IBM

VS FORTRAN Version 2

SC26-4339-1

Installation and Customization for VM

Release 3

VS FORTRAN Version 2

SC26-4339-1

Installation and Customization for VM

Release 3

| **Second Edition (March 1988)**

| This is a major revision and makes obsolete SC26-4339-0.

| This edition applies to Release 3 of VS FORTRAN Version 2, Program Numbers 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 publication 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, 4300, and 9370 Processors Bibli-*
| *ography*, 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, Programming Publishing, P.O. Box 49023,
| San Jose, California, U.S.A. 95161-9023. 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 book describes how to install, customize, and service VS FORTRAN Version 2 for VM. It is designed for system programmers and planners who supervise the generation and maintenance of an organization's operating system.

How This Book Is Organized

- **Chapter 1, "Getting Acquainted with VS FORTRAN Version 2"**, describes VS FORTRAN Version 2 and discusses where to find more information about VS FORTRAN Version 2.
- **Chapter 2, "Planning for Installation"**, specifies the required systems and hardware, as well as the amount of virtual and DASD storage space you will need as you plan for the installation of VS FORTRAN Version 2.
- **Chapter 3, "Installing VS FORTRAN Version 2"**, provides the procedures and data you need to install VS FORTRAN Version 2.
- **Chapter 4, "Making Interactive Debug Available to the User"**, provides instructions to make Interactive Debug available to ISPF/PDF users, users of ISPF without PDF, and non-ISPF users.
- **Chapter 5, "Customizing VS FORTRAN Version 2"**, describes how to make the alternative math library subroutines available, build composite modules, put the compiler in a DCSS, and change the option defaults.
- **Appendix A, "Program Materials"**, describes the VS FORTRAN Version 2 system tapes available from IBM Software Distribution (ISD) for your installation.
- **Appendix B, "Customization Macros"**, describes the customization macros and how to invoke them.
- **Appendix C, "Composite Modules"**, provides information about the required and optional composite modules.
- **Appendix D, "Servicing VS FORTRAN Version 2"**, discusses problem reporting, corrective service, and preventive service.

How to Use This Book

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

1. Chapter 2, "Planning for Installation" on page 3.
2. Chapter 3, "Installing VS FORTRAN Version 2" on page 11.

Following successful installation, you can perform the following optional activities:

- Make VS FORTRAN Version 2 Interactive Debug available, if you purchased the complete product. Refer to Chapter 4, "Making Interactive Debug Available to the User" on page 19. Separate procedures are provided for using ISPF with PDF, for using ISPF without PDF, and for non-ISPF users.
- Customize VS FORTRAN Version 2 to your environment. Refer to Chapter 5, "Customizing VS FORTRAN Version 2" on page 31; Appendix B, "Customization Macros" on page 45; and Appendix C, "Composite Modules" on page 65.
- Apply service to VS FORTRAN Version 2. Refer to Appendix D, "Servicing VS FORTRAN Version 2" on page 73.

Syntax Notation

The following items explain how to interpret the syntax used in this manual:

- Uppercase letters and special characters (such as commas and parentheses) are to be coded exactly as shown, except where otherwise noted. You can, however, mix lowercase and uppercase letters; lowercase letters are equivalent to their uppercase counterparts, except in most character constants.
- Italicized, lowercase letters or words indicate variables, such as array names or data types, and are to be substituted.
- Underlined letters or words indicate IBM-supplied defaults.
- Ellipses (...) indicate that the preceding optional items may appear one or more times in succession.
- Braces ({ }) group items from which you must choose one.
- Square brackets ([ ]) group optional items from which you may choose none, one, or more.
- OR signs (|) indicate you may choose only one of the items they separate.
- Blanks in FORTRAN statements are used to improve readability; they have no significance, except when shown within apostrophes (' '). In non-FORTRAN statements, blanks may be significant. Code non-FORTRAN statements exactly as shown.

Documentation of IBM Extensions

In addition to the statements available in FORTRAN 77, IBM provides "extensions" to the language. In *VS FORTRAN Version 2 Language and Library Reference*, these extensions are printed in color.

Summary of the VS FORTRAN Version 2 Publications

The following table lists the VS FORTRAN Version 2 publications and the tasks they support.

Task	VS FORTRAN Version 2 Publications	Order Numbers
Evaluation and Planning	General Information	GC46-4219
	Licensed Program Specifications	GC26-4225
Installation and Customization	Installation and Customization for VM	SC26-4339
	Installation and Customization for MVS	SC26-4340
Application Programming	Language and Library Reference	SC26-4221
	Programming Guide	SC26-4222
	Interactive Debug Guide and Reference	SC26-4223
	Reference Summary	SX26-3751
Diagnosis	Diagnosis Guide	LY27-9516

Industry Standards

The VS FORTRAN Version 2 Compiler and Library licensed program is designed according to the specifications of the following industry standards, as understood and interpreted by IBM as of March, 1988.

The following two standards are technically equivalent. In the publications, references to **FORTRAN 77** are references to these two standards:

- American National Standard Programming Language FORTRAN, ANSI X3.9-1978 (also known as FORTRAN 77)
- International Organization for Standardization ISO 1539-1980 Programming Languages-FORTRAN

The bit string manipulation functions are based on ANSI/ISA-S61.1.

The following two standards are technically equivalent. References to **FORTRAN 66** are references to these two standards:

- American Standard FORTRAN, X3.9-1966
- International Organization for Standardization ISO R 1539-1972 Programming Languages-FORTRAN

At both the FORTRAN 77 and the FORTRAN 66 levels, the VS FORTRAN Version 2 language also includes IBM extensions. References to **current FORTRAN** are references to the FORTRAN 77 standard, plus the IBM extensions valid with it. References to **old FORTRAN** are references to the FORTRAN 66 standard, plus the IBM extensions valid with it.

Summary of Changes

Release 3, March 1988

Major Changes to the Product

- Enhancements to the vector feature of VS FORTRAN Version 2
  - Automatic vectorization of user programs is improved by relaxing some restrictions on vectorizable source code. Specifically, VS FORTRAN Version 2 can now vectorize MAX and MIN intrinsic functions, COMPLEX compares, adjustably dimensioned arrays, and DO loops with unknown increments.
  - Ability to specify certain vector directives globally within a source program.
  - Addition of an option to generate the vector report in source order.
  - Ability to collect tuning information for vector source programs.
    - Ability to record compile-time statistics on vector length and stride and include these statistics in the vector report.
    - Ability to record and display run-time statistics on vector length and stride. Two new commands, VECSTAT and LISTVEC, have been added to Interactive Debug to support this function.
    - Enhancements to Interactive Debug to allow timing and sampling of DO loops.
    - Inclusion of vector feature messages in the on-line HELP function of Interactive Debug.
  - Vectorization is allowed at OPTIMIZE(2) and OPTIMIZE(3).
- Enhancements to the language capabilities of VS FORTRAN Version 2
  - Ability to specify the file or data-set name on the INCLUDE statement.
  - Ability to write comments on the same line as the code to which they refer.
  - Support for the DO WHILE programming construct.
  - Support for the ENDDO statement as the terminal statement of a DO loop.
  - Enhancements to the DO statement so that the label of the terminal statement is optional.
  - Support for statements extending to 99 continuation lines or a maximum of 6600 characters.
  - Implementation of IBM's Systems Application Architecture (SAA) FORTRAN definition; support for a flagger to indicate source language that does not conform to the language defined by SAA.

- Support for the use of double-byte characters as variable names and as character data in source programs, I/O, and for Interactive Debug input and output.
- Support for the use of a comma to indicate the end of data in a formatted input field, thus eliminating the need for the user to insert leading or trailing zeros or blanks.

- Enhancements to the programming aids in VS FORTRAN Version 2
  - Enhancements to the intercompilation analysis function to detect conflicting and undefined arguments.
  - Support for the Data-In-Virtual facility of MVS/XA.
  - Ability to allocate certain commonly used files and data sets dynamically.
  - Enhancements to the Multitasking Facility to allow large amounts of data to be passed between parallel subroutines using a dynamic common block.
  - Support for named file I/O in parallel subroutines using the Multitasking Facility.
  - Ability to determine the amount of CPU time used by a program or a portion of a program by using the CPUTIME service subroutine.
  - Ability to determine the FORTRAN unit numbers that are available by using the UNTANY and UNTNOFD service subroutines.
- Enhancements to the full screen functions of Interactive Debug

Major Changes to This Manual

- Documentation of the major product enhancements has been added.
- Addition of section for installing Interactive Debug using ISPF without PDF.
- Additional information, including Program Materials, Program Support, and DASD space requirements for files and libraries.

Release 2, June 1987

Major Changes to the Product

- Support for 31-character symbolic names, which can include the underscore (_) character.
- The ability to detect incompatibilities between separately-compiled program units using an intercompilation analyzer. The ICA compile-time option invokes this analysis during compilation.
- Addition of the NONE keyword for the IMPLICIT statement.
- Enhancement of SDUMP when specified for programs vectorized at LEVEL(2), so that ISNs of vectorized statements and DO-loops appear in the object listing.
- The ability of run-time library error-handling routines to identify vectorized statements when a program interrupt occurs, and the ability under Interactive Debug to set breakpoints at vectorized statements.
- The ability, using the INQUIRE statement, to report file existence information based on the presence of the file on the storage medium.
- Addition of the OCSTATUS execution-time option to control checking of file existence during the execution of OPEN statements, and to control whether files are deleted from their storage media.
- Under MVS, addition of a data set and an optional DD statement to be used during execution for loading library modules and Interactive Debug.
- Under VM, the option of creating during installation a single VSF2LINK TXTLIB for use in link mode in place of VSF2LINK and VSF2FORT.
- The ability to sample CPU use within a program unit using Interactive Debug. The new commands LISTSAMP and ANNOTATE have been added to support this function.
- The ability to automatically allocate data sets for viewing in the Interactive Debug source window.

Major Changes to This Manual

- Documentation of the above product enhancements has been added.
- Separation of *VS FORTRAN Version 2 Installation and Customization* manual into two new manuals:
  - *VS FORTRAN Version 2 Installation and Customization for VM*
  - *VS FORTRAN Version 2 Installation and Customization for MVS*

  These two new manuals include all the information in the original manual regarding their respective operating systems.

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 has been added.

Contents

Figures

Chapter 1. Getting Acquainted with VS FORTRAN Version 2

What It Is

VS FORTRAN Version 2 is an application programming tool, made up of the following components:

- The Compiler, which translates programs written in the VS FORTRAN Version 2 language and produces object modules for subsequent running of the program with the support of the VS FORTRAN Version 2 Library.
- The Library, which 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.
- Interactive Debug (IAD), which allows the programmer to monitor running of VS FORTRAN Version 2 programs and to examine and change data at run time.

It is available as two separate products as stated below:

- VS FORTRAN Version 2 (5668-806), the complete licensed program containing the Compiler, Library, and Interactive Debug. It also provides sample programs that enable you to verify the installation procedures.
- VS FORTRAN Version 2 (5668-805), a licensed program containing only the Library.

VS FORTRAN Version 2 is distributed by IBM Software Distribution (ISD). It is available on either an unlabeled 9-track tape, written at 1600 or 6250 bpi, 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.

Where to Find More Information

- The program directory, which accompanies the VS FORTRAN Version 2 product tape, contains additional information regarding program and service level information and supplemental installation notes.
- A description of the functions supported by VS FORTRAN Version 2 can be found in the program announcement materials. Refer to *VS FORTRAN Version 2 General Information* or contact your IBM marketing representative for this information.
- You can obtain updates to the information and procedures in this book from either your IBM Support Center or Facility, as discussed on page 3.

Chapter 2. Planning for Installation

The purpose of this chapter is to help you plan for the installation of VS FORTRAN Version 2. It specifies the required systems and hardware, as well as the amount of virtual and DASD storage space you will need.

Before installing VS FORTRAN Version 2 Release 3, contact your IBM Support Center or check the RETAIN/370 PSP (Preventive Service Planning) Facility for updates to the information and procedures in this manual. To obtain this information from the PSP Facility specify:

UPGRADE	**Subset**
VSFORTRAN230	VSFORT/830

Meeting the Basic Requirements

System Requirements

VS FORTRAN Version 2 supports vector and scalar compilation, and scalar processing, under:

- VM/System Product (5664-167)—Release 4 or later, with or without VM/SP HPO (5664-173)—Release 4 or later[1,3]
- VM/XA System Product (5664-308)—Release 1, with bimodal CMS [2]
- VM/XA System Facility (5664-169)—Release 2

VS FORTRAN Version 2 supports vector processing under:

- VM/System Product (5665-167)—Release 4 or later, with VM/SP HPO (5664-173)—Release 4.2, with Vector Facility Support[1,3]
- VM/XA System Facility (5664-169)—Release 2
- VM/XA System Product (5664-308)—Release 1, with bimodal CMS [2]

If you are using a VM/SP System Release 4 or 5 with or without HPO and intend to use Dynamic File Allocation, then you must apply APAR VM29025 to your system.

On VM, interactive debugging requires CMS. Interactive debugging in full-screen mode also requires ISPF Version 2 for VM (5664-282), with or without ISPF/PDF Version 2 for VM (5664-285). For enhanced full-screen functions, 5664-282 is required.

In the requirements given above for interactive debugging in full-screen mode, the appropriate ISPF/PDF product must be selected in order to use the following capabilities:

- PDF browse and edit facilities in split-screen mode
- Automatic browse of the debug print file and log at session end
- Start of debugging using Interactive Debug's foreground invocation panel

Customizing VS FORTRAN Version 2 for execution on an XA-mode machine under VM/XA requires Assembler H Version 2.

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

---

[1] Use of the double-byte character set requires VM/System Product Release 5.

[2] VS FORTRAN Version 2 Release 3 will support VM/XA System Product Release 1 with bimodal CMS concurrent with the availability of ISPF support.

[3] Processing of VSAM files under VM/SP, and under VM/XA in 370 compatibility mode, requires VSE/VSAM (5746-AM2) Release 1, 2, or 3.

Machine Requirements

The compile-time machine requires both of the following:

- Any processing unit supported by VM, and
- I/O devices used by the compiler; these are normally disks.

The run-time machine requires all of the following:

- Any processing unit supported by VM,
- I/O devices used by the object program when running, and
- An appropriate vector processor, if required by your environment.

Virtual Storage Requirements

The VS FORTRAN Version 2 Compiler requires 1770K 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 400K bytes of virtual storage to begin running, in addition to the storage required for the application program being debugged. The main storage requirements for debugging a program with VS FORTRAN Version 2 Interactive Debug will depend on the function of the user data. Interactive Debug also acquires additional dynamic storage when the application program is running. The amount varies according to the nature of the program being debugged and the type and quantity of debugging commands issued.

DASD Storage Requirements

To install VS FORTRAN Version 2, you need space available on 2 target disks: the **work** disk and the **product** disk. Figure 1 shows the minimum DASD storage required. The space includes vector routines, the "combined" link library, and inclusion of ALL optional modules in the composite modules. Data shown in the table denote cylinders, except for fixed block devices (3310, 3370, 9332, and 9335) which are shown in blocks.

SPACE	T3330-11	T3340	T3350	T3375	T3380	T3310, T3370, T9332, T9335
Work Disk	66	120	32	40	25	30000
Product Disk	53	100	25	32	20	24000

Figure 1. DASD Storage Requirements

Specify a block size of at least 1024 bytes.

Note: After installation is complete, you should not have access to the installation work disk or any disk containing VS FORTRAN TEXT files. If this work disk is accessed, the NOAUTO option on the LOAD command must be used; otherwise, a TEXT file could be used instead of a library member and an error could result.

Identifying the VS FORTRAN Version 2 Files

Installation of VS FORTRAN Version 2 will put the following files onto your product disk:

Text Libraries

VSF2FORT TXTLIB	Run-time library routines needed for the creation of a program that is ready to run. (In the following sections, this text library is referred to as the principal text library.)
VSF2LINK TXTLIB	Interface routines used in link mode only. (In the following sections, this text library is referred to as the link mode text library.)
VSF2MATH TXTLIB	Alternative mathematical routines from VS FORTRAN Version 1 Release 4

Load Library

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

Separation Tools

AFBVRSEP MODULE	Separates a compiler-produced object module into its shareable and nonshareable parts.
VSF2RCS EXEC	Uses the AFBVRSEP module to compile a VS FORTRAN program using the RENT option, and to separate the object deck from the compilation into its shareable and nonshareable parts.
VSF2RSEP EXEC	Uses the AFBVRSEP module to separate a TEXT file produced by the compiler into its shareable and nonshareable parts.

Compiler Files

FORTVS2 MODULE	Compiler
ILX0TRCE TXTLIB	Trace Modules

Interactive Debug Files

VSF2MLIB MACLIB	IAD, with ISPF, Message Library
VSF2PLIB MACLIB	IAD, with ISPF, Panel Library
AFFLOADF TEXT	IAD, with ISPF, Invocation Module
AFFFX11 EXEC	IAD, with ISPF/PDF, Invocation EXEC
IAD EXEC	IAD, with ISPF, Invocation EXEC
IAD Help Files	Interactive Debug help information
IADCMDS Table	Interactive Debug, under ISPF, Help Table

Preparing for Shared Segment (DCSS) Installation

This step is optional. If you want to install the compiler or the Library's composite modules AFBVRENA, AFBVRENB, or AFBVRENC in a discontiguous shared segment (DCSS), perform the steps described below. If not, proceed with Chapter 3, "Installing VS FORTRAN Version 2" on page 11.

Note: You must have class E privileges to install a DCSS.

1. Determine the Characteristics of each DCSS

- Decide the name of each DCSS. This may be any name you specify.

  CAUTION: If it is likely that you will have more than one release of VS FORTRAN Version 2 installed on your system at the same time, choose different DCSS names for each release of the product.

- Determine the starting address of each DCSS. You can establish 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 will use the DCSS.
  - If the DCSS is set unnecessarily high, storage will be 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 shared 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 shared segments that contain parts of ISPF.

- Determine the number of segments in each DCSS. For the compiler, this number is fixed at twenty-three 64K segments. For each composite module, this number depends on whether you choose to include any optional modules in the composite module.

  - If you are not modifying the composite module, then the number of segments is as follows:

Composite Module	Number of Segments
AFBVRENA	2
AFBVRENB	1
AFBVRENC	2

Figure 2. Composite Module Sizes

  - If you are modifying the composite module, then you must:

    1. Determine which modules to include in the composite module. See Appendix C, "Composite Modules" on page 65 for a list of required and optional modules for each composite module.
    2. Calculate the total space required for the modules to be included, as follows: add the space needed for the required modules plus the space required for each optional module you want to include.
    3. Determine the number of 64K segments needed by dividing the total space, calculated during step 2, by 64K. The result, rounded up to the next integer, is the number of segments you will need.

2. Define each DCSS to VM

For VM/SP:

1. Allocate permanent space on a CP-owned DASD volume to contain each DCSS. For more information, refer to *VM/SP Planning Guide and Reference*, SC19-6201.
2. Add 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). There must be one NAMESYS macro for each DCSS being created.

   Figure 3 on page 9 defines a compiler DCSS named DSSVFORT. The data is based on a DCSS calculated to begin at location X'500000'. Although it is not shown in the figure, remember to include a continuation indicator in column 72. Otherwise the NAMESYS macro instruction will not assemble successfully.

   Figure 4 on page 9 defines a composite module DCSS named FTNLIB10. The sample data illustrate one possible set of numbers and are not intended as the only location or size for a 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. Load the nucleus on the IPL volume, then re-IPL.

For VM/XA:

Issue a DEFSEG command. See Figure 5 and Figure 6 on page 10. Refer to *VM/XA Systems Facility CP Command Reference*, GC19-6215, and *VM/XA Systems Product CP Command Reference*, SC23-0358, for more information.

You may now proceed with Chapter 3, "Installing VS FORTRAN Version 2" on page 11.

```
DSSVFORT NAMESYS SYSNAME=DSSVFORT,              Note 1
                 SYSSIZE=1472K,
                 SYSHRSG=(80,81,82,83,84,85,86,
                     87,88,89,90,91,92,93,94,95,
                     96,97,98,99,100,101,102),  Note 2
                 SYSPGNM=(1280-1647),           Note 3
                 VSYSADR=IGNORE,
                 SYSVOL=VMSRES,                 Note 4
                 SYSSTRT=(049,1)                Note 5
```

Notes:

1. The SYSNAME parameter specifies the name of the DCSS (DSSVFORT in this example). Change the name to whatever you desire.
2. 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 was established at X'500000' or 5120K. Dividing 5120K by 64K gives a starting segment number of 80.
3. 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. The compiler requires twenty-three 64K segments, a total of 1472K. Dividing 1472K by 4K (the page size) yields a total of 368 pages. Starting at page 1280, 368 pages results in an ending page of 1647.
4. The SYSVOL parameter gives the volume serial number of the CP-owned volume that holds the DCSS.
5. The SYSSTRT parameter gives the starting cylinder and page address (on the volume specified by the SYSVOL parameter) that holds the DCSS.

Figure 3. VM/SP Sample NAMESYS Macro Instruction for Compiler DCSS

```
FTNLIB10 NAMESYS SYSNAME=FTNLIB10,              Note 1
                 SYSSIZE=128K,
                 SYSHRSG=(48,49),               Note 2
                 SYSPGNM=(768-799),             Note 3
                 VSYSADR=IGNORE,
                 SYSVOL=VMSRES,                 Note 4
                 SYSSTRT=(072,1)                Note 5
```

Notes:

1. The SYSNAME parameter specifies the name of the DCSS (FTNLIB10 in this example). Change the name to whatever you desire.
2. 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 was established at X'300000' or 3072K. Dividing 3072K by 64K gives a starting segment number of 48.
3. 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.
4. The SYSVOL parameter gives the volume serial number of the CP-owned volume that holds the DCSS.
5. The SYSSTRT parameter gives the starting cylinder and page address (on the volume specified by the SYSVOL parameter) that holds the DCSS.

Figure 4. VM/SP Sample NAMESYS Macro Instruction for Composite Module DCSS

```
DEFSEG DSSVFORT  900-A70 SR
```

Notes:

1. DSSVFORT is name of DCSS for the compiler. Change the name to whatever you desire.
2. The hex numbers 900-A70 are the first and last page numbers of the DCSS. In this example, we want the compiler DCSS to be loaded starting at X'0900000' (or the 9M line). To compute the hexadecimal page numbers,
   a. for the starting page, see the following examples,
   b. for the ending page, add a hex factor of 170 (or 1472K).

Start	Storage-Address	HexPage
5M	00500 000	500
6M	00600 000	600
7M	00700 000	700
8M	00800 000	800
9M	00900 000	900

   Warning: the ending page for the compiler DCSS cannot be greater than hex FFF.
3. SR indicates the segments are to be shared and page protected, rather than exclusive (EW).

Figure 5. VM/XA Sample DEFSEG Command for Compiler DCSS

```
DEFSEG FTNLIB20 1100-1120 SR
```

Notes:

1. FTNLIB20 is the name of a DCSS for a composite module. Change the name to whatever you desire.
2. The hex numbers 1100-1120 are the first and last page numbers of the DCSS. In this example, we want AFBVRENA at its default size (2 segments) to be loaded starting at X'1100000' (or the 17M line) and ending at X'1120000'. To compute the hexadecimal page numbers,
   a. for the starting page, see the following examples,

Start	Storage-Address	HexPage
5M	00500 000	500
8M	00800 000	800
16M	01000 000	1000
20M	01400 000	1400

   b. for the ending page, add a hex factor based on number of segments.

Segments	Size(dec)	Size(hex)	Factor
1	64K	10000	10
2	128K	20000	20

   Warning: the ending page for composite modules AFBVRENB and AFBVRENC cannot be greater than hex FFF. For module AFBVRENA, the starting address should be greater than hex FFF.
3. SR indicates the segments are to be shared and page protected, rather than exclusive (EW).

Figure 6. VM/XA Sample DEFSEG Command for Composite Module DCSS

Chapter 3. Installing VS FORTRAN Version 2

This chapter provides the procedures and data you will need to install VS FORTRAN Version 2.

Proceed only if you have completed Chapter 2, "Planning for Installation" on page 3.

Logging On

1. Log on to VM and IPL CMS.
2. Define the virtual machine size:
   - If you are not installing the compiler or library in a DCSS, your virtual machine size must be defined at a minimum of 4 megabytes. For example, type:

     ```
     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 example, if you want the VS FORTRAN Version 2 compiler in a DCSS to start at 5 megabytes, and since VS FORTRAN Version 2 requires approximately 2 megabytes to load the compiler, specify the virtual machine size no less than 7 megabytes. In this example, you would type:

     ```
     CP DEFINE STORAGE 7M
     ```

     and then re-IPL CMS.

     For more information about determining storage requirements, refer to "Preparing for Shared Segment (DCSS) Installation" on page 7.
   - If you wish to use VS FORTRAN Version 2 in XA mode (under VM/XA SP), regardless of whether you are installing in a DCSS, you must define your storage to be greater than 16M to gain the advantages of XA. For example, you might type:

     ```
     CP DEFINE STORAGE 32M
     SET MACHINE XA
     ```

     and then re-IPL CMS.
3. Attach and mount the product tape at virtual address 181.

Linking to the Disks

1. Provide the **work disk**, where you will store the components required for service application to this product. It must be different from the product disk. Link in write mode to the work disk and access it as your A-disk.

   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 Figure 1 on page 5 for space requirements.
2. Provide the **product disk**, where you will install the product when it is ready to run. It must be different from the work disk, with enough space to hold the entire product, and it must not be the S-disk or the Y-disk. See Figure 1 on page 5 for space requirements.

   Specify a block size of at least 1024 bytes for the product disk in order to avoid the performance degradation that is likely to occur with a block size of 800.

   Link in write mode to the product disk.

Loading the Installation EXEC

Load the first tape file containing the I5668806 EXEC (or I5668805 if you are installing the VS FORTRAN Version 2 Library-only product) and the product identifier file onto the work disk by typing the command:

```
VMFPLC2 LOAD * * A
```

Running the Installation EXEC

In order to run the installation EXEC, you will have to utilize the information in the tables shown under "Installation Items" on page 14 and the accompanying notes.

Start by entering I5668806 (or I5668805, if you are installing the Library only).

The EXEC will then give you the opportunity to either accept or change the IBM-supplied defaults for various items shown in Figure 7, Figure 8, and Figure 9 on page 14. If you are installing the Library-only product, you will not be asked about items regarding the Compiler or Interactive Debug.

Note: You can halt the running of the installation EXEC by responding to any prompt with QUIT. To restart the installation, enter I5668806 (or I5668805) again. You will be given the choice of either restarting at the beginning, or (assuming you had made some progress before quitting) restarting after the last major step successfully completed just before your quitting point.

Installation Items

Item	Default
Product disk file mode [1]	E
Product disk address	19E
Macro library name	VSF2MAC
Change compile-time options [2]	NO
Install compiler in DCSS [3]	NO

Figure 7. Compiler Installation Items

Item	Default
Product disk file mode [1,4]	E
Product disk address [4]	19E
Macro library name [4]	VSF2MAC
Change Unit Attribute Table Defaults [2,5]	NO
Change run-time options [2]	NO
Principal text library name	VSF2FORT
Alternative math library name	VSF2MATH
Install vector library routines [6]	NO
Link mode text library name	VSF2LINK
Install link text library separate from principal text library or combined with principal text library [7]	Separate
Composite Modules: install-DCSS/Change[3]	NO
Load library name	VSF2LOAD

Figure 8. Library Installation Items

Item	Default
Install Interactive Debug	YES
Install ISPF files	YES
Install line mode IAD help files	YES
Compile and run sample program	YES

Figure 9. Interactive Debug Installation Items

Notes to Installation Items:

1. Do not specify A, S or Y; the A-disk is the work disk.
2. Refer to "Changing Option Defaults" on page 35 and Appendix B, "Customization Macros" on page 45 for more information.
3. If you install the compiler or a composite module in a DCSS, refer to "Preparing for Shared Segment (DCSS) Installation" on page 7 **before** starting the installation. Refer to Appendix C, "Composite Modules" on page 65 for more information.
4. This data will be requested only if you are installing the Library-only product.
5. File characteristics (CMS and OS/VS) are now covered under this option. To change the default file characteristics from CMS to OS/VS, see Figure 19 on page 39.
6. Install the vector library routines only if you plan to run 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 under VM/SP Release 4, the correction for APAR VM24666 must be applied to upgrade the CMS TXTLIB command before 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 PTF allows this limit to be increased.
7. Refer to "Specifying Libraries in Link Mode" on page 16 for more information.

Specifying the Mode of Run-Time Library Modules

Programmers may choose to have all run-time library modules either made a part of their program along with the compiler-generated code (link mode), or loaded dynamically at run time (load mode).

After installation of the VS FORTRAN Version 2 library, you may 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, provide an EXEC to issue the appropriate GLOBAL or FILEDEF commands.

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     (see note)
```

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

```
GLOBAL LOADLIB VSF2LOAD
```

Specifying Libraries in Link Mode

For operation in link mode, make the appropriate specification for use by the LOAD command in CMS when it includes VS FORTRAN Version 2 library modules:

- During installation, if you chose to produce a separate link mode text library, then VSF2LINK was created as a separate TXTLIB from VSF2FORT. Thus, during run-time, you must specify both VSF2LINK and VSF2FORT in the GLOBAL statement. Concatenate VSF2LINK ahead of VSF2FORT:

  ```
  GLOBAL TXTLIB VSF2LINK VSF2FORT CMSLIB     (see note)
  ```

  In this case, you *cannot* use the LKED command to create a program that will be ready to run.
- During installation, if you chose to produce a combined link mode text library, then VSF2LINK was created as a combined TXTLIB with VSF2FORT. Thus, during run-time, do not specify VSF2FORT:

  ```
  GLOBAL TXTLIB VSF2LINK CMSLIB
  ```

  In this case, you *can* use the LKED command to create a program that is ready to run. Specify this library in a CMS FILEDEF command for use by the LKED command:

  ```
  FILEDEF SYSLIB DISK VSF2LINK TXTLIB fm
  ```

  where fm is the file mode of the product disk.

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

Note: The CPU that you use to run VS FORTRAN programs must support one or more extended precision arithmetic operations (add, subtract, multiply, or divide). Otherwise, you must include CMSLIB in the TXTLIB global statement in order to load the required simulation modules (IEAXPSIM, IEAXPDXR, IEAXPALL) at run time to avoid an abend.

Verifying a Successful Installation

The last question displayed by the installation EXEC asks if you want to run the verification program. Your response and subsequent actions will be based on how you have installed the compiler.

- If you have not installed the compiler/library in a DCSS:

  1. Answer YES to this prompt.
  2. The installation EXEC will then compile and run sample program AFBIVP automatically, and you will receive the informational messages.
  3. If you have installed Interactive Debug, the EXEC will then compile and run sample program AFFIVP using Interactive Debug. When you see the FORTIAD prompt, enter GO. You will then be asked to enter the value of the circle radius. Enter %352.67. After receiving a return code of 0 and another FORTIAD prompt, enter QUIT to exit Interactive Debug. For a more complete test of Interactive Debug, refer to Figure 15 on page 29.
  4. The installation EXEC will then complete the installation. You will see the following message, indicating successful completion of the installation.

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

  5. If you created a temporary disk for the work disk (refer to item 1 on page 12 in the section "Linking to the Disks"), unload the contents of this temporary disk to tape after the installation. This will retain the files you will need later to install service. You can perform the unload by typing the command

     ```
     VMFPLC2 DUMP * * A
     ```

  Installation of VS FORTRAN Version 2 is now complete. You may now proceed with Chapter 4, "Making Interactive Debug Available to the User" on page 19 or Chapter 5, "Customizing VS FORTRAN Version 2" on page 31.

- If you have installed the compiler/library in a DCSS:

  1. Answer NO to this prompt.
  2. The installation EXEC will then complete the installation. You will see the following message, indicating successful completion of the installation.

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

     Because you have installed in a DCSS, verification must be done manually.

3. If you created a temporary disk for the work disk (refer to item 1 on page 12 in the section "Linking to the Disks"), unload the contents of this temporary disk to tape after the installation. This will retain the files you will need later to install service. You perform the unload by typing the command

   ```
   VMFPLC2 DUMP * * A
   ```

4. Define a virtual machine to fit below the address of the shared segment and re-IPL. For example, if the compiler begins at 2 megabytes (2Mb), enter the commands shown at the top of Figure 10.

```
CP DEFINE STORAGE 2M
CP IPL CMS

RELEASE A                       See Note 1

ACCESS yyy B                    See Note 2

Notes:

 1. Release A only if accessed.  Note that 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 running of the LOAD command unless the option NOAUTO is speci-
    fied.  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.

 2. yyy is the virtual address of the product disk.
```

Figure 10. Product Verification for Compiler DCSS

5. Execute the verification EXEC, specifying, in the following order, the library names used when installing and "YES" to signify that Interactive Debug was installed.

   ```
   V5668806 (VSF2FORT VSF2LINK VSF2LOAD VSF2MATH YES
   ```

6. The verification EXEC will then compile and run sample program AFBIVP automatically, and you will receive the informational messages.

7. If you have installed Interactive Debug, the EXEC will then compile and run sample program AFFIVP using Interactive Debug. When you see the FORTIAD prompt, enter GO. You will then be asked to enter the value of the circle radius. Enter %352.67. After receiving a return code of 0 and another FORTIAD prompt, enter QUIT to exit Interactive Debug. For a more complete test of Interactive Debug, refer to Figure 15 on page 29.

   Installation of VS FORTRAN Version 2 is now complete. You may now proceed with Chapter 4, "Making Interactive Debug Available to the User" on page 19 or Chapter 5, "Customizing VS FORTRAN Version 2" on page 31.

Chapter 4. Making Interactive Debug Available to the User

This is an optional step. If you wish to proceed, make sure you have completed all steps under Chapter 3, "Installing VS FORTRAN Version 2" on page 11, and have installed Interactive Debug as prompted by the installation EXEC.

Instructions for ISPF/PDF Users

Proceed only if you are using Interactive System Product Facility (ISPF) Version 2 and ISPF Program Development Facility (ISPF/PDF) Version 2. If you are using ISPF without PDF, go to "Instructions for Users of ISPF Without PDF" on page 26. If you are not using ISPF Version 2, go to "Instructions for Non-ISPF Users" on page 27.

In order to use ISPF with PDF, complete the following steps, as required. Each of these steps is described in more detail in the sections that follow, along with a procedure for verifying successful installation of Interactive Debug.

1. Modify the Foreground Selection Panel to provide the VS FORTRAN Version 2 Interactive Debug option, if the panel does not already include this option.
2. Modify the Help Panel to provide the VS FORTRAN Version 2 Interactive Debug option, if the panel does not already include this option.
3. Modify the compilation EXEC to accommodate VS FORTRAN Version 2 programs.
4. Modify the invocation EXEC to invoke ISPF/PDF.
5. Modify the selection EXEC to reflect any changes in the names of text or load libraries.

Note: In order to use Interactive Debug through ISPF/PDF, you need AFFLOADF with a filetype of TEXT. (AFFLOADF is the ISPF/PDF invocation module shipped with the product.)

1. Modifying the Foreground Selection Panel

Using ISPF/PDF Edit, modify foreground selection panel ISRFPA, located in your site's ISPF/PDF panel MACLIB (normally ISRPLIB).

Do **one** of the following steps:

- **Change:**

  1. Any option at the top part of the panel to specify VS FORTRAN Version 2 Interactive Debug.
  2. The corresponding numbered line at the bottom part of the panel to specify AFFFP11C.

  For example, as shown in Figure 11 on page 21, 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 (type the data in uppercase).

- **Add:**

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

     to the list of options at the upper part of the panel (where xx is the number of the option).

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

     to the entries at the lower part of the panel.

```
%----------------------  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 11. ISPF/PDF Foreground Selection Panel

2. Modifying the Help Panel

Using ISPF/PDF Edit, follow the procedures in step 1 to either change or add to panel ISR40000, located in your site's ISPF/PDF panel MACLIB (normally ISRPLIB). The corresponding numbered line at the bottom of the panel should specify AFF4B000. See Figure 12.

```
%TUTORIAL ------------- FOREGROUND PROCESSING OPTION ----------------- TUTORIAL
%OPTION  ===>_ZCMD                                                          +
+
%                    -----------------------------------
                     |     FOREGROUND PROCESSING       |
                     -----------------------------------
+
   The foreground processing option allows a processing program to be run
   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 12. ISPF/PDF Foreground Processing Help Panel

3. Modifying the Compilation EXEC

In order to compile VS FORTRAN Version 2 programs in the ISPF/PDF environment, modify ISRFX03 EXEC on your location's ISPF/PDF minidisk, as follows:

Replace the line

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

with

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

4. Building or Modifying the Invocation EXEC

Use an editor to build or modify an EXEC to invoke ISPF/PDF. 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 13 on page 24. This example assumes that you have ISPF/PDF Version 2 as well as ISPF Version 2.

CAUTION: We advise you to use the procedures in this manual to establish your Interactive Debug libraries; they are the only ones supported by Interactive Debug. If you use another approach, such as the ISPF LIBDEF service, you may get unpredictable results.

5. Modifying the Selection EXEC

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

```
/* exec to invoke ISPF */
TRACE ERROR
PARSE UPPER ARG ISPFPARM

/*********************************************************************/
/* 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,   */
/* TABLES AND/OR SKELETON LIBRARIES FROM WHICH YOU PLAN TO OPERATE,  */
/* AS WELL AS FOR THE FILE TO BE USED FOR ISPPROF.  THE ISPDCS       */
/* COMMAND HAS AN OPTIONAL KEYWORD PARAMETER (KEYWORD IS "PDFDCSS")  */
/* WHICH SPECIFIES THE PDF DCSS NAME (IF OMITTED THEN THE DEFAULT PDF */
/* DCSS NAME OF "ISRDCSS" IS USED).                                  */
/*********************************************************************/

/*********************************************************************/
/* LINK TO ISPF AND PDF DISK(S)                                      */
/*********************************************************************/
'CP LINK ISPF2 191 396 RR'
'ACC 396 T'

/*********************************************************************/
/* PERFORM FILEDEFS                                                  */
/* NOTE: PRIVATE PANELS, MSGS, SKELS, TABLES AND PROFILE FILES       */
/*       SHOULD BE PLACED AHEAD OF THE PDF AND ISPF SUPPLIED FILES.  */
/*       FILEMODE MAY NEED TO BE CHANGED DEPENDING ON HOW DISK WAS   */
/*       ACCESSED.                                                   */
/*********************************************************************/
'FILEDEF ISPPROF CLEAR'
'FILEDEF ISPPROF DISK DEFAULTS MACLIB A (PERM '

'FILEDEF ISPPLIB CLEAR'
'FILEDEF ISPPLIB DISK VSF2PLIB MACLIB * (PERM CONCAT'
'FILEDEF ISPPLIB DISK ISRPLIB MACLIB T (PERM CONCAT'
'FILEDEF ISPPLIB DISK ISPPLIB MACLIB T (PERM CONCAT'

'FILEDEF ISPMLIB CLEAR'
'FILEDEF ISPMLIB DISK VSF2MLIB MACLIB * (PERM CONCAT'
'FILEDEF ISPMLIB DISK ISRMLIB MACLIB T (PERM CONCAT'
'FILEDEF ISPMLIB DISK ISPMLIB MACLIB T (PERM CONCAT'

'FILEDEF ISPSLIB CLEAR'
'FILEDEF ISPSLIB DISK ISRSLIB MACLIB T (PERM CONCAT'

'FILEDEF ISPTLIB CLEAR'
'FILEDEF ISPTLIB DISK AFFCMDS TABLE  * (PERM CONCAT'
'FILEDEF ISPTLIB DISK ISRTLIB MACLIB T (PERM CONCAT'
'FILEDEF ISPTLIB DISK ISPTLIB MACLIB T (PERM CONCAT'

'FILEDEF FT05F001 TERMINAL (PERM'
'FILEDEF FT06F001 TERMINAL (PERM'

IF ISPFPARM ¬= '' THEN
  ISPFPARM = 'OPT('ISPFPARM')'

'ISPDCS ISPDCSS2 ISPVM2 PDFDCSS(ISRDCSS2) PANEL(ISR@PRIM) NEWAPPL(ISR)',
                        ISPFPARM
exit rc
```

Figure 13. Sample EXEC to Invoke ISPF/PDF

6. Verifying Availability of Interactive Debug

Use the sample program AFFIVP provided on the product tape. This program computes the diameter, circumference and area of a circle. AFFIVP can be edited, printed, and processed using normal CMS commands.

1. Compile and link edit the sample program using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default, by entering the following:

   ```
   FORTVS2 AFFIVP (SDUMP
   ```

   - If you receive the following messages with no error or warning messages, proceed to step 2.

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

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

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

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

   - If you do not receive these messages or you receive error or warning messages, review your installation procedures to this point. Insure that sufficient DASD file space has been allocated. Repeat any steps if necessary. If you cannot identify the source of the problem by reviewing and repeating the procedure, contact your IBM service representative.

2. Run the load module with VS FORTRAN Version 2 Interactive Debug.

   a. Before invoking ISPF/PDF, make sure that "Instructions for ISPF/PDF Users" on page 20 have been completed.

   b. Invoke ISPF/PDF by typing the name of the EXEC established at your installation. See Figure 13 on page 24.

   c. When the PRIMARY OPTION PANEL appears, select the option that causes the FOREGROUND SELECTION PANEL to be displayed. This is usually option 4.

   d. Find the line specifying VS FORTRAN Version 2 Interactive Debug and enter the associated number. (In the example in Figure 11 on page 21, number 11 was used.) The FOREGROUND VS FORTRAN VERSION 2.3.0 INTERACTIVE DEBUG panel is displayed.

   e. Enter AFFIVP on the line labeled FILE ID, and type DEBUG opposite the DEBUG option. The next panel displayed should be the Interactive Debug panel. However, if any debuggable program's listing is not correctly specified in AFFON, the next panel displayed will be the listings file specification panel, not the Interactive Debug panel. You can enter missing file definitions on the listings file specification panel. Enter END when you are done. For further information, refer to *VS FORTRAN Version 2 Interactive Debug Guide and Reference*, SC26-4223.

      To fully verify successful installation of Interactive Debug you can now issue Interactive Debug commands as described in "A Sample Interactive Debug Session" on page 29. If you choose not to perform this step, type QUIT.

Instructions for Users of ISPF Without PDF

Proceed only if you are using Interactive System Product Facility (ISPF) Version 2 without ISPF Program Development Facility (PDF). If you are not using ISPF Version 2, go to "Instructions for Non-ISPF Users" on page 27.

1. Modifying the IAD Invocation EXEC

The product tape contains the IAD EXEC, which may require modifications for your installation. Before you use the IAD EXEC, please review the following:

1. All of the FILEDEF statements in the IAD EXEC for filenames you may need to change.
2. The ISPDCS statements in the IAD EXEC for names you may need to change.

2. Verifying Availability of Interactive Debug

Use the sample program AFFIVP provided on the product tape. This program computes the diameter, circumference and area of a circle. AFFIVP can be edited, printed, and processed using normal CMS commands.

1. Compile and link edit the sample program using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default, by entering the following:

   ```
   FORTVS2 AFFIVP (SDUMP
   ```

   - If you receive the following messages with no error or warning messages, proceed to step 2.

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

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

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

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

   - If you do not receive these messages or you receive error or warning messages, review your installation procedures to this point. Insure that sufficient DASD file space has been allocated. Repeat any steps, as necessary. If you cannot identify the source of the problem by reviewing and repeating the procedure, contact your IBM service representative.

2. Run the load module with VS FORTRAN Version 2 Interactive Debug.

   a. Before invoking IAD, link to the product minidisk(s) containing ISPF and VS FORTRAN Version 2.

   b. Invoke IAD by typing IAD AFFIVP.

   c. The next panel displayed should be the Interactive Debug panel. However, if the AFFON file does not specify any debuggable program's listing, the next panel displayed will be the listings file specification panel, where you can enter missing file definitions. Enter END when you are done. For further information, refer to *VS FORTRAN Version 2 Interactive Debug Guide and Reference*, SC26-4223.

To fully verify successful installation of Interactive Debug you can now issue Interactive Debug commands as described in "A Sample Interactive Debug Session" on page 29. If you choose not to perform this step, type QUIT.

Instructions for Non-ISPF Users

1. Creating a Run EXEC

Using the System Product editor, create an EXEC named FORTIAD to run a VS FORTRAN Version 2 program with VS FORTRAN Version 2 Interactive Debug by typing:

```
XEDIT FORTIAD EXEC A
```

Code an EXEC as shown in Figure 14. 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 or LOADLIBs during installation, be sure to refer to your new library names in the EXEC.

The sample EXEC shown in Figure 14 has one positional parameter: the name of the program to be run. You can add run-time options after the first parameter.

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

```
GLOBAL TXTLIB VSF2LINK VSF2FORT CMSLIB
```

For more information on link mode versus load mode, see "Specifying the Mode of Run-Time Library Modules" on page 16.

```
/* rexx exec to run VS FORTRAN Version 2 IAD in line mode          */
parse arg name options

/* if you want to run in load mode, then use the following statement */
'GLOBAL TXTLIB VSF2FORT CMSLIB'

/* if you want to run in link mode, replace the above statement with */
/* the following global statement:                                   */
/* 'GLOBAL TXTLIB VSF2LINK VSF2FORT CMSLIB'                          */

'GLOBAL LOADLIB VSF2LOAD'
'FILEDEF AFFON    DISK' name 'INCLUDE A'
'FILEDEF AFFPRINT DISK' name 'PRINT   A'
'LOAD' name '(CLEAR'
'START * DEBUG' options
exit rc
```

Figure 14. Sample FORTIAD EXEC

2. Verifying Availability of Interactive Debug

Use the sample program AFFIVP provided on the product tape. This program computes the diameter, circumference and area of a circle. It can be edited, printed, and processed using normal CMS commands.

1. Compile the sample program using the VS FORTRAN Version 2 compiler with the SDUMP option specified explicitly or by default, by entering the following:

   ```
   FORTVS2 AFFIVP (SDUMP
   ```

   - If you receive the following messages with no error or warning messages; proceed to step 2.

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

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

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

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

   - If you do not receive these messages or you receive error or warning messages, review your installation procedures to this point. Insure that sufficient DASD file space has been allocated. Repeat any steps if necessary. If you cannot identify the source of the problem by reviewing and repeating the procedure, contact your IBM service representative.

2. Using the invocation EXEC you created in "1. Creating a Run EXEC" on page 27, run AFFIVP with VS FORTRAN Version 2 Interactive Debug, as follows:

   a. Invoke Interactive Debug by typing

      ```
      FORTIAD AFFIVP
      ```

   b. You will receive an informational message and several lines of copyright information, followed by a WHERE message, identifying the statement about to be run. This is followed by the IAD prompt FORTIAD, indicating that the FORTRAN has reached the first debugging hook. For example,

      ```
      VS FORTRAN VERSION 2 RELEASE 3 INTERACTIVE DEBUG
      5668-806 (C) COPYRIGHT IBM CORP. 1985, 1987
      LICENSED MATERIALS-PROPERTY OF IBM
      WHERE: CIRCLE.7
      FORTIAD
      ```

      If you are using an AFFON file, you will receive information indicating whether or not the AFFON file processed correctly.

      To fully verify successful installation of Interactive Debug, you can now issue Interactive Debug commands, as described in the following section. If you choose not to perform this option, type QUIT.

A Sample Interactive Debug Session

Running of sample program AFFIVP under Interactive Debug is illustrated in Figure 15, showing input and output for a set of Interactive Debug commands. You can enter the commands and data shown in this example to verify successful installation of Interactive Debug. You can also enter other Interactive Debug commands for further verification. All lines beginning with "=*" indicate commands or data entered on the terminal. However, when you enter the commands or data, do not type the preceding "=*" symbols. (The FORTIAD prompt does not appear, since this log was obtained running under ISPF/PDF.) In line mode (for non-ISPF users) the initial equal signs (=) shown in Figure 15 will not appear.

```
=VS FORTRAN VERSION 2 RELEASE 3 INTERACTIVE DEBUG
=5668-806 (C) COPYRIGHT IBM CORP. 1985, 1987
=LICENSED MATERIALS-PROPERTY OF IBM
=WHERE: CIRCLE.7
=* listsubs
=PROGRAM UNIT                    COMPILER   OPT   HOOKED TIMING
=CIRCLE                         VSF 2.3.0    0      YES    OFF
=DIAM                           VSF 2.3.0    0      YES    OFF
=CIRCUM                         VSF 2.3.0    2      YES    OFF
=AREA                           VSF 2.3.0    3      YES    OFF
=* describe (data pi)
=CIRCLE.DATA:                    REAL*4
=   RANK = 1,  SIZE = 3 ELEMENTS
=   DIM 1:    EXTENT = 3   LBOUND (1),  UBOUND (3)
=CIRCLE.PI:                      REAL*8
=* at diam.entry (step)
=* go
=FTO6F001  ENTER THE VALUE OF THE CIRCLE RADIUS (xxx.xx):
=FTO5F001 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
=FTO6F001  THE DIAMETER OF THE CIRCLE IS  705.34
=FTO6F001  THE CIRCUMFERENCE OF THE CIRCLE IS 2215.89
=FTO6F001  THE AREA OF THE CIRCLE IS  390738.94
=AT: CIRCLE.42
== READY FOR TERMINATION =
=PROGRAM HAS TERMINATED; RC ( 0)
=* quit
```

Figure 15. CMS Interactive Debug Input/Output

Chapter 5. Customizing VS FORTRAN Version 2

This is an optional step. If you wish to proceed, make sure you have completed all steps under Chapter 3, "Installing VS FORTRAN Version 2" on page 11.

Note: If you expect to run VS FORTRAN Version 2 on an XA-mode machine under VM/XA, you must use Assembler H Version 2.

Making the Alternative Math Library Subroutines Available

The alternative math library contains the VS FORTRAN Version 1 standard math subroutines. These subroutines are placed in VSF2MATH by the installation process. When programmers create a program that is ready to run they may choose to have these math subroutines either made a part of their program, along with the compiler-generated code (link mode), or loaded dynamically at run time (load mode). Run-time loading has the advantages of reducing auxiliary storage requirements for programs and decreasing link-edit time. However, it slightly increases execution time.

After installation of the VS FORTRAN Version 2 library, you may update your site's operational procedures to specify the alternative math library for use in load mode or link mode. To select the mode you want, provide an EXEC to issue the appropriate GLOBAL commands.

- To use VSF2MATH in load mode, specify:

```
GLOBAL TXTLIB VSF2MATH VSF2FORT CMSLIB
GLOBAL LOADLIB VSF2LOAD
```

- To use VSF2MATH in link mode:
  - During installation, if you chose to produce a separate link mode text library, then VSF2LINK was created as a separate TXTLIB from VSF2FORT. Thus, during run-time, you must specify both VSF2LINK and VSF2FORT in the GLOBAL statement. Concatenate VSF2LINK ahead of VSF2FORT:

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

  - During installation, if you chose to produce a combined link mode text library, then VSF2LINK was created as a combined TXTLIB with VSF2FORT. Thus, during run-time, do not specify VSF2FORT:

    ```
    GLOBAL TXTLIB VSF2MATH VSF2LINK CMSLIB
    ```

Building Composite Modules

This section provides the procedures necessary to develop or change composite modules. You will find it useful should you ever need to make changes to a composite module after initial installation.

1. If you are putting a composite module in a DCSS, complete all the preparatory steps in "Preparing for Shared Segment (DCSS) Installation" on page 7.
2. Log on to VM. If you are not installing a composite module in a DCSS, skip step 3.
3. If you are installing in a DCSS, you must have class E privileges. 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. Or, if you are installing AFBVRENA (above the line), you will need a 34 megabyte virtual storage size in order to place the shared segment at a starting address of 33 megabytes.

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

4. Link to and access the work disk (that you accessed as your A-disk during installation) in read/write status as file mode A.
5. 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. Make a note of the file mode you use; you will need to specify it later.
6. Invoke the library installation EXEC with the COMPOSITE parameter:

   ```
   I5668805 COMPOSITE
   ```

   As prompted, you can now modify the list of modules that are in the composite modules.
7. After you are through with the EXEC, you will have a new copy of the load library on the work disk, the A-disk. This load library contains your customized composite modules. It also refers to any DCSSs that were built.

   If you built one or more DCSSs, perform the following activities for each one:

   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 4.
   c. Re-access the product disk using the file mode (fm) that you established in step 5.
   d. Run a FORTRAN program using the new library in load mode as shown below.
8. 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
   ```
9. Once you have verified that the new load library works, copy it from the 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
   ```

Putting the Compiler in a DCSS

This section provides you with a step-by-step procedure for installing the compiler in a discontiguous shared segment (DCSS) after initial installation.

1. Complete all the preparatory steps described in "Preparing for Shared Segment (DCSS) Installation" on page 7.
2. Log on to a userid that has E privileges.

3. Define a virtual storage size that exceeds the starting address of the DCSS by at least 3 megabytes. For example, if the DCSS starting address is X'500000' (or 5 megabytes), a minimum of 8 megabytes is needed.

   Note: The 3 megabyte figure depends on machine configuration and is only an approximate value; you may need more space.

4. Link and access the work disk (that you accessed as your A-disk during initial installation) in read/write mode. If you have unloaded the work disk to tape after installation, you must restore it before proceeding by typing:

   ```
   VMFPLC2 LOAD * * A
   ```

5. Link and access the product disk (this is the disk that contains your VS FORTRAN Version 2 libraries and modules) in read/write mode. Establish a file mode (fm) other than A, S, or Y. Make a note of this file mode; you will need to specify it later.

6. 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 shared segment, and be prepared to give the DCSS names.

7. 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 4).
   c. Compile the sample program AFBIVP by issuing the command:

      ```
      FORTVS2 AFBIVP
      ```

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

   a. Access the product disk using the file mode established in step 5.

   b. Perform either of the following steps:

      - Replace the previous compiler with the new compiler by specifying:

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

        where fm is the file mode.
      - Place both the previous compiler and the new compiler on the product disk by specifying:

        ```
        COPY FORTVS2 MODULE A fn = fm
        COPY FORTVS2 MAP A fn = fm
        ```

        where fn is the filename you choose, and fm is the file mode.

Changing Option Defaults

This section will assist you in modifying the IBM-supplied defaults for the following options:

- Compile-time
- Unit Attribute Table (sets up the standard I/O units and file characteristics)
- Run-time
- Custom error-option table

Note that modifications to the first three options could have been done during installation.

For guidelines on invoking macros, refer to Appendix A, page 45.

Changing Compile-Time Option Defaults

For additional information on the VSF2COM macro, see page 46.

Note: If you installed the library-only product, skip this section.

The product tape provides an assemble file ILX0OPTS, which contains a set of defaults for compile-time options. To customize this module, do the following:

1. Restore the compiler text files by either re-accessing the work disk you used during installation, or by loading them from tape by issuing:

   ```
   VMFPLC2 LOAD * * A
   ```

2. Edit the ILX0OPTS ASSEMBLE file:

   ```
   XEDIT ILX00PTS ASSEMBLE fm
   ```

   where fm = A, the file mode of the work disk that you established during installation. Change the options specified on the VSF2COM macro instruction according to your needs.

3. Issue the following commands to access the library containing VSF2COM macro and to assemble module ILX0OPTS:

   ```
   GLOBAL MACLIB VSF2MAC
   ASSEMBLE ILX00PTS
   ```

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

4. Correct any coding errors in ILX0OPTS ASSEMBLE until it assembles without error. Then create a new FORTVS2 MODULE containing the changed ILX0OPTS module by issuing the following commands:

   ```
   GLOBAL TXTLIB CMSLIB
   LOAD IEAXPALL (CLEAR
   INCLUDE LOADORD (CLEAR RESET ILX0CMS
   GENMOD FORTVS2 (STR MAP FROM ILX0CMS
   ```

5. If you installed the compiler in a DCSS, you must re-install it after you have changed any compile-time option defaults. Refer to "Putting the Compiler in a DCSS" on page 33.

Changing the Unit Attribute Table Defaults

For additional information on macros VSF2UAT, VSF2UNIT, and VSF2DCB, see Appendix B, "Customization Macros" on page 45.

CAUTION: If you change the IBM-supplied default DCB values, the existing FORTRAN programs which depend on the original defaults may not work. For more information on DCB values, refer to *VS FORTRAN Version 2 Programming Guide*.

The product tape provides an assemble file AFBCUAT, the Unit Attribute Table. It contains a set of I/O unit number options and DCB value defaults. To customize this module, perform the following steps:

1. Edit the AFBCUAT ASSEMBLE file:

   ```
   XEDIT AFBCUAT ASSEMBLE fm
   ```

   where fm = A, the file mode of the **work** disk that you established during installation. Change the IBM-supplied default values specified on the VSF2UAT, VSF2UNIT, and VSF2DCB macro instructions, according to your needs. See "IBM-Supplied Default Values" and "Examples of Changing Default Values" on page 37.

2. Issue the following commands to access the library containing the VSF2UAT, VSF2UNIT, and VSF2DCB macros and to assemble module AFBCUAT:

   ```
   GLOBAL MACLIB VSF2MAC
   ASSEMBLE AFBCUAT
   ```

   Notes:

   a. VSF2MAC is the name of the macro library generated during initial installation.

   b. If you expect to run VS FORTRAN on an XA-mode machine under VM/XA, specify HASM instead of ASSEMBLE.

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

   ```
   TXTLIB DEL VSF2FORT AFBCUAT
   TXTLIB ADD VSF2FORT AFBCUAT
   ```

   During installation, if you chose to produce a combined link mode text library, issue the following additional commands:

   ```
   TXTLIB DEL VSF2LINK AFBCUAT
   TXTLIB ADD VSF2LINK AFBCUAT
   ```

4. Rebuild composite module AFBVNREN. Because the module AFBCUAT is a required module in the library composite module AFBVNREN, your new AFBCUAT module must be replaced in AFBVNREN. Several other customization tasks also involve modules that are part of AFBVNREN and require that it be rebuilt. Therefore, you can rebuild AFBVNREN now or you can wait until after you have completed all of your customization tasks. For instructions on how to rebuild AFBVNREN, see "Building Composite Modules" on page 32.

IBM-Supplied Default Values: The following macro instructions are provided in the AFBCUAT ASSEMBLE file. It sets up the IBM-supplied default values for the standard I/O units, as well as file characteristics such as the DCB information. Note that the last VSF2DCB macro does not have a label; its set of defaults apply to all units except 5, 6, and 7.

```
AFBCUAT VSF2UAT UNTABLE=99,
                DECIMAL=PERIOD,
                READER=5,
                ERRMSG=6,
                PRINTER=6,
                PUNCH=7

      VSF2UNIT  5,DCBSET=DCBRDR
      VSF2UNIT  6,DCBSET=DCBPRT
      VSF2UNIT  7,DCBSET=DCBPUN

DCBRDR VSF2DCB  SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                SURECFM=F,SULRECL=80,SUBLKSI=80

DCBPRT VSF2DCB  SFRECFM=UA,SFLRECL=133,SFBLKSI=133

DCBPUN VSF2DCB  SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                SURECFM=F,SULRECL=80,SUBLKSI=80

       VSF2DCB  SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                SURECFM=VS,SULRECL=-1,SUBLKSI=800,
                DMAXRE=50

       VSF2UAT  TYPE=FINAL
```

Note: The above format is given for readability purposes. However, if you want to change the AFBCUAT file, remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

Figure 16. Sample 1: IBM-Supplied Unit Attribute Table Macro

Examples of Changing Default Values: The following examples show how you could modify the IBM-supplied defaults for your own environment. You can alter instructions by striking over existing data or you can add more VSF2UNIT and VSF2DCB macro instructions.

Example 1

In this example, we have added an instruction to assign unit 11 to a direct file with a maximum of 135 records. The rest of the DCB values for the file will default to the values in the VSF2DCB macro, which you will find in Appendix B, "Customization Macros." We have also modified the default values for the non-labelled (last) VSF2DCB instruction, which will apply to all units except 5, 6, 7, and 11.

```
AFBCUAT VSF2UAT
        VSF2UNIT  5,DCBSET=DCBRDR
        VSF2UNIT  6,DCBSET=DCBPRT
        VSF2UNIT  7,DCBSET=DCBPUN
        VSF2UNIT  (11),DCBSET=USERDCB

DCBRDR  VSF2DCB   SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                  SURECFM=F,SULRECL=80,SUBLKSI=80

DCBPRT  VSF2DCB   SFRECFM=UA,SFLRECL=133,SFBLKSI=133

DCBPUN  VSF2DCB   SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                  SURECFM=F,SULRECL=80,SUBLKSI=80

USERDCB VSF2DCB   DMAXRE=135

        VSF2DCB   SFRECFM=U,SFLRECL=800,SFBLKSI=800,
                  SURECFM=VS,SULRECL=-1,SUBLKSI=800,
                  DMAXRE=100

        VSF2UAT   TYPE=FINAL
```

Note: The above format is given for readability purposes. However, if you want to change the AFBCUAT file, remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

Figure 17. Sample 2: Modified Unit Attribute Table Macro

Note that VSF2UAT, VSF2UNIT, and VSF2DCB must all be coded, in that order, followed by the TYPE=FINAL statement.

Example 2

If you wanted to change the default standard I/O unit values for READER, ERRMSG, PRINTER, and PUNCH to 1, 2, 3, 4, respectively, you would modify the IBM-supplied macros as follows:

```
AFBCUAT VSF2UAT READER=1, ERRMSG=2, PRINTER=3, PUNCH=4

          VSF2UNIT   1,DCBSET=DCBRDR
          VSF2UNIT   2,DCBSET=DCBTERM
          VSF2UNIT   3,DCBSET=DCBPRT
          VSF2UNIT   4,DCB=DCBPUN
             .
             .
             .
```

Figure 18. Sample 3: Modified Unit Attribute Table Macro

Example 3

This example highlights the changes you should make to run a FORTRAN program with OS/VS file characteristics. To change the default file characteristics from CMS to OS/VS, the only macro instruction you need to code is a VSF2DCB macro instruction, as shown below. See Figure 16 on page 37 for the original IBM-supplied default values.

```
AFBCUAT VSF2UAT
                    .
                    .
                    .

DCBRDR VSF2DCB  SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                SURECFM=F,SULRECL=80,SUBLKSI=80

DCBPRT VSF2DCB  SFRECFM=UA,SFLRECL=133,SFBLKSI=133

DCBPUN VSF2DCB  SFRECFM=F,SFLRECL=80,SFBLKSI=80,
                SURECFM=F,SULRECL=80,SUBLKSI=80

       VSF2DCB  SFRECFM=U,SFLRECL=800,SFBLKSI=800,   <----OS/VS characteristics
                SURECFM=VS,SULRECL=-1,SUBLKSI=800,
                DMAXRE=50

       VSF2UAT  TYPE=FINAL
```

Note: The above format is given for readability purposes. However, if you want to change the AFBCUAT file, remember to add the necessary continuation flags in column 72, and to begin continued lines in column 16.

Figure 19. Sample 4: Modified Unit Attribute Table Macro

Changing Run-Time Option Defaults

For additional information on the VSF2PARM macro, see page 54.

The product tape provides a text file AFBVGPRM, which contains a set of global defaults for run-time options. To customize this module, perform the following steps:

1. Edit the AFBVGPRM ASSEMBLE file:

   ```
   XEDIT AFBVGPRM ASSEMBLE fm
   ```

   where fm = A, the file mode of the work disk that you established during installation. (If the run-time option defaults have never been changed, the above command gives you an empty file, and you must code the VSF2PARM macro instruction.) Change the options specified on the VSF2PARM macro instruction according to your needs.

2. Issue the following commands to access the library containing the VSF2PARM macro and to assemble module AFBVGPRM:

```
GLOBAL MACLIB VSF2MAC
ASSEMBLE AFBVGPRM
```

Notes:

a. VSF2MAC is the name of the macro library generated during installation.

b. If you are running on a XA-mode machine under VM/XA, specify HASM instead of ASSEMBLE.

3. 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
```

If during installation you chose to produce a combined link mode text library, issue the following additional commands:

```
TXTLIB DEL VSF2LINK AFBVGPRM
TXTLIB ADD VSF2LINK AFBVGPRM
```

4. Rebuild composite module AFBVNREN. Because the module AFBVGPRM is a required module in the library composite module AFBVNREN, your new AFBVGPRM module must be replaced in AFBVNREN. Several other customization tasks also involve modules that are part of AFBVNREN and require that it be rebuilt. Therefore, you can rebuild AFBVNREN now or you can wait until after you have completed all of your customization tasks. For instructions on how to rebuild AFBVNREN, see "Building Composite Modules" on page 32.

Customizing the Error Option Table

For additional information on the VSF2UOPT macro, see page 58.

This is the only option that cannot be modified during installation.

The product tape provides a text file AFBUOPT, which is an error option table that specifies the actions that can be taken when an error occurs during the running of a FORTRAN program:

- 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.

Note: Customization applies to the permanent copy of the error option table in the VS FORTRAN Version 2 Library. Each individual FORTRAN program can also make run-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* for more information.

To customize AFBUOPT, do the following:

1. Edit the AFBUOPT ASSEMBLE file:

   ```
   XEDIT AFBUOPT ASSEMBLE fm
   ```

   where fm = A, the file mode of the work disk you established during installation. (If the error option defaults have never been changed, the above command gives you an empty file, and you must code the VSF2UOPT macro instruction.) Change the options specified on the VSF2UOPT macro instruction(s) in this file.

2. Issue the following commands to access the library containing the VSF2UOPT macro and to assemble module AFBUOPT:

   ```
   GLOBAL MACLIB VSF2MAC
   ASSEMBLE AFBUOPT
   ```

   Notes:

   a. VSF2MAC is the name of the macro library generated during installation.

   b. If you are running on an XA-mode machine under VM/XA, specify HASM instead of ASSEMBLE.

3. 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
   ```

   If during installation you chose to produce a combined link mode text library, issue the following additional commands:

   ```
   TXTLIB DEL VSF2LINK AFBUOPT
   TXTLIB ADD VSF2LINK AFBUOPT
   ```

4. Rebuild composite module AFBVNREN. Because the module AFBUOPT is a required module in the library composite module AFBVNREN, your new AFBUOPT module must be replaced in AFBVNREN. Several other customization tasks also involve modules that are part of AFBVNREN and require that it be rebuilt. Therefore, you can rebuild AFBVNREN now or you can wait until you have completed all of the customization tasks you plan to do and rebuild AFBVNREN only once. For instructions on how to rebuild AFBVNREN, see "Building Composite Modules" on page 32.

Appendix A. Program Materials

This appendix identifies the basic machine-readable materials available from IBM Software Distribution (ISD).

Compiler, Library, and Interactive Debug Product (5668-806)

Figure 20 identifies characteristics of the various distribution tapes for basic VS FORTRAN Version 2 for the complete product (Compiler, Library, and Interactive Debug). Figure 21 shows the tape format.

Medium	Track/ Density	Feature Number	External Label
Magnetic Tape	9/1600	5006	VM BASIC MATERIAL
Magnetic Tape	9/6250	5007	VM BASIC MATERIAL
3480 Cartridge	18/38000	5872	VM BASIC MATERIAL

Figure 20. Distribution Tape (Basic) Identifiers: Complete Product

File 1	Product Identifier File Installation EXEC I5668806
File 2	Memo to Users
File 3	Default Files Library Installation EXEC I5668805 IAD Installation EXEC D5668805
File 4	Compiler Text Files and Macros
File 5	Library Text Files and Macros
File 6	IAD Text Files
File 7	Panel Library for IAD (VSF2PLIB MACLIB)
File 8	IAD, with ISPF, Message Library (VSF2MLIB MACLIB)
File 9	IAD Help Files for Line Mode
File 10	IAD, with ISPF/PDF, Invocation (AFFFX11 EXEC) IAD, with ISPF, Invocation (IAD EXEC)
File 11	Verification EXEC V5668806 Sample Program for FORTRAN (AFBIVP FORTRAN)
File 12	Sample Program for IAD (AFFIVP FORTRAN)

Figure 21. Distribution Tape Format: Complete Product

Library-only Product (5668-805)

Figure 22 identifies characteristics of the various distribution tapes for basic VS FORTRAN Version 2 for the library-only product. Figure 23 shows the tape format.

Medium	Track/ Density	Feature Number	External Label
Magnetic Tape	9/1600	5002	VM BASIC MATERIAL
Magnetic Tape	9/6250	5003	VM BASIC MATERIAL
3480 Cartridge	18/38000	5872	VM BASIC MATERIAL

Figure 22. Distribution Tape (Basic) Identifiers: Library Only

File 1	Product Identifier File Installation EXEC I5668805
File 2	Memo to Users
File 3	Default Files
File 4	Library Text Files and Macros

Figure 23. Distribution Tape Format: Library Only

Appendix B. Customization Macros

This appendix is to be used in conjunction with Chapter 5, "Customizing VS FORTRAN Version 2" on page 31.

Guidelines for Invoking Macros

1. Column 1 must be blank.
2. The macro name may appear anywhere before column 72 but must precede the options by at least one blank.
3. The options are separated by commas, with no intervening blanks, and may be continued on any number of lines as long as column 72 contains a non-blank character and the data on the following line begins in column 16.
4. A comma must follow the last option on a line when a continuation line follows.
5. You only need to code the options whose default values you wish to change.

VSF2COM: Changes Compile-Time Option Defaults

The VSF2COM macro allows you to change the IBM-supplied default values for most of the compile-time options. The default values you assign will be assumed if the user does not override them.

The macro options that set the compile-time defaults listed in this manual are designed to be used at installation. They have a different format than the corresponding compiler options in the *Programmer's Guide*.

Note: There are no IBM-supplied default values for the compile-time options AUTODBL, CI, DC, DIRECTIVE, ICA, and VECTOR, and therefore these options *cannot* be changed at installation time. They can, however, be changed at compile time.

Each row in Figure 24 shows incompatible compile-time options. If any of these combinations of values are specified, the VSF2COM macro instruction will not assemble properly.

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

Figure 24. Incompatible Compile-Time Options

Syntax of VSF2COM Macro

```
VSF2COM
     SYSTEM=CMS
     [,ADJ=IL(DIM) | IL(NODIM)]
     [,CHARLEN=number | 500]
     [,DATE=MDY | YMD]
     [,DBCS=DBCS | NODBCS]
     [,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]
     [,SAA=SAA | NOSAA]
     [,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 values are underlined in the following option list. If an option is not specified, its default value will be used, with the exception of the SYSTEM option, which must always be specified.

SYSTEM = CMS
must **always** be specified.

ADJ = IL(DIM) | IL(NODIM)
specifies whether the code for adjustable-dimensioned arrays is to be placed inline—IL(DIM), or computations are to be done via library call—IL(NODIM). Inline placement may result in faster run time. The library call may result in slower run time, but it checks whether the lower bound is greater than the upper bound. If it is, message AFB187I will be issued. 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.

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

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

DBCS = DBCS | NODBCS
specifies whether or not the source file contains double byte characters in symbolic names or in character constants. DBCS indicates that the source file may contain double byte characters; NODBCS indicates that it does not.

FIPS = S | F | NOFIPS
specifies whether or not to flag standard language, and, if so, specifies 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. For example, if you want to suppress all messages that are warning or informational, specify FLAG = E.

IGNORE = DISABLED | ENABLED
specifies whether the IGNORE vector directive will be made available to the user. Users will have access to the IGNORE directive only if the IGNORE installation option is ENABLED. The IGNORE directive allows the applica-

tion programmer to specify that certain vectorization dependencies do not exist. For further details, see *VS FORTRAN Version 2 Programming Guide.* Note that there is no corresponding compile-time option.

INSTERR = NOLIST | LIST
specifies whether 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 1977 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 reentrant object code is to be generated by the compiler.

OBJID = GOSTMT | NOGOSTMT
specifies whether internal statement numbers (for traceback purposes) are to be generated for a call sequence to a subprogram.

OBJLIST = LIST | NOLIST
specifies whether the object module listing is to be produced.

OBJPROG = OBJECT | NOOBJECT
specifies whether 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 run time, but usually a longer compile time, use OPTIMIZE(2) or (3).

PUNCH = DECK | NODECK
: specifies whether the object module is to be produced in card-image format. If DECK is specified, it requires a punch output file.

SAA = SAA | NOSAA
: specifies whether the compiler is to flag language elements that are not part of the FORTRAN System Application Architecture (SAA) Common Programming Interface (CPI). SAA causes flagging to be performed; NOSAA disables flagging.

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

SORLIST = SOURCE | NOSOURCE
: specifies whether the source listing is to be produced.

SORTERM = TERMINAL | NOTERMINAL
: specifies whether error messages and compiler diagnostics are to be written on the terminal or a SYSTERM file.

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

SORXREF = XREF | NOXREF
: specifies whether 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 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.

SYMDUMP = SDUMP | NOSDUMP
: specifies whether symbol table information is to be generated in the object module and in the object module listing. If SYMDUMP=SDUMP is coded, the SDUMP suboption ISN will be in effect. This specifies that SDUMP tables be generated using internal statement numbers.

TEST = TEST | NOTEST
specifies whether to create input for VS FORTRAN Version 2 Interactive Debug symbol table information. TEST overrides any optimization level above OPTIMIZE(0), and adds run-time overhead.

TRMFLG = TRMFLG | NOTRMFLG
presents the statement in error and the diagnostic message together, whenever possible, on your terminal.

VSF2UAT: Changes I/O Unit Number Option Defaults

The first form of the VSF2UAT macro allows you to specify default values for I/O information that is required by the run-time input/output routines of the VS FORTRAN Version 2 Library.

Syntax of VSF2UAT Macro: Statement Form

```
AFBCUAT VSF2UAT
    [DECIMAL=PERIOD | COMMA]
    [,PUNCH=unit1 | 7 ]
    [,ERRMSG=unit2 | 6 ]
    [,PRINTER=unit3 | 6 ]
    [,READER=unit4 | 5 ]
    [,UNTABLE=number | 99]
```

You may code as many VSF2UAT statements as you need. However, you must use the following form of VSF2UAT as the final macro instruction when you change the I/O unit options and default values. See "Examples of Changing Default Values" on page 37.

Syntax of VSF2UAT Macro: Final Statement

```
    VSF2UAT  TYPE=FINAL
```

The IBM-supplied default values are underlined in the following option list. If an option is not specified, its default value will be used.

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

PUNCH = *unit1* | 7
specifies, for LANGLVL(66) only, the standard I/O unit number to be used with the PUNCH statement to send data to the card punch. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for ERRMSG, PRINTER, or READER.

ERRMSG = *unit2* | 6
specifies the standard I/O unit number to be used for the error messages generated by VS FORTRAN. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for PUNCH or READER.

PRINTER = *unit3* | **6**
specifies the standard I/O unit number to be used with the PRINT statement, and with any WRITE statement specifying an installation-dependent form of the unit. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as that specified for PUNCH and READER; it can be the same number specified for ERRMSG.

READER = *unit4* | **5**
specifies the standard I/O unit number to be used with any READ statement, specifying an installation-dependent form of the unit. The number specified must be between 0 and 99 or the value specified for the UNTABLE option, for UNTABLE values less than or equal to 99. It must not be the same as the number specified for either PUNCH, ERRMSG, or PRINTER.

UNTABLE = *number* | **99**
specifies the largest unit number you can include in a VS FORTRAN program. It can be specified as any integer between 8 and 2000.

TYPE = FINAL
is the required last statement of the VSF2UAT macro.

Note: VS FORTRAN Version 2 refers to PUNCH, ERRMSG, PRINTER, and READER as the standard I/O units.

VSF2UNIT: Identifies I/O Units to Have DCB Defaults

The VSF2UNIT macro allows you to specify a single unit, or group of units, that are to be assigned DCB default values. It is used in conjunction with the VSF2DCB macro.

Syntax of VSF2UNIT Macro

VSF2UNIT
unitno | (*unitno* **[**,*qty*])
,DCBSET = *label*

unitno
specifies the unit number, or the first in a series of consecutive unit numbers, that is to have DCB default value(s) assigned.

qty
specifies, if there are more than one, the number of consecutive unit numbers, beginning with *unitno*, that are to have DCB default values assigned.

DCBSET=*label*
specifies the label that is applied to the unit(s) designated by *unitno*|(*unitno*[,*qty*]). The VSF2DCB macro must have a corresponding label.

VSF2DCB: Identifies DCB Default Information

The VSF2DCB macro allows you to specify DCB default information for the I/O unit(s) identified by the **DCBSET**=*label* option of the VSF2UNIT macro.

Syntax of VSF2DCB Macro

```
[label] VSF2DCB
    [,SFBLKSI=number | 80]
    [,SUBLKSI=number | 800]
    [,SFLRECL=number | 80]
    [,SULRECL=number. | -1]
    [,SFRECFM=char | F]
    [,SURECFM=char | VS]
    [,DMAXRE=number | 50]
```

label
specified in macro VSF2UNIT to identify the I/O unit(s) that are to be assigned DCB default values.

If *label* is omitted, the DCB data is assigned to all units defined in the default table by the VSF2UAT macro, but which have not been defined by the VSF2UNIT macro. If any of the units defined in the attribute table do not have their own associated DCB set coded, you must provide a VSF2DCB macro **without a label** to apply defaults to these units.

SFBLKSI = *number* | **80**
specifies the block size for sequential formatted files. *Number* is an integer expression of length 4; valid range of the blocksize is from 1 to 32760.

SUBLKSI = *number* | **800**
specifies the block size for sequential unformatted files. *Number* is an integer expression of length 4; valid range of the blocksize is from 1 to 32760.

SFLRECL = *number* | **80**
specifies the data length for sequential formatted files. *Number* is an integer expression of length 4; valid range is from 1 to 32760, or -1, which specifies an unlimited record length.

SULRECL = *number* | **-1**
specifies the data length for sequential unformatted files. *Number* is an integer expression of length 4; valid range is from 1 to 32760, or -1, which specifies an unlimited record length.

SFRECFM = *char* | **F**
specifies the record format for sequential formatted files. The value of *char* must be F, FA, FB, FBA, V, VA, VB, VBA, U, or UA. For more information on I/O, see the VS FORTRAN Version 2 Programming Guide.

SURECFM = *char* | **VS**
specifies the record format for sequential unformatted files. The value of *char* must be F, FA, FB, FBA, V, VA, VB, VBA, VS, VBS, U, or UA. For more information on I/O, see the VS FORTRAN Version 2 Programming Guide.

DMAXRE = *number* | **50**
specifies the number of records in a direct file. It is only valid for new DASD files; if specified for an existing file, it will be ignored. *Number* is an integer expression of length 4. **DMAXRE** is equivalent to the XTENT option on the FILEDEF command.

VSF2PARM: Changes Run-Time Option Defaults

The VSF2PARM macro allows you to change the IBM-supplied default values for run-time options. The default values you assign will be assumed if the user does not override them.

VSF2PARM can also be used to create a local table AFBVLPRM, which supplies options that are used only for a specific program. AFBVLPRM is discussed in *VS FORTRAN Version 2 Programming Guide*.

Syntax of VSF2PARM Macro

```
VSF2PARM
    SCOPE = GLOBAL
    [,ABSDUMP | NOABSDUMP]
    [,DEBUG | NODEBUG]
    [,DEBUNIT | NODEBUNIT]
    [,IOINIT | NOIOINIT]
    [,OCSTATUS | NOOCSTATUS]
    [,SPIE | NOSPIE]
    [,STAE | NOSTAE]
    [,XUFLOW | NOXUFLOW]
```

The IBM-supplied default values are underlined in the following option list. If an option is not specified, its default will be used, with the exception of the SCOPE option, which must always be specified.

SCOPE = GLOBAL
required to replace the global run-time options table AFBVGPRM, which supplies default values for all users of the VS FORTRAN Version 2 Library.

There is no default value for this option. Thus SCOPE=GLOBAL must always be specified.

ABSDUMP | NOABSDUMP
specifies whether 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 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 FORTRAN unit numbers will be treated as if connected to a terminal device.

DEBUNIT

passes a list of FORTRAN unit numbers to the running environment. The specified unit numbers will be treated as if they were connected to a terminal device, so that the Interactive Debug command TERMIO can be used to control whether the I/O is done by IAD or the library. 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 s1-s2, where both s1 and s2 are unit numbers, and the ending unit number, s2, is not less than the starting number, s1.

The unit numbers specified must be one- to four-digit numbers within the range of numbers allowed on your system, as specified by the UNTABLE option of the VSF2UAT macro. See page 52 for information on the UNTABLE option.

NODEBUNIT

suppresses treating FORTRAN unit numbers as if connected to a terminal device.

IOINIT | NOIOINIT

specifies whether the normal initialization for I/O processing will occur during initialization of the run-time environment.

IOINIT

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

NOIOINIT

suppresses initialization for I/O processing. This means:

- The error message unit will not be opened during initialization of the run-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 FILEDEF statements are not given.)
- 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.

OCSTATUS | NOOCSTATUS

specifies whether file existence will be checked during the running of OPEN statements, whether files are deleted from their storage media, and whether files that have been closed can be reconnected without an OPEN statement.

OCSTATUS

specifies:

1. file existence will be checked for consistency with the OPEN statement specifiers STATUS='OLD' and STATUS='NEW'.
2. file deletion will occur when the CLOSE statement specifier STATUS='DELETE' is given (on devices which allow deletion).

3. a preconnected file will be disconnected when a CLOSE statement is given or when another file is opened on the same unit. It can be reconnected only by an OPEN statement when there is no other file currently connected to that unit.

NOOCSTATUS
specifies:

1. file existence will not be checked for consistency with the OPEN statement specifiers STATUS='OLD' and STATUS='NEW'.
2. file deletion will not occur when the CLOSE statement specifier STATUS='DELETE' is given.
3. a preconnected file will be disconnected when a CLOSE statement is given or when another file is opened on the same unit. It can be reconnected by a sequential READ or WRITE, BACKSPACE, OPEN, REWIND, or ENDFILE statement when there is no other file currently connected to that unit.

SPIE | NOSPIE
specifies whether the run-time environment will take control when a program interrupt occurs.

SPIE
specifies that the run-time environment take control when a program interrupt occurs.

NOSPIE
specifies that the run-time environment does not take control when a program interrupt occurs.

If you specify NOSPIE, various run-time functions, that depend on a return of control after a program interrupt, are not available. 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-running 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 the VS FORTRAN run-time environment gains control at the time of the abend.

Recommendation

If you are using the DEBUG option, specify SPIE to avoid termination of Interactive Debug when an interrupt occurs.

STAE | NOSTAE
specifies whether the run-time environment will take control if an abnormal termination occurs.

STAE
specifies that the run-time environment will take control when an abnormal termination occurs.

NOSTAE
specifies that the run-time environment does not take control when an abnormal termination occurs. If NOSTAE is specified, abnormal termination is handled by the operating system rather than by the VS FORTRAN run-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 run-time environment or by the debugging device cause system abends rather than VS FORTRAN messages. For example, some errors that occur during running of an OPEN statement result in a system abend rather than the printing of message AFB219I, which allows the program to possibly continue running.

Recommendation

If you are using the DEBUG option, specify STAE to avoid termination of Interactive Debug when an interrupt occurs.

XUFLOW | NOXUFLOW
specifies whether 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: Customizes the Error Option Table

The VSF2UOPT macro allows you to customize the error option table as follows:

- Add new error messages to the table, without changing existing ones, by coding the **required macro instruction,** followed by an END statement.
- Change existing error messages in the table, with or without adding new ones, by coding the **required macro instruction,** followed by the necessary number of **optional macro instructions,** followed by an END statement.

For information on IBM-supplied error messages, refer to "Extended Error-Handling Subroutines and Error Option Table" in *VS FORTRAN Version 2 Language and Library Reference.*

Syntax of VSF2UOPT Required Macro Instruction

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

ADDNTRY=*n*

specifies the number of new error message numbers to be added to the error option table. Additional error message numbers will begin at 302 and continue sequentially, up to a maximum of 899, for a maximum of 598 new messages. If you want to change existing messages but do not want to add new ones, omit ADDNTRY=n.

n

is a positive integer from 1 to 598.

Syntax of VSF2UOPT Optional Macro Instruction

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

The MSGNO option must always be specified. The default values of the five options INFOMSG, IOERR, MODENT, PRTBUF, and TRACBAK vary according to the following conditions:

- If the value of MSGNO specifies an IBM-supplied message number, and *none* of the five options is changed, then the default values are found in "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2 Language and Library Reference*.
- If either
  - the value of MSGNO specifies an IBM-supplied message number, and *one or more* of the five options is changed, or
  - the value of MSGNO specifies a new message number,

  then the default values for the *unspecified* options are:

```
INFOMSG    NO
IOERR      NO
MODENT    YES
PRTBUF     NO
TRACBAK   YES
```

MSGNO = (*ermsno*[,*qty*])
specifies which error messages are affected by the default changes.

ermsno
specifies either one message number, or the first error message number in a series of consecutive numbers.

qty
specifies the number of consecutive error message numbers, beginning with *ermsno*, if there are more than one.

For example, if the option is coded MSGNO=(153), then the default values for message 153 will be changed. If the option is coded MSGNO=(153,4), then the default values for messages 153 through 156 will be changed.

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 value of MSGNO specifies an IBM-supplied message number, the default value for this option is listed in "Extended Error-Handling Subroutines and Error Option Table" of *VS FORTRAN Version 2 Language and Library Reference*. If the value of MSGNO specifies a new message number, 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. Running 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.

IOERR = YES | NO
specifies whether this error message represents an I/O error for which error counting is to be suppressed when an ERR or IOSTAT option is given on the I/O statement.

YES
specifies that if an ERR or IOSTAT option is given, the occurrence of the error is not to be counted toward the maximum number specified by the ALLOW option 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 options are honored.

NO
specifies that the error occurrence is to be counted toward the maximum number of errors allowed.

MODENT = YES | NO
specifies whether 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.

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 value of MSGNO specifies an IBM-supplied message number, the default value for this option is listed in the chapter "Extended Error-Handling Subroutines and Error Option Table" in *VS FORTRAN Version 2 Language and Library Reference*. If the value of MSGNO specifies a new message number, the default value is 5.

PRTBUF = YES | NO
specifies whether 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.

TRACBAK = YES | NO
specifies whether 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.

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 an option 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 AFBVNREN. Then, 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.

VSF2AMTB: Customizes an Auxiliary Error Option Table

The VSF2AMTB macro allows you to customize an auxiliary error option table that can be used by a product other than VS FORTRAN Version 2 to make use of the error handling facility of VS FORTRAN Version 2. Note that this auxiliary error option table is different from the regular error option table used only by VS FORTRAN Version 2.

To use the VSF2AMTB macro, do the following:

1. Define the name and scope of the table by coding the **first form** of the macro instruction. The syntax is shown below.
2. Define the table contents for individual error message entries by following the **first form** of the macro instruction with the necessary number of the **second form** of the macro instruction, one for each message you wish to create, followed by an END statement.
3. Assemble the necessary macros after you invoke them. This will produce your auxiliary error option table, with a name of *pid*UOPT (where *pid* is the auxiliary product identifier you supplied on the **first form** of the 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.

Syntax of VSF2AMTB First Form

```
VSF2AMTB
    COMPID = pid,
    MSGNUM1 = firstnum,
    MSGNUM2 = lastnum
```

Note: There are no default values for any of the three options; they must always be specified.

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. The minimum value is 10000.

MSGNUM2=*lastnum*
specifies the ending number of the table. The maximum value is 19999.

Syntax of VSF2AMTB Second Form

```
VSF2AMTB
    MSGNO = (ermsno[,qty])
    [,ALLOW = errs]
    [,INFOMSG = YES | NO]
    [,MODENT = YES | NO]
    [,PRINT = prmsg]
    [,PRTBUF = YES | NO]
    [,TRACBAK = YES | NO]
    [,USREXIT = exitname]
```

The option list is identical to that of VSF2UOPT, beginning on page 59. Note, however, that VSF2AMTB does *not* have an IOERR option.

If you omit either of the following

- a macro instruction for any error message whose number is in the auxiliary table range you specified in the first macro instruction, or
- any individual option in a particular macro instruction,

then the default values are:

```
ALLOW       10
INFOMSG     NO
MODENT      YES
PRINT        5
PRTBUF      NO
TRACBAK     YES
USREXIT     no user exit taken
```

Appendix C. Composite Modules

This appendix is to be used in conjunction with "Building Composite Modules" on page 32.

Why They are Used

If programmers choose run-time loading of library modules (load mode), each module is loaded the first time it is used, unless it has been previously loaded. Because run-time performance suffers if a large number of library modules are individually loaded, they are combined into composite modules.

As part of its initialization procedure in load mode, the run-time library loads the composite modules listed in Figure 25 on page 66. 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 customize the composite modules by adding or deleting optional modules. For example, if direct access and keyed access are not normally used at your site, you can reduce the size of the composite modules by deleting the applicable optional modules. The direct access and keyed access I/O modules would then have to be loaded individually when they are needed.

Characteristics of the Composite Modules

Name	Can Be Installed In a DCSS	Used for Non-XA Support	Used for XA Support	Location In Relation to 16M line (XA only)	Contents
AFBVNREN	No	Yes	Yes	Above	Nonreentrant library modules, including the library common work area and various system services routines.
AFBVRENA	Yes	No	Yes	Above	Reentrant library modules
AFBVRENB	Yes	No	Yes	Below	Reentrant library modules
AFBVRENC	Yes	Yes	No	Not Applicable	Reentrant library modules

Figure 25. Composite Module Characteristics

Customization Tips

- Because AFBVNREN contains the nonreentrant modules, it must be loaded into your virtual machine for each run 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 shared 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.
- Each library module not in the applicable composite module is loaded from the VSF2LOAD library when the module is first referenced during running.
- If you select some optional modules for inclusion in a DCSS or if you delete any of them from a composite module, they will continue to be included in the "Principal Text" library.

Tables of Required and Optional Modules

Composite Module AFBVNREN

Module	Approx. Size	Default Set	Function
AFBCLBC0	15F0	X	Library common work area
AFBCLOAD	4F8	X	Loader
AFBCUAT	A28	X	Unit attribute table
AFBCVIO$	A8	X	Internal linkage routine
AFBUOPT	690	X	Error option table
AFBVBLN$	58	X	Internal linkage routine
AFBVCNI$	58	X	Internal linkage routine
AFBVCNO$	58	X	Internal linkage routine
AFBVCOM$	58	X	Internal linkage routine
AFBVCOM2	10A0	X	Initialization/termination
AFBVCVT$	310	X	Internal linkage routine
AFBVDEB$	58	X	Internal linkage routine
AFBVDIO$	60	X	Internal linkage routine
AFBVDYN$	38	X	Internal linkage routine
AFBVEMG$	98	X	Internal linkage routine
AFBVERE$	58	X	Internal linkage routine
AFBVERS$	80	X	Internal linkage routine
AFBVFNTH	9F8	X	Program interrupt handler
AFBVGPRM	98	X	Default run-time options
AFBVIAD$	58	X	Internal linkage routine
AFBVIIO$	60	X	Internal linkage routine
AFBVINI$	58	X	Internal linkage routine
AFBVKIO$	60	X	Internal linkage routine
AFBVLOC$	58	X	Internal linkage routine
AFBVPARM	7C0	X	Run-time options processor
AFBVPOS$	58	X	Internal linkage routine
AFBVSPIE	170	X	Interrupt interceptor
AFBVSTA$	138	X	Internal linkage routine
AFBVSTAE	158	X	Abend control routine
AFBVTRC$	58	X	Internal linkage routine
Total	**5C80**	**30**	

Figure 26. Required Modules for AFBVNREN

Module	Approx. Size	Default Set	Function
AFBCRNAM	0		DCSS name list
AFBDIOCP	1E0		Define file (LANGLVL 66)
AFBDSPAP [5]	548		Dimension calculator
AFBIBCOP [4]	920		Pre-VS FORTRAN interface
AFBLDFIP [5]	490		List-directed I/O
AFBNAMEP [5]	3C8		Namelist I/O
AFBSDUMQ	3130		SDUMP subroutine
AFBTFORP	148		Debugging Interface
AFBVASGP [7]	2238		DBCS assignment processor
AFBVBALG	618		Boundary alignment routine
AFBVDBUP	1460		Debugging packet
AFBVDEBU	208		DEBUNIT parameter processor
AFBVDUMQ	700		DUMP/PDUMP subroutine
AFBVINTH	520		Vector program interrupt handler
AFBVIONP	13F0		Namelist I/O
AFBVLOCA	6D8		Statement number locator
AFBVMOPP	4C0		Extended error handling
AFBVPOSA	39F0		Post ABEND processor
AFBVSCOP [6]	6B0		Pre-Release 4 interface
AFBVSPAP	590		Dimension calculator
AFBVSPIP	510		Dynamic spill area processor
AFBVUNIN	208		Unnormalized operand interrupt handler
AFBVVINI	228		Vector initialization

Figure 27. Optional Modules for AFBVNREN

[4] 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.

[5] 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.

[6] Module AFBVSCOP is used when running object decks produced by the VS FORTRAN Version 1 compiler from prior to Release 4. It is needed for formatted and unformatted I/O and for initialization from a main program or from a subroutine with character arguments.

[7] Included in AFBVASGP are AFBDBGVB, AFBDBMOV, AFBDBMVE, AFBDBPAD, AFBDBTRC, and AFBDBTRT. The size of AFBVASGP also incorporates the sizes of the other modules.

Composite Module AFBVRENA (XA Mode Only)

Module	Approx. Size	Default Set	Function
AFBCAREN	1A8	X	Internal linkage module
AFBCSTIO	310	X	Standard I/O unit initialization
AFBVGMFM	250	X	GETMAIN/FREEMAIN
AFBVRDCB	2A0	X	DCB attributes resolution
AFBVTRMF	D0	X	File closing at termination
Total	**A78**	**5**	

Figure 28. Required Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBCCPTP	A8	X	CPU time processing routine
AFBCDYNA	778	X	Dynamic file allocation processor
AFBCFISC	450	X	FILE specifier scan routine
AFBDDCMP	580	X	Dynamic common
AFBVABEX	1408	X	ABEND processor
AFBVAMTP	168		Alt. error option table processor
AFBVBLNT	208	X	Implied DO in I/O
AFBVCLOP	2B0	X	CLOSE statement
AFBVCOMH	15F8	X	Formatted I/O
AFBVCONI	330	X	Input floating-point conversion
AFBVCONO	828	X	Output floating-point conversion
AFBVCVTH	1400	X	Data conversion
AFBVEMGN	12D8	X	Error message generator
AFBVERRE	258	X	Error summary
AFBVEXIP	B8	X	Return code processor
AFBVFINP	AC8	X	FILEINF call file information block
AFBVFMTP	190		Language Conversion Program define file
AFBVIIOS	270	X	Internal file services
AFBVINQP	1A38	X	INQUIRE statement
AFBVIOCP	2E0	X	BACKSPACE, REWIND, ENDFILE
AFBVIOFP	7D0	X	Formatted I/O
AFBVIOLP	15C0	X	List-directed I/O
AFBVIOUP	D18	X	Unformatted I/O
AFBVLINP	298	X	Link to reentrant CSECT
AFBVMSKL	6520	X	Message skeletons
AFBVOPEP	10B0	X	OPEN statement
AFBVTEN	2C0	X	Powers of ten table
AFBVTRCH	C10	X	Traceback generator

Figure 29. Optional Modules for AFBVRENA

Composite Module AFBVRENB (XA Mode Only)

Module	Approx. Size	Default Set	Function
AFBCBREN	D0	X	Internal linkage module
AFBCFIST	1148	X	File status processor
AFBVSIOS	3928	X	Sequential I/O services
Total	**4B40**	**3**	

Figure 30. Required Modules for AFBVRENB

Module	Approx. Size	Default Set	Function
AFBVDIOS	1FA8		Direct access I/O services

Figure 31. Optional Modules for AFBVRENB

Composite Module AFBVRENC (Non-XA Mode Only)

Module	Approx. Size	Default Set	Function
AFBCFIST	1148	X	File status processor
AFBCREN	1D0	X	Internal linkage module
AFBCSTIO	310	X	Standard I/O initialization
AFBVGMFM	250	X	GETMAIN/FREEMAIN
AFBVRDCB	2A0	X	DCB attributes resolution
AFBVSIOS	3928	X	Sequential I/O services
AFBVTRMF	D0	X	File closing at termination
Total	**5510**	**7**	

Figure 32. Required Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBCCPTP	A8		CPU time processing routine
AFBCDYNA	778		Dynamic file allocation processor
AFBCFISC	450		FILE specifier scan routine
AFBCVIOS	21F8		Nonkeyed VSAM I/O services
AFBDDCMP	580		Dynamic common
AFBVABEX	1408		ABEND processor
AFBVAMTP	168		Alternate error option table processor
AFBVBLNT	208	X	Implied DO in I/O
AFBVCLOP	2B0	X	CLOSE statement
AFBVCOMH	15F8	X	Formatted I/O
AFBVCONI	330	X	Input floating-point conversion
AFBVCONO	828	X	Output floating-point conversion
AFBVCVTH	1400	X	Data conversion
AFBVDIOS	1FA8		Direct access I/O services
AFBVEMGN	12D8	X	Error message generator
AFBVERRE	258	X	Error summary
AFBVEXIP	B8		Return code processor
AFBVFINP	AC8		FILEINF call file information block
AFBVFMTP	190		Language Conversion Program define file
AFBVIIOS	270		Internal file services
AFBVINQP	1A38		INQUIRE statement
AFBVIOCP	2E0		BACKSPACE, REWIND, ENDFILE
AFBVIOFP	7D0	X	Formatted I/O
AFBVIOLP	15C0	X	List-directed I/O
AFBVIOUP	D18	X	Unformatted I/O
AFBVKIOS	2DF0		Keyed access I/O services
AFBVLINP	298		Link to reentrant CSECT
AFBVMSKL	6520	X	Message skeletons

Figure 33 (Part 1 of 2). Optional Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBVOPEP	10B0	X	OPEN statement
AFBVTEN	2C0	X	Powers of ten table
AFBVTRCH	C10	X	Traceback generator

Figure 33 (Part 2 of 2). Optional Modules for AFBVRENC

219B0

Appendix D. Servicing VS FORTRAN Version 2

IBM National Services Division (NSD) provides corrective and preventive service for product defects, as well as 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.

Problem Reporting

When you encounter a failure in the product, follow this procedure:

1. Use the *VS FORTRAN Version 2 Diagnosis Guide* to assist you in describing the failure as a keyword string.
2. Compare that keyword string with the index of keywords in the software support data base of documented failures. Contact the IBM Support Center for assistance in the keyword search if necessary.
3. If you cannot find a documented failure similar to yours, report your failure to the IBM Support Center as an authorized programming analysis report (APAR).

   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).

When reporting a problem, you may need the Field Engineering Service Numbers (FESN). These are shown in Figure 34.

Product	Component ID	FESN
Library	5668-80501	6480501
Compiler	5668-80601	6480601
IAD	5668-80602	6480602

Figure 34. Field Engineering Service Numbers

Corrective Service

Corrective service is distributed as a **PTF**, or Program Temporary Fix, 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.

The format of the corrective service tape is:

File 1: Memo to Users

File 2 and on: PTFs requested

and 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 should perform the following steps:

1. Before applying the new corrective service, you may want to consider backing up both your current product and installation work disks. (The "installation work disks" are the disks that contain all of the FORTRAN serviceable TEXT files.)
2. Read the memo that accompanies the corrective service tape entirely before starting the service installation.
3. According to the instructions in the memo, install corrective service.

Note: VS FORTRAN Version 2 is not supported by the VMFMERGE EXEC discussed in the corrective service memo.

4. Apply service to product as follows:
   - The FORTRAN service EXEC requires that its own installation work disk be accessed as the A-disk. Re-access the service disk which holds the rename service files to a file mode other than "A."
   - Access the current FORTRAN installation work disk as your A-disk.
   - Copy the corrective service TEXT files, created in Step 3, to this A-disk, replacing the previous versions of these files.
   - Access the VS FORTRAN product disk. This may be a complete copy of the original product disk or the original product disk itself, depending on chosen backup/test procedures. Be aware that the service application procedure does not regenerate all the files originally installed on the product disk.
5. 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

6. The EXEC will then prompt you for information necessary to install the service.
7. When it has completed, the EXEC will generate the product on the A-disk, and then copy service from the A-disk to the product disk.

Preventive Service

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 VM PUT tape contains cumulative information; thus, you need only the original product 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

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

VS FORTRAN Version 2
Installation and
Customization for VM

Reader's Comment Form

SC26-4339-1

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

Chapter/Section _____

_____ Page No. _____

Comments:

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

If you want a reply, please complete the following information.

Name _____ Phone No. (_____) _____

Company _____

Address _____

_____

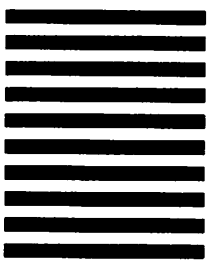
Thank you for your cooperation. No postage is necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail them directly to the address in the Edition Notice on the back of the title page.)

SC26-4339-1

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 MAIL PERMIT NO. 40 ARMONK, NY

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Programming Publishing
P.O. Box 49023
San Jose, CA 95161-9023

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

IBM®

IBM®

Program Number
5668-805
5668-806

File Number
S370-34

The VS FORTRAN Version 2 Library

Diagnosis Guide	LY27-9516
General Information	GC26-4219
Installation and Customization for MVS	SC26-4340
Installation and Customization for VM	SC26-4339
Interactive Debug Guide and Reference	SC26-4223
Language and Library Reference	SC26-4221
Licensed Program Specifications	GC26-4225
Programming Guide	SC26-4222
Reference Summary	SX26-3751

SC26-4339-1

Printed in U.S.A.