VS FORTRAN Version 2

Installation and Customization for CMS

Release 6

SC26-4339-05

VS FORTRAN Version 2

Installation and Customization for CMS

Release 6

SC26-4339-05

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page iv.

Sixth Edition (November 1993)

This edition replaces and makes obsolete the previous edition, SC26-4339-04.

This edition applies to VS FORTRAN Version 2 Release 6, Program Numbers 5668-805, 5688-087, 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 “About This Book.” Specific changes for this edition are indicated by a vertical bar to the left of the change. A vertical bar to the left of a figure caption indicates that the figure has changed. 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 Bibliography*, GC20-0001, for the editions that are applicable and current.

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 Reader's Comment Form is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department J58, 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.

Contents

Notices

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 product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

AIX
AIX/ESA
BookMaster
ES/3090
ES/9000
IBM
MVS/ESA
MVS/XA
RETAIN
SAA
Systems Application Architecture
VM/ESA
VM/XA

About This Book

How This Book Is Organized

This book is organized as follows:

Chapter 1, "Getting Acquainted with VS FORTRAN Version 2," describes VS FORTRAN Version 2 and discusses where to find more information about it.

Chapter 2, "Planning for Installation," specifies the required systems, hardware, virtual storage, and DASD space you will need when you install 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 if you are using both Interactive System Product Facility (ISPF) and Program Development Facility (PDF), ISPF without PDF, or not using ISPF at all.

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 shared segment, and change the option defaults.

Appendix A, "Program Materials," describes the VS FORTRAN Version 2 system tapes available from IBM Software Manufacturing and Delivery (ISMD).

Appendix B, "Customization Macros," describes the customization macros and how to use 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 2
2. Chapter 3, "Installing VS FORTRAN Version 2" on page 13

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 20. Separate procedures are provided for users of ISPF with PDF, 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 44; and Appendix C, "Composite Modules" on page 71.

- Apply service to VS FORTRAN Version 2. Refer to Appendix D, “Servicing VS FORTRAN Version 2” on page 80.

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 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.

Publications

Figure 1 lists the VS FORTRAN Version 2 publications and the tasks they support.

Figure 1. VS FORTRAN Version 2 Publication Library

Task	VS FORTRAN Version 2 Publication	Order Number
Evaluation and Planning	*General Information*	GC26-4219
	Licensed Program Specifications	GC26-4225
Installation and Customization	*Installation and Customization for CMS*	SC26-4339
	Installation and Customization for MVS	SC26-4340
Application Programming	*Language and Library Reference*	SC26-4221
	Programming Guide for CMS and MVS	SC26-4222
	Interactive Debug Guide and Reference	SC26-4223
	Reference Summary	SX26-3751
Library Reference	*Master Index and Glossary*	SC26-4603
Diagnosis	*Diagnosis Guide*	LY27-9516
Migration	*Migration from the Parallel FORTRAN PRPQ*	SC26-4686

Figure 2 lists publications supply information you might need while you are installing, customizing or servicing VS FORTRAN.

* IBM is a trademark of the International Business Machines Corporation.

Figure 2. Related Publications

Title	Order Number
Field Engineering Programming System General Information Manual	G229-2228
VM/ESA: ESA/Extended Configuration Principles of Operation	SC24-5594
VM/SP Planning Guide and Reference	SC19-6201
VM/SP System Programmer's Guide	SC19-6203
VM/XA Systems Product CP Command Reference	SC23-0358
IBM High Level Assembler/MVS & VM & VSE: Language Reference	SC26-4940
IBM High Level Assembler/MVS & VM & VSE: Programmer's Guide	SC26-4941

Industry Standards

The VS FORTRAN Version 2 compiler and library are designed according to the specifications of the following industry standards, as understood and interpreted by IBM as of December 1990.

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.

Documentation of IBM Extensions

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

Summary of Changes

Release 6, October 1993

Major Changes to the Product

Support for AIX*/370 is not included in VS FORTRAN Version 2 Release 6. Support for AIX/ESA* is found in the AIX VS FORTRAN/ESA product.

- Printable copies of certain of the VS FORTRAN Version 2 Release 6 publications are provided on the product tape.
- Support for additional language has been added, much of which address previous incompatibilities between XL FORTRAN and VS FORTRAN
  - Additional constant and operator support:
    - Quote-delimited literal character constants
    - Maximum negative integer literal value
    - Typeless constants (binary, octal and hexadecimal)
    - Named constants in complex constants
    - .XOR. logical operator
    - <> graphical relational operator
  - Storage classes:
    - STATIC and AUTOMATIC storage classification statements
    - IMPLICIT STATIC and IMPLICIT AUTOMATIC statement support
  - Additional data type support:
    - LOGICAL*2 and LOGICAL*8
    - INTEGER*1 and INTEGER*8
    - UNSIGNED*1
    - BYTE
    - DOUBLE COMPLEX and DOUBLE COMPLEX FUNCTION type specifications
  - XREF and MAP Processing has been improved to remove storage-ordering perturbations.
  - Additional intrinsic function support:
    - INTEGER*8, INTEGER*2, and INTEGER*1 support for existing functions
    - XOR, LSHIFT, RSHIFT, ISHFTC, IBITS, and LOC have been added.
  - HALT compiler option to reduce compilation times for unsuccessful compilations
  - MVBITS and ARGSTR service routines
  - Dynamic storage support, for allocating and deallocating storage at run-time
    - Integer POINTER data type

* AIX and AIX/ESA are trademarks of the International Business Machines Corporation.

- Including dynamically dimensioned arrays
- ALLOCATED, DEALLOCATE, and NULLIFY statements
- Intrinsic function ALLOCATED
- Compile-time option DDIM

– Dynamically loaded module support (via DYNAMIC option)

– I/O features enhancements:

- Support for dummy arguments in NAMELIST lists.
- NML keyword for NAMELIST READ/WRITE statements
- OPEN and INQUIRE statement support for the POSITION, PAD, and DELIM specifiers
- B and O format control codes
- Expansion of the Z format control code
- $ format control code

• Optimization improvements:

– Optimization for pipelined instruction scheduling
– Improved optimization for Inter-loop register assignments

• Message improvements:

– All messages written to SYSTERM file
– Information messages no longer marked as errors

• Run-time options improvements:

– Abbreviations are now allowed.

– Default I/O units are now user-specifiable with ERRUNIT, RDRUNIT, PRTUNIT, and PUNUNIT

• Vector support enhancements:

– The VECTOR option defaults can be set at installation.
– VECTOR suboptions MODEL and SPRECOPT.
– Vectorized SIGN, ANINT, DNINT, NINT and IDNINT intrinsic functions
– Vector performance improvements

• Parallel support enhancements:

– LOCAL statement allowing arrays for Parallel Do / Parallel Sections

– NAMELIST processing for parallel environment allows local variables

– Support for user-generated parallel trace (including service routines PTWRIT and PTPARM and suboption TRACE)

– Parallel execution controls (affinity control)

– Load module support for routines invoked via SCHEDULE and PARALLEL CALL

Release 5, September 1991

Major Changes to the Product

- Support has been added in the compiler and library for the Advanced Interactive Executive/370 (AIX/370) operating system. In addition, programs designed to be run on AIX/370 may use the following enhancements to VS FORTRAN Version 2:
  - The **fvs** command to invoke the VS FORTRAN Version 2 compiler.
  - Use of tab characters in source programs.
  - Communication between Fortran programs and C programs.
  - Coding of indirectly recursive Fortran routines.
- Support for parallel programs[1] has been added for programs running under CMS and MVS. These enhancements support:
  - Automatically generating parallel code for DO loops using a compile-time option.
  - Explicit coding of parallel loops, sections, and calls with parallel language extensions.
  - Using lock and event services to control synchronization in parallel programs.
  - Directing the compiler to generate parallel or serial code using enhanced directives.
  - Determining the number of virtual processors available and specification of the number of virtual processors to use during run time.
  - Using I/O within parallel programs.
  - Calling subroutines within parallel loops and sections.
  - Obtaining information for tuning your parallel program using a compiler report listing.
- Support for extended common blocks has been added for programs running under MVS/ESA* or VM/ESA*. Compile-time and run-time enhancements and options have been added to support the three types of common blocks that now exist.
- Virtual storage constraint relief has been added for compiling under MVS/XA*, MVS/ESA, VM/XA*, or VM/ESA. This support allows the compiler to reside above the 16MB line and to process in 31-bit addressing mode. Therefore, larger programs can now be compiled.
- The default name for a main program has been changed from MAIN to MAIN#, which allows MAIN to be used as a user name for a common block or other global entity.

---

[1] The VS FORTRAN Version 1 standard math routines (VSF2MATH) are not supported for parallel processing. Interactive debug can be used only with non-parallel programs, and with serial portions of parallel programs when the run-time option PARALLEL(1) is specified.

* MVS/ESA, MVS/XA, VM/ESA, and VM/XA are trademarks of the International Business Machines Corporation.

- Array declarator expressions for object-time dimensions can now be of type Integer*2.

Release 4, August 1989

Major Changes to the Product

- Enhancements to the vector and optimization features of VS FORTRAN Version 2
  - Automatic vectorization of user programs is enhanced by improvements to the dependence analysis done by the compiler. Specifically, the following constructs are eligible for vectorization:
    - Loops containing simple READ, WRITE, and PRINT statements
    - Loops containing equivalenced arrays
    - Loops containing branches out of the loop
    - Loop bound appearing in a subscript expression
    - Loop nests that process a triangular matrix
    - Simple IF loops
    - Integer sum reduction.
  - Additional advanced vector optimization.
  - In programs optimized at either OPT(2) or OPT(3), arithmetic constants will be propagated globally for scalar variables.
  - In programs optimized at either OPT(2) or OPT(3), informational messages are issued when local scalar variables may be referenced before they have been initialized.
  - Improved performance when the CHAR, ICHAR, and LEN character intrinsic functions are used.
  - Single and double precision complex divide routines can be vectorized.
- Enhancements to input/output support in VS FORTRAN Version 2
  - Improved data transfer rate for sequential DASD and tape input/output on MVS systems.
  - Ability to perform data striping (parallel I/O) on sequential data sets.
  - Ability to detect input conversion and record length errors.
- Enhancements to the intercompilation analyzer (ICA)
- Enhancements to the language capabilities of VS FORTRAN Version 2
  - Ability to use graphic relational operators as an alternative to FORTRAN 77 relational operators.
- Enhancements to the pseudo-assembler listing provided by VS FORTRAN Version 2
- Enhancements to the math routines:
  - Single, double, and extended precision MOD library routines are more precise.
  - Single and double precision scalar complex divide routines are more precise and faster, and are vectorizable.

- The old complex divide and MOD routines are in the alternate math library.

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 online 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 and I/O, and for interactive debug input and output.

* Systems Application Architecture and SAA are trademarks of the International Business Machines Corporation.

  - 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 and MVS/ESA.
  - 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

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 run-time option to control checking of file existence during the processing 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 processing 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.

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 run-time option
- Addition of support for VM/XA System Facility Release 2.0 (5664-169) operating system

Chapter 1. Getting Acquainted with VS FORTRAN Version 2

What Is VS FORTRAN Version 2

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 component.
- 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 for non-parallel programs and the serial portions of parallel programs.

It is available as three 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 verification programs that enable you to verify the installation procedures.
- VS FORTRAN Version 2 (5688-087), a licensed program containing the compiler and library. It also provides a sample verification program that enables 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 Manufacturing and Delivery (ISMD). It is available on either an unlabeled 9-track tape, written at 1600 or 6250 bpi, or a 3480 tape cartridge, or a 1/4" tape 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 are in the program announcement materials. Refer to *VS FORTRAN Version 2 General Information* or contact your IBM marketing representative for this information.
- Updates to the information and procedures in this book can be obtained from either your IBM Support Center or through the RETAIN*/370 Preventive Service Planning (PSP) Facility, as discussed on page 2.

* RETAIN is a trademark of the International Business Machines Corporation.

Chapter 2. Planning for Installation

This chapter helps you plan for the installation of VS FORTRAN Version 2. It specifies the required systems, hardware, virtual storage, and DASD space you will need.

Before installing VS FORTRAN Version 2 Release 5, 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
VSFORTRAN260	VSFORT/860

Meeting the Basic Requirements

The basic VS FORTRAN Version 2 system and machine requirements, and virtual and DASD storage requirements are discussed in the following sections.

Operating System Requirements

VS FORTRAN Version 2 supports vector, parallel, and scalar features as shown in Figure 3:

Figure 3 (Page 1 of 2). Compilation and Execution Support for Vector and Parallel Features

Product	Vector	Parallel	Scalar/ Serial
VM/SP (5664-167) Release 5 or later	+	+	*
VM/SP HPO (5664-173) Release 5 or later	*	+	*
VM/XA SP (5664-308):			
• Release 1.0 or later with bimodal CMS	*	+	*
• Release 2.0 or later with bimodal CMS, with PRPQ P81051, CMS Support for Parallel VS FORTRAN (5799-DGW)	*	*	*
VM/ESA (5684-112):			
• Release 1.0 or later	*	+	*

Figure 3 (Page 2 of 2). Compilation and Execution Support for Vector and Parallel Features

Product	Vector	Parallel	Scalar/ Serial
• with the CMS PRPQ to Support Parallel Processing in VS FORTRAN (5799-DGW)	*	*	*

Legend:

+ indicates that compilation is supported for the specified feature.
***** indicates that compilation and execution are supported for the specified feature.

VM/ESA requires VSE/VSAM (5746-AM2) Release 4 (or later) to process VSAM files.

VM/SP requires VSE/VSAM (5746-AM2) Release 1, 2, or 3 to process VSAM files. VM/XA requires VSE/VSAM (5746-AM2) Release 3 to process VSAM files.

Assembler H Version 2 Release 1.0 or IBM High Level Assembler/MVS & VM & VSE is required for customization of VS FORTRAN Version 2 in VM/XA, MVS/XA and MVS/ESA environments.

Interactive debugging requires CMS (on VM). Interactive debugging in full-screen mode requires:

- Under VM/ESA, ISPF Version 3 Release 2 for VM (5684-043) with or without ISPF/PDF Version 3 Release 2 for VM (5684-123). For enhanced full-screen functions, 5684-123 is required.
- Under VM, ISPF Version 2 for VM (5664-282) with or without ISPF/PDF Versions 2 for VM (5664-285). For enhanced full-screen functions, 5664-282 is required.
- Under VM/XA, ISPF Version 2 Release 2 for VM (5684-015) with or without ISPF/PDF Version 2 Release 2, for VM (5684-014). For enhanced full-screen functions, 5684-015 is required.
- Under VM/SP Release 6, ISPF Version 2 Release 2.1 for VM (5664-285).

In addition to the requirements given above for interactive debugging in full-screen mode, the appropriate ISPF/PDF product must be selected 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 by means of the interactive debugging foreground invocation panel.

Machine Requirements

The compile-time machine requires both of the following:

- Any processing unit supported by VM
- I/O devices, normally disks, used by the compiler

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
- An appropriate vector processor, if required by your environment

Virtual Storage Requirements

The VS FORTRAN Version 2 compiler requires 3 megabytes 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.

The virtual storage utilization above the 16-megabyte line for a parallel program will be greater than its serial equivalent. In an XA environment, virtual storage used below the 16-megabyte line will be equivalent.

DASD Storage Requirements

You need space available on two target disks: the **work** disk and the **product** disk. Figure 4 shows the minimum DASD storage required. The space includes vector routines, the “combined” link library, inclusion of all optional modules in the composite modules, and publication files. Data shown in Figure 4 denote cylinders, except for fixed block devices (3310, 3370, 9332, and 9335) which are shown in blocks.

Figure 4. DASD Storage Requirements (cylinders/fixed blocks)

SPACE	T3330-11	T3340	T3350	T3375	T3380, T3390	T3310, T3370 T9332, T9335
Work Disk	142	335	68	82	54	64000
Product Disk	94	223	45	56	36	43000

Specify a block size of at least 1024 bytes. For maximum efficiency, specify a block size of 4K.

Extended Common Requirements

VS FORTRAN Version 2 programs that use extended common blocks allocate storage for these blocks from VM/ESA data spaces. To obtain this storage, VS FORTRAN must first allocate the necessary data spaces. This imposes the following requirements:

- FORTRAN programs using extended common blocks must run on VM/ESA Release 1.1 or later systems and must be under the control of an XC virtual machine.
- The user's VM directory must contain XCONFIG directory control statements that specify values for ACCESSLIST and ADDRSPACE.

Two different values must be specified with the XCONFIG statement. The values are ACCESSLIST, which defines the maximum number of concurrent data spaces that a FORTRAN program will use, and ADDRSPACE, which defines the maximum number and total size of the data spaces that will be used by a FORTRAN program.

The following are provided to assist you in determining these values:

1. VS FORTRAN allocates data spaces only as needed to contain extended common blocks. When the ECPACK run-time option is specified, each data space is allocated with a size of 2 gigabytes. When using the ECPACK option (which is the default option) you must be able to allocate at least one 2-gigabyte data space for the first extended common block. Subsequent extended common blocks are placed in this data space if room is available. If additional space is needed, a new 2-gigabyte data space is allocated, as long as the number and limit on the data spaces does not exceed the value specified in the XCONFIG ADDRSPACE directory control statement. When these limits are exceeded, the execution of the FORTRAN program is terminated.
2. When the NOECPACK run-time option is specified, each extended common block is placed in its own data space, and the values in the XCONFIG ADDRSPACE directory control statement must be sufficient to allow those data spaces to be defined. Because the size of each data space is limited to the size of the single extended common block within the data space, the size of each data space can be reduced.
3. Each data space allocated requires an entry in a CP table known as the *host access list*. The maximum number of entries in this table is specified by the XCONFIG ACCESSLIST directory control statement. Therefore, the value specified in this statement should be at least the number specified as the maximum number of data spaces in the XCONFIG ADDRSPACE directory control statement.

   Different users may need different values for the XCONFIG ACCESSLIST and XCONFIG ADDRSPACE directory control statements.

Identifying the VS FORTRAN Version 2 Files

Installation 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
VSF2PARA TXTLIB	Run-time parallel routines
Load Library	
VSF2LOAD LOADLIB	Routines that can 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
ILX0FORT MODULE	Compiler (CMS/XA or CMS/ESA only)
ILX0TRCE TXTLIB	Trace modules. Provides compiler diagnostic information.
Interactive Debug Files (only if interactive debug is installed)	
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
AFFCMDS Table	Interactive debug, under ISPF, help table

Text Libraries	
Publication PostScript Files	
AFBLLMST LISTPS	VS FORTRAN Version 2 Language and Library Reference
AFBP6MST LISTPS	VS FORTRAN Version 2 Programming Guide for CMS and MVS
AFBG5MST LISTPS	VS FORTRAN Version 2 General Information
AFBC5MST LISTPS	VS FORTRAN Version 2 Installation and Customization for CMS
AFBM5MST LISTPS	VS FORTRAN Version 2 Installation and Customization for MVS
AFBS5MST LISTPS	VS FORTRAN Version 2 Licensed Program Specifications
AFBR6MST LISTPS	VS FORTRAN Version 2 Reference Summary
AFBX5MST LISTPS	VS FORTRAN Version 2 Master Index and Glossary
Publication BookMaster* Files	
AFBLLMST LIST3820	VS FORTRAN Version 2 Language and Library Reference
AFBP6MST LIST3820	VS FORTRAN Version 2 Programming Guide for CMS and MVS
AFBG5MST LIST3820	VS FORTRAN Version 2 General Information
AFBC5MST LIST3820	VS FORTRAN Version 2 Installation and Customization for CMS
AFBM5MST LIST3820	VS FORTRAN Version 2 Installation and Customization for MVS
AFBS5MST LIST3820	VS FORTRAN Version 2 Licensed Program Specifications
AFBR6MST LIST3820	VS FORTRAN Version 2 Reference Summary
AFBX5MST LIST3820	VS FORTRAN Version 2 Master Index and Glossary
Product Support Files	
PARTRACE INFO	Description of Trace File records
AFBTRAC FORTRAN	Sample program to format contents of Trace File
Product Information	
VSF2 README	Update VS FORTRAN Version 2 information

* BookMaster is a trademark of the International Business Machines Corporation.

Preparing for Shared Segment Installation

This step is optional. If you want to install the compiler or the library's composite modules AFBVRENA, AFBVRENB, AFBVRENC, or AFBVRENP in a shared segment, perform the steps described below. If not, proceed with Chapter 3, “Installing VS FORTRAN Version 2” on page 13.

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

1. Determine the Characteristics of Each Shared Segment

- Decide the name of each shared segment. This can be any name you specify within the CMS naming conventions. If you have more than one release of VS FORTRAN Version 2 on your system, you must choose different shared segment names for each release.
- Determine the starting address of each shared segment. You can use the following guidelines:
  - The address should be at least as large as the virtual machine of any user.
  - Setting the shared segment unnecessarily high wastes storage for unreferenced CP segment table entries. However, to accommodate users with different virtual machine sizes, you can create several shared segments, each with a different starting address.
  - The address should not allow a shared segment 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 shared segment should not overlap any shared segments that contain parts of ISPF.
- Determine the number of segments in each shared segment. For the compiler, this number is fixed at **twenty-nine** 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 shown in Figure 5:

Figure 5. Composite Module Sizes

Composite Module	Number of Segments	
	Default	Maximum
AFBVRENA	3	4
AFBVRENB	1	1
AFBVRENC	3	5
AFBVRENP	2	2

  - 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 71 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.

   Note: The composite module AFBVRENP cannot be modified.

2. Define Each Shared Segment to VM

For VM/SP or VM/ESA with the 370 feature:

1. Allocate permanent space on a CP-owned DASD volume to contain 1 each shared segment. For more information, refer to *VM/SP Planning Guide and Reference*.
2. Add a NAMESYS macro instruction to your site's DMKSNT ASSEMBLE module (see *VM/SP Planning Guide and Reference*, and *VM/SP System Programmer's Guide*). There must be one NAMESYS macro for each shared segment being created.

   Figure 6 on page 10 defines a compiler shared segment named DSSVFORT. The data is based on a shared segment 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 7 on page 11 defines a composite module shared segment named FTNLIB10. The sample data illustrate one possible set of numbers and are not intended as the only location or size for a shared segment.
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*. Load the nucleus on the IPL volume, then re-IPL.
4. When making preparations to install VS FORTRAN in a shared segment on VM/SP, define and load the compiler/library on a 64K boundary.

For VM/XA or VM/ESA with the ESA feature:

1. Issue a DEFSEG command. See Figure 8 and Figure 9 on page 12. Refer to *VM/XA Systems Product CP Command Reference* for more information.
2. When making preparations to install VS FORTRAN in a saved segment on VM/XA, define and load the compiler/library on a megabyte boundary unless segment packing is used.
3. VS FORTRAN can run in 31-bit mode under VM/XA or VM/ESA in an XA or XC virtual machine. In a 370 virtual machine, VS FORTRAN runs only in 24-bit mode.

   You may want to define multiple segments, one below the 16-megabyte addressing limit and one above to take maximum advantage of VM's XA capabilties.

You may now proceed with Chapter 3, "Installing VS FORTRAN Version 2" on page 13.

```
DSSVFORT NAMESYS SYSNAME=DSSVFORT,              Note 1
                 SYSSIZE=1856K,
                 SYSHRSG=(80,81,82,83,84,85,86,
                     87,88,89,90,91,92,93,94,95,
                     96,97,98,99,100,101,102,
                     103,104,105,106,107,108)   Note 2
                 SYSPGNM=(1280-1744),           Note 3
                 VSYSADR=IGNORE,
                 SYSVOL=VMSRES,                 Note 4
                 SYSSTRT=(049,1)                Note 5
```

Notes:

1. The SYSNAME parameter specifies the name of the shared segment (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 shared segment 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 shared segment. Compute the first page number by dividing the starting address of the shared segment by 4K. In this example, dividing X'500000' or 5120K by 4K gives a starting page number of 1280. The compiler requires **twenty-nine** 64K segments, a total of 1856K. Dividing 1856K by 4K (the page size) yields a total of 464 pages. Starting at page 1280, 464 pages results in an ending page of 1744.
4. The SYSVOL parameter gives the volume serial number of the CP-owned volume that holds the shared segment.
5. The SYSSTRT parameter gives the starting cylinder and page address (on the volume specified by the SYSVOL parameter) that holds the shared segment.

Figure 6. VM/SP Sample NAMESYS Macro Instruction for Compiler Shared Segment

Shared Segment

```
FTNLIB10 NAMESYS SYSNAME=FTNLIB10,            Note 1
                 SYSSIZE=192K,
                 SYSHRSG=(48,49,50),          Note 2
                 SYSPGNM=(768-815),           Note 3
                 VSYSADR=IGNORE,
                 SYSVOL=VMSRES,               Note 4
                 SYSSTRT=(072,1)              Note 5
```

Notes:

1. The SYSNAME parameter specifies the name of the shared segment (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 shared segment 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 shared segment. Compute the first page number by dividing the starting address of the shared segment by 4K. In this example, dividing X'300000' or 3072K by 4K gives a starting page number of 768. A range of 48 pages is specified here to correspond to the 3 segments.
4. The SYSVOL parameter gives the volume serial number of the CP-owned volume that holds the shared segment.
5. The SYSSTRT parameter gives the starting cylinder and page address (on the volume specified by the SYSVOL parameter) that holds the shared segment.

Figure 7. VM/SP Sample NAMESYS Macro Instruction for Composite Module

DEFSEG Command for Compiler Shared Segment

```
DEFSEG DSSVFORT  900-AD0 SR
```

Notes:

1. DSSVFORT is the name of the shared segment for the compiler. Change the name to whatever you desire.
2. The hexadecimal numbers 900-AC0 are the first and last page numbers of the shared segment. In this example, we want the compiler shared segment to be loaded starting at X'0900000' (or the 9 megabyte line). To compute the hexadecimal page numbers,
   a. for the starting page, see the following examples,

```
Start  Storage-Address  HexPage

5M     00500 000        500
6M     00600 000        600
7M     00700 000        700
8M     00800 000        800
9M     00900 000        900
```

   b. for the ending page, add a hex factor of 1D0 (or 1856K).

   Warning: The ending page for the compiler shared segment can be greater than X'FFF'.
3. SR indicates that the segments are to be shared and page protected, rather than exclusive (EW).

   Note: Starting page for VM/ESA can be above the 16-megabyte line.

Figure 8. VM/XA and VM/ESA Sample

for Composite Module Shared Segment

```
DEFSEG FTNLIB20 1100-1130 SR
```

Notes:

1. FTNLIB20 is the name of a shared segment for a composite module. Change the name to whatever you desire.
2. The hexadecimal numbers 1100-1130 are the first and last page numbers of the shared segment. In this example, we want AFBVRENA at its default size (3 segments) to be loaded starting at X'1100000' (or the 17M line) and ending at X'1130000'. 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
 3       192K      30000      30
 4       256K      40000      40
 5       320K      50000      50
```

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

Figure 9. VM/XA and VM/ESA Sample DEFSEG Command

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 2.

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 shared segment, your virtual machine size must be defined at a minimum of 5-megabytes. In this example you would type:

     ```
     CP DEFINE STORAGE 5M
     ```

     Then re-IPL CMS.
   - If you are installing the compiler or library in a shared segment, 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 compiler in a shared segment to start at 5-megabytes, and because the compiler requires approximately 3-megabytes, specify the virtual machine size no less than 8-megabytes. In this example, you would type:

     ```
     CP DEFINE STORAGE 8M
     ```

     Then re-IPL CMS.

     For more information about determining storage requirements, refer to "Preparing for Shared Segment Installation" on page 8.
   - If you wish to use VS FORTRAN Version 2 in XA mode (under VM/XA SP), regardless of whether you are installing in a shared segment, you must define your storage to be greater than 16 megabytes to gain the advantages of XA. In this example, you would type:

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

     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 4 on page 4 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 4 on page 4 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 `I5688087` if you are installing the VS FORTRAN Version 2 compiler and library product 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 need the information in the figures shown under “Installation Items” and the accompanying notes.

Start by entering `I5668806` for the complete product (or `I5688087` for the compiler and library 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 10, Figure 11, and Figure 12 on page 15. If you are installing the library-only product, you will not be asked about items regarding the compiler or interactive debug. If you are installing the compiler and library product, you will not be asked about items regarding interactive debug.

You can halt the installation EXEC by responding to any prompt with `QUIT`. To restart the installation, enter `I5668806` (or `I5688087` or `I5668805`). You will be given the choice of either restarting at the beginning, or restarting after the last major step successfully completed.

Installation Items

Figure 10 (Page 1 of 2). Compiler Installation Items

Item	Default
Product disk file mode [1]	E
Product disk address	19E
Macro library name	VSF2MAC

Figure 10 (Page 2 of 2). Compiler Installation Items

Item	Default
Change compile-time options [2]	NO
Install compiler in a shared segment [3]	NO

Figure 11. Library 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-Shared Segment/Change[3]	NO
Load library name	VSF2LOAD

Figure 12. Interactive Debug Installation Items

Item	Default
Install interactive debug [8,10]	YES
Install ISPF files [8]	YES
Install line mode IAD help files [8]	YES
Compile and run sample verification programs [9]	YES

Figure 13. Publications Installation Items

Item	Default
Install publication files	YES
Install PostScript files	YES
Install BookMaster files	NO

Figure 14. Support and ReadMe Infomation Installation Items

Item	Default
Install support files	YES
Install ReadMe file	YES

Notes to Installation Items:

1. Do not specify A, S, or Y. The A-disk is the **work** disk; S and Y are system disks.
2. Refer to "Changing Option Defaults" on page 34 and Appendix B, "Customization Macros" on page 44 for more information.
3. If you install the compiler or a composite module in a shared segment, refer to "Preparing for Shared Segment Installation" on page 8 **before** starting the installation. Refer to Appendix C, "Composite Modules" on page 71 for more information.
4. This data will be requested only if you are installing the Library-only product.
5. The 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 24 on page 38.
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).
7. Refer to "Specifying Libraries in Link Mode" on page 17 for more information.
8. The default for these installation items is NO when you are installing the compiler and library product (`I5688087`).
9. For the compiler and library product, there is only one sample verification program (AFBIVP).
10. Interactive debug can be used only with non-parallel programs, and the serial portions of parallel programs when the run-time option PARALLEL(1) is specified. Interactive debug cannot be used with the features introduced with VS FORTRAN Version 2 Release 6.

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). Only load mode is supported for a parallel program.

After installation of the library, you can 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, use an EXEC to issue the appropriate GLOBAL or FILEDEF commands.

Specifying Libraries in Load Mode

Specify the VSF2FORT TXTLIB in the CMS GLOBAL command for use by the LOAD command:

```
GLOBAL TXTLIB VSF2FORT CMSLIB
```

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

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

where `fm` is the file mode of the product disk.

To 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, specify VSF2LINK in a GLOBAL command:

- 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
  ```

  In this case, you *cannot* use the LKED command to create a program that will be ready to run.

  Note: The processor 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.

- 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
  ```

  Note: 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.

Verifying a Successful Installation

If you are installing the library-only product, continue with Chapter 5, “Customizing VS FORTRAN Version 2” on page 31. The verification of installation is not a valid procedure for the library-only product.

If you are not installing the library-only product, 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 and library in a shared segment:
  1. Answer YES to this prompt.
  2. The installation EXEC will then compile and run the compiler and library verification program AFBIVP automatically, and you will receive the informational messages.
  3. If you have installed interactive debug, the EXEC will then compile and run the interactive debug verification 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 20 on page 30.

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 14 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. Unload by using 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 20 or Chapter 5, “Customizing VS FORTRAN Version 2” on page 31.

- If you have installed the compiler and library in a shared segment:

  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 the compiler and library in a shared segment, verification must be done manually.
  3. If you created a temporary disk for the work disk (refer to item 1 on page 14 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 2M, enter the commands shown at the top of Figure 15.

```
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 specified. During installation of the VS FORTRAN Version 2 library, the library text files are placed on a different minidisk from the text libraries so as to eliminate the problems that would occur because of the omission of the NOAUTO option on the LOAD command. However, the installer has access to both disks and must use the NOAUTO option of the LOAD command or an error results.
2. yyy is the virtual address of the product disk.

Figure 15. Product Verification for Compiler Shared Segment

5. Execute the verification EXEC, specifying, in the following order, the library names used when installing and either “YES” or “NO” to signify whether or not the interactive debug was installed.

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

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

7. If you have installed interactive debug, the EXEC will then compile and run the interactive debug verification 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 20 on page 30.

   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 20 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 13, and have installed interactive debug as prompted by the installation EXEC.

If you have previously installed interactive debug for a prior release of VS FORTRAN Version 2, you may want to go directly to the appropriate section of "Verifying Availability of the Interactive Debug" and proceed to "A Sample Interactive Debug Session" both of which are discussed later in this chapter.

Note: Interactive debug can be used only with non-parallel programs, and the serial portions of parallel programs when the run-time option PARALLEL(1) is specified. Interactive debug cannot be used with the features introduced with VS FORTRAN Version 2 Release 6.

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 27. If you are not using ISPF Version 2, go to "Instructions for Non-ISPF Users" on page 28.

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. The last section provides 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 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 16 on page 22, 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 V2 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
  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
```

Figure 16. 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 17 on page 23.

```
%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
                 1,ISR41000
                 2,ISR42000
                 3,ISR43000
                 4,ISR44000
                 5,ISR45000
                 6,ISR46000
                 7,ISR47000
                 8,ISR48000
                 9,ISR49000
                10,ISR4A000
                11,AFF4B000
                12,ISR4C000
                 )
   &ZUP = ISR00003
)END
```

Figure 17. 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 18 on page 25. This example assumes that you have ISPF/PDF Version 2 as well as ISPF Version 2.

CAUTION: 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 18. Sample EXEC to Invoke ISPF/PDF

6. Verifying Availability of Interactive Debug

Use the interactive debug verification 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 CMS commands.

1. Compile the verification 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.

     ```
     VS FORTRAN VERSION 2 Release 6 ENTERED.  hh:mm:ss

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

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

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

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

     VS FORTRAN VERSION 2 Release 6 EXITED.  hh:mm:ss
     ```

   - If you do not receive these messages or you receive error or warning messages, review your installation procedures to this point. Ensure 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 18 on page 25.

   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 16 on page 22, number 11 was used.) The FOREGROUND VS FORTRAN VERSION 2.5.0 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*.

      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 30. 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 (ISPF/PDF). If you are not using ISPF Version 2, go to "Instructions for Non-ISPF Users" on page 28.

1. Modifying the Interactive Debug 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:

- You may need to change the FILEDEF statements for file names in the IAD EXEC.
- You may need to change the ISPDCS statements for names in the IAD EXEC.

2. Verifying Availability of Interactive Debug

Use the interactive debug verification 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 CMS commands.

1. Compile the verification program 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.

     ```
     VS FORTRAN VERSION 2 Release 6 ENTERED.  hh:mm:ss

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

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

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

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

     VS FORTRAN VERSION 2 Release 6 EXITED.  hh:mm:ss
     ```

   - 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 interactive debug, link to the product minidisk(s) containing ISPF and VS FORTRAN Version 2.

   b. Invoke interactive debug 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. The AFFIVP program does not require any additional file definitions. Enter `END` when you are done. For further

information, refer to *VS FORTRAN Version 2 Interactive Debug Guide and Reference*.

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 30. 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 interactive debug by typing:

```
XEDIT FORTIAD EXEC A
```

Code an EXEC as shown in Figure 19. The EXEC must include FILEDEFs for all files used by 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 19 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 19. Sample FORTIAD EXEC

2. Verifying Availability of Interactive Debug

Use the interactive debug verification 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 CMS commands.

1. Compile the verification program 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.

     ```
     VS FORTRAN VERSION 2 Release 6 ENTERED.  hh:mm:ss

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

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

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

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

     VS FORTRAN VERSION 2 Release 6 EXITED.  hh:mm:ss
     ```

   - 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 28, 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 interactive debug prompt `FORTIAD`, indicating that FORTRAN has reached the first debugging hook. For example,

      ```
      VS FORTRAN VERSION 2 RELEASE 5 INTERACTIVE DEBUG
      5668-806 (C) COPYRIGHT IBM CORP. 1986, 1990
      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, 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 the interactive debug verification program AFFIVP under interactive debug is illustrated in Figure 20, 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 20 will not appear.

```
=VS FORTRAN VERSION 2 RELEASE 5 INTERACTIVE DEBUG
=5668-806 (C) COPYRIGHT IBM CORP. 1986, 1990
=LICENSED MATERIALS-PROPERTY OF IBM
=WHERE: CIRCLE.7
=* listsubs
=PROGRAM UNIT                    COMPILER   OPT   HOOKED TIMING
=CIRCLE                         VSF 2.5.0    0      YES    OFF
=DIAM                           VSF 2.5.0    0      YES    OFF
=CIRCUM                         VSF 2.5.0    2      YES    OFF
=AREA                           VSF 2.5.0    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
=FT06F001  ENTER THE VALUE OF THE CIRCLE RADIUS (xxx.xx):
=FT05F001 INPUT: PRECEDE INPUT WITH % OR ENTER IAD COMMAND
=* %352.67
=AT: DIAM.ENTRY
=NEXT: DIAM.3
=* set diam.value = 0.0
=* when test value
=* go
=WHEN: "TEST" SATISFIED;
=CURRENTLY AT DIAM.4
=* offwn test
=* at circle.42 (list '= READY FOR TERMINATION ='%go)
=* listbrks
=CURRENT BREAKPOINTS:
=  CIRCLE.42
=  DIAM.ENTRY
=CURRENT WHEN CONDITIONS:
=  TEST  OFF   DIAM.VALUE
=CURRENT HALT STATUS: OFF
=* go
=FT06F001  THE DIAMETER OF THE CIRCLE IS  705.34
=FT06F001  THE CIRCUMFERENCE OF THE CIRCLE IS 2215.89
=FT06F001  THE AREA OF THE CIRCLE IS  390738.94
=AT: CIRCLE.42
== READY FOR TERMINATION =
=PROGRAM HAS TERMINATED; RC ( 0)
=* quit
```

Figure 20. 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 13.

Note: You must have IBM High Level Assembler/MVS & VM & VSE or Assembler H Version 2 Release 1.0 to customize VS FORTRAN 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 can 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.

Note: The VS FORTRAN Version 1 standard math routines (VSF2MATH) are not supported for a parallel program.

After installation of the VS FORTRAN Version 2 library, you can 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 shared segment, complete all the preparatory steps in “Preparing for Shared Segment Installation” on page 8.
2. Log on to VM. If you are not installing a composite module in a shared segment, skip step 3.
3. If you are installing in a shared segment, you must have class E privileges. Define a virtual storage size that exceeds the starting address of the shared segment by at least 1 megabyte. For example, if the shared segment 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.
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 product disk. This load library contains your customized composite modules. It also refers to any shared segments that were built.

   If you built one or more shared segments, 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 shared segment, 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
   ```

Putting the Compiler in a Shared Segment

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

1. Complete all the preparatory steps described in “Preparing for Shared Segment Installation” on page 8.
2. Log on to a userid that has E privileges.
3. Define a virtual storage size that exceeds the starting address of the shared segment by at least 3 megabytes. For example, if the shared segment 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 shared segment parameter, as follows:

   ```
   I5668806 DCSS (or I5688087 DCSS)
   ```

   Reply `YES` to the prompt asking if you are installing the compiler as a shared segment, and be prepared to give the shared segment names.
7. Verify that the compiler was successfully installed in the shared segment by doing the following for each shared segment being installed:
   a. Redefine your virtual machine storage size to be the same as or less than the starting address of the shared segment, and re-IPL CMS.
   b. Re-access the work disk as the A-disk (as in step 4).
   c. Compile the compiler and library verification program AFBIVP by issuing the command:

      ```
      FORTVS2 AFBIVP
      ```
8. Once you are satisfied that the compiler has been successfully installed, 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

Modifications to the first three options could have been done during installation. For guidelines on invoking macros, refer to Appendix B, “Customization Macros” on page 44.

Changing Compile-Time Option Defaults

For additional information on the VSF2COM macro, see page 44.

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

The product tape provides a file, ILX0OPTS ASSEMBLE, 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 positioning the tape to the correct file (file 4 in Figure 26 on page 41) and issuing:

   ```
   VMFPLC2 LOAD * * fm
   ```

   where `fm` is the file mode of the **work** disk that you established during installation.

2. Edit the ILX0OPTS ASSEMBLE file:

   ```
   XEDIT ILX0OPTS ASSEMBLE fm
   ```

   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
   HLASM ILX0OPTS (OBJECT
   ```

   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 HLASM or HASM instead of ASSEMBLE.

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 shared segment, you must re-install it after you have changed any compile-time option defaults. Refer to "Putting the Compiler in a Shared Segment" 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 44.

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 for CMS and MVS*.

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` is 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 36.

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

   ```
   GLOBAL MACLIB VSF2MAC
   HLASM AFBCUAT (OBJECT
   ```

   Notes:

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

   b. You may specify HASM instead of HLASM.

   c. 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 commands:

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

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

IBM-Supplied Default Values: The macro instructions provided in the AFBCUAT ASSEMBLE file set up the IBM-supplied default values for the standard I/O units, as well as file characteristics such as the DCB information. 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
```

Figure 21. Sample 1: IBM-Supplied Unit Attribute Table Macro

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.

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 the example shown in Figure 22 on page 37, 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
```

Figure 22. Example 1: Modified Unit Attribute Table Macro

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

Example 2

To change the default standard I/O unit values for READER, ERRMSG, PRINTER, and PUNCH to 1, 2, 3, 4, respectively, modify the IBM-supplied macros as shown in Figure 23:

```
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 23. Example 2: Modified Unit Attribute Table Macro

Example 3

The example shown in Figure 24 on page 38 highlights the changes you should make to run a FORTRAN program with MVS file characteristics. To change the default file characteristics from CMS to MVS, the only macro instruction you need

to code is a VSF2DCB macro instruction, as shown below. See Figure 21 on page 36 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
```

Figure 24. Example 3: Modified Unit Attribute Table Macro

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.

Changing Run-Time Option Defaults

For additional information on the VSF2PARM macro, see page 59.

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` is 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
HLASM AFBVGPRM (OBJECT
```

Notes:

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

b. You may specify HASM instead of HLASM.

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

Customizing the Error Option Table

For additional information on the VSF2UOPT macro, see page 65.

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 VS 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 CMS and MVS* for more information.

To customize AFBUOPT, do the following:

1. Edit the AFBUOPT ASSEMBLE file:

   ```
   XEDIT AFBUOPT ASSEMBLE fm
   ```

   where `fm` is 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
   HLASM AFBUOPT (OBJECT
   ```

   Notes:

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

   b. You may specify HASM instead of HLASM.

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

Appendix A. Program Materials

This appendix identifies the basic machine-readable materials available from IBM Software Manufacturing and Delivery (ISMD).

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

Figure 25 identifies characteristics of the various distribution tapes for basic VS FORTRAN Version 2 for the complete product (compiler, library, and interactive debug). Figure 26 shows the tape format.

Figure 25. Distribution Tape (Basic) Identifiers: Complete Product

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
1/4" Tape Cartridge	—	5813	VM BASIC MATERIAL

Figure 26. Distribution Tape Format: 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) IAD, with ISPF, Message Library (VSF2MLIB MACLIB) IAD, with ISPF/PDF, Invocation (AFFFX11 EXEC) IAD, with ISPF, Invocation (IAD EXEC)
File 8	IAD Help Files for Line Mode
File 9	Verification EXEC V5668806 Sample Verification Program for Fortran (AFBIVP FORTRAN)
File 10	Sample Verification Program for IAD (AFFIVP FORTRAN)
File 11	Product Publications in PostScript format Product Publications in BookMaster format
File 12	Product Support Information Product ReadMe file

Compiler and Library Product (5688-087)

Figure 27 identifies characteristics of the various distribution tapes for basic VS FORTRAN Version 2 for the Compiler and Library product. Figure 28 shows the tape format.

Figure 27. Distribution Tape (Basic) Identifiers: Compiler and Library Product

Medium	Track/ Density	Feature Number	External Label
Magnetic Tape	9/1600	5015	VM BASIC MATERIAL
Magnetic Tape	9/6250	5016	VM BASIC MATERIAL
3480 Cartridge	18/38000	5872	VM BASIC MATERIAL
1/4" Tape Cartridge	—	5814	VM BASIC MATERIAL

Figure 28. Distribution Tape Format: Compiler and Library

File 1	Product Identifier File Installation EXEC I5688087
File 2	Memo to Users
File 3	Default Files Library Installation EXEC I5668805
File 4	Compiler Text Files and Macros
File 5	Library Text Files and Macros
File 6	Verification EXEC V5668806 Sample Verification Program for Fortran (AFBIVP FORTRAN)
File 7	Product Publications in PostScript format
	Product Publications in BookMaster format
File 8	Product Support Information
	Product ReadMe file

Library-Only Product (5668-805)

Figure 29 identifies characteristics of the various distribution tapes for basic VS FORTRAN Version 2 for the library-only product. Figure 30 on page 43 shows the tape format.

Figure 29. Distribution Tape (Basic) Identifiers: Library Only

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
1/4" Tape Cartridge	—	5813	VM BASIC MATERIAL

Figure 30. Distribution Tape Format: 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

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 can 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 can be continued on any number of lines as long as column 72 contains a nonblank character and the data on the following line begins in column 16.
4. A comma must follow the last option on a line when a continuation line follows.
5. You need to code only the options whose default values you want 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 compile-time options in the *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

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

Each row in Figure 31 on page 45 shows incompatible compile-time options. If any of these combinations of values are specified, the VSF2COM macro instruction will terminate without producing a new ILX0OPTS module.

Figure 31. Incompatible Compile-Time Options

Option	Conflicting Option
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, PUNCH=NODECK	SYM=SYM
OBJPROG=NOOBJECT	TEST=TEST
OPTIMIZ=1 | 2 | 3	TEST=TEST
SAA=SAA	SORCIN=FREE
SORLIST=NOSOURCE	SRCFLG=SRCFLG
TEST=TEST	SYMDUMP=NOSDUMP

Syntax of VSF2COM Macro

```
VSF2COM
      SYSTEM=CMS
      [,ADJ=IL(DIM) | IL(NODIM)]
      [,CHARLEN=number | 500]
      [,DATE=MDY | YMD]
      [,DBCS=DBCS | NODBCS]
      [,DDIM=DDIM | NODDIM]
      [,EMODE=EMODE | NOEMODE]
      [,FIPS=S | F | NOFIPS]
      [,FLAG=I | W | E | S]
      [,HALT=I | W | E | S]
      [,ICA=NOICA | ICA [ (
        [,MXREF ( S | L ) | NOMXREF ]
        [,CLEN | NOCLEN ]
        [,CVAR | NOCVAR ]
        [,MSG ( { NEW | NONE | ALL } ) ]
        [,MSGON (number1,number2,...) |
         MSGOFF (number1,number2,...) ]
        [,RCHECK | NORCHECK ] ) ]
      [,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]
      [,PTRSIZE=4 | 8 ]
      [,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]
      [,VECTOR=NOVECTOR | VECTOR [ (
       , [ REPORT [ ( optionlist ) ] | NOREPORT ]
       , [ INTRINSIC | NOINTRINSIC ]
       , [ IVA | NOIVA ]
       , [ REDUCTION | NOREDUCTION ]
       , [ SIZE ( { ANY | LOCAL | n } )
      ]
       , [ MODEL ( { ANY | VF2 | LOCAL } ) ]
       , [ SPRECOPT | NOSPRECOPT ]
       , [ ANZCALL | NOANZCALL ]
       , [ CMPLXOPT | NOCMPLXOPT ] ) ]
```

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

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. Changing CHARLEN can change the size of the object file.

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.

DDIM | NODDIM
Specifies whether pointee arrays that specify object-time dimensions are to be evaluated at each array element reference (DDIM) or at the time of entry into the subprogram (NODDIM).

For dynamically dimensioned arrays (DDIM), the array declarations can contain any integer variable of length 4 or 2. These variables are not restricted to appearing in a COMMON block or as dummy arguments, but the value of the integer variables must be known when the array element containing it is referenced.

Dynamically dimensioned arrays can be used in a Fortran main program.

EMODE = EMODE | NOEMODE
Specifies whether the code that is compiled for a subroutine or function can receive arguments that reside in an extended common block.

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).

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.

HALT (I | W | E | S)

Causes termination of the compile after any phase if the compiler return code is at or above the specified level.

I Selects return code 0 (Informational diagnostic level).
W Selects return code 4 (Warning diagnostic level).
E Selects return code 8 (Error diagnostic level).
S Selects return code 12 (Severe error diagnostic level).

The HALT(I) option terminates the compile after the first compiler phase, which is the syntax analysis phase.

If a compile of a source file is terminated because of the HALT option, an object file is not produced.

ICA = NOICA | ICA [(
[,MXREF (S | L) | NOMXREF]
[,CLEN | NOCLEN]
[,CVAR | NOCVAR]
[,MSG ({ NEW | NONE | ALL })]
[,MSGON (*number1,number2***,...) |**
MSGOFF (*number1,number2***,...)]**
[,RCHECK | NORCHECK])]

Specifies whether intercompilation analysis is to be performed, and controls the intercompilation analysis output. Specify ICA when you have a group of separately-compiled programs and subprograms that you want to process together and you need to know if there are any conflicting external references.

MXREF(S|L) | NOMXREF
Specifies whether to produce external cross-reference listings. To produce the shortest MXREF listing, use the default suboption with MXREF(S).

S Eliminates names from the list of references that make no references and are not referenced by other subprograms.

L Includes names in the list of references that make no references and are not referenced by other subprograms.

CLEN | NOCLEN
Specifies whether to check the length of named common blocks.

CVAR | NOCVAR
Specifies whether usage information for variables in a named common block is to be collected.

MSG({NEW | NONE | ALL})
Specifies the type of diagnostic messages to be printed.

NEW
Specifies that only messages about the new compilations will appear on the printout.

NONE
Specifies that only messages about deleting entries in an intercompilation analysis file will appear on the printout.

ALL
Specifies that all messages will be printed.

MSGON(*number1,number2,...***) | MSGOFF(***number1,number2,...***)**
Specifies the intercompilation analysis messages to be issued. MSGON and MSGOFF are mutually exclusive. With MSGON, the messages you specify are issued; all others are suppressed. With MSGOFF, the messages you specify are suppressed; all others are issued. If you specify neither MSGON nor MSGOFF, all messages are issued.

number
The message number; for example, 61 for message ILX0061I.

RCHECK | NORCHECK
Specifies whether to produce a report of recursive subroutine calls or function invocations. If you specify RCHECK, and recursion is detected, ICA will print the calling sequence of calls that can produce recursion.

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 application 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 between 7 and 32765. The advantage of using a large LINECNT number is that there are fewer page headings to look through if you are using a terminal. Your printed output 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 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.

NOOPTIMIZE
Specifies no optimization.

Note: 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).

PTRSIZE (4|8)
Sets the default length for pointer variables to 4 or 8 bytes. This can be overridden by explicit typing. Be aware that changing the pointer size will alter any common or equivalence association.

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 specified, 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.

TRMFLG = TRMFLG | NOTRMFLG

Presents the statement in error and the diagnostic message together, whenever possible, on your terminal.

VECTOR [(
[REPORT [(*optionlist*)] | NOREPORT]
[INTRINSIC | NOINTRINSIC]
[IVA | NOIVA]
[REDUCTION | NOREDUCTION]
[SIZE ({ ANY | LOCAL | *n* })]
[MODEL ({ ANY | VF2 | LOCAL })]
[SPRECOPT | NOSPRECOPT]
[ANZCALL | NOANZCALL]
[CMPLXOPT | NOCMPLXOPT])]
| NOVECTOR

Specifies whether to invoke the vectorization process, which produces programs that can use the speed of the vector facility. VECTOR instructs the compiler to transform eligible statements in loops into vector instructions. As much of a loop as possible is translated into vector object code and the remainder is translated into scalar object code.

Vectorization requires that the optimization level in effect be OPT(2) or OPT(3). If the optimization level is not OPT(2) or OPT(3), the compiler upgrades the optimization level to OPT(3) for you. However, the use of DEBUG or invalid Fortran statements downgrades the optimization level. As a result, the optimization level may be downgraded to OPT(0) and no vectorization occurs.

The VECTOR compile-time option includes the following suboptions:

REPORT [(*optionlist*)] | NOREPORT

Specifies whether to produce a report of vectorization information. REPORT instructs the compiler to either display the report on a terminal screen, or provide the information in a printed report.

The format and content of the report varies, depending on the REPORT options specified. In general, the report consists of a listing of source statements. Additional information is supplied in the margins of the listing and in tables at the end of the listing; for example, brackets that indicate loop nesting, flags that identify vectorized and nonvectorized loops, and messages that give more detailed information about the vectorization process.

If both PARALLEL and VECTOR are specified and the REPORT suboption is specified in one or both of the options, a single listing containing the vector and parallel reports is produced.

optionlist
Where the options specified are one or more of the strings, TERM, LIST, XLIST, SLIST, or STAT, separated by blanks or commas. Figure 32 shows the options, their abbreviations, and describes the output produced by each option.

Figure 32. VECTOR(REPORT) Options

Option	Abbreviation	Description
TERM	TE	Produces a vector report at the terminal when TRMFLG is specified.
LIST	LI	Produces a simple format of the vector report on an output listing. This suboption will help you understand how the statements and loops in the pseudo-assembler listing have been reordered.
XLIST	XL	Produces an extended format of the vector report on an output listing. Use XLIST when you want to tune your program to increase vectorization. It will help you understand why statements did or did not vectorize.
SLIST	SL	Produces a vector report, in a format similar to that produced by the SOURCE compile-time option, on an output listing. Use SLIST in conjunction with the VEC(IVA) option when you are trying to tune your program to improve overall performance. This allows you to use the vector tuning tools available in the IAD. Use SLIST in conjunction with the NOSOURCE option when you want to create a single listing for archive purposes.
STAT	ST	Produces a vector statistics table on an output listing. Use STAT when you are tuning a program for performance without the IAD; it will tell you about some of the assumptions made by the compiler that could have affected its economic decisions.

You can specify the options in any order; however, in the printed listing, the sections of the report appear in the following order: LIST, XLIST, SLIST, STAT.

The above reports can be requested individually or in any combination.

INTRINSIC | NOINTRINSIC
Specifies whether out-of-line intrinsic function references are to be vectorized.

The VS FORTRAN Version 2 math library routines (VSF2FORT) have been revised to be more accurate. The results generated by these new routines may be different from the results generated by the old VS FORTRAN Version 1 standard math routines (VSF2MATH).

For scalar out-of-line intrinsic function references in your program, you can choose which math library to use by accessing libraries in the desired order. If the intrinsic function references in your program are vectorized, however, the new VS FORTRAN Version 2 math library routines will always be used.

Therefore, if you wish to always use the old math routines for compatibility of results, you should not allow out-of-line intrinsic function references to vectorize.

If you use the INTRINSIC suboption, out-of-line intrinsic function references in your program are eligible for vectorization. If the new math library rou-

tines are used, the scalar results will be the same as the vector results. The new math routines yield the same results in both vector and scalar.

If you use the NOINTRINSIC suboption, out-of-line intrinsic function references in your program are vectorized. The scalar math routines of the library you have accessed first will be used.

IVA | NOIVA

Specifies whether to produce a program information file. This file is required by Interactive Debug if you use the interactive vectorization aid functions.

REDUCTION | NOREDUCTION

Specifies whether reduction functions are to be vectorized.

The translation of vector operations from scalar to vector code may produce different results. For example, because floating-point addition is not associative, vector code operations may not be performed in scalar processing order. This causes the results to be dependent on the order of accumulation of the floating-point data.

For more information, see *IBM Enterprise Systems Architecture/370 and System/370 Vector Operations* and *Enterprise Systems Architecture/390 Vector Operations*.

SIZE ({ANY | LOCAL | *n*})

Specifies the section size to be used to perform vector operations.

Specifying SIZE(ANY) causes the compiler to generate object code to use the section size of the computer on which the routine is running. The code may be slightly less efficient as that generated by SIZE(LOCAL) or SIZE(*n*) but you can move the program to a computer with a different section size without recompiling.

When you specify SIZE(LOCAL) the compiler uses the specified section size of the computer that compiled the program. If the section size is 0 or -1, the compiler uses the IBM-supplied default, ANY.

SIZE(*n*) allows you to specify an explicit section size. Using LOCAL or *n* produces slightly more efficient object code. However, if the specified section size is different from that of the computer's actual section size the program terminates upon the first processing of the routine compiled with the incompatible section size. The library issues a diagnostic message with a return code of 16.

If the specified section size is invalid—not a power of 2, or less than 8—the compiler issues an informational message and uses the IBM-supplied default, ANY.

MODEL ({ANY | VF2 | LOCAL })

Identifies the level of the ES/9000* vector facility on which the compiled program will be executed. The compiler generates code to use the enhanced features of that level.

When a program has been compiled for a specific level of the vector facility, the VS FORTRAN run-time vector support will allow the program to be executed only on a processor that can support that level of the vector facility.

ANY
Specifies that any level of the vector facility can be used to execute this program. The program will run on any IBM ES/3090* or ES/9000 processor that has the vector facility installed.

VF2
Specifies that the program will be executed on an ES/9000 processor with the basic level of the enhanced vector facility. Code compiled with this option can execute faster than code generated with MODEL(ANY).

LOCAL
Specifies that the program will be compiled for the level of the vector facility installed on the processor where it is compiled.

- If the compiling processor
  - Does not have the vector facility installed,
  - Is an ES/3090 processor
  - Is an ES/9000 processor without an enhanced vector facility

  MODEL(LOCAL) is equivalent to MODEL(ANY).
- If the compiling processor is an ES/9000 with the basic level of the enhanced vector facility, MODEL(LOCAL) is equivalent to MODEL(VF2).

SPRECOPT | NOSPRECOPT
Allows single-precision multiply-and-add and multiply-and-subtract sequences to be generated. If SPRECOPT is specified, the vectorization process will look for single-precision multiplications followed by single-precision additions or subtractions, and attempt to perform these sequences with a vector multiply-and-add or multiply-and-subtract instruction. These calculations give double-precision results, and use double-precision values for the intermediate product. Processing is faster and the result is more accurate than for two separate single-precision operations, but you should be aware that the result might be different from what you expected.

ANZCALL | NOANZCALL
Specifies whether an analysis of CALL statements and user function references within DO loops is to be performed when the VECTOR option is in effect. While greater vectorization might be achieved with ANZCALL, compilation time will increase for eligible loops.

CMPLXOPT | NOCMPLXOPT
Specifies whether vectorization is to optimize loading and storing of single-precision complex (COMPLEX*8) data using long-precision real vector instructions.

* ES/3090 and ES/9000 are trademarks of the International Business Machines Corporation.

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=number | 7 ]
     [,ERRMSG=number | 6 ]
     [,PRINTER=number | 6 ]
     [,READER=number | 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 36.

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 = *number* **| 7**
Specifies, for LANGLVL(66) only, the standard I/O unit number for 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 = *number* **| 6**
Specifies the standard I/O unit number for the FORTRAN 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 = *number* **|** **6**

Specifies the standard I/O unit number for 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 = *number* **|** **5**

Specifies the standard I/O unit number for 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 logical record length for sequential formatted files. *Number* is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SFRECFM = V, VA, VB or VBA) or 1 to 32760 for all other record formats.

SULRECL = *number* **| -1**
Specifies the logical record length for sequential unformatted files. *Number* is an integer expression of length 4 bytes; valid range is from 1 to 32756 for variable record formats (SFRECFM = V, VA, VB or VBA), 1 to 32760 for all other record formats or -1 which specifies an unlimited record length. For SURECFM of VS or VBS, -1 is valid.

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 for CMS and MVS*.

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 infor-

mation on I/O, see the *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

DMAXRE = *number* **| 50**
Specifies the number of records in a direct file. It is valid for new DASD files only; 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 the *VS FORTRAN Version 2 Programming Guide for CMS and MVS*.

There are no operands to set the default values for the run-time options AUTOTASK, PARALLEL, and PARTRACE; therefore, these options cannot be changed at installation time. They can, however, be changed at run time.

There are no operands in the VSF2PARM macro to set the default values for the run-time options ERRUNIT, RDRUNIT, PRTUNIT, and PUNUNIT. However, the default I/O unit values for these units can be changed during installation through the Unit Attribute Table. See "VSF2UAT: Changes I/O Unit Number Option Defaults" on page 56.

Syntax of VSF2PARM Macro

```
VSF2PARM
     SCOPE = GLOBAL
     [,ABSDUMP | NOABSDUMP]
     [,CNVIOERR | NOCNVIOERR]
     [,DEBUG | NODEBUG]
     [,DEBUNIT(s1[,s2,...]) | NODEBUNIT]
     [,ECPACK | NOECPACK]
     [,FAIL(ABEND | RC |ABENDRC)]
     [,FILEHIST | NOFILEHIST
     [,INQPCOPN | NOINQPCOPN]
     [,IOINIT | NOIOINIT]
     [,OCSTATUS | NOOCSTATUS]
     [,RECPAD[(ALL)] | NORECPAD]
     [,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.

CNVIOERR | NOCNVIOERR

Specifies whether input conversion errors 206, 212, 213, 215, 225, 226, and 238 will be treated as I/O errors. (See *VS FORTRAN Version 2 Language and Library Reference* for additional information.)

CNVIOERR

Causes ERR and IOSTAT specifiers on the READ statements to be honored, regardless of whether they were specified, for the errors listed above. (ERR and IOSTAT are also honored on formatted WRITE statements for error 212.)

NOCNVIOERR

Causes the ERR and IOSTAT specifiers not to be honored for the errors listed above.

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 57 for information on the UNTABLE option.

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

ECPACK | NOECPACK
Specifies whether a data space should be filled with as many extended common blocks as possible before a new data space is allocated.

ECPACK
Specifies that extended common blocks be placed into the fewest possible number of data spaces. This option reduces some of the overhead associated with referencing data spaces.

NOECPACK
Specifies that each extended common block be placed into a separate data space. As a result, reference errors made beyond the bounds of an extended common block might be more easily detected.

FAIL (ABEND | RC | ABENDRC)
Indicates how unsuccessfully executed programs are to be terminated: either by a nonzero return or by an abnormal termination (ABEND). The suboptions of the FAIL option have the following meanings:

ABEND
Causes the program to end by an abnormal termination (ABEND) with a user completion code of 240.

RC
Causes the program to end normally, but with a nonzero return code (16).

ABENDRC
Causes the program to end by abnormal termination (ABEND) when failure is because of a condition for which the operating system would usually cause an ABEND; and to end with a nonzero return code when failure is by some condition detected by VS FORTRAN.

FILEHIST | NOFILEHIST
Specifies whether to allow the file definition of a file referred to by a ddname to be changed at run time.

FILEHIST
Causes the history of a file to be used in determining its existence. In particular it checks to see whether:

- The file was ever internally opened (in which case it exists)
- The file was deleted by a CLOSE statement (in which case it does not exist)

When FILEHIST is specified, you cannot change the file definition of a file at run time and have the same results produced as previous VS FORTRAN releases.

NOFILEHIST
Causes the history of a file to be disregarded in determining its existence.

If you specify NOFILEHIST you should consider:

- **If you change file definitions at run time:** the file is treated as if it were being opened for the first time. Note that before the file definition can be changed, the existing file must be closed.
- **If you do not change file definitions at run time:** you must use STATUS=NEW to re-open an empty file that has been closed with STATUS=KEEP, because the file does not appear to exist to Fortran.

INQPCOPN | NOINQPCOPN

Specifies whether or not a unit is connected to a file when executing an INQUIRE by unit.

INQPCOPN

Indicates that, if a unit is connected to a file, even if it was preconnected and no I/O statement has been executed, a value of true is returned in the variable or an array element given in the OPENED specifier from an INQUIRE by unit statement.

NOINQPCOPN

Indicates that, if a unit is internally open, a value of true is returned in the variable or an array element given in the OPENED specifier for an INQUIRE by unit statement.

"Internally open" means that the unit is connected to a file by an OPEN statement, or if the unit has been preconnected, that a READ, WRITE, PRINT, REWIND or ENDFILE statement has been successfully executed.

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.

RECPAD[(ALL)] | NORECPAD
Specifies whether a formatted input record is padded with blanks.

RECPAD
Causes a formatted input record within an internal file or a varying/undefined length record (RECFM=U or V) external file to be padded with blanks when an input list and format specification require more data from the record than the record contains. Blanks added for padding are interpreted as though the input record actually contains blanks in those fields. If ALL is specified, a formatted input record is padded regardless of the record format of the file.

NORECPAD
Specifies that an input list and format specification must not require more data from an input record than the record contains. If more data is required, error message AFB212I is issued.

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

Recommendation: If you are using the DEBUG option, specify SPIE to avoid termination of interactive debug when an 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.

STAE | NOSTAE

Specifies whether the run-time environment will take control if an abnormal termination occurs.

Recommendation: If you are using the DEBUG option, specify STAE to avoid termination of interactive debug when an interrupt 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.

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 the "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 500 and continue sequentially, up to a maximum of 899, for a maximum of 400 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 between 1 and 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 can 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 "0," or any number greater than 255, means the error can occur an unlimited number of times.

Note: 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 the *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 "0" 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 the *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. The routine should not be written in VS FORTRAN and should be reentrant.

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 AFBVRENA and AFBVRENC. Then, if the user error exit routine must communicate with the 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 VSF2AMTB required macro instruction. The syntax is shown on page 69.
2. Define the table contents for individual error message entries. Follow the syntax for the VSF2AMTB required macro instruction with the necessary number of optional macro instructions, one for each message you wish to create, followed by an END statement. The syntax for the VSF2AMTB optional macro instruction is shown on page 70.
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 VSF2AMTB required macro instruction shown below.

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: Required Macro Instruction

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 *pid*UOPT. 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: Optional Macro Instruction

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 66. 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.

Reasons for Using Composite Modules

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 33. 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

Figure 33. Composite Module Characteristics

Composite Module Name	Can Be Installed In a shared segment	Used for 370 Environments	Used for XA Environments	Location In Relation to 16M line (XA environment only)	Contents
AFBVRENA	Yes	No	Yes	Above	Reentrant library modules
AFBVRENB	Yes	No	Yes	Below	Reentrant library modules
AFBVRENC	Yes	Yes	Yes[1]	Not Applicable	Reentrant library modules
AFBVRENP	Yes	No	Yes	Above	Reentrant library modules for parallel environment

Note:

[1] Only used for XA environment if the system size is less than 16 megabytes.

Customization Tips

- If AFBVRENC is not in a discontiguous shared segment, 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 shared segment, 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 shared segment or if you delete any of them from a composite module, they will continue to be included in the "Principal Text" library.
- AFBVRENP cannot be customized. It can be placed in a shared segment to reduce the region size required to run each parallel program, or it can be loaded into your region to avoid using extra storage in the shared segment.

Tables of Required and Optional Modules

Composite Module AFBVRENA (XA Environment Only)

Figure 34 (Page 1 of 2). Required Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBCAREN	254	X	Internal linkage module
AFBCLBC0	2138	X	Library common work area
AFBCLOAD	664	X	Loader
AFBCSTIO	400	X	Standard I/O unit initialization
AFBCUAT(1)	D8E	X	Unit attribute table
AFBCVIO$	C0	X	Internal linkage routine
AFBUOPT	D30	X	Error option table
AFBVABEX	1C68	X	Abend processor
AFBVBLN$	80	X	Internal linkage routine
AFBVCDM$	80	X	Internal linkage routine
AFBVCNI$	80	X	Internal linkage routine
AFBVCNO$	80	X	Internal linkage routine
AFBVCOM$	80	X	Internal linkage routine
AFBVCVT$	3E0	X	Internal linkage routine
AFBVDEB$	80	X	Internal linkage routine
AFBVDIO$	88	X	Internal linkage routine
AFBVDYN$	80	X	Internal linkage routine
AFBVEMG$	B8	X	Internal linkage routine
AFBVERE$	80	X	Internal linkage routine
AFBVERS$	A0	X	Internal linkage routine

Figure 34 (Page 2 of 2). Required Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBVFNTH	9D0	X	Program interrupt handler
AFBVGMFM	270	X	GETMAIN/FREEMAIN
AFBVGPRM	F0	X	Default run-time options
AFBVIAD$	80	X	Internal linkage routine
AFBVIIO$	88	X	Internal linkage routine
AFBVINI$	80	X	Internal linkage routine
AFBVINTP	2008	X	Initialization/termination
AFBVKIO$	88	X	Internal linkage routine
AFBVLOC$	80	X	Internal linkage routine
AFBVPARM	17E8	X	Run-time options processor
AFBVPIO$	88	X	Internal linkage routine
AFBVPOS$	80	X	Internal linkage routine
AFBVRDCB	2A0	X	DCB information resolution
AFBVSPIE	1A0	X	Interrupt interceptor
AFBVSTAE	1C0	X	Abend control routine
AFBVTRC$	80	X	Internal linkage routine
AFBVTRMF	140	X	File closing at termination
Total	**C00E**	**37**	

Notes:

(1) May increase in size when the unit attribute table is customized.

Figure 35 (Page 1 of 3). Optional Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBCCPTP	128	X	Processor time processing routine
AFBCDYNA	9B0	X	Dynamic file allocation processor
AFBCFISC	4CA	X	FILE specifier scan routine
AFBDIOCP	1E0	X	Define file (LANGLVL 66)
AFBDSPAP(1)	560	X	Dimension calculator
AFBIBCOP(2)	D20	X	Pre-VS FORTRAN interface
AFBLDFIP(1)	700	X	List-directed I/O
AFBNAMEP(1)	588	X	Namelist I/O
AFBPNPRP	D8		NPROCS processor
AFBSDUMQ	3940		SDUMP subroutine
AFBTFORP	158		Debugging Interface
AFBVAMTP	1D8		Alt. error option table processor
AFBVASGP(3)	2040		DBCS assignment processor

Figure 35 (Page 2 of 3). Optional Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBVBALG	5C8	X	Boundary alignment routine
AFBVBLNT	3A8	X	Implied DO in I/O
AFBVCDMA	278	X	Common block directory maintenance
AFBVCLOP	478	X	CLOSE statement
AFBVCOMH	2030	X	Formatted I/O
AFBVCONI	330	X	Input floating-point conversion
AFBVCONO	840	X	Output floating-point conversion
AFBVCVTH	29F8	X	Data conversion
AFBVDBUP	1468		Debugging packet
AFBVDEBU	208		DEBUNIT parameter processor
AFBVDOCP	168	X	Divide check and overflow test
AFBVDUMQ	8D8	X	DUMP/PDUMP subroutine
AFBVEMGN	1510	X	Error message generator
AFBVERRE	268	X	Error summary
AFBVEXIP	128	X	Return code processor
AFBVFINP	C38	X	FILEINF call file information block
AFBVFMTP	260		Language Conversion Program define file
AFBVIIOS	370	X	Internal file services
AFBVINQP	20E8	X	INQUIRE statement
AFBVINTH	500	X	Vector program interrupt handler
AFBVIOCP	4E0	X	BACKSPACE, REWIND, ENDFILE
AFBVIOFP	EA0	X	Formatted I/O
AFBVIOLP	1EE0	X	List-directed I/O
AFBVIONP	1E58	X	Namelist I/O
AFBVIOUP	1520	X	Unformatted I/O
AFBVLINP	298	X	Link to reentrant CSECT
AFBVLOCA	8F0	X	Statement number locator
AFBVMOPP	660	X	Extended error handling
AFBVMSKL	8378	X	Message skeletons
AFBVOCMP	1C00	X	Obtain common blocks
AFBVOPEP	1CA8	X	OPEN statement
AFBVPOSA	4168		Post ABEND processor
AFBVSCOP(4)	9D0	X	Pre-Release 4 interface
AFBVSPAP	8A8	X	Dimension calculator
AFBVSPIP	4B0	X	Dynamic spill area processor
AFBVTEN	2C0	X	Powers of ten table
AFBVTIMP	6E8	X	Date/time/clock routine
AFBVTRCH	EA8	X	Traceback generator

Figure 35 (Page 3 of 3). Optional Modules for AFBVRENA

Module	Approx. Size	Default Set	Function
AFBVUNIN	220	X	Unnormalized operand interrupt handler
AFBVVINI	390	X	Vector initialization

Notes:

(1) 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.

(2) 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.

(3) Included in AFBVASGP are AFBDBGVB, AFBDBMOV, AFBDBMVE, AFBDBPAD, AFBDBTRC, and AFBDBTRT. The size of AFBVASGP also incorporates the sizes of the other modules.

(4) 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.

Composite Module AFBVRENB (XA Environment Only)

Figure 36. Required Modules for AFBVRENB

Module	Approx. Size	Default Set	Function
AFBCBREN	D4	X	Internal linkage module
AFBCFIST	1884	X	File status processor
AFBVSIOS	53A8	X	Sequential I/O services
Total	**6D00**	**3**	

Figure 37. Optional Modules for AFBVRENB

Module	Approx. Size	Default Set	Function
AFBVDIOS	24F0		Direct access I/O services
AFBVPIOS	2EF0		Sequential striped I/O processing

Composite Module AFBVRENC (370 or XA Environment)

AFBVRENC is used in a 370 environment, or an XA environment if the system size is less than 16 megabytes.

Figure 38 (Page 1 of 2). Required Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBCFIST	1860	X	File status processor
AFBCLBC0	2130	X	Library common work area

Figure 38 (Page 2 of 2). Required Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBCLOAD	664	X	Loader
AFBCREN	284	X	Internal linkage module
AFBCSTIO	400	X	Standard I/O initialization
AFBCUAT(1)	D8E	X	Unit attribute table
AFBCVIO$	C0	X	Internal linkage routine
AFBUOPT	D30	X	Error option table
AFBVABEX	1C68	X	Abend processor
AFBVBLN$	80	X	Internal linkage routine
AFBVCDM$	80	X	Internal linkage routine
AFBVCNI$	80	X	Internal linkage routine
AFBVCNO$	80	X	Internal linkage routine
AFBVCOM$	80	X	Internal linkage routine
AFBVCVT$	3E0	X	Internal linkage routine
AFBVDEB$	80	X	Internal linkage routine
AFBVDIO$	88	X	Internal linkage routine
AFBVDYN$	80	X	Internal linkage routine
AFBVEMG$	B8	X	Internal linkage routine
AFBVERE$	80	X	Internal linkage routine
AFBVERS$	A0	X	Internal linkage routine
AFBVFNTH	9D0	X	Program interrupt handler
AFBVGMFM	270	X	GETMAIN/FREEMAIN
AFBVGPRM	F0	X	Default run-time options
AFBVIAD$	80	X	Internal linkage routine
AFBVIIO$	88	X	Internal linkage routine
AFBVINI$	80	X	Internal linkage routine
AFBVINTP	2008	X	Initialization/termination
AFBVKIO$	88	X	Internal linkage routine
AFBVLOC$	80	X	Internal linkage routine
AFBVPARM	17E8	X	Run-time options processor
AFBVPIO$	88	X	Internal linkage routine
AFBVPOS$	80	X	Internal linkage routine
AFBVRDCB	2A0	X	DCB attributes resolution
AFBVSIOS	53A8	X	Sequential I/O services
AFBVSPIE	1A0	X	Interrupt interceptor
AFBVSTAE	1C0	X	Abend control routine
AFBVTRC$	80	X	Internal linkage routine
AFBVTRMF	140	X	File closing at termination
Total	**12C3E**	**39**	

Note:

(1) May increase in size when the unit attribute table is customized.

Figure 39 (Page 1 of 2). Optional Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBCCPTP	128		Processor time processing routine
AFBCDYNA	9B0		Dynamic file allocation processor
AFBCFISC	4CA		FILE specifier scan routine
AFBCVIOS	248C		Nonkeyed VSAM I/O services
AFBDIOCP	1E0		Define file (LANGLVL 66)
AFBDSPAP(1)	560		Dimension calculator
AFBIBCOP(2)	D20		Pre-VS FORTRAN interface
AFBLDFIP(1)	700		List-directed I/O
AFBNAMEP(1)	588		Namelist I/O
AFBPNPRP	D8		NPROCS processor
AFBSDUMQ	3940		SDUMP subroutine
AFBTFORP	158		Debugging Interface
AFBVAMTP	1D8		Alternate error option table processor
AFBVASGP(3)	2040		DBCS assignment processor
AFBVBALG	5C8		Boundary alignment routine
AFBVBLNT	3A8	X	Implied DO in I/O
AFBVCDMA	278		Common block directory maintenance
AFBVCLOP	478	X	CLOSE statement
AFBVCOMH	2030	X	Formatted I/O
AFBVCONI	330	X	Input floating-point conversion
AFBVCONO	840	X	Output floating-point conversion
AFBVCVTH	29F8	X	Data conversion
AFBVDBUP	1468		Debugging packet
AFBVDEBU	208		DEBUNIT parameter processor
AFBVDIOS	24F0		Direct access I/O services
AFBVDOCP	168		Divide check and overflow test
AFBVDUMQ	8D8		DUMP/PDUMP subroutine
AFBVEMGN	1510	X	Error message generator
AFBVERRE	268	X	Error summary
AFBVEXIP	128		Return code processor
AFBVFINP	C38		FILEINF call file information block
AFBVFMTP	260		Language Conversion Program define file
AFBVIIOS	370		Internal file services
AFBVINQP	20E8		INQUIRE statement
AFBVINTH	500		Vector program interrupt handler

Figure 39 (Page 2 of 2). Optional Modules for AFBVRENC

Module	Approx. Size	Default Set	Function
AFBVIOCP	4E0		BACKSPACE, REWIND, ENDFILE
AFBVIOFP	EA0	X	Formatted I/O
AFBVIOLP	1EE0	X	List-directed I/O
AFBVIONP	1E58		Namelist I/O
AFBVIOUP	1520	X	Unformatted I/O
AFBVKIOS	3278		Keyed access I/O services
AFBVLINP	298		Link to reentrant CSECT
AFBVLOCA	8F0	X	Statement number locator
AFBVMOPP	660		Extended error handling
AFBVMSKL	8378	X	Message skeletons
AFBVOCMP	1C00		Obtain common blocks
AFBVOPEP	1CA8	X	OPEN statement
AFBVPIOS	2EF0		Sequential striped I/O processing
AFBVPOSA	4168		Post ABEND processor
AFBVSCOP(4)	9D0		Pre-Release 4 interface
AFBVSPAP	8A8		Dimension calculator
AFBVSPIP	4B0		Dynamic spill area processor
AFBVTEN	2C0	X	Powers of ten table
AFBVTIMP	6E8		Date/time/block routine
AFBVTRCH	EA8	X	Traceback generator
AFBVUNIN	220		Unnormalized operand interrupt handler
AFBVVINI	390		Vector initialization

Notes:

(1) 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.

(2) 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.

(3) Included in AFBVASGP are AFBDBGVB, AFBDBMOV, AFBDBMVE, AFBDBPAD, AFBDBTRC, and AFBDBTRT. The size of AFBVASGP also incorporates the sizes of the other modules.

(4) 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.

Composite Module AFBVRENP (XA Environment only)

The composite module AFBVRENP cannot be customized. The approximate size for the module is 13980.

Note: Used for 370 environments only in an XA machine.

Appendix D. Servicing VS FORTRAN Version 2

IBM National Service 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*.

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**).

With VS FORTRAN Version 2 Release 6, PTFs will also include updates to the ReadMe file, which is updated with new information pertaining to this product as the result of service updates, as well as any additional information provided in response to Reader Comment Forms.

When reporting a problem, you may need the Field Engineering service numbers (**FESN**). These are shown in Figure 40.

Figure 40. Field Engineering Service Numbers

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

Corrective Service

Corrective service is distributed as a PTF 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 (complete product)	EXEC I5668806 PTFINST
VS FORTRAN Version 2 (compiler and library)	EXEC I5688087 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 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: 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 the command `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

L

M

N

O

P

Q

R

S

T

U

V

W

X

We'd Like to Hear from You

VS FORTRAN Version 2
Installation and Customization for CMS
Release 6

Publication No. SC26-4339-05

Please use one of the following ways to send us your comments about this book:

- Mail—Use the Readers' Comments form on the next page. If you are sending the form from a country other than the United States, give it to your local IBM branch office or IBM representative for mailing.
- Fax—Use the Readers' Comments form on the next page and fax it to this U.S. number: 800-426-7773.
- Electronic mail—Use one of the following network IDs:
  - IBMLink: HLASMPUB at STLVM27
  - Internet: COMMENTS@VNET.IBM.COM

  Be sure to include the following with your comments:
  - Title and publication number of this book
  - Your name, address, and telephone number if you would like a reply

Your comments should pertain only to the information in this book and the way the information is presented. To request additional publications, or to comment on other IBM information or the function of IBM products, please give your comments to your IBM representative or to your IBM authorized remarketer.

IBM may use or distribute your comments without obligation.

Readers' Comments

VS FORTRAN Version 2
Installation and Customization for CMS
Release 6

Publication No. SC26-4339-05

How satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Technically accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐
Grammatically correct and consistent	☐	☐	☐	☐	☐
Graphically well designed	☐	☐	☐	☐	☐
Overall satisfaction	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

May we contact you to discuss your comments? ☐ Yes ☐ No

Name

Address

Company or Organization

Phone No.

Readers' Comments
SC26-4339-05

Cut or Fold
Along Line

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, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

Department J58
International Business Machines Corporation
PO BOX 49023
SAN JOSE CA 95161-9945

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

File Number: S370-34
Program Number: 5668-805
5668-806
5688-087

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

The VS FORTRAN Version 2 Library

LY27-9516	Diagnosis Guide
GC26-4219	General Information
SC26-4340	Installation and Customization for MVS
SC26-4339	Installation and Customization for CMS
SC26-4420	Installation and Customization for AIX/370
SC26-4223	Interactive Debug Guide and Reference
SC26-4221	Language and Library Reference
GC26-4225	Licensed Program Specifications
SC26-4603	Master Index and Glossary
SC26-4686	Migration from the Parallel FORTRAN PRPQ
SC26-4741	Programming Guide for AIX/370
SC26-4222	Programming Guide for CMS and MVS
SX26-3751	Reference Summary

SC26-4339-05

Spine information:

VS FORTRAN Version 2 Installation and Customization for CMS *Release 6*