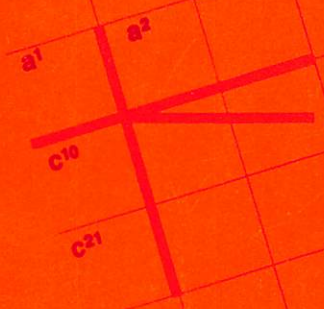
IBM

VS FORTRAN Version 2
Installation and Customization

Licensed Program

$$\sinh 3x = \sinh x(4\cosh^2 x - 1)$$
$$\sinh 4x = \sinh x \cosh x(8\cosh^2 x - 4)$$
$$\sinh 5x = \sinh x(1 - 12\cosh^2 x + 16\cosh^4 x)$$
$$\cosh 3x = \cosh x(4\cosh^2 x - 3)$$
$$\cosh 4x = 1 - 8\cosh^2 x + 8\cosh^4 x$$
$$\cosh 5x = \cosh x(5 - 20\cosh^2 x + 16\cosh^4 x)$$



$$x = \frac{2y - b - a}{b - a}$$

$$f(1 + a^2)^{1/2}$$

Order Number
SC26-4224-1

Program Numbers
5668-805
5668-806

Release 1.1

VS FORTRAN Version 2
Installation and Customization

Licensed Program

Order Number
SC26-4224-1

Program Numbers
5668-805
5668-806

Release 1.1

Second Edition (September 1986)

This is a major revision of, and makes obsolete, SC26-4224-0.

This edition applies to Release 1.1 of VS FORTRAN Version 2, Licensed Programs 5668-805 and 5668-806, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

The changes for this edition are summarized under "Summary of Changes" following the preface ("About This Book"). Specific changes are indicated by a vertical bar to the left of the change. These bars will be deleted at any subsequent republication of the page affected. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

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

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality. If you request publications from the address given below, your order will be delayed because publications are not stocked there.

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

About This Book

This manual is designed for system programmers and planners who supervise the generation and maintenance of an organization's operating system. It contains information about installing VS FORTRAN Version 2 and is to be used in conjunction with the VS FORTRAN Version 2 Program Directory (hereafter referred to as the "program directory") that applies to your system.

This manual is organized to give you information for installing and customizing under MVS in one section, and similar information for VM in a separate section.

We recommend that you read the entire book once (except those chapters that cover the operating system you are not using) before attempting to actually perform any of the steps described.

Summary of Changes

Release 1.1, September 1986

Major Changes to the Product

- Addition of vector directives, including compile-time option (DIRECTIVE) and installation-time option (IGNORE)
- Addition of NOIOINIT execution-time option
- Addition of support for VM/XA System Facility Release 2.0 (5664-169) operating system

Major Changes to This Manual

- Documentation of the above product enhancements
- Addition of appendixes on how to apply service (under both VM and MVS)

Contents

Figures

Chapter 1. Introduction

VS FORTRAN Version 2 is available as two separate products. VS FORTRAN Version 2 (5668-806) is the complete licensed program containing the Compiler, Library, and Interactive Debug. VS FORTRAN Version 2 Library (5668-805) contains only the Library.

Each licensed program is distributed as a single tape containing the necessary modules. VS FORTRAN Version 2 also contains sample programs that verify the installation procedures.

Overview of the Product

The VS FORTRAN Version 2 Compiler translates programs written in the VS FORTRAN Version 2 language and produces object modules for subsequent execution with the support of the VS FORTRAN Version 2 Library.

The VS FORTRAN Version 2 Library contains mathematical, character, bit, service, input/output, and error routines. The Library is designed to support all the features of the VS FORTRAN Version 2 language.

VS FORTRAN Version 2 Interactive Debug allows the programmer to monitor the execution of VS FORTRAN Version 2 programs and to examine and change data at execution time.

The following sections of this book describe the installation requirements and the steps you must take to install VS FORTRAN Version 2, customize it, and service it. We recommend that you read the entire book once, minus the sections that cover the operating system you are not using, before you attempt to actually perform any of the steps described.

Note: Before installing VS FORTRAN Version 2, contact your IBM Support Center or check the RETAIN/370 Preventive Service Planning (PSP) Facility for possible updates to the information and procedures in this book.

Overview of the Installation and Customization Process

Installation is the process of adding to your system the materials on the distribution tape supplied by IBM Information and Software Distribution (ISD). This process produces a fully operational VS FORTRAN Version 2 product.

Customization is an optional process. It gives you the opportunity to change some of the operational characteristics of the standard product to better suit the needs of your site. You can tailor such things as the default values for the compiler options and execution-time options, the individual modules contained in the library composite modules, the error-handling actions, and so forth. The materials you have received contain several macros that will aid you in this process, should you choose to customize.

Where to Find More Installation Information

For specific information on space allocations and other details needed to install VS FORTRAN Version 2, see the program directory for your system. The program directory is shipped in the same package as the installation tape for the VS FORTRAN Version 2 product. The directory describes all the installation materials and gives installation instructions specific to the product release level, the modification level, and the operating system, if any beyond those supplied in this book are necessary.

System, Machine, and Storage Requirements

To install VS FORTRAN Version 2, you need the ISD distribution tape for the VS FORTRAN Version 2 licensed program. You must refer to this manual, the program directory for your system, and the RETAIN/370 PSP facility. You also need to satisfy the following system, machine, and storage requirements before beginning installation of VS FORTRAN Version 2 products:

System Requirements

- VS FORTRAN Version 2 operates under the following operating system environments:
  - VM/System Product (5664-167) Release 3 or later, with or without VM/SP HPO (5664-173) Release 3 or later
  - MVS/System Product Version 1 (5740-XYN or 5740-XYS), all releases, with or without TSO/E
  - MVS/XA: MVS/System Product Version 2 (5665-291 or 5740-XC6), all releases, and MVS/XA DFP Version 1 (5665-284) or MVS/XA DFP Version 2 (5665-XA2) with or without TSO/E
  - VM/XA Systems Facility Release 2.0 (5664-169)

- Execution of vector code compiled by the VS FORTRAN Version 2 Compiler is supported under:
  - MVS/XA: MVS/SP 2.1.3 (5665-291 or 5740-XC6) Vector Facility Enhancement and MVS/XA DFP Version 1 (5665-284) or MVS/XA DFP Version 2 (5665-XA2) with or without TSO/E
  - VM/SP HPO 4.2 with Vector Facility Support
  - VM/XA Systems Facility Release 2.0 (5664-169)
- Execution of scalar code compiled by the VS FORTRAN Version 2 Compiler is supported under the following operating system environments:
  - VM/System Product (5664-167) Release 3 or later, with or without VM/SP HPO Release 3 or later
  - MVS/System Product Version 1 (5740-XYN or 5740-XYS), all releases, with or without TSO/E
  - MVS/XA: MVS System Product Version 2 (5665-291 or 5740-XC6), all releases, and MVS/XA DFP Version 1 (5665-284) or MVS/XA DFP Version 2 (5665-XA2) with or without TSO/E
  - VM/XA Systems Facility Release 2.0 (5664-169)
- If VSAM files are processed under VM, the following program is required:
  - VSE/VSAM (5746-AM2) through Release 3
- Under MVS/370 or MVS/XA, interactive debugging by VS FORTRAN Version 2 requires TSO/E.
- In full screen mode, interactive debugging by VS FORTRAN Version 2 has the following requirements:
  - Under MVS -

    ISPF Version 1 (5668-960). ISPF/PDF Version 1 for MVS (5665-268) is also required if you want to use the browse and edit function.

    or

    ISPF Version 2 for MVS (5665-319). ISPF/PDF Version 2 for MVS (5665-317) is also required if you want to use the browse and edit function. Version 2 is required for enhanced full-screen functions.
  - Under VM -

    ISPF Version 1 (5668-960). ISPF/PDF Version 1 for VM (5664-172) is also required if you want to use the browse and edit function.

    or

ISPF Version 2 for VM (5664-282). ISPF/PDF Version 2 for VM (5664-285) is also required if you want to use the browse and edit function. Version 2 is required for enhanced full-screen functions.

- If customizing VS FORTRAN Version 2 for running on MVS/XA, Assembler H Version 2 is required.

Note: Later versions, releases, and modifications of all of the above products are supported unless explicitly stated otherwise.

Machine Requirements

Before installing VS FORTRAN Version 2, you need the following machine configuration:

- Compile-time Machine Requirements:
  - Any processing unit supported by MVS/SP (with or without TSO), MVS/XA (with or without TSO/E), or VM
  - I/O devices used by the compiler, normally disks
- Execution-time Machine Requirements
  - Any processing unit supported by MVS/SP (with or without TSO), MVS/XA (with or without TSO/E), or VM
  - I/O devices used by the object program during execution
  - An appropriate vector processor, if necessary
- Supported Devices
  - Under MVS/SP, MVS/XA, and VM, IBM devices supported by the BSAM, BDAM, and VSAM access methods can be used by object programs produced by the VS FORTRAN Version 2 compiler when used with the VS FORTRAN Version 2 library.
  - Under VM, any devices supported by VSAM, or by BSAM or BDAM for OS compatibility, are supported by VS FORTRAN Version 2.

Storage Requirements

To install VS FORTRAN Version 2, you need space available on one of the following:

- Under MVS/SP or MVS/XA, space for the various product libraries on your disks
- Under VM, space on 2 target disks

For specific DASD space requirements, refer to the program directory. The VS FORTRAN Version 2 Compiler requires 1400K bytes of virtual storage to handle a typical FORTRAN source program of 100 statements. Storage requirements for the VS FORTRAN Version 2 Library vary according to the customization features selected, and according to the size of user programs.

VS FORTRAN Version 2 Interactive Debug requires about 250K bytes of storage to begin execution, apart from the storage required for the program being debugged.

The main storage requirements for debugging a program with VS FORTRAN Version 2 Interactive Debug vary depending on the function of the user data. Additional dynamic storage is acquired for interactive debugging during execution. The amount varies according to the nature of the program being debugged and the type and quantity of debugging commands issued.

Chapter 2. Installation under MVS

This chapter describes the standard installation of VS FORTRAN Version 2 under MVS/SP and MVS/XA. The general procedure for installing the VS FORTRAN Version 2 Library only is the same as the procedure for installing the VS FORTRAN Version 2 (Compiler, Library, and Interactive Debug).

For specific information on space allocations and other details needed to install VS FORTRAN Version 2, see the program directory. For information on the features you can customize to fit your site's needs, see Chapter 3 on page 33.

Note:

We recommend that you read the entire book once (except Chapter 4, Chapter 5, and Appendix E, which cover VM) **before** you actually begin the installation process.

Basic Machine-Readable Material

The distribution medium for VS FORTRAN Version 2 and VS FORTRAN Version 2 Library is either a standard-labeled 9-track tape or a 3480 tape cartridge. The distribution tape contains SMP modification control statements, JCLIN, modules, macros, and installation jobs and procedures.

See the program directory for the sequence of files and their descriptions.

Storage Requirements

See the program directory for information on the track and directory block space required by VS FORTRAN Version 2.

Data Sets SMP and SMPTLIB

SMP

The following table shows the SMP data sets needed during the installation process. For exact block sizes and DASD space requirements, see the program directory.

SMP4	SMP/E
SMPACDS	SMPE.DATA.CSI
SMPACRQ	SMPE.INDX.CSI
SMPCDS	SMPED.DATA.CSI
SMPCRQ	SMPED.INDX.CSI
SMPLOG	SMPET.DATA.CSI
SMPMTS	SMPET.INDX.CSI
SMPPTS	SMPLOG
SMPSCDS	SMPMTS
SMPSTS	SMPPTS
	SMPSCDS
	SMPSTS

SMPTLIB

SMPTLIB data sets are allocated during the SMP receive process. The DSSPACE subentry of the PTS SYSTEM entry as provided in the installation job on your product tape is large enough to accommodate a maximum SMPTLIB data set. Refer to the program directory for additional size information. The SMPTLIB data sets will be used in the APPLY and ACCEPT steps described under "Preparing to Install VS FORTRAN Version 2" on page 11. They are uncataloged data sets and are deleted after the ACCEPT step is complete.

Target and Distribution Libraries

The following tables show the libraries needed for the installation process. In all cases, the libraries are shown with the default prefix of VSF2. After installing and testing your VS FORTRAN Version 2 product in your own private library, you will want to install the product in your system library. In this case, you should change the VSF2 prefix on all data set names to your system library prefix. You must do this in all installation procedures and jobs where the prefix occurs.

The following table shows the target libraries required for your VS FORTRAN Version 2 product data sets:

Compiler Target Libraries	Library Target Libraries	Interactive Debug Target Libraries
VSF2.VSF2COMP SYS1.PROCLIB SYS1.SAMPLIB	VSF2.VSF2MATH VSF2.VSF2FORT VSF2.VSF2LINK	VSF2.VSF2FORT VSF2.VSF2PLIB VSF2.VSF2MLIB VSF2.VSF2CLIB SYS1.HELP SYS1.SAMPLIB

The following table shows the distribution libraries required for the installation process:

Compiler Distribution Libraries	Library Distribution Libraries	Interactive Debug Distribution Libraries
VSF2.ILXCCM VSF2.ILXCCS	VSF2.AFBLBM VSF2.AFBLBS	VSF2.AFFMOD VSF2.AFFSRC VSF2.AFFPLIB VSF2.AFFMLIB VSF2.AFFCLIB

Note: All the target and distribution libraries must be allocated for each of the VS FORTRAN Version 2 components you are installing.

Library Descriptions

Target Libraries

VSF2COMP Compiler load modules.

PROCLIB Cataloged procedures to run VS FORTRAN Version 2 jobs.

SAMPLIB Sample FORTRAN source programs and release migration tool.

VSF2MATH Alternative mathematical routines.

VSF2FORT Execution-time library routines needed for creation of an executable program and for routines that may be loaded during execution or when Interactive Debug is used.

VSF2LINK Interface routines used in link mode only. This library must be concatenated ahead of VSF2FORT for the creation of an executable program in link mode.

VSF2PLIB Interactive Debug ISPF Panel library.

VSF2MLIB Interactive Debug ISPF Message library.

VSF2CLIB Interactive Debug ISPF CLIST library.

HELP Interactive Debug TSO Help.

Distribution Libraries

ILXCCM Compiler load modules.

ILXCCS Compiler macros and cataloged procedures.

AFBLBM Execution-time library load modules.

AFBLBS Execution-time library macros and SMP installation procedures.

AFFMOD Interactive Debug load modules.

AFFSRC Interactive Debug TSO Help and sample program.

AFFPLIB Interactive Debug ISPF Panel library.

AFFMLIB Interactive Debug ISPF Message library.

AFFCLIB Interactive Debug ISPF CLIST library.

Installation Overview

In the following discussions these jobs and procedures are referred to by generic names such as ILXyALOC or VSFINTyz, where "y" represents the version of SMP you are using—either SMP4 ("4") or SMP/E ("E")—and "z" represents the product being installed—either the whole product ("C") or the Library only ("L").

To install VS FORTRAN Version 2, you must take the following steps:

1. Unload the SMP installation jobs and procedures from the ISD tape. These jobs and procedures are described in more detail under "Preparing to Install VS FORTRAN Version 2" on page 11.

2. Examine the installation jobs and procedures and change them as necessary to suit your local requirements. If you don't want Interactive Debug installed, edit the installation jobs and procedures and remove all references to Interactive Debug data sets and Interactive Debug function modification identifiers (FMIDs). See the program directory for a listing of VS FORTRAN Version 2 FMIDs and their descriptions.

3. Execute the ILXyALOC or AFByALOC job to allocate data sets that SMP will require. This job performs the following functions:

   a. Invokes the VSFALOyz procedure to allocate SMP data sets

   b. Invokes the VSFPROCz procedure to allocate product data sets

4. Execute the ILXyINST or AFByINST job to install VS FORTRAN Version 2. This job performs the following functions:

   a. Invokes the VSFINTyz procedure to initialize SMP data sets.

   b. Invokes the VSFACCyz procedure to access the data sets required to receive and apply the product.

5. Optionally, install Interactive Debug. There are separate procedures for both ISPF and non-ISPF users.

6. Optionally, run the sample program to verify that installation is complete. If you are installing Interactive Debug, you can also verify that installation of Interactive Debug is successful by running the Interactive Debug sample program.

7. Execute the ILXyACPT or AFByACPT job to accept the product into the target and distribution libraries. This job invokes the VSFACCyz procedure to perform the SMP accept.

After installation is complete, you can customize your VS FORTRAN Version 2 product by using the methods described in Chapter 3, "Customization under MVS" on page 33.

Preparing to Install VS FORTRAN Version 2

The following sections give detailed descriptions of the steps outlined in the installation overview.

To install VS FORTRAN Version 2, use the System Modification Program Release 4 (SMP4) or System Modification Program Extended (SMP/E). Refer to *System Modification Program (SMP): System Programmer's Guide*, GC28-0673, or *System Modification Program Extended (SMP/E): User's Guide*, SC28-1302, for information regarding the use of SMP.

Four sets of installation jobs and procedures are provided:

- To install the VS FORTRAN Version 2 Compiler, Library and Interactive Debug using SMP4, you will use these jobs:

  ILX4ALOC, ILX4INST, ILX4ACPT which invoke these procedures:

  VSFALO4C, VSFINT4C, VSFPROCC, VSFACC4C

- To install the VS FORTRAN Version 2 Compiler, Library and Interactive Debug using SMP/E, you will use these jobs:

  ILXEALOC, ILXEINST, ILXEACPT which invoke these procedures:

  VSFALOEC, VSFINTEC, VSFPROCC, VSFACCEC

- To install the VS FORTRAN Version 2 Library product only, using SMP4, you will use these jobs:

  AFB4ALOC, AFB4INST, AFB4ACPT which invoke these procedures:

  VSFALO4L, VSFINT4L, VSFPROCL, VSFACC4L

- To install the VS FORTRAN Version 2 Library product only, using SMP/E, you will use these jobs:

  AFBEALOC, AFBEINST, AFBEACPT which invoke these procedures:

  VSFALOEL, VSFINTEL, VSFPROCL, VSFACCEL

Each step of the installation process is described below, including that of setting up and verifying the success of Interactive Debug. Where differences exist in how to install VS FORTRAN Version 2, either with or without the Compiler and Interactive Debug while using either SMP4 or SMP/E, they are described in detail. Use the set of instructions that fits your needs.

Installing VS FORTRAN Version 2

Step 1: Unloading the SMP Installation Procedures and Jobs

With the sample Job Control Language (JCL) shown in Figure 1 on page 13, you can load the installation procedures from the ISD tape to disk. By executing this JCL, you copy the installation procedures and jobs off the distribution tape; the system then catalogs them in a data set and prints a listing. The listing is useful if you must modify either the JCL or SMP statements; for example, to change the allocation parameters when a device other than an IBM 3330 device is used.

Note that the following sample JCL will install the entire product. To install the library only, replace the line

```
SELECT MEMBER=(ILXyALOC,ILXyINST,ILXyACPT)
```

with

```
SELECT MEMBER=(AFByALOC,AFByINST,AFByACPT)
```

Before you execute the JCL you must choose the set of installation jobs and procedures that match the VS FORTRAN Version 2 product you are installing and the SMP product you are using, and set up the SELECT MEMBER statement accordingly. Notes at the bottom of Figure 1 will help you make these and other required changes.

```
//GETPROCS JOB  .... (user information)
//*
//*   GET SMP INSTALLATION PROCEDURES FROM ISD TAPE
//*
//        EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=A
//IN       DD  DSN=filename,UNIT=tape,VOL=SER=volser,DISP=SHR,
//             LABEL=(x,SL)                                    (See Note 1)
//OUT      DD  DSN=VSF2.INSTALL,DISP=(NEW,PASS),SPACE=(TRK,(1,1,2)),
//             UNIT=SYSDA,VOL=SER=volid,                       (See Note 2)
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=4000)
//SYSUT3   DD  SPACE=(TRK,(1)),UNIT=SYSDA
//SYSUT4   DD  SPACE=(TRK,(1)),UNIT=SYSDA
//SYSIN    DD  *
 COPY INDD=IN,OUTDD=OUT
 SELECT MEMBER=(VSFALOyz,VSFINTyz,VSFPROCz,VSFACCyz)       (See Note 3)
 SELECT MEMBER=(ILXyALOC,ILXyINST,ILXyACPT)                (See Note 3)
//*
//*   PRINT THE PROCEDURES
//*
//    EXEC PGM=IEBPTPCH
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=VSF2.INSTALL,DISP=(OLD,CATLG)
//SYSUT2   DD  SYSOUT=A
//SYSIN    DD  *
      PRINT TYPORG=PO,MAXFLDS=1
      RECORD FIELD=(80)
/*
```

Notes:

1. See the program directory for the specific names and file position you need to insert in place of "x," "filename," and "volser" in the example above. "Tape" is the tape unit designation you supply.

2. "Volid" is the volume serial number you supply.

3. The member names to specify for your installation of VS FORTRAN Version 2 are either:

   a. Entire product using SMP4 (y=4, z=C):
      VSFALO4C,VSFINT4C,VSFPROCC,VSFACC4C,ILX4ALOC,ILX4INST,ILX4ACPT

   b. Entire product using SMP/E (y=E, z=C):
      VSFALOEC,VSFINTEC,VSFPROCC,VSFACCEC,ILXEALOC,ILXEINST,ILXEACPT

   c. Library product only, using SMP4 (y=4, z=L):
      VSFALO4L,VSFINT4L,VSFPROCL,VSFACC4L,AFB4ALOC,AFB4INST,AFB4ACPT

   d. Library product only, using SMP/E (y=E, z=L):
      VSFALOEL,VSFINTEL,VSFPROCL,VSFACCEL,AFBEALOC,AFBEINST,AFBEACPT

Figure 1. GETPROCS Job for VS FORTRAN Version 2

Step 2: Preparing the Installation Jobs and Procedures for Use

The installation jobs and the installation procedures they invoke are shown in the tables below. Refer to the table which corresponds to the type of installation you are doing.

Job Name	Job Function	Invokes Procedure	Procedure Function
ILXyALOC	Allocates all data sets needed to install VS FORTRAN Version 2	VSFALOyz	Allocates SMP data sets
		VSFPROCz	Allocates VS FORTRAN Version 2 distribution and target data sets
ILXyINST	Receives and applies VS FORTRAN Version 2	VSFINTyz	Initializes SMP data sets
		VSFACCyz	Accesses all data sets needed to install VS FORTRAN Version 2
ILXyACPT	Accepts the VS FORTRAN Version 2 product.	VSFACCyz	Accesses all data sets needed to accept VS FORTRAN Version 2

Figure 2. Installation Jobs and the Procedures They Invoke—Entire Product

Job Name	Job Function	Invokes Procedure	Procedure Function
AFByALOC	Allocates all data sets needed to install VS FORTRAN Version 2	VSFALOyz	Allocates SMP data sets
		VSFPROCz	Allocates VS FORTRAN Version 2 distribution and target data sets
AFByINST	Receives and applies VS FORTRAN Version 2	VSFINTyz	Initializes SMP data sets
		VSFACCyz	Accesses all data sets needed to install VS FORTRAN Version 2
AFByACPT	Accepts the VS FORTRAN Version 2 product.	VSFACCyz	Accesses all data sets needed to accept VS FORTRAN Version 2

Figure 3. Installation Jobs and the Procedures They Invoke—Library Only

There are several steps you must take to adapt the installation jobs to your site's specific needs. You can, if you prefer, write your own jobs to invoke the procedures provided on the tape. When you use the IBM-supplied jobs, you must take the following steps:

- Following the prolog of each of the three installation jobs is a list of search keys for global variables for such things as data set prefixes, volume serial numbers, generic unit names and block sizes. You **must** supply your own definitions for these variables. You can do this by using an editor to globally replace the search keys and the variables with the values you need at your site. The search keys in the list and the variables coded in the job all begin with "?".

- You must also search for other optional statements in the jobs and select or modify them as instructed by comments in the jobs. You can do this by searching for the word "optional" embedded within the comments of the JCL.

You might also need to make some changes to the procedures:

- The procedures define variables by using symbolic parameters. If you are using the IBM-supplied jobs to invoke the installation procedures, the variable definitions you supply in the jobs define the symbolic parameters in the procedures they invoke.

  If you are using your own jobs instead of those supplied by IBM, you can allow the symbolic parameters to default to the definitions provided on the PROC statements of the procedures, or you can override the defaults by using the EXEC statements that invoke the procedures from your jobs.

- If you are not installing Interactive Debug, delete all references to Interactive Debug data sets and FMIDs. All IAD statements are preceded by the comment delimiter "*." The Interactive Debug data sets are listed under "Target and Distribution Libraries" on page 8. The FMIDs are listed in the program directory and also at the beginning of the installation jobs.

- If you are using storage devices other than the IBM 3330, you must change data set space allocations. Refer to the program directory for precise information on block sizes and DASD space requirements.

- If you are mixing unit types, data set prefixes, or volumes, you must specify these instead of relying on the symbolic parameters.

Preparing the VSFALOyz and VSFINTyz Procedures

The two procedures VSFALOyz and VSFINTyz set up the SMP environment. VSFALOyz allocates the required data sets; VSFINTyz initializes them. You should execute VSFALOyz only if you do not intend to use existing SMP data sets.

If you are using existing SMP data sets, the space and directory blocks allocated are required in addition to existing allocations. You should have a distribution library (DLIB) volume with adequate space for the VS FORTRAN Version 2 data sets; the space requirements are included in the JCL and are also listed in the program directory.

These data sets are used only for software service, and the volume containing them must be online only when VS FORTRAN Version 2 is being updated.

The following procedure keywords and default values are used in the VSFALOyz procedures:

SMPPRFX = 'VSF2'
Data set name prefix to use for all SMP data sets

SMPVOL = 'VSFRES'
Serial number of the volume that is to contain the SMP data sets

UNIT = 'SYSDA'
Type of device on which the volume is mounted

BLKSZ = '3120'
Block size to be used when allocating all SMP data sets

The following procedure keyword and default value are used only in the VSFALOEC and VSFALOEL procedures:

CATDSN = 'SYS1.ICFCAT.VSMPRES'
Private VSAM catalog DSNAME

The following procedure keywords and default values are used in the VSFINTyz procedures:

SMPPRFX = 'VSF2'
Data set name prefix used for all SMP data sets

SOUT = 'A'
SYSOUT class to use for all printable output

The following additional procedure keywords and default values are used only in the VSFINTEC and VSFINTEL procedures:

SMPVOL = 'VSFRES'
Serial number of the volume that is to contain the SMPTLIB (temporary RELFILE) data set

UNIT = 'SYSDA'
Type of device on which the SMPTLIB data set volume is mounted

Preparing the VSFPROCz Procedure

This procedure allocates the data sets that contain VS FORTRAN Version 2 load modules and macros. Allocations given in the procedure are for a 3330 device; you must adjust for other device types. If you choose to add your VS FORTRAN Version 2 data sets to existing libraries such as SYS1.SAMPLIB, you must allocate enough space for the existing library **plus** the new VS FORTRAN Version 2 data set. If you want to allocate private libraries for these data sets, you must change the default data set name prefix.

If you are not installing Interactive Debug, you do not need to allocate Interactive Debug data sets as listed in the section "Target and Distribution Libraries" on page 8. You should delete the DD statements for these data sets or make them into comments by inserting an asterisk (*) after the // preceding them.

The following procedure keywords and default values are used in the VSFPROCz procedures:

DISPRFX = 'VSF2'
Data set name prefix to use for all distribution library data sets

TARPRFX = 'VSF2'
Data set name prefix to use for all target library data sets

DISVOL = 'VSFRES'
Volume serial number of the volume that is to contain the distribution library data sets

TARVOL = 'VSFRES'
Volume serial number of the volume that is to contain the target library data sets

UNIT = 'SYSDA'
Type of device on which the volume is mounted

BLKSZ = '3120'
Block size to be used when allocating the AFFSRC, AFFPLIB, AFFMLIB, AFFCLIB, ILXCCS, and AFBLBS distribution library data sets

Preparing the VSFACCyz Procedure

This procedure invokes SMP to access the data sets needed for initial installation or periodic service of the product. For ease of reference, the data sets to which it refers have been grouped as:

- SMP data sets
- FORTRAN data sets

The following procedure keywords and default values are used in the VSFACCyz procedures:

SMPPRFX = 'VSF2'
Data set name prefix used for all SMP data sets

DISPRFX = 'VSF2'
Data set name prefix to use for all distribution library data sets

TARPRFX = 'VSF2'
Data set name prefix to use for all target library data sets

SAMPRFX = 'VSF2'
Data set name prefix used for the SAMPLIB target library data set. This data set must have been previously allocated.

HELPRFX = 'VSF2'
Data set name prefix used for the system HELP target library data set. This data set must have been previously allocated.

PROPRFX = 'VSF2'
Data set name prefix used for the system PROCLIB target library data set. This data set must have been previously allocated.

SMPVOL = 'VSFRES'
Volume serial number of the volume that contains, or is to contain, the SMPTLIB (temporary RELFILE) data set

UNIT = 'SYSDA'
Type of device on which the SMPTLIB data set volume is mounted and which is to be used when allocating the SMP work data sets

BLKSZ = '3120'
Block size to be used when allocating the SMP work data sets

SOUT = 'A'
SYSOUT class to use for all printable output

The following procedure keyword and default value are used only in the VSFACCEC and VSFACCEL procedures:

CATDSN = 'SYS1.ICFCAT.VSMPRES'
Private VSAM catalog DSNAME

Using the Procedures

After making the required changes to the four installation procedures, insert them into either a PROCLIB or copy them into the installation jobs that invoke them. If you copy the procedures into the jobs, you must place them before the EXEC statement that calls them, and add the following statement after the last line of each procedure:

```
// PEND
```

See Figure 2 on page 14 for a reminder of which procedures are invoked by which jobs.

Step 3: Executing the Job to Allocate Data Sets

This job is named ILXyALOC (if you are installing the entire VS FORTRAN Version 2 product), or AFByALOC (if you are installing the library only), where "y" represents the version of SMP you are using—either SMP4 ("4") or SMP/E ("E").

This job invokes the VSFALOyz and VSFPROCz procedures to allocate the SMP data sets and VS FORTRAN Version 2 target and distribution libraries that SMP will need in step 4.

Before you run this job, examine its contents and make any changes necessary to suit your particular requirements:

1. All search keys and variables beginning with "?" must be changed to the values you have decided upon for your site. You can use an editor to make these changes globally.

2. Procedures VSFALOyz and VSFPROCz must either be inserted into a PROCLIB, or copied into the ILXyALOC (or AFByALOC) job. If the procedures are copied into the ILXyALOC (or AFByALOC) job, then you must remember to add the following statement after the last line of each procedure:

   ```
   // PEND
   ```

3. There are several statements provided in the job but treated as comments. They can be found by searching for the word "optional." If you need to use any of these statements, you can make them active by changing the "//*" preceding them to "//".

4. If you are using existing SMP data sets, you should not allocate new ones. Delete or make into comments all lines of the EXEC PROC=VSFALOyz statement by changing the "//" preceding them to "//*".

5. In the ILXEALOC (or AFBEALOC) job for use with SMP/E, there are statements that define a VSAM user catalog. If a VSAM user catalog has not already been defined, you can use the UCAT.SYSIN DEFINE USERCATALOG statements following the VSFALOEz procedure invocation to do so.

Step 4: Executing the Job to Install VS FORTRAN Version 2

This job is named ILXyINST (if you are installing the entire VS FORTRAN Version 2 product), or AFByINST (if you are installing the library only), where "y" represents the version of SMP you are using—either SMP4 ("4") or SMP/E ("E").

This job invokes the VSFINTyz and VSFACCyz procedures described above tc accomplish the installation. The job initializes the SMP data sets, then receives and applies the licensed program.

Remember to change all search keys and variables beginning with "?" to the values you have decided upon for your site.

When using **SMP4** to install VS FORTRAN Version 2 (the ILX4INST job for installing the whole product: the AFB4INST job for installing the Library only), do the following:

1. Procedures VSFINTyz and VSFACCyz must either be inserted into a PROCLIB, or copied into the ILX4INST (or AFB4INST) job before the EXEC statement which invokes that procedure. If the procedures are copied into the ILX4INST (or AFB4INST) job, then you must remember to add the following statement after the last line of each procedure:

   ```
   // PEND
   ```

2. SMPCNTL statements are provided following the VSFINT4z procedure invocation to define the SMPCDS, SMPPTS, and SMPACDS initialization parameters. If you do not have Assembler H Version 2 at your site, change the UCLIN PTS command to specify `ASMNAME(IEUASM)` instead of `ASMNAME(IEV90)`.

3. The DSSPACE parameter defines the SMPTLIB space values in tracks for a 3330 device. You may want to change this parameter if you are using a larger capacity device.

4. The SMPCNTL RECEIVE and APPLY statements that follow the VSFACC4z procedure invocation list the FMIDs to be received and applied during installation. If you do not wish to use Interactive Debug at your site, delete the RECEIVE and APPLY statements that specify the Interactive Debug FMIDs. Refer to the job prolog or the program directory for a list of FMIDs and their contents.

If you are installing in a target library that has not previously contained this product, message **IEW0342** is generated during link-editing. You will receive message **IEW0461** from the linkage editor during apply processing of the library due to unresolved library routines. A condition code of 4 will result from SMP, and a condition code of 4 or 8 will result from the linkage editor. These resulting messages and condition codes are normal and may be **ignored.**

When using **SMP/E** to install VS FORTRAN Version 2 (the ILXEINST job for installing the whole product; the AFBEINST job for installing the Library only): do the following:

1. Procedures VSFINTyz and VSFACCyz must either be inserted into a PROCLIB, or copied into the ILXEINST (or AFBEINST) job before the EXEC statement which invokes that procedure. If the procedures are copied into the ILXEINST (or AFBEINST) job, then you must remember to add the following statement after the last line of each procedure:

   ```
   // PEND
   ```

2. SMPCNTL statements, which follow the VSFINTEz procedure invocation, initialize the global, distribution and target zones. If you are not using Assembler H Version 2 at your site, change the ADD UTILITY statement's NAME parameter to specify `NAME(IEUASM)` instead of `NAME(IEV90)`.

3. The DSSPACE parameter defines the SMPTLIB space values in tracks for a 3330 device. You may want to change this parameter if you are using a larger capacity device.

4. Other control variables also may require changing. Check the comments in the job to determine what these changes are.

5. The SMPCNTL RECEIVE and APPLY statements that follow the VSFACCEz procedure invocation list the FMIDs to be received and applied during installation. If you do not wish to use Interactive Debug at your site, delete the FMIDs for Interactive Debug data sets. Refer to the program directory for a list of FMIDs and their contents.

If you are using SMP/E prior to Release 3, and if you are installing in a target library that has not previously contained this product, message **IEW0342** is generated during link-editing. You may also receive message **IEW0461** from the linkage editor during apply processing of the library. A condition code of 4 may result from SMP, and a condition code of 4 or 8 from the linkage editor. These resulting messages and condition codes are normal and may be **ignored.**

If you are using SMP/E release 3 or later you will not receive message IEW0342.

Step 5: Installing VS FORTRAN Version 2 Interactive Debug

If you are an ISPF (Interactive System Product Facility) user, proceed with the following section "Installing Interactive Debug (ISPF/PDF Users Only)." If you are *not* an ISPF user, go to the section "Installing Interactive Debug (Non-ISPF Users Only)" on page 26.

If you are *not* installing VS FORTRAN Version 2 Interactive Debug, skip this step and go to "Step 6: Verifying a Successful Installation" on page 27.

Installing Interactive Debug (ISPF/PDF Users Only)

1. Using ISPF Edit, modify foreground selection panel ISRFPA, located in your site's ISPF panel data set. (Figure 4 on page 22 is an example of panel ISRFPA being updated for VS FORTRAN Version 2 Interactive Debug.)

   To set up Interactive Debug, change any option at the top of the screen to specify VS FORTRAN Version 2 Interactive Debug and change the correspondingly numbered line at the bottom of the screen to specify AFFFP11.

   For example, as shown in Figure 4, you can change option 11 from the old FORTRAN interactive debug product to VS FORTRAN Version 2 Interactive Debug, and change entry 11 at the bottom of the panel from ISRFP11 to AFFFP11 (enter in uppercase).

OR — you may **add** VS FORTRAN Version 2 Interactive Debug to your ISPF panel. Enter

```
%xx+- VS FORTRAN 2 Interactive Debug
```

on the upper part of the panel (where xx is the number of the option). You must also enter

```
xx,'PGM(ISRFPR) PARM(AFFFP11) NEWPOOL'
```

on the lower part of the panel. During installation, AFFFP11 will be included in the library containing the ISPF panel definitions of VS FORTRAN Version 2 Interactive Debug (VSF2.VSF2PLIB).

```
%----------------------  FOREGROUND SELECTION PANEL  -------------------------
%OPTION  ===>_ZCMD                                                          +
%
%    1+- System assembler                  % 7+- Linkage editor
%    2+- OS/VS COBOL compiler              % 8+- Load
%    3+- VS FORTRAN compiler               % 9+- SCRIPT/VS
%    4+- PL/I checkout compiler            %10+- COBOL interactive debug
%    5+- PL/I optimizing compiler          %11+- VS FORTRAN V2 interactive debug
%    6+- PASCAL/VS compiler                %12+- Member parts list
%                                                                    |
+                                                                   |
+SOURCE DATA PACKED%===>_ZFPACK  +(YES or NO)                        |
)INIT                                                    (name change)
  .HELP = ISR40000
  IF (&ZXPACK ¬= '        ')
    &ZFPACK = &ZXPACK
    &ZXPACK = '        '
  &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,*,NO) /* DATA FORMAT CHECK  */
)REINIT
  &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,*,NO) /* DATA FORMAT CHECK  */
)PROC
  &ZFPACK = TRUNC(&ZFPACK,1)
  VER (&ZFPACK,NB,LIST,Y,N)          /* Y=EXPAND PACKED DATA         */
  &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,N,NO)
  VPUT (ZFPACK) PROFILE
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                1,'PGM(ISRFPR) PARM(ISRFP01) NEWPOOL'
                2,'PGM(ISRFPR) PARM(ISRFP02) NEWPOOL'
                3,'PGM(ISRFPR) PARM(ISRFP03) NEWPOOL'
                4,'PGM(ISRFPR) PARM(ISRFP04) NEWPOOL'
                5,'PGM(ISRFPR) PARM(ISRFP05) NEWPOOL'
                6,'PGM(ISRFPR) PARM(ISRFP06) NEWPOOL'
                7,'PGM(ISRFPR) PARM(ISRFP07) NEWPOOL'
                8,'PGM(ISRFPR) PARM(ISRFP08) NEWPOOL'
                9,'PGM(ISRFPR) PARM(ISRFP09) NEWPOOL'
               10,'PGM(ISRFPR) PARM(ISRFP10) NEWPOOL'
               11,'PGM(ISRFPR) PARM(AFFFP11) NEWPOOL'<-------
               12,'PGM(ISRFPR) PARM(ISRFP12) NEWPOOL'          |
              ,' ',' '                                         |
                *,'?' )                                        |
)END                                          (required change)
```

Figure 4. ISPF Foreground Selection Panel Definition

2. Using ISPF Edit, modify foreground help panel ISR40000, which is found in your location's ISPF panel library (normally, ISRPLIB). Figure 5 is an example of this panel, and identifies the two fields that you need to change. Change an option as you did in the foreground selection panel (or add an option here if you added one there), and change (or add) the correspondingly numbered entry in the bottom left part of the panel to specify AFF41100. (These changes are similar to the ones you made in step 1, and you can make them in the same way.)

   The name AFF41100 is used to obtain the VS FORTRAN Version 2 Interactive Debug primary Help panel. During installation, AFF41100 will be included in the library containing the VS FORTRAN Version 2 Interactive Debug ISPF panel definitions (VSF2.VSF2PLIB).

```
%TUTORIAL ------------- FOREGROUND PROCESSING OPTION ---------------- TUTORIAL
%OPTION  ===>_ZCMD                                                          +
%                        ----------------------------------
                         |     FOREGROUND PROCESSING      |
                         ----------------------------------
   The foreground processing option allows certain processing programs to
   be executed in the foreground under ISPF.  The foreground selection menu
   which is displayed when option%4+is entered on the primary option menu
   allows the selection of one of these processing programs.

 The following topics are presented in sequence, or may be selected by number:
    %0+- Foreground general information   % 6+- PASCAL/VS compiler
    %1+- System assembler                % 7+- Linkage editor
    %2+- OS/VS COBOL compiler            % 9+- SCRIPT/VS
    %3+- VS FORTRAN compiler             %10+- COBOL interactive debug
    %4+- PL/I checkout compiler          %11+- VS FORTRAN V2 interactive debug
    %5+- PL/I optimizing compiler        %12+- Member parts listing
                                                                    |
)PROC                                                               |
   &ZSEL = TRANS( &ZCMD                                             |
                 0,ISR40001                                  (name change)
                 1,ISR41000
                 2,ISR42000
                 3,ISR43000
                 4,ISR44000
                 5,ISR45000
                 6,ISR46000
                 7,ISR47000
                 9,ISR49000
                10,ISR4A000
                11,AFF41100  <-------- (required change)
                12,ISR4C000
                 )
   &ZUP = ISR00003
)END
```

Figure 5. ISPF Foreground Processing Help Panel

3. In order to compile VS FORTRAN Version 2 programs through the ISPF environment, modify the CLIST ISRFC03, which is found in your location's ISPF CLIST data set.

   Replace the line

   ```
   ISPEXEC SELECT PGM(FORTVS)
   ```

   with

   ```
   ISPEXEC SELECT PGM(FORTVS2)
   ```

4. Allocate the data sets needed to invoke ISPF in *one* of the following ways:

   - Use an editor to build or modify a CLIST. This CLIST must include ALLOC statements to allocate the libraries created during installation of VS FORTRAN Version 2 Interactive Debug. (See Figure 6 on page 25 for an example of a CLIST containing the required names.)

   - Build or modify the TSO logon procedure. See Figure 7 on page 26 for a sample logon procedure. If VSF2COMP and VSF2FORT are in LNKLST, then it is not necessary to include them in STEPLIB.

   The data sets VSF2.VSF2CLIB, VSF2.VSF2MLIB, and VSF2.VSF2PLIB are distributed in fixed block format. If this format is incompatible with the format of the concatenated data sets, you may have to modify the attributes of the data sets so that proper concatenation will occur.

   For example, to modify the attributes of the VSF2.VSF2CLIB data set, take the following steps:

   a. Allocate a new data set for VSF2.VSF2CLIB with the correct attributes and a different name.

   b. Copy the contents of VSF2.VSF2CLIB into your new data set.

   c. Delete the original copy of VSF2.VSF2CLIB.

   d. Rename the new data set VSF2.VSF2CLIB.

5. Go to "Step 6: Verifying a Successful Installation" on page 27.

```
PROC 0
  CONTROL NOLIST NOSYMLIST NOCONLIST  NOFLUSH  NOMSG
   /*********************************************************************/
   /*                                                                   */
   /* THIS IS AN EXAMPLE OF A CLIST FOR SETTING UP ISPF, WHICH          */
   /* INCLUDES DEFINITIONS FOR THE DATA SETS REQUIRED BY VS FORTRAN     */
   /* VERSION 2 INTERACTIVE DEBUG.  THE ISPF NAMES SHOWN IN THIS        */
   /* EXAMPLE MAY BE DIFFERENT FROM THE ONES USED AT YOUR LOCATION.     */
   /*                                                                   */
   /*********************************************************************/
   FREE  FI(SYSPROC ISPMLIB ISPPLIB ISPLLIB)
   ALLOC FI(SYSPROC) DA('VSF2.VSF2CLIB' +
                        'ISP.V2R1M0.ISPCLIB' +
                        'ISR.V2R1M0.ISRCLIB' ) SHR REUSE
   ALLOC FI(ISPMLIB) DA('VSF2.VSF2MLIB' +
                        'ISP.V2R1M0.ISPMLIB' +
                        'ISR.V2R1M0.ISRMLIB' ) SHR REUSE
   ALLOC FI(ISPPLIB) DA('VSF2.VSF2PLIB' +
                        'ISP.V2R1M0.ISPPLIB' +
                        'ISR.V2R1M0.ISRPLIB' ) SHR REUSE
   /*********************************************************************/
   /*                                                                   */
   /* THERE IS NO SKELETON FILE FOR VS FORTRAN VERSION 2 INTERACTIVE    */
   /* DEBUG.  THE NORMAL ALLOCATION FOR ISPSLIB CAN BE USED.            */
   /*                                                                   */
   /* THERE IS NO TABLE FILE FOR VS FORTRAN VERSION 2 INTERACTIVE       */
   /* DEBUG.  THE NORMAL ALLOCATION FOR ISPTLIB CAN BE USED.            */
   /*                                                                   */
   /* IF THE HELP MEMBER FOR VS FORTRAN VERSION 2 INTERACTIVE DEBUG     */
   /* WAS NOT INSTALLED IN SYS1.HELP, THE CORRECT DATA SET NAME         */
   /* SHOULD BE CONCATENATED TO SYS1.HELP.  FOR EXAMPLE:                */
   /* FREE  FI(SYSHELP)                                                 */
   /* ALLOC FI(SYSHELP) DA('SYS1.HELP' +                                */
   /*                     'XXX.HELP') SHR REUSE                         */
   /*                                                                   */
   /*********************************************************************/
   /*                                                                   */
   /* THE VS FORTRAN VERSION 2 EXECUTION-TIME LIBRARY IS:               */
   /*                                                                   */
   /*         VSF2.VSF2FORT                                             */
   /*                                                                   */
   /*********************************************************************/
   ALLOC FI(ISPLLIB) DA('VSF2.VSF2COMP' +
                        'VSF2.VSF2FORT' +
                        'ISP.V2R1M0.ISPLLIB' +
                        'ISR.V2R1M0.ISRLLIB' ) SHR REUSE
   /*********************************************************************/
   /*                                                                   */
   /* ALLOCATE SYSTEM INPUT AND OUTPUT FILES TO THE TERMINAL.           */
   /* THIS CAN BE CHANGED BY USING TSO COMMANDS ANY TIME BEFORE         */
   /* INVOKING THE PROGRAM.                                             */
   /*                                                                   */
   /*********************************************************************/
   ALLOC FI(FT05F001) DA(*)
   ALLOC FI(FT06F001) DA(*)
```

Figure 6. Sample ISPF CLIST

```
//IADUSER EXEC PGM=IKJEFT01,DYNAMNBR=64,REGION=1024K
//*
//* IF VS FORTRAN VERSION 2'S VSF2.VSF2FORT WAS INSTALLED IN A LIBRARY
//* WHICH IS INCLUDED IN LINKLIST (MEMBER LNKLSTxx OF SYS1.PARMLIB),
//* THE STEPLIB STATEMENT FOR VSF2.VSF2FORT MAY BE DELETED.
//*
//STEPLIB   DD DSNAME=VSF2.VSF2COMP,DISP=SHR
//          DD DSNAME=VSF2.VSF2FORT,DISP=SHR
//          DD DSNAME=ISP.V2R1M0.ISPLLIB,DISP=SHR
//          DD DSNAME=ISR.V2R1M0.ISRLLIB,DISP=SHR
//SYSEDIT   DD DSNAME=&&EDIT,UNIT=SYSDA,SPACE=(1688,(50,20))
//SYSEDIT2  DD DSNAME=&&EDIT2,UNIT=SYSDA,SPACE=(4096,(20,10))
//SYSIN     DD TERM=TS,SYSOUT=A
//SYSPRINT  DD TERM=TS,SYSOUT=A
//SYSTERM   DD TERM=TS,SYSOUT=A
//*
//* IF THE HELP MEMBER FOR VS FORTRAN VERSION 2 INTERACTIVE DEBUG
//* WAS NOT INSTALLED IN SYS1.HELP, CONCATENATE THE CORRECT DATA SET
//* NAME TO THE SYSHELP STATEMENT BELOW.
//*
//SYSHELP   DD DSNAME=SYS1.HELP,DISP=SHR
//SYSLBC    DD DSNAME=SYS1.BRODCAST,DISP=SHR
//*
//* IF ISPF IS NOT AVAILABLE,
//* OR IF YOU WANT TO USE A CLIST TO ALLOCATE THE ISPF
//* DATASETS, DELETE THE REMAINING STATEMENTS
//*
//ISPPLIB   DD DSNAME=VSF2.VSF2PLIB,DISP=SHR
//          DD DSNAME=ISP.V2R1M0.ISPPLIB,DISP=SHR
//          DD DSNAME=ISR.V2R1M0.ISRPLIB,DISP=SHR
//ISPMLIB   DD DSNAME=VSF2.VSF2MLIB,DISP=SHR
//          DD DSNAME=ISP.V2R1M0.ISPMLIB,DISP=SHR
//          DD DSNAME=ISR.V2R1M0.ISRMLIB,DISP=SHR
//ISPSLIB   DD DSNAME=ISP.V2R1M0.ISPSLIB,DISP=SHR
//          DD DSNAME=ISR.V2R1M0.ISRSLIB,DISP=SHR
//ISPTLIB   DD DSNAME=ISP.V2R1M0.ISPTLIB,DISP=SHR
//          DD DSNAME=ISR.V2R1M0.ISRTLIB,DISP=SHR
//SYSPROC   DD DSNAME=ISP.V2R1M0.ISPCLIB,DISP=SHR
//          DD DSNAME=ISR.V2R1M0.ISRCLIB,DISP=SHR
//          DD DSNAME=VSF2.VSF2CLIB,DISP=SHR
```

Figure 7. Example of a TSO LOGON Procedure for ISPF Users

Installing Interactive Debug (Non-ISPF Users Only)

Execute VS FORTRAN Version 2 Interactive Debug in line mode in *one* of the following ways:

- Include a STEPLIB DD statement for VSF2FORT (the load library containing the Interactive Debug load module) in your TSO logon procedure.
- Include VSF2FORT in the LNKLST concatenation.

If you installed the Interactive Debug HELP member in a library other than SYS1.HELP, you must concatenate that library's name to the SYSHELP DD statement in your TSO logon procedure.

Step 6: Verifying a Successful Installation

You can verify the success of your installation of VS FORTRAN Version 2 by running the sample programs. One sample program verifies the installation for the Compiler and Library, and the other verifies installation of VS FORTRAN Version 2 Interactive Debug.

Verifying Success for the Compiler and Library

To verify the success of the installation process, you may want to run a sample program. Below is sample JCL that executes the compile-link-go procedure, VSF2CLG, provided in SYS1.PROCLIB. It runs the sample program, AFBIVP, provided in SYS1.SAMPLIB.

```
//AFBIVP      JOB . . .
//FORTRAN     EXEC VSF2CLG
//FORT.SYSIN  DD DSN=SYS1.SAMPLIB(AFBIVP),DISP=SHR
```

Note: If you have changed data set name prefixes to install your product into private libraries, you must change the corresponding prefixes in the JCL and in the VSF2CLG procedure.

Figure 8. Sample Program to Verify Success of VS FORTRAN Version 2 Compiler and Library

Successful execution of the sample program causes the following message to be issued:

```
'SAMPLE PROGRAM COMPLETED SUCCESSFULLY'
```

Verifying Success for Interactive Debug

To verify the successful installation of VS FORTRAN Version 2 Interactive Debug, a FORTRAN program is provided on the distribution tape. This program computes the diameter, circumference, and area of a circle. The sample program is installed as member AFFIVP in SYS1.SAMPLIB, and can be edited and printed using normal TSO commands.

Compile the sample program using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default. Use your site's standard procedures for compiling FORTRAN programs.

You should receive the following messages from the compilation with no error or warning messages:

```
**CIRCLE** END OF COMPILATION 1 ******

**DIAM** END OF COMPILATION 2 ******

**CIRCUM** END OF COMPILATION 3 ******

**AREA** END OF COMPILATION 4 ******
```

After AFFIVP has been compiled and link-edited into a load module, execute the load module with VS FORTRAN Version 2 Interactive Debug. The program may be executed either under ISPF or in line mode. The following two sections tell how to do this.

ISPF (Interactive System Product Facility) Environment

If you use ISPF at your site, continue with the following steps. If you do not use ISPF, go to the section "Line Mode Environment" below.

Before invoking ISPF, make sure that all steps have been completed, as described under "Installing Interactive Debug (ISPF/PDF Users Only)" on page 21. Invoke ISPF in the standard manner for your site.

When the PRIMARY OPTION MENU appears, select option 4, FOREGROUND PROCESSING, which causes the FOREGROUND SELECTION MENU panel to be displayed. Find the line specifying VS FORTRAN Version 2 Interactive Debug, and enter the associated number. (In the example in Figure 4 on page 22, number 11 was used.) The FOREGROUND VS FORTRAN VERSION 2.1.1 INTERACTIVE DEBUG panel is displayed. Enter the data set on the correct line and specify AFFIVP as the member. Enter DEBUG on the first entry line below the line labeled EXECUTION-TIME OPTIONS.

The next panel displayed should be the Interactive Debug panel. You are now ready to issue Interactive Debug commands as described under "Expected Results" on page 29.

Line Mode Environment

If you are not using ISPF, set up the Interactive Debug verification program as follows.

Using an editor, build a CLIST to execute a FORTRAN program with Interactive Debug. (See Figure 9 on page 29.) The CLIST must include ALLOCATE statements for all data sets used by VS FORTRAN Version 2 Interactive Debug. The CLIST shown below has three parameters. The first is the member to be executed, which has to be compiled and link-edited to form an executable load module. The second is the data set containing that member (defaults to FORTRAN.LOAD). The third parameter is the execution-time option, which defaults to DEBUG in this CLIST.

To invoke the CLIST and debug program AFFIVP, enter:

```
clistname AFFIVP
```

where "clistname" is replaced by the name you used for the CLIST.

```
PROC 1 MEMBER DSN(FORTRAN.LOAD) OPTION(DEBUG)
CONTROL NOMSG NOFLUSH NOLIST NOSYMLIST NOCONLIST
IF &OPTION = DEBUG THEN DO
   FREE FI(AFFPRINT AFFON)
   ALLOC FI(AFFON) DA(&MEMBER..INCLUDE) SHR
   IF &LASTCC ¬= 0 THEN ALLOC FI(AFFON) DUMMY
   ALLOC FI(AFFPRINT) DA(&MEMBER..PRINT) SHR
   IF &LASTCC ¬= 0 THEN +
      ALLOC FI(AFFPRINT) DA(&MEMBER..PRINT) +
      NEW CATALOG SPACE(5 5) TRACKS
END
CALL '&SYSUID..&DSN.(&MEMBER)' '&OPTION'
SET RCODE = &LASTCC
FREE FI(AFFPRINT AFFON)
WRITE RETURN CODE: &RCODE
EXIT
END
```

Figure 9. Sample TSO CLIST to Invoke a VS FORTRAN Program

If your CLIST executes properly, you will receive the VS FORTRAN Interactive Debug prompt preceded by the product name and copyright information:

```
WHERE: CIRCLE.7
FORTIAD
```

You are now ready to issue Interactive Debug commands, as described under "Expected Results" below.

Expected Results

Input and output for a set of Interactive Debug commands are shown in Figure 10 on page 30. You may enter any other commands you want in order to further verify correct installation. In the figure, all lines beginning with an asterisk (*) are lines that were entered on the terminal. However, when entering the commands, do not type the leading asterisk. This log was obtained during execution under ISPF.

```
WHERE: CIRCLE.7
* listsubs
PROGRAM UNIT  COMPILER   OPT   TIMING
   CIRCLE     VSF 2.1.1   0      OFF
   DIAM       VSF 2.1.1   0      OFF
   CIRCUM     VSF 2.1.1   2      OFF
   AREA       VSF 2.1.1   3      OFF
* describe (data pi)
DATA:         REAL*4
   RANK = 1, SIZE = 3 ELEMENTS
   DIM 1:     EXTENT = 3,  LBOUND = 1,  UBOUND = 3
PI:           REAL*8
* at diam.entry (step)
* go
FT06F001  ENTER THE VALUE OF THE CIRCLE RADIUS (xxx.xx):
FT05F001 INPUT: PRECEDE INPUT WITH % OR ENTER IAD COMMAND
* %352.67
AT: DIAM.ENTRY
NEXT: DIAM.3
* set diam.value = 0.0
* when test value
* go
WHEN: "TEST" SATISFIED;
CURRENTLY AT DIAM.4
* offwn test
* at circle.42 (list '= READY FOR TERMINATION ='%go)
* listbrks
CURRENT BREAKPOINTS:
  CIRCLE.42
  DIAM.ENTRY
CURRENT WHEN CONDITIONS:
  TEST OFF  DIAM.VALUE
CURRENT HALT STATUS: OFF
* go
FT06F001  THE DIAMETER OF THE CIRCLE IS 705.34
FT06F001  THE CIRCUMFERENCE OF THE CIRCLE IS 2215.89
FT06F001  THE AREA OF THE CIRCLE IS 390738.94
AT: CIRCLE.42
= READY FOR TERMINATION =
PROGRAM HAS TERMINATED; RC = 0
* quit
```

Figure 10. TSO Interactive Debug Input/Output

Step 7: Executing the Job to Accept VS FORTRAN Version 2

When you are satisfied that VS FORTRAN Version 2 is operating correctly, use this job to store the product in your system's distribution libraries (DLIBs). This job is named ILXyACPT (if you are installing the entire VS FORTRAN Version 2 product), or AFByACPT (if you are installing the library only), where "y" represents the version of SMP you are using—either SMP4 ("4") or SMP/E ("E").

If you are installing in distribution libraries that have not previously contained VS FORTRAN Version 2, you will receive message **HMA2471** (SMP4) or **GIM2471** (SMP/E) for modules AFBUATBL and ILX0OPTS. These messages may be **ignored.**

When using **SMP4** to accept VS FORTRAN Version 2 (the ILX4ACPT job for accepting the whole product; the AFB4ACPT job for accepting the Library only), do the following:

- The SMPCNTL ACCEPT statement that follows the DD statement defines the FMIDs to be accepted. Refer to the program directory for the list of FMIDs to be accepted. Make sure that the list in the ILX4ACPT (or AFB4ACPT) job matches the list of FMIDs you received and applied with the ILX4INST (or AFB4INST) job.

Installation of VS FORTRAN Version 2 is now complete.

When using **SMP/E** to accept VS FORTRAN Version 2 (the ILXEACPT job for accepting the whole product; the AFBEACPT job for accepting the Library only), do the following:

- SMPCNTL statements that follow the DD statement set the distribution zone and define the FMIDs to be accepted. Refer to the program directory for the list of FMIDs to be accepted. Make sure that the list in the ILXEACPT (or AFBEACPT) job matches the list of FMIDs you received and applied with the ILXEINST (or AFBEINST) job.

Installation of VS FORTRAN Version 2 is now complete.

Chapter 3. Customization under MVS

The following features, which can be customized under MVS, are discussed in this chapter:

- Alternative mathematical library subroutines
- Cataloged procedures
- Reentrant I/O library modules
- Compiler options
- I/O unit numbers
- Execution-time options
- Extended error handling facility
- Execution-time loading of library modules

If customizing VS FORTRAN Version 2 under MVS/XA, Assembler H Version 2 is required.

Alternative Mathematical Library Subroutines

The alternative mathematical library contains the VS FORTRAN Version 1 standard mathematical routines. This library allows users to have access to the routines that were available to them as part of the VS FORTRAN Version 1 product. However, the routines that were alternative in the VS FORTRAN Version 1 product are not available in VS FORTRAN Version 2.

The alternative mathematical library subroutines are link-edited in VSF2.VSF2MATH by the installation process.

To make the alternative mathematical routines available to all users, change the cataloged procedures, such as VSF2CL and VSF2CLG, provided in SYS1.PROCLIB, to concatenate VSF2.VSF2MATH ahead of VSF2.VSF2FORT in the link-edit step SYSLIB DD statement. For example, use these statements in load mode:

```
//SYSLIB DD DSN=VSF2.VSF2MATH,DISP=SHR
//        DD DSN=VSF2.VSF2FORT,DISP=SHR
```

or use these statements in link mode:

```
//SYSLIB DD DSN=VSF2.VSF2MATH,DISP=SHR
//        DD DSN=VSF2.VSF2LINK,DISP=SHR
//        DD DSN=VSF2.VSF2FORT,DISP=SHR
```

Cataloged Procedures

The installation job places cataloged procedures in the procedure library, SYS1.PROCLIB. You may want to edit the supplied procedures to fit your system's requirements. For additional information on writing and processing cataloged procedures under MVS, see *VS FORTRAN Version 2: Programming Guide.*

The cataloged procedures provided on the VS FORTRAN Version 2 distribution tape are listed here:

VSF2C Compiles a VS FORTRAN Version 2 source program

VSF2CL Compiles a VS FORTRAN Version 2 source program and link-edits the object module into a load module

VSF2CG Compiles a VS FORTRAN Version 2 source program and loads and executes the object module

VSF2CLG Compiles a VS FORTRAN Version 2 source program, link-edits the object module into a load module, and executes the load module

VSF2L Link-edits a VS FORTRAN Version 2 object module into a load module

VSF2LG Link-edits a VS FORTRAN Version 2 object module into a load module and executes the load module

VSF2G Loads and executes a VS FORTRAN Version 2 object module

VFT2RCL Compiles a VS FORTRAN Version 2 reentrant source program and link-edits the object module into a load module

VFT2RCLG Compiles a VS FORTRAN Version 2 reentrant source program, link-edits the object module into a load module, and executes the load module

VFT2RLG Link-edits a VS FORTRAN Version 2 reentrant object module into a load module and executes it

A typical example of a cataloged procedure is VSF2CLG, shown in Figure 11 on page 35, which compiles, link-edits, and executes a VS FORTRAN Version 2 program in load mode.

```
//VSF2CLG PROC FVPGM=FORTVS2,FVREGN=1400K,FVPDECK=NODECK,
//             FVPOLST=NOLIST,FVPOPT=0,FVTERM='SYSOUT=A',GOREGN=100K,
//             FVLNSPC='3200,(25,6)',
//             GOF5DD='DDNAME=SYSIN',GOF6DD='SYSOUT=A',
//             GOF7DD='SYSOUT=B'
//*
//*            PARAMETER  DEFAULT-VALUE     USAGE
//*
//*              FVPGM    FORTVS2           COMPILER NAME
//*              FVREGN   1400K             FORT-STEP REGION
//*              FVPDECK  NODECK            COMPILER DECK OPTION
//*              FVPOLST  NOLIST            COMPILER LIST OPTION
//*              FVPOPT   0                 COMPILER OPTIMIZATION
//*              FVTERM   SYSOUT=A          FORT.SYSTERM OPERAND
//*              FVLNSPC  3200,(25,6)       FORT.SYSLIN SPACE
//*              GOREGN   100K              GO-STEP REGION
//*              GOF5DD   DDNAME=SYSIN      GO.FT05F001 OPERAND
//*              GOF6DD   SYSOUT=A          GO.FT06F001 OPERAND
//*              GOF7DD   SYSOUT=B          GO.FT07F001 OPERAND
//*
//FORT   EXEC  PGM=&FVPGM,REGION=&FVREGN,
//             PARM='&FVPDECK,&FVPOLST,OPT(&FVPOPT)'
//STEPLIB      DD DSN=VSF2.VSF2COMP,DISP=SHR
//SYSPRINT     DD SYSOUT=A,
//SYSTERM      DD &FVTERM
//SYSPUNCH     DD SYSOUT=B,DCB=BLKSIZE=3440
//SYSLIN       DD DSN=&&LOADSET,DISP=(NEW,PASS),UNIT=SYSDA,
//             SPACE=(&FVLNSPC),DCB=BLKSIZE=3200
//LKED   EXEC  PGM=IEWL,REGION=200K,COND=(4,LT),
//             PARM='LET,LIST,XREF'
//SYSPRINT     DD SYSOUT=A
//SYSLIB       DD DSN=VSF2.VSF2FORT,DISP=SHR
//SYSUT1       DD UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSLMOD      DD DSN=&&GOSET(MAIN),DISP=(,PASS),UNIT=SYSDA,
//             SPACE=(TRK,(10,10,1),RLSE)
//SYSLIN       DD DSN=&&LOADSET,DISP=(OLD,PASS)
//             DD DDNAME=SYSIN
//GO     EXEC  PGM=*.LKED.SYSLMOD,REGION=&GOREGN,COND=(4,LT)
//STEPLIB      DD DSN=VSF2.VSF2FORT,DISP=SHR
//FT05F001     DD &GOF5DD
//FT06F001     DD &GOF6DD
//FT07F001     DD &GOF7DD
```

Figure 11. Example of a Cataloged Procedure (VSF2CLG)

The first job control statement in each cataloged procedure is the PROC statement. The PROC statement assigns default values to symbolic parameters.

Symbolic parameters make it easier for you to modify a cataloged procedure when it is called. You may assign values to symbolic parameters when a cataloged procedure is called, or you may accept the default value assigned by the PROC statement.

The following are some of the changes you can make to VSF2CLG to customize it for use at your site.

- You may want to increase the FVREGN size to accommodate the needs of your site's typical compilation.

- If you changed the names of any of the libraries into which you installed the VS FORTRAN Version 2 data sets, you must change the corresponding data set name (DSN) parameters on the STEPLIB and SYSLIB DD statements to specify your new library names.

- The SYSLIB DD statement defines the library that contains the FORTRAN library routines. For the LKED step, you can adjust the SYSLIB DD statement for using link mode or load mode. Link mode and load mode are explained under "Execution-Time Loading of Library Modules" on page 46.

  The cataloged procedures supplied by IBM are set up for using load mode, as follows:

  ```
  //SYSLIB DD DSN=VSF2.VSF2FORT,DISP=SHR
  ```

  To set up your cataloged procedure to use link mode, change the SYSLIB DD statement as follows:

  ```
  //SYSLIB DD DSN=VSF2.VSF2LINK,DISP=SHR
  //        DD DSN=VSF2.VSF2FORT,DISP=SHR
  ```

- If you want to make the alternative mathematical library routines available, see the section "Alternative Mathematical Library Subroutines" on page 33.

- For the execution, or GO step, a STEPLIB DD statement defines the execution-time modules needed for proper execution of the program using load mode. The statement looks like this:

  ```
  //STEPLIB DD DSN=VSF2.VSF2FORT,DISP=SHR
  ```

  If you are modifying the cataloged procedure to use link mode, delete this STEPLIB DD statement.

You can make similar changes to the other cataloged procedures as appropriate. For more detailed information on all of the cataloged procedures provided on the distribution tape, see *VS FORTRAN Version 2: Programming Guide*.

Reentrant I/O Library Modules—Transitional Support

The facility that was available in VS FORTRAN Version 1 before Release 4.0 for loading the reentrant I/O library modules has been replaced by more extensive loading of library modules during execution. In VS FORTRAN Version 2 an AFBVRENT (previously called IFYVRENT) module is installed in VSF2.VSF2FORT. This module contains versions of the previous modules that are compatible with load modules created with VS FORTRAN Version 1 Releases 2.0, 3.0, or 3.1. However, these modules contain no new functions.

If your load module contains any code compiled with VS FORTRAN Version 2 or VS FORTRAN Version 1 Release 4.0 or later, or contains any VS FORTRAN Version 2 or Version 1 Release 4.0 or later library modules, then all library modules linked into that load module must be at the VS FORTRAN Version 2 or VS FORTRAN Version 1 Release 4.0 or later level. The former

IFYVRENT mechanism, which was used in VS FORTRAN Version 1 before Release 4.0, will then no longer be used for that load module.

Load modules created with Release 1.0 or Release 1.1 of VS FORTRAN Version 1 are not compatible with the module AFBVRENT from VS FORTRAN Version 2. If you have such load modules, they must be relink-edited using the VS FORTRAN Version 2 library.

To make the reentrant compatibility module AFBVRENT available to all users at execution, do one of the following:

- In any JCL that executes VS FORTRAN Version 1 programs that use the Reentrant I/O Library Facility, add a STEPLIB DD statement for VSF2.VSF2FORT for loading AFBVRENT (alias IFYVRENT).
- You may choose to put the module AFBVRENT and its alias IFYVRENT in the pageable link pack area by moving it to SYS1.LPALIB. In an MVS/XA system, this module will reside below the 16-megabyte virtual storage line.

Changing Compiler Option Defaults

Note: This section is not applicable to those who have installed VS FORTRAN Version 2 Library only, without the compiler. Go on to the section "Changing Execution-Time Option Defaults" on page 42.

IBM provides, on the VS FORTRAN Version 2 distribution tape, a module ILX0OPTS that contains a set of default values for compiler options. You can code an SMP USERMOD to change those defaults to better suit the needs of your site.

The compiler option defaults provided in ILX0OPTS are described under "VSF2COM: For Changing Compiler Option Defaults" on page 96. To change the compiler default options, code a VSF2COM macro instruction with the desired new default options, as described in that section.

Figure 12 on page 38 and Figure 13 on page 39 are examples of jobs that receive, apply, and accept a USERMOD and replace module ILX0OPTS with a new copy that contains the desired options.

```
//COMOPT4C JOB...
//OPTION1  EXEC PROC=VSFACC4C                          (See Note 1)
//SMPPTFIN   DD  *
++ USERMOD(USR0001).
++ VER(Z038) FMID(put VS FORTRAN Version 2 compiler common FMID here).
++ SRC(ILXOOPTS) DISTLIB(ILXCCS).
       VSF2COM SYSTEM=OS/VS[,option,...]               (See Notes 2 and 3)
       END
/*
//SMPCNTL   DD *
  RECEIVE S(USR0001).
  APPLY   S(USR0001) ASSEM DIS(WRITE).                 (See Note 4)
/*
//OPTION1A  EXEC PROC=VSFACC4C                         (See Note 1)
//SMPCNTL   DD *
  ACCEPT  S(USR0001) USERMOD ASSEM DIS(WRITE).
/*
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. Specify new desired options. The SYSTEM option must be specified as shown.
3. If programmers at your site will be compiling with VS FORTRAN Version 2 in a batch environment and will not be using a SYSTERM data set, specify NOTRMFLG and NOTERMINAL in the VSF2COM macro instruction to avoid messages about having no terminal online.
4. When you change the compiler options, the space used for the data set VSF2.VSF2COMP is doubled. You may want to use the COMPRESS option on the APPLY statement to minimize the storage used for this data set.

Figure 12. Example of JCL for an SMP4 USERMOD to Change Compiler Option Defaults

```
//COMOPTEC JOB
//OPTION1  EXEC PROC=VSFACCEC                          (See Note 1)
//SMPPTFIN  DD  *
++ USERMOD(USR0001).
++ VER(Z038) FMID(put VS FORTRAN Version 2 compiler common FMID here).
++ SRC(ILX0OPTS) DISTLIB(ILXCCS).
       VSF2COM SYSTEM=OS/VS[,option,...]               (See Notes 2 and 3)
       END
/*
//SMPCNTL   DD *
  SET BDY(GLOBAL).
  RECEIVE LIST SYSMODS S(USR0001).
  SET BDY(TVSF2).
  APPLY S(USR0001) USERMOD ASSEM.                     (See Note 4)
/*
//OPTION1A EXEC PROC=VSFACCEC                          (See Note 1)
//SMPCNTL  DD *
  SET BDY(DVSF2).
  ACCEPT S(USR0001) USERMOD ASSEM.
/*
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. Specify new desired options. The SYSTEM option must be specified as shown.
3. If programmers at your site will be compiling with VS FORTRAN Version 2 in a batch environment and will not be using a SYSTERM data set, specify NOTRMFLG and NOTERMINAL in the VSF2COM macro instruction to avoid messages about having no terminal online.
4. When you change the compiler options, the space used for the data set VSF2.VSF2COMP is doubled. You may want to use the COMPRESS option on the APPLY statement to minimize the storage used for this data set.

Figure 13. Example of JCL for an SMP/E USERMOD to Change Compiler Option Defaults

Changing Library I/O Unit Number Defaults

IBM provides, on the product distribution tape, a table AFBUATBL that contains a set of default values for I/O unit numbers. You can code an SMP USERMOD to change those defaults to better suit the needs of your site.

The I/O unit numbers and their default values provided by IBM in AFBUATBL are described under "VSF2LIB: For Changing I/O Unit Number Defaults" on page 102. To change the I/O unit number defaults, code a VSF2LIB macro instruction with the desired new default values, as described in that section.

Figure 14 and Figure 15 on page 41 show examples of jobs that receive, apply, and accept a USERMOD and replace AFBUATBL with a new copy that contains the desired unit number defaults.

```
//IOUNIT4   JOB ...
//OPTION2   EXEC PROC=VSFACC4C                        (See Notes 1 and 4)
//SMPPTFIN  DD   *
++ USERMOD(USR0002).
++ VER(Z038) FMID(put VS FORTRAN Version 2 library common FMID here).
++ SRC(AFBUATBL) DISTLIB(AFBLBS).
      VSF2LIB [option,...]                          (See Notes 2 and 3)
      END
/*
//SMPCNTL   DD *
  RECEIVE S(USR0002).
  APPLY   S(USR0002) ASSEM DIS(WRITE).
/*
//OPTION2A  EXEC PROC=VSFACC4C                        (See Notes 1 and 4)
//SMPCNTL   DD *
  ACCEPT  S(USR0002) USERMOD ASSEM DIS(WRITE).
/*

   Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.

2. Specify new desired options.

3. The default number of units in VSF2LIB is 99 (UNTABLE option).  You may
   want to specify a smaller number of units.

4. If you have installed VS FORTRAN Version 2 Library product only, using
   SMP4, use EXEC VSFACC4L instead.
```

Figure 14. Example of JCL for an SMP4 USERMOD for Changing I/O Unit Number Defaults

```
//IOUNITE   JOB ...
//OPTION2   EXEC PROC=VSFACCEC                          (See Notes 1 and 4)
//SMPPTFIN  DD   *
++ USERMOD(USR0002).
++ VER(Z038) FMID(put VS FORTRAN Version 2 library common FMID here).
++ SRC(AFBUATBL) DISTLIB(AFBLBS).
       VSF2LIB [option,...]                             (See Notes 2 and 3)
       END
/*
//SMPCNTL    DD *
  SET BDY(GLOBAL).
  RECEIVE LIST SYSMODS S(USR0002).
  SET BDY(TVSF2).
  APPLY S(USR0002) USERMOD ASSEM.
/*
//OPTION2A  EXEC PROC=VSFACCEC                          (See Notes 1 and 4)
//SMPCNTL   DD *
  SET BDY(DVSF2).
  ACCEPT S(USR0002) USERMOD ASSEM.
/*
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. Specify new desired options.
3. The default number of units in VSF2LIB is 99 (UNTABLE option). You may want to specify a smaller number of units.
4. If you have installed VS FORTRAN Version 2 Library product only, using SMP/E, use EXEC VSFACCEL instead.

Figure 15. Example of JCL for an SMP/E USERMOD to Change I/O Unit Number Defaults

Changing Execution-Time Option Defaults

IBM provides, on the VS FORTRAN Version 2 distribution tape, a table AFBVGPRM that contains a set of default values for execution-time options. You can, if you want, code an SMP USERMOD to change those defaults to better suit the needs of your site.

The execution-time option defaults provided in AFBVGPRM are described under "VSF2PARM: For Changing Execution-Time Option Defaults" on page 104. To change these defaults, code a VSF2PARM macro instruction with the new default options you choose, as described in that section.

Figure 16 and Figure 17 on page 43 show examples of jobs that receive, apply, and accept a USERMOD and replace AFBVGPRM with a new copy that contains the desired options.

```
//EXOPT4   JOB...
//OPTION1  EXEC PROC=VSFACC4C                          (See Notes 1 and 4)
//SMPPTFIN   DD  *
++ USERMOD(USR0003).
++ VER(Z038) FMID(put VS FORTRAN Version 2 library common FMID here).
++ SRC(AFBVGPRM) DISTLIB(AFBLBS).
       VSF2PARM [option,...]SCOPE=GLOBAL               (See Notes 2 and 3)
       END
/*
//SMPCNTL   DD *
  RECEIVE S(USR0003).
  APPLY   S(USR0003) ASSEM DIS(WRITE).
/*
//OPTION1A  EXEC PROC=VSFACC4C                         (See Notes 1 and 4)
//SMPCNTL   DD *
  ACCEPT  S(USR0003) USERMOD ASSEM DIS(WRITE).
/*

   Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.

2. Specify new desired options.

3. You must specify SCOPE=GLOBAL.

4. If you have installed VS FORTRAN Version 2 Library product only, using
   SMP4, use EXEC VSFACC4L instead.
```

Figure 16. Example of JCL for an SMP4 USERMOD to Change Execution-Time Option Defaults

```
//EXOPTE   JOB...
//OPTION1  EXEC PROC=VSFACCEC                          (See Notes 1 and 4)
//SMPPTFIN  DD  *
++ USERMOD(USR0003).
++ VER(Z038) FMID(put VS FORTRAN Version 2 library common FMID here).
++ SRC(AFBVGPRM) DISTLIB(AFBLBS).
      VSF2PARM [option,...]SCOPE=GLOBAL              (See Notes 2 and 3)
      END
/*
//SMPCNTL   DD *
  SET BDY(GLOBAL).
  RECEIVE LIST SYSMODS S(USR0003).
  SET BDY(TVSF2).
  APPLY S(USR0003) USERMOD ASSEM.
/*
//OPTION1A EXEC PROC=VSFACCEC                          (See Notes 1 and 4)
//SMPCNTL  DD *
  SET BDY(DVSF2).
  ACCEPT S(USR0003) USERMOD ASSEM.
/*
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. Specify new desired options.
3. You must specify SCOPE=GLOBAL.
4. If you have installed VS FORTRAN Version 2 Library product only, using SMP/E, use EXEC VSFACCEL instead.

Figure 17. Example of JCL for an SMP/E USERMOD to Change Execution-Time Option Defaults

Extended Error Handling Facility

The **error option table** is a VS FORTRAN Version 2 library module that specifies what actions are taken when an error occurs during execution of a FORTRAN program. For each execution-time error defined by VS FORTRAN Version 2, the table specifies:

- The number of times the error is allowed to occur before the user's program terminates
- The maximum number of times the message may be printed
- Whether or not the traceback map is to be printed with the message
- Whether or not a user-written error exit routine is called

The error numbers and values in the standard table you have received as part of the VS FORTRAN Version 2 product are documented in *VS FORTRAN Version 2: Language and Library Reference.*

You can customize this table in two ways:

1. Alter the IBM-supplied values in the standard error table. These are the values for the error conditions detected by VS FORTRAN Version 2 (numbers between 112 and 301).
2. Extend the table to include your own error numbers and associated actions. These would be error conditions that an individual program (rather than the FORTRAN product) would recognize and would deal with by calling the supplied subroutine ERRMON to take the action specified in your table extension. These user-defined numbers can be in the range 302 through 899.

The customization changes/extensions you make here apply to the permanent copy of the error option table in the VS FORTRAN Version 2 Library. (Each individual FORTRAN program can also make execution-time changes to its copy of the table by calling the supplied subroutines ERRSET, ERRSAV, and ERRSTR. See *VS FORTRAN Version 2: Language and Library Reference* and *VS FORTRAN Version 2: Programming Guide* if you want more information on this programming feature.)

Changing or Adding Error Option Table Entries

The default values for the error options are established in table AFBUOPT. You can code an SMP USERMOD to modify the table to better suit the needs of your site.

The error options and their default values provided by IBM in AFBUOPT are described under "VSF2UOPT: For Altering/Extending the Error Option Table" on page 108. To change the error option defaults, or to add new message numbers, code one or more VSF2UOPT macro instructions with the desired new default values, as described in that section.

Figure 18 and Figure 19 on page 46 show examples of jobs that receive, apply, and accept a USERMOD and replace AFBUOPT with a new copy that contains the desired error option defaults.

```
//ERROPT4   JOB ...
//OPTION2   EXEC PROC=VSFACC4C                          (See Notes 1 and 4)
//SMPPTFIN  DD   *
++ USERMOD(USR0004).
++ VER(Z038) FMID(put VS FORTRAN Version 2 library common FMID here).
++ SRC(AFBUOPT) DISTLIB(AFBLBS).
      VSF2UOPT [ADDNTRY=n]                              (See Notes 2 and 3)
      END
/*
//SMPCNTL   DD *
  RECEIVE S(USR0004).
  APPLY   S(USR0004) ASSEM DIS(WRITE).
/*
//OPTION2A  EXEC PROC=VSFACC4C                          (See Notes 1 and 4)
//SMPCNTL   DD *
  ACCEPT  S(USR0004) USERMOD ASSEM DIS(WRITE).
/*

   Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.

2. Specify new desired options.

3. There are two forms of the VSF2UOPT macro instruction.  Only the first
   form is shown here.

4. If you have installed VS FORTRAN Version 2 Library product only, using
   SMP4, use EXEC VSFACC4L instead.
```

Figure 18. Example of JCL for an SMP4 USERMOD for Changing Error Option Defaults

```
//ERROPTE   JOB ...
//OPTION2   EXEC PROC=VSFACCEC                          (See Notes 1 and 4)
//SMPPTFIN  DD   *
++ USERMOD(USR0004).
++ VER(Z038) FMID(put VS FORTRAN Version 2 library common FMID here).
++ SRC(AFBUOPT) DISTLIB(AFBLBS).
       VSF2UOPT [ADDNTRY=n]                             (See Notes 2 and 3)
       END
/*
//SMPCNTL    DD *
  SET BDY(GLOBAL).
  RECEIVE LIST SYSMODS S(USR0004).
  SET BDY(TVSF2).
  APPLY S(USR0004) USERMOD ASSEM.
/*
//OPTION2A  EXEC PROC=VSFACCEC                          (See Notes 1 and 4)
//SMPCNTL   DD *
  SET BDY(DVSF2).
  ACCEPT S(USR0004) USERMOD ASSEM.
/*
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. Specify new desired options.
3. There are two forms of the VSF2UOPT macro instruction. Only the first form is shown here.
4. If you have installed VS FORTRAN Version 2 Library product only, using SMP/E, use EXEC VSFACCEL instead.

Figure 19. Example of JCL for an SMP/E USERMOD to Change Error Option Defaults

Execution-Time Loading of Library Modules

When programmers link-edit a program, they can choose to have all execution-time library modules (other than the mathematical routines) either link-edited into the load module with compiler-generated code (link mode), or to have many of them loaded dynamically at execution time (load mode). Execution-time loading has several advantages. It reduces auxiliary storage requirements for load modules, speeds link-editing, and, in an MVS/XA environment, allows some library routines to be placed in the extended link pack area.

Composite Modules

If programmers choose to have all the library modules link-edited with their programs (link mode), no loading of library modules is required at execution time. If they choose execution-time loading (load mode), each module is loaded the first time it is used, unless it has been previously loaded. Because execution-time performance suffers if a large number of library modules are

individually loaded, the modules to be loaded at execution time are combined into composite modules.

Four composite modules are installed in VSF2FORT:

AFBVLBCM
: contains nonreentrant library modules, including the library common work area and various system services routines.

AFBVRENA
: contains reentrant library modules that can reside above 16 megabytes in an MVS/XA system. Many library modules that previously resided in the VS FORTRAN Version 1 module IFYVRENT (below 16 megabytes) may now be placed in AFBVRENA for virtual storage constraint relief. In non-XA systems, this module is not used.

AFBVRENB
: contains reentrant library modules that must reside below 16 megabytes in an MVS/XA system. In non-XA systems, this module is not used.

AFBVRENC
: includes all the loadable reentrant modules for a non-XA system. In an MVS/XA system, this module is not used.

As part of its initialization procedure in load mode, the execution-time library loads the composite modules listed above. The only modules that need to be loaded separately after initialization are those that are not contained in the composite modules.

At any time after installation, you may add or delete library modules from the composite load modules to further tune your system. For example, if keyed access and direct access are not normally used at your site, you may choose not to place the modules that perform these functions in the composite load modules. This reduces their size. The execution-time library will then have to load the direct access and keyed access I/O modules individually when they are needed. (These modules may reside in the link pack area so they don't need to be brought into your region, or they may be brought into your region from the library containing them.)

Selecting Load Mode or Link Mode

After installing the VS FORTRAN Version 2 Library, you may update your site's cataloged procedures (for example, VSF2CLG for compile, link-edit, and go), to specify the libraries needed for use in load mode or link mode. All cataloged procedures provided with the product are set up for load mode. The methods for specifying libraries in load mode or link mode are described below.

Specifying Libraries in Load Mode

For operation in load mode, specify only VSF2.VSF2FORT in the DD statement for SYSLIB in the linkage editor step:

```
//SYSLIB   DD   DSN=VSF2.VSF2FORT,DISP=SHR
```

In order for a program that has been link-edited in load mode to be executed, VSF2FORT must be made available for the execution step by performing one of the following:

- Concatenate VSF2.VSF2FORT to SYS1.LINKLIB in the system link list so that VSF2.VSF2FORT is searched as part of the link library without JOBLIB or STEPLIB DD statements.

  Place the reentrant composite modules AFBVRENA (MVS/XA only), AFBVRENB (MVS/XA only), and AFBVRENC (non-XA only), as well as selected individual reentrant modules, in the link pack area (SYS1.LPALIB).

  Use the copy of the modules in the link pack area without searching VSF2.VSF2FORT. (If maintenance affects any modules in the link pack area, copy the updated copies of the modules into the link pack area from VSF2.VSF2FORT.)

- If it is preferable at your site to use a step library or job library in addition to loading reentrant modules from the link pack area, you must do the following:

  1. After modifying the composite modules, place the reentrant composite modules AFBVRENA (MVS/XA only), AFBVRENB (MVS/XA only), and AFBVRENC (non-XA only) in the link pack area (library SYS1.LPALIB).

  2. Optionally, place any reentrant modules that are *not* in a composite module into the link pack area.

  3. Create a new library that contains all modules from VSF2.VSF2FORT *minus* the modules (either composite modules or individual modules) that have been placed in the link pack area. Make this library available as either a step library or as a job library for the execution of the VS FORTRAN Version 2 program.

  If maintenance affects any of the modules in the link pack area or your new library, copy the updated copies of the modules from VSF2.VSF2FORT.

- At execution time, the application programmer can place the following JOBLIB DD statement in the job control language for the job that executes the VS FORTRAN Version 2 program:

  ```
  //JOBLIB   DD   DSN=VSF2.VSF2FORT,DISP=SHR
  ```

  or place the following STEPLIB DD statement in the job control language for the step that executes the VS FORTRAN Version 2 program:

  ```
  //STEPLIB  DD   DSN=VSF2.VSF2FORT,DISP=SHR
  ```

This technique does not let you use reentrant modules that are in the link pack area, because step libraries and job libraries are searched before the link pack area. (Refer to *OS/VS2 MVS Supervisor Services and Macro Instructions*, GC28-1114, or *MVS/Extended Architecture Supervisor Services and Macro Instructions*, GC28-1154, in the discussion of program management.)

Specifying Libraries in Link Mode

For operation in link mode, concatenate VSF2LINK ahead of VSF2FORT for use by the linkage editor when it includes VS FORTRAN Version 2 library modules. Specify both VSF2LINK and VSF2FORT in the DD statement for SYSLIB in the linkage editor step:

```
//SYSLIB   DD   DSN=VSF2.VSF2LINK,DISP=SHR
//         DD   DSN=VSF2.VSF2FORT,DISP=SHR
```

A program link-edited in link mode does not require specification of any VS FORTRAN Version 2 libraries at execution time. However, there are two exceptions:

1. If your program is reentrant, you must specify the library containing the load module with the nonshareable parts of the reentrant program.
2. If you are using VS FORTRAN Version 2 Interactive Debug, you must specify VSF2FORT.

Deciding What to Include in Composite Modules

You may update the composite modules AFBVLBCM, AFBVRENA (for MVS/XA only), AFBVRENB (for MVS/XA only), and AFBVRENC (non-XA only) to include only the library routines commonly used at your installation. Use the following guidelines to decide whether to include a module in the composite module:

- Because AFBVLBCM contains the nonreentrant modules, it must be loaded into your region for each execution of a VS FORTRAN Version 2 program. Including all possible nonreentrant modules may require the region size to be larger than would otherwise be necessary.
- If AFBVRENA, AFBVRENB, and AFBVRENC are not in the LPA, they must be loaded into your region. Including all possible reentrant modules in them may require the region size to be larger than would otherwise be necessary.
- If AFBVRENA, AFBVRENB, and AFBVRENC are in the LPA, including a large number of the reentrant modules in these composite modules has no effect upon the region size. However, the larger AFBVRENA, AFBVRENB, and AFBVRENC do require additional virtual storage in the LPA.

Each library module not in the applicable composite module is loaded from the VSF2FORT library when the module is first referenced at execution-time.

However, these individual modules could be placed in the link pack area under their own module names, and then loaded when needed.

Building the Composite Modules

The following section contains tables that list the library modules you can include in the various composite modules. The "Size" column lists approximate module sizes, in hexadecimal. The "Default Set" column indicates which modules are placed into the composite modules during installation. Except for those modules that must be in the composite modules, you can subsequently add or delete modules in this set to match the needs of your site. If a module performs a function used frequently at your site, you should consider including it in your composite module even if you are trying to limit the size of the composite module.

Following each list of modules is information on building the composite modules.

Composite Module AFBVLBCM

Figure 20 on page 51 and Figure 21 on page 52 list the modules, both required and optional, that can be included in composite module AFBVLBCM.

Module	Approx. Size	Default Set	Function
AFBVBLN$	34	X	Internal linkage routine
AFBVCOM$	34	X	Internal linkage routine
AFBVCOM2	FF4	X	Initialization/termination
AFBVCNO$	34	X	Internal linkage routine
AFBVCNI$	34	X	Internal linkage routine
AFBVCVT$	30C	X	Internal linkage routine
AFBVDEB$	34	X	Internal linkage routine
AFBVDIO$	3A	X	Internal linkage routine
AFBVEMG$	78	X	Internal linkage routine
AFBVERE$	34	X	Internal linkage routine
AFBVERS$	60	X	Internal linkage routine
AFBVFNTH	89E	X	Program interrupt handler
AFBVGPRM	C9	X	Default execution-time options
AFBVIAD$	34	X	Internal linkage routine
AFBVINI$	34	X	Internal linkage routine
AFBVIIO$	3A	X	Internal linkage routine
AFBVKIO$	3A	X	Internal linkage routine
AFBVLBC0	14B0	X	Library common work area
AFBVLOAD	178	X	Loader
AFBVLOC$	34	X	Internal linkage routine
AFBVMIN$	34	X	Internal linkage routine
AFBVMMA$	34	X	Internal linkage routine
AFBVMPR$	34	X	Internal linkage routine
AFBVPARM	714	X	Execution time options processor
AFBVPOS$	34	X	Internal linkage routine
AFBVSPIE	154	X	Interrupt interceptor
AFBVSTA$	10C	X	Internal linkage routine
AFBVTRC$	34	X	Internal linkage routine
AFBVVIO$	88	X	Internal linkage routine
AFBUATBL	648	X	Unit assignment table
AFBUOPT	60C	X	Error option table
Total	4DA8	31	

Figure 20. Required Modules for AFBVLBCM

Module	Note	Approx. Size	Default Set	Function
AFBDIOCP		1B0		Define file (LANGLVL 66)
AFBDSPAP	2	51C		Dimension calculator
AFBIBCOP	1	8D8		Pre-VS FORTRAN interface
AFBLDFIP	2	450		List-directed I/O
AFBNAMEP	2	39C		Namelist I/O
AFBSDUMQ		21CE		SDUMP subroutine
AFBTFORP		118		Debugger interface
AFBVASYP		B28		Asynchronous I/O
AFBVBALG		5F4		Boundary alignment routine
AFBVDBUP		11B6		Debugging packet
AFBVDUMQ		6CC		DUMP/PDUMP subroutine
AFBVINTH		4E8		Vector program interrupt handler
AFBVIONP		1018		Namelist I/O
AFBVLOCA		64C		Statement number locator
AFBVMOPP		470		Extended error handling
AFBVPOSA		30C4		Post ABEND processor
AFBVSCOP	3	654		Pre-Release 4.0 interface
AFBVSPAP		46C		Array dimension calculator
AFBVSPIP		4EC		Dynamic spill area processor
AFBVUNIN		1E4		Unnormalized operand interrupt handler

Figure 21. Optional Modules for AFBVLBCM

Notes:

1. Module AFBIBCOP is used when running object decks produced by FORTRAN compilers prior to VS FORTRAN. It is needed for formatted and unformatted I/O and for initialization from a main program.

2. These modules are used for the specified functions that are performed from object decks produced by FORTRAN compilers prior to VS FORTRAN Version 1 Release 4.0.

3. The module AFBVSCOP is used when running object decks produced by the VS FORTRAN Version 1 compiler from releases prior to Release 4.0. It is needed for formatted and unformatted I/O and for initialization from a main program or from a subroutine with character arguments.

Building the composite module AFBVLBCM: The composite module AFBVLBCM is created in a linkage editor step as follows:

```
//LKEDLBCM  EXEC PGM=IEWL,PARM='XREF,REUS'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=VSF2.VSF2FORT,DISP=OLD
//SYSLIB    DD   DSN=VSF2.VSF2FORT,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(AFBVLBC0)
 INCLUDE SYSLIB(AFBxxxxx)
  .
  .
  .
 ORDER AFBVLBC0
 ENTRY AFBLBCOM
 ALIAS IFYVLBCM
 NAME  AFBVLBCM(R)
/*
```

The linkage editor step creates the load module AFBVLBCM in the library VSF2.VSF2FORT, and replaces the previous copy of the load module. The inclusion of any of the optional modules in the composite module AFBVLBCM is controlled by the linkage editor INCLUDE statement, which refers to *AFBxxxxx*, where *AFBxxxxx* is to be replaced by the name of the module to be included. You must use a separate INCLUDE statement for each optional module that is included. Except for the module AFBVLBC0, no INCLUDE statements should be provided for the modules listed above as "Required."

Running SMP UCLIN for the composite module AFBVLBCM: Run SMP UCLIN to reflect how you customized your composite module AFBVLBCM. This will ensure the rebuilding of the composite module by SMP when possible future service affects those modules included in AFBVLBCM.

```
//UCLIN   JOB ....
//STEP     EXEC VSFACC4C              (See Notes 1 and 3)
//SMPCNTL  DD   *
  UCLIN CDS.

  ADD MOD(AFBxxxxx) LMOD(AFBVLBCM).  /* add new optional modules */
  or
  DEL MOD(AFBxxxxx) LMOD(AFBVLBCM).  /* delete optional modules  */
            .
            .
            .
  ENDUCL.
  LIST CDS LMOD(AFBVLBCM) XREF.       (See Note 2)

Notes:
1.  This is the same procedure you used to install VS FORTRAN Version 2.
2.  This will list the names of the modules that SMP considers to be
    contained in the composite module AFBVLBCM.  You should compare this
    list with the link-edit output after rebuilding AFBVLBCM to make sure
    the same modules are in both lists.
3.  If you have installed VS FORTRAN Version 2 Library product using SMP4,
    use EXEC VSFACC4L instead.
```

Figure 22. Example of JCL for SMP4 UCLIN for AFBVLBCM

```
//UCLIN   JOB ....
//STEP     EXEC VSFACCEC              (See Notes 1 and 3)
//SMPCNTL  DD   *
  SET BDY(TVSF).                     /* set target zone bound */
  UCLIN.

  ADD MOD(AFBxxxxx) LMOD(AFBVLBCM).  /* add new optional modules  */
  or
  DEL MOD(AFBxxxxx) LMOD(AFBVLBCM).  /* delete optional modules   */
            .
            .
            .
  ENDUCL.
  LIST TZONE LMOD(AFBVLBCM) XREF.     (See Note 2)

Notes:
1.  This is the same procedure you used to install VS FORTRAN Version 2.
2.  This will list the names of the modules that SMP considers to be
    contained in the composite module AFBVLBCM.  You should compare this
    list with the link-edit output after rebuilding AFBVLBCM to make sure
    the same modules are in both lists.
3.  If you have installed VS FORTRAN Version 2 Library product using SMP/E,
    use EXEC VSFACCEL instead.
```

Figure 23. Example of JCL for SMP/E UCLIN for AFBVLBCM

Composite Module AFBVRENC (Non-MVS/XA Only)

Figure 24 and Figure 25 on page 55 list the modules, both required and optional, that can be included in composite module AFBVRENC.

Module	Approx. Size	Default Set	Function
AFBVREN	19C	X	Internal linkage module
AFBVFIST	800	X	File status processor
AFBVGMFM	22D	X	GETMAIN/FREEMAIN
AFBVSIOS	26D4	X	Sequential I/O services
Total	32A8	4	

Figure 24. Required Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBDDCMP	244		Dynamic common
AFBVAMTP	12E		Alt. error option table processor
AFBVASUB	D34		Asynchronous I/O
AFBVBLNT	1EC	X	Implied DO in I/O
AFBVCLOP	21A	X	CLOSE statement
AFBVCOMH	15B2	X	Formatted I/O
AFBVCONI	314	X	Input floating-point conversion
AFBVCONO	810	X	Output floating-point conversion
AFBVCVTH	1174	X	Data conversion
AFBVDIOS	1834		Direct access I/O services
AFBVEMGN	FBC	X	Error message generator
AFBVERRE	238	X	Error summary
AFBVEXIP	8C		Return code processor
AFBVFMTP	15C		LCP define file
AFBVIIOS	260		Internal file services
AFBVINQP	968		INQUIRE statement
AFBVIOCP	2E8		BACKSPACE, REWIND, ENDFILE
AFBVIOFP	708	X	Formatted I/O
AFBVIOLP	12C4	X	List-directed I/O
AFBVIOUP	CC0	X	Unformatted I/O
AFBVKIOS	2B20		Keyed access I/O services
AFBVLINP	270		Link to reentrant CSECT
AFBVMSKL	5525	X	Message skeletons
AFBVOPEP	7D2	X	OPEN statement
AFBVSTAE	1148		ABEND processor
AFBVTEN	2C0	X	Powers of ten table
AFBVTRCH	B04	X	Traceback generator
AFBVVIOS	19BC		Nonkeyed VSAM I/O services

Figure 25. Optional Modules for AFBVRENC

Building the composite module AFBVRENC in MVS/SP (non-XA): For use in a non-XA version of an MVS/SP system, the composite module AFBVRENC is created in a linkage editor step as follows:

```
//LKEDRENC  EXEC PGM=IEWL,PARM='XREF,RENT'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=VSF2.VSF2FORT,DISP=OLD
//SYSLIB    DD   DSN=VSF2.VSF2FORT,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(AFBVREN)
 INCLUDE SYSLIB(AFBxxxxx)
  .
  .
  .
 ORDER AFBVREN
 ENTRY AFBVREN
 NAME  AFBVRENC(R)
/*
```

The linkage editor step creates the load module AFBVRENC in library VSF2.VSF2FORT; the previous copy of the load module is replaced. The inclusion of any of the optional reentrant modules in the composite module AFBVRENC is controlled by the linkage editor INCLUDE statement, which refers to *AFBxxxxx*, where *AFBxxxxx* is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each

optional module that is included. Except for the module AFBVREN, no INCLUDE statements should be provided for the modules listed above as "Required."

After the composite module AFBVRENC has been created, you may place it in the pageable link pack area for shared use by all regions. If it is not placed in the link pack area, it is loaded from the library VSF2.VSF2FORT.

Running SMP UCLIN for the composite module AFBVRENC: Run SMP UCLIN to reflect how you customized your composite module AFBVRENC. This will ensure the rebuilding of the composite module by SMP when possible future service affects those modules included in AFBVRENC.

```
//UCLIN   JOB ....
//STEP     EXEC VSFACC4C              (See Notes 1 and 3)
//SMPCNTL  DD   *
  UCLIN CDS.

  ADD MOD(AFBxxxxx) LMOD(AFBVRENC).  /* add new optional modules */
  or
  DEL MOD(AFBxxxxx) LMOD(AFBVRENC).  /* delete optional modules  */
            .
            .
            .
  ENDUCL.
  LIST CDS LMOD(AFBVRENC) XREF.       (See Note 2)


   Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.

2. This will list the names of the modules that SMP considers to be
   contained in the composite module AFBVRENC.  You should compare this list
   with the link-edit output after rebuilding AFBVRENC to make sure the same
   modules are in both lists.

3. If you have installed VS FORTRAN Version 2 Library product only, using
   SMP4, use EXEC VSFACC4L instead.
```

Figure 26. Example of JCL for SMP4 UCLIN for AFBVRENC

```
//UCLIN   JOB ....
//STEP     EXEC VSFACCEC              (See Notes 1 and 3)
//SMPCNTL  DD   *
  SET BDY(TVSF).                     /* set target zone bound */
  UCLIN.

  ADD MOD(AFBxxxxx) LMOD(AFBVRENC).  /* add new optional modules */
  or
  DEL MOD(AFBxxxxx) LMOD(AFBVRENC).  /* delete optional modules */
            .
            .
            .
  ENDUCL.
  LIST TZONE LMOD(AFBVRENC) XREF.     (See Note 2)
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. This will list the names of the modules that SMP considers to be contained in the composite module AFBVRENC. You should compare this list with the link-edit output after rebuilding AFBVRENC to make sure the same modules are in both lists.
3. If you have installed VS FORTRAN Version 2 Library product only, using SMP/E, use EXEC VSFACCEL instead.

Figure 27. Example of JCL for SMP/E UCLIN for AFBVRENC

Composite Module AFBVRENA (MVS/XA only)

Figure 28 and Figure 29 on page 58 list the modules, both required and optional, that can be included in composite module AFBVRENA.

Module	Approx. Size	Default Set	Function
AFBVAREN	166	X	Internal linkage module
AFBVGMFM	22D	X	GETMAIN/FREEMAIN
Total	398	2	

Figure 28. Required Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBDDCMP	244	X	Dynamic common
AFBVAMTP	12E		Alt. error option table processor
AFBVBLNT	1EC	X	Implied DO in I/O
AFBVCLOP	21A	X	CLOSE statement
AFBVCOMH	15B2	X	Formatted I/O
AFBVCONI	314	X	Input floating-point conversion
AFBVCONO	810	X	Output floating-point conversion
AFBVCVTH	1174	X	Data conversion
AFBVEMGN	FBC	X	Error message generator
AFBVERRE	238	X	Error summary
AFBVEXIP	8C	X	Return code processor
AFBVFMTP	15C		LCP define file
AFBVIIOS	260	X	Internal file services
AFBVINQP	968	X	INQUIRE statement
AFBVIOCP	2E8	X	BACKSPACE, REWIND, ENDFILE
AFBVIOFP	708	X	Formatted I/O
AFBVIOLP	12C4	X	List-directed I/O
AFBVIOUP	CC0	X	Unformatted I/O
AFBVLINP	270	X	Link to reentrant CSECT
AFBVMSKL	5525	X	Message skeletons
AFBVOPEP	7D2	X	OPEN statement
AFBVSTAE	1148	X	ABEND processor
AFBVTEN	2C0	X	Powers of ten table
AFBVTRCH	B04	X	Traceback generator

Figure 29. Optional Modules for AFBVRENA

Building the Composite Module AFBVRENA for MVS/XA: The composite module AFBVRENA is created for use in an MVS/XA system in a linkage editor step as follows:

```
//LKEDRENA  EXEC PGM=IEWL,PARM='XREF,RENT'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=VSF2.VSF2FORT,DISP=OLD
//SYSLIB    DD   DSN=VSF2.VSF2FORT,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(AFBVAREN)
 INCLUDE SYSLIB(AFBxxxxx)
  .
  .
  .
 ORDER AFBVAREN
 ENTRY AFBVAREN
 NAME  AFBVRENA(R)
/*
```

The linkage editor step creates the load module AFBVRENA in library VSF2.VSF2FORT; the previous copy of the load module is replaced. The module should have a residence mode of ANY so that it can reside above the 16-megabyte virtual storage line. The inclusion of any of the optional reentrant modules in the composite module AFBVRENA is controlled by the linkage editor INCLUDE statement, which refers to *AFBxxxxx*, where *AFBxxxxx* is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module you decide to include. Except for

the module AFBVAREN, no INCLUDE statements should be provided for the modules listed above as "Required."

After the composite module AFBVRENA has been created, it may be placed in the extended pageable link pack area (ELPA) for shared use by all regions. If it is not placed in the link pack area, it is loaded from the library VSF2.VSF2FORT.

Running SMP UCLIN for the composite module AFBVRENA: Run SMP UCLIN to reflect how you customized your composite module AFBVRENA. This will ensure the rebuilding of the composite module by SMP when possible future service affects those modules included in AFBVRENA.

```
//UCLIN   JOB ....
//STEP     EXEC VSFACC4C              (See Notes 1 and 3)
//SMPCNTL  DD   *
  UCLIN CDS.

  ADD MOD(AFBxxxxx) LMOD(AFBVRENA).  /* add new optional modules */
    or
  DEL MOD(AFBxxxxx) LMOD(AFBVRENA).  /* delete optional modules  */
         .
         .
         .
  ENDUCL.
  LIST CDS LMOD(AFBVRENA) XREF.       (See Note 2)


   Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.

2. This will list the names of the modules that SMP considers to be
   contained in the composite module AFBVRENA.  You should compare this list
   with the link-edit output after rebuilding AFBVRENA to make sure the same
   modules are in both lists.

3. If you have installed VS FORTRAN Version 2 Library product only, using
   SMP4, use EXEC VSFACC4L instead.
```

Figure 30. Example of JCL for SMP4 UCLIN for AFBVRENA

```
//UCLIN   JOB ....
//STEP     EXEC VSFACCEC              (See Notes 1 and 3)
//SMPCNTL  DD   *
  SET BDY(TVSF).                     /* set target zone bound */
  UCLIN.

  ADD MOD(AFBxxxxx) LMOD(AFBVRENA).  /* add new optional modules  */
   or
  DEL MOD(AFBxxxxx) LMOD(AFBVRENA).  /* delete optional modules  */
           .
           .
           .
  ENDUCL.
  LIST TZONE LMOD(AFBVRENA) XREF.     (See Note 2)
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. This will list the names of the modules that SMP considers to be contained in the composite module AFBVRENA. You should compare this list with the link-edit output after rebuilding AFBVRENA to make sure the same modules are in both lists.
3. If you have installed VS FORTRAN Version 2 Library product only, using SMP/E, use EXEC VSFACCEL instead.

Figure 31. Example of JCL for SMP/E UCLIN for AFBVRENA

Composite Module AFBVRENB (MVS/XA Only)

Figure 32 and Figure 33 on page 61 list the modules, both required and optional, that you can include in composite module AFBVRENB.

Module	Approx. Size	Default Set	Function
AFBVBREN	19C	X	Internal linkage module
AFBVFIST	800	X	File status processor
AFBVSIOS	2614	X	Sequential I/O services
Total	2FB8	3	

Figure 32. Required Modules for AFBVRENB

Module	Approx. Size	Default Set	Function
AFBVASUB	D34		Asynchronous I/O
AFBVDIOS	1834		Direct access I/O services
AFBVKIOS	2B20		Keyed access I/O services
AFBVVIOS	19BC		Nonkeyed VSAM I/O services

Figure 33. Optional Modules for AFBVRENB

Building the Composite Module AFBVRENB in MVS/XA: The composite module AFBVRENB is created for use in an MVS/XA system in a linkage editor step as follows:

```
//LKEDRENB  EXEC PGM=IEWL,PARM='XREF,RENT'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=VSF2.VSF2FORT,DISP=OLD
//SYSLIB    DD   DSN=VSF2.VSF2FORT,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(AFBVBREN)
 INCLUDE SYSLIB(AFBxxxxx)
  .
  .
  .
 ORDER AFBVBREN
 ENTRY AFBVBREN
 NAME  AFBVRENB(R)
/*
```

The linkage editor step creates the load module AFBVRENB in library VSF2.VSF2FORT; the previous copy of the load module is replaced. The module must have a residence mode of 24 so that it resides below the 16-megabyte virtual storage line. The inclusion of any of the optional reentrant modules in the composite module AFBVRENB is controlled by the linkage editor INCLUDE statement, which refers to *AFBxxxxx*, where *AFBxxxxx* is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module you decide to include. Except for the module AFBVBREN, no INCLUDE statements should be provided for the modules listed above as "Required."

After the composite module AFBVRENB has been created, it may be placed in the pageable link pack area for shared use by all regions. If it is not placed in the link pack area, it is loaded from the library VSF2.VSF2FORT.

Running SMP UCLIN for the composite module AFBVRENB: Run SMP UCLIN to reflect how you customized your composite module AFBVRENB. This will ensure the rebuilding of the composite module by SMP when possible future service affects those modules included in AFBVRENB.

```
//UCLIN   JOB ....
//STEP     EXEC VSFACC4C              (See Notes 1 and 3)
//SMPCNTL  DD   *
  UCLIN CDS.

  ADD MOD(AFBxxxxx) LMOD(AFBVRENB).  /* add new optional modules */
  or
  DEL MOD(AFBxxxxx) LMOD(AFBVRENB).  /* delete optional modules  */
           .
           .
           .
  ENDUCL.
  LIST CDS LMOD(AFBVRENB) XREF.       (See Note 2)
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. This will list the names of the modules that SMP considers to be contained in the composite module AFBVRENB. You should compare this list with the link-edit output after rebuilding AFBVRENB to make sure the same modules are in both lists.
3. If you have installed VS FORTRAN Version 2 Library product only, using SMP4, use EXEC VSFACC4L instead.

Figure 34. Example of JCL for SMP4 UCLIN for AFBVRENB

```
//UCLIN   JOB ....
//STEP     EXEC VSFACCEC                (See Notes 1 and 3)
//SMPCNTL  DD   *
  SET BDY(TVSF).                        /* set target zone bound */
  UCLIN.

  ADD MOD(AFBxxxxx) LMOD(AFBVRENB).     /* add new optional modules */
  or
  DEL MOD(AFBxxxxx) LMOD(AFBVRENB).     /* delete optional modules */
            .
            .
            .
  ENDUCL.
  LIST TZONE LMOD(AFBVRENB) XREF.       (See Note 2)
```

Notes:

1. This is the same procedure you used to install VS FORTRAN Version 2.
2. This will list the names of the modules that SMP considers to be contained in the composite module AFBVRENB. You should compare this list with the link-edit output after rebuilding AFBVRENB to make sure the same modules are in both lists.
3. If you have installed VS FORTRAN Version 2 Library product only, using SMP/E, use EXEC VSFACCEL instead.

Figure 35. Example of JCL for SMP/E UCLIN for AFBVRENB

Chapter 4. Installation under VM

This chapter describes the standard installation of VS FORTRAN Version 2 under VM. If you have ordered the VS FORTRAN Version 2 Library product only (5668-805), follow the installation procedures as described for VS FORTRAN Version 2, but use the library installation EXEC as noted under "Basic Machine-Readable Material," below.

For specific information on space allocations and other details needed for installation, see the program directory. For information on the features you can customize to fit your site's needs, see Chapter 5, "Customization under VM" on page 81.

Note:

We recommend that you read the entire book once (except Chapter 2, Chapter 3, and Appendix D, which cover MVS) **before** you actually begin the installation process.

Basic Machine-Readable Material

The distribution medium for VS FORTRAN Version 2 and VS FORTRAN Version 2 Library is either an unlabeled 9-track tape or a 3480 tape cartridge, written in EBCDIC in CMS VMFPLC2 DUMP format. It is intended to be used under the Conversational Monitor System (CMS) component of VM.

See the program directory for the order of files and their descriptions when installing either the complete VS FORTRAN Version 2 product or the Library product only.

If you are installing the Library product only, follow the instructions for installing VS FORTRAN Version 2, but use the EXEC I5668805 instead of EXEC I5668806.

Storage Requirements

See the program directory for details on the additional system library storage required to install VS FORTRAN Version 2.

Files Created or Used during Installation

Text Libraries

VSF2FORT TXTLIB	Execution-time library routines needed for the creation of an executable program in both link mode and load mode. (In the following sections, this text library is referred to as the principal text library.)
VSF2LINK TXTLIB	Interface routines used in link mode only. This library must be concatenated ahead of VSF2FORT for the creation of an executable program in link mode. (In the following sections, this text library is referred to as the link mode text library.)
VSF2MATH TXTLIB	Alternative mathematical routines

Load Library

VSF2LOAD LOADLIB	Routines that may be loaded during execution in load mode or when Interactive Debug is used. (In the following sections, this library is referred to as the load library.)

Interactive Debug Files

VSF2MLIB MACLIB	ISPF Message Library
VSF2PLIB MACLIB	ISPF Panel Library
AFFLOADF TEXT	ISPF Invocation Module
AFFFX11 EXEC	ISPF Invocation EXEC

CMS Help Files

Installation Overview

You install VS FORTRAN Version 2 by invoking the installation EXEC supplied on the distribution tape. The installation EXEC copies the files needed to execute a VS FORTRAN program onto a work disk. It then gives you the opportunity to customize the product. If you are satisfied with the IBM-supplied defaults for compiler and library options and do not want to customize your VS FORTRAN Version 2 product, you do not need to take any further action.

If you want to customize your VS FORTRAN Version 2 product to better suit the needs of your site, you will be able to do so by responding to prompts from the installation EXEC. When you have answered all the prompts, the EXEC builds the VS FORTRAN Version 2 product and places it on your product disk.

Note: You should keep the distribution tape and documents prepared by IBM as backup in case the product has to be re-created or you want to further customize your VS FORTRAN Version 2 product sometime in the future. Any customizing you do at installation time can also be done at any time after installation, if the needs of your site change.

Preparing to Install VS FORTRAN Version 2

The following sections give detailed descriptions of the steps outlined in the installation overview. If a previous version of VS FORTRAN is already installed on your system, you may want to save it somewhere other than your product disks to prevent it from being written over by this release.

Before executing the EXEC, complete the following steps:

1. Determine where you will store the components required for service application to this product. The disk you choose will be your work disk. You **must** access this work disk as your A-disk. It must not be the disk containing the installed executable product.

   If you do not want to keep the installation materials on a permanent disk, create a temporary work disk and access it as your A-disk. See the program directory for space requirements.

   If you use a temporary disk, unload its contents to tape after the installation to retain the files you need later to install service. You can do this by issuing the command

   ```
   VMFPLC2 DUMP * * A
   ```

2. Determine where you will install the executable product. This may be the system disk or any other minidisk with enough space to hold the entire product. See the program directory for space requirements.

   To avoid the performance degradation that is likely to occur with a block size of 800, the product disk should have a block size of at least 1024 bytes.

3. During installation, you are prompted for information. To reply to the prompts, consider the following:

   a. If you are satisfied with the IBM-supplied defaults for the compiler options, reply NO to the prompt about editing the VSF2COM macro instruction. (For more information on the VSF2COM macro, see Appendix A, "Macros for Customizing VS FORTRAN Version 2" on page 95.)

b. Determine whether you want the compiler to be installed as a discontiguous saved segment (DCSS). (You must be a class E user to do this.) If so, be sure that space has been reserved on a CP-owned DASD volume for the saved segment, and that the segment name has been placed in the VM system table **prior** to starting the installation.

   Also, determine the size of virtual storage needed for including the compiler saved segment during the installation. See "Installing the Compiler as a Discontiguous Saved Segment (DCSS)" on page 82 for more detail **prior** to starting the installation.

c. You will be asked to choose the CMS DCB characteristics for the files created and used by your FORTRAN programs:

   ```
   RECFM F, LRECL 80, BLOCK 80
   ```

   or to choose OS/VS characteristics:

   ```
   RECFM U, LRECL 800, BLOCK 32756
   ```

   If you are migrating from an earlier version or release of VS FORTRAN Version 2, use the DCB characteristics that match the ones you are currently using. Otherwise, we recommend that you use the CMS characteristics shown.

d. You will be asked if you want to change the I/O unit number defaults. The default number of units in the FORTRAN unit assignment table is 99. If you want to change the number of units in the FORTRAN unit assignment table, or the FORTRAN unit numbers that are used as defaults for READ, PUNCH, and WRITE statements, respond YES to the prompt about editing the VSF2LIB macro instruction. For information on what you can change and how to code your changes, see "VSF2LIB: For Changing I/O Unit Number Defaults" on page 102.

e. You will be asked if you want to change the execution-time option defaults. If you reply YES to this question, you will be allowed to change the defaults for all users of the VS FORTRAN Version 2 Library. For information on what you can change and how to code your changes, see "VSF2PARM: For Changing Execution-Time Option Defaults" on page 104.

f. You will be asked if you want to install the vector library routines. Answer YES to this prompt only if you plan to execute vector programs on the system. The vector library routines will then be added to the principal text library (VSF2FORT TXTLIB).

   Note: If you are going to install the vector routines, a PTF must be applied to upgrade the CMS TXTLIB command prior to installing VS FORTRAN Version 2. At present, the current maximum limit of entries allowed in a TXTLIB is 1000. With the addition of the vector subroutines, this limit is exceeded. Application of the necessary PTF allows this limit to be increased. Refer to the program directory for the numbers of the applicable PTFs.

g. You will be asked if you want to change the IBM-supplied default set of modules that are included in the LOADLIB version of the composite module AFBVRENC. If you answer YES to this prompt, you must decide which modules you want to include in this composite. See "Composite Module AFBVRENC" on page 115 for the list of modules you can include. You will also be asked if you want to build one or more DCSSs, each with a version of this module.

   If you decide to install a copy of the composite module AFBVRENC in one or more DCSSs, you must be sure that space has been reserved on a CP-owned DASD volume for the saved segments, and that the segment names have been placed in the VM system table **prior** to starting the installation.

   Also, determine the size of virtual storage needed for including the DCSS during the installation. For more information on AFBVRENC as a DCSS, see "AFBVRENC as a Discontiguous Saved Segment" on page 91 **prior** to starting the installation.

h. You will be asked if you want to change the IBM-supplied default module list for composite module AFBVLBCM. If you answer YES to this prompt, you must decide which modules are to be included in this composite. For more information, see "Building Composite Module AFBVLBCM" on page 119.

Installing VS FORTRAN Version 2

Step 1: Beginning the Installation

1. Log on to VM and IPL CMS.

2. If you are installing the compiler or library outside of a DCSS, your virtual machine size must be defined at 3 megabytes or larger.

   For example, if you want the VS FORTRAN Version 2 compiler to start at 2 megabytes, then your virtual machine size should be at least 4 megabytes. In this example, you would issue the command:

   ```
   CP DEFINE STORAGE 4M
   ```

   and then re-IPL CMS.

   If you are installing the compiler or library in a DCSS, your virtual machine must have privilege class E in addition to class G, and the virtual machine size must be defined large enough to contain the shared segment. For more information, see "Installing the Compiler as a Discontiguous Saved Segment (DCSS)" on page 82.

   For example, if you want the VS FORTRAN Version 2 compiler in a DCSS to start at 5 megabytes, then your virtual machine size should be at least 7 megabytes. In this example, you would issue the command:

```
CP DEFINE STORAGE 7M
```

and then re-IPL CMS.

3. Attach and mount the distribution tape at virtual address 181.

Step 2: Linking to the Disks

1. Link in write mode to a work disk and access it as your A-disk. Refer to step 1 on page 67 under "Preparing to Install VS FORTRAN Version 2."

2. Link in write mode and access a second disk as a product disk. Remember: you may not use the A-disk because the A-disk is used as a work disk. We recommend that you access your product disk as your E-disk.

Step 3: Loading the EXEC

Load the first tape file containing the I5668806 EXEC (or I5668805 if installing VS FORTRAN Version 2 Library product only) onto the work disk by giving the command:

```
VMFPLC2 LOAD * * A
```

Step 4: Executing the EXEC

Execute the I5668806 EXEC (or the I5668805 EXEC) by entering

```
I5668806 (or I5668805)
```

This will load VS FORTRAN Version 2 modules onto the A-disk and begin installation of the product.

Respond to the prompts from the installation EXEC which ask you sequentially to do the following (you can halt execution of the EXEC by responding to any prompt with QUIT):

1. Provide the file mode of the product disk, for example E. Remember that you may not use the A-disk because the A-disk is used as a work disk.

2. Verify that the distribution tape is mounted on device 181.

3. Change the name of the macro library or accept the default (VSF2MAC).

4. Edit the VSF2COM macro instruction to establish new compiler option defaults or accept the IBM-supplied defaults. For more information, see "VSF2COM: For Changing Compiler Option Defaults" on page 96.

5. Choose whether or not to install the compiler in a DCSS and, if so, provide the name or names of the DCSSs to be generated. For more information, refer to "Installing the Compiler as a Discontiguous Saved Segment (DCSS)" on page 82.

6. Choose the CMS file characteristics or IBM-supplied characteristics found in the unit assignment table. Refer to step 3c on page 68.

7. Edit the VSF2LIB macro instruction to establish new I/O unit number defaults or accept the IBM-supplied defaults. See "VSF2LIB: For Changing I/O Unit Number Defaults" on page 102.

8. Edit the VSF2PARM macro instruction to establish new execution-time option defaults or or accept the IBM-supplied defaults. See "VSF2PARM: For Changing Execution-Time Option Defaults" on page 104.

9. Change the name of the principal text library or accept the default (VSF2FORT).

10. Choose whether or not to install the vector library routines. Refer to step 3f on page 68.

11. Change the name of the alternative math library or accept the default (VSF2MATH).

12. Change the name of the link mode text library or accept the default (VSF2LINK).

13. Choose whether or not to install a copy of the library composite module AFBVRENC in one or more DCSSs; if so, provide the DCSS names. Refer to step 3g on page 69.

14. Choose whether or not to change the default module list for composite module AFBVLBCM. Refer to "Building Composite Module AFBVLBCM" on page 119.

15. Change the name of the load library or accept the default (VSF2LOAD).

16. Choose whether you want Interactive Debug installed:

    a. Choose whether to load ISPF libraries for Interactive Debug.

    b Choose whether to install Interactive Debug line mode HELP files. If you do choose to install the HELP files, you will be prompted on whether you want to use them under CMS, Release 3, or another release.

17. Choose whether you want to compile and execute the sample program if you have not installed a compiler DCSS. See "Verifying Success for the Compiler and Library" on page 77.

Remember, you can halt execution of the installation EXEC by responding to any prompt with QUIT. To restart the installation, execute the I5668806 (or I5668805) EXEC again. You will be given the choice of either restarting at the beginning, or (assuming you had made some progress before quitting) restarting at the first major step just before your quitting point. This can be at any step from 3 on page 70 through 17 above.

After you have responded to all the prompts, the installation EXEC will complete the installation. You will see the following message, indicating successful completion of the installation.

```
'PRODUCT INSTALLATION IS COMPLETE'
```

Note: You should keep the installation materials supplied by IBM. If you used a temporary disk as a work disk during installation, unload its contents to tape after the installation to retain the files you will later need to install service. You can do this by issuing the command

```
VMFPLC2 DUMP * * A
```

Step 5: Installing VS FORTRAN Version 2 Interactive Debug

If you are an ISPF (Interactive System Product Facility) user, proceed with the following section "Installing Interactive Debug (ISPF/PDF Users Only)." If you are *not* an ISPF user, go to the section "Installing Interactive Debug (Non-ISPF Users Only)" on page 76.

If you are *not* installing VS FORTRAN Version 2 Interactive Debug, skip this step and go to "Step 6: Verifying a Successful Installation" on page 77. If you have installed the Library product only, skip this step and go to Chapter 5, "Customization under VM" on page 81.

Installing Interactive Debug (ISPF/PDF Users Only)

1. Using ISPF Edit, modify foreground selection panel ISRFPA, which is in your site's ISPF panel MACLIB (normally ISRPLIB). Figure 36 on page 73 is an example of panel ISRFPA being updated for VS FORTRAN Version 2 Interactive Debug.

   Enter the name AFFFP11C (in uppercase) to be used to invoke VS FORTRAN Version 2 Interactive Debug. You may change any of the titles in the panel definition, but you must also change the corresponding line in the bottom of the panel definition.

   To add Interactive Debug, change any option at the top of the screen to specify VS FORTRAN Version 2 Interactive Debug and change the corresponding numbered line at the bottom of the screen to specify AFFFP11C.

```
%----------------------  FOREGROUND SELECTION PANEL  -------------------------
%OPTION  ===>_ZCMD                                                          +
%
%   1+- System assembler                  % 7+- Linkage editor
%   2+- OS/VS COBOL compiler              % 8+- Load
%   3+- VS FORTRAN compiler               % 9+- SCRIPT/VS
%   4+- PL/I checkout compiler            %10+- COBOL interactive debug
%   5+- PL/I optimizing compiler          %11+- VS FORTRAN V2 interactive debug
%   6+- PASCAL/VS compiler                %12+- Member parts list
%                                                                    |
+                                                                   |
+SOURCE DATA PACKED%===>_ZFPACK  +(YES or NO)                        |
)INIT                                                               |
  .HELP = ISR40000                                          (name change)
  IF (&ZXPACK ¬= '        ')
    &ZFPACK = &ZXPACK
    &ZXPACK = '        '
  &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,*,NO) /* DATA FORMAT CHECK  */
)REINIT
  &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,*,NO) /* DATA FORMAT CHECK  */
)PROC
  &ZFPACK = TRUNC(&ZFPACK,1)
  VER (&ZFPACK,NB,LIST,Y,N)          /* Y=EXPAND PACKED DATA         */
  &ZFPACK = TRANS(TRUNC(&ZFPACK,1),Y,YES,N,NO)
  VPUT (ZFPACK) PROFILE
  &ZSEL = TRANS( TRUNC (&ZCMD,'.')
                1,'PGM(ISRFPR) PARM(ISRFP01) NEWPOOL'
                2,'PGM(ISRFPR) PARM(ISRFP02) NEWPOOL'
                3,'PGM(ISRFPR) PARM(ISRFP03) NEWPOOL'
                4,'PGM(ISRFPR) PARM(ISRFP04) NEWPOOL'
                5,'PGM(ISRFPR) PARM(ISRFP05) NEWPOOL'
                6,'PGM(ISRFPR) PARM(ISRFP06) NEWPOOL'
                7,'PGM(ISRFPR) PARM(ISRFP07) NEWPOOL'
                8,'PGM(ISRFPR) PARM(ISRFP08) NEWPOOL'
                9,'PGM(ISRFPR) PARM(ISRFP09) NEWPOOL'
               10,'PGM(ISRFPR) PARM(ISRFP10) NEWPOOL'
               11,'PGM(ISRFPR) PARM(AFFFP11C) NEWPOOL'<------
              ,12,'PGM(ISRFPR) PARM(ISRFP12) NEWPOOL'         |
              ,'','',                                         |
               *,'?' )                                        |
)END                                          (required change)
```

Figure 36. ISPF Foreground Selection Panel Definition

For example, as shown in Figure 36, you can change option 11 from the old FORTRAN Interactive Debug product to VS FORTRAN Version 2 Interactive Debug, and change entry 11 at the bottom of the panel from ISRFP11 to AFFFP11C (enter in uppercase).

Or, you may **add** VS FORTRAN Version 2 Interactive Debug to your ISPF panel by entering:

```
%xx+- VS FORTRAN Version 2 Interactive Debug
```

on the upper part of the panel (where xx is the number of the option). You must also enter

```
xx,'PGM(ISRFPR) PARM(AFFFP11C) NEWPOOL'
```

on the lower part of the panel. During installation, AFFFP11C is included in the library containing the VS FORTRAN Version 2 Interactive Debug's ISPF panel definitions (VSF2PLIB MACLIB).

2. Using ISPF Edit, modify foreground help panel ISR40000, which is in your site's ISPF panel MACLIB (normally ISRPLIB).

   Figure 37 on page 75 shows an example of this panel and identifies the two fields you need to change. Change an option as you did in the foreground selection panel (or add an option here if you added one there), and change (or add) the correspondingly numbered entry in the bottom left part of the panel to specify AFF4B000. (These changes are similar to the ones you made in step 1 on page 72, and you make them in the same way.)

   The name AFF4B000 is used to obtain the VS FORTRAN Version 2 Interactive Debug primary Help panel. During installation, AFF4B000 is included in the library containing the VS FORTRAN Version 2 Interactive Debug ISPF panel definitions (VSF2PLIB MACLIB).

3. In order to compile VS FORTRAN Version 2 programs through the ISPF environment, modify the EXEC ISRFX03 in your location's ISPF minidisk.

   Replace the line

   ```
   FORTVS &ZFNAME (&FFORT &FFOR)
   ```

   with

   ```
   FORTVS2 &ZFNAME (&FFORT &FFOR)
   ```

4. Use an editor to build or modify an EXEC to invoke ISPF. This EXEC must include FILEDEFs for the MACLIBs created during installation of VS FORTRAN Version 2 Interactive Debug. An example of such an EXEC is shown in Figure 38 on page 76. This example assumes that you have PDF as well as ISPF Version 2.

5. If you have changed the names of the principal text library (VSF2FORT TXTLIB) or the load library (VSF2LOAD LOADLIB) during installation, then the ISPF foreground EXEC (AFFFX11), which allocates files for IAD needs to be changed. Edit the file and locate the CMS GLOBAL commands and change the appropriate names.

```
%TUTORIAL ------------- FOREGROUND PROCESSING OPTION ---------------- TUTORIAL
%OPTION  ===>_ZCMD                                                          +
%                      ----------------------------------
                       |     FOREGROUND PROCESSING      |
                       ----------------------------------
   The foreground processing option allows a processing program to be executed
   in the foreground under ISPF.  The foreground selection panel, which is
   displayed when option%4+is entered on the primary option panel, allows you to
   select one of the processing programs listed below. %Note+- The foreground
   processor may be customized by changing values of variables in the PROC
   section of the foreground panels.  If you or your System Programmer modified
   any of these panels, some of the following information may not apply.

 The following topics are presented in sequence, or may be selected by number:
    %0+- General information                %6+- PASCAL/VS compiler
    %1+- System assembler                   %7+- CMS LKED command
    %2+- OS/VS COBOL compiler               %8+- CMS LOAD command
    %3+- VS FORTRAN compiler                %9+- SCRIPT/VS
    %4+- PL/I checkout compiler            %10+- COBOL Interactive Debug
    %5+- PL/I optimizing compiler          %11+- VS FORTRAN V2 Interactive Debug
                                           %12+- Member Parts listing
)PROC                                                                  |
   &ZSEL = TRANS( &ZCMD                                                |
                 0,ISR40001                                  (name change)
                 1,ISR41000
                 2,ISR42000
                 3,ISR43000
                 4,ISR44000
                 5,ISR45000
                 6,ISR46000
                 7,ISR47000
                 8,ISR48000
                 9,ISR49000
                10,ISR4A000
                11,AFF4B000  <-------- (required change)
                12,ISR4C000
                 )
   &ZUP = ISR00003
)END
```

Figure 37. ISPF Foreground Processing Help Panel

```
&TRACE  OFF
*
* THIS IS THE ISPF (SYSTEM PRODUCTIVITY FACILITY) COMMAND EXEC USED TO
* RUN THE PROGRAM DEVELOPMENT FACILITY.  BEFORE INVOKING THIS EXEC, YOU
* MUST ISSUE FILEDEFS FOR ANY ADDITIONAL PANEL, MESSAGE, TABLE, AND/OR
* SKELETON LIBRARIES FROM WHICH YOU PLAN TO OPERATE.
*
* THE ISPDCS COMMAND HAS AN OPTIONAL KEYWORD PARAMETER (KEYWORD
* IS "PDFDCSS") USED TO SPECIFY A PDF DCSS NAME. (IF OMITTED,
* THEN THE DEFAULT PDF DCSS NAME OF "ISRDCSS" WILL BE USED).
*
* IF YOU ARE USING ISPF VERSION 1, CHANGE ISPDCSS2 TO ISPDCSS AND
* ISPVM2 to ISPVM IN THE LAST LINE.
*
FILEDEF ISPPLIB DISK VSF2PLIB MACLIB * (PERM CONCAT
FILEDEF ISPPLIB DISK ISRPLIB MACLIB * (PERM CONCAT
FILEDEF ISPPLIB DISK ISPPLIB MACLIB * (PERM CONCAT
*
FILEDEF ISPMLIB DISK VSF2MLIB MACLIB * (PERM CONCAT
FILEDEF ISPMLIB DISK ISRMLIB MACLIB * (PERM CONCAT
FILEDEF ISPMLIB DISK ISPMLIB MACLIB * (PERM CONCAT
*
FILEDEF ISPSLIB DISK ISRSLIB MACLIB * (PERM CONCAT
*
FILEDEF ISPTLIB DISK ISRTLIB MACLIB * (PERM CONCAT
FILEDEF ISPTLIB DISK ISPTLIB MACLIB * (PERM CONCAT
*
FILEDEF ISPPROF DISK DEFAULTS MACLIB A (PERM
FILEDEF FT05F001 TERMINAL (PERM
FILEDEF FT06F001 TERMINAL (PERM
*
ISPDCS  ISPDCSS2 ISPVM2 PANEL(ISR@PRIM) NEWAPPL(ISR) &ARGSTRING
```

Figure 38. Example of an EXEC to Invoke ISPF

Installing Interactive Debug (Non-ISPF Users Only)

Using an editor, build an EXEC to execute a VS FORTRAN Version 2 program with VS FORTRAN Version 2 Interactive Debug. The EXEC must include FILEDEFs for all files used by VS FORTRAN Version 2 Interactive Debug, and definitions for the required TXTLIBs and LOADLIBs.

Note: If you changed the names of TXTLIBs and LOADLIBs during installation, be sure to refer to your new library names in the EXEC.

The EXEC shown in Figure 39 on page 77 has one positional parameter: the name of the program to be executed. You can add execution-time options after the first parameter. To invoke the EXEC and debug program AFFIVP, enter FORTIAD AFFIVP.

The sample EXEC assumes that the program to be debugged will execute in load mode If the program is to execute in link mode, the EXEC needs to specify VSF2LINK first in the GLOBAL TXTLIB statement, as follows:

```
GLOBAL TXTLIB VSF2LINK VSF2FORT CMSLIB TSOLIB
```

For more information on link mode versus load mode, see "Execution-Time Loading of Library Modules" on page 89.

```
&TRACE
GLOBAL TXTLIB VSF2FORT CMSLIB TSOLIB
* THE ABOVE STATEMENT ASSUMES YOU WILL RUN IN LOAD
* MODE.  IF YOU WANT TO RUN IN LINK MODE, REPLACE THE
* ABOVE STATEMENT WITH THE FOLLOWING GLOBAL STATEMENT:
* GLOBAL TXTLIB VSF2LINK VSF2FORT CMSLIB TSOLIB
*
GLOBAL LOADLIB VSF2LOAD
FILEDEF AFFON    DUMMY
FILEDEF AFFPRINT DISK &1 AFFPRINT A
LOAD &1 (CLEAR
START * DEBUG &2 &3 &4 &5 &6 &7 &8
&EXIT &RETCODE
```

Figure 39. Sample FORTIAD EXEC

Step 6: Verifying a Successful Installation

You can verify the success of your installation of VS FORTRAN Version 2 by using the sample programs provided on the distribution tape. One sample program verifies the installation for the Compiler and Library, and the other verifies installation of VS FORTRAN Version 2 Interactive Debug.

Verifying Success for the Compiler and Library

The installation EXEC allows you to run a sample program, AFBIVP, to verify the success of your installation of VS FORTRAN Version 2. Your use of the installation EXEC is then complete.

However, if you have just installed the compiler as a discontiguous saved segment, you must define a virtual machine to fit below the address of the saved segment and re-IPL. For example, if the compiler begins at 2 megabytes (2Mb), you must enter the following to run the sample program:

```
CP DEFINE STORAGE 2M
CP IPL CMS
ACCESS xxx A
ACCESS yyy B
FORTVS2 AFBIVP
GLOBAL TXTLIB VSF2FORT CMSLIB
GLOBAL LOADLIB VSF2LOAD
LOAD AFBIVP (NOAUTO START
```

where xxx is the work disk on which the sample program, AFBIVP, is loaded, and yyy is the product disk.

Note: Library text files are CMS files with names beginning with AFB and with file types of TEXT. These files must not be on any accessed disk during the execution of the LOAD command unless the option NOAUTO is specified. During installation of the VS FORTRAN Version 2 library, the library text files are placed on a different minidisk from the text libraries so as to eliminate the problems that would occur because of the omission of the NOAUTO option on

the LOAD command. However, the installer has access to both disks and must use the NOAUTO option of the LOAD command or an error results.

Verifying Success for Interactive Debug

A second sample FORTRAN program provided on the installation tape computes the diameter, circumference and area of a circle. It is used to verify that VS FORTRAN Version 2 Interactive Debug has been correctly installed. The sample program is unloaded as file AFFIVP FORTRAN on the installation work disk. It is a standard CMS file that can be edited, printed, and processed using normal CMS commands.

The sample program should be compiled using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default. To compile AFFIVP FORTRAN under VS FORTRAN Version 2 enter the following:

```
FORTVS2 AFFIVP (SDUMP
```

This command should produce the following messages with no error or warning messages:

```
**CIRCLE** END OF COMPILATION 1 ******

**DIAM** END OF COMPILATION 2 ******

**CIRCUM** END OF COMPILATION 3 ******

**AREA** END OF COMPILATION 4 ******
```

The next two sections describe how to execute the compiled program with VS FORTRAN Version 2 Interactive Debug. The program may be executed either under ISPF or in line mode.

ISPF (Interactive System Product Facility) Environment

Before invoking ISPF, make sure that all EXECs and panels have been modified, as described under "Step 5: Installing VS FORTRAN Version 2 Interactive Debug" on page 72. Invoke ISPF in the standard manner for your site.

- When the PRIMARY OPTION PANEL appears, select option 4, FOREGROUND, which causes the FOREGROUND SELECTION MENU panel to be displayed.
- Find the line specifying VS FORTRAN Version 2 Interactive Debug, and enter the associated number. (In the example in Figure 36 on page 73, number 11 was used.) The FOREGROUND VS FORTRAN VERSION 2.1.1 INTERACTIVE DEBUG panel is displayed.
- Enter the name of the program to be executed, AFFIVP, on the line labeled FILE ID, and enter DEBUG as the first entry below the line labeled EXECUTION TIME OPTIONS. The next panel displayed should be the Interactive Debug panel.

You are now ready to issue Interactive Debug commands as described in the "Expected Results" section below.

Line Mode Environment

Using the invocation EXEC you created in "Step 5: Installing VS FORTRAN Version 2 Interactive Debug" on page 72, execute the sample program, AFFIVP, under the control of VS FORTRAN Version 2 Interactive Debug. Assuming you have given the invocation EXEC the name FORTIAD, enter

```
FORTIAD AFFIVP
```

After the FORTIAD EXEC has executed, you will receive some messages, then the VS FORTRAN Version 2 Interactive Debug prompt, preceded by the product name and copyright information:

```
WHERE: CIRCLE.7
FORTIAD
```

You are now ready to issue Interactive Debug commands as described in the next section.

Expected Results

Input and output for a set of Interactive Debug commands is shown in Figure 40 on page 80. You may enter other commands to further verify correct installation. All lines beginning with an asterisk (*) are lines that were entered on the terminal. However, when entering the commands, do not type the leading asterisk. This log was obtained during execution under ISPF.

```
WHERE: CIRCLE.7
* listsubs
PROGRAM UNIT  COMPILER   OPT   TIMING
   CIRCLE     VSF 2.1.1   0      OFF
   DIAM       VSF 2.1.1   0      OFF
   CIRCUM     VSF 2.1.1   2      OFF
   AREA       VSF 2.1.1   3      OFF
* describe (data pi)
DATA:         REAL*4
   RANK = 1, SIZE = 3 ELEMENTS
   DIM 1:     EXTENT = 3,  LBOUND = 1,  UBOUND = 3
PI:           REAL*8
* at diam.entry (step)
* go
FT06F001  ENTER THE VALUE OF THE CIRCLE RADIUS (xxx.xx):
FT05F001 INPUT: PRECEDE INPUT WITH % OR ENTER IAD COMMAND
* %352.67
AT: DIAM.ENTRY
NEXT: DIAM.3
* set diam.value = 0.0
* when test value
* go
WHEN: "TEST" SATISFIED;
CURRENTLY AT DIAM.4
* offwn test
* at circle.42 (list '= READY FOR TERMINATION ='%go)
* listbrks
CURRENT BREAKPOINTS:
  CIRCLE.42
  DIAM.ENTRY
CURRENT WHEN CONDITIONS:
  TEST OFF  DIAM.VALUE
CURRENT HALT STATUS: OFF
* go
FT06F001  THE DIAMETER OF THE CIRCLE IS 705.34
FT06F001  THE CIRCUMFERENCE OF THE CIRCLE IS 2215.89
FT06F001  THE AREA OF THE CIRCLE IS 390738.94
AT: CIRCLE.42
= READY FOR TERMINATION =
PROGRAM HAS TERMINATED; RC = 0
* quit
```

Figure 40. CMS Interactive Debug Input/Output

Chapter 5. Customization under VM

The following features can be customized under VM after you have installed the product:

- Alternative mathematical library subroutines
- The compiler as a discontiguous saved segment
- Execution-time options
- Extended error handling facility
- Execution-time loading of library modules

Alternative Mathematical Library Subroutines

The alternative mathematical library contains the VS FORTRAN Version 1 standard mathematical routines. This library allows users to have access to the routines that were available to them as part of the VS FORTRAN Version 1 product. Those routines that were alternative in the VS FORTRAN Version 1 product are not available in VS FORTRAN Version 2.

The alternative mathematical library subroutines are placed in VSF2MATH by the installation process.

To make the alternative mathematical library routines available to all users, create an EXEC which will issue the following CMS statement for use by the CMS LOAD command for execution in load mode:

```
GLOBAL TXTLIB VSF2MATH VSF2FORT CMSLIB
```

or this statement for execution in link mode:

```
GLOBAL TXTLIB VSF2MATH VSF2LINK VSF2FORT CMSLIB
```

Installing the Compiler as a Discontiguous Saved Segment (DCSS)

If you decide to install the compiler as a discontiguous saved segment (DCSS), your VM system programmer must define the DCSS to VM. To define the DCSS, the system programmer must obtain from you the following information:

- The name of the DCSS. This may be any name you wish.

  However, if you expect in the future to have more than one release of VS FORTRAN Version 2 installed on your system at the same time, you should **not** use the name DSSVFORT for a DCSS, nor should you use any other DCSS name that you will use with another release of the product.

- The starting address of the DCSS. With the assistance of your VM system programmer, determine this address, using the following guidelines:

  - The address should be at least as large as the virtual machine of any of the VS FORTRAN Version 2 users you expect to use this DCSS.

  - The address should not be unnecessarily high; if it is, storage is wasted for unreferenced CP segment table entries. However, in order to accommodate users with different virtual machine sizes, you can create several DCSSs, each with a different starting address.

  - The address should not allow this DCSS to overlap any other saved segment that may be used at the same time. For example, if you invoke the compiler from an ISPF panel, the compiler DCSS should not overlap any saved segments that contain parts of ISPF. Your VM system programmer can help you determine a potential overlap.

After the VM system programmer has defined the segments, you can proceed with installing the compiler as a DCSS.

Defining the Compiler DCSS to VM

Although a discontiguous saved segment (DCSS) need not be shared among all users, the VS FORTRAN Version 2 compiler is reentrant and can be shared.

When installing the compiler as a DCSS, your system programmer must complete the following steps before you execute the installation EXEC (I5668806):

1. Allocate permanent space on a CP-owned DASD volume to contain the DCSS. The compiler needs 336 4K pages, which is equivalent to 21 64K segments. (Refer to *VM/SP Planning Guide and Reference*, SC19-6201, for information on the amount of disk space needed.)

2. Define the DCSS by adding a NAMESYS macro instruction to your site's DMKSNT ASSEMBLE module (see *VM/SP Planning Guide and Reference*, SC19-6201, and *VM/SP System Programmer's Guide*, SC19-6203). If more than one DCSS is to be built to hold copies of the compiler module at different addresses, then there must be a NAMESYS macro instruction defining each DCSS.

The following example of the NAMESYS macro instruction defines DSSVFORT. (DSSVFORT is the name of the DCSS that will be used if this name is not overridden during compiler installation.) This example illustrates a DCSS beginning at location X'500000'; this is not intended as the only location for a DCSS.

```
DSSVFORT NAMESYS SYSNAME=DSSVFORT,                  See Note a
                 SYSSIZE=1344K,
                 SYSHRSG=(80,81,82,83,84,85,
                     86,87,88,89,90,91,92,93,
                     94,95,96,97,98,99,100),    See Note b
                 SYSPGNM=(1280-1615),           See Note c
                 VSYSADR=IGNORE,
                 SYSVOL=VMSRES,                 See Note d
                 SYSSTRT=(049,1)                See Note e
```

Notes to example:

a. The SYSNAME parameter specifies the name of the DCSS (DSSVFORT in this example). Change the name to whatever you desire.

b. The SYSHRSG parameter provides a list of consecutive segment numbers. (Specifying these numbers allows the segments to be shared by all users.) The compiler DCSS requires 21 segments. Compute the first segment number by dividing the starting address of the DCSS by 64K. In this example, the starting address is X'500000' or 5120K. Dividing this by 64K gives a starting segment number of 80.

c. The SYSPGNM parameter specifies the range of page numbers that comprise the DCSS. Compute the first page number by dividing the starting address of the DCSS by 4K. In this example, dividing X'500000' or 5120K by 4K gives a starting page number of 1280. A range of 336 pages must be specified.

d. The SYSVOL parameter gives the volume serial number of the CP-owned volume that holds the DCSS.

e. The SYSSTRT parameter gives the starting cylinder and page address (on the volume specified by the SYSVOL parameter) that holds the DCSS.

3. Assemble the new system name table DMKSNT and regenerate the CP nucleus by using the GENERATE EXEC procedure, as described in *VM/SP Planning Guide and Reference*, SC19-6201.

When the system programmer has completed the above steps, you can complete the installation of the VS FORTRAN Version 2 compiler as a DCSS. If this is an initial installation, you should continue with the installation process described in Chapter 4, "Installation under VM" on page 65.

If the VS FORTRAN Version 2 product is already installed and you are putting the compiler in a DCSS, perform the following steps:

1. Log on to VM with class E privileges (to allow execution of the SAVESYS command from the installation EXEC).

2. Define a virtual storage size that exceeds the starting address of the DCSS by at least 2 megabytes. For example, if the DCSS starting address is X'500000' (or 5 megabytes), a virtual storage of 7 megabytes is needed.

   Note: The 2 megabyte figure depends on machine configuration and is only an approximate value: you may need more space.

3. Link and access the work disk (this was the A-disk during initial installation) in read/write mode.

4. Link and access the product disk (this is the disk that contains your VS FORTRAN Version 2 libraries and modules) in read/write mode. Remember the file mode with which you access this disk; you will need to use it later.

5. Invoke the installation EXEC with the DCSS parameter, as follows:

   ```
   I5668806 DCSS
   ```

   Reply YES to the prompt asking if you are installing the VS FORTRAN Version 2 Compiler as a discontiguous saved segment, and be prepared to give the DCSS name(s).

6. Verify that the compiler was successfully installed in the DCSS by doing the following for each DCSS being installed:

   a. Redefine your virtual machine storage size to be the same as or less than the starting address of the DCSS, and re-IPL CMS.

   b. Re-access the work disk as the A-disk (as in step 3 above).

   c. Compile the sample program AFBIVP by issuing the following command:

      ```
      FORTVS2 AFBIVP
      ```

7. Once you are satisfied that the compiler has been successfully rebuilt, copy it to the product disk as follows:

   a. Access the product disk (as in step 4 on page 84 above) as mode fm (where fm is a file mode other than A, S, or Y).

   b. Issue the following commands:

```
COPY FORTVS2 MODULE A = = fm (REPLACE
COPY FORTVS2 MAP A = = fm (REPLACE
```

Changing the Execution-Time Option Defaults

The installation EXEC offers you an opportunity to change the IBM-supplied default values for execution-time options during the installation procedure. You can also change the defaults at any time after installation by using the following method.

Among the VS FORTRAN Version 2 library modules is a table, AFBVGPRM, that contains the global default values for execution-time options. The macro VSF2PARM can be used to create a new global table, AFBVGPRM, which supplies default options for all users of the VS FORTRAN Version 2 library. This section describes the steps in creating a new execution-time options table.

Step 1: Coding the VSF2PARM Macro Instruction

To change the execution-time option defaults, invoke XEDIT to edit the AFBVGPRM ASSEMBLE file:

```
XEDIT AFBVGPRM ASSEMBLE A
```

where A is the file mode of the work disk you used during installation.

If the execution-time option defaults have been changed before, there will already be a copy of AFBVGPRM ASSEMBLE on the work disk. Change the options specified on the VSF2PARM macro instruction in this file according to your site's needs.

If the defaults have never been changed before, the above command gives you a new file. Code the VSF2PARM macro instruction according to your site's needs.

For details on the contents and format of this macro instruction, see "VSF2PARM: For Changing Execution-Time Option Defaults" on page 104.

Step 2: Assembling the Module AFBVGPRM

After you have coded all the options you want to change on the VSF2PARM macro instruction in AFBVGPRM ASSEMBLE, issue the following commands to access the library containing the VSF2PARM macro definition and to assemble module AFBVGPRM:

```
GLOBAL MACLIB VSF2MAC
ASSEMBLE AFBVGPRM
```

Note: VSF2MAC is the name of the macro library generated during installation.

Step 3: Replacing the Object Module AFBVGPRM

Correct any coding errors in AFBVGPRM ASSEMBLE until it assembles without error. Then replace the original AFBVGPRM module in VSF2FORT TXTLIB by issuing the following commands:

```
TXTLIB DEL VSF2FORT AFBVGPRM
TXTLIB ADD VSF2FORT AFBVGPRM
```

Step 4: Rebuilding the Composite Module ABFVLBCM.

Because the module AFBVGPRM is a required module in the library composite module AFBVLBCM, your new AFBVGPRM module must be replaced in AFBVLBCM. Several other customization tasks involve modules that are also part of AFBVLBCM and require that it be rebuilt. You can rebuild AFBVLBCM now or you can wait until after you have completed all of your customization tasks. For instructions on how to rebuild AFBVLBCM, see "Building the Composite Modules" on page 93.

Extended Error Handling Facility

The **error option table** is a VS FORTRAN Version 2 library module that specifies what actions will be taken when an error occurs during execution of a FORTRAN program. For each execution-time error defined by VS FORTRAN Version 2, the table specifies:

- The number of times the error is allowed to occur before the user's program terminates.
- The maximum number of times the message may be printed.
- Whether or not the traceback map is to be printed with the message.
- Whether or not a user-written error exit routine is called.

The error numbers and values in the standard table you have received as part of the VS FORTRAN Version 2 product are documented in Chapter 10 of *VS FORTRAN Version 2: Language and Library Reference.*

You can customize this table in two ways:

1. Alter the IBM-supplied values in the standard error table. These are the values for the error conditions detected by VS FORTRAN Version 2 (numbers between 112 and 301).
2. Extend the table to include your own error numbers and associated actions. These would be error conditions that an individual program (rather than the VS FORTRAN Version 2 product) would recognize and would deal with by calling the supplied subroutine ERRMON to take the action specified in your table extension. These user-defined numbers can be in the range of 302 to 899.

The customization changes/extensions you make here affect the permanent copy of the error option table in the VS FORTRAN Version 2 Library. (Each individual FORTRAN program can also make execution-time changes to its copy of the table by calling the supplied subroutines ERRSET, ERRSAV, and ERRSTR. See *VS FORTRAN Version 2: Language and Library Reference* and *VS FORTRAN Version 2: Programming Guide* if you desire more information on this application programming feature.)

Changing or Adding Error Option Table Entries

Among the VS FORTRAN Version 2 library modules is a table, AFBUOPT, that contains the default values for the error options. The macro VSF2UOPT can be used to create a new set of error option defaults in AFBUOPT. This section describes the steps involved in creating a new AFBUOPT table.

Step 1: Coding the VSF2UOPT Macro Instructions

To change the error option defaults or to extend the error options table, first invoke XEDIT to edit the AFBUOPT ASSEMBLE file:

```
XEDIT AFBUOPT ASSEMBLE A
```

where A is the file mode of the work disk used to install the VS FORTRAN Version 2 product.

If the error options have been changed before, there is already a copy of AFBUOPT ASSEMBLE on the work disk. Change the options specified on the VSF2UOPT macro instruction in this file according to your site's needs.

If the defaults have never been changed before, the above command gives you a new file. Code the VSF2UOPT macro instruction according to your site's needs.

See the instructions under "VSF2UOPT: For Altering/Extending the Error Option Table" on page 108.

Step 2: Assembling the Module AFBUOPT

After you have coded your VSF2UOPT macro instruction in AFBUOPT ASSEMBLE, issue the following commands to access the library containing the VSF2UOPT macro definition and to assemble module AFBUOPT:

```
GLOBAL MACLIB VSF2MAC
ASSEMBLE AFBUOPT
```

Note: VSF2MAC is the name of the macro library generated during installation.

Step 3: Replacing the Object Module AFBUOPT

Correct any coding errors in AFBUOPT ASSEMBLE until it assembles without error. Then replace the original AFBUOPT module in VSF2FORT TXTLIB by issuing the following commands:

```
TXTLIB DEL VSF2FORT AFBUOPT
TXTLIB ADD VSF2FORT AFBUOPT
```

Step 4: Rebuilding the Composite Module AFBVLBCM

Because the module AFBUOPT is a required module in the library composite module AFBVLBCM, your new AFBUOPT module must be replaced in AFBVLBCM. Several other customization tasks involve modules that are also part of AFBVLBCM and require that it be rebuilt. You can rebuild AFBVLBCM now or you can wait until you have completed all of the customization tasks you plan to do, and rebuild AFBVLBCM once. For instructions on how to rebuild AFBVLBCM, see "Building the Composite Modules" on page 93.

Execution-Time Loading of Library Modules

When programmers create an executable program, they may choose to have all execution-time library modules (other than the mathematical routines) either made a part of their executable program along with the compiler-generated code (link mode), or loaded dynamically at execution time (load mode). Execution-time loading has several advantages. It reduces auxiliary storage requirements for executable programs and speeds link-editing.

Composite Modules

If programmers choose to have all the library modules included as part of their executable program (link mode), no further loading is required at execution time. If they choose execution-time loading (load mode), each module is loaded the first time it is used, unless it has been previously loaded. Because execution-time performance suffers if large numbers of library modules are individually loaded, the modules to be loaded at execution time are combined into composite modules. The two composite modules reside in VSF2LOAD.

AFBVLBCM contains nonreentrant library modules, including the library common work area and various system services routines.

AFBVRENC contains all the loadable reentrant modules. Copies of this composite module may be placed in one or more discontiguous saved segments. For more information, see "AFBVRENC as a Discontiguous Saved Segment" on page 91.

As part of its initialization procedure in load mode, the execution-time library loads the composite modules listed above. The only modules that need to be loaded separately after initialization are those not contained in the composite modules.

At any time after installation, you may add or delete library modules from the composite load modules to further tune your system. For example, if direct access and keyed access are not normally used at your site, you may choose not to place the modules that perform these functions in the composite modules. This reduces the size of these modules. The direct access and keyed access I/O modules would then have to be loaded individually should they ever be needed.

Selecting Load Mode or Link Mode

After installation of the VS FORTRAN Version 2 library, you must update your site's operational procedures to specify the libraries needed for use in load mode or link mode. To select the mode you want, you can provide an EXEC to issue the appropriate GLOBAL commands for either load mode or link mode, as described below.

Specifying Libraries in Load Mode

Specify the VSF2FORT TXTLIB but not the VSF2LINK TXTLIB in the CMS GLOBAL command for use by the LOAD command:

```
GLOBAL TXTLIB VSF2FORT CMSLIB
```

or specify this library in a CMS FILEDEF command for use by the LKED command:

```
FILEDEF SYSLIB DISK VSF2FORT TXTLIB fm
```

where fm is the file mode of the product disk.

To execute a program that has been created for execution in load mode, make VSF2LOAD available for the execution step. Use the following command:

```
GLOBAL LOADLIB VSF2LOAD
```

Specifying Libraries in Link Mode

For operation in link mode, concatenate VSF2LINK ahead of VSF2FORT for use by the LOAD command in CMS when it includes VS FORTRAN Version 2 library modules.

Specify the TXTLIBs VSF2LINK and VSF2FORT in the CMS GLOBAL command for use by the LOAD command:

```
GLOBAL TXTLIB VSF2LINK VSF2FORT CMSLIB
```

You cannot use the LKED command to create an executable program that operates in link mode.

A program created for execution in link mode does not require any VS FORTRAN Version 2 libraries at execution time unless Interactive Debug is used.

Deciding What to Include in Composite Modules

Composite modules AFBVLBCM and AFBVRENC may be updated to include only the library routines commonly used at your installation. The choice to include or not to include a module in the composite module is based upon the following considerations:

- Because AFBVLBCM contains the nonreentrant modules, it must be loaded into your virtual machine for each execution of a VS FORTRAN Version 2 program. Including all possible nonreentrant modules may cause the storage required for the program to be larger than necessary.
- If AFBVRENC is not in a discontiguous saved segment (DCSS), it must be loaded into your virtual machine. Including all possible reentrant modules may cause the storage required for the program to be larger than necessary.

- If AFBVRENC is in a DCSS, including a large number of the reentrant modules in the composite module has no effect upon the storage required for the program. However, the larger AFBVRENC does require additional virtual storage for the DCSS.

Each library module not in the applicable composite module is loaded from the VSF2LOAD library when the module is first referenced during execution.

AFBVRENC as a Discontiguous Saved Segment

If you decide to put the composite module AFBVRENC in one or more discontiguous saved segments (DCSSs), your VM system programmer must define the DCSSs to VM. To define the DCSS, the system programmer must obtain from you the following information for **each** DCSS:

- The name of the DCSS. This may be any name you wish, including the name AFBVRENC.

  However if, in the future, you expect to have more than one release of VS FORTRAN Version 2 installed on your system at the same time, you should **not** use the name AFBVRENC for a DCSS, nor should you use any other DCSS name that you will use with another release of the product.

- The number of segments in the DCSS. This depends on the size of the composite module AFBVRENC, which in turn depends on which of the optional modules you choose to include in the composite module. To determine the size of the composite module AFBVRENC and the number of segments needed, do the following:

  1. Determine which modules to include in AFBVRENC. (See Figure 42 on page 116 for a list of the optional modules and what they do.)

  2. Calculate the space required for the modules to be included: Take the total space needed for the required modules (see Figure 41 on page 116) and add to it the space required for each optional module you want to include (see Figure 42 on page 116).

  3. Now determine the number of 64K segments needed by taking the total from the last step and dividing it by 64K. The dividend rounded up to the next integer is the number of segments you will need (either 1 or 2).

- The starting address of the DCSS. With the assistance of your VM system programmer, choose this address using the following guidelines:

  - The address should be at least as large as the virtual machine size of any of the VS FORTRAN Version 2 users you expect to use this DCSS.

  - The address should not be unnecessarily high; if it is, storage is wasted for unreferenced CP segment table entries. However, in order to accommodate users with different virtual machine sizes, you can create several DCSSs, each with a different starting address.

- The address should not allow this DCSS to overlap any other saved segment that may be used at the same time. For example, if you start execution from an ISPF panel, the DCSS should not overlap any saved segments that contain parts of ISPF. Your VM system programmer can help you determine a potential overlap.

After the VM system programmer has defined the segments, you can proceed with installing the composite AFBVRENC in a DCSS.

Defining the AFBVRENC DCSS to VM

Although a DCSS need not be shared among all users, the composite module AFBVRENC is reentrant and can be shared. Your VM system programmer must complete the following steps before the composite module AFBVRENC can be installed in a discontiguous saved segment (DCSS):

1. Allocate permanent space on a CP-owned DASD volume to contain the DCSS. The composite module AFBVRENC requires either 16 4K pages (one 64K segment) or 32 4K pages (two 64K segments), depending on which modules are included in the composite module. (For more information on the amount of disk space needed, refer to *VM/SP Planning Guide and Reference*, SC19-6201.)

2. Define the DCSS by adding a NAMESYS macro instruction to your installation's DMKSNT ASSEMBLE module. (See *VM/SP Planning Guide and Reference*, SC19-6201, and *VM/SP System Programmer's Guide*, SC19-6203.) If more than one DCSS is to be built to hold copies of the composite module AFBVRENC at different addresses, then there must be a NAMESYS macro instruction defining each DCSS. The following example of the NAMESYS macro instruction defines a DCSS named FTNLIB10. The sample numbers given illustrate a possible set of numbers and are not intended as the only location or size for a DCSS.

```
FTNLIB10 NAMESYS SYSNAME=FTNLIB10,          See Note a
                 SYSSIZE=128K,
                 SYSHRSG=(48,49),           See Note b
                 SYSPGNM=(768-799),         See Note c
                 VSYSADR=IGNORE,
                 SYSVOL=VMSRES,             See Note d
                 SYSSTRT=(072,1)            See Note e
```

Notes to example:

a. The SYSNAME parameter specifies the name of the DCSS (FTNLIB10 in this example). Change the name to whatever you desire.

b. The SYSHRSG parameter provides a list of consecutive segment numbers. (Specifying these numbers allows the segments to be shared by all users.) Compute the first segment number by dividing the starting address of the DCSS by 64K. In this example, the starting address is X'300000' or 3072K. Dividing this by 64K gives a starting segment number of 48.

c. The SYSPGNM parameter specifies the range of page numbers that comprise the DCSS. Compute the first page number by dividing the starting address of the DCSS by 4K. In this example, dividing X'300000' or 3072K by 4K gives a starting page number of 768. A range of 32 pages is specified here to correspond to the 2 segments.

d. The SYSVOL parameter gives the serial number of the CP-owned volume which will hold the DCSS.

e. The SYSSTRT parameter gives the starting cylinder and page address (on the volume specified by the SYSVOL parameter) which will hold the DCSS.

---

3. Assemble the new system name table (DMKSNT) and regenerate the CP nucleus by using the GENERATE EXEC procedure, as described in *VM/SP Installation Guide*, SC24-5237.

Building the Composite Modules

If you are doing an initial installation, you should continue where you left off in Chapter 4, "Installation Under VM."

If you have dumped the installation work disk to tape and no longer have access to the installation EXECs, you can use the manual procedure described in Appendix B, "Another Way to Build Library Composite Modules (VM)" on page 115.

Otherwise, if the product has already been installed, and you just want to put the composite module AFBVRENC in a DCSS or regenerate the composite modules AFBVLBCM or AFBVRENC, you can use the following procedure.

1. Log on to VM. If you are installing in a DCSS, you must have class E privileges (to allow execution of the SAVESYS command used by the installation EXEC).

   (Skip to step 3 if not generating a DCSS.)

2. Define a virtual storage size that exceeds the starting address of the DCSS by at least 1 megabyte. For example, if the DCSS starting address is X'500000' (or 5 megabytes), a virtual storage of at least 6 megabytes is needed.

   Note: The 1 megabyte figure depends on CMS storage utilization and is only an approximate value; you may need more.)

3. Link to and access the product work disk (A-disk during initial installation) in read/write status as file mode A.

4. Link to and access the product disk (disk containing libraries and modules at the end of initial installation) in read/write status as some file mode other than A, S, or Y. Remember the file mode you use, because you will be asked for it later.

5. Invoke the library installation EXEC with the COMPOSITE parameter, as follows:

   ```
   I5668805 COMPOSITE
   ```

   You will be asked for the names of the product macro library, principal text library, and load library. In response to the prompts, indicate whether you want to modify the list of modules that are in the composite modules.

6. You now have a new copy of the load library on the product work disk, the A-disk. This load library contains your tailored composite modules. It also refers to any DCSSs that were built.

   Verify that the library works properly by running a sample FORTRAN program using the following series of commands:

   ```
   FORTVS2 AFBIVP

   GLOBAL TXTLIB VSF2FORT CMSLIB

   GLOBAL LOADLIB VSF2LOAD

   LOAD AFBIVP (NOAUTO START
   ```

   If you built one or more DCSSs, follow these additional steps for each DCSS:

   a. Redefine your virtual machine storage size to be the same as or less than the starting address of the DCSS, and re-IPL CMS.

   b. Re-access the product work disk as the A-disk, as in step 3 on page 93 above.

   c. Re-access the product disk, as in step 4 above.

   d. Run a FORTRAN program using the new library in load mode as shown above.

7. Once you have verified that the new load library works, copy it from the product work disk to the product disk as follows (fm is the file mode of the product disk):

   ```
   ERASE VSF2LOAD LOADLIB fm

   FILEDEF SYSIN DUMMY

   LOADLIB COPY VSF2LOAD LOADLIB A VSF2LOAD LOADLIB fm

   ERASE VSF2LOAD LOADLIB A
   ```

Appendix A. Macros for Customizing VS FORTRAN Version 2

To help you customize your VS FORTRAN Version 2 product, several macros are included in the materials you have received:

VSF2COM to change the compiler options default values

VSF2LIB to change I/O unit number default values

VSF2PARM to change the execution-time options default values

VSF2UOPT to change or expand the execution-time error message table

VSF2AMTB to create or replace an auxiliary error option table

You need only concern yourself with these macros if you are not satisfied with the standard default values provided with the VS FORTRAN Version 2 product and you wish to change them.

Guidelines for Coding Macro Instructions

When coding any of these macro instructions, follow these guidelines:

- Column 1 must be blank.
- The macro name may appear anywhere before column 72 but must precede the operands by at least one blank. The suggested starting position is column 10.
- The operands are separated by commas, with no intervening blanks, and may be continued on any number of lines as long as column 72 contains a nonblank character and the data on the following line begins in column 16.
- You do not need to code all keyword parameters. Code only those whose defaults you wish to change.
- A comma must follow the last operand on a line when a continuation line follows.

VSF2COM: For Changing Compiler Option Defaults

The compiler macro instruction VSF2COM enables you to change the IBM-shipped default values for most of the compiler options.

When you code the VSF2COM macro instruction, you establish system defaults for the compiler options that can be specified by the individual user. These defaults are assumed if the parameters are not overridden by the user.

The following table shows pairs of options that create an error message, if both options are used. The VSF2COM macro does not allow the installation or customization process to continue if any of these conditions occurs.

FIPS = F or FIPS = S	SORCIN = FREE
FIPS = F or FIPS = S	FLAG = W | E | S
FIPS = F or FIPS = S	LANGLVL = 66
NAME = name	LANGLVL = 77
SRCFLG = SRCFLG	SORLIST = NOSOURCE
SYM = SYM	PUNCH = NODECK and OBJPROG = NOOBJECT
TEST = TEST	NAME = name
TEST = TEST	OBJPROG = NOOBJECT
TEST = TEST	OPTIMIZ = 1 | 2 | 3

Format of the Macro Instruction

The macro instruction has the following syntax:

```
VSF2COM SYSTEM=OS/VS|CMS
        [,ADJ=IL(DIM)|IL(NODIM)]
        [,CHARLEN=number|500]
        [,DATE=MDY|YMD]
        [,FIPS=S|F|NOFIPS]
        [,FLAG=I|W|E|S]
        [,IGNORE=DISABLED|ENABLED]
        [,INSTERR=NOLIST|LIST]
        [,LANGLVL=66|77]
        [,LINECNT=number|60]
        [,NAME=name|MAIN]
        [,OBJATTR=RENT|NORENT]
        [,OBJID=GOSTMT|NOGOSTMT]
        [,OBJLIST=LIST|NOLIST]
        [,OBJPROG=OBJECT|NOOBJECT]
        [,OPTIMIZ=0|1|2|3|NOOPTIMIZE]
        [,PUNCH=DECK|NODECK]
        [,SORCIN=FREE|FIXED]
        [,SORLIST=SOURCE|NOSOURCE]
        [,SORTERM=TERMINAL|NOTERMINAL]
        [,SORXREF=XREF|NOXREF]
        [,SRCFLG=SRCFLG|NOSRCFLG]
        [,STORMAP=MAP|NOMAP]
        [,SXM=SXM|NOSXM]
        [,SYM=SYM|NOSYM]
        [,SYMDUMP=SDUMP|NOSDUMP]
        [,TEST=TEST|NOTEST]
        [,TRMFLG=TRMFLG|NOTRMFLG]
```

The IBM-supplied *default parameters* are underlined in the following parameter lists. If a given operand is not coded in the VSF2COM macro instruction, these parameters are assumed when the macro is assembled, with the exception of the SYSTEM operand, which must always be specified if you code the macro instruction.

Note: There are no IBM-supplied defaults for the compiler options AUTODBL, CI, DC, DIRECTIVE, and VECTOR, and therefore these options can *not* be specified on the VSF2COM macro instruction.

The options that can be specified on the VSF2COM macro instruction are as follows:

SYSTEM = OS/VS | CMS
specifies the system on which VS FORTRAN Version 2 will run.

If you accept the IBM-supplied defaults for the compiler options, the SYSTEM option is established during the installation process. If you code the VSF2COM macro instruction to change any compiler options, you must also specify the SYSTEM option to indicate the system on which you are installing VS FORTRAN Version 2.

There is no default for this option.

ADJ = IL(DIM) | IL(NODIM)

specifies whether the code for adjustable-dimensioned arrays is to be placed inline—IL(DIM), or done via library call—IL(NODIM). Inline placement may result in faster execution but does not check for dimensioning errors; the library call may result in slower execution but does check for such errors. You may also specify IL(NODIM) as NOIL.

CHARLEN = number | 500

specifies the maximum length for any character variable, character array element, or character function. Specify number as an integer from 1 to 32767. Within a program unit, you cannot specify a length for a character variable, array element. or function greater than the CHARLEN specified.

DATE = MDY | YMD

specifies the format of the date to be printed by the compiler.

YMD

specifies that DATE is to be in the format yymmdd (y = year, m = month, d = day).

MDY

specifies that DATE is to be in the format mmddyy (m = month, d = day, y = year).

FIPS = S | F | NOFIPS

specifies whether or not standard language flagging is to be performed, and, if it is, the standard language flagging level:

S

specifies subset standard language flagging.

F

specifies full standard language flagging.

NOFIPS

specifies no standard language flagging.

Items not defined in the current American National Standard are flagged. Flagging is valuable only if you want to write a program that conforms to the American National Standard for FORTRAN implemented in LANGLVL(77). If you specify LANGLVL(66) and FIPS flagging at either level, the FIPS option is ignored.

FLAG = I | W | E | S

specifies the level of diagnostic messages to be written.

I

specifies that all messages, including informational messages (return code 0 or higher), are to be written.

W

specifies that warning messages (return code 4 or higher) are to be written.

E

specifies that error messages (return code 8 or higher) are to be written.

S

specifies that severe error messages (return code 12 or higher) are to be written.

FLAG allows you to suppress messages that are below the level desired. Thus, if you want to suppress all messages that are warning or informational, specify FLAG(E).

IGNORE = DISABLED | ENABLED

specifies whether or not the IGNORE directive will be made available to the user. Users will not have access to the IGNORE directive if the IGNORE installation option is DISABLED. Note that there is no corresponding compiler option.

INSTERR = NOLIST | LIST

specifies whether or not to list the messages that could be issued by the VSF2COM macro. If LIST is chosen, all possible messages are listed and no object file is produced. When LIST is specified, a return code of 16 is generated by the assembler.

LANGLVL = 66 | 77

specifies the language level at which the input source program is written.

66

specifies the old FORTRAN level—the 1966 language standard plus IBM extensions.

77

specifies the current FORTRAN level—the 1978 language standard plus IBM extensions.

LINECNT = number | 60

specifies the maximum number of lines on each page of the printed source listing. Specify number as an integer from 7 to 32765. The advantage of using a large LINECNT number is that there are fewer page headings to look through if you are using only a terminal. Your output, if printed, will run together from page to page without a break.

NAME = name | MAIN

can only be specified when LANGLVL(66) is specified. It specifies the name that is generated on the output and the name of the CSECT generated in the object module. It only applies to main programs.

OBJATTR = RENT | NORENT

specifies whether or not reentrant object code is to be generated by the compiler.

OBJID = GOSTMT | NOGOSTMT

specifies whether or not internal sequence numbers (for traceback purposes) are to be generated for a call sequence to a subprogram.

OBJLIST = LIST | NOLIST
specifies whether or not the object module listing is to be produced.

OBJPROG = OBJECT | NOOBJECT
specifies whether or not the object module is to be produced. If OBJECT is specified, it requires an object output file.

OPTIMIZ = 0 | 1 | 2 | 3 | NOOPTIMIZE
specifies the optimizing level to be used during compilation.

0 or NOOPTIMIZE
specifies no optimization.

1
specifies register and branch optimization.

2
specifies partial code-movement optimization, code movement that can not introduce logic changes into the program.

3
specifies full code-movement optimization, which can possibly introduce logic changes into the program.

If you are debugging your program, it is advisable to use NOOPTIMIZE. To create more efficient code and, therefore, a shorter execution time, but usually a longer compile time, use OPTIMIZE(2) or (3).

PUNCH = DECK | NODECK
specifies whether or not the object module is to be produced in card-image format. If DECK is specified, it requires a punch output file.

SORCIN = FREE | FIXED
specifies whether the input source program is to be in free format or in fixed format.

SORLIST = SOURCE | NOSOURCE
specifies whether or not the source listing is to be produced.

SORTERM = TERMINAL | NOTERMINAL
specifies whether or not error messages and compiler diagnostics are to be written on the terminal or a SYSTERM data set.

Note: If your users are compiling in a batch environment and are not using a SYSTERM data set, specify NOTERMINAL to avoid messages about having no terminal online.

SORXREF = XREF | NOXREF
specifies whether or not a cross-reference listing of all variables and labels in the source program is to be produced.

SRCFLG = SRCFLG | NOSRCFLG
allows error diagnostics to be inserted into the source listing immediately following the statement in error.

STORMAP = MAP | NOMAP
specifies whether or not a table of source program names and statement labels is to be written.

SXM = SXM | NOSXM
improves readability of XREF or map listing output at a terminal. SXM formats listing output for an 80-character wide terminal screen; NOSXM formats listing output for a printer.

SYM = SYM | NOSYM
invokes the production of SYM cards in the object text file. The SYM cards contain location information for variables within a FORTRAN program. SYM cards may be useful to MVS users.

SYMDUMP = SDUMP | NOSDUMP
specifies whether or not symbol table information is to be generated in the object module and in the object module listing.

TEST = TEST | NOTEST
specifies whether or not to create input for VS FORTRAN Version 2 Interactive Debug and symbol table information. TEST overrides any optimization level above OPTIMIZE(0), and adds execution-time overhead.

TRMFLG = TRMFLG | NOTRMFLG
presents the statement in error and the diagnostic message together, whenever possible, on your terminal. Specify the NOTRMFLG option if you are running batch jobs on MVS.

Note: If your users are compiling in a batch environment and are not using a SYSTERM data set, specify NOTRMFLG to avoid messages about having no terminal online.

VSF2LIB: For Changing I/O Unit Number Defaults

The VSF2LIB macro instruction specifies input/output information for the VS FORTRAN Version 2 Library. The execution-time input/output routines of the VS FORTRAN Version 2 Library require information on the maximum number of logical input/output units that are available to the user programs. The UNTABLE operand provides this information.

These routines also require that defaults be established for the logical input/output units to be used for READ statements, PUNCH statements, error messages, and dumps. The ONLNRD, ONLNPCH, and OBJERR operands establish default data set reference numbers. The FORTRAN programmer using the library may accept these defaults and need not supply a data set definition (DD) statement.

Format of the Macro Instruction

The macro instruction has the following syntax:

```
VSF2LIB [DECIMAL=PERIOD|COMMA]
        [,OBJERR=unit|06]
        [,ONLNPCH=unit|07]
        [,ONLNRD=unit|05]
        [,UNTABLE=number|99]
```

All VSF2LIB keyword operands are optional. If any operand is omitted, the default value for that operand is used. The IBM-supplied *default parameters* are underlined in the following parameter lists. The VSF2LIB macro instruction keyword operands and their parameters are:

DECIMAL = PERIOD | COMMA
specifies the character to be used as the decimal indicator in printed output.

OBJERR = unit | 06
specifies the logical I/O unit number to be used with execution-time error messages, with any WRITE statement specifying an installation-dependent form of the unit, and with the PRINT statement. The number specified must be a two-digit number between 00 and the value specified for the UNTABLE operand, and must not be the same as the number specified for ONLNRD or ONLNPCH.

ONLNPCH = unit | 07
specifies, for LANGLVL(66) only, the logical I/O unit number to be used with the PUNCH statement to output data to the card punch. The number specified must be a two-digit number between 00 and the value specified for the UNTABLE operand, and must not be the same as the number specified for ONLNRD or OBJERR.

ONLNRD = unit | 05
specifies the logical I/O unit number to be used with any READ statement specifying an installation-dependent form of the unit. The number specified must be a two-digit number between 00 and the value specified for the UNTABLE operand, and must not be the same as the number specified for either ONLNPCH or OBJERR.

UNTABLE = **number | 99**

specifies the largest unit number you can include in a VS FORTRAN Version 2 program. Because the unit numbers begin with 0, the UNTABLE number plus 1 indicates how many units are allowed.

Specify number as a 2-digit integer from 08 to 99. The storage required for a unit table using UNTABLE = 08 is 160 bytes. Each additional unit added to the table adds 16 bytes of storage. If you use the default of 99 (UNTABLE = 99), your table will occupy 1616 bytes of storage.

VSF2PARM: For Changing Execution-Time Option Defaults

The macro VSF2PARM establishes a set of execution-time option default values that differ from those installed with the VS FORTRAN Version 2 product. As described below, this macro can be used to replace the global table, AFBVGPRM, which supplies default options for all users of the VS FORTRAN Version 2 library. Or, as described in *VS FORTRAN Version 2: Programming Guide*, the macro can be used to create a local table, AFBVLPRM, which supplies options that are used only for a specific program.

Format of the Macro Instruction

The macro instruction has the following syntax:

```
VSF2PARM [option,...]SCOPE=GLOBAL
```

SCOPE = GLOBAL creates a global execution-time options table, which supplies installation-wide execution-time options. SCOPE must be specified.

The IBM-supplied *default parameters* are underlined in the following parameter lists. The following are the options that may be specified in the macro instruction:

ABSDUMP | NOABSDUMP
specifies whether or not the post-abend symbolic dump information is printed.

ABSDUMP
causes the post-abend symbolic dump information to be printed in the event of an abnormal termination.

NOABSDUMP
suppresses the printing of the post-abend symbolic dump information

DEBUG | NODEBUG
specifies whether or not Interactive Debug will be invoked.

DEBUG
causes Interactive Debug to be invoked.

NODEBUG
does not cause Interactive Debug to be invoked.

DEBUNIT | NODEBUNIT
specifies whether or not Interactive Debug units will be used in conjunction with the DEBUG option.

DEBUNIT
passes a list of FORTRAN unit numbers to the execution environment. Interactive Debug will use the list for DEBUG input/output. The format of the option is

```
DEBUNIT(s1[,s2,...])
```

where s is a single unit number or a range of unit numbers. A range of unit numbers is expressed as us-ue, where both us and ue are unit numbers, and the ending unit number, ue, is not less than the starting number, us.

The unit numbers specified must be one- or two-digit numbers within the range of numbers allowed on your system, as specified on the UNTABLE operand of the VSF2LIB macro instruction. See page 103 for information on the UNTABLE operand.

NODEBUNIT
provides no list of unit numbers for use by Interactive Debug.

IOINIT | NOIOINIT
specifies whether or not the normal initialization for I/O processing will occur during initialization of the execution-time environment.

IOINIT
causes the normal initialization for I/O processing to occur during initialization of the execution-time environment.

NOIOINIT
suppresses initialization for I/O processing. This means:

- The error message unit will not be opened during initialization of the execution-time environment. However, this does not prevent I/O from occurring on this or on any other unit. (Such I/O may fail if proper DD statements or FILEDEF statements are not given.)

- Under VM, the CMS FILEDEF commands for the reader, printer, and punch will not be issued. Should subsequent I/O be directed to these units, the default FILEDEFs that are provided by CMS, not by VS FORTRAN, will be used.

SPIE | NOSPIE
specifies whether or not the SPIE (or ESPIE) macro instruction will be executed.

SPIE
causes a SPIE (or ESPIE) macro instruction to be executed during initialization of the execution-time environment so that VS FORTRAN execution-time environment can take control in the event of program interrupts.

NOSPIE
suppresses execution of the SPIE (or ESPIE) macro instruction. We do not recommend choosing NOSPIE if you are using the DEBUG option. If you specify NOSPIE, various execution-time functions that depend on a return of control after a program interrupt are unavailable. These include:

- The messages and corrective action for a floating-point overflow

- The messages and corrective action for a floating-point underflow interrupt (unless the underflow is to be handled by the hardware based upon the XUFLOW option)
- The messages and corrective action for a floating-point or fixed-point divide exception
- The simulation of extended precision floating-point operations on processors that do not have these instructions
- The realignment of vector operands that are not on the required storage boundaries and the re-execution of the failed instruction

Instead of the corrective action, abnormal termination results. In this case, the STAE or NOSTAE option that is in effect governs whether or not the VS FORTRAN execution-time environment gains control at the time of the abend.

STAE | NOSTAE
specifies whether or not a STAE (or ESTAE) macro instruction will be executed during initialization.

STAE
causes a STAE (or ESTAE) macro instruction to be executed during the initialization of the execution-time environment so that the VS FORTRAN execution-time environment can take control in the event of abnormal termination.

NOSTAE
suppresses execution of the STAE (or ESTAE) macro instruction. We do not recommend choosing NOSTAE if you are using the DEBUG option. If NOSTAE is specified, abnormal termination is handled by the operating system rather than by the VS FORTRAN execution-time environment. In this case:

- Message AFB240I, which shows the PSW and register contents at the time of the abend, is not printed. However, this information will be provided by the operating system.
- The indication of which FORTRAN statement caused the failure will not be printed.
- The traceback of the routines will not be printed.
- The post-abend symbolic dump will not be printed even with the option ABSDUMP in effect.
- Certain exceptional conditions handled by the execution-time environment or by the debugging device cause system abends rather than VS FORTRAN messages. For example, some errors that occur during execution of an OPEN statement result in a system abend rather than the printing of message AFB219I, which allows possible continuation of program execution.

- If the QUIT command is used in an attention exit to terminate a program during a TSO debugging session, a user ABEND 500 occurs instead of the normal termination of the execution-time environment.

- An MTF subtask that terminates unexpectedly causes a user ABEND 922 in the main task rather than message AFB922I.

XUFLOW | NOXUFLOW

specifies whether or not an exponent underflow will cause a program interrupt.

XUFLOW

allows an exponent underflow to cause a program interrupt, followed by a message from the VS FORTRAN Version 2 Library, followed by a standard fixup.

NOXUFLOW

suppresses the program interrupt caused by an exponent underflow. The hardware sets the result to zero.

VSF2UOPT: For Altering/Extending the Error Option Table

This macro can be used to change the defaults for existing entries in the error option table, or to create new entries. You code one or more VSF2UOPT macro instructions, followed by an END statement. In all cases, you **must** code at least the first macro instruction and the END statement.

Format of the Macro Instruction

The first macro instruction has the following syntax:

```
VSF2UOPT [ADDNTRY=n]
```

ADDNTRY = n
is a positive integer specifying the number of additional error message numbers to be added to those supplied by IBM. Include this parameter if you want to add your own new message numbers to the error option table. Additional error message numbers will begin at 302 and continue sequentially, up to a maximum of 899. Thus, the maximum value for ADDNTRY is 598.

If you want to add additional error messages without modifying any existing entries in the error option table, follow your VSF2UOPT instruction with an END statement.

If you want to modify defaults for IBM-supplied message numbers, but you do not want to add your own new message numbers, you must still code the first VSF2UOPT instruction. Then code one or more optional VSF2UOPT macro instructions followed by an END statement.

Format of the Optional Macro Instruction

If you want to modify the default values in the error option table, for either IBM-supplied message numbers or your own additional message numbers, you must also code one or more of the following VSF2UOPT macro instructions. Follow your final macro instruction with an END statement. The optional macro instructions have the following syntax:

```
VSF2UOPT MSGNO=(ermsno,qty)
         [,ALLOW=errs]
         [,PRINT=prmsg]
         [,IOERR={YES|NO}]
         [,MODENT={YES|NO}]
         [,PRTBUF={YES|NO}]
         [,INFOMSG={YES|NO}]
         [,TRACBAK={YES|NO}]
         [,USREXIT=exitname]
```

ALLOW = errs
specifies the number of times the error may occur before the program is terminated.

errs
specifies the number of errors allowed. To specify an exact number of errors allowed, errs must be a positive integer with a maximum of

255. A zero, or any number greater than 255, means the error can occur an unlimited number of times.

Be aware that altering an error option table entry to allow "unlimited" error occurrence may cause a program to loop indefinitely.

If the message number is an IBM-supplied message number, the default value for this parameter is listed in the chapter "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2: Language and Library Reference*. If the message number has been added by your site, the default value is 10.

INFOMSG = YES | NO
specifies whether the message is an informational or an error message.

YES
specifies that the message is informational only. In this case:

- No user error exit is taken.
- The value of ALLOW is ignored. Execution will not terminate, even if it reaches the designated number of errors allowed.
- The error summary printed after termination of your program does not include a count of the number of times the condition occurred.

NO
specifies that the message is an error message.

For the default for this parameter, see "Default Values for the Optional Macro Instruction Parameters" on page 112.

IOERR = YES | NO
specifies whether or not this error message represents an I/O error for which error counting is to be suppressed when an ERR or IOSTAT parameter is given on the I/O statement.

YES
specifies that if an ERR or IOSTAT parameter is given, the occurrence of the error is not to be counted toward the maximum number specified by the ALLOW parameter above. This should be specified only for those errors, listed in *VS FORTRAN Version 2: Language and Library Reference* for which the ERR and IOSTAT parameters are honored.

NO
specifies that the error occurrence is to be counted toward the maximum number of errors allowed.

MODENT = YES | NO
specifies whether or not the ERRSET subroutine may be used to modify the error option table entry for this message.

YES
specifies that the entry may be modified.

NO
specifies that the entry may not be modified.

If you code a YES value for an IBM-supplied error message whose default is NO, and you subsequently modify this entry using the ERRSET subroutine, you may receive undesirable results. Check the chapter "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2: Language and Library Reference*, to find out which message numbers have a "Modifiable Entry" value of NO.

For the default for this parameter, see "Default Values for the Optional Macro Instruction Parameters" on page 112.

MSGNO = (ermsno, qty)
specifies which error message numbers are affected by the default changes.

ermsno
specifies the first error message number in a series of consecutive numbers.

qty
specifies the number of consecutive error message numbers, beginning with **ermsno**. If the defaults for only one error message number are to be changed, then **qty**, the preceding comma, and the surrounding parentheses may be omitted.

For example, if the parameter is coded MSGNO=(153,4), then the defaults for four error messages, beginning with number 153, are changed as specified by the remaining parameters. Thus, the defaults for messages 153 through 156 are changed.

PRINT = prmsg
specifies the number of times the error message is to be printed. Subsequent occurrences of the error do not cause the message to be printed again.

prmsg
specifies the number of times the message is to be printed. To specify an exact number of times printed, prmsg must be a positive integer, with a maximum of 254. A zero means the message will not be printed. Specifying 255 means the message can be printed an unlimited number of times.

If the message number is an IBM-supplied message number, the default value for this parameter is listed in the chapter "Extended Error-Handling Subroutines and Error Option Table" in *VS FORTRAN Version 2: Language and Library Reference*. If the message number has been added by your site, the default value is 5.

PRTBUF = YES | NO

specifies whether or not the I/O buffer is to be printed following certain I/O errors.

YES

specifies that the contents of the buffer are to be printed.

NO

specifies that the contents of the buffer are not to be printed.

This option applies only to IBM-supplied error messages. Do not code YES unless the IBM-supplied default for this error message number already allows the buffer to be printed. Check the chapter "Extended Error-Handling Subroutines and Error Option Table" in *VS FORTRAN Version 2: Language and Library Reference*, to find out which message numbers have a "Print Buffer" value of YES.

For the default for this parameter, see "Default Values for the Optional Macro Instruction Parameters" on page 112.

TRACBAK = YES | NO

specifies whether or not a module traceback listing is to be printed following the error message.

YES

specifies that the traceback listing is to be printed.

NO

specifies that the traceback listing is not to be printed.

For the default for this parameter, see "Default Values for the Optional Macro Instruction Parameters" on page 112.

USREXIT = exitname

specifies the user error exit routine that is invoked following the printing of the error message.

exitname

specifies the entry point name of the user error exit routine. If the routine is specified here, instead of being specified as a parameter passed to the ERRSET subroutine, the routine is invoked when the error occurs for any user. In this case, the routine will be invoked, regardless of whether the ERRSET routine was used or not. (However, unless a MODENT value of NO is in effect, programs can still call ERRSET dynamically to specify their own exit routine instead of the one specified by USREXIT.)

For programs operating in link mode, the user error exit routine must be link-edited with all users' programs. To make the user error exit routine available to users who operate in load mode, the routine must be included in the composite module AFBVLBCM.

If the user error exit routine must communicate with the VS FORTRAN Version 2 program in which the error was detected, it must do so using a dynamic common area, not a static one.

Default Values for the Optional Macro Instruction Parameters: The default values for five parameters on the optional VSF2UOPT macro instruction vary according to two conditions. These conditions and the default values are as follows:

1. The message number is an IBM-supplied message number, and *none* of the default values for IOERR, MODENT, PRTBUF, INFOMSG, TRACBAK are changed.

   For this condition, the default values are those found in the chapter "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2: Language and Library Reference.*

2. The message number is an IBM-supplied message number, and the default values for one or more of the following *are* changed: IOERR, MODENT, PRTBUF, INFOMSG, or TRACBAK.

   Or, the message number has been added by your installation.

   For this condition, the default values for the unspecified parameters are as follows:

Parameter	*Default*
IOERR	NO
MODENT	YES
PRTBUF	NO
INFOMSG	NO
TRACBAK	YES

VSF2AMTB: For Creating or Replacing an Auxiliary Error Option Table

VS FORTRAN Version 2 supplies a macro that can be used to create a new or a replacement auxiliary error option table. An auxiliary error option table is a table used by a product other than VS FORTRAN Version 2 to make use of VS FORTRAN Version 2's error handling facility. Auxiliary error option tables should not be confused with the regular error option table in VS FORTRAN Version 2 (which is used by the VS FORTRAN Version 2 extended error handling facility).

The macro supplied by VS FORTRAN Version 2 is designed to create an entire auxiliary error option table. This table can be either a new table, or a replacement for an existing table.

To use this macro successfully, you must have the following auxiliary product information (information not supplied by VS FORTRAN Version 2):

- Error number ranges of the table (starting and ending numbers)

  Note: The numbers 10000 to 19999 are reserved for customer use.

- Prefix for the table name (---UOPT)
- Any table values other than VS FORTRAN Version 2-supplied defaults
- Location where assembled table is to be stored

The VSF2AMTB macro instruction has two forms:

1. It defines the name and scope of the table.
2. It defines table contents for individual error number entries.

You must specify the first form of the macro instruction **once**, then follow it with the second form of the macro instruction, repeated as many times as you need for all your entries.

After you code the necessary macro instructions, assemble them. This will produce your auxiliary error option table, with a name of pidUOPT (where *pid* is the auxiliary product identifier you supplied on the first macro instruction).

Refer to your auxiliary product's documentation to determine where and how to store your assembled table to make it part of the auxiliary product.

Format of the First Macro Instruction

The first form of the macro has the following syntax:

```
VSF2AMTB COMPID=pid,
         MSGNUM1=firstnum,
         MSGNUM2=lastnum
```

The keyword operands and their parameters are:

COMPID = pid
specifies the first three characters of the table name. The macro concatenates these three characters with the characters UOPT, creating a name for the table of the form pidUOPT. The first character of *pid* must be a letter, the following two characters must be alphanumeric.

MSGNUM1 = firstnum
specifies the starting number of the table.

MSGNUM2 = lastnum
specifies the ending number of the table.

Format of the Second Macro Instruction

For each error number for which you wish to specify table values (other than defaults), use the following syntax:

```
VSF2AMTB MSGNO=(ermsno,qty)
  [,ALLOW=errs]
  [,PRINT=prmsg]
  [,MODENT={YES|NO}]
  [,PRTBUF={YES|NO}]
  [,INFOMSG={YES|NO}]
  [,TRACBAK={YES|NO}]
  [,USREXIT=exitname]
```

The parameters on this form of the macro instruction have the same function as the identically-named ones on the VSF2UOPT macro instruction. See the parameter explanations beginning on page 108. (Note, however, that this VSF2AMTB macro instruction does not have an IOERR parameter.)

If you omit a macro instruction for any error numbers in the auxiliary table range, you receive defaults for those numbers. Similarly, if you omit any individual parameter on a particular macro instruction, you receive defaults for that parameter on that particular number or numbers. These default values are:

```
ALLOW      10
PRINT      05
MODENT     YES
PRTBUF     NO
INFOMSG    NO
TRACBAK    YES
USREXIT    no user exit taken
```

Appendix B. Another Way to Build Library Composite Modules (VM)

The EXEC for installing and customizing VS FORTRAN Version 2 provides an automated method for building library composite modules AFBVRENC and AFBVLBCM both during and after installation.

The automated method may, however, require more storage than you can allocate, and so the process may fail. Or, if you are rebuilding composite modules after having dumped your installation work disk to tape, you no longer have the installation EXEC accessible on disk.

The following section provides a "manual" (that is, not EXEC-driven) method of rebuilding the composite modules. This method reduces the storage constraint.

The following tables list the library modules you can include in the various composite modules. The "Size" column lists approximate module sizes in hexadecimal. The "Default Set" column indicates which modules are placed into the composite modules during the installation process, if you accept the IBM-supplied defaults. Except for the modules that must be in the composite modules, you can subsequently add or delete modules in this set to match the needs at your installation.

If a module performs a function used frequently at your installation, you should consider including it in your composite module even if you are trying to limit the size of the composite module.

Following each list of modules is information on building the composite modules.

Composite Module AFBVRENC

Figure 41 on page 116 and Figure 42 on page 116 list the modules, both required and optional, that can be included in composite module AFBVRENC.

Module	Approx. Size	Default Set	Function
AFBCREN	18A	X	Internal linkage module
AFBCFIST	800	X	File status processor
AFBVGMFM	22D	X	GETMAIN/FREEMAIN
AFBVSIOS	2614	X	Sequential I/O services
Total	31D8	4	

Figure 41. Required Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBCVIOS	1C6C		Nonkeyed VSAM I/O services
AFBDDCMP	244		Dynamic common
AFBVAMTP	12E		Alternate error option table processor
AFBVBLNT	1EC	X	Implied DO in I/O
AFBVCLOP	21A	X	CLOSE statement
AFBVCOMH	1552	X	Formatted I/O
AFBVCONI	314	X	Input floating-point conversion
AFBVCONO	810	X	Output floating-point conversion
AFBVCVTH	1174	X	Data conversion
AFBVDIOS	1834		Direct access I/O services
AFBVEMGN	F80	X	Error message generator
AFBVERRE	234	X	Error summary
AFBVEXIP	8C		Return code processor
AFBVFMTP	160		LCP define file
AFBVIIOS	260		Internal file services
AFBVINQP	968		INQUIRE statement
AFBVIOCP	2E8		BACKSPACE, REWIND, ENDFILE
AFBVIOFP	708	X	Formatted I/O
AFBVIOLP	12C4	X	List-directed I/O
AFBVIOUP	CC0	X	Unformatted I/O
AFBVKIOS	2B20		Keyed access I/O services
AFBVLINP	270		Link to reentrant CSECT
AFBVMSKL	54D9	X	Message skeletons
AFBVOPEP	7D2	X	OPEN statement
AFBVSTAE	1134		ABEND processor
AFBVTEN	2C0	X	Powers of ten table
AFBVTRCH	ADC	X	Traceback generator

Figure 42. Optional Modules for AFBVRENC

Building the composite module AFBVRENC: The steps for building composite module AFBVRENC and placing it in a LOADLIB follow:

1. Re-access the VS FORTRAN Version 2 product disk as your A-disk.

2. Edit file LKEDRENC TEXT to add or delete the optional modules. This file consists of linkage editor control statements. Column 1 of each line must be blank, and all lines must be entered in upper case. LKEDRENC TEXT has the following format:

```
INCLUDE SYSLIB(AFBCREN)
INCLUDE SYSLIB(AFBxxxxx)
.
.
.
ORDER AFBCREN
ENTRY AFBCREN
NAME  AFBVRENC(R)
```

Add optional reentrant modules to those listed in LKEDRENC TEXT by adding a separate linkage editor INCLUDE statement for each module you want to add to AFBVRENC. Except for the module AFBCREN, do *not* code INCLUDE statements for the modules listed above as "Required."

3. When you have finished modifying LKEDRENC TEXT, build composite module AFBVRENC and place it into your LOADLIB using the following commands:

```
FILEDEF SYSLIB DISK VSF2FORT TXTLIB A
LKED LKEDRENC (NOTERM XREF LIBE VSF2LOAD
```

The LKED command creates the load module AFBVRENC in the LOADLIB called VSF2LOAD; any previous copy of the load module is replaced.

AFBVRENC as a Discontiguous Saved Segment

Composite module AFBVRENC may be built and placed into a discontiguous saved segment (DCSS). The virtual storage address selected for the DCSS must be greater than the virtual machine size of anyone who accesses it. In order to accommodate different virtual machine sizes, a facility is available to save multiple copies of composite module AFBVRENC, each with a different name and virtual storage address.

There are certain steps your VM system programmer must take before you can install AFBVRENC as a DCSS. These are described under "AFBVRENC as a Discontiguous Saved Segment" on page 91.

When your VM system programmer has made the proper preparations, continue with the following process:

1. Place AFBVRENC in the shared segment.

   A DCSS for the composite module is built with the following commands. In order to issue the SAVESYS command shown below, you must be a class E user.

```
CP DEFINE STORAGE mach-size
CP IPL CMS
.
.
.
GLOBAL  TXTLIB VSF2FORT
LOAD    AFBCREN (NOAUTO CLEAR ORIGIN seg-addr
INCLUDE AFBxxxxx (SAME
.
.
.
CP SAVESYS sys-name
ERASE   sys-name MAP A
RENAME  LOAD MAP A   sys-name = =
```

where

mach-size specifies a virtual machine size at least as large as the address at which the shared segment resides (seg-addr) plus the length of the composite module, plus storage for CMS data.

seg-addr specifies the virtual storage address at which the shared segment is to reside, as defined in the NAMESYS macro instruction for the system name table.

sys-name specifies the name of the shared segment, as defined in the NAMESYS macro instruction for the system name table and in the VSF2RNAM macro instruction for assembling the module AFBCRNAM.

The SAVESYS command saves the composite module as a DCSS using the specified name; any previous copy of this DCSS is replaced. The inclusion of any of the optional reentrant modules in composite module AFBVRENC is controlled by the CMS INCLUDE command, which refers to *AFBxxxxx*, where *AFBxxxxx* is to be replaced by the name of the module to be included. No INCLUDE command should be provided for the modules listed as "Required" in the table above.

2. Use VSF2RNAM to assemble the AFBCRNAM text deck.

   The VSF2RNAM macro builds the CSECT AFBCRNAM, which supplies the shared segment names that are available and initialized to hold the module AFBVRENC. None of the names supplied can be prefixed by the letters AFB. (However, AFBVRENC *can* be used as a valid shared segment name.)

   When coding the macro instruction, follow this guide:

   a. Column 1 must be blank.

   b. VSF2RNAM may appear anywhere before column 72 but must precede the operands by at least one blank.

   c. The operands may be continued on any number of cards as long as column 72 contains a nonblank character and the data on the following card begins in column 16.

Place your VSF2RNAM macro instruction in a file whose file name is AFBCRNAM, and whose file type is ASSEMBLE. This file must have the following format:

```
VSF2RNAM SYSNAME=(name1,name2,...)
END
```

where **name1**, **name2**, and so on are the names of one or more shared segments that contain the AFBVRENC composite module. You must list the names in increasing order of their virtual storage addresses. You must *not* list any names beginning with "AFB."

Assemble the module AFBCRNAM as follows:

```
GLOBAL   MACLIB VSF2MAC
ASSEMBLE AFBCRNAM
```

3. Insert the AFBCRNAM text deck in VSF2FORT TXTLIB.

   The TEXT file that results from the assembly of the DCSS name list must be placed in the VSF2FORT TXTLIB as follows:

   ```
   TXTLIB DEL VSF2FORT AFBCRNAM
   TXTLIB ADD VSF2FORT AFBCRNAM
   ```

4. Update composite module AFBVLBCM.

   Finally, the DCSS name list module AFBCRNAM must be placed in composite module AFBVLBCM. Place the following linkage editor INCLUDE statement in the TEXT file LKEDLBCM:

   ```
   INCLUDE SYSLIB(AFBCRNAM)
   ```

   and rebuild AFBVLBCM as described below.

Building Composite Module AFBVLBCM

Figure 43 on page 120 and Figure 44 on page 121 list the library modules you can include in the AFBVLBCM composite module. The "Size" column lists approximate module sizes in hexadecimal. The "Default Set" column indicates which modules are placed into the composite modules during the installation process. Except for the modules that must be in the composite module, you can subsequently add or delete modules in this set to match the needs at your site.

If a module performs a function used frequently at your installation, you should consider including it in your composite module even if you are trying to limit the size of the composite module.

Following Figure 43 is information on building the composite module AFBVLBCM.

Module	Approx. Size	Default Set	Function
AFBCLBC0	1428	X	Library common work area
AFBCLOAD	220	X	Loader
AFBCVIO$	C8	X	Internal linkage routine
AFBUATBL	648	X	Unit assignment table
AFBUOPT	60C	X	Error option table
AFBVBLN$	34	X	Internal linkage routine
AFBVCMSS	404	X	CMS interface
AFBVCNI$	34	X	Internal linkage routine
AFBVCNO$	34	X	Internal linkage routine
AFBVCOM$	34	X	Internal linkage routine
AFBVCOM2	F9C	X	Initialization/termination
AFBVCVT$	30C	X	Internal linkage routine
AFBVDEB$	34	X	Internal linkage routine
AFBVDIO$	3A	X	Internal linkage routine
AFBVEMG$	78	X	Internal linkage routine
AFBVERE$	34	X	Internal linkage routine
AFBVERS$	60	X	Internal linkage routine
AFBVFNTH	8D6	X	Program interrupt handler
AFBVGPRM	7B	X	Default execution-time options
AFBVIAD$	34	X	Internal linkage routine
AFBVIIO$	3A	X	Internal linkage routine
AFBVINI$	34	X	Internal linkage routine
AFBVKIO$	3A	X	Internal linkage routine
AFBVLOC$	34	X	Internal linkage routine
AFBVPARM	6F4	X	Execution-time options processor
AFBVPOS$	34	X	Internal linkage routine
AFBVSPIE	154	X	Interrupt interceptor
AFBVSTA$	10C	X	Internal linkage routine
AFBVTRC$	34	X	Internal linkage routine
Total	50D8	29	

Figure 43. Required Modules for AFBVLBCM

Building AFBVLBCM: The steps for building composite module AFBVLBCM and placing it in a LOADLIB follow:

1. Re-access the VS FORTRAN Version 2 product disk as your A-disk.

2. Edit file LKEDLBCM TEXT to add or delete the optional modules. This file consists of linkage editor control statements. Column 1 of each line must be blank, and all lines must be entered in upper case.

   LKEDLBCM TEXT has the following format:

```
INCLUDE SYSLIB(AFBCLBC0)
INCLUDE SYSLIB(AFBxxxxx)
.
.
.
ORDER AFBCLBC0
ENTRY AFBLBCOM
ALIAS IFYVLBCM
NAME  AFBVLBCM(R)
```

Add optional modules to those listed in LKEDLBCM by adding a separate linkage editor INCLUDE statement. Except for the module AFBCLBC0, do *not* code INCLUDE statements for the modules listed above as "Required."

3. When you have finished modifying LKEDLBCM TEXT, build composite module AFBVLBCM and place it into a LOADLIB using the following commands:

```
FILEDEF SYSLIB DISK VSF2FORT TXTLIB A
LKED LKEDLBCM (NOTERM REUS XREF LIBE VSF2LOAD
```

The LKED command creates the load module AFBVLBCM in the LOADLIB called VSF2LOAD; any previous copy of the load module is replaced.

Module	Note	Approx. Size	Default Set	Function
AFBCRNAM	4			DCSS name list
AFBDIOCP		1B0		Define file (LANGLVL 66)
AFBDSPAP	2	51C		Dimension calculator
AFBIBCOP	1	8D4		Pre-VS FORTRAN interface
AFBLDFIP	2	450		List-directed I/O
AFBNAMEP	2	39C		Namelist I/O
AFBSDUMQ		21CE		SDUMP subroutine
AFBTFORP		118		Debugging Interface
AFBVBALG		5F4		Boundary alignment routine
AFBVDBUP		11B6		Debugging packet
AFBVDUMQ		6CC		DUMP/PDUMP subroutine
AFBVINTH		4E8		Vector program interrupt handler
AFBVIONP		1010		Namelist I/O
AFBVLOCA		63C		Statement number locator
AFBVMOPP		490		Extended error handling
AFBVPOSA		30C4		Post ABEND processor
AFBVSCOP	3	654		Pre-Release 4.0 interface
AFBVSPAP		46C		Dimension calculator
AFBVSPIP		4E4		Dynamic spill area processor
AFBVUNIN		1E4		Unnormalized operand interrupt handler

Figure 44. Optional Modules for AFBVLBCM

Notes to Figure 44:

1. Module AFBIBCOP is used when running object decks produced by FORTRAN compilers prior to VS FORTRAN. It is needed for formatted and unformatted I/O and for initialization from a main program.

2. These modules are used for the specified functions that are performed from object decks produced by FORTRAN compilers prior to VS FORTRAN Version 1, Release 4.0.

3. Module AFBVSCOP is used when running object decks produced by the VS FORTRAN Version 1 compiler from prior to Release 4.0. It is needed for formatted and unformatted I/O and for initialization from a main program or from a subroutine with character arguments.

4. If you have placed AFBVRENC in a DCSS, you must assemble module AFBCRNAM by coding a VSF2RNAM macro instruction, as described in step 2 on page 118. The macro instruction specifies the names of your DCSSs that have copies of the reentrant composite module AFBVRENC. After assembling module AFBCRNAM, you must incorporate it into composite module AFBVLBCM, as described in the section "Building AFBVLBCM" on page 120. If module AFBCRNAM is not in the composite module, only the DCSS name AFBVRENC will be accessed.

Appendix C. Program Support

The VS FORTRAN Version 2 Compiler, Library, and Interactive Debug is classified as a licensed program (LP) with S G program services. S G program services provide corrective and preventive service for product defects and support for resolving program problems through Central Service, including the IBM Support Center. For details of these facilities and a list of all the products supported, refer to *Field Engineering Programming System General Information Manual*, G229-2228.

When you encounter a problem or defect, follow this procedure:

1. Consult the *VS FORTRAN Version 2: Diagnosis Guide* for a solution to the problem.

2. If a solution to the problem is not found there, report the problem to the IBM Support Center, using the manual as a guide.

At the IBM Support Center either the problem will be resolved, or your report will be accepted as an authorized programming analysis report (APAR) describing a probable product defect.

An APAR is resolved by Central Service with either an explanation or a new corrective service program temporary fix (PTF) for the defect. A PTF is a replacement text module that is installed in the product to correct the defect. Collections of new PTFs for products are provided to all customers as preventive service program update tapes (PUTs).

Assistance in resolving problems and correcting defects is available through Marketing Product Support.

Appendix D. Applying Service under MVS

The VS FORTRAN Version 2 administrator or system programmer applies service to the VS FORTRAN Version 2 product with service tapes that are distributed by IBM. Enclosed with each service tape is a Memo to Users, which contains a list of tape contents, and a cover letter for each PTF (Program Temporary Fix) on the tape. Service is distributed in a format acceptable to both SMP4 and SMP/E:

Label: Nonlabeled (NL)

DCB attributes: (RECFM = FB,LRECL = 80,BLKSIZE = 7200)

File description: File 1 = Requested PTFs

Applying Service Using SMP4

The following steps will apply service using SMP4 to the *entire* VS FORTRAN Version 2 product. To apply service to the VS FORTRAN Version 2 Library *only*, replace VSFACC4C with VSFACC4L in the following JCL.

1. RECEIVE the service using the following sample JCL:

```
//RECEIVE      JOB
//SMP          EXEC VSFACC4C
//SMPWRK3  DD  UNIT=SYSDA,DISP=(,DELETE),SPACE=(CYL,(3,1,10)),
//            DCB=BLKSIZE=3120
//SMPPTFIN DD  DISP=OLD,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200),
//            DSN=PTF,VOL=SER=PTFTPE,LABEL=(1,NL),UNIT=TAPE
//SMPCNTL  DD  *
   RECEIVE
/*
```

2. APPLY the service using the following sample JCL:

```
//APPLY        JOB
//SMP          EXEC VSFACC4C
//SMPCNTL  DD  *
   APPLY SELECT (xxxxxx) ASSEM.
/*
```

where xxxxxx is the PTF number, found in the Memo to Users enclosed with each service tape.

The following steps will apply service using SMP/E to the *entire* VS FORTRAN Version 2 product. To apply service to the VS FORTRAN Version 2 Library *only*, replace VSFACCEC with VSFACCEL in the following JCL.

1. RECEIVE the service using the following sample JCL:

```
//RECEIVE      JOB
//SMP          EXEC VSFACCEC
//SMPPTFIN DD  DISP=OLD,DCB=(RECFM=FB,LRECL=80,BLKSIZE=7200),
//             DSN=PTF,VOL=SER=PTFTPE,LABEL=(1,NL),UNIT=TAPE
//SMPCNTL  DD  *
    SET BDY (GLOBAL) .        /* Set to Global Zone */
    RECEIVE SOURCEID(xxxxxx) /* Receive sysmods and */
            SYSMODS .         /* assign a common id */
/*
```

2. APPLY the service using the following sample JCL:

```
//APPLY        JOB
//SMP          EXEC VSFACCEC
//SMPCNTL  DD  *
    SET BDY (TVSF2) .        /* Set to Target Zone */
    APPLY SOURCEID(xxxxxx)  /* Specify id used in RECEIVE */
          APAR ASSEM .       /* Apply APAR(s) with all requisites */
/*
```

where xxxxxx is the PTF number, found in the Memo to Users enclosed with each service tape.

Appendix E. Applying Service under VM

Service under VM is distributed in two formats: PTFs (corrective) and PUTs (preventive).

Corrective Service

A **PTF**, or Program Temporary Fix, is distributed to a specific customer, and contains a correction for a single known problem in a particular product. As soon as a new problem is identified and the solution is established, a PTF is created and made available on a corrective service tape. You can request a cumulative tape on which all the PTFs you need for your site are stacked. The corrective service tape is distributed in VMFPLC2 format. It contains *only* the TEXT file or files required for the particular PTFs you have requested.

To apply VS FORTRAN Version 2 fixes distributed as PTFs, you must perform the following steps:

1. If you are going to apply service to the original product disk, you may want to provide different names for the libraries from those you now have on the product disk. After testing to make sure the new libraries work, you can replace the existing libraries with these new ones.

2. Please refer to the memo accompanying the corrective service tape for setting up minidisks and load files from tape to disk.

   Note: VS FORTRAN Version 2 is not supported by the VMFMERGE EXEC discussed in the corrective service memo.

3. Invoke the EXEC used to install the product in "service mode," as follows:

If you want to install service to the:	Invoke the installation EXEC as follows:
VS FORTRAN Version 2 (entire product)	EXEC I5668806 PTFINST
VS FORTRAN Version 2 (library only)	EXEC I5668805 PTFINST

4. The EXEC will then prompt you for information necessary to install the service.

5. When it is through, the EXEC will generate the product on the A disk, and then install service from the A disk to the product disk.

PUTs, or Program Update Tapes, are distributed to all customers on a regular basis. They are made up of all the PTFs to problems in IBM licensed programs that operate under your operating system. A PUT contains cumulative information; thus, you need only the original distribution tape and the latest PUT to construct a system at the most up-to-date level. The VM PUT is distributed in VMFLPC2 format, and the files on the tape are organized as follows:

File 1: VMSERV EXEC

File 2: Memo to Users for all program products represented on this PUT

File 3 and on: Each product's service EXEC and service files

The service EXECs necessary to apply service are contained on the PUT, and are invoked by VMSERV.

To apply VS FORTRAN Version 2 fixes distributed on a PUT, you must perform the following steps:

1. Access the original installation work disk with a filemode of **A**. This disk must contain the product TEXT files, as well as the product installation EXEC.

2. Choose a second disk to be used as a **staging** disk. Access this disk with a filemode of **B**.

3. Choose a third disk to contain the VMSERV EXEC. Access this disk with a filemode of **C**.

4. Mount the PUT at virtual address 181.

5. Issue the command `VMFPLC2 LOAD * * C` to load the VMSERV EXEC onto the C disk.

6. Issue `VMSERV`. The VMSERV EXEC asks if you want to print the Memo to Users. If you answer "YES", the EXEC issues the print command and then terminates.

7. Read the Memo to Users for the VS FORTRAN Version 2 service file(s). They contain more specific and detailed instructions for installing this service. When you have read the Memo to Users, issue `VMSERV` again, and answer "NO" to the Memo to Users prompt.

8. The VMSERV EXEC now asks if you want to install service. Answer "YES" to this prompt. VMSERV now loads the first service EXEC, advances the tape to the beginning of the first service file, and invokes the service EXEC to install the first service file.

9. The service EXEC then loads the service file(s) onto the B-disk and moves them to the A-disk with the replace option. Lastly, the service EXEC invokes the install EXEC with the PTFINST option.

If an error occurs, VMSERV issues an error message and either terminates or indicates what you should do next.

When all the VS FORTRAN Version 2 service has been installed, VMSERV will apply service for the remaining products on the tape or allow you to exit.

Index

A

C

M

N

O

P

R

S

T

U

V

W

X

VS FORTRAN Version 2:
Installation and Customization
SC26-4224-1

Reader's Comment Form

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

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

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

If you wish a reply, give your name, company, mailing address, and telephone number.

_____

_____

_____

_____

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

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

Last TNL _____

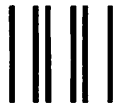
Previous TNL _____

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

SC26-4224-1

Reader's Comment Form

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



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

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

POSTAGE WILL BE PAID BY ADDRESSEE

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

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

VS FORTRAN Version 2: Installation and Customization (File No. S370-34) Printed in U.S.A. SC26-4224-1

The VS FORTRAN Version 2 Library

LY27-9516	Diagnosis Guide
GC26-4219	General Information
SC26-4224	Installation and Customization
SC26-4223	Interactive Debug Guide and Reference
SC26-4221	Language and Library Reference
GC26-4225	Licensed Program Specifications
SC26-4222	Programming Guide
SX26-3751	Reference Summary

SC26-4224-01

Printed in U.S.A.