that is to be scratched.

Note: If you are using more than one private library, you must include a SCRATCH statement for each library name. If the library was created with an expiration date, it cannot be scratched unless you specify PURGE.

D This statement uncatalogs the old data set. Supply the following information:

library-name - must be the same as the library name that you specified in the TLNK, VLNK, or HELP DD statements in step 1 .

Note: If you are using more than one private library, you must include an UNCATLG statement for each library name. Release 2.0 of OS/VS2 does not support UNCATLG, use a JCL procedure for uncataloging.

3 Allocate and catalog the new private libraries. The following JCL procedure will accomplish this:

```
A  //ALLOC      JOB   accounting-information,MSGLEVEL=(1,1)
   //           EXEC  PGM=IEHLIST
   //SYSPRINT   DD    SYSOUT=A
B  //LINK       DD    DSN=library-name,UNIT=unit
   //                 VOL=(PRIVATE,RETAIN,SER=serial-number)
   //                 SPACE=(TRK,(tracks,1,directory-records))
   //                 DISP=(NEW,CATLG)
C  //HELP       DD    DSN=library-name,UNIT=unit,
   //                 VOL=PRIVATE,RETIAN,SER=serial-number)
   //                 SPACE=TRK,(tracks,1,directory-records))
   //                 DCB=DSORG=PO,RECFM=F,LRECL=80,
   //                 BLKSIZE=7280),DISP=(NEW,CATLG)
   /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B This statement catalogues and allocates a new private library. You must supply the following information:

library-name - must be the same as the library name that you specified in the TLNK or VLNK DD statements in step 1 .

unit - identifies the direct access unit on which the new private library will be created.

serial-number - identifies the volume serial number of the volume on which the new private library is to be created.

tracks - indicates the number of tracks that will be required. See Table 3 in the "Storage Estimates" section for the amount of storage required by SYS1.LINKLIB.

directory-records - indicates the number of directory records that are required. See Table 3 in the "Storage Estimates" section for the number of records required by SYS1.LINKLIB.

Note: If you are using more than one private library, you must include one of these statements for each new library that you specified in the TLNK or VLNK DD statements in step 1 . You must, however, use a different ddname on each statement (for example, LINK and LINK1).

C This statement catalogues and allocates a new private library for the HELP facility modifications. You must supply the following information:

library-name - must be the same as the library name that you specified in the HELP DD statement in step 1 .

unit - identifies the direct access unit on which the new private library will be created.

serial-number - identifies the volume serial number of the volume on which the new private library will be created.

tracks - indicates the number of tracks that will be required. See Table 3 in the "Storage Estimates" section for the amount of storage required by SYS1.HELP.

directory-records - indicates the number of directory records that are required. See Table 3 in the "Storage Estimates" section for the number of records required by SYS1.HELP.

4 Mount the VS BASIC distribution tape on the magnetic tape device described by the TAPE DD statement in step 1 .

5 Start the reader to the tape device. Use the following command:

```
START RDR,cuu,LABEL=(3,NL)
```

where:

cuu - is the channel and unit address of the tape unit on which the distribution is mounted.

The JCL is read off the tape. The tape must then be readied again to read the actual installation procedure, VSBPP. After the START

RDR, issue a VARY command to take the tape device off-line making it available to be allocated for the second read. During the processing, the VSBDEF installation JCL procedure will be executed. Then the JCL procedure on the distribution tape (VSBPP for TSO) will link edit the compiler, library, batch and/or interactive executors, debug processor, and RENUM facility and place them and the HELP facility members into the libraries that you choose in the TLNK, VLNK, or HELP DD statements in step 1 . If the installation has been successful, a sample VS BASIC source program will be punched into a card deck. If the card deck is not produced, attempt to reinstall the processor. If the deck is still not produced, contact your IBM representative.

6 Before you can begin using VS BASIC at a terminal or make it available to your users, you must, first, consider an assumption that OS/VS makes. OS/VS2 assumes that the VS BASIC Processor resides on SYS1.LINKLIB. Therefore, if you have placed it there, you may omit this step and go on to the next step. However, if you have placed VS BASIC into a private library, you must do either step 6A or 6B. Step 6A describes how to concatenate private libraries with SYS1.LINKLIB using the Link Library List option of SYS1.PARMLIB. Step 6B describes using a STEPLIB DD statement in the TSO LOGON procedure or batch JCL to define private libraries.

A Prepare the following JCL procedure that will utilize the Link Library List option of SYS1.PARMLIB to concatenate your private libraries with SYS1.LINKLIB:

```
A  //CONCAT     JOB     accounting-information,MSGLEVEL=(1,1)
   //           EXEC    PGM=IEBUPDTE,PARM=NEW
   //SYSPRINT   DD      SYSOUT=A
   //SYSUT2     DD      DSN=SYS1.PARMLIB,DISP=OLD
   //SYSIN      DD      DATA
   ./           ADD     NAME=LNKST00,LIST=ALL
   ./           NUMBER  NEW1=01,INCR=02
B   SYS1.LINKLIB,...,library-name1[,library-name2]
   ./           ENDUP
   /*
```

An explanation of the lettered statement follows:

A Supply any accounting-information that your computing center requires.

B This statement concatenates your private library names with SYS1.LINKLIB. Supply the following information:

library-name$_1$[,library-name$_n$] - must be the same as the library names that you specified in the TLNK or VLNL DD statements in step 1 . Be sure to include any libraries that are already specified in the link library list.

Note: After concatenating your private libraries to SYS1.LINKLIB, you must re-IPL your system before you can use VS BASIC.

B When you prepare the TSO LOGON procedure for your terminal users, be sure to include a STEPLIB DD statement of the following form for each private library used in place of SYS1.LINKLIB:

```
//STEPLIB  DD  DSN=library-name,DISP=SHR
```

where:

library-name - is the same as that specified in the TLNK
DD statements in step 1 .

Note: You must inform your batch users that they must include a similar STEPLIB DD statement for the TLNK and VLNK libraries in the JCL that they submit with their jobs. If you are using a separate library for the batch version of the executor, they must also include the following DD statement immediately after the STEPLIB DD statement:

```
//     DD     DSN=library-name,DISP=SHR
```

where:

library-name - is the same as that specified in the VLNK
DD statement in step 1.

7 TSO assumes that the HELP facility resides on SYS1.HELP. Therefore, if you have placed it there, you may omit this step and go on to the next step. However, if you placed the HELP facility into a private library, you must include a STEPLIB DD statement of the following form in the LOGON procedure that you prepare for your terminal users:

```
//STEPLIB  DD   DSN=library-name,DISP=SHR
//         DD   DSN=SYS1.HELP,DISP=SHR
```

where:

library-name - is the same as that specified in the HELP
statement in step 1 .

8 Prepare a LOGON procedure for your TSO terminal users. See the section "TSO LOGON Procedure Considerations" in the "System Programming" part of this book for detailed information.

9 Using the sample program deck produced by the installation procedure, test the operation of the VS BASIC processor in your system. To run the sample program under TSO, you must first place the program into any data set to make it available at your terminal. The following JCL procedure will accomplish this:

```
A  //SAMPLE    JOB   accounting-information,MSGLEVEL=(1,1)
   //          EXEC  PGM=IEBUPDTE,PARM=NEW
   //SYSPRINT  DD    SYSOUT=A
B  //SYSUT2    DD    DSN=data-set-name,UNIT=unit,DISP=(MOD,CATLG),
   //                VOL=SER=serial-number,SPACE=(TRK(1,1,1)),
   //                DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
   //SYSIN     DD
   ./          ADD   NAME=VSBSAMP
               .
               .
               .
    Card Deck for VSBSAMP
               .
               .
               .
   ./          ENDUP
   /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B This statement defines the data set that will contain the sample program. Supply the following information:

data-set-name - is the name of any data set in which you choose to place the sample program.

unit - identifies the direct access unit on which the data set resides.

serial-number - identifies the volume serial number of the volume on which the data set resides.

Once the sample program deck has been placed into a data set, refer to the publication System/370 VS BASIC TSO Terminal User's Guide, Order No. SC28-8304, for information on running the sample program in an interactive environment. You will need the following command:

```
     run vsbsamp source
```

To run the sample program in a batch environment, refer to the publication System/370 VS BASIC OS/VS and DOS/VS Programmer's Guide, Order No. SC28-8308. You will need the following job control statements:

```
    //SAMPRUN    JOB    accounting-information,MSGLEVEL=(1,1)
    //           EXEC   PGM=ICDOSBSC
    //SYSPRINT   DD     SYSOUT=A
    //CONTROL    DD     *
                 RUN    VSBSAMP SOURCE
                  .
                  .
                  .
     Card Deck for VSBSAMP
                  .
                  .
                  .
    /*
    //SYS005     DD     UNIT=SYSDA,SPACE=(TRK,(10,10))
    //SYS009     DD     UNIT=SYSDA,SPACE=(TRK,(10,10))
    /*
```

Note: When running the sample program, the two data sets SYS005 and SYS009 are required. If the VS BASIC Processor was not installed on SYS1.LINKLIB, you must also include a STEPLIB DD statement of the following form:

```
     //STEPLIB  DD  DSN=library-name,DISP=SHR
```

INSTALLING VS BASIC AS A BATCH PROCESSOR UNDER OS/VS1 OR OS/VS2

This section describes installing VS BASIC as a batch processor only under OS/VS1 and OS/VS2. For information on installing VS BASIC as an interactive and batch processor under OS/VS2(TSO) or VM/370(CMS), or as a batch processor under DOS/VS, see the appropriate section of this book.

REQUIREMENTS FOR INSTALLATION UNDER OS/VS1 AND OS/VS2

Equipment Configuration for OS/VS1 and OS/VS2

- A System/370 machine configuration that can support the OS/VS1 (Model 135 or equivalent) or the OS/VS2 (Model 145 or equivalent) environments.

- At least one magnetic tape device. For OS/VS1 and OS/VS2, the VS BASIC Processor is distributed only on magnetic tape.

OS/VS1 and OS/VS2 System Generation Requirements

- An installed release of OS/VS1 or OS/VS2.
- The Floating-point Instruction Set.
- The Extended-precision, Floating-point Instruction Set. (This feature is optional; however, if your users intend to make use of the VS BASIC DOT, PRD, and SUM functions in extended-precision, it must be available.)
- The following access methods:

  QSAM
  BSAM
  VSAM (optional)

- The Level F Linkage Editor (alias IEWL).
- The following OS/VS utilities:

  IEBGENER
  IEBUPDTE (optional)
  IEHLIST
  IEHPROGM

- The line printer must be output class A.
- The card punch must be output class B.
- SYSDA must be available.

OS/VS1 and OS/VS2 Installation Requirements

- The distribution tape for VS BASIC.
- A minimum region or partition size of 128K.
- Space available on SYS1.LINKLIB or a private library for the VS BASIC executor compiler, and library. (See Table 4 in the "Storage Estimates" section for the storage requirements.)
- Space available on SYS1.PROCLIB or a private library for the installation JCL procedure, VSBDEF, which you will write to define the target libraries for the installation procedure. (See Table 4 in the "Storage Estimates" section for the storage requirements.)

OVERVIEW OF THE INSTALLATION PROCEDURE UNDER OS/VS1 OR OS/VS2

To help you understand and select the information required for the installation of VS BASIC as a batch processor only under OS/VS1 or OS/VS2, the following sequence of events is given:

- Ensure that your system conforms to the installation requirements of the VS BASIC Processor.
- Determine the target library that you will use. The VS BASIC processor under OS/VS requires one library for the batch executor, the compiler, and the run-time library.
- Prepare and run a JCL procedure that will be placed on SYS1.PROCLIB and that will define the target library to your system.
- Prepare and run a JCL procedure that will ensure that a new private library that you may be creating for the VS BASIC Processor does not already exist in your system.
- Allocate and catalog the new private library, if used.
- Mount the distribution tape and start the reader to the second file on the tape.
- Decide whether you will concatenate the private library, if used, with SYS1.LINKLIB or whether you will identify it with a STEPLIB DD statement.
- Test the success of the installation procedure using the sample program, card deck that is provided.

INSTALLATION PROCEDURE FOR VS BASIC UNDER OS/VS1 OR OS/VS2

This procedure is designed to install VS BASIC as a batch processor only under OS/VS1 or OS/VS2. If you wish to install VS BASIC as both a batch and an interactive processor under OS/VS2(TSO), refer to the section "Installing VS BASIC as an Interactive and Batch Processor under OS/VS2(TSO)."

1 Prepare the following JCL procedure that will add to SYS1.PROCLIB, VSBDEF, a JCL procedure that defines the libraries that will contain the VS BASIC compiler, library, and executor modules:

```
A  //INSTALL   JOB     accounting-information,MSGLEVEL=(1,1)
   //DEFINE    EXEC    PGM=IEBUPDTE,PARM=NEW
   //SYSPRINT  DD      SYSOUT=A
   //SYSUT2    DD      DSN=SYS1.PROCLIB,DISP=OLD
   //SYSIN     DD      DATA
   ./          ADD     NAME=VSBDEF,LIST=ALL
   ./          NUMBER  NEW1=10,INCR=10
   //VSB       EXEC    PGM=IEHLIST
   //SYSPRINT  DD      DUMMY
   //SYSIN     DD      DUMMY
B  //VLNK      DD      DSN=library-name,DISP=(OLD,PASS)
C  //TAPE      DD      LABEL=(,NL),UNIT=(2400,,DEFER),
   //                  DCB=DEN=density,VOL=(,RETAIN,SER=VSBAS),
   //                  DISP=(OLD,PASS)
   ./          ENDUP
   /*
```

An explanation of the lettered statements follows:

A Supply any <u>accounting-information</u> that your computing center requires.

B This statement defines the library that will contain the VS BASIC compiler, library, and executor. You must supply the following information:

<u>library-name</u> - is the name of the library to be used. You may specify either SYS1.LINKLIB or a private library name. The private library name may refer to a library that already exists or indicate the name of a new library that will be created later in the installation procedure.

C This statement defines the magnetic tape unit on which the distribution tape will be mounted. You must supply the following information:

<u>density</u> - indicates the density of the distribution tape. Specify 2 if the tape is 800 BPI or 3 if the tape is 1600 BPI.

2 If in step 1 , you specified a private, library name for the library that you plan to create, make sure that its name does not already exist in your system. The following JCL procedure can be used if you are not sure and it will also delete a data set that may have the same name:

```
A  //DELETE    JOB     accounting-information,MSGLEVEL=(1,1)
   //          EXEC    PGM=IEHPROGM
   //SYSPRINT  DD      SYSOUT=A
B  //TARGET    DD      UNIT=unit,VOL=(PRIVATE,RETAIN,
   //                  SER=serial-number),DISP=OLD
   //SYSIN     DD      *
C              SCRATCH DSNAME=library-name,VOL=unit=serial-number
D              UNCATLG DSNAME=library-name
   /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B This statement locates the volume that is to be searched for an old data set with the name as the new library that is to be created. Supply the following information:

unit - indicate the direct access unit on which the volume is mounted.

serial-number - indicate the volume serial number of the volume to be searched.

C This statement scratches the old data set. Supply the following information:

library-name - must be same as the library name that you specified in the VLNK DD statement in step 1 .

unit - indicates the direct access unit on which the library is mounted.

serial-number - indicates the volume serial number of the volume to be searched.

Note: If the library was created with an expiration date, it cannot be scratched unless you specify PURGE.

D This statement uncataloges the old data set. Supply the following information:

library-name - must be the same as the library name that you specified in the VLNK DD statement in step 1 .

Note: Release 2.0 of OS/VS2 does not support UNCATLG; use a JCL procedure for uncataloging.

3 Allocate and catalog the new private library. The following JCL procedure will accomplish this:

```
A  //ALLOC      JOB     accounting-information,MSGLEVEL=(1,1)
   //           EXEC    PGM=IEHLIST
   //SYSPRINT   DD      SYSOUT=A
B  //LINK       DD      DSN=library-name,UNIT=unit,
   //                   VOL=(PRIVATE,RETAIN,SER=serial-number),
   //                   SPACE=(TRK,(tracks,1,directory-records)),
   //                   DISP=(NEW,CATLG)
   /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B This statement cataloges and allocates the new private library. Supply the following information:

library-name - must be the same as the library name that you specified in the VLNK DD statement in step 1 .

unit - indicates the direct access unit on which the new private library will be created.

serial-number - identifies the volume serial number of the volume on which the new private library is to be created.

tracks - indicates the nubmer of tracks that will be required. See Table 4 in the "Storage Estimates" section for the amount of storage required by SYS1.LINKLIB.

directory-records - indicates the number of directory records that are required. See Table 4 in the "Storage Estimates" section for the number of directory records required by SYS1.LINKLIB.

4 Mount the VS BASIC distribution on the tape device described by the TAPE DD statement in step 1 .

5 Start the reader to the tape device. Use the following command:

```
START RDR,cuu,LABEL=(2,NL)
```

where:

cuu - is the channel and unit address of the tape unit on which the the distribution tape is mounted.

The JCL will be read from the tape. The tape must then be readied again to read the actual installation procedure VSBPP. After the START RDR, issue a VARY command to take the tape device off-line making it available to be allocated for the second read. During its processing, the VSBDEF installation JCL procedure will be executed. Then the JCL procedure on the distribution tape (VSBPP for OS) will link edit the compiler, library, and executor and place them into the library that you chose in the VLNK DD statement in step 1 . If the installation has been successful, a sample VS BASIC source program will be punched into a card deck. If the card deck is not produced, attempt to reinstall the processor. If the deck is still not produced, contact your IBM representative.

6 Before you can begin using VS BASIC in a batch environment, you must, first, consider an assumption that OS/VS makes OS/VS assumes that the VS BASIC Processor resides on SYS1.LINKLIB. Therefore, if you have placed it there, you may omit this step and go on to the next step. However, if you have placed VS BASIC into a private library, you must do either step 6A or 6B. Step 6A describes how to concatenate private libraries with SYS1.LINKLIB using the Link Library List option of SYS1.PARMLIB. Step 6B describes using a STEPLIB DD statement in your batch JCL to define private libraries.

A Prepare the following JCL procedure that will utilize the Link Library List option of SYS1.PARMLIB to concatenate your private library with SYS1.LINKLIB:

```
A  //CONCAT     JOB     accounting-information,MSGLEVEL=((1,1)
   //           EXEC    PGM=IEBUPDTE,PARM=NEW
   //SYSPRINT   DD      SYSOUT=A
   //SYSUT2     DD      DSN=SYS1.PARMLIB,DISP=OLD
   //SYSIN      DD      DATA
   ./           ADD     NAME=LNKLST00,LIST=ALL
   ./           NUMBER  NEW1=01,INCR=02
B   SYS1.LINKLIB,library-name
   ./           ENDUP
   /*
```

An explanation of the lettered statement follows:

A Supply any accounting-information that your computing center requires.

B This statement concatenates your private library name with SYS1.LINKLIB. Supply the following information:

library-name - must be the same as the library name that you specified in the VLNK DD statement in step 1 . Be sure to include any libraries that are already specified in the link library list.

Note: After concatenating your private library to SYS1.LINKLIB, you must re-IPL your system before you can use VS BASIC.

B Advise all the potential users of VS BASIC that they will have to include a STEPLIB DD statement of the following form in the JCL of any program that uses VS BASIC.

```
//STEPLIB   DD     DSN=library-name,DISP=SHR
```

where:

library-name - must be the same as that specified in the VLNK DD statement in step 1 .

7 Using the sample program produced by the installation procedure, test the operation of the VS BASIC processor. See the publication System/370 VS BASIC: OS/VS and DOS/VS Programmer's Guide, Order No. SC28-8308, for information on compiling and executing VS BASIC programs. You will need the following control statements:

```
//SAMPRUN    JOB     accounting-information,MSGLEVEL=(1,1)
//           EXEC    PGM=ICDOSBSC
//SYSPRINT   DD      SYSOUT=A
//CONTROL    DD      /*
             RUN     VSBSAMP SOURCE
              .
              .
              .
 Card Deck for VSBSAMP
              .
              .
              .
/*
//SYS005     DD      UNIT=SYSDA,SPACE=(TRK,(10,10))
//SYS009     DD      UNIT=SYSDA,SPACE=(TRK,(10,10))
/*
```

Note: When running the sample program, the two data sets SYS005 and SYS009, are required. In addition, if the VS BASIC Processor was not installed on SYS1.LINKLIB, you must also include a STEPLIB DD statement of the following form:

```
//STEPLIB  DD  DSN=library-name,DISP=SHR
```

INSTALLING VS BASIC AS AN INTERACTIVE AND BATCH PROCESSOR UNDER VM/370(CMS)

This section describes installing VS BASIC as an interactive and batch processor under VM/370(CMS). For information on installing the VS BASIC as an interactive and batch processor under OS/VS2(TSO) or as a batch processor under OS/VS1, OS/VS2, or DOS/VS, see the appropriate section of this book.

REQUIREMENTS FOR INSTALLATION UNDER VM/370(CMS)

Equipment Configuration for VM/370(CMS)

- A System/370 machine configuration that can support the VM/370(CMS) environment (Model 135 or equivalent).
- At least one magnetic tape device. For VM/370(CMS), the VS BASIC processor is distributed only on magnetic tape.

VM/370(CMS) System Generation Requirements

- An installed Release 1 PLC 13 or a subsequent release of VM/370(CMS).
- The Floating-point Instruction Set.
- The Extended-precision, Floating-point Instruction Set. (This feature is optional; however, if your users intend to make use of the VS BASIC DOT, PRD, and SUM functions in extended-precision, it must be available.)
- The TSOLIB TXTLIB. (This feature is optional; however, if your users intend to use the VS BASIC Debug Processor it must be available.)
- Support for the following OS access methods:

  QSAM
  BSAM

  Note: VSAM is available in VS BASIC; however, VM/370(CMS) does not support it.

VM/370(CMS) Installation Requirements

- The distribution tape for VS BASIC.
- The virtual machine in which you will install VS BASIC must be defined with a minimum of 256K of virtual storage.
- A virtual printer must be defined for your virtual machine.
- Space available on the system (S) disk for the VS BASIC compiler, library, executor, debug processor, utility program, HELP facility,

and HELP message file. (See Table 5 in the "Storage Estimates" section for more specific information on the the various devices that can be used.)

- The following file identifiers must not exist on the system (S) disk:

VSBINSTL	EXEC
VSB1	TEXT
VSB2	TEXT
VSB3	TEXT
VSB4	TEXT
VSB5	TEXT
VSB6	TEXT
VSBMSG	LIST
SAMPLE	VSBASIC
VSB	TXTLIB

OVERVIEW OF THE INSTALLATION PROCEDURE UNDER VM/370(CMS)

To help you understand and select information required for the installation of VS BASIC under VM/370(CMS), the following sequence of events is given:

- After complying with the installation requirements listed above, log onto VM/370 and IPL CMS.
- Move the VS BASIC installation EXEC procedure from the distribution tape into CMS.
- Ensure that the files that the VS BASIC installation procedure will create do not conflict with any files that you may already have on your system.
- Execute the VS BASIC installation procedure.
- Test the success of the installation procedure using the sample program that is provided in the SAMPLE VS BASIC file.

INSTALLATION PROCEDURE FOR VS BASIC UNDER VM/370(CMS)

This procedure is designed to install the VS BASIC processor under VM/370(CMS) only.

1 Mount the distribution tape on any available magnetic tape device.

2 Log onto VM/370 with a user identification that has been assigned a privilege class of B.

3 Attach a real tape device to your user identification. The device must be attached at virtual address 181. Use the following command:

```
attach cuu to userid as 181
```

where:

cuu - is the channel and unit address of the actual tape device.

userid - is the user identification that you logged on with in step 2 .

4 IPL CMS.

5 Access the system disk (S) as your A disk. Use the following command:

```
access 191 a
```

6 Move the VS BASIC installation EXEC procedure (VSBINSTL EXEC) from the distribution tape to disk. Use the following commands (the system responses have been included):

A
```
filedef input-file tap1 (recfm fb block 3200)
R;
```

B
```
filedef output-file disk vsbinstl exec (recfm f block 80)
R;

tape rew
R;

tape fsf 7
R;
```

C
```
movefile input-file output-file
R;
```

An explanation of the lettered commands follows:

A This command defines the input file on the distribution tape. Supply the following information:

input-file - is any unique ddname (that is, one that does not already exist in any other FILEDEF command).

B This command defines the output file into which the installation EXEC procedure will be placed. Supply the following information:

output-file - is any unique ddname. In addition, it must not be the same name that you specified for input-file above.

C This command moves the installation EXEC procedure from the distribution tape onto your A disk and assigns it the file identifier of VSBINSTL EXEC. Supply the following information:

<u>input-file</u> - must the same ddname that you specified for <u>input-file</u> above.

<u>output-file</u> - must be the same ddname that you specified for <u>output-file</u> above.

7 Execute the VSBINSTL EXEC pricedure. Use the following command:

```
exec vsbinstl install
```

The installation procedure begins with the following message acknowledging the start of the process:

```
INSTALLATION FOR VSBASIC PROGRAM PRODUCT (5748-XX1)
```

A From this point on the installation procedure will prompt you for additional information.

Specify the characteristics of the distribution tape. The following message will be printed at your terminal:

```
IF THE TAPE BEING INSTALLED IS OTHER THAN 9 TRACK DENSITY
800, ENTER IT'S MODE AND DENSITY AS FOLLOWS:

FOR 9 TRACK 6250 ENTER.........9 6250
FOR 9 TRACK 1600 ENTER.........9 1600
```

You must respond with one of the following:

$$\left\{\begin{array}{l}6250\\1600\\CR\end{array}\right\}$$

B Ensure that the system disk (S) has been accessed as your A disk. The following message will be typed at your terminal:

```
THE SYSTEM DISK TO RECEIVE THE COMPILER MUST BE ACCESSED
IN READ/WRITE STATUS AS THE 'A' DISK. IF NOT, ENTER
'END', ACCESS THE SYSTEM DISK IN THE PROPER STATUS AND
EXECUTE THIS EXEC AGAIN.

IF IT IS ACCESSED AS THE READ/WRITE 'A' DISK, PRESS
RETURN.
```

You must respond with one of the following:

$$\left\{\begin{array}{l}end\\CR\end{array}\right\}$$

If the system disk is not accessed as your read/write A disk, enter END. The following response will be typed at your terminal: 'EXIT FOR SYSTEM DISK ACCESS' Do step 5 , and repeat step 7 from the beginning. If the system disk is correctly accessed, enter CR and continue.

C The installation procedure then checks that to be created do not already exist. One or more of the following messages will be typed at your terminal if the corresponding file cannot be found:

```
FILE 'VSB1 TEXT A' NOT FOUND
FILE 'VSB2 TEXT A' NOT FOUND
FILE 'VSB3 TEXT A' NOT FOUND
FILE 'VSB4 TEXT A' NOT FOUND
FILE 'VSB5 TEXT A' NOT FOUND
FILE 'VSB6 TEXT A' NOT FOUND
FILE 'VSBMSG LIST A' NOT FOUND
FILE 'SAMPLE VSBASIC A' NOT FOUND
FILE 'VSB TXTLIB A' NOT FOUND
```

However, if any of these files already exist, the following error message is typed for each file found and the installation procedure is terminated:

```
'filename filetype filemode' ALREADY EXISTS ...  RENAME
OR ERASE IT AND TRY AGAIN
R(00002);
```

D If any of the error messages are typed, use the appropriate RENAME or ERASE commands to change the name of the files or to eliminate them completely, and repeat step 7 from the beginning.

E After the VS BASIC modules have been loaded, the following message is typed:

```
THE FOLLOWING NAMES ARE UNDEFINED:
ICDJUSTB
```

ICDJUSTB is the module name reserved for a user-written routine used under the Separable Library Facility (SLF). In addition, load maps are printed off-line.

F At this point, the installation procedure is finished. The following message acknowledges that:

```
INSTALLATION/REGEN COMPLETE
R;
```

During its processing, the VSBINSTL procedure moves the components of the VS BASIC Processor into the system and creates the following files.

File Identifier	Contains
VSB TXTLIB A2	Text Decks for All Members (retain for PTFs and SLF)
VSBCOMP MODULE A2	Object Code for Compiler (for CHAIN requests only)
VSBRUN MODULE A2	Object Code for Run-time Library
VSBTEST MODULE A2	Object Code for Debug Facility
VSBUTIL MODULE A2	Object Code for Conversion Utility (ICDLUTIL)
VSBASIC MODULE A2	Object Code for Compiler and Executor
VSBMSG LIST A2	Error Messages for HELP Facility
VSBHELP MODULE A2	Object Code for the HELP Command Interface
SAMPLE VSBASIC A1	Sample VS BASIC Source Program

If these files are not available, and no errors were detected, attempt to reinstall the VS BASIC Processor by reentering the EXEC VSBINSTL INSTALL command and repeating step 7 from the beginning. If these files are still not available, contact your IBM representative.

Note: Before you attempt to reinstall, you should delete all of the files that may have been installed prior to the point of failure.

8 Using the sample VS BASIC program in the file SAMPLE VSBASIC, test the operation of the VS BASIC processor. See the publication System/370 VS BASIC: CMS Terminal User's Guide, Order No. SC28-8306, for information on compiling and executing VS BASIC programs. You will need the following command:

```
vsbasic  sample  (source)
```

Note: Because of the need for TSOLIB TXTLIB and formatting the users A disk and because certain terminal keyboard characters may not be available or may conflict with the CMS line editing characters, it is recommended that you make the appropriate changes in your user's directories and provide your terminal users with profiles that define these items. See the section "CMS Preparations for New VS BASIC User's" for more specific information.

INSTALLING VS BASIC AS A BATCH PROCESSOR UNDER DOS/VS

This section describes installing VS BASIC as a batch processor under DOS/VS. For information on installing VS BASIC as an interactive and batch processor under OS/VS2(TSO) or VM/370(CMS) or as a batch processor under OS/VS1 or OS/VS2, see the appropriate section of this book.

REQUIREMENTS FOR INSTALLATION UNDER DOS/VS

Equipment Configuration for DOS/VS

- A System/370 machine configuration that can support the DOS/VS environment (Model 115 or equivalent).
- At least one magnetic tape device or one disk device. For DOS/VS, the VS BASIC processor is distributed on either magnetic tape or disk.

DOS/VS System Generation Requirements

- An installed Release 28 or subsequent release of DOS/VS.
- The Floating-point Instruction Set.
- The Extended-precision, Floating-point Instruction Set. (This feature is optional; however, if your users intend to make use of the VS BASIC DOT, PRD, and SUM in extended-precision, it must be available.)
- The following access methods:

  QSAM
  BSAM
  VSAM (optional)

- The following DOS/VS utility programs:

  DSTRB
  DITTO
  MAINT
  ASSEMBLY

- The following system options:

FOPT	(Optional Features Macro)
AB=YES	(ABEND)
PC=YES	(Program Check)
TOD=YES	(Time of Day)

DOS/VS Installation Requirements

- The distribution tape or disk for VS BASIC.
- A temporary scratch tape or disk file for the installation procedure.

- A minimum partition size of 128K.
- Space available in the system or a private relocatable library for the VS BASIC processor modules. (See Table 6 in the "Storage Estimates" section of this book for the storage requirements.)
- Space available in the system or a private core image library for the link edited VS BASIC processor modules. (See Table 6 in the "Storage Estimates" section of this book for the storage requirements.)
- Optionally, space available in the system or a private source statement library for macros and ICDKBFTB source that is required for the Separable Library Facility. (See Table 6 in the "Storage Estimates" section of this book for the storage requirements.)

OVERVIEW OF THE INSTALLATION PROCEDURE UNDER DOS/VS

To help you understand and select information required for the installation of VS BASIC under DOS/VS, the following sequence of events is given:

- Ensure that your system conforms to the installation requirements of the VS BASIC Processor.
- Determine the target libraries that you will use. The VS BASIC Processor under DOS/VS requires a relocatable and a core image library for the batch executor, the compiler, and the run-time library. You may, optionally, use a source statement library if you wish to make the Separable Library Facility available when you install VS BASIC.
- Allocate space for the new libraries and define their extents to the system.
- Deblock the distribution tape or disk file.
- Assign the system input to the tape or disk device.
- Assign any private libraries that you may be using.
- Test the success of the installation procedure using the sample program, card deck that is provided.

INSTALLATION PROCEDURE FOR VS BASIC UNDER DOS/VS

This procedure is designed to install VS BASIC as a batch processor under DOS/VS only.

1 Select a DOS/VS partition in whcih you will be running the VS BASIC processor. The installation procedure assumes that the background partition will be used.

2 Make sure that the timer is assigned to the partition in which VS BASIC will be running. If it is not, enter the following command:

$$\text{TIMER}\ \left\{\begin{matrix}\text{BG}\\\text{F1}\\\text{F2}\\\text{F3}\\\text{F4}\end{matrix}\right\}$$

3 If you are using the system relocatable, core image and optional source statement libraries or if you are using pre-allocated, private libraries in place of the system libraries, you may skip this step and go to step 4 . If you are using private libraries that have not been pre-allocated, do step 3A , 3B , or 3C , as required. Step 3A allocates a private core image library and step 3B allocates a private relocatable library. If you plan to make the Separable Library Facility available during the installation procedure, do step 3C, which allocates a private source statement library. Finally, do step 3D, which places label information about your private libraries on SYSRES.

A The following JCL procedure allocates a private core image library:

```
   // JOB VSBPCL
A  // ASSGN SYS003,X'cuu'
B  // DLBL IJSYSPC,'VS BASIC PCIL',date,SD
C  // EXTENT SYS003,disk-label,1,0,first-track,total-tracks
   // EXEC CORGZ
D     NEWVOL CL=cylinders(directory-tracks)
   /*
   /&
```

An explanation of the lettered statements follows:

A This statement assigns SYS003 to an actual disk device. You must supply the channel and unit address of the device on which the private core image library is to reside. The device chosen must be of the same type as the device assigned to SYSRES.

B This statement assigns a label to the private core image library that you will be using. You may supply an expiration date or accept the date defined for your system. Without the date specified, the statement would appear as:

```
// DLBL IJSYSPC,'VS BASIC PCIL',,SD
```

C This statement defines the amount of storage that the private library will require. You must supply the following information:

<u>disk-label</u> - specifies the label of the disk to be used.

<u>first-track</u> - indicates the first track of the private library.

total-tracks - indicates the total number of tracks required by the VS BASIC processor modules. See Table 6 in the "Storage Estimates" section of this book for the number of tracks required by SYSCLB.

D This statement provides additional storage information about the private core image library. You must supply the following information:

cylinders - indicates the number of cylinders required by the VS BASIC processor modules. See Table 6 in the "Storage Estimates" section for the number of cylinders required by SYSCLB. This must include at least 10 tracks more than the space required for the library and its directory.

directory-tracks - indicates the number of tracks that are required by the directory. See Table 6 in the "Storage Estimates" section for the number of directory records required by SYSCLB.

B The following JCL procedure allocates a private relocatable library:

```
   // JOB VSBPRLB
A  // ASSGN SYSRLB,X'cuu'
B  // DLBL IJSYSRL,,'VS BASIC PRLB',,date,,SD
C  // EXTENT SYSRLB,disk-label,,1,,0,first-track,,total-tracks
   // EXEC CORGZ
D     NEWVOL RL=cylinders(directory-records)
   /*
   /&
```

An explanation of the lettered statements follows:

A This statement assigns SYSRLB to an actual disk device. You must supply the channel and unit address of the device on which the private relocatable library will reside. The device chosen must be of the same type as the device assigned to SYSRES.

B This statement assigns a label to the private relocatable library that you will be using. You may supply an expiration date or accept the date defined for your system. Without the date specified, the statement would appear as:

```
// DLBL IJSYSRL,'VS BASIC PRLB',,SD
```

C This statement defines the amount of storage that the private library will require. You must supply the following information:

disk-label - specifies the label of the disk to be used.

first-track - indicates the first track of the private library.

total-tracks - indicates the total number of tracks required by the VS BASIC processor modules. See Table 6 in the "Storage Estimates" section of this book for the number of tracks required by SYSRLB.

D This state provides additional storage information about the private relocatable library. You must supply the following information:

cylinders - indicates the number of cylinders required by the VS BASIC processor modules. See Table 6 in the "Storage Estimates" section of this book for the number of cylinders required by SYSRLB. This must include at least 10 tracks more than the space required for the library and its directory.

directory-tracks - indicates the number of tracks that are required by the directory. See Table 6 in the "Storage Estimates" section of this book for the number of directory records required by SYSRLB.

C The following JCL procedure allocates a private source statement library:

```
    // JOB VSBPSSL
A   // ASSGN SYSSLB,X'cuu'
B   // DLBL IJSYSSL,'VS BASIC PSSL',date,SD
C   // EXTENT SYSSLB,disk-label,1,0,first-track,total-tracks
    // EXEC CORGZ,REAL
D      NEWVOL CL=cylinders(directory-tracks)
    /*
    /&
```

An explanation of the lettered statements follows:

A This statement assigns SYSSLB to an actual disk device. You must supply the channel and unit address of the device on which the private source statement library is to reside. The device chosen must be of the same type as the device assigned to SYSRES.

B This statement assigns a label to the private source statement library that you will be using. You may supply an expiration date or accept the date defined for your system. Without the date specified, the statement would appear as:

```
// DLBL IJSYSSL,'VS BASIC PSSL',,SD
```

C This statement defines the amount of storage that the private library will require. You must supply the following information:

disk-label - specifies the label of the disk to be used.

first-track - indicates the first track of the private library.

total-tracks - indicates the total number of tracks required by the macros and ICDKBTFB source for the Separable Library Facility. See Table 6 in the "Storage Estimates" section of this book for the number of tracks required by SYSSLB.

D This statement provides additional storage information about the private source statement library. You must supply the following information:

cylinders - indicates the number of cylinders required by the macros and ICDKBFTB source for the Separable Library Facility. See Table 6 in the "Storage Estimates" section for the number of cylinders required by SYSSLB. This must include at least 10 tracks more than the space required for the library and its directory.

directory-tracks - indicates the number of tracks that are required by the directory. See Table 6 in the "Storage Estimates" section for the number of directory records required by SYSSLB.

D The following JCL procedure stores label information in SYSRES, thus eliminating the need to repeat these statements each time SYSRLB is assigned:

```
// JOB VSBLABEL
// OPTION PARSTD
        .
        .
        .
   Existing SYSRES Label Information
        .
        .
        .
// DLBL IJSYSCL,'VS BASIC PCIL',date,SD
// EXTENT SYSCLB,disk-label,1,0,first-track,total-tracks
// DLBL IJSYSRL,'VS BASIC PRLB',date,SD
// EXTENT SYSRLB,disk-label,1,0,first-track,total-tracks
// DLBL IJSYSSL,'VS BASIC PSSL',date,SD
// EXTENT SYSSLB,disk-label,1,0,first-track,total-tracks
```

Include only those DLBL and EXTENT statements that are required for the private libraries that you are using. The information specified here is the same as that specified in steps 3a, 3b, and 3c.

4 Deblock the distribution tape or disk. Do step 4A or 4B. Step 4A deblocks a distribution tape and step 4B deblocks a distribution disk.

A The following JCL procedure deblocks a distribution tape:

```
     // JOB DEBLOCK TAPE
A    // ASSGN SYS004,X'cuu'
B    // ASSGN SYS005,X'cuu'
     // UPSI 10100
     // EXEC DSTRB
     // UDS DBL
     // END
     /&
```

An explanation of the lettered statements follows:

A This statement assigns the tape device on which the distribution tape will be mounted. You must supply the channel and unit address of the tape device to be used.

B This statement assigns the tape device on which the deblocked output will be placed. You must supply the channel and unit address of the tape device to be used.

B The following JCL procedure deblocks a distribution disk:

```
     // JOB DEBLOCK
A    // ASSGN SYS004,X'cuu'
B    // ASSGN SYS005,X'cuu'
     // DLBL UIN,'A5748XX1.SYSIN.V1M1.DOSJCL',,SD
C    // EXTENT SYS004,disk-label,1,0,first-track,total-tracks
D    // DLBL UOUT,'any-file-id',date,SD
E    // EXTENT SYS005,disk-label,1,0,first-track,total-tracks
     // EXEC DSTRB DKDK
     // UDS DBL
     // END
     /&
```

An explanation of the lettered statements follows:

A This statement assigns the disk device on which the distribution disk will be mounted. You must supply the channel and unit address of the disk device to be used.

B This statement assigns the disk device on which the deblocked output will be placed. You must supply the channel and unit address of the disk device to be used.

C This statement defines the extent of the installation file. You must supply the following information:

disk-label - specifies the label of the distribution disk.

first-track - indicates the first track of the installation file.

total-tracks - indicates the total number of tracks required by the installation file. This information should be obtained from a VTOC listing of the distribution disk.

D This statement assigns a label to the data set that will contain the deblocked distribution disk. You must supply the following information:

any-file-id - supply any appropriate file-id for the deblocked data set.

date - indicates an expiration date for the data set. You may accept the default that is defined for your system. Without the date specified, the statement would appear:

```
// DLBL UOUT,'any-file-id',,SD
```

E This statement defines the amount of storage that the deblocked data set will require. You must supply the following information:

disk-label - specifies the label of the disk that you have selected to contain the data set.

first-track - indicates the first track of the data set.

total-tracks - indicates the total number of tracks required by the data set. See Table 6 in the "Storage Estimates" section of this book for the number of tracks required by the deblocked installation file.

5 Read the deblocked distribution tape or disk. Use the following command for a tape installation.

```
ASSGN SYSIN,X'cuu'
```

where:

cuu - is the channel and unit address of the device containing either the deblocked tape.

For a disk installation, use the following commands:

```
// DLBL IJSYSIN,'any-file-id',,SD
// EXTENT SYSIN
   ASSGN SYSIN,X'cuu'
```

where:

any-file-id - is the same file-id that you specified for the deblocked installation file in the // DLBL UOUT statement in step 4B .

cuu - is the channel and unit address of the device containing the deblocked installation file.

6 The actual installation is begun. From this point on the installation procedure will prompt you at the console for additional information that may be required. The following message is typed out acknowledging the start of the procedure:

```
*   VS BASIC DOS INSTALLATION
*   5748-XX1 COPYRIGHT IBM CORP.  1972
*   REFER TO INSTRUCTIONS ON COPYRIGHT NOTICE, 120-2083
*          NOTE TO USERS
* TO ALLOW USERS TO SKIP JOBS OR INSERT JCL DEFINING
LIBRARIES,
* THE SYSTEM WILL PAUSE FOR OPERATOR RESPONSE.  A MESSAGE WILL
* ACCOMPANY EACH PAUSE EXPLAINING THE RESPONSE REQUIRED.
// JOB 1 CONDS OF VS BASIC DOS/VS BATCH
// OPTION LOG
* IF YOU ARE USING PRIVATE LIBRARIES FOR THE CIL AND RLB,
* PLEASE ASSIGN THEM PERMANENTLY AT THIS TIME.
* RESPOND WITH EOB TO CONDENSE THESE LIBRARIES OR CANCEL.
// PAUSE
```

A Job 1, referred to in the message, condenses the core image and relocatable libraries and deletes the VS BASIC modules from the relocatable library. Use the following commands for a permanent assignment of your private libraries:

```
ASSGN SYSRLB,X'cuu'
ASSGN SYSCLB,X'cuu'
```

where:

cuu - is the channel and unit address of the device that contains the pre-allocated private libraries.

Signal EOB if you want to condense the libraries or cancel the job if you are not using private libraries.

The installation procedure continues with jobs 2 and 3. Job 2 executes the MAINT utility program to catalog the VS BASIC object modules into SYSRLB or your private library if you specified one. After the modules have been cataloged, job 3 link edits the modules into SYSCLB or your private core image library. This job produces a link edit map.

B Before job 4 executes. the following message is printed at your console:

```
// JOB 4 SLF PLACE SOURCE AND MACROS IN SSL
* THIS JOB PLACES MACROS AND A SOURCE MODULE FOR THE SLF
* FACILITY IN THE SSL.  IF YOU ARE USING A PRIVATE SSL,
* PLEASE ASSIGN IT PERMANENTLY AT THIS TIME.
* THEN RESPOND EOB TO CONTINUE.  OTHERWISE, CANCEL.
// PAUSE
```

Job 4 places source macros and ICDKBFTB into a source statement library. If you wish to install SLF at this time and you are using a private source statement library, enter the following command for a permanent assignment:

```
ASSGN SYSSLB,X'cuu'
```

where:

  cuu - is the channel and unit address of the device that contains the pre-allocated private library.

Then signal EOB. If you do not wish to install SLF, cancel this job.

C If you did not cancel job 4, it will assemble the macros. You must define a tape or a disk file for the temporary output of the assembly. The following message is printed at your console:

```
// OPTION LOG,NODECK,EDECK
* TO ASSEMBLE VS BASIC MACROS FOR THE SEPARABLE LIBRARY
* FEATURE, ASSIGN SYSPCH TO A TAPE OR TO A FILE ON DISK
* USING DLBL AND EXTENT CARDS.
// PAUSE
```

Use the following commands for a scratch tape:

```
      ASSGN SYSPCH,X'cuu'
```

or for a scratch disk file:

```
      // DLBL IJSYSPH,'any-file-id',,SD
      // EXTENT SYSPCH,disk-label,1,0,first-track,60
         ASSGN SYSPCH,X'cuu'
```

where:

  cuu - is the channel and unit address of the tape or disk device that contains the scratch file.

  any-file-id - is any appropriate file-id.

  first-track - indicates the first track of the scratch file.

D Job 4 continues by cataloging the macros, in E-deck form, in the source statement library. The following message is printed at your console:

```
* IF YOU HAVE USED A SCRATCH TAPE,
* PLEASE ASSIGN SYSIPT TO THE SAME TAPE.
* IF YOU HAVE USED A DISK, ASSIGN SYSIPT TO THAT DISK FILE
* RESPOND EOB
// PAUSE
```

Use the following commands for tape:

```
      MTC WTM,X'cuu',1
      MTC REW,X'cuu'
      ASSGN SYSPCH,X'00D'
      ASSGN SYSIPT,X'cuu'
```

or for disk:

```
         CLOSE SYSPCH,X'00D'
      // DLBL IJSYSIN,'any-file-id'
      // EXTENT SYSIPT
         ASSGN SYSIPT,X'cuu'
```

where cuu and any-file-id are the same as specified in step 6C.

The installation procedure continues by executing the MAINT utility program to catalog the macros. When the end of file is reached, the following message will be printed at the console:

```
* DOS/VS BASIC INSTALLATION COMPLETE
```

7 Punch the sample program onto a card deck. If you are using a distribution tape, you must use DITTO. For a distribution disk, use the following JCL procedure:

```
// JOB COPY
// ASSGN SYS005,X'cuu'
// ASSGN SYS006,X'00D'
// DLBL UIN,'A5748XX1.SYSIN.V1M1.SAMPLE',,SD
// EXTENT SYS005,disk-label,1,0,first-track,total-tracks
// EXEC CDKCD
// URC TF,A=(3200)
/&
```

where:

cuu - is the channel and unit address of the disk device containing the distribution disk.

disk-label - specifies the label of the distribution disk.

first-track - indicates the first track of the sample program.

total-tracks - indicates the total number of tracks required by the sample program. This information should be obtained from a VTOC listing of the distribution disk.

8 Using the sample program deck, test the operation of the VS BASIC Processor. See the publication System/370 VS BASIC: OS/VS and DOS/VS Programmer's Guide, Order No. SC28-8303, for information on compiling and executing VS BASIC programs. You will need the following job control statements to run this program:

```
// JOB SAMPLE
// ASSGN SYS005,x'cuu'
// DLBL SYS005,'IJSYSxx',,SD
// EXTENT SYS005,'disk-label',1,0,first-track,10
// ASSGN SYS009,x'cuu'
// DLBL SYS009,'IJSYSxx',,SD
// EXTENT SYS009,'disk-label',1,0,first-track,10
// EXEC PGM=ICDDSBSC,SIZE=64K
   RUN * SOURCE
         .
         .
         .
Sample Program Deck
         .
         .
         .
         .
/*
/&
```

where:

<u>cuu</u> - is the channel and unit address of the devices that will contain the data sets used by the sample program.

<u>xx</u> - is any valid two-digit number that will complete the DOS/VS file-id.

<u>disk-label</u> - is the label of the disk that you are using for the files required.

<u>first-track</u> - is the first track of each disk file.

<u>Note</u>: When running the sample program, the two data sets SYS005 and SYS009 are required.

If the sample program card deck is not produced and no errors were detected, attempt to reinstall the processor. If the deck is still not produced, contact you IBM representative.

Table 2. Dynamic Storage Required for Installing and Executing VS BASIC

System	Minimum Region, Partition, or Virtual Machine Size		
	For Installation	For Execution without Debug	For Execution with Debug
OS/VS1	128K	128K	
OS/VS2	128K	128K	
OS/VS2 (TSO)	128K	128K	256K
VM/370 (CMS)	256K	300K[1]	384K[1]
DOS/VS	128K	256K	

[1]These storage estimates for CMS are given with the understanding that users who will be accessing a large number of disks or whose disks contain a large number of files may require additional storage for in-storage indexes.

Table 3. Auxilliary Storage Required for Installing VS BASIC under OS/VS2(TSO)

Date Set	Number of Directory Records[1]	Cylinders Required					Tracks Required				
		2305-1 Drum	2305-2 Drum	2314/ 2319 Disk	3330/ 3333 Disk	3400 Disk	2305-1 Drum	2305-2 Drum	2314/ 2319 Disk	3330/ 3333 Disk	3340 Disk
SYS1. PROCLIB	1						1	1	1	1	1
SYS1. LINKLIB or a Private Library (includ-ing TOS and batch execu-tors)	15			6	4	5	62	60	120	67	104
SYS1. LINKLIB or a Private Library (for the batch executor only)	1						4	4	7	4	6
SYS1. HELP or a Private Library	1			5	3	4	50	48	96	55	90

[1]The number of 256-byte records allocated for a directory when a new partitioned data set is being defined. (See the description of the SPACE parameter of the DD statement in the publication <u>OS/VS JCL Reference</u>, Order No. GC28-0618.) The number of directory records that can be contained on a track is as follows:

IBM 2305-1	Drum Storage	16
IBM 2305-2	Drum Storage	26
IBM 2314	Disk Storage	17
IBM 2319	Disk Storage	17
IBM 3330	Disk Storage	28
IBM 3333	Disk Storage	28
IBM 3340	Disk Storage	16

Table 4. Auxilliary Storage Required for Installing VS BASIC under OS/VS1 or OS/VS2

Data Set	Number of Directory Records[1]	Cylinders Required					Tracks Required				
		2305-1 Drum	2305-2 Drum	2314/ 2319 Disk	3330/ 3333 Disk	3340 Disk	2305-1 Drum	2305-2 Drum	2314/ 2319 Disk	3330/ 3333 Disk	3340 Disk
SYS1. PROCLIB	1						1	1	1	1	1
SYS1. LINKLIB or a Private Library	8			3	2	3	26	25	50	28	49

[1]The number of 256-byte records allocated for a directory when a new partitioned data set is being defined. (See the description of the SPACE parameter of the DD statement in the publication OS/VS JCL Reference, Order No. GC28-0618.) The number of directory records that can be contained on a track is as follows:

IBM 2305-1 Drum Storage	16
IBM 2305-2 Drum Storage	26
IBM 2314 Disk Storage	17
IBM 2319 Disk Storage	17
IBM 3330 Disk Storage	28
IBM 3333 Disk Storage	28
IBM 3340 Disk Storage	16

Table 5. Auxilliary Storage Required for Installing VS BASIC under VM/370(CMS)

Component	Number of Blocks	Cylinders Required	
		2314/2319 Disk	3330/3333 Disk
VS BASIC Processor	241	4	2
HELP Facility and Messages	371	7	4
Conversion Utility	11	1	1
Sample Program	10	1	1
VSBINSTL Installation Procedure	40	1	1
Total (without VSB TXTLIB)	673	14	7
VSB TXTLIB	412	8	4
Total	1085	22	11

Table 6. Auxilliary Storage Required for Installing VS BASIC under DOS/VS

Library	Number of Directory Records[1],[3]	Cylinders Required[1] 2314/ 2319 Disk	3330/ 3333 Disk	3340 Disk	Tracks Required[2] 2314/ 2319 Disk	3330/ 3333 Disk	3340 Disk
SYSRLB or a Private Relocatable Library	1	2	1	2	40	21	23
SYSCLB or a Private Core Image Library	1	1	1	1	18	10	11
SYSSLB or a Private source Statement Library	1	1	1	1	20	11	19
SYS005 (Deblocked Installation File)	1	6	4	7	106	69	124

[1]See the description of the NEWVOL statement in the publication DOS/VS System Control Statements, Order No. GC33-5376.

[2]See the description of the EXTENT statement in the publication DOS/VS System Control Statements, Order No. GC33-5376.

[3]The number of 256-byte directory records that can be contained on one track is as follows:

IBM 2314 Disk Storage	17
IBM 2319 Disk Storage	17
IBM 3330 Disk Storage	28
IBM 3333 Disk Storage	28
IBM 3340 Disk Storage	16

OBTAINING A LISTING OF ALL VS BASIC DIAGNOSTIC MESSAGES

This section describes how to obtain a reference copy of all the VS BASIC messages under OS/VS, CMS, and DOS/VS.

UNDER OS/VS

The OS/VS and OS/VS2(TSO) installation procedures will print out a complete listing of all the VS BASIC messages as part of their normal operation. However, should you, at some time, wish to obtain additional copies, follow this procedure:

1 Mount the VS BASIC distribution tape on a magnetic tape device.

2 Prepare and execute the following procedure:

```
//LISTMSG      JOB  ...
//             EXEC PGM=IEBGENER
//SYSPRINT     DD   SYSOUT=A
//SYSUT1       DD   UNIT=2400,LABEL=(15,NL),DISP=(OLD,PASS),
//                  VOL=(,RETAIN,SER=VSBAS),
//                  DCB=(DEN=density,RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSUT2       DD   SYSOUT=A,DCB=(RECFM=F,BLKSIZE=80)
//SYSIN        DD   DUMMY
/*
```

UNDER CMS

The CMS installation procedure does not print a copy of the error messages for you; therefore, if you want a complete listing as a reference copy, you must issue the following CMS command after the installation is complete:

```
PRINT VSBMSG LIST *
```

UNDER DOS/VS

The DOS/VS installation procedure does not print a copy of the error messages for you; therefore, if you want a complete listing as a reference copy, you must use DITTO. The file-id of the message data set on the distribution disk is 'A5748XX1.SYSIN.V1M1.MESSAGE'.

SYSTEM PROGRAMMING CONSIDERATIONS FOR ALL SYSTEMS USERS

SEPARABLE LIBRARY FACILITY

The Separable Library Facility (SLF) is a feature of the VS BASIC processor that permits a terminal user of system programmer to write his own assembly language routine which can be added to the VS BASIC run-time library as an intrinsic function. At execution-time, this user-written routine will operate like any other VS BASIC intrinsic function, for example, SIN, DAT, or SUM. As a result, frequently used functions need not be repeatedly defined with DEF statements in each program that uses them. With SLF, a function can defined once, installed in the library, and made available to all users. In addition, through SLF, functions not provided by VS BASIC or not possible at the source code level can be used.

Note: As with all user modifications to IBM Program Products, the responsibility for using and maintaining routines written under SLF remains with you, the system programmer. Extreme caution should be exercised in using SLF because of the possibility of adverse effects that errors could have on your system.

Requirements for Writing Routines under SLF

For each new routine or collection of routines to be added to the VS BASIC library, there are four things that must be done:

- Write the routine that is to be added to the library. This routine will contain the code that will actually evaluate the function that is required. The code in the routine must conform to the standards that are outlined in the section "Writing a Function Evaluating Routine under SLF".

- Write a scanning routine that will be added to the compiler. This routine will receive control whenever a 3 character name that has a pound sign (#) as the last character is encountered. Your scan routine must determine whether the name encountered is the name of your intrinsic function and notify the caller. If so, the compiler will then handle it as a valid function reference and process it accordingly. The code in the scanning routine must conform to the standards outlined in the section "Writing a Scanning Routine under SLF".

- Modify the run-time routine (ICDKBFTB) that contains a table of addresses (ICDBIFTB) of all the run-time routines. You must add, through a V-type address constant, the address of the function evaluating routines that you will be adding.

- Reinstall the VS BASIC processor to incorporate the changes that you have made.

Writing a Function Evaluating Routine under SLF

All routines written under SLF that are to be used to evaluate intrinsic functions must conform to the following standards or they will not operate successfully in conjunction with the VS BASIC processor.

ROUTINE NAME: The name of each routine that you will write to evaluate your intrinsic functions must be:

ICDKxx#

where:

xx - is the first two characters of the three character name that will be used in your VS BASIC program as the function reference. The last character must always be a pound sign (#).

SOURCE LANGUAGE FOR YOUR ROUTINE: All routines written under SLF to evaluate intrinsic functions must be written in assembler language. Assembler language is required because there is the need to carefully specify register usage and to resolve addresses through specific base and displacement schemes. Neither of capabilities is available in higher level languages.

REGISTER USAGE: The following registers contain addresses that will be of use to you in writing evaluating routine:

Register	Contents
7	Address of the Array Area Base
8	Address of the User Area Base
11	Address of the User Routine (use this register as your base register)
13	Address of the Run-time Library
14	Address of the Parameter List

Registers 0, 2, 3, 4, 5, and 15 are available as work registers. If additional registers are required, these registers must be saved on entry to your routine and restored before the return is made. This also applies to the floating point registers with the exception of register 0, which is used to return the result of a numeric function.

EVALUATING THE PARAMETERS PASSED BY THE FUNCTION REFERENCE: Naturally, the needs of your application and the function to be performed will determine the number and type of parameters that are to be included with the function reference. In your source program, the parameters must be enclosed by parentheses and separated by commas. The standard compiler routines will process your function reference and its parameters after your scanning routine has informed the compiler that it is a valid function reference. These compiler routines will prepare a parameter list and pass the address of it to your evaluating routine via register 14.

The parameter list is pointed to by register 14 has the following format:

```
                        Parameter List
Register 14         ┌─────────────────────────────────────┐
                    │        Number of Parameters         │
                    ├─────────────────────────────────────┤
                    │ Type and Displacement of Parameter 1│
                    ├─────────────────────────────────────┤
                    └~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~┘

                    ┌~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~┐
                    ├─────────────────────────────────────┤
                    │ Type and Displacement of Parameter n│
                    ├─────────────────────────────────────┤
                    │ Result Entry for Character Functions│
                    │                                     │
                    │ (not present for numeric)           │
                    ├─────────────────────────────────────┤
                    │<──────────────1 word───────────────>│
```

Each parameter in the parameter list that describes the type and displacement of a parameter and the result entry has the following format:

Byte	Contents
0	Argument Code

Bit Value	Meaning
B'00001000'	Character Type
B'00000100'	Array Element
B'00000010'	Array
B'00000001'	Simple Expression or Variable

Note: These bit values may be combined where required (for example, a character array would be B'00001010').

1	Character String Length (if known at compile-time)
2,3	Address of Parameter (in BDDD form, where B is the base register and DDD is the displacement)

The base and displacement (BDDD) has a different meaning for each type of argument code.

Argument Type	Corresponding BDDD
Character Variables and Array Elements	The first byte of the address pointed to by BDDD contains the length of the string.
Array Element	The BDDD is the address of the displacement of the element from the array area base (register 7).
Array	The BDDD is a displacement (B is 0) into the array entry table, which is located at the displacement contained in ARRPTRS INTO THE USER AREA BASE (REGISTER 8). At that location, the array descriptor is 12 bytes long and has the following format:

Byte	Contents
0-3	Offset to array data from data from beginning of array area (register 7).
4,5	Maximum value of the first subscript.

Argument Type	Byte	Contents
	6,7	Maximum value of the second subscript.
	8	Length, in bytes, of each element.
	9	Number of dimensions (0 for a one dimensional array and 1 for a two dimensional array.
	10-11	Maximum number of elements.
Simple Expression or Variable		The BDDD is the address of the data.

WORK AREA: If you want your evaluating routine to be re-entrant, you must put any data fields that will be changed (that is, save areas of flags) into a work space in the user area. This includes any execute instructions that are to be built. The work space is located at the offset contained in the data area LEV0ABS from the user area base (register 8). The work space is 16 words in length. If you are not concerned with having a re-entrant routine your work space may be within your routine.

Note: If you have your compiler installed in the OS/VS link pack area and you write an SLF routine that is not re-entrant, you will not be able to keep the compiler in the link pack area.

DETERMINING WHETHER LONG OR SHORT PRECISION IS REQUIRED: When evaluating numeric arguments or function, your routine must know whether the VS BASIC processor is running with long or short precision. A byte in the user area indicates the current precision. The byte is located at the offset contained in the data area SLBYTE from the user area base (register 8). A value of X'00' indicates short precision and a value of X'08' indicates long precision.

ERROR PROCESSING: The VS BASIC processor has two run-time routines that will handle execution-time errors. Their names are ICDKERRR and ICDKERRT and their addresses can be resolved with an EXTERN or a V-type constant (VCON). ICDKERRR will print an error message and return control to your program; ICDKERRT will print an error message and terminate your program. Any VS BASIC error message can be issued by doing the following:

1. Loading register 0 with the error number.
2. Loading register 2 with the address of your routine name in EBCDIC. (This is required only for messages that contain a variable field in the text.)
3. Branching to ICDKERRR or ICDKERRT.

The error number that is loaded into register 0 is the last two digits of the number in the message identifier. For example, if you want to print the message:

```
ICD414 OVERFLOW
```

and return to your program, load the number 14 into register 0 and branch to ICKDERRR with the following instructions:

```
L         any-register,=V(ICDKERRR)
BALR R    11,any-register
```

Use these instructions for ICDRERRT:

```
     L          any-register,=V(ICDKERRT)
     BALR R     11,any-register
```

VS BASIC PROCESSOR MACROS: SLF makes available VS BASIC Processor macros that are useful in writing your own library routines. These macros provide you with access to the data areas and information used by the VS BASIC Processor during compilation and execution. The information provided includes the format of the Communications Region, the User Terminal Table, the Object Code Area, the Variable and Constant Area, the Branch Information Table, and a list of the equates for the run-time registers. The macros are:

ICDKMAP	- Communications Region, User Terminal Table
ICDOBJA	- Object Code Area
ICDVARCN	- Variable and Constant Area
ICDBIFTB	- Branch Information Table

See the publication System/370 VS BASIC Program Logic, Order No. LY28-6422 for a complete description of these areas and their contents.

RETURNING VALUES FROM THE EVALUATING ROUTINE: The results of a numeric function are returned in floating-point register 0. For character functions, the last word of the parameter list is used to return the result. The format of the result entry is the same as that of the parameter entries. See the description in the section "Evaluating the Parameters Passed by the Function Reference" for a detailed description of the result entry.

RETURNING CONTROL TO THE CALLING ROUTINE: Located immediately following the result entry for character arrays of the last parameter for numeric arrays is the return point in the object code. Point register 14 past the last entry in the parameter list and branch via register 14.

Sample SLF Function Evaluating Routines

Example 1:

This is an example of a VS BASIC SLF, user-written routine to evaluate a character function. The name of the function is BA# and its purpose is to receive a character string contained in a character variable or array element and fill any blanks with asterisks. The result will be returned in the area specified by the last item in the parameter list. The name of the routine is ICDKBA# and it is not re-entrant.

The compiler will process the BA# function whenever it appears in a VS BASIC program and produce object code that will call the routine ICDKBA#. The object code contains the following instructions:

```
     L      R11,=V(ICDKBA#)
     BALR   R14,R11
```

and the argument list:

```
┌─────────────────────────────────┬─────┬─────┐
│                                 │  1  │     │
├─────┬─────┬──────────┬─────┬────┼─────┼─────┤
│  0  │ 91C │          │  B     D │  D    D1  │
├─────┴─────┴──────────┼─────┼────┼─────┼─────┤
│                      │  B     D │  D    D2  │
├──────────────────────┴─────┴────┴─────┴─────┤
|<-----------------4 bytes------------------>|
```

where:

1 - is the number of arguments

9 - indicates that the argument is a character variable

C - indicates that the argument is a character array element

$BDDD_1$ - is the displacement of the actual argument if scalar or the displacement of an array element displacement if the argument is an array element.

$BDDD_2$ - is the displacement of the area in which the return value is to be placed. This is the character variable to which the function is assigned.

The actual argument has the following format:

```
┌─────┬─────┬─────────────────────────────────────────────────────┐
│ L     L   │                   Character String                  │
└─────┴─────┼─────────────────────────────────────────────────────┤
            │<---------------------LL bytes---------------------->│
```

The array element displacement has the following format:

```
┌─────┬─────┬─────┬─────┬─────┬─────┬─────┬─────┐
│ L     L   │ D     D     D     D     D     D   │
├─────┴─────┴─────┴─────┴─────┴─────┴─────┴─────┤
│<------------------4 bytes-------------------->│
```

where:

LL - is the length of the character string.

DDDDDD - is the displacement into the array area of the actual array element. The contents of register 7 (R7) must be added to this value.

The assembler language code for the routine ICDKBA# follows:

```
ICDKBA#   CSECT
          USING   ICDKBA#,R11
          CLC     0(1,R14),ONE          Is there only one argument?
          BNE     ERR1                  If not, print an error message.
          TM      4(R14),X'08'          Is it a character argument?
          BNE     ERR2                  If not, print an error message.
          MVC     ICDKB1+2(2),6(R14)    Develop the address of the
ICDKB1    LA      R2,0(0)               argument.
          MVC     ICDKB2+2(2),10(R14)   Develop the address of the return
ICDKB2    LA      R3,0(0)               area.
          SR      R4,R4
          IC      R4,0(R2)              Obtain the length of the argument.
          TM      4(R14),X'04'          Is the argument an array element?
          BNO     ICDKBN3               If not, skip array element processing.
          L       R2,0(R2)              Obtain array displacement address.
          LA      R2,0(R2,R7)           Develop address of array element.
          B       ICDKB4
ICDKB3    LA      R2,1(R2)              Skip over length indicator.
ICDKB4    STC     R4,0(R3)              Place the length in the return area.
ICDKB5    LA      R3,1(R3)              Locate the character to be moved.
          MVC     0(R3,1),0(R2)         Move the character.
          CLI     0(R3),C' '            Is it a blank?
          BNE     ICDKB5                If not, branch to get next character.
          MVI     0(R3),C'*'            If blank, substitute an asterisk.
          LA      R2,1(R2)              Locate next character to be moved.
          BCT     R4,ICDKB5             Are there more characters to move?
          B       12(R14)               If not, return to the caller.
ERR1      LA      R0,41                 Locate message for incorrect number
          B       ERR3                  of arguments.
ERR2      LA      R0,42                 Locate message for incorrect type.
ERR3      LA      R2,ROUTNAME           Indicate the name of this function.
          L       R11,=V(ICDKERRT)      Branch to the run-time error
          BR      R11                   routine to terminate execution.
ONE       DC      F'1'                  The constant one.
ROUTNAME  DC      CL4'BA# '             The name of this function in EBCDIC.
R0        EQU     0                     Equate for register 0.
R2        EQU     2                     Equate for register 2.
R3        EQU     3                     Equate for register 3.
R4        EQU     4                     Equate for register 4.
R7        EQU     7                     Equate for register 7.
R11       EQU     11                    Equate for register 11.
R14       EQU     14                    Equate for register 14.
          END
```

Example 2:

This is an example of a VS BASIC SLF, user-written routine to evaluate a numeric function. The name of the function is SQ# and its purpose is to receive two numbers contained in numeric variables or array elements and calculate the sum of their squares. The result is returned in register 0. The name of the routine will be ICDKSQ# and it is to be re-entrant.

The compiler will process the SQ# function whenever it appears in a VS BASIC program and produce object code that will call the routine ICDKSQ#. The object code contains the following instruction:

```
L     R11,=V(ICDKSQ#)
BALR  R14,R11
```

and the argument list:

```
┌──────────────────────────────────┬─────┬─────┐
│                                  │  2  │     │
├─────┬─────┬──────────┬─────┬─────┼─────┼─────┤
│  0  │ 1/2 │          │  B     D     D     D1 │
├─────┼─────┼──────────┼─────┼─────┼─────┼─────┤
│  0  │ 1/2 │          │  B     D     D     D2 │
├─────┴─────┴──────────┴─────┴─────┴─────┴─────┤
|<-----------------4 bytes------------------->|
```

where:

2 - is the number of arguments

1 - indicates that the argument is a numeric variable

2 - indicates that the argument is a numeric array element

$BDDD_1$ - is the displacement of the actual argument if scalar or the displacement of an array element displacement for the first argument

$BDDD_2$ - is the displacement of the actual argument is scalar or the displacement of an array element displacement for the second argument

Each actual argument may have one of the following formats:

Short Precision

```
┌────┬────┬────┬────┬────┬────┬────┬────┐
│                 Data                  │
├────┴────┴────┴────┴────┴────┴────┴────┤
|<--------------4 bytes---------------->|
```

Long Precision

```
┌────┬────┬────┬────┬────┬────┬────┬────┐
│                 Data                  │
├────┼────┼────┼────┼────┼────┼────┼────┤
│                 Data                  │
├────┴────┴────┴────┴────┴────┴────┴────┤
|<--------------4 bytes---------------->|
```

The array displacement has the following format:

```
┌───────────┬─────┬────┬─────┬─────┬─────┬─────┐
│           │ D     D    D     D     D     D   │
├───────────┴─────┴────┴─────┴─────┴─────┴─────┤
|<-----------------4 bytes------------------->|
```

where:

DDDDDD - is the displacement into the array area of the actual array element. The contents of register 7 (R7) must be added to this value.

The assembler language code for the routine ICDKSQ# follows:

```
ICDKSQ#  CSECT
         USING    PRG,R8                 ESTABLISH ADDRESSABILITY FOR PRG.
         USING    ICDKSQ#,R11            ESTABLISH ADDRESSABILITY.
         STD      FR2,SV0FLT2            SAVE REGISTER 2.
         CLC      0(4,R14),TWO           ARE THERE ONLY TWO ARGUMENTS?
         BNE      ERR1                   IF NOT, PRINT AN ERROR MESSAGE.
         TM       4(R14),X'08'           IS IT A CHARACTER ARGUMENT?
         BE       ERR2                   IF NOT, PRINT AN ERROR MESSAGE.
         TM       8(R14),X'08'           IS THE SECOND CHARACTER?
         BE       ERR2                   IF NOT, PRINT AN ERROR.
         MVC      SV0FLT4(2),LA2         SET UP AN INSTRUCTION TO DEVELOP
         MVC      SV0FLT4+2(2),6(R14)      THE ADDRESS OF THE FIRST ARGUMENT.
         EX       R0,SV0FLT4
         TM       4(R14),X'04'           IS THE ARGUMENT AN ARRAY ELEMENT?
         BNO      ICDKA1                 IF NOT, SKIP ARRAY PROCESSING.
         L        R2,0(R2)               OBTAIN ARRAY DISPLACEMENT ADDRESS.
         LA       R2,0(R2,R7)            DEVELOP ACTUAL ADDRESS OF ELEMENT.
ICDKA1   MVC      SV0FLT4(2),LA3         SET UP AN INSTRUCTION TO DEVELOP
         MVC      SV0FLT4+2(2),10(R14)     THE ADDRESS OF THE 2ND ARGUMENT.
         EX       R0,SV0FLT4
         TM       4(R14),X'04'           IS THE ARGUMENT AN ARRAY ELEMENT?
         BNO      ICDKA2                 IF NOT, SKIP ARRAY PROCESSING.
         L        R3,0(R3)               OBTAIN ARRAY DISPLACEMENT ADDRESS.
         LA       R3,0(R3,R7)            DEVELOP ACTUAL ADDRESS OF ELEMENT.
ICDKA2   TM       SLBTYPE,DLPREC         IS PROCESSOR IN LONG PRECISION MODE?
         BO       ICDKA3                 IF SO, CALCULATE IN LONG PRECISION.
         LE       FR0,0(R2)              OBTAIN VALUE OF THE FIRST ARGUMENT.
         MER      FR0,FR0                SQUARE THAT VALUE.
         LE       FR2,0(R3)              OBTAIN VALUE OF THE SECOND ARGUMENT.
         MER      FR2,FR2                SQUARE THAT VALUE.
         AER      FRO,FR2                ADD, LEAVING RESULT IN REGISTER 0.
         LE       FR2,SV0FLT2            RESTORE REGISTER 2.
         B        12(R14)                RETURN TO THE CALLER.
ICDKA3   LD       FR0,0(R2)              OBTAIN VALUE OF THE FIRST ARGUMENT.
         MDR      FR0,FR0                SQUARE THAT VALUE.
         LD       FR2,0(R3)              OBTAIN VALUE OF THE SECOND ARGUMENT.
         MDR      FR2,FR2                SQUARE THAT VALUE.
         ADR      FR0,FR2                ADD, LEAVING RESULT IN REGISTER 0.
         LD       FR2,SV0FLT2            RESTORE REGISTER 2.
         B        12(R14)                RETURN TO THE CALLER.
ERR1     LA       R0,41                  LOCATE MESSAGE FOR INCORRECT NUMBER
         B        ERR3                     OF ARGUMENTS.
ERR2     LA       R0,42                  LOCATE MESSAGE FOR INCORRECT TYPE.
ERR3     LA       R2,ROUTNAME            INDICATE THE NAME OF THIS FUNCTION.
         L        R11,=V(ICDKERRT)       BRANCH TO THE RUN-TIME ERROR
         BR       R11                      ROUTINE TO TERMINATE EXECUTION.
TWO      DC       F'2'                   THE CONSTANT TWO.
ROUTNAME DC       CL4'SQ# '              THE NAME OF THIS FUNCTION IN EBCDIC.
R0       EQU      0                      EQUATE FOR REGISTER 0.
R2       EQU      2                      EQUATE FOR REGISTER 2.
R3       EQU      3                      EQUATE FOR REGISTER 3.
R7       EQU      7                      EQUATE FOR REGISTER 7.
R8       EQU      8                      EQUATE FOR REGISTER 8.
R11      EQU      11                     EQUATE FOR REGISTER 11.
R14      EQU      14                     EQUATE FOR REGISTER 14.
FR0      EQU      0                      EQUATE FOR FLOATING REGISTER 0.
FR2      EQU      2                      EQUATE FOR FLOATING REGISTER 2.
         ICDKMAP                         ISSUE MACRO FOR ICDKMAP.
PRG      DSECT                           LOCATE SAVE AREA IN COMMUNICATIONS
         ORG      LEV0ABS                  REGION FOR RE-ENTRANT PROGRAMS.
SV0FLT2  DS       D                      SAVE AREA FOR FLOATING REGISTER 2.
SV0FLT4  DS       D                      SAVE AREA FOR FLOATING REGISTER 4.
         END
```

Example 3 (CMS only):

This an example of a VS BASIC SLF, user-written routine to evaluate a character function. The name of the function is RD# and its purpose is to receive the name of a CMS file with a filetype of VSBREC and the name of an 80-byte variable and read one record from the file into the variable. It will return a code in register 0 indicating whether the read was successful. The name of the routine is ICDKRD#.

The compiler will process the RD# function whenever it appears in a VS BASIC program and produce object code that will call the routine ICDKRD#. The object code contains the following instructions:

```
     L      R11,=V(ICDKRD#)
     BALR   R14,R11
```

and the argument list:

```
┌──────────────────────────────────────┬─────┬─────┐
│                                      │  2  │     │
├─────┬─────┬──────────┬─────┬─────────┼─────┼─────┤
│  0  │  9  │          │  B  │    D    │  D  │ D1  │
├─────┼─────┼──────────┼─────┼─────────┼─────┼─────┤
│  0  │  9  │          │  B  │    D    │  D  │ D2  │
├─────┴─────┴──────────┴─────┴─────────┴─────┴─────┤
|<─────────────────4 bytes───────────────────────>|
```

where:

2 - is the number of arguments

9 - indicates that the argument is a character variable

$BDDD_1$ - is the displacement of the first actual argument

$BDDD_2$ - is the displacement of the second actual argument

The actual arguments have the following format:

First Argument

```
┌───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┐
│ 0 │ 8 │                      Filename                               │
└───┴───┼───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘
        |<────────────────────8 bytes─────────────────────────────────>|
```

Second Argument

```
┌───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┐   ┌───┬───┬───┐
│ 5 │ 0 │        Variable for 80-byte Record          }   {           │
└───┴───┼───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘   └───┴───┴───┘
        |<────────────────────80 bytes────────────────────────────────>|
```

The assembler language code for the routine ICDKRD# follows:

```
ICDKRD#   CSECT
          USING    ICDKRD#,R11           ESTABLISH ADDRESSABILITY.
          LR       R4,R1                 SAVE REGISTER 1, USED BY I/O.
          L        R15,4(R14)            OBTAIN FILENAME DESCRIPTOR.
          STH      R15,RESOLVE+2         SET UP AN INSTRUCTION TO DEVELOP
          EX       0,RESOLVE               THE ADDRESS OF THE FIRST ARGUMENT.
          MVC      P1,1(R2)              PUT THE FILENAME INTO THE PARM LIST.
          L        R15,8(R14)            OBTAIN VARIABLE DESCRIPTOR.
          STH      R15,RESOLVE+2         SET UP AN INSTRUCTION TO DEVELOP
          EX       0,RESOLVE               THE ADDRESS OF THE 2ND ARGUMENT.
          LA       R2,1(R2)              SKIP OVER STRING LENGTH INDICATOR.
          LA       R15,PLIST             LOCATE THE READ PARAMETER LIST.
          FSREAD (R15),BUFFER=(R2),BSIZE=80,ERROR-ERRET   READ THE RECORD
          SDR      R0,R0                 INDICATE THAT THE READ IS SUCCESSFUL
          LR       R1,R4                 RESTORE REGISTER 1.
          B        12(R14)               RETURN TO THE CALLER.
ERRET     LD       R0,ONE                INDICATE THAT READ FAILED.
          LR       R1,R4                 RESTORE REGISTER 1.
          B        12(R14)               RETURN TO THE CALLER.
ONE       DC       D'1'                  THE CONSTANT ONE.
PLIST     DS       0D                    THE READ PARAMETER LIST.
P1        DC       CL8'  '
          DC       CL8'VSBREC'
          DC       CL2'A1'
RESOLVE   LA       R2,0                  INSTRUCTION TO BE EXECUTED.
R0        EQU      0                     EQUATE FOR REGISTER 0.
R1        EQU      1                     EQUATE FOR REGISTER 1.
R2        EQU      2                     EQUATE FOR REGISTER 2.
R4        EQU      4                     EQUATE FOR REGISTER 4.
R11       EQU      11                    EQUATE FOR REGISTER 11.
R14       EQU      14                    EQUATE FOR REGISTER 14.
R15       EQU      15                    EQUATE FOR REGISTER 15.
          END
```

Note: This program uses CMS macros and does not do any error checking.

Example 4 (CMS only):

This is an example of a VS BASIC SLF, user-written routine to evaluate a character function. The name of the function is WR# and its purpose is to receive a CMS filename and the name of an 80-byte character variable containing data. The routine will write the 80-byte variable into a CMS file that it will create with the filename indicated and assign it a filetype of VSBREC. It will return a code in register 0 indicating whether the read was successful. The name of the routine is ICDKWR#.

The compiler will process the WR# function whenever it appears in a VS BASIC program and produce object code that will call the routine ICDKWR#. The object code contains the following instructions:

```
L      R11,=V(ICDKWR#)
BALR   R14,R11
```

and the argument list:

```
┌─────────────────────────────────────────┬─────┬─────┐
│                                         │  2  │     │
├─────┬─────┬───────────┬─────┬─────┼─────┼─────┤
│  0  │ 9   │           │  B     D     D     D1 │
├─────┼─────┼───────────┼─────┼─────┼─────┼─────┤
│  0  │ 9   │           │  B     D     D     D2 │
├─────┴─────┴───────────┴─────┴─────┴─────┴─────┤
|<-----------------4 bytes------------------->|
```

where:

2 - is the number of arguments

9 - indicates that the argument is a character variable

BDDD$_1$ - is the displacement of the first actual argument

BDDD$_2$ - is the displacement of the second actual argument

The actual arguments have the following format:

First Argument

```
┌───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┬───┐
│ 0   8 │       Filename of CMS File to be Created                      │
└───┴───┼───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┴───┘
        |<--------------------8 bytes---------------------------------->|
```

Second Argument

```
┌───┬───┬───┬───┬───┬───┬───┬───┬─────────┬───┬───┬───┐   ┌───┬───┬───┬───┐
│ 5   0 │       Variable Containing an 80-byte Record  }   {               │
└───┴───┼───┴───┴───┴───┴───┴───┴─────────┴───┴───┴───┘   └───┴───┴───┴───┘
        |<--------------------80 bytes-------------------------------------->|
```

The assembler language code for the routine ICDKWR# follows:

```
ICDKWR#   CSECT
          USING    ICDKWR#,R11          ESTABLISH ADDRESSABILITY.
          LR       R4,R1                SAVE REGISTER 1, USED BY I/O.
          L        R15,4(R14)           OBTAIN FILENAME DESCRIPTOR.
          STH      R15,RESOLVE+2        SET UP AN INSTRUCTION TO DEVELOP
          EX       0,RESOLVE              THE ADDRESS OF THE FIRST ARGUMENT.
          MVC      P1,1(R2)             PUT THE FILENAME INTO THE PARM LIST.
          L        R15,8(R14)           OBTAIN VARIABLE DESCRIPTOR.
          STH      R15,RESOLVE+2        SET UP AN INSTRUCTION TO DEVELOP
          EX       0,RESOLVE              THE ADDRESS OF THE 2ND ARGUMENT.
          LA       R2,1(R2)             SKIP OVER STRING LENGTH INDICATOR.
          LA       R15,PLIST            LOCATE THE WRITE PARAMETER LIST.
          FSWRITE (R15),BUFFER=(R2),BSIZE=80,ERROR=ERRET   WRITE A RECORD.
          SDR      R0,R0                INDICATE THAT THE WRITE IS SUCCESSFUL.
          LR       R1,R4                RESTORE REGISTER 1.
          B        12(R14)              RETURN TO THE CALLER.
ERRET     LD       R0,ONE               INDICATE THAT THE WRITE FAILED.
          LR       R1,R4                RESTORE REGISTER 1.
          B        12(R14)              RETURN TO THE CALLER.
ONE       DC       D'1'                 THE CONSTANT ONE.
PLIST     DS       0D                   THE WRITE PARAMETER LIST.
P1        DC       CL8' '
          DC       CL8'VSBREC'
          DC       CL2'A1'
RESOLVE   LA       R2,0                 INSTRUCTION TO BE EXECUTED.
R0        EQU      0                    EQUATE FOR REGISTER 0.
R1        EQU      1                    EQUATE FOR REGISTER 1.
R2        EQU      2                    EQUATE FOR REGISTER 2.
R4        EQU      4                    EQUATE FOR REGISTER 4.
R11       EQU      11                   EQUATE FOR REGISTER 11.
R14       EQU      14                   EQUATE FOR REGISTER 14.
R15       EQU      15                   EQUATE FOR REGISTER 15.
          END
```

Note: This program uses CMS macros and does not do any error checking.

Writing a Scanning Routine under SLF

All routines written under SLF that are to be used to scan function names must conform to the following standards or they will not operate successfully in conjunction with the compiler portion of the VS BASIC processor.

ROUTINE NAME: The name of the routine that you will write to scan all three character names ending in a pound sign must be:

ICDJUSTB

This is the name the compiler has been designed to check for whenever an appropriate name has been encountered in your source program.

REGISTER USAGE: The following registers contain addresses that will be of use to you in writing your scanning routine:

Register	Contents
6	Address of the User Routine (use this register as your base register)
10	Address of the Return Point
15	Address of the 3-character Name to be Checked

Registers 2 and 3 are available as work registers. There is no need to save and restore them. However, if additional registers are required, they must be saved and restored. This should be avoided since your scanning routine would not be re-entrant. Register 9 is the register to be used to return an indication to the compiler of the results of the search.

DETERMINING THE SIZE OF ICDKBFTB: Insert at the end of your program the macro ICDBIFTB, which is required to calculate the entry number in BIFTAB for your function name. This macro must be followed by an equate that will use the addresses contained in the first and last entries in BIFTAB to calculate the number of entries in the table.

Example:

```
                .
                .
                .
                ICDBIFTB
      ENTNUM    EQU     (LCSTEND-LCSTART)/4
                END
```

CREATING AN INTERANL TABLE TO IDENTIFY YOUR INTRINSIC FUNCTION ROUTINES: To identify each of your intrinsic functions that were written under SLF, you must create an internal table that contains one entry for each routine that you created. The entries are placed into the table sequentially. Each entry has the following format:

Internal Function Table Entry

Function Name (in EBCDIC)	Reserved	Function Type	Flags	Not Used	Entry Number
0	3	4	5	6	7

Byte	Contents
0-2	Name of the function in EBCDIC. This name must be 3 characters in length with a pound sign (#) as the last character.
3	Reserved
4	Function type: if the function is numeric this byte should be X'00'; if the function is character this byte should be X'08'. All other bits should be turned off.
5	Function flags:

Bit Values	Meaning
B'00110000'	A parameter is required
B'00100100'	A parameter is not required
B'00100000'	A parameter is optional
B'00110001'	An array is required as a parameter
B'00100001'	An array may optionally be used as a parameter

Byte	Contents
6	Not used. You may use this byte for your search of the table, if you are using a hasing algorithm.
7	Entry number in the branch information table (ICDBIFTB) of the entry for this function. The first entry number available is determined by computing ENTNUM(LCLSTEND-LCSTART)/4. This corresponds to adding a V-type address constant for your routine to the end of the execution-table. Each succeeding entry should be kept in sequential order (that is, ENTNUM is the number of the first address added, ENTNUM + 1 is the number of the second address added, ENTNUM + 2 is the number of the third address added, and so on). Refer to the sample program in the section "Sample for SLF Scanning Routine" for more information.

SEARCHING YOUR INTERNAL FUNCTION TABLE: The easiest method for scanning your internal function table is a sequential search. However, as you add more routines and the internal table gets larger, the time required to compiler a program will increase. The additional time is required because this your scanning routine is called every time an appropriate three-character name is encountered. Therefore, if you notice that your compile-time has increased considerably, you might want to write a more sophisticated search routine using some type of hashing algorithm.

<u>OBTAINING THE FUNCTION NAME TO BE CHECKED FOR VALIDITY</u>: The compiler will place the address of the three-byte name to be checked in register 15.

<u>NOTIFYING THE COMPILER OF THE RESULTS OF THE SEARCH</u>: After you have searched your internal table for the name that the compiler passed to your scan routine, you must indicate to the compiler whether the name is valid. If the name is one of your intrinsic function, place the address of the corresponding entry in your internal function table in register 9. If the name is not valid set register 9 to zeros. Finally, return control to the caller with a branch to register 10.

Sample SLF Scanning Routine

When the compiler encounters a reference to a three character name that has a pound sign (#) as the last character it calls a routine called ICDJUSTB. ICDJUSTB is the user-written scanning routine that will determine if the user function name is valid under SLF. The compiler passes ICDJUSTB the name to be checked in register 15. The entries in an Internal Function Table are checked for a match and the address of the corresponding user-written function evaluating routine is returned to the compiler in register 9.

The assembler language code for an ICDJUSTB that will check for the routines ICDKBA#, ICDKSQ#, ICDKRD#, and ICDKWR# follows:

```
ICDJUSTB CSECT
         USING    ICDJUSTB,R6          Establish addressability.
         USING    INPUT,R15            Establish addressability for input.
         LA       R9,FCTTBL            Get address of your function table.
         L        R3,NUMENT            Get number of entries in table.
LOOP     EQU      *                    Beginning of table scanning loop.
         CLC      NAME,0(R9)           Is the name passed in the table.
         BE       RETURN               If so, return to the caller.
         LA       R9,8(R9)             If not, continue scanning table.
         BCT      R3,LOOP              Branch to compare next entry.
         SR       R9,R9                If name is not found, zero register 9
RETURN   BR       R10                  Return to the calling routine.
FCTTBL   DS       0D                   Start of your SLF function table.
         DC       CL3'BA#'             Name of function BA#.
         DS       X
         DC       XL1'08'              Type of function BA#.
         DC       XL1'30'              Function codes for BA#.
         DS       X
         DC       AL1(ENTNUM)          ICDKBFTB entry number for BA#.
         DC       CL3'SQ#'             Name of function SQ#.
         DS       X
         DC       XL1'00'              Type of function SQ#.
         DC       XL1'30'              Function codes for SQ#.
         DS       X
         DC       AL1(ENTNUM+1)        ICDKBFTB entry number for SQ#.
         DC       CL3'RD#'             Name of function RD#.
         DS       X
         DC       XL1'00'              Type of function RD#.
         DC       XL1'30'              Function codes for RD#.
         DS       X
         DC       AL1(ENTNUM+2)        ICDKBFTB entry number for RD#.
         DC       CL3'WR#'             Name of function WR#.
         DS       X
         DC       XL1'00'              Type of function WR#.
         DC       XL1'30'              Function codes for WR#.
         DS       X
         DC       AL1(ENTNUM+3)        ICDKBFTB entry number for WR#.
ENDTBL   EQU      *                    End of your SLF function table
NUMENT   DC       A((ENDTBL-FCTTBL)/8) Number of functions in your table.
R3       EQU      3                    Equate for register 3.
R6       EQU      6                    Equate for register 6.
R9       EQU      9                    Equate for register 9.
R10      EQU      10                   Equate for register 10.
R15      EQU      15                   Equate for register 15.
INPUT    DSECT
NAME     DS       CL3                  Three-byte input name to be checked.
         ICDBIFTB                      Issue the macro ICDBIFTB.
ENTNUM   EQU      (LCLSTCND-LCSTART)/4 Next available entry number.
         END
```

Modifying the Branch Information Table (ICDBIFTB) and Reassembling the Run-Time Routine ICDKBFTB

The branch information table (ICDBIFTB) must be updated to include V-type address constants for each of the new intrinsic functions that you wish to add. The source code for the module ICDKBFTB, which contains the branch information table, has been provided to you in the form of a card deck. V-type address constants for your routines must be placed after LCLSTEND in ICDKBFTB. Make sure that the order of the address constants corresponds to the entry numbers that were assigned in the scanning routine ICDJUSTB.

Sample SLF Branch Information Table (ICDKBFTB) Modifications

To locate the new library routines that were shown in the preceding examples, the following V-type address constants must be added to ICDKBFTB just before the END statement in the order in which they appear in the scanning routine ICDJUSTB:

```
     DC    V(ICDKBA#)
     DC    V(ICDKSQ#)
     DC    V(ICDKRD#)
     DC    V(ICDKWR#)
```

Example:

```
     ICDKBFTB  CSECT
               ICDSIFTB TYPE=CSECT
     LCSTART   DS     0F
     LCKORGE   DC     V(ICKDORGE)
     LCKATTN   DC     V(ICDKATTN)
     LCKINTP   DC     V(ICDKINTP)
               .
               .
               .
     LCKVEXT   DC     V(ICDKVEXT)
     LCKVXTR   DC     V(ICDKVXTR)
     LCKOPN1   DC     V(ICDKIPN1)
     LCLSTEND  DS     0F
               DC     V(ICDKBA#)
               DC     V(ICDKSQ#)
               DC     V(ICDKRD#)
               DC     V(ICDKWR#)
               END
```

Installing Your SLF Modules

UNDER OS/VS: The procedure descirbed in this section will install your SLF modules into the VS BASIC Processor under OS/VS.

1 Write the required routines (ICDKxx# and ICDJUSTB).

2 Either punch the contents of file 18 or using IEBUPDTE place its contents on a disk. File 18 contains the source for ICDKBFTB.

3 Modify ICDKBFTB.

4 Using IEBUPDTE, place the contents of file 17 into SYS1.MACLIB or a private macro library. File 17 contains the macros required for assembling the SLF modules.

5 Assembly ICDKxx#, ICDJUSTB, and ICDKBFTB.

After all the necessary routines have been written and assembled, it is necessary to link edit them into the VS BASIC processor. You can use the following JCL procedure to accomplish this under OS/VS:

```
A  //REP        JOB      accounting-information,MSGLEVEL=(1,1)
   //STEP1      EXEC     PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',
   //                    REGION=128K
   //SYSPRINT   DD       SYSOUT=A,SPACE=(121,(1000,50),RLSE)
B  //SYSUT1     DD       DSN=&&any-name,UNIT=SYSDA,
   //                    SPACE=(1500,(35,5),,,ROUND)
C  //SYSLMOD    DD       DSN=new-library-name,DISP=(NEW,CATLG),
   //                    UNIT=unit,VOL=SER=serial-number,
   //                    SPACE=(TRK,(60,1,1))
   //SYSLIN     DD       *
                .
                .
                .
    Object Code for ICDKBFTB
                .
                .
                .
                NAME     ICDBFTB(R)
                .
                .
                .
    Object Code for ICDKxx#
                .
                .
                .
                NAME     ICDKxx#(R)
                .
                .
                .
    Object Code for ICDJUSTB
                .
                .
                .
                NAME     ICDJUSTB
   /*
   //STEP2      EXEC     PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',
   //                    REGION=128K
   //SYSPRINT   DD       SYSOUT=A
B  //SYSUT1     DD       DSN=&&any-name,UNIT=SYSDA,
   //                    SPACE=(1500,(35,5),,,ROUND)
D  //SYSLIB     DD       DSN=old-library-name,DISK=SHR
C  //SYSLMOD    DD       DSN=new-library-name,DISP=SHR
   //SYSLIN     DD       *
                INCLUDE  SYSLIB(ICDJCOMP)
                INCLUDE  SYSLIB(ICDJNUCL)
                INCLUDE  SYSLIB(ICDJNUC1)
                INCLUDE  SYSLIB(ICDJNUC2)
                INCLUDE  SYSLIB(ICDJNUC3)
                INCLUDE  SYSLIB(ICDJNUC4)
                INCLUDE  SYSLIB(ICDJNUC5)
                INCLUDE  SYSLIB(ICDJDEFR)
                INCLUDE  SYSLIB(ICDJRUNA)
                INCLUDE  SYSLIB(ICDJCMPA)
                INCLUDE  SYSLIB(ICDJERR)
                INCLUDE  SYSLIB(ICDJMATV)
                INCLUDE  SYSLIB(ICDJFUTS)
                INCLUDE  SYSLIB(ICDFINFO)
                INCLUDE  SYSLIB(ICDJIOVB)
                INCLUDE  SYSLIB(ICDJVERB)
                INCLUDE  SYSLIB(ICDUSFN)
                INCLUDE  SYSLMOD(ICDJUSTB)
                INCLUDE  SYSLIB(ICDJVREC)
                INCLUDE  SYSLIB(ICDJDUMY)
                NAME     ICDJCOMP(R)
   /*
   //STEP3      EXEC     PGM=IEWL,PARM='LIST,MAP',COND=(8,LE),
   //                    REGION=128K
   //SYSPRINT   DD       SYSOUT=A,SPACE=(121,(1000,50),RLSE)
B  //SYSUT1     DD       DSN=&&any-name,UNIT=SYSDA,
   //                    SPACE=(1500,(35,5),,,ROUND)
C  //SYSLMOD    DD       DSN=new-library-name,DISP=(OLD,PASS),
   //                    VOL=SER=serial-number,
E  //SYSLIB     DD       DSN=new-library-name
   //           DD       DSN=old-library-name,DISP=SHR
   //SYSLIN     DD       *
                INCLUDE  SYSLIB(ICDKBFTB,ICDKERR,ICDKGSUB,
                         ICDKDSUB,ICDKSSUB)
                INCLUDE  SYSLIB(ICDKMAT,ICDKVIOR)
                NAME     ICDKRTNS(R)
   /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B Supply any appropriate name for the utility data set. The same name can be used for both statements.

C This statement defines the new SLF library in which you wish to place the SLF modules. Supply the following information:

new-library-name - is the name of the lbirary into which you wish to install the SLF version of the VS BASIC compiler. You may specify either SYS1.LINKLIB or a private library.

unit - indicates the direct-access on which the new library will be placed.

serial-number - indicates the volume serial number of the volume on which the VS BASIC compiler resides.

Note: This procedure assumes that the output from your assembly of ICDJUSTB, ICDKBFTB, and ICDKxx# will be in the form of a card deck. If you have placed the object code for these modules onto disk or tape you must substitute in the SYSLIN DD statement the appropriate parameters to define the tape or disk. You may use your old library, which contains the VS BASIC modules, however, remember that ICDKBFTB, ICDJUSTB, and VS BASIC load modules will be replaced.

D This statement refers to the old library containing the VS BASIC compiler modules.

E This statement refers to the new library that will contain the modified ICDKBFTB and ICDKxx#. It is concatenated with the old library from which it will obtain the remainder of the required object modules.

UNDER CMS: The procedure described in this section will install your SLF modules into the VS BASIC Processor under CMS.

1 Mount the distribution tape on any available magnetic tape device.

2 Log onto VM/370 with a user identification that has been assigned a privilege class of B.

3 Attach a real tape device to your user identification. The device must be attached at virtual address 181. Use the following command:

```
attach cuu to userid as 181
```

where:

cuu - is the channel and unit address of the actual tape device.

userid - is the user identification that you logged on with in step 2 .

4 IPL CMS.

5 Access the system disk (S) as your A disk. Use the following command:

```
access 191 a
```

6 Place the last two files on the distribution tape into CMS files. File 17 is to be split into macro files and file 18 is to be placed into an assembler language source file. Use the following commands (the system responses have been included):

```
tape fsf 16
R;

tappds * macro a1 (update col1)
R;

tappds * assemble a1 (update col1)
R;
```

Note: The TAPPDS * MACRO command will create 12 files. They are:

```
ICDKMAP   MACRO A1
ICDOBJA   MACRO A1
ICDVARCN  MACRO A1
ICDBIFTB  MACRO A1
ICDKPRGA  MACRO A1
ICDKREG   MACRO A1
ICDGBIF1  MACRO A1
ICDSET1   MACRO A1
ICDKMSG   MACRO A1
PRG       MACRO A1
Z#UTT     MACRO A1
Z#COMM    MACRO A1
```

The TAPPDS * ASSEMBLE command will create the file:

```
ICDKBFTB ASSEMBLE A1
```

7 Create a macro library. Use the following command:

```
maclib gen vsbmac icdkmap prg icdojba icdvarcn
     z#utt icdbiftb icdkprga icdkreg icdgbifl icdsetl
     icdkmsg z#comm
```

Note: To type the pound sign (#) you must precede it with a double quote ("), (for example, z"#comm).

8 Edit the source file for ICDKBFTB to include the V-type address constants for your new library routines.

9 Make the macro library that you just created available. Issue the following command:

```
global maclib vsbmac
```

10 Assemble the file containing ICDKBFTB.

11 Create CMS files containing the assembler language source code for your new library routines and for the scanning routine ICDJUSTB. Assemble these files.

12 Create a text library that contains the text files of your new library routines, the new ICDKBFTB and the text file of your scanning routine ICDJUSTB. Use the following command:

```
txtlib gen vsbnew icdkustb icdkbftb icdkxx# ...
```

13 Issue the following command to regenerate the VS BASIC modules:

```
exec vsbinstl regen
```

Note: This command will replace the old VS BASIC modules, therefore, if you wish to retain the old modules you must rename them temporarily.

From this point on the regeneration procedure will prompt you for responses. The procedure types out the following message:

```
REGEN OF THE VS BASIC MODULES
```

A Ensure that the system disk (S) has been accessed as your A disk. The following message will be typed at your terminal:

```
THE SYSTEM DISK SHOULD BE ACCESSED IN READ/WRITE STATUS
AS THE 'A' DISK.  IF NOT, ENTER 'END', REACCESS THE
SYSTEM DISK IN THE PROPER STATUS AND REISSUE THIS EXEC
AGAIN.

IF IT IS ACCESSED AS THE READ/WRITE 'A' DISK, PRESS
RETURN.
```

You must respond with one of the following:

```
end
CR
```

If the system disk is not access as your read/write A disk, enter END. The following response will be typed at your terminal:

```
'EXIT FOR SYSTEM DISK ACCESS'
```

Do 5 , and repeat step 13 from the beginning. If the system disk is correctly accessed, the procedure continues.

B Enter the names to be used for the regeneration. The following message will be typed at your terminal:

```
ENTER THE TXTLIB NAMES TO BE SEARCHED DURING THE REGEN OF
THE COMPILER (TO A MAXIMUM OF 7)

IF ONLY THE TXTLIB 'VSB' AS CREATED AT INSTALLATION TIME
IS TO BE SEARCHED, PRESS THE RETURN KEY.
```

At this point enter the name of your SLF txtlib followed by VSB (for example, slf vsb).

C After the VS BASIC modules have been regenerated, the following message is typed and the regeneration procedure is completed:

```
          INSTALLATION/REGEN COMPLETE
          R;
```

UNDER DOS/VS: The required macros were automatically placed in your source statement library, in EDECK form, if you did not cancel job 4 during the installation of VS BASIC under DOS/VS (step 6B). The source for ICDKBFTB was placed in the source statement library at that time also. Using the maintenance program, MAINT, with a CATALR statement, catalog the assembled SLF modules in the same library as the old VS BASIC modules. The old modules will be replaced if you do not rename them. To relink edit the VS BASIC modules, use the following procedure:

```
// JOB
// OPTION CATAL
    ACTION MAP
PHASE ICDJCOMP,ICDDSBSC+16K,NOAUTO
    INCLUDE ICDJNUCL
    INCLUDE ICDJNUC1
    INCLUDE ICDJNUC2
    INCLUDE ICDJNUC3
    INCLUDE ICDJNUC4
    INCLUDE ICDJNUC5
    INCLUDE ICDJDEFR
    INCLUDE ICDJRUNA
    INCLUDE ICDJCMPA
    INCLUDE ICDJERR
    INCLUDE ICDJMATV
    INCLUDE ICDJFUTS
    INCLUDE ICDJINFO
    INCLUDE ICDJIOVB
    INCLUDE ICDJVERB
    INCLUDE ICDJUSFN
    INCLUDE ICDJUSTB
    INCLUDE ICDJVREC
    INCLUDE ICDJDUMY
PHASE ICDKRTNS,ICDDSBSC+16K
    INCLUDE ICDKBFTB
    INCLUDE ICDKERR
    INCLUDE ICDKGSUB
    INCLUDE ICDKDSUB
    INCLUDE ICDKSSUB
    INCLUDE ICDKIOVB
    INCLUDE ICDKVIOR
// LBLTYPE NDS(4)
// EXEC LNKEDT
/&
```

Space Considerations for SLF

As you increase the size of the VS BASIC library by adding new SLF routines, the library may reach a point at which it will exceed the size of the compiler. This is a problem since the library is designed to overlay the compiler during execution. If the library exceeds the size of the compiler it will extend past the compiler when it is loaded and will overlay the beginning of the user area, which immediately follows the compiler in storage.

Using the load maps that are produced when you install your SLF routines, you can determine how much space is available for additional SLF routines. The compiler load map will list the location of the module ICDJDUMY, the last module in the compiler. This routine is an empty DSECT occupying 9K of storage; therefore, the location shown for ICDJDUMY plus 9K will produce the size of the compiler. The library load map will list the location of the last SLF routine that was added as the last routine in the library. The location of the last SLF routine plus its size will produce the size of the library. The difference between the size of the compiler and the size of the library is the amount of space available to you for additional SLF routines. The size of the compiler must always be greater than or equal to the size of the library.

Should the size of the library exceed the size of the compiler and you wish to install additional SLF routines, can do the following to adjust the VS BASIC processor for the additional space requirements.

1 Replace ICDJDUMY with a new, larger ICDJDUMY that contains an DS instruction defining the size of the new SLF routine.

2 For CMS only, locate in the CMS distribution tape installation procedure VSBINSTL EXEC (see the section "Distribution Tape Installation EXEC Procedure for VM/370(CMS)") the following LOAD command:

```
LOAD ICDBLDTB ICDPRSCN ICDWNSCN ICDLSSCN ICDSTSCN (ORIGIN
2F000)
```

Change the ORIGIN parameter by adding to it the size of the new SLF routine.

3 Install the new SLF routine.

SYSTEM PROGRAMMING CONSIDERATIONS FOR OS/VS1, OS/VS2, OS/VS2(TSO) AND DOS/VS USERS

GUIDELINES FOR CREATING VSAM FILES USING ACCESS METHOD SERVICES

This section of the book is designed as a brief introduction to creating VSAM files for your VS BASIC programmers using the facilities of Access Method Services. Primary emphasis will be on certain JCL statements and on the DEFINE command of Access Method Services. This section is not intended to teach you how to use or maintain VSAM; therefore, most of the parameters and options of the required JCL and DEFINE commands will not be discussed nor will any of the other commands available through Access Method Services. You should refer to the following publciations for a complete description of managing and using VSAM and Access Method Services:

Under OS/VS

<u>OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide</u>
Order No. GC26-3818

<u>OS/VS Access Method Services</u>
Order No. GC35-0009

Under DOS/VS:

<u>DOS/VS Data Management Guide</u>
Order No. GC33-5372

<u>DOS/VS Utilities. Access Method Services</u>
Order No. GC33-5382

CREATING VSAM FILES FOR VS BASIC USERS

There are three steps to creating a VSAM file:

1 Defining a VSAM Master Catalog

Create a VSAM master catalog. The VSAM master catalog is a central information point for all VSAM files and the direct access storage volumes that contain them. The catalog provides VSAM with the information to allocate space for files, verify that the user is authorized to use them, compile statistics on their use, and relate relative addresses to physical locations. All VSAM files and indexes must be cataloged in the VSAM master catalog (under OS/VS, a user catalog may be used in place of the master catalog; however, the user catalogs must have a pointer to them in the master catalog). The master catalog is created only once by defining it through job control and the DEFINE command of Access Method Services.

2 Defining a VSAM Data Space

Allocate an area of direct access storage to VSAM for a non-unique file. (A unique file does not require a predefined data space. When it is defined, it acquires its own unique data space. This data area, called a VSAM data space, is owned by VSAM and is available for VSAM files that will be created later. A data space consists of one or more extents on a volume and is described in the VTOC as well as in the VSAM master catalog. Data spaces are allocated through job control and the DEFINE command of Access Method Services, and are created as needed to contain VSAM files.

3 Defining a VSAM File

Define a VSAM file in a VSAM data space and enter information about the file characteristics in the VSAM master catalog is the file that the VS BASIC programmer will process through his programs. These VSAM files must be defined for your VS BASIC users whenever they need to create a new VSAM file in their programs. The BASIC users will have to notify you in advance so that you can define the needed files through job control and the DEFINE command of Access Method Services.

DEFINING A VSAM MASTER CATALOG

Usually, the first job that you will run after you have installed VSAM in your operating system for the first time is a job to create your master catalog. (Some systems have a predefined master catalog, and it is not necessary to define one.) Without a master catalog you cannot define data spaces or files. The volume on which the master catalog is defined must always be available to the operating system. (Under OS/VS, user catalogs may also be defined and may be used in place of or in addition to the master catalog. The user catalog performs the same functions as the master catalog; however, the master catalog is still required to contain pointers to the user catalog. If a user catalog is used, it must be identified by the VS BASIC programmer in his job control with a JOBCAT or STEPCAT DD statement.)

Job Control Statements Required for a VSAM Master Catalog

UNDER OS/VS: An OS/VS DD statement of the following form is required in the job that executes Access Method Services to create a master catalog:

```
     //ddname  DD   DISP=OLD,UNIT=unit,VOL=SER=serial-number
```

where:

ddname - is any appropriate ddname.

unit - is the direct access device on which you want the master catalog to reside.

serial-number - is the volume serial number of the volume on which the master catalog is to reside.

UNDER DOS/VS: A DLBL and an EXTENT statement of the following form are required in the job that executes Access Method Services to create a master catalog:

```
     //   DLBL    IJSYSCT,'file-id',,VSAM
     //   EXTENT  SYSCAT,disk-label,1,,first-track,total-tracks
```

where:

file-id - is the name that is to be assigned to the master catalog.

disk-label - is the label of the disk on which the master catalog is to reside.

first-track - is the number of the first track of the extent.

total-tracks - is the number of tracks required by the master catalog.

DEFINE Command Required for a VSAM Master Catalog

The DEFINE command of Access Method Services for a master catalog is as follows:

```
     DEFINE MASTERCATALOG (NAME(catalog-name) FILE(ddname)-
            VOLUMES(serial-number)...)
```

where:

catalog-name - is the name that you choose for the catalog.

ddname - is the ddname on the DD statement or the filename on the DLBL statement that defines the data set on which the master catalog will reside.

serial-number - is the serial number of the volume on which the master catalog is to be placed.

Note: If you wish to define a user catalog under OS/VS, you can use the same DD statement and DEFINE command; however, you must substitute the keyword USERCATALOG for MASTERCATALOG in the DEFINE command.

See the section "Examples of Using Access Method Services to Define VSAM Files" for an illustration of how these statements are actually used to define a master catalog.

DEFINING A VSAM DATA SPACE

All VSAM files that your users will process are stored in VSAM data spaces. Non-unique files may share the same data space; however, a unique file will allocate and occupy its own data space. To define a VSAM file that can share a data space with other files, the data space in which you are going to place the file must have been defined previously. A data space can occupy all or a part of a direct-access volume. It cannot occupy more than one volume. Several data spaces can, however, share the same volume. And, in addition, the same data space can be defined on several different volumes.

Job Control Statements Required for a VSAM Data Space

UNDER OS/VS: An OS/VS DD statement that is identical to the DD statement for the master catalog is required in the job that executes Access Method Services to create a data space. The only exception is that the *ddname* will be different; all other information is the same. If you are using a user catalog you must also include a JOBCAT or STEPCAT DD statement for the user catalog.

UNDER DOS/VS: A DLBL and an EXTENT statement that are identical to the statements for the master catalog are required in the job that executes Access Method Services to create the data space. The only difference on the DLBL statement is that IJSYSCB is replaced by any DOS file-name that you choose. The only exception on the EXTENT statement is that the track information should reflect the amount of space required by the data space.

DEFINE Command Required for a VSAM Data Space

The DEFINE command of Access Method Services for a data space is as follows:

```
DEFINE SPACE (FILE(ddname) VOLUMES(serial-name) space...)
```

where:

ddname - is the ddname of the OS DD statement or the file name assigned on the DOS DLBL statement that defines the data set on which the data space will reside.

serial-number - is the volume serial number of the volume on which the data space is to reside.

space - is the amount of space required by the data space. This is specified by the appropriate CYLINDER,TRACK, or RECORD parameter of the DEFINE command.

Note: If you are a user catalog under OS/VS, you must include the CATALOG option specifying the name of the user catalog.

See the section "Examples of Using Access Method Services to Define VSAM Files" for an illustration of how these statements are actually used to define a data space.

DEFINING A VSAM FILE

VSAM can be defined as either key-sequenced or entry-sequenced. The optional parameters of the DEFINE command determine the type of file that is defined. Each file is defined as a cluster. For key-sequenced files, the cluster contains two parts, a data portion that is the file and an index portion. Entry-sequenced are defines with only a data portion.

Job Control Statements Required for a VSAM File

There are no special requirements for job control statements when using Access Method Services to define a VSAM file. The only exception occurs under OS/VS when you are using a user catalog. In this case, you must include a JOBCAT or STEPCAT DD statement to identify the user catalog.

DEFINE Command Required for a VSAM File

The DEFINE command for key-sequenced files is as follows:

```
  DEFINE CLUSTER (NAME(cluster-name) FILE(ddname) VOLUMES(serial-number)
          KEYS(key-length key-position) space ...)
```

where:

cluster-name - is the name of the cluster being defined.

ddname - is the ddname of the DD statement or the name assigned on the DLBL statement that defines the data set on which the corresponding data space resides.

serial-number - is the volume serial number of the volume on which the file is to reside.

key-length - is the length of the key in each record in the file.

key-position - is the starting position of the key in each record. The key position specified here is calculated from position 0 not position 1. Therefore, position 0 is really the first position in the record. This differs from the way VS BASIC specifies the key position. In VS BASIC the key position is calculated from position 1.

space - is the amount of space required by the file. This is specified by the appropriate CYLINDER, TRACK, or RECORD parameter of the DEFINE command.

Note: If you are using a user catalog under OS/VS, you must include the CATALOG option specifying the name of the user catalog.

The define command for an entry-sequenced file is the same as for a key-sequenced file except that you do not specify the KEYS parameter and you must specify the NONINDEXED parameter.

See the section "Examples of Using Access Method Services to Define VSAM Files" for an illustration of how these statements are actually used to define a data space.

EXAMPLES OF USING ACCESS METHOD SERVICES TO DEFINE VSAM FILES

The following examples show how to define a catalog, a data space, and a key sequenced file under OS/VS and DOS/VS. A set of examples has also been included for the TSO terminal user so that the Access Method Services jobs can be entered from the terminal. It is assumed, for OS/VS only, that a master catalog has been previously created, and that this example will create a user catalog. For DOS/VS, a master catalog will be created since DOS does not permit user catalogs.

The catalog is to have 300 records, a name of NEWCAT, and is to reside on volume 326444 on a 3330 disk device. The data space is to occupy 100 cylinders; the key-sequenced file is to be called NEWFILE and will contain 1000, 100-byte records with a 10 byte key starting in the first position of the record.

Example of Using Access Method Services under OS/VS

Defining a User Catalog under OS/VS

```
//CATALOG   JOB     ...
//          EXEC    PGM=IDCAMS
//SYSPRINT  DD      SYSOUT=A
//VOL       DD      DISP=OLD,UNIT=3330,VOL=SER=326444
//SYSIN     DD      *
            DEFINE  USERCATALOG (NAME(NEWCAT) FILE(VOL)-
                    VOLUMES(326444) RECORDS(300))
/*
```

Defining a Data Space under OS/VS

```
//DATASP    JOB     ...
//JOBCAT    DD      DSNAME=NEWCAT,DISP=OLD
//          EXEC    PGM=IDCAMS
//SYSPRINT  DD      SYSOUT=A
//VOL       DD      DISP=OLD,UNIT=3330,VOL=SER=326444
//SYSIN     DD      *
            DEFINE  SPACE (FILE(VOL) VOLUMES(326444)-
                    CYLINDERS(100)) CATALOG(NEWCAT)
/*
```

Note: If the master catalog was to be used instead of the user catalog, the JOBCAT DD statement and the CATALOG option of the DEFINE command would be omitted.

Defining a Key-Sequenced VSAM File under OS/VS

```
//CLUST     JOB     ...
//JOBCAT    DD      DSNAME=NEWCAT,DISP=OLD
//          EXEC    PGM=IDCAMS
//SYSPRINT  DD      SYSOUT=A
//SYSIN     DD      *
            DEFINE  CLUSTER (NAME(NEWFILE) FILE(VOL)-
                    VOLUMES(326444) RECORDS(1000)-
                    RECORDSIZE(100 100)-
                    KEYS(10,0)) CATALOG(NEWCAT)
/*
```

Note: If the master catalog was to be used instead of the user catalog, the JOBCAT DD statement and the CATALOG option of the DEFINE command would be omitted unless the master catalog is password protected.

Example of Using Access Method Services under OS/VS2(TSO) (Without Command Procedures)

The following commands free and allocate files that are necessary for running Access Method Services (system responses are not shown):

```
free f(sysin,sysprint)
alloc f(sysprint) da(*)
alloc f(sysin) da(*)
alloc f(vol) old volume(326444)
```

The following command calls Access Method Services:

```
call 'sys1.linklib(idcams)'
```

The following command defines the usercatalog:

```
define usercatalog (name(newcat) file (vol) volumes(326444)-
records(300))
```

The following command defines the data space:

```
define space (file(vol) volumes(326444) cylinders(100))-
catalog(newcat)
```

The following command defines the key-sequenced file:

```
define cluster (name(newfile) file(vol) volumes(326444)-
records(1000) recordsize(100 100) keys(10 0)) catalog(newcat)
```

Note: If the master catalog was to be used instead of the user catalog, the CATALOG option of the DEFINE command would be omitted. In addition, the following JOBCAT or STEPCAT DD statement must be placed in the user's LOGON procedure and he must Logon again:

```
// JOBCAT     DD   DSN=NEWCAT,DISP=SHR
   STEPCAT
```

Example of Using Access Method Services under OS/VS2(TSO) (With Command Procedures)

The following are the input files for:

```
edit 'defcat' data nonum
 define usercatalog (name(newcat) file(vol) volumes(326444)--
 records(300))
save

edit 'defspace' data nonum
 define space (file(vol) volumes(326444) cylinders(100))--
 catalog(newcat)
save

edit 'defclust' data nonum
 define cluster (name(newfile) file(vol) volumes(326444)--
 records(1000) recordsize(100 100) keys(10,0)) catalog(newcat)
```

Note: If the master catalog was to be used instead of the user catalog, the CATALOG option of the DEFINE command would be omitted. In addition, when continuing lines under EDIT, you must enter two consecutive hyphens to represent the continuation character.

The following file is the command procedure that will execute Access Method Services. It locates the correct DEFINE command to pass to

Access Method Services through the parameter ('& command.'). This parameter corresponds to the name of the file that contains the DEFINE command.

```
edit 'vsamproc' clist new
00010 proc 1 command
00020 free f(sysin,sysprint)
00030 alloc f(sysprint) da(*)
00040 alloc f(sysin) da('&command.')
00050 alloc f(vol) old volume(326444)
00060 call 'sys1.linklib(idcams)'
00070 free f(sysin,sysprint)
00080 end
save
end
```

Note: Before this command procedure can be executed, the following JOBCAT DD statement must be placed in your logon procedure and you must logon again:

```
//STEPCAT DD DSN=NEWCAT,DISP=SHR
```

The following are the commands that you must enter to execute your command procedure to define the catalog, data space, and file:

```
exec 'vsamproc' 'defcat'
exec 'vsamproc' 'defspace'
exec 'vsamproc' 'defclust'
```

Example of Using Access Method Services under DOS/VS

Defining a Master Catalog under DOS/VS

```
//  JOB
//  DLBL  IJSYSCT,'NEWCAT',,VSAM
//  EXTENT  SYSCAT,326444,1,,100,25
//  EXEC  IDCAMS,SIZE=26K
    DEFINE MASTERCATALOG (NAME(NEWCAT) VOLUMES(326444)-
           FILE(IJSYSCT) RECORDS(300))
/*
/&
```

Note: Before this sample job to define the master catalog can be run the following command must be issued during IPL (the choice of devices is arbitrary here):

```
CAT UNIT=X'130'
```

Defining a Data Space under DOS/VS

```
//  JOB
//  ASSGN SYS001,X'130'
//  DLBL  VOL,,,VSAM
//  EXTENT  SYS001,326444,1,,50,2000
//  EXEC  IDCAMS,SIZE=26K
    DEFINE  SPACE FILE(VOL) VOLUMES(326444) CYLINDERS(100))
/*
/&
```

Defining a Key-Sequenced File under DOS/VS

```
    //  JOB
    //  EXEC  IDCAMS,SIZE=26K
        DEFINE  CLUSTER (NAME(NEWFILE) FILE(VOL) VOLUMES(326444)-
                RECORDS(1000) RECORDSIZE(100 100) KEYS(10 0))
    /*
    /&
```

SYSTEM PROGRAMMING CONSIDERATIONS FOR OS/VS2 (TSO) USERS

PREPARING A LOGON PROCEDURE UNDER TSO

Before your VS BASIC user can log onto TSO and begin using VS BASIC, you must, first, create a TSO LOGON procedure for them and place it on SYS1.PROCLIB. The following job will do this:

```
A    //VSBPROC    JOB     accounting-information,MSGLEVEL=(1,1)
     //           EXEC    PGM=IEBUPDTE,PARM=NEW
     //SYSPRINT   DD      SYSOUT=A
     //SYSUT2     DD      DSN=SYS1.PROCLIB,DISP=OLD
     //SYSIN      DD      DATA
B    ./           ADD     NAME=any-name,LIST=ALL
     ./           NUMBER  NEW1=10,INCR=10
C  1 //stepname   EXEC    PGM=IKJEFT01
     //SYSPRINT   DD      SYSOUT=A
     //SYSUADS    DD      DSN=SYS1.UADS,DISP=SHR
D    //SYSHELP    DD      DSN=library-name,DISP=SHR
     //           DD      DSN=SYS1.HELP,DISP=SHR
E    //name1      DD      DYNAM
                  .
                  .
                  .
E    //namen      DD      DYNAM
     /*
```

An explanation of the lettered statements follows:

A Supply any accounting-information that your computing center requires.

B This statement identifies the procedure being added to SYS1.PROCLIB. Supply any-name that is appropriate.

C This statement executes the terminal monitor program that is distributed with TSO (IKJEFT01). If you have written your own monitor program substitute its name for IKJEFT01. You must supply a stepname, which the terminal user is to include in his LOGON command.

D This statement identifies the library that contains the VS BASIC modifications to the HELP facility. Supply the same library-name that you specified in the //HELP DD statement in step 1 of the TSO installation procedure.

E This statement and the succeeding DD DYNAM statements reserve entries for user files that may be used from the terminal. The name selected should not conflict with any other names that the user is likely to have. Suggested names that will work for VS BASIC users are DD01 through DD99. The number of DD DYNAM statements that you include should be based on the estimated needs of your users.

<u>Note</u>: This procedure assumes that the VS BASIC Processor resides on SYS1.LINKLIB. If it does not, you must insert a STEPLIB DD statement of the following form:

```
//STEPLIB  DD  DSN=library-name,DISP=SHR
```

At the point marked ➊ in the LOGON procedure.

If you are using VSAM files, you must insert an additional DD statement of the form:

```
//ddname  DD  UNIT=unit,DISP=SHR,VOL=SER=serial-number
```

This statement identifies the data set on which your VSAM catalog resides. See the example of defing a VSAM catalog under TSO in the section "Creating VSAM Files" for an illustration of the relationship between this DD statement and your DEFINE command.

One additional DD statement is required if you are using a VSAM user catalog in place of a master catalog for your files. Its form is:

```
//STEPCAT  DD  DSN=catalog-name,DISP=SHR
```

This STEPCAT DD statement identifies the user catalog that you are using. It must be placed in the LOGON procedure at the place marked ➊ ; however, it must precede a STEPLIB DD statement, if used. Here, too see the example of defining a VSAM for the relationship between this statement and your DEFINE command.

Table 7. Terminal Keyboard Special Characters for TSO

VS BASIC Special Characters	Corresponding TSO Terminal Keyboard Representations[1]						
	1050	2260/5	2741 BCD	2741 PTTC	2741 Corr	3270	Teletype[2] 33 & 35
.	.	.	.	.	.	.	.
<	<	<		<		<	<
(	(	(	(	(	(	(	(
+	+	+	+	+	+	+	+
|	|	|	±	|	±	|	
&	&	&	&	&	&	&	&
!	!		!	!	!	!	!
$	$	$	$	$	$	$	$
*	*	*	*	*	*	*	*
)	)	)	)	)	)	)	)
;	;	;	;	;	;	;	;
-	-	-	-	-	-	-	-
/	/	/	/	/	/	/	/
,	,	,	,	,	,	,	,
>	>	>		>	]	>	>
?	?	?	?	?	?	?	?
:	:	:	:	:	:	:	:
#	#	#	#	#	#	#	#
@	@	@	@	@	@	@	@
'	'	'	'	'	'	'	'
=	=	=	=	=	=	=	=
"	"		"	"	"	"	"
↑	**	**	**	**	**	**	↑ or **
≤	>=	>=		>=	]=	>=	>=
≥	<=	<=		<=	[=	<=	<=
≠	<>	<>		<>	[]	<>	<>

[1]Wherever a blank appears in this table it means that the character is not available on the terminal in question.

[2]Teletype is a trademark of the Teletype Corporation, Skokie, Illinois.

SYSTEM PROGRAMMING CONSIDERATIONS FOR OS/VS1 AND OS/VS2 USERS

PLACING COMPONENTS OF THE VS BASIC PROCESSOR INTO THE LINK PACK AREA OF OS/VS

Since the VS BASIC compiler, library, and executors are designed to be re-entrant, it is recommended that these components be installed in the link pack area of OS/VS1 or OS/VS2. A considerable savings in storage that the speed with which these components can be accessed will result.

Naturally, if you have the space available in your link pack area, all three components should be placed there. If you do not have room for all of these components, the following are general guidelines which will help you you decide which components to place in the link pack area:

If Your Programs Require	Then Place These Components in the Link Pack Area
Lengthy Compilations	Executor and Compiler
Large Amounts of I/O	Executor and Library
Lengthy or Numerous Executions	Executor and Library

Note: Because parts of the Debug facility are not re-entrant, they cannot be placed in the link pack area. In addition, if you create any SLF routine that is not re-entrant you will not be able to place the re-link edited compiler or library into the link pack area. The RENUM modules are also re-entrant; however, unless you plan to use the RENUM facility frequently, do not place them into the link pack area.

Installing VS BASIC Components in the Link Pack Area of OS/VS

UNDER OS/VS2: There are two methods that can be used to place re-entrant components of the VS BASIC Processor into the OS/VS link pack area. The first method requires that you copy the link edited components into SYS1.LPALIB. You must then perform a cold start of your system. A faster variation of the method can be achieved by adding to the list of names in the member IEALOD00 the names of the components you wish added to the link pack area before you perform the cold start.

The second method used a modified link pack area but can be accomplished using either a warm or cold start. During your IPL (either cold or warm) include an MLPA command that specifies the member IEASYSnn containing a list of names of the components to be placed in the modified link pack area. You must add your component names to the list.

For more information on either method, refer to the publications:

OS/VS2 Planning and Use Guide, Order No. GC28-0600
OS/VS2 System Generation Reference, Order No. GC26-3792

UNDER OS/VS1: Copy the new components into SYS1.LINKLIB. Include the new components in the list IEAIGGxx (where xx are any alphameric characters other than those specified in the system member names). IEAIGGxx is a member of SYS1.PARMLIB. In addition, IEAIGGxx must be included in the RAM parameter list during IPL.

For more information on this method, refer to the publications:

OS/VS1 Planning and Use Guide, Order No. GC24-5090
OS/VS1 System Generation Reference, Order No. GC26-3790

CMS Preparations for New VS BASIC Users

Because it is assumed that your VS BASIC users may not be familiar with the operation or requirements of CMS, it is recommended that you make some additional preparations for them that will facilitate their use of VS BASIC under CMS. The major considerations are:

- Conflicts between the CMS logical line editing characters and characters that may be required by the VS BASIC FORM, Image Arithmetic Assignment, and certain I/O statements. The characters in conflict are the pound sign (#), the at sign (@), and the double quote (").
- The need to format the user's A disk.
- The need to provide a GLOBAL TXTLIB command for TSOLIB when VS BASIC debug is to be used.
- The lack of support by certain CMS terminals of characters that are a part of the VS BASIC character set (for example, the less than sign (<) on a 2741 terminal with a corresponding keyboard).

Before you permit any of your VS BASIC users to begin operating, it is recommended that you do the following:

1 Change the line editing characters with the USER statement in the user's directory.

2 Log onto CMS as the user and format his A disk with a FORMAT command.

3 Copy a PROFILE EXEC procedure onto his A disk that contains a GLOBAL TXTLIB command for TSOLIB, if required, an SET INPUT and OUTPUT commands for any VS BASIC characters that are not supported by the terminal that users will be using. This can be limited to the more important characters such as the less than sign. Refer to Table 8,for a list of the characters supported by each type of terminal and the corresponding hexadecimal value required to create the missing character.

4 Notify your users of the new characters that will be available.

Table 8. Terminal Keyboard Special Characters for CMS

VS BASIC Special Characters	Hexadecimal Value[2]	Correspond Terminal Keyboard Representations[1]					
		1050	2741			3270	Teletype[3] 33&35
			PTTC	Corr	APL		
.	4B	.	.	.	.	.	.
<	4C	<	<	<	<	<	<
(	4D	(	(	(	(	(	(
+	4E	+	+	+	+	+	+
|	4F	|	|	!	|	|	↑
&	50	&	&	&		&	&
!	5A	!	!			!	!
$	5B	$	$	$		$	$
*	5C	*	*	*	*	*	*
)	5D	)	)	)	)	)	)
;	5E	;	;	;	;	;	;
-	60	-	-	-	-	-	-
/	61	/	/	/		/	/
,	6B	,	,	,	,	,	,
>	6E	>	>	±	>	>	>
?	6F	?	?	?	?	?	?
:	7A	:	:	:	:	:	:
#	7B	#	#	#		#	#
@	7C	@	@	@		@	@
'	7D	'	'	'	'	'	'
=	7E	=	=	=	=	=	=
↑	8A	**	**	**	**	**	**
≤	8C	>=	>=	>=	>=	>=	>=
≥	AE	<=	<=		<=	<=	<=
≠	BE	<>	<>		<>	<>	<>

[1]Wherever a blank appears in this table it means that the character is not available on the terminal in question.

[2]CMS offers, through the SET command, the ability to create or change any of the keyboard symbols. These hexadecimal values can be used in the SET INPUT or SET OUTPUT commands. See the publication IBM Virtual Machine Facility/370: Terminal User's Guide, Order No. GC20-1810, for more detailed information on using the set command.

[3]Teletype is a trademark of the Teletype Corporation, Skokie, Illinois.

REPLACING ROUTINES OF THE VS BASIC PROCESSOR UNDER CMS

The CMS exec procedure VSBINSTL that you used to install VS BASIC has the facility to replace compiler and library modules distributed on tape at PTFs.

The only reserved fileid is 'xxxxxxxx TEXT' where xxxxxxxx is the name of the TXTLIB that the new compiler or library routines will be added to during the PTF installation. (See the sample console input in step 3D below). Before you can install a PTF the file VSB TEXT, which was created during the installation of the VS BASIC Processor, must be available.

To execute the exec procedure in PTF mode follow the steps below:

1 Mount the PTF tape at virtual address 181.

2 If the VSBINSTL EXEC is no longer available refer to the CMS installation section for the commands necessary to reload it from one of the installation tapes.

3 Execute the exec as follows:

```
     exec vsbinstl ptf
```

From this point on, the PTF procedure will prompt you for responses. The procedure types out the following message:

```
     PTFS ON TAPE FOR VS BASIC PROGRAM PRODUCE 5748-XX1
```

Note: If you regenerate the VS BASIC modules, the existing modules will be replaced. Rename any existing modules that you wish to retain.

A Indicate the position on the PTF tape of the PTF to be applied. Most PTF tapes contain only one PTF; however, refer to PTF cover letter. If there is more than one PTF on the tape, you must repeat this procedure to each PTF. The following message is typed at your terminal:

```
     NEXT ENTER THE PTF POSITION ON TAPE

        I.E.    FIRST PTF ENTER ....   1
                SECOND PTF ENTER ...   2
                ETC.
```

Respond with

$$\begin{Bmatrix} 1 \\ 2 \\ \cdot \\ \cdot \\ \cdot \end{Bmatrix}$$

B Specify the characteristics of the PTF tape. The procedure types the following at your terminal:

```
NOW, IF THE PTF TAPE IS OTHER THAN 9 TRACK DENSITY 800, ENTER
IT'S
MODE AND DENSITY AS FOLLOWS:

     FOR 9 TRACK 6250 ENTER.......................  6250
     FOR 9 TRACK 1600 ENTER.......................  1600

ELSE PRESS RETURN
```

You must enter one of the following:

$$\begin{Bmatrix} 6250 \\ 1600 \\ CR \end{Bmatrix}$$

C Ensure that the system disk (S) has been accessed as your A disk. The following message will be typed at your terminal:

```
THE SYSTEM DISK TO RECEIVE THE COMPILER MUST BE ACCESSED IN
READ/WRITE STATUS AS THE 'A' DISK.  IF NOT, ENTER 'END',
ACCESS THE SYSTEM DISK IN THE PROPER STATUS AND EXECUTE
THIS EXEC AGAIN.

IF IT IS ACCESSED AS THE READ/WRITE 'A' DISK, PRESS RETURN.
```

You must respond with one of the following:

$$\begin{Bmatrix} \text{end} \\ \text{CR} \end{Bmatrix}$$

If the system disk is not accessed as your read/write A disk, enter END. The following response will be typed at your terminal:

```
     'EXIT FOR SYSTEM DISK ACCESS'
```

Do 5 , and repeat step 7 from the beginning. If the system disk is correctly accessed, enter CR and continue.

D Indicate the txtlib to receive the new module. The following message is typed:

```
SPECIFY THE NAME OF THE TXTLIB TO RECEIVE THE
MODULES ...  BEWARE, THE TXTLIB CANNOT ALREADY CONTAIN MEMBERS
WITH
THE SAME NAMES AS THOSE BEING REPLACED.
```

You must enter a new TXTLIB name, for example:

```
     testlib
```

A TXTLIB file with the filename TESTLIB will be created for the modules if this file does not already exist. If the file TESTLIB TXTLIB already exists on an accessed disk, the PTF modules will be added to the existing file. If the file does not exist, the following message will be typed at your terminal:

```
     FILE 'TESTLIB TXTLIB' NOT FOUND
```

This message can be ignored. In addition, a TEXT file will also be created with the filename TESTLIB. If a file with the name TESTLIB TEXT already exists, the following error message will be typed at your terminal:

```
     'TESTLIB TEXT' ALREADY EXISTS...  RENAME OR ERASE IT AND
     REISSUE
     THIS EXEC
     R (00002);
```

You must rename or erase the existing file and repeat step 3 from the beginning. If this file does not exist, the following message is typed:

```
     FILE 'TESTLIB TEXT' NOT FOUND
```

This message can be ignored. The installation procedure will then create the TXTLIB and TEXT files.

The procedure continues by moving the PTF text to disk as 'TESTLIB TEXT', and generates or adds to 'TESTLIB TXTLIB' the new text modules.

The system will then type:

```
IF THERE IS AN ADDITIONAL PTF ON THIS TAPE WHICH YOU WISH TO
APPLY AT
THIS TIME, ENTER THE TAPE POSITION FOR THAT PTF AS EXPLAINED
EARLIER
IF NOT PRESS RETURN
```

You must enter either the tape position or CR. Additional PTF's will then be applied, or the procedure finishes at this point.

F Indicate whether you wish to regenerate the VS BASIC modules now or later. The procedure types:

```
THE REGENERATION OF THE VS BASIC MODULES MAY BE DEFERRED UNTIL
A LATER TIME.  IF YOU WISH TO DO SO ENTER 'NOREGEN'.
OTHERWISE PRESS RETURN AND THE REGENERATION WILL BE DONE NOW.
```

You must enter either NOREGEN or CR. If you wish to delay regenerating the VS BASIC modules until a later time enter NOREGEN. The procedure finishes at this point, and the following message is typed:

```
PTF INSTALLATION COMPLETE
R;
```

When you decide to regenerate the VS BASIC modules, you will have to execute the VSBINSTL procedure as described in the section "Regenerating the VS BASIC Load Modules under CMS."

If you wish to regenerate the VS BASIC module immediately enter CR. The system will type the following message:

```
TO REGEN THE VS BASIC MODULES THE PROPER TXTLIBS MUST BE
GLOBALED.
IF 'VSB TXTLIB' THAT WAS CREATED AT INSTALL TIME IS NOT
AVAILABLE. BY THAT NAME ENTER IT'S CORRECT NAME; ELSE, PRESS
RETURN.
```

Enter the TXTLIB filename or CR. The following message will be printed at your console:

```
REGEN OF COMPILER MODULES WITH TXTLIBS aaaaaaaa bbbbbbbb
```

where:

aaaaaaaa - is the txtlib containing the modules replaced in the PTF,

bbbbbbbb - is the install time txtlib containing all the VS BASIC modules (VSB).

The compiler modules are regenerated. During this procedure, the following message is typed and can be ignored:

```
THE FOLLOWING NAMES ARE UNDEFINED
ICDJSRCH
```

When the installation is complete, the following message indicates this:

```
INSTALLATION/REGEN COMPLETE
R;
```

Load maps of the VS BASIC modules are printed during the regeneration.

REGENERATING THE VS BASIC LOAD MODULES UNDER CMS

The exec VSBIBSTL has the facility to regenerate the VS BASIC modules. This facility may be required if the modules are moved on disk or if modules are modified. The file VSB TEXT must be available.

Issue the following command to regenerate the VS ABSIC modules:

```
     exec vsbinstl regen
```

Note: This command will replace existing VS BASIC modules; therefore, if you wish to retain the old modules you must rename them.

From this point on the regeneration procedure will prompt you for responses. The procedure types out the following message:

```
REGEN OF THE VS BASIC MODULES
```

A Ensure that the system disk (S) has been accessed as your A disk. The following message will be typed at your terminal:

```
THE SYSTEM DISK SHOULD BE ACCESSED IN READ/WRITE STATUS AS THE 'A'
DISK.  IF NOT, ENTER 'END' REACCESS THE SYSTEM DISK IN THE PROPER
STATUS AND REISSUE THIS EXEC AGAIN.

IF IT IS ACCESSED AS THE READ/WRITE 'A' DISK, PRESS RETURN.
```

You must respond with one of the following:

```
end
CR
```

If the system disk is not accessed as your read/write A disk, enter END. The following response will be typed at your terminal:

```
     'EXIT FOR SYSTEM DISK ACCESS'.
```

Re-enter the EXEC VSBINSTL REGEN command. If the system disk is correctly accessed, the procedure continues.

B Enter the names to be used for the regeneration. The following message will be typed at your terminal:

```
ENTER THE TXTLIB NAMES TO BE SEARCHED DURING THE REGEN OF THE
COMPILER (TO A MAXIMUM OF 7)

IF ONLY THE TXTLIB 'VSB' AS CREATED AT INSTALLATION TIME IS TO
BE SEARCHED, PRESS THE RETURN KEY.
```

At this point enter the name of your txtlib followed by VSB (for example, testlib vsb) or enter CR.

C After the VS BASIC modules have been regenerated, the following message is typed and the regeneration procedure is completed:

```
INSTALLATION/REGEN COMPLETE
R;
```

DISTRIBUTION TAPE INSTALLATION JCL PROCEDURE FOR OS/VS

```
//VSBPP  JOB 1,PP.NUMBER.5748,MSGLEVEL=(1,1)
//*
//*   5748-XX1 COPYRIGHT IBM CORP. 1974
//*   REFER TO INSTRUCTIONS ON COPYRIGHT NOTICE 120-2083
//*   OS VS1/VS2 ONLY
//*
//*       DEFINE TARGET LIBRARIES
//STP1  EXEC  VSBDEF
//*
//STP2 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',REGION=128K,COND=(8,LE)
//SYSPRINT DD SYSOUT=A,SPACE=(121,(1000,50),RLSE)
//SYSUT1   DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLMOD  DD  DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSLIN DD LABEL=(04,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//     DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*    LINKEDIT VS EXECUTOR
//*
//STP3 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A,SPACE=(121,(1000,50),RLSE)
//SYSUT1   DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLMOD DD DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSLIN DD LABEL=(09,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//     DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*    LINKEDIT COMPILER MODULES
//*
//STP4 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A,SPACE=(121,(1000,50),RLSE)
//SYSLMOD DD DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSUT1   DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIB   DD  DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSLIN DD *
  INCLUDE SYSLIB(ICDJNUCL)
  INCLUDE SYSLIB(ICDJNUC1)
  INCLUDE SYSLIB(ICDJNUC2)
  INCLUDE SYSLIB(ICDJNUC3)
  INCLUDE SYSLIB(ICDJNUC4)
  INCLUDE SYSLIB(ICDJNUC5)
  INCLUDE SYSLIB(ICDJDEFR)
  INCLUDE SYSLIB(ICDJRUNA)
  INCLUDE SYSLIB(ICDJCMPA)
  INCLUDE SYSLIB(ICDJERR)
  INCLUDE SYSLIB(ICDJMATV)
  INCLUDE SYSLIB(ICDJFUTS)
  INCLUDE SYSLIB(ICDJINFO)
  INCLUDE SYSLIB(ICDJIOVB)
  INCLUDE SYSLIB(ICDJVERB)
  INCLUDE SYSLIB(ICDJUSFN)
  INCLUDE SYSLIB(ICDJVREC)
  INCLUDE SYSLIB(ICDJDUMY)
   NAME  ICDJCOMP(R)
/*
//*    LINKEDIT COMPILER MEMBERS INTO LOAD MODULE
//*
//STP5 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',COND=(8,LE),REGION=128K
```

```
//SYSPRINT DD SYSOUT=A,SPACE=(121,(1000,50),RLSE)
//SYSUT1    DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLMOD DD DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSLIN DD LABEL=(10,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//       DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*      LINKEDIT RUNTIME ROUTINES
//*
//STP6 EXEC PGM=IEWL,PARM='LIST,MAP,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A,SPACE=(121,(1000,50),RLSE)
//SYSUT1    DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIB DD DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSLMOD DD DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSLIN DD *
 INCLUDE SYSLIB(ICDKBFTB,ICDKERR,ICDKGSUB,ICDKDSUB,ICDKSSUB,ICDKIOVB)
 INCLUDE SYSLIB(ICDKVIOR)
    NAME ICDKRTNS(R)
/*
//*      LINKEDIT RUNTIME MEMBERS INTO LOAD MODULE
//*
//STP7 EXEC PGM=IEBGENER,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD LABEL=(15,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//       DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//SYSUT2 DD SYSOUT=A,DCB=(BLKSIZE=3200,LRECL=80,RECFM=FB)
//SYSIN DD DUMMY
//*      PRINT VS BASIC MESSAGES
//*
//STP8 EXEC   PGM=IEBGENER,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD SYSOUT=B,DCB=(RECFM=F,BLKSIZE=80)
//SYSUT1 DD LABEL=(16,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//       DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
//*      PUNCH SAMPLE PROGRAM
//
//
```

```
//VSBPP  JOB 1,PP.NUMBER.5748,MSGLEVEL=(1,1)
//*
//*   5748-XX1 COPYRIGHT IBM CORP. 1974
//*   REFER TO INSTRUCTIONS ON COPYRIGHT NOTICE 120-2083
//*   TSO FOR VS2 AND VS1/VS2 BATCH
//*
//*       DEFINE TARGET LIBRARIES
//STP1  EXEC  VSBDEF
//*
//STP2 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DSN=*.STP1.VSB.VLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD LABEL=(04,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//        DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*    LINKEDIT VS EXECUTOR
//*
//STP3 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD LABEL=(05,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//        DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*    LINKEDIT TSO EXECUTOR
//*
//STP4 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD LABEL=(06,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//        DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*    LINKEDIT RENUM
//*
//STP5 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',REGION=128K,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD  DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD LABEL=(09,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//      DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*    LINKEDIT COMPILER ROUTINES
//*
//STP6 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',REGION=128K,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSLIB DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSLMOD DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD *
  INCLUDE SYSLIB(ICDJNUCL)
  INCLUDE SYSLIB(ICDJNUC1)
  INCLUDE SYSLIB(ICDJNUC2)
  INCLUDE SYSLIB(ICDJNUC3)
  INCLUDE SYSLIB(ICDJNUC4)
  INCLUDE SYSLIB(ICDJNUC5)
  INCLUDE SYSLIB(ICDJDEFR)
  INCLUDE SYSLIB(ICDJRUNA)
  INCLUDE SYSLIB(ICDJCMPA)
  INCLUDE SYSLIB(ICDJERR)
  INCLUDE SYSLIB(ICDJMATV)
  INCLUDE SYSLIB(ICDJFUTS)
  INCLUDE SYSLIB(ICDJINFO)
  INCLUDE SYSLIB(ICDJIOVB)
  INCLUDE SYSLIB(ICDJVERB)
  INCLUDE SYSLIB(ICDJUSFN)
  INCLUDE SYSLIB(ICDJVREC)
  INCLUDE SYSLIB(ICDJDUMY)
   NAME  ICDJCOMP(R)
/*
//*     LINKEDIT COMPILER LOAD MODULE
//*
//STP7 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD LABEL=(10,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//      DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*     LINKEDIT RUNTIME ROUTINES
//*
//STP8 EXEC PGM=IEWL,PARM='LIST,MAP,RENT',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSLIB DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSLMOD DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD *
 INCLUDE SYSLIB(ICDKBFTB,ICDKERR,ICDKGSUB,ICDKDSUB,ICDKSSUB,ICDKIOVB)
 INCLUDE SYSLIB(ICDKVIOR)
  NAME ICDKRTNS(R)
/*
//*     LINKEDIT RUNTIME ROUTINES INTO LOAD MODULE
//STP9 EXEC PGM=IEWL,PARM='LIST,MAP,NCAL',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD LABEL=(11,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//      DCB=(LRECL=80,BLKSIZE=3200,RECFM=FB)
//*     LINKEDIT DEBUG ROUTINES
//*
//STP10 EXEC PGM=IEWL,PARM='LIST,MAP',COND=(8,LE),REGION=128K
//SYSPRINT DD SYSOUT=A
//SYSLIB DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSLMOD DD DSN=*.STP1.VSB.TLNK,DISP=(OLD,PASS)
//SYSUT1 DD DSN=&&VSBUT1,UNIT=SYSDA,SPACE=(1500,(35,5),,,ROUND)
//SYSLIN DD *
  INCLUDE SYSLIB(ICDBLDTB)
  INCLUDE SYSLIB(ICDPRSCN)
  INCLUDE SYSLIB(ICDWNSCN)
  INCLUDE SYSLIB(ICDLSSCN)
  INCLUDE SYSLIB(ICDSTSCN)
  INCLUDE SYSLIB(ICDPSCL)
  INCLUDE SYSLIB(ICDPMACS)
  INCLUDE SYSLIB(ICDZERO)
  INCLUDE SYSLIB(ICDTSTYP)
  INCLUDE SYSLIB(ICDIDCHK)
  INCLUDE SYSLIB(ICDPGMCK)
  NAME ICDLDDBG(R)
/*
//*    LINKEDIT DEBUG MEMBERS INTO LOAD MODULE
//*
//STP11 EXEC PGM=IEBGENER,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD SYSOUT=B,DCB=(RECFM=F,BLKSIZE=80)
//SYSUT1 DD LABEL=(16,NL),DISP=(OLD,PASS),VOL=REF=*.STP1.VSB.TAPE,
//       DCB=(LRECL=80,RECFM=FB,BLKSIZE=3200)
//*    PUNCH SAMPLE PROGRAM
//*
//STP12 EXEC  PGM=IEBGENER,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=&&HELPTMP,DISP=(NEW,PASS),UNIT=SYSDA,
//       SPACE=(CYL,(2,2)),DCB=(RECFM=FB,BLKSIZE=3200,LRECL=80)
//SYSUT1 DD  DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200),
//       VOL=REF=*.STP1.VSB.TAPE,LABEL=(07,NL),DISP=(OLD,PASS)
//*    COPY VS BASIC HELP COMMAND AND ERROR MESSAGES
//*
//STP13 EXEC  PGM=IEBGENER,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=&&HELPTMP,DISP=(MOD,PASS),UNIT=SYSDA,
//       SPACE=(CYL,(2,2)),DCB=(RECFM=FB,BLKSIZE=3200,LRECL=80)
//SYSUT1 DD  *
./ ADD NAME=EDIT,LIST=ALL
/*
//*    COPY ADD STATEMENT
//*
//STP14  EXEC PGM=IEBGENER,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=&&HELPTMP,DISP=(MOD,PASS),UNIT=SYSDA,
//       SPACE=(CYL,(2,2)),DCB=(RECFM=FB,BLKSIZE=3200,LRECL=80)
//SYSUT1 DD  DSN=*.STP1.VSB.SYSHELP,DISP=(SHR,PASS),
//       DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280),VOL=REF=*.STP1.VSB.SYSHELP
//*    COPY SYSTEM EDIT DATASET FROM HELP LIBRARY
//*
//STP15 EXEC  PGM=IEBGENER,COND=(8,LE)
//SYSPRINT DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT2 DD DSN=&&HELPTMP,DISP=(MOD,PASS),UNIT=SYSDA,
//       SPACE=(CYL,(2,2)),DCB=(RECFM=FB,BLKSIZE=3200,LRECL=80)
//SYSUT1 DD LABEL=(15,NL),VOL=REF=*.STP1.VSB.TAPE,DISP=(OLD,PASS),
//       DCB=(RECFM=FB,BLKSIZE=3200,LRECL=80)
//*    COPY VS BASIC ERROR MESSAGES
//*
//STP16 EXEC   PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A,SPACE=(CYL,(1,1)),
//       DCB=(RECFM=FB,BLKSIZE=1210,LRECL=121)
//SYSUT2 DD DSN=*.STP1.VSB.HELP,DISP=(OLD,PASS),
//      VOL=REF=*.STP1.VSB.HELP,
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280)
//SYSIN DD DSN=&&HELPTMP,DISP=(OLD,DELETE),UNIT=SYSDA,
//      DCB=(RECFM=FB,BLKSIZE=3200,LRECL=80)
//*     ADD MEMBERS TO PRIVATE HELP LIBRARY
//*
//
```

```
***********************************************************************
*
*
*   5748-XX1 COPYRIGHT IBM CORP. 1974
*
*   VSBINSTL EXEC FOR RELEASE 1 VERSION 1 OF VS BASIC PROGRAM PRODUCT
*
*********************************************************************
&CONTROL OFF
*
*
&IF &INDEX EQ 0 &GOTO -CKMODE
*
&OPT = &1
&IF &OPT EQ INSTALL &GOTO -BASIN
&IF &OPT EQ PTF &GOTO -PTF
&IF &OPT EQ REGEN &GOTO -REGEN
*
&TYPE &OPT IS AN INVALID ARGUMENT
-CKMODE &BEGTYPE
ENTER THE ARGUMENT 'INSTALL','PTF', OR 'REGEN' TO SPECIFY THE PURPOSE
FOR THIS EXEC RUN.
&END
*
&READ ARGS
&IF &INDEX NE 1 &GOTO -REPERR
&OPT = &1
&IF &OPT EQ INSTALL &GOTO -BASIN
&IF &OPT EQ PTF &GOTO -PTF
&IF &OPT EQ REGEN &GOTO -REGEN
&GOTO -REPERR
*
*
*********************************************************************
-BASIN &CONTINUE
*
&BEGTYPE


INSTALLATION FOR VS BASIC PROGRAM PRODUCT(5748-XX1)


&END
*
-CKTRK &CONTINUE
&TRACK = 9TRACK
&DEN = 800
&BEGTYPE

IF THE TAPE BEING INSTALLED IS OTHER THAN 9 TRACK DENSITY 800,
ENTER ITS DENSITY AS FOLLOWS:
   FOR 9 TRACK 6250 ENTER . . . 6250
   FOR 9 TRACK 1600 ENTER . . . 1600
ELSE PRESS RETURN.
```

```
&END
*
&READ ARGS
&IF &INDEX EQ 0 &GOTO -CKDISK
&IF &INDEX GT 1 &GOTO -REPERR
&IF &1 EQ 800 &GOTO -L2
&IF &1 EQ 1600 &GOTO -L2
&IF &1 NE 6250 &GOTO -REPERR
*
-L2 &CONTINUE
&DEN = &1
*
*
-CKDISK &CONTINUE
*
&BEGTYPE

THE SYSTEM DISK TO RECEIVE THIS PRODUCT MUST BE ACCESSED IN
READ/WRITE STATUS AS THE 'A' DISK.  IF NOT, ENTER 'END',
ACCESS THE SYSTEM DISK IN THE PROPER STATUS AND EXECUTE THIS
EXEC AGAIN.
IF IT IS ACCESSED AS THE READ/WRITE 'A' DISK, PRESS RETURN.
&END
*
&READ ARGS
&IF &INDEX EQ 0 &GOTO -SETAC
&IF &1 EQ END &GOTO -RETRY
&GOTO -REPERR
*
-SETAC &CONTINUE
&FM = A
&FM1 = &CONCAT &FM 1
&FM2 = &CONCAT &FM 2
*
&IF &OPT EQ PTF &GOTO -PTFWK
*
&BEGSTACK
VSB1 TEXT
VSB2 TEXT
VSB3 TEXT
VSB4 TEXT
VSB5 TEXT
VSB6 TEXT
VSBMSG LIST
SAMPLE VSBASIC
VSB TXTLIB
&END
*
&STERR = 0
&NUM = 9
&N = 0
&LOOP -ENDLP &N EQ &NUM
&N = &N + 1
&READ ARGS
&FN = &1
```

```
&FT = &2
&ERROR &GOTO -ENDLP
STATE &FN &FT &FM
&STERR = 1
&TYPE '&FN &FT &FM ' ALREADY EXISTS ... ERASE OR RENAME IT.
&BEGTYPE
AND TRY AGAIN.
&END
-ENDLP &CONTINUE
*
&IF &STERR NE 0 &GOTO -STERR
*
&ERROR &GOTO -FDERR
FI TAPE TAP1 (RECFM FB LRECL 80 BLOCK 3440 DEN &DEN &TRACK )
*
FILEDEF VSB1 DISK VSB1 TEXT &FM1 (RECFM F BLOCK 80)
FILEDEF VSB2 DISK VSB2 TEXT &FM1 (RECFM F BLOCK 80)
FILEDEF VSB3 DISK VSB3 TEXT &FM1 (RECFM F BLOCK 80)
FILEDEF VSB4 DISK VSB4 TEXT &FM1 (RECFM F BLOCK 80)
FILEDEF VSB5 DISK VSB5 TEXT &FM1 (RECFM F BLOCK 80)
FILEDEF VSB6 DISK VSB6 TEXT &FM1 (RECFM F BLOCK 80)
FILEDEF VSBMSG DISK VSBMSG LIST &FM1 (RECFM F BLOCK 80)
FILEDEF SAMPLE DISK SAMPLE VSBASIC &FM1 (RECFM F BLOCK 80)
*
&ERROR &GOTO -TAPERR
TAPE REW
TAPE FSF 8
*
&ERROR &GOTO -MOVERR
MOVE TAPE VSB1
MOVE TAPE VSB2
MOVE TAPE VSB3
MOVE TAPE VSB4
MOVE TAPE VSB5
MOVE TAPE VSB6
MOVE TAPE VSBMSG
MOVE TAPE SAMPLE
*
&ERROR &GOTO -TXTERR
TXTLIB GEN VSB VSB1 VSB2 VSB3 VSB4 VSB5 VSB6
*
&E = 0
&LOOP -ERALOOP &E EQ 6
&E = &E + 1
&VSBN = &CONCAT VSB &E
&ERROR &GOTO -ERAERR
ERASE &VSBN TEXT &FM1
-ERALOOP &CONTINUE
*
&ERROR &GOTO -GLOERR
GLOBAL TXTLIB VSB
*
**********************************************************************
-INSTALL &ERROR &CONTINUE
&BEGTYPE
```

```
&END
&STACK HT
LOAD ICDWEXEC ICDJNUCL ICDJNUC1 ICDJNUC2 ICDJNUC3 ICDJNUC4 ICDJNUC5
INCLUDE ICDJDEFR  ICDJRUNA ICDJCMPA ICDJERR ICDJMATV ICDJFUTS ICDJINFO
&STACK RT
INCLUDE ICDJIOVB ICDJVERB ICDJUSFN ICDJUSTB ICDJVREC ICDJDUMY
PRINT LOAD MAP
GENMOD VSBASIC
GENMOD VSBCOMP (FROM ICDJNUCL TO ICDJDUMY)
&STACK HT
LOAD ICDKBFTB ICDKERR ICDKDSUB ICDKGSUB ICDKSSUB ICDKIOVB (ORIGIN 22000
&STACK RT
INCLUDE ICDKVIOR
PRINT LOAD MAP
GENMOD VSBRUN
&STACK HT
LOAD ICDBLDTB ICDPRSCN  ICDWNSCN ICDLSSCN ICDSTSCN  (ORIGIN 2F000
&STACK RT
INCLUDE ICDPSCL ICDPMACS ICDZERO ICDTSTYP ICDIDCHK ICDPGMCK
PRINT LOAD MAP
GENMOD VSBTEST
LOAD ICDLUTIL
PRINT LOAD MAP
GENMOD VSBUTIL
LOAD ICDLHELP
PRINT LOAD MAP
ERASE LOAD MAP
GENMOD VSBHELP
&ERROR &GOTO -RENERR
RENAME VSBASIC MODULE &FM = = &FM2
RENAME VSBCOMP MODULE &FM = = &FM2
RENAME VSBRUN MODULE &FM = = &FM2
RENAME VSBTEST MODULE &FM = = &FM2
RENAME VSBUTIL MODULE &FM = = &FM2
RENAME VSBMSG LIST &FM = = &FM2
RENAME VSBHELP MODULE &FM = = &FM2
RENAME VSB TXTLIB &FM = = &FM2
-END &BEGTYPE


INSTALLATION/REGEN COMPLETE
&END
&EXIT
*
*
**********************************************************************
-PTF &CONTINUE
&BEGTYPE


PTF INSTALLATION FROM TAPE FOR VS BASIC PROGRAM PRODUCT (5748-XX1)

ENTER THE PTF POSITION ON TAPE
```

```
   I.E.      FIRST PTF ENTER ... 1
             SECOND PTF ENTER... 2
             ETC.
&END
*
&READ ARGS
&IF &INDEX NE 1 &GOTO -REPERR
&PTFNO = &1
&GOTO -CKTRK
*
-PTFWK &CONTINUE
&BEGTYPE

SPECIFY THE NAME OF THE TXTLIB TO RECEIVE THE PTF MODULES. BEWARE
THE TXTLIB CANNOT ALREADY CONTAIN MEMBERS WITH THE SAME NAMES AS THOSE
BEING REPLACED.
&END
*
&READ ARGS
*
&IF &INDEX NE 1 &GOTO -REPERR
&TXTNAM = &1
&OP = GEN
&ERROR &GOTO -NEW
STATE &TXTNAM TXTLIB &FM
&OP = ADD
*
-NEW &ERROR &GOTO -NOTEXT
STATE &TXTNAM TEXT &FM
&TYPE '&TXTNAM TEXT &FM ' ALREADY EXISTS ... RENAME OR ERASE IT
&TYPE AND REISSUE THIS EXEC.
&GOTO -STERR
*
-NOTEXT &ERROR &GOTO -FDERR
FI TAPE TAP1 (RECFM FB LRECL 80 BLOCK 3440 DEN &DEN &TRACK )
FI PTF DISK &TXTNAM TEXT &FM1 (RECFM F BLOCK 80)
-PTFLOOP &CONTINUE
&FILES = &PTFNO + &PTFNO - 1
*
-PTFPOS &CONTINUE
&ERROR &GOTO -TAPERR
TAPE REW
TAPE FSF &FILES
*
&ERROR &GOTO -MOVERR
MOVEFILE TAPE PTF
*
&ERROR &GOTO -TXTERR
TXTLIB &OP &TXTNAM &TXTNAM
*
&ERROR &GOTO -ERAERR
ERASE &TXTNAM TEXT
*
&BEGTYPE
```

```
IF THERE IS AN ADDITIONAL PTF ON THIS TAPE WHICH YOU WISH TO
APPLY AT THIS TIME, ENTER THE TAPE POSITION FOR THAT PTF AS
EXPLAINED EARLIER, IF NOT PRESS RETURN.
&END
*
&READ ARGS
&IF &INDEX EQ 0 &GOTO -ENDPTF
&PTFNO = &1
&OP = ADD
&GOTO -PTFLOOP
*
-ENDPTF &CONTINUE
&BEGTYPE

THE REGEN OF THE LOAD MODULES MAY BE DEFERED UNTIL A LATER
TIME.  IF YOU WISH TO DO SO, ENTER 'NOREGEN'.
OTHERWISE PRESS RETURN AND THE REGEN WILL BE DONE NOW.
&END
*
-PTFRA &READ ARGS
&IF &INDEX EQ 0 &GOTO -PTFREGEN
&IF &1 EQ NOREGEN &GOTO -FINIPTF
&TYPE &1 IS AN INVALID REPLY ... TRY AGAIN WITH NOREGEN
&GOTO -PTFRA
*
-PTFREGEN &BEGTYPE

TO REGEN THE MODULES, THE PROPER TXTLIBS MUST BE GLOBALED. IF
'VSB TXTLIB' CREATED AT INSTALLATION TIME IS NOT AVAILABLE
BY THAT NAME, ENTER IT'S CORRECT NAME.  ELSE, PRESS RETURN.
&END
*
&INSTLIB = VSB
*
&READ ARGS
&IF &INDEX EQ 0 &GOTO -REGENV
&INSTLIB = &1
-REGENV &CONTINUE
&TYPE REGEN OF MODULES WITH TXTLIBS &TXTNAM &INSTLIB
*
&ERROR &GOTO -GLOERR
GLOBAL TXTLIB &TXTNAM &INSTLIB
&GOTO -INSTALL
*
-FINIPTF &ERROR &GOTO -RENERR
RENAME &TXTNAM TXTLIB &FM = = &FM2
*
&BEGTYPE
PTF INSTALLATION COMPLETE
&END
&EXIT
*
*
**********************************************************************
-REGEN &CONTINUE
```

```
&BEGTYPE


REGEN OF THE VS BASIC LOAD MODULES

THE SYSTEM SHOULD BE ACCESSED IN READ/WRITE STATUS AS THE 'A'
DISK.  IF NOT, ENTER 'END'; REACCESS THE SYSTEM DISK AND REISSUE
THIS EXEC.
IF IT IS ACCESSED AS THE READ/WRITE 'A' DISK, PRESS RETURN.
&END
*
&READ ARGS
&IF &INDEX EQ 0 &GOTO -SETA1
&IF &1 EQ END &GOTO -RETRY
&GOTO -REPERR
*
-SETA1 &FM = A
&FM1 = &CONCAT &FM 1
&FM2 = &CONCAT &FM 2
*
&BEGTYPE

ENTER THE TXTLIB NAMES TO BE SEARCHED DURING REGEN OF THE COMPILER (TO
A MAXIMUM OF 7).  IF ONLY THE TXTLIB 'VSB' AS CREATED AT INSTALLATION
TIME IS TO BE SEARCHED, PRESS THE RETURN KEY.
&END
*
&TXTLIB = VSB
&READ ARGS
&IF &INDEX EQ 0 &GOTO -REGENLIB
&IF &INDEX GT 7 &GOTO -REPERR
&TXTLIB = &1
*
-REGENLIB &ERROR &GOTO -GLOERR
GLOBAL TXTLIB &TXTLIB &2 &3 &4 &5 &6 &7
&GOTO -INSTALL
*
*
*********************************************************************
-REPERR &TYPE INVALID REPLY
&RETCODE = 1
&EXIT &RETCODE
*
-RETRY &TYPE EXIT FOR SYSTEM DISK ACCESS
&EXIT
*
-STERR &RETCODE = 2
&EXIT &RETCODE
*
-FDERR &RETCODE = 3
&GOTO -ERREXIT
*
-TAPERR &RETCODE = 4
&GOTO -ERREXIT
*
```

```
-MOVERR &RETCODE = 5
&GOTO -ERREXIT
*
-TXTERR &RETCODE = 6
&GOTO -ERREXIT
*
-ERAERR &RETCODE = 7
&GOTO -ERREXIT
*
-GLOERR &RETCODE = 8
&GOTO -ERREXIT
*
-RENERR &RETCODE = 11
*
*
-ERREXIT &TYPE INTERNAL ERROR &RETCODE
&EXIT &RETCODE
*
*
* END OF CMS INSTALLATION PROCEDURE
**********************************************************************
```

DISTRIBUTION TAPE INSTALLATION JCL PROCEDURE FOR DOS/VS

```
*   VS BASIC DOS INSTALLATION
*   5748-XX1 COPYRIGHT IBM CORP.1972
*   REFER TO INSTRUCTIONS ON COPYRIGHT NOTICE, 120-2083
*          NOTE TO USERS
* TO ALLOW USERS TO SKIP JOBS OR INSERT JCL DEFINING LIBRARIES,
* THE SYSTEM WILL PAUSE FOR OPERATOR RESPONSE.  A MESSAGE WILL
* ACCOMPANY EACH PAUSE EXPLAINING THE RESPONSE REQUIRED.
// JOB 1 CONDS OF VS BASIC DOS/VS BATCH
// OPTION LOG
* IF YOU ARE USING PRIVATE LIBRARIES FOR THE CIL AND RLB,
* PLEASE ASSIGN THEM PERMANENTLY AT THIS TIME.
* RESPOND WITH EOB TO CONDENSE THESE LIBRARIES OR CANCEL
// PAUSE
// EXEC MAINT
  DELETR ICD.ALL
  CONDS RL,CL
/&
// JOB 2 CATALR ALL VS BASIC MODULES IN RLB
// EXEC MAINT
              .
              .
              .
 CATALR STATEMENTS FOR THE VS BASIC MODULES
              .
              .
              .
/*
/&
//  JOB 3 LINKEDIT VS BASIC MODULES INTO THE CIL
// OPTION CATAL
  ACTION MAP
  PHASE ICDDSBSC,ROOT
   INCLUDE ICDZEXEC
   INCLUDE ICDQZOPN
   INCLUDE ICDQZPUT
   INCLUDE ICDQZENT
   INCLUDE ICDQZDEL
   INCLUDE ICDQZPNT
   INCLUDE ICDQZGET
   INCLUDE ICDQZCLS
   INCLUDE ICDQZERR
  PHASE ICDJCOMP,*,NOAUTO
   INCLUDE ICDJNUCL
   INCLUDE ICDJNUC1
   INCLUDE ICDJNUC2
   INCLUDE ICDJNUC3
   INCLUDE ICDJNUC4
   INCLUDE ICDJNUC5
   INCLUDE ICDJDEFR
   INCLUDE ICDJRUNA
   INCLUDE ICDJCMPA
   INCLUDE ICDJERR
   INCLUDE ICDJMATV
   INCLUDE ICDJFUTS
   INCLUDE ICDJINFO
```

```
    INCLUDE ICDJIOVB
    INCLUDE ICDJVERB
    INCLUDE ICDJUSFN
    INCLUDE ICDJVREC
   PHASE ICDKRTNS,ICDJCOMP
    INCLUDE ICDKBFTB
    INCLUDE ICDKERR
    INCLUDE ICDKGSUB
    INCLUDE ICDKDSUB
    INCLUDE ICDKSSUB
    INCLUDE ICDKIOVB
    INCLUDE ICDKVIOR
// LBLTYP NSD(4)
// EXEC LNKEDT
/&
// JOB 4 SLF PLACE SOURCE AND MACROS IN SSL
* THIS JOB PLACES MACROS AND A SOURCE MODULE FOR THE SLF
* FACILITY IN THE SSL. IF YOU ARE USING A PRIVATE SSL,
* PLEASE ASSIGN IT PERMANENTLY AT THIS TIME.
* THEN RESPOND EOB TO CONTINUE. OTHERWISE,CANCEL.
// PAUSE
// EXEC MAINT
              .
              .
              .
 CATALS STATEMENT FOR ICDKBFTB
              .
              .
              .
/*
// OPTION LOG,NODECK,EDECK
* TO ASSEMBLE VS BASIC MACROS FOR THE SEPARABLE LIBRARY
* FEATURE, ASSIGN SYSPCH TO A SCRATCH TAPE OR TO A FILE ON DISK
* USING DLBL AND EXTENT CARDS.
// PAUSE
// EXEC ASSEMBLY
   TITLE 'ASSEMBLY OF VS BASIC SLF MACROS'
              .
              .
              .
 CATALS STATEMENTS FOR VS BASIC MACROS
              .
              .
              .
/*
* IF YOU HAVE USED A SCRATCH TAPE,
* PLEASE ASSIGN SYSIPT TO THE SAME TAPE.
* IF YOU HAVE USED A DISK, ASSIGN SYSIPT TO THAT DISK FILE
* RESPOND EOB
// PAUSE
// EXEC MAINT
* DOS/VS BASIC INSTALLATION COMPLETE
/&
```

EXECUTOR MODULES

TSO ONLY

ICDQEXEC

OS/VS ONLY

ICDYEXEC

CMS ONLY

ICDWEXEC

DOS/VS ONLY

ICDZEXEC
ICDZVCLS
ICDZVDEL
ICDZVENT
ICDZVERR
ICDZVGET
ICDZVOPN
ICDZVPNT
ICDZVPUT

TSO AND OS/VS ONLY

ICDQVCLS
ICDQVDEL
ICDQVENT
ICDQVERR
ICDQVGET
ICDWVOPN
ICDQVPNT
ICDQVPUT

COMPILER MODULES

ALL SYSTEMS

ICDJCMPA
ICDJDEFR
ICDJDUMY
ICDJERR
ICDJFUTS
ICDJINFO
ICDJIOVB
ICDJMATV
ICDJNUC1
ICDJNUC2
ICDJNUC3

ICDJNUC4
ICDJNUC5
ICDJNUCL
ICDJRUNA
ICDJUSFN
ICDJVERB

TSO, OS/VS, AND DOS/VS ONLY

ICDJVREC

LIBRARY MODULES

ALL SYSTEMS

ICDKBFTB	ICDKMAT
ICDKCNVT	ICDKMINV
ICDKDSUB	ICDJNCPD
ICDKERR	ICDJORGE
ICDKETOF	ICDKPLIN
ICDKGSUB	ICDKPRNT
ICDKINPT	ICDKREAD
ICDKINTP	ICDKSSUB
ICDKIOVB	ICDKTOUT

TSO, OS/VS, AND DOS/VS ONLY

ICDKKLN
ICDKKPS
ICDKRLN
ICDKVIOR

DEBUG MODULES (TSO AND CMS ONLY)

ICDADRES	ICDIDCHK	ICDONITR	ICDSTSCN
ICDATTN	ICDIFOB	ICDPGMCK	ICDTBACK
ICDATTO	ICDISCAN	ICDPMAC	ICDTSCN
ICDBLDTB	ICDLBKO	ICDPMACS	ICDTSRCH
ICDCHAIN	ICDLISTO	ICDPRSCN	ICDTSTYP
ICDCDSCN	ICDLFQO	ICDPSCL	ICDVSCN
ICDCMTBL	ICDLSSCN	ICDRDIM	ICDWHENO
ICDDBG	ICDMSSG	ICDRUNO	ICDWHRO
ICDDSCAN	ICDMSSGS	ICDSCAN	ICDWNSCN
ICDEVALU	ICDNSCAN	ICDSETO	ICDWNTST
ICDFLOW	ICDOBEY	ICDSSCAN	ICDZERO
ICDFOSUB	ICDOFFO	ICDSSCN	
ICDGOGO	ICDOFFWO	ICDSTCNV	

TSO ONLY

ICDHELPO

MISCELLANEOUS MODULES

TSO ONLY

ICDQRNME
ICDQRNMS

CMS ONLY

ICDLHELP
ICDLUTIL

```
0 REM DO AN OPEN, CLOSE, GET, PUT, RESET, RESET END, EOF, CONV, ETC.
1 B$='SAMPLE'
2 IF X$ = 'CHAIN' GO TO 997
3 C$(1) = 'SYS005'
5 X = FN1 ('SAMPLE')
10 FNA = A+B
15 DIM E$2
20  GO TO 40,30 ON FNA+1
30 LET X=FN2(30)
40  MAT READ X$(3),X(4)
50 FOR X=1 TO 3
55 IF X<>NUM(FNZ$(X$(X))) THEN Y=FN2(50)
57 IF X(X)=6 THEN 59 ELSE Y=FN2(57)
59 NEXT X
65 DEF FN2(I)
70 PRINT 'ERROR AT STATEMENT'; I
80 E=E+1
85 E$='UN'
90 RETURN X
100 REM REM 'ABC'  'XYZ
105 FNEND
110 :*****PROGRAM ######## BEGINNING.
120 REM
130 GO TO 300
150 FN1(A$)
160 IF X$='CHAIN' GO TO 190 ELSE 170
170 PRINT USING 110,A$
178 LET B$=A$
180 RETURN X
190PRINTUSING300,'PROGRAM ';STR(B$,1,LEN(B$));' ENDS ';E$;'SUCCESSFULLY.','****'
200 PRINT USING 320, E
210 RETURN X
220 FNEND
230 MAT GET 'SYS005',C$(5),EOF 250
240 X=FN2(240)
250 FOR X=1 TO 3
253 IF C$(X) =A$(X) THEN 254 ELSE Y=FN2(253)
254 NEXT X
255 CLOSE 'SYS005'
256 PUT 'SYS005','SYS005','TWO','THREE'
259 PUT 'SYS005' , 'FOUR', 5
260 CLOSE C$(RND*0.8+1)
261 GET 'SYS005', R$,S$,T$,U$,U
262 IF T$ <> 'THREE' THEN A=FN2(262)
263 IF U$ <> 'FOUR' THEN A=FN2(263)
264 IF U<>5 THEN A = FN2(264)
270 RESET 'SYS005'
280 GET 'SYS005', T$
290 IF T$ <> 'SYS005' THEN A=FN2(190)
295 GO TO 420
300 FORM POS 6, 3*C, C2, C, POS , C, SKIP 2
320:*****NUMBER OF ERRORS =###
340 DATA 'SYS005','TWO','THREE',3*6,2,1,5,16,19,30,40,25,21,7,2
350 DIM D$(3)
```

```
361  RESTORE
362  MAT READ D$
363  RESTORE
365 MAT A$(3)=D$
370 PUT D$(1), MAT D$
380 CLOSE 'SYS'||'005'
390 FOR J=1 TO 3
395 IF NUM(FNZ$(D$(J))) = J GO TO 400 ELSE Y=FN2(395)
400 GOSUB 600  ON K
402 IF K= 1THEN405
405 NEXT J
410 GO TO 150
420 RESTORE
440 USE X$
450 REM   *************
452 PUT 'SYS009',E,E$
460 GOTO 997
462 X1=FN2(462)
464 GO TO 997
470 DEF FNZ$(B$)
480 IF B$='SYS005' THEN RETURN '1'
490 IF B$='TWO' THEN RETURN '2'
500 IF B$='THREE' THEN RETURN '3' ELSE X1=FN2(500)
510 RETURN '4'
520 FNEND
600 IF ABS(K)=J THEN X1=FN2(600)
610 K=J
620 RETURN
690 RESTORE
700 FOR I=1 TO 3
710 K=2*I-1
720 IF I<> THEN GOSUB 830,888,888 ON ABS(K) ELSE GOSUB 800
722 GO TO 730
725 READ E$(I)
730 NEXT I
740 IF I=3 THEN RETURN ELSE X=FN2(740)
800 IF I=2 THEN RETURN  ELSE X=FN2(800)
810 RETURN
830 IF I=2 THEN X=FN2(830)
840 IF E$(1) <> 'SYS005' & E$ (3) <> 'THREE' THEN 850 ELSE X=FN2(840)
850 GO TO 870,8880 ON I
860 X=FN2(860)
870 IF E$(1) = 'SYS005' THEN X1=FN2(870)
880 IF E$(2)<>'TWO' THEN RETURN ELSE X1=FN2(880)
890 RETURN
995  X1=FN2(995)
996 STOP
997 PRINT '*********************************'
998 CLOSE 'SYS009'
999 GET 'SYS009',E,E$
1000 GOSUB 690
1002 RESTORE
1003  READ MAT D$,G1,H1,J1,K1
1005 FOR I = 1 TO 10
2000 READ W(I)
```

```
3000 NEXT I
4000 N= W(1)
5000 FOR I = 2 TO 10
6000 N = N+W(I)
7000 NEXT I
8000 N= 100/N
9000 K=0
10000 FOR I = 1 TO 10
11000 S(I) = INT(N*W(I)+.5)
12000 K = MAX(K,S(I))
13000 NEXT I
14000 PRINT
15000 PRINT USING 15500,'PERCENTAGE BAR CHART'
15500 FORM POS11,C,SKIP
16000 PRINT
17000 FOR W = K TO 1 STEP -1
17500 Q=W
18000 FOR J=1 TO W/5
18500 Q=Q-5
18600 IF Q=000 THEN PRINT W;
18700 NEXT J
19000 B=(W-(5*A))
20000 IF B = 0 THEN PRINT B
21000 FOR I = 1 TO 10
22000 IF S(I) < B THEN 26000
23300 Z= (3*I+3)
24000 PRINT USING 24500 ,'XX'
24500 FORM POSZ,C
26000 NEXT I
27000 PRINT
28000 NEXT W
29000 PRINT
29100 PRINT USING 29200
29200 FORM POS6
30000 FOR I = 1 TO 10
31000 PRINT USING 31500,I
31500 FORM PIC(Z#),X
32000 NEXT I
32002 X$='CHAIN'
32003 X=FN1(X$)
34000 END
```

OUTPUT FROM SAMPLE PROGRAM

```
*****PROGRAM SAMPLE   BEGINNING.
********************************

          PERCENTAGE BAR CHART

                    XX
                    XX
                    XX
                    XX
 20                 XX
                    XX
                 XX XX
                 XX XX
                 XX XX
 15              XX XX XX
                 XX XX XX
                 XX XX XX XX
                 XX XX XX XX
              XX XX XX XX XX
 10        XX XX XX XX XX XX
           XX XX XX XX XX XX
           XX XX XX XX XX XX
           XX XX XX XX XX XX
           XX XX XX XX XX XX
 5         XX XX XX XX XX XX
           XX XX XX XX XX XX XX
        XX XX XX XX XX XX XX XX
        XX XX XX XX XX XX XX XX
     XX XX XX XX XX XX XX XX XX XX

**** PROGRAM SAMPLE ENDS   SUCCESSFULLY.

*****NUMBER OF ERRORS =  0
R; T=0.64/1.42 11:14:04
```

READER'S COMMENTS

TITLE:	System/370 VS BASIC Installation Reference Material	**ORDER NO.**	SC28-8309-0

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used in preparing updates to the publications. All comments and suggestions become the property of IBM.

Please do not use this form for technical questions about the system or for requests for additional publications; this only delays the response. Instead, direct your inquiries or requests to your IBM representative or to the IBM Branch Office serving your locality.

Corrections or clarifications needed:

Page *Comment*

Please include your name and address in the space below if you wish a reply.

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

SC28-8309-0

cut along this line

fold

fold

FIRST CLASS
PERMIT NO. 33504
NEW YORK, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION
1271 Avenue of the Americas
New York, New York 10020

Attention: PUBLICATIONS

fold

fold

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
[U.S.A. only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

S/370 VS BASIC IRM Printed in U.S.A. SC28-8309-0

SC28-8309-0

S/370 VS BASIC IRM Printed in U. S. A. SC28-8309-0

IBM

International Business Machines Corporation
Data Processing Division
1133 Westchester Avenue, White Plains, New York 10604
(U.S.A. only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)