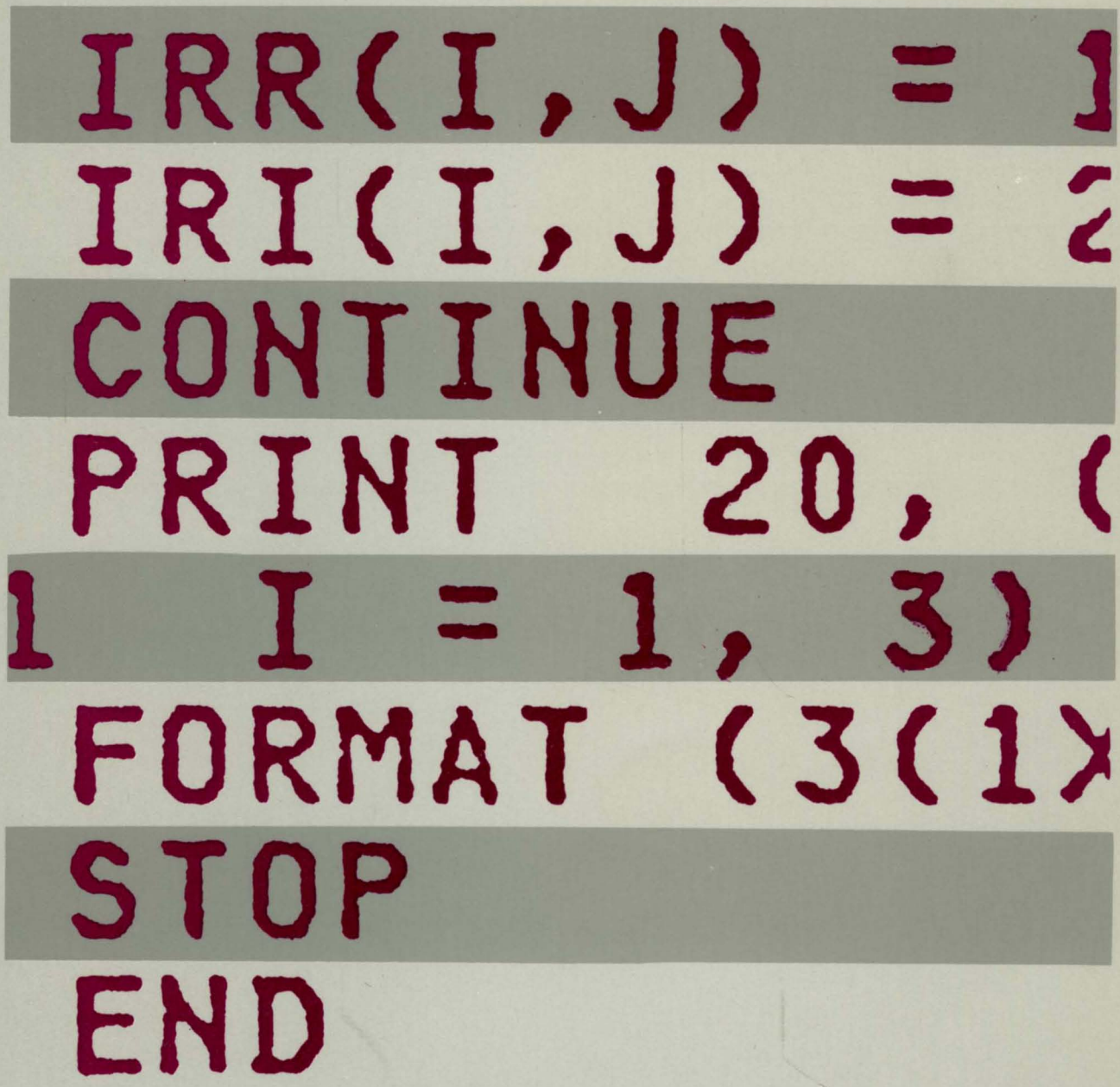
IBM

VS FORTRAN Compiler and Library Installation and Customization

Program Product



Program Numbers:

5748-FO3 Compiler and Library
5748-LM3 Library Only

Release 4.0

VS FORTRAN Compiler and Library Installation and Customization

Program Numbers
5748-FO3 (Compiler and Library)
5748-LM3 (Library Only)
Release 4.0

SC26-3987-3

Fourth Edition (October 1984)

This is a major revision of, and makes obsolete, SC26-3987-2.

This edition applies to Release 4.0 of VS FORTRAN, Program Products 5748-F03 (Compiler and Library) and 5748-LM3 (Library only), and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

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

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

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

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

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

PREFACE

This manual is designed for system programmers and planners who supervise the generation and maintenance of an organization's operating system. It contains material for installing the VS FORTRAN Compiler and Library and is to be used in conjunction with the VS FORTRAN Program Directory that applies to your system. Version 1, Release 4, Modification 0 of VS FORTRAN is referred to in this manual as Release 4.0.

ORGANIZATION OF THIS BOOK

- **Part 1** contains high-level information for installing VS FORTRAN. It includes a brief description of the VS FORTRAN Compiler and Library, and a list of installation prerequisites. This section also lists the system, machine and storage requirements for installation of the compiler and library, and describes the macros needed at installatior time.
- **Part 2** describes the installation process as it applies to the particular operating systems under which VS FORTRAN can be installed. It includes chapters for MVS/SP and MVS/XA, VSE/Advanced Functions, and VM/SP.
- **Part 3** describes the features of VS FORTRAN that may be customized for your installation. It includes chapters for customization features that are specific to MVS/SP and MVS/XA, VSE/Advanced Functions, and VM/SP.

There are also two Appendixes.

- **Appendix A** explains the program product support services and structure.
- **Appendix B** lists Compiler and Library modules.

INDUSTRY STANDARDS

The VS FORTRAN Compiler and Library program product is designed according to the specifications of the industry standards listed below, as understood and interpreted by IBM as of May, 1982.

The following two standards are technically equivalent. In this manual, references to the **current standard** 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 defined in ANSI/ISA-S61.1.

The following two standards are technically equivalent. In this manual, references to the **old standard** are references to these two standards:

- American Standard FORTRAN, X3.9-1966 (also known as FORTRAN 66)
- International Organization for Standardization ISO R 1539-1972 Programming Languages-FORTRAN

Both the FORTRAN 77 and the FORTRAN 66 standard languages include IBM extensions. In this book 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.

RELATED PUBLICATIONS

VS FORTRAN

VS FORTRAN Programming Guide, SC26-4118

VS FORTRAN Language and Library Reference, SC26-4119

VS FORTRAN Compiler and Library Reference Summary, SX26-3731

VS FORTRAN Compiler and Library: Diagnosis Guide, SC26-3990

MVS/SP

OS/VS Virtual Storage Access Method (VSAM) Programmer's Guide, GC26-3838

OS/VS System Modification Program (SMP): System Programmer's Guide, GC28-0673

OS/VS System Modification Program (SMP): Messages and Codes, GC38-1047

System Modification Program Extended (SMP/E): User's Guide, SC28-1302

System Modification Program Extended (SMP/E): Messages and Codes, GC28-1108

OS/VS2 MVS Data Management Services Guide, GC26-3875

OS/VS2 Access Method Services, GC26-3841

OS/VS2 MVS JCL, GC28-0692

OS/VS2 MVS Supervisor Services and Macro Instructions, GC28-1114

MVS/EXTENDED ARCHITECTURE (MVS/XA)

MVS/Extended Architecture Integrated Catalog Administration: Access Method Services Reference, GC26-4019

MVS/Extended Architecture JCL, GC28-1148

MVS/Extended Architecture Data Administration Guide, GC26-4013

MVS/Extended Architecture VSAM Administration Guide, GC26-4015

MVS/Extended Architecture Supervisor Services and Macro Instructions, GC28-1154

VSE/ADVANCED FUNCTIONS

VSE/Advanced Functions System Management Guide, SC33-6094

VSE System Data Management Concepts, GC24-5209

VSE/Advanced Functions System Control Statements, SC33-6095

VSE/Advanced Functions Maintain System History Program (MSHP) User's Guide, SC33-6101

VSE/VSAM Programmer's Reference, SC24-5145

Using VSE/VSAM Commands and Macros, SC24-5144

VM/SP

VM/SP CP Command Reference for General Users, SC19-6210

VM/SP CMS Command and Macro Reference, SC19-6209

VM/SP CMS User's Guide, SC19-6210

VM/SP Planning Guide and Reference, SC19-6201

VM/SP Installation Guide, SC24-5237

VM/SP System Programmer's Guide, SC19-6203

VM/PC

IBM Virtual Machine/Personal Computer User's Guide, SC24-5254

VS FORTRAN INTERACTIVE DEBUG

VS FORTRAN Interactive Debug Guide and Reference, SC26-4116

VS FORTRAN Interactive Debug Installation, SC26-4117

VS FORTRAN Interactive Debug Reference Summary, SX26-3742

PROGRAM SUPPORT

Field Engineering Programming Systems General Information, G229-2228

RELEASE NOTES

VS FORTRAN COMPILER AND LIBRARY

RELEASE 4.0, OCTOBER 1984

VSAM Key-Sequenced Data Sets

VS FORTRAN programs can now load and access VSAM KSDS files:

- Records can be retrieved, added, replaced, and deleted, using key values (designated fields within the records).
- Both direct and sequential processing (by key value) are allowed.
- Multiple alternate keys, as well as a primary key, can be used.

REWRITE and DELETE statements have been added to the language to process these files, and some existing I/O statements have been expanded.

Reentrant Object Code (MVS and VM)

The compiler can create a reentrant version of the object-code portion of a program. When object code is reentrant (and placed in a reentrant area), multiple end-users can share a single copy of it, thereby saving execution-time storage.

Execution-Time Loading of Library Routines

The library has been restructured to allow more execution-time loading of library routines. This has multiple benefits:

- Reduces auxiliary storage requirements for load modules.
- Speeds execution for users in compile-link-go mode.
- In an MVS/XA environment, allows many library routines to reside above 16 megabytes, thus providing virtual-storage constraint relief.

(This new library design will not impact users who have Release 2 or Release 3 load modules that access the old reentrant I/O library (via IFYVRENT), and who do not want to relink. Maintenance is automatically provided, and relinking is only necessary if Release 4.0 function is desired.)

Automatic Precision Increase

This feature allows a user to boost the precision of floating-point items in an existing program without recoding it. Single precision items can be made double; double can be made extended. Users merely recompile the program with a specified option (AUTODBL).

Faster Character Handling

Character assignment and comparison operations are now handled by in-line code, rather than by calls to the Library. This speeds execution time, and eliminates all error messages previously issued, including overlap detection.

Improved Diagnostic Support

The following enhancements allow easier program maintenance and debugging.

- MAP and XREF output can be formatted to fit a terminal screen.
- LIST output gives ISNs, and XREF output identifies variables referenced but not initialized.
- An explicit SDUMP compiler option is available (previously, this was available only as an installation-wide default).
- Object module size has been decreased by condensing and simplifying the SDUMP table. The symbol table size, however, remains the same.
- Execution-time error messages have been expanded to supply line numbers, ISNs, and offsets.

Improved I/O Support

The following improvements have been made to VS FORTRAN I/O statements:

- For sequential unformatted I/O, you can now use all record formats. Fixed (blocked or unblocked), undefined, variable (blocked or unblocked), and variable spanned (blocked or unblocked) formats are supported.
- You can specify a character type unit designator for list-directed READ and WRITE statements. This allows you to do list-directed reads and writes to an internal file.
- The NUM parameter is now a valid control list parameter for all unformatted I/O statements for LANGLVL(77). The NUM parameter returns the number of bytes transferred, and suppresses the indication of an error if the I/O list represents more data contained in the record.
- Several extensions have been made to the NAMELIST READ and WRITE statements. You can now use the keywords UNIT and FMT. The unit designator for NAMELIST I/O can be character type, so you can do NAMELIST reads and writes to an internal file. The unit designator can also be an asterisk to represent an installation-dependent unit. You can now use a NAMELIST in the PRINT statement at LANGLVL(77).

Miscellaneous Changes

- You can now use data initialization values in the character and double precision, explicit-type statements.
- The SC compiler option has been deleted.

RELEASE 3.1, MARCH 1984

VS FORTRAN Interactive Debug Support

When a VS FORTRAN program is executed, the user has a choice of two different execution options:

- DEBUG, which activates VS FORTRAN Interactive Debug immediately; and
- NODEBUG, the IBM default, which does not invoke VS FORTRAN Interactive Debug.

Note: The TEST compiler option is not necessary for VS FORTRAN Interactive Debug.

RELEASE 3.0, MARCH 1983

Character Data Type Handling

VS FORTRAN Release 3.0 provides for passing CHARACTER length arguments in a manner that is not apparent to the user.

In addition:

- CHARACTER and non-CHARACTER data types are allowed in the same COMMON block.
- CHARACTER and non-CHARACTER data types are allowed in an EQUIVALANCE relationship.
- The CHARLEN compiler option may be specified to set the maximum length of the CHARACTER data type to a range of 1 through 32767. The default maximum length remains 500 characters, or whatever was set at installation time.
- The SC option has been removed because the character length is now passed in a manner that is not apparent to the user.

Debugging and Diagnostic Aids

- The TRMFLG compiler option may be specified to display a source statement in error on the SYSTERM data set, along with the diagnostic message.
- A symbolic dump of variables at abnormal termination can be obtained for modules not compiled with the NOSDUMP compiler option.
- A symbolic dump of variables in a module not compiled with the NOSDUMP option can be obtained on request by calling the SDUMP library routine.
- The SYM compiler option may be specified to produce SYM cards along with the object deck.
- The SRCFLG compiler option may be specified to insert diagnostic messages in the printed source listing.

INCLUDE Statement Improvement

- INCLUDE statements can be selectively activated during compilation.
- Blocked file support has been added to the INCLUDE facility.

Miscellaneous Changes

- OPEN, CLOSE, and INQUIRE parameters that are constants are checked at compile time.
- VS FORTRAN continues executing after transmission input/output errors have occurred.
- Formatting for a new direct-access data set has been provided for the OPEN statement.
- For direct-access I/O, the records of a file must be either all formatted or all unformatted, not mixed.
- Various service changes have been made.

Warning: Every program that has been compiled with versions of VS FORTRAN previous to Release 3.0, and that either references or defines a user subprogram that has character-type arguments or is itself of character type, must be recompiled with VS FORTRAN Release 3.0.

CONTENTS

PART 1. INSTALLATION PLANNING GUIDE

CHAPTER 1. INTRODUCTION

The VS FORTRAN Compiler and Library, and the VS FORTRAN Library only, are available as separate program products. Each program product is distributed on its own tape containing the necessary modules, as well as the test program for verifying the installation procedures. If you have ordered the VS FORTRAN Library only, refer to the appropriate section about your system for any special installation considerations.

OVERVIEW OF THE PRODUCT

The VS FORTRAN Compiler translates programs written in the VS FORTRAN language and produces object modules for subsequent execution with the support of a suitable FORTRAN library.

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

WHERE TO FIND MORE INSTALLATION INFORMATION

In the following sections of this book, we describe the installation requirements and the steps needed to install the Compiler and Library. Before installing VS FORTRAN Release 4.0, contact your IBM Support Center or check the RETAIN/370 Preventive Service Planning (PSP) Facility for updates to the information and procedures in this book.

For specific information on space allocations, module and macro numbers, and other details needed to install the Compiler and Library, see the VS FORTRAN Program Directory for your system. The program directory is shipped in the same package as the installation tapes for the VS FORTRAN product. It describes all the installation materials, and gives installation instructions specific to the product release level and modification level, and to the operating system, if any beyond that supplied in this book are necessary.

CHAPTER 2. SYSTEM, MACHINE, AND STORAGE REQUIREMENTS

To install the VS FORTRAN Compiler and Library, you need the distribution tape for the VS FORTRAN program product. You will need to refer to this manual, the Program Directory for your system, and the RETAIN/370 PSP facility. You also need to meet the following system, machine, and storage requirements before beginning installation of VS FORTRAN:

SYSTEM REQUIREMENTS

- The VS FORTRAN Compiler and Library runs under the following systems:
  - MVS/SP (All releases)
  - MVS/XA (Release 1.0 and any subsequent releases)
  - VSE/Advanced Functions (Release 3.0 and any subsequent releases)
  - VM/SP (All releases)
  - VM/PC (Release 1.0 and any subsequent releases)
- The distribution tape for the VS FORTRAN program product requires one of the following:
  - Under MVS/SP and MVS/XA, the System Modification Program 4 (SMP4) or the System Modification Program Extended (SMP/E)
  - Under VSE/Advanced Functions, the Maintain System History Program (MSHP)
  - Under VM/SP, the EXEC procedures for installation provided as part of VS FORTRAN

A NOTE ABOUT VM/PC

Because the VS FORTRAN Compiler and Library cannot be directly installed on the VM/PC system, this manual does not contain information about installation under VM/PC. If you are using VM/PC, the VS FORTRAN Compiler and Library must first be installed on your host system. For more information about VM/PC, see IBM Virtual Machine/Personal Computer User's Guide.

MACHINE REQUIREMENTS

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

- Compile-time Machine Requirements:
  - Any processing unit supported by MVS/SP (with or without TSO), MVS/XA (with or without TSO-E), VSE/Advanced Functions, or VM/SP
  - I/O devices used by the compiler, normally disks
- Object-time Machine Requirements
  - Any processing unit supported by MVS/SP (with or without TSO), MVS/XA (with or without TSO-E), VSE/Advanced Functions, or VM/SP
  - I/O devices used by the object program during execution

- Supported Devices
  - Under MVS/SP, MVS/XA, and VM/SP, IBM devices supported by the BSAM, BDAM, and VSAM access methods can be used by object programs produced by the VS FORTRAN compiler when used with the VS FORTRAN library.
  - Under VM/SP, any devices supported by VSAM, or by BSAM or BDAM for MVS compatibility, are supported by VS FORTRAN.
  - Under VSE/Advanced Functions, the VS FORTRAN device-independent interface supports IBM devices transparent through the VSAM interface, the SAM interfaces for non-DASD devices, and the SAM/DAM device-independent interface provided by the Basic Access Method.

STORAGE REQUIREMENTS

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

- Under MVS/SP or MVS/XA, space for the various product libraries on your disks
- Under VSE/Advanced Functions, space in the core image (system or private), relocatable, and source statement libraries
- Under VM/SP, space on 2 target disks

For specific DASD space requirements, refer to the VS FORTRAN Program Directory. The VS FORTRAN Compiler requires 980K bytes of virtual storage to handle a typical FORTRAN source program of 100 statements. Storage requirements for the VS FORTRAN Library vary according to the customization features selected, and according to the size of user programs.

CHAPTER 3. THE INSTALLATION MACROS: VSFORTC AND VSFORTL

Two installation macros, VSFORTC and VSFORTL, are provided with VS FORTRAN. The compiler macro instruction VSFORTC specifies the compiler default options. The library macro instruction VSFORTL specifies library default options and input/output information. This section will help you plan which options to include as defaults for your installation. For information on how to code and use the macros, see the chapter for your system in the "Installation Guide," in Part 2 of this book.

VSFORTC MACRO

You must use the VSFORTC installation macro to specify the system on which VS FORTRAN will run. VSFORTC also permits you to specify the default values for compiler options.

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

If you want to find out what possible errors can occur during VSFORTC macro assembly, use the INSTERR option with the appropriate SYSTEM option to produce a list of error messages. After you have the list, be sure to remove the INSTERR option from the macro. Otherwise, you will continue to receive a return code of 16 from the assembler.

The following table shows the pairs of options that will create an error message, if both are used. The VSFORTC macro will not allow the installation to proceed if any of these conditions occurs.

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

The IBM-supplied default parameters are underlined in the following parameter lists. If a given operand is not coded in the VSFORTC installation macro, these parameters are assumed when the macro is assembled. The options that can be specified for the VSFORTC installation macro are as follows:

CHARLEN = number|500
specifies the maximum length for any character variable, character array element, or character function. Specify number as an integer from 1 to 32767. Within a program unit, you cannot specify a length for a character variable, array element, or function greater than the CHARLEN specified.

DATE = MDY|YMD
specifies the format of the date to be printed by the compiler.

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

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

FIPS = S|F|NOFIPS
specifies whether or not standard language flagging is to be performed, and, if it is, the standard language flagging level:

FIPS = S
specifies subset standard language flagging.

FIPS = F
specifies full standard language flagging.

FIPS = NOFIPS
specifies no standard language flagging.

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

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

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

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

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

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

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

INSTERR = NOLIST|LIST
specifies whether or not to list the messages that could be issued by the compiler installation macro. If LIST is chosen, all possible messages are listed and compiler installation does not occur.

LANGLVL = 66|77
specifies the language level at which the input source program is written.

LANGLVL = 66
specifies the old FORTRAN level—the 1966 language standard plus IBM extensions.

LANGLVL = 77
specifies the current FORTRAN level—the 1978 language standard plus IBM extensions.

LINECNT = number|60
specifies the maximum number of lines on each page of the printed source listing. Specify number as an integer from 5 to 32765. The advantage of using a large LINECNT number is that there are fewer page headings to look through if you are using only a terminal. Your output, if printed, will run together from page to page without a break.

NAME = name|MAIN
can only be specified when LANGLVL(66) is specified. It specifies the name that is generated on the output and the name of the CSECT generated in the object module. It only applies to main programs. When NAME is omitted, the default name (MAIN) is used.

OBJATTR = RENT|NORENT
specifies whether or not reentrant object code is to be generated by the compiler.

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

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

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

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

OPTIMIZ = 0 or NOOPTIMIZE
specifies no optimization.

OPTIMIZ = 1
specifies register and branch optimization.

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

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

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

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

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

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

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

Note: If your users are not using a SYSTERM data set, specify NOTERMINAL because there is no terminal available.

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

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

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

SXM = SXM|NOSXM
improves readability of XREF or map listing output at a terminal. SXM formats listing output for a 72-character wide terminal screen; NOSXM formats listing output for a printer.

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

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

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

Note: You must specify the system on which you are installing VS FORTRAN.

There is no system default.

TEST = TEST|NOTEST
TEST overrides any optimization level above OPTIMIZE(0), and adds execution-time overhead. Specifies whether or not to create input for VS FORTRAN Interactive Debug (5668-903) and symbol table information.

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

Note: If your users are not using a SYSTERM data set, specify NOTRMFLG because there is no terminal available.

VSFORTL MACRO

The VSFORTL macro instruction specifies input/output information for the VS FORTRAN Library. The VS FORTRAN Library object-time input/output routines require information on the number of logical input/output units that are available to the system; the UNTABLE operand provides this information.

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

The SYSTEM operand is required; all other VSFORTL keyword operands are optional. If any other operand is omitted, the default value for that operand is used.

The IBM-supplied default parameters are underlined in the following parameter lists. The VSFORTL macro instruction keyword operands and their parameters are:

ARCH = STD|XA
specifies whether you want standard (non-XA) or XA installation. You may specify ARCH=XA on a non-XA installation if Assembler H Version 2 Release 1 (5668-962) is used. You must specify XA when the library is to run on MVS/XA.

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

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

Specify unit as a 2-digit number from 01 to 99.

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

Specify unit as a 2-digit number from 01 to 99.

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

Specify unit as a 2-digit number from 01 to 99.

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

Note: You must specify the system on which you are installing VS FORTRAN.

There is no system default.

UNTABLE = number|08
specifies the largest unit number you can include in a VS FORTRAN program. Because the unit numbers begin with 0, the UNTABLE number plus 1 indicates how many units are allowed.

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

PART 2. INSTALLATION GUIDE

CHAPTER 4. INSTALLATION UNDER MVS

This chapter describes the standard installation of the VS FORTRAN Compiler and Library under MVS/SP and MVS/XA. The procedure for installing the Library only is the same as the procedure for the Library when installed with the Compiler.

For specific information on space allocations, module and macro names and other details needed to install the Compiler and Library, see the VS FORTRAN Program Directory. For information on the features that you can customize to fit your installation's needs, see Chapter 8, "Customization under MVS" on page 38.

BASIC MACHINE-READABLE MATERIAL

The distribution medium for VS FORTRAN Compiler and Library, and the VS FORTRAN Library only, is a standard-labeled 9-track tape written at either 1600 or 6250 BPI, which contains SMP modification control statements, JCLIN, modules, and macros.

See the Program Directory for the order of files and their descriptions.

ADDITIONAL STORAGE REQUIREMENTS

See the Program Directory for information on the additional track and directory block space required by VS FORTRAN when the COMPRESS(ALL) keyword is not used in the SMP ACCEPT and APPLY processing.

DATA SETS

SMP

The following SMP data sets are needed during the installation process:

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

For exact block sizes and DASD space requirements, see the Program Directory. The SMP space allocations contained there reflect the minimum requirement. If you plan to do subsequent maintenance, you will need to increase the size.

SMPTLIB

Eight SMPTLIB data sets are allocated by the RECEIVE process. The DSSPACE subentry of the PTS SYSTEM entry must be of sufficient size to accommodate a maximum SMPTLIB data set size of six cylinders and 50 directory blocks on a 3330. The SMPTLIB data sets will be used in the APPLY and ACCEPT steps described under "Installation Procedures" on page 14. They are uncataloged data sets, and are deleted if the ACCEPT is successful.

TARGET AND DISTRIBUTION LIBRARIES

The following libraries are needed for the installation process:

Compiler Target Libraries	Library Target Libraries	Compiler Distribution Libraries	Library Distribution Libraries
SYS1.FORTVS	SYS1.PPOPTION	SYS1.VSFCCM	SYS1.VSFLBM
SYS1.PPOPTION	SYS1.VALTLIB	SYS1.VSFCCS	SYS1.VSFLBS
SYS1.PROCLIB	SYS1.VFORTLIB		
SYS1.SAMPLIB	SYS1.VLNKMLIB		

Note that all the target libraries must be allocated to complete SMP install, whether you choose to use them later or not.

LIBRARY DESCRIPTIONS

FORTVS Compiler load modules

PPOPTION Compiler and library options macro input; members IFX0OPTS and IFYUATBL

PROCLIB Cataloged procedures to run VS FORTRAN jobs; for a listing of each procedure, see the VS FORTRAN Programming Guide

SAMPLIB Sample FORTRAN source program and SMP installation procedures

VALTLIB Library of alternative mathematical modules

VFORTLIB Library of modules used for linkage editing in both link mode and load mode, and for executing in load mode

VLNKMLIB Library of interface modules used in link mode only

VSFCCM Compiler object modules

VSFCCS Compiler macros and cataloged procedures

VSFLBM Library object modules

VSFLBS Library macros

INSTALLATION OVERVIEW

To install VS FORTRAN using SMP, take the following steps:

1. Unload the SMP installation jobs and procedures from the PID tape using JCL similar to the example given in "Installation Procedures" on page 14.
2. Examine the installation jobs and procedures and change them as necessary to suit your local requirements. You must supply local defaults for the VSFORTC and VSFORTL macros in the installation job.

3. Execute the INSTALL job to receive and apply VS FORTRAN. This job performs the following functions:

   a. Allocation of SMP data sets using the procedure ALLCPR

   b. Initialization of SMP data sets using the procedure INITPROC

   c. Allocation of product data sets using the procedure FORTPROC

   d. Creation of PPOPTION members for VSFORTC and VSFORTL macro defaults

   e. Invocation of SMP to accomplish the receive and apply using the procedure ACCVSF

4. Optionally, run a sample program to verify that installatic is complete.

5. Use the ACCEPT job to accept the product into the distribution libraries.

INSTALLATION PROCEDURES

The following sections give you detailed descriptions of the steps outlined in the installation overview. If a previous version of VS FORTRAN is already installed on your system, you may want to save it in an alternate library to prevent it from being overlaid by Release 4.0.

To install VS FORTRAN Compiler and Library, use the System Modification Program (SMP) Release 4 or System Modification Program Extended (SMP/E) service aid. Refer to the OS/VS Syste Modification Program (SMP): System Programmer's Guide or System Modification Program Extended (SMP/E): User's Guide for information regarding the use of SMP.

UNLOADING THE SMP INSTALLATION PROCEDURES AND JOBS

With the sample Job Control Language (JCL) shown below, you can copy the installation procedures from the PID tape to disk. Executing this JCL takes the jobs and procedures off the installation tape, catalogs them in a data set, and prints a copy.

```
//GETPROCS JOB  .... (user information)
//*
//*   GET SMP INSTALLATION PROCEDURES FROM PID TAPE
//*
//        EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=A
//IN       DD  DSN=filename,UNIT=2400-3,VOL=SER=volser,DISP=SHR,
//             LABEL=(x,SL)
//OUT      DD  DSN=FORTRAN.SMPPROCS,DISP=(NEW,PASS),SPACE=(TRK,(1,1,2)),
//             UNIT=SYSDA,VOL=SER=volid,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=4000)
//SYSUT3   DD  SPACE=(TRK,(1)),UNIT=SYSDA
//SYSUT4   DD  SPACE=(TRK,(1)),UNIT=SYSDA
//SYSIN    DD  *
 COPY INDD=IN,OUTDD=OUT
 SELECT MEMBER=(ALLCPROC,INITPROC,FORTPROC,ACCVSF,INSTALL,ACCEPT)
/*
//*
//*   PRINT THE PROCEDURES
//*
//    EXEC PGM=IEBPTPCH
//SYSPRINT DD  SYSOUT=A
//SYSUT1   DD  DSN=FORTRAN.SMPPROCS,DISP=(OLD,CATLG)
//SYSUT2   DD  SYSOUT=A
//SYSIN    DD  *
      PRINT TYPORG=PO,MAXFLDS=9
      RECORD FIELD=(80)
/*
```

Note: See the VS FORTRAN Program Directory for the specific names and profile position you need to insert in place of x, filename and volser in the example above. (Volid is your own data.)

Each of the SMP installation procedures has the format shown below. The section of each procedure required for SMP/E is commented out. If you are using SMP/E, the SMP4 portions of the procedures should be deleted or commented out, and the leading //* should be removed from the SMP/E statements.

```
//*************************************************************
//*
//* PROCEDURE TO: --------------
//*
//*************************************************************
//*               SMP4 VERSION
//*
//*
                    ...
//*
//*************************************************************
//*
//*               SMP/E VERSION
//*
//*************************************************************
//*//
//*//
                    ...
//*//
```

PREPARING TO INVOKE THE INSTALLATION PROCEDURES

Before invoking the installation procedures, you may want to change the job control statements for UNIT, VOLID, or BLKSZ in ACCVSF, or any of the data set names used in the procedures. If you do, you must make the corresponding changes in each procedure. You must examine the INSTALL job and supply the required variable information that is needed there.

After making the required changes to all the procedures, insert them into either a PROCLIB or the INSTALL job where they will be invoked.

The space allocations in the procedures are for an IBM 3330 Disk Storage and will have to be adjusted for other devices. Refer to the Program Directory for precise information on block sizes and DASD space requirements.

USING THE ALLCPROC AND INITPROC PROCEDURES

The two procedures, ALLCPROC and INITPROC, set up the SMP environment. ALLCPROC allocates the data sets required for SMP. INITPROC initializes the SMP data sets SMPCDS, SMPACDS, and SMPPTS for SMP4, and initializes the global, target, and distribution zones for SMP/E. You should execute these procedures only if you do not intend to use existing SMP data sets.

Note that, if you are using existing SMP data sets, the space and directory block allocations given will be required in addition to existing allocations. You should have a distribution library (DLIB) pack with adequate space for the VS FORTRAN Compiler and Library data sets; the space requirements are included in the JCL.

These data sets are used only for software service, and the DLIB pack will normally be online only when the system is being updated.

USING THE FORTPROC PROCEDURE

This procedure allocates the data sets that contain VS FORTRAN Compiler and Library load modules and macros. Allocations given in the procedure are for a 3330 device; you must adjust for other device types, and for any existing data sets (for example, SYS1.SAMPLIB).

SPECIFYING DEFAULTS

In the CREATE step of the INSTALL job, you must specify the default VS FORTRAN Compiler and Library options for your installation. The CREATE step executes IEBGENER to place your defaults for the macros VSFORTC and VSFORTL in members IFXOOPTS and IFYUATBL of the PPOPTION data set.

Code the VSFORTC and VSFORTL macro instructions as follows: Column 1 must be blank. VSFORTC or VSFORTL may appear anywhere before column 72 but must precede the operands by at least one blank. The operands are separated by commas and may be continued on any number of logical records as long as column 72 contains a nonblank character and the data on the following record begins in column 16.

Refer to Chapter 3, "The Installation Macros: VSFORTC and VSFORTL" on page 5, for the VSFORTC and VSFORTL options you may choose, and the IBM-supplied defaults. In both macros, you must specify the SYSTEM option.

Note that the default number of units in VSFORTL is 8 (UNTABLE option). You will probably want to specify a larger number of units. If you will not be using a SYSTERM data set and will be

compiling with VS FORTRAN in a batch environment in the VSFORTC macro, specify NOTRMFLG and NOTERMINAL to avoid messages about having no terminal "online." You only need to code the options you specifically want to control.

USING THE ACCVSF PROCEDURE

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

- SMP data sets
- FORTRAN data sets

USING THE INSTALL JOB

The INSTALL job invokes the four procedures described above to accomplish the installation. The job allocates, initializes, receives, and applies the program product.

If you are installing in a target library that has not previously contained this product, the message **IEW0342** will be generated during link-editing. A condition code of 4 may result from SMP, and a condition code of 4 or 8 from the linkage editor. **This message and the condition codes resulting from it may safely be ignored.**

VERIFYING SUCCESS

In order to verify the success of the installation process, you may want to run a sample program. Below is sample JCL to execute the compile-load-go procedure, FORTVCLG, provided in SYS1.PROCLIB, and to run the sample program, IFYSMPFT, provided in SYS1.SAMPLIB.

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

USING THE ACCEPT JOB

When you are satisfied that VS FORTRAN Release 4.0 is operating correctly, use the ACCEPT job to store the product in your system's distribution libraries (DLIBs). Installation of VS FORTRAN Release 4.0 is now complete.

If you are installing in distribution libraries that have not previously contained VS FORTRAN, you will receive the message **HMA2471** for modules IFYUATBL and IFX00PTS. **These messages can be ignored.**

The DIS(WRITE) at the end of the APPLY and ACCEPT specifies that the directory for the SMPCDS and SMPACDS data sets is to be in storage during processing. This decreases the wait time for I/O operations. A description of the DIS operand is found in the OS/VS System Modification Program (SMP): System Programmer's Guide.

CHAPTER 5. INSTALLATION UNDER VSE

This chapter describes the standard installation of the VS FORTRAN Compiler and Library under VSE/Advanced Functions. The procedure for installing the Library only is the same as the procedure for installing the Compiler and Library.

For specific information on space allocations, module and macro names and other details needed to install the Compiler and Library, see the VS FORTRAN Program Directory. For information on the features that you can customize to fit your installation's needs, see Chapter 9, "Customization Under VSE" on page 52.

BASIC MACHINE-READABLE MATERIAL

The distribution medium for the basic material of the VS FORTRAN Compiler and Library, and the VS FORTRAN Library only, is an unlabeled 9-track tape written at either 1600 or 6250 BPI.

The basic material tape is in a format suitable for installation with MSHP.

For more information on the order of files and their descriptions, see the Program Directory.

ADDITIONAL STORAGE REQUIREMENTS

See the Program Directory for details on the additional system library storage required to install the VS FORTRAN Compiler and Library.

TARGET LIBRARIES

The libraries required by the installation process for VSE are:

Relocatable

A5748F03.SYSRLB.VLNKMLIB

A5748F03.SYSRLB.VFORTLIB

Core Image

A5748F03.SYSCLB.VFLODLIB

Source Statement

A5748F03.SYSSLB.VSRCLIB

DESCRIPTION OF LIBRARIES

VLNKMLIB Library of interface modules used in the link-edit step in link mode only.

VFORTLIB Library of VS FORTRAN library processing modules needed in the link-edit step for both link mode and load mode.

VFLODLIB Library of VS FORTRAN library modules required at execution in load mode only. After installation, this library also contains the VS FORTRAN compiler. It is required at compilation time.

VSRCLIB Library of VS FORTRAN options macros and sample program.

INSTALLATION OVERVIEW

To install VS FORTRAN under VSE/Advanced Functions using MSHP, take the following steps:

1. Use the INSTALL COMPONENT to install the compiler and the library.

2. Optionally, code the VSFORTC macro instruction to change the compiler default options, and code the VSFORTL macro instruction to change the library default options. Assemble these macros, and relink-edit the compiler to put the changed defaults into effect.

INSTALLATION PROCEDURES

The following sections give detailed descriptions of the steps outlined in the installation overview. If a previous release of VS FORTRAN is already installed on your system, you may want to move it to a private relocatable library to prevent it from being overlaid by Release 4.0.

USING MSHP

You must use the Maintain System History Program (MSHP) to install the VS FORTRAN Compiler and Library. Refer to VSE/Advanced Functions Maintain System History Program (MSHP) User's Guide for information regarding the use of MSHP.

Run the MSHP function INSTALL COMPONENT. See "Installing Basic Material into Private Libraries" on page 20 or "Installing Basic Material into Work Libraries" on page 21, for a sample job stream to install the VS FORTRAN Compiler and Library.

If you are installing the VS FORTRAN Compiler and Library in private libraries using MERGE, you will have to add appropriate ASSGN, DLBL, and EXTENT statements for the private libraries.

MSHP will invoke the appropriate system utilities to catalog the affected modules in your private core image, relocatable, and source statement libraries.

INSTALLING BASIC MATERIAL INTO PRIVATE LIBRARIES

You can use the following sample job stream to install basic material into private libraries.

```
// JOB INSTALL   INSTALLATION OF 5748-F03
*        THIS JOB WILL RESTORE THE FORTRAN LIBRARIES
*        FROM THE DISTRIBUTION TAPE INTO PRIVATE LIBRARIES
*        (DEFINED BY THE 'DEFINE' MSHP STATEMENTS).
*        THE RESTORED LIBRARIES CAN BE INCLUDED IN YOUR
*        SYSTEM LIBRARIES BY USING THE 'MERGE' OPTION ON
*        THE MSHP INSTALL STATEMENT INSTEAD OF 'ATTACH'.
*
*      NOTE:    ALLOCATIONS GIVEN ARE FOR IBM 3330.
*               FOR OTHER DASD TYPES, ADJUST ACCORDINGLY.
*
// ASSGN SYS006,182                    INPUT DISTRIBUTION TAPE
// ASSGN SYS007,130,VOL=VOLSER,SHR     OUTPUT UNIT FOR PRIV CL
// ASSGN SYS008,SYS007                 OUTPUT UNIT FOR PRIV RL
// ASSGN SYS009,SYS007                 OUTPUT UNIT FOR PRIV SL
// ASSGN SYS002,SYS007                 AUX HISTORY FILE
// MTC REW,SYS006                      REWIND INPUT TAPE
// OPTION CATAL
// EXEC MSHP
INSTALL COMPONENT FROMTAPE ATTACH
  DEFINE CLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSCLB.VFLODLIB'
  DEFINE RLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSRLB.VFORTLIB'
  DEFINE SLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSSLB.VSRCLIB'
  DEFINE HISTORY AUX EXTENT = xxxx:yyy UNIT = SYS002 -
     ID = 'A5748F03.HISTORY.FILE'
RETRACE COMP ID=5748-F0-300
/*
// ASSGN SYS006,182                    INPUT DISTRIBUTION TAPE
// ASSGN SYS007,UA
// ASSGN SYS008,130,VOL=VOLSER,SHR     OUTPUT UNIT FOR PRIV RLIB
// ASSGN SYS009,UA
// ASSGN SYS002,SYS008                 AUX HISTORY FILE
/*
// DLBL IJSYSHF,'A5748F03.HISTORY.FILE'
// EXTENT SYSREC,VOLSER,1,0,xxxx,yyy
// MTC REW,SYS006                      REWIND INPUT TAPE
// MTC FSF,SYS006,5                    FORWARD TAPE TO 6TH FILE
// OPTION CATAL
// EXEC MSHP
INSTALL COMPONENT FROMTAPE ATTACH
  DEFINE RLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSRLB.VLNKMLIB'
  DEFINE HISTORY AUX EXTENT = xxxx:yyy UNIT = SYS002 -
     ID = 'A5748F03.HISTORY.FILE'
RETRACE COMP ID=5748-F0-300
/*
/&
```

Notes:

1. Core image library allocations should be doubled if you plan to relink-edit the compiler later.
2. SYS006 shows the device address of a 9-track product tape unit on 182.
3. SYS007 shows the device address of a 3330 DASD unit on 130. The 'VOLSER' should be replaced with the VOLID of the DASD to be used.

4. xxxx specifies the relative track or block of the start of the private libraries and auxiliary history file which are created on SYS007, SYS008, SYS009, and SYS002. For count-key-data devices, xxxx must be on a cylinder boundary.

5. yyy specifies the number of tracks or blocks to be allocated for the private library.

6. v specifies the number of tracks or blocks allocated for the library directory.

7. For exact space requirements (xxxx, yyy, and v), see the VS FORTRAN Program Directory.

INSTALLING BASIC MATERIAL INTO WORK LIBRARIES

You can use the following sample JCL to install basic material into work libraries, and merge them into system or previously assigned private libraries.

```
// JOB INSTALL     INSTALLATION OF 5748.F03
*        THIS JOB WILL RESTORE THE FORTRAN LIBRARIES
*        FROM THE DISTRIBUTION TAPE INTO WORK LIBRARIES
*        (DEFINED BY THE 'DEFINE' MSHP STATEMENTS).
*        THEN MERGE THE WORK LIBRARIES INTO YOUR SYSTEM,
*        OR PRIVATE LIBRARIES (IF ASSIGNED).
*
*      NOTE:    ALLOCATIONS GIVEN ARE FOR IBM 3330.
*               FOR OTHER DASD TYPES, ADJUST ACCORDINGLY.
*
// ASSGN SYS006,182                      INPUT DISTRIBUTION TAPE
// ASSGN SYS007,130,VOL=VOLSER,SHR       WORK UNIT FOR PRIV CL
// ASSGN SYS008,SYS007                   WORK UNIT FOR PRIV RL
// ASSGN SYS009,SYS007                   WORK UNIT FOR PRIV SL
// ASSGN SYS002,SYS007                   AUX HISTORY FILE
// MTC REW,SYS006                        REWIND INPUT TAPE
// OPTION CATAL
// EXEC MSHP
INSTALL COMPONENT FROMTAPE MERGE
  DEFINE CLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSCLB.VFLODLIB'
  DEFINE RLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSRLB.VFORTLIB'
  DEFINE SLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSSLB.VSRCLIB'
  DEFINE HISTORY AUX EXTENT = xxxx:yyy UNIT = SYS002 -
     ID = 'A5748F03.HISTORY.FILE'
RETRACE COMP ID=5748-F0-300
/*
// ASSGN SYS006,182                      INPUT DISTRIBUTION TAPE
// ASSGN SYS007,UA
// ASSGN SYS008,130,VOL=VOLSER,SHR       OUTPUT UNIT FOR PRIV RLIB
// ASSGN SYS009,UA
// ASSGN SYS002,SYS008                   AUX HISTORY FILE
/*
// DLBL IJSYSHF,'A5748F03.HISTORY.FILE'
// EXTENT SYSREC,VOLSER,1,0,xxxx,yyy
// MTC REW,SYS006                        REWIND INPUT TAPE
// MTC FSF,SYS006,5                      FORWARD TAPE TO 6TH FILE
// OPTION CATAL
// EXEC MSHP
INSTALL COMPONENT FROMTAPE MERGE
  DEFINE RLIB PRIV EXTENT = xxxx:yyy DIR = v -
     ID = 'A5748F03.SYSRLB.VLNKMLIB'
  DEFINE HISTORY AUX EXTENT = xxxx:yyy UNIT = SYS002 -
     ID = 'A5748F03.HISTORY.FILE'
RETRACE COMP ID=5748-F0-300
/*
/&
```

Notes:

1. Core image library allocations should be doubled if you plan to relink-edit the compiler later.
2. SYS006 shows the device address of a 9-track product tape unit on 182.
3. SYS007 shows the device address of a 3330 DASD unit on 130. The 'VOLSER' should be replaced with the VOLID of the DASD to be used.
4. xxxx specifies the relative track or block of the start of the private libraries and auxiliary history file which are created on SYS007, SYS008, SYS009, and SYS002. For count-key-data devices, xxxx must be on a cylinder boundary.
5. yyy specifies the number of tracks or blocks to be allocated for the private library.
6. v specifies the number of tracks or blocks allocated for the library directory.
7. For exact space requirements (xxxx, yyy, v), see the VS FORTRAN Program Directory.

MAKING MODIFICATIONS

If your installation requirements are satisfied by the IBM-supplied compiler defaults in IFX00PTS and unit assignment table values in IFYUATBL, then this completes the installation of the VS FORTRAN Compiler and Library.

If you want to change either or both of the IFX00PTS and IFYUATBL modules, then proceed to "Compiler and Library Defaults" on page 53 for instructions, and run the sample program after completing your modifications.

VERIFYING SUCCESS

A sample VS FORTRAN program (IFYSMPFT) is available in the source statement library to verify the success of the compiler installation. You may, optionally, now run the sample program to verify that VS FORTRAN has been installed correctly. To do this, code and execute the following job:

```
// JOB SAMPLEPG   THIS JOB WILL COMPILE AND EXEC THE SAMPLE PGM
// OPTION LINK,PARTDUMP
// EXEC VFORTRAN,SIZE=AUTO,PARM='LIST,XREF,MAP,FIXED'
     INCLUDE (IFYSMPFT)
/*
// EXEC LNKEDT
/*
// EXEC  ,SIZE=AUTO
/*
/&
```

CHAPTER 6. INSTALLATION UNDER VM

This chapter describes the standard installation of the VS FORTRAN Compiler and Library under VM/SP. If you have ordered the Library only, follow the installation procedures as described for the Compiler and Library, but use the library installation exec as noted in "Basic Machine-Readable Material," below.

For specific information on space allocations, module and macro names and other details needed to install the Compiler and Library, see the VS FORTRAN Program Directory. For information on the features that you can customize to fit your installation's needs, see Chapter 10, "Customization Under VM" on page 61.

BASIC MACHINE-READABLE MATERIAL

The distribution medium for the VS FORTRAN Compiler and Library, and the VS FORTRAN Library only, is an unlabeled 9-track tape written at either 1600 or 6250 BPI in EBCDIC in CMS tape dump format. It is intended to be used under the Conversational Monitor System (CMS) component of the IBM Virtual Machine Facility/370 (VM/370).

See the Program Directory for the order of files and their descriptions when installing either the library only, or the compiler and library.

If you are installing the library only, follow the instructions for installing the Compiler and Library, but use EXEC I5748LM3.

ADDITIONAL STORAGE REQUIREMENTS

See the VS FORTRAN Program Directory for details on the additional system library storage required to install the VS FORTRAN Compiler and Library.

TARGET LIBRARIES

The libraries required for the installation process for VM are:

Text Libraries

VALTLIB

VLNKMLIB

VFORTLIB

Load Libraries

VFLODLIB

Macro Libraries

VFMACLIB

LIBRARY DESCRIPTIONS

VALTLIB Library of alternative mathematical routine modules.

VLNKMLIB Library of interface modules used in link mode only. This library must be concatenated ahead of VFORTLIB for the creation of an executable program.

VFORTLIB Library of VS FORTRAN library processing modules needed for the creation of an executable program in both link mode and load mode.

VFLODLIB Library of VS FORTRAN library modules required at execution in load mode only.

VFMACLIB Library of VS FORTRAN library customization macros.

INSTALLATION OVERVIEW

To install VS FORTRAN under VM, take the following steps:

1. Log on to VM. In write status, link to the disk that will hold the files needed for service applications, and access it as your A-disk. This disk must not be accessed during execution of a VS FORTRAN program.

2. Link to a second disk in write status. This disk will be the product disk to hold executable modules and txtlibs. You will be prompted during the installation process for the access mode of this disk.

3. Mount the distribution tape on virtual address 181.

4. Load the first file onto the work disk. (This file contains the installation EXEC procedure.)

5. Execute EXEC I5748F03, the installation EXEC, which begins product installation.

6. Respond to the prompts from the installation EXEC. As you answer the prompts, the EXEC performs the installation.

For complete details on the installation process under VM, see the VS FORTRAN Program Directory and the following section, "Installation Procedures."

INSTALLATION PROCEDURES

The following sections give detailed descriptions of the steps outlined in the installation overview. If a previous version of VS FORTRAN is already installed on your system, you may want to save it somewhere other than your product disks to prevent it from being overlaid by Release 4.0.

Load the install EXEC for the VS FORTRAN Compiler and Library using the CMS TAPE LOAD commands from File 1 of the distributior tape. Complete the installation by executing the installation EXEC.

PREPARING TO EXECUTE THE EXEC

Before executing the EXEC, complete the following steps:

1. Determine where you will store the components required for service application to this product. The disk you choose must be your A-disk during the installation process. It must not be the disk containing the installed executable product.

   If you do not have enough free space, create a temporary work disk and access it as your A-disk. This disk will be

used as a work disk during installation, and must contain the equivalent of 25 cylinders on a 3330 disk drive.

If you use a temporary disk, unload its contents to tape after the install to retain the files needed for service installation.

2. Determine where you will install the executable product. This may be the system disk or a minidisk containing 10 cylinders of space on a 3330 disk drive, or the equivalent.

3. During installation, you will be prompted for information. To be prepared for the prompts, consider the following:

   a. You will be asked to accept the CMS default DCB characteristics:

      RECFM F, LRECL 80, BLOCK 80, BUFNO 1,

      or to choose OS/VS characteristics:

      RECFM U, LRECL 800, BLOCK 32756, BUFNO 2.

   b. The default number of units in the FORTRAN unit assignment table is 8. If you want to change the number of units in the FORTRAN unit assignment table, or the FORTRAN unit numbers that will be used as defaults for READ, PUNCH, and WRITE statements, respond EDIT to the prompt about editing the VSFORTL macro.

      When coding the macro, column 1 must be blank. VSFORTL may appear anywhere before column 72 but must precede the operands by at least one blank. The operands are separated by commas and may be continued on any number of cards as long as column 72 contains a nonblank character and the data on the following card begins in column 16. You do not need to code all keyword parameters. Code only those whose default you wish to change. Chapter 3, "The Installation Macros: VSFORTC and VSFORTL" on page 5, describes the options you may choose, and the IBM-supplied defaults.

   c. Determine whether or not the alternative mathematical routines are to be installed. If your response is YES, they will be loaded into an alternate TXTLIB containing only those routines. Refer to VS FORTRAN Programming Guide for more information on the use of these routines.

   d. If you are satisfied with the IBM-supplied options to the compiler, reply NOEDIT to the prompt about editing the VSFORTC macro. If you are compiling with VS FORTRAN in a batch environment and will not be using a SYSTERM data set, specify the options NOTRMFLG and NOTERMINAL to avoid messages about having no terminal online.

   e. Determine whether or not you want the compiler to be installed as a discontiguous shared segment. If so, be sure that space has been reserved on a CP-owned DASD volume for the shared segment, and that the segment name has been placed in the VM/SP system table.

      Also, determine the size of virtual storage you need to include the compiler shared segment during the install. See "The Compiler as a Discontiguous Shared Segment" on page 61 for more detail.

Note: To execute the compiler as a module, you need a 2-megabyte virtual machine size.

BEGINNING THE INSTALLATION

Log on to VM/SP and mount the distribution tape on virtual address 181.

If necessary, define a size of virtual storage large enough to include a compiler shared segment. For example, if you want the VS FORTRAN compiler to be located from 2 megabytes to 3 megabytes, then your virtual machine size should be at least 3 megabytes. In this example, you would issue the command:

```
DEFINE STORAGE 3M
```

(You must be a Class E user to initialize the DCSS.)

LINKING TO THE DISK

Link, in write status, to the system disks or minidisks that will hold the product. Remember, you may not choose the A-disk because the A-disk is used as a work disk.

LOADING THE EXEC

Load the first tape file containing the I5748F03 EXEC onto the work disk by giving the command:

```
TAPE LOAD * * A
```

EXECUTING THE EXEC

Execute the I5748F03 EXEC. This will load VS FORTRAN modules onto the A-disk and begin installation of the product.

Respond to the prompts from the installation EXEC which ask you to:

- Provide the file mode of the product disk, for example: C.
- Verify that the distribution tape is mounted on device 181.
- Choose the CMS file characteristics or VS FORTRAN-supplied characteristics found in the unit assignment table.
- Edit the library options or accept defaults.
- Change the default VFORTLIB name or accept the default.
- Change the default VLNKMLIB name or accept the default.
- Change the default VFLODLIB name or accept the default.
- Choose whether or not to install alternate mathematical library subroutines.
- Edit the compiler options or accept the defaults.
- Choose whether or not to install the compiler in a DCSS, and if so, provide the hexadecimal starting address of the DCSS. For example, a 2-megabyte shared segment starting address converts to 200000. Enter this hexadecimal number in response to the prompt.
- Compile and execute the sample program if you have not installed a shared segment.

You can halt execution of the installation EXEC by responding to any request with QUIT. If you do this, you must start the EXEC again from the beginning.

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

'VS FORTRAN COMPILER & LIBRARY INSTALLATION IS COMPLETE'

VERIFYING SUCCESS

The EXEC will allow you to run a sample program, IFYSMPFT, to verify the success of your installation. That will complete your use of the installation EXEC.

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

```
CP DEFINE STORAGE 2M
CP IPL CMS
ACCESS 333 A
FORTVS IFYSMPFT
GLOBAL TXTLIB VFORTLIB CMSLIB
GLOBAL LOADLIB VFLODLIB
LOAD IFYSMPFT (NOAUTO START
```

where 333 A is the disk where the sample program, IFYSMPFT, is loaded.

SHARED SYSTEM INSTALLATION

If you want to install the library module IFYVRENC as a shared system, see "IFYVRENC as a Discontiguous Shared Segment" on page 70 for more information. IFYVRENC and its copies must be installed separately because they may contain a variable number of CSECTs. CSECT IFYCRNAM must be built separately, to accommodate user installation requirements.

PART 3. CUSTOMIZATION GUIDE

CHAPTER 7. CUSTOMIZATION UNDER ALL SYSTEMS

The VS FORTRAN extended error-handling facility, which can be customized regardless of which operating system you are using, is discussed in this chapter. Other customization features are discussed in separate chapters for each operating system.

EXTENDED ERROR-HANDLING FACILITY

By allowing you to modify information in an area of main storage called the **error option table**, the extended error-handling facility gives you considerable control over errors that occur during execution. The error option table specifies the action that will take place when an error occurs. A permanent copy of the error option table is maintained as a VS FORTRAN library module.

When an error is detected by the VS FORTRAN Library, you can:

- Continue execution after the error with standard VS FORTRAN corrective action,
- Or, optionally, specify your own corrective action.

When an error occurs, a brief message is printed, along with an error identification number. The data in error (or some other associated information) is printed as part of the message text. A summary error count, printed when a job is completed, informs you how many times each error occurred. (A complete listing of library messages can be found in VS FORTRAN Language and Library Reference.)

For each error condition detected, you have both dynamic and default control over:

- The number of times the error is allowed to occur before your 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

MODIFYING THE ACTION TAKEN BY THE ERROR MONITOR

When an error is detected by a VS FORTRAN library routine, the VS FORTRAN error monitor receives control. The error monitor prints the necessary diagnostic and informative messages and then takes **one** of the following actions:

- Terminates the job.
- Returns control to the library routine, which takes a standard corrective action and then continues execution.
- Calls a subroutine that you provide to resolve the error situation, and then returns to the library routine. The routine then continues execution.

The actions of the error monitor are controlled by settings in the error option table. You can, if you choose, modify the action taken by the error monitor by modifying the error option table. IBM provides a standard set of 182 entries in the table. The default error option table supplied by IBM is shown in Chapter 10 of VS FORTRAN Language and Library Reference.

You can modify the action taken by the error monitor at either of two times:

- Following installation, during customization

  You can customize the error option table by providing additional entries, or by modifying the IBM-supplied defaults for the standard entries. These modifications change the copy of the error option table that is permanently maintained as a VS FORTRAN library module.

- At execution

  If an error option table entry is specified as "modifiable," you can update an error option table entry dynamically at execution time. You can specify a user exit address, or change several other options in the error option table entry. These changes are temporary, and apply only to the program you are executing. To make the changes dynamically, your program must call the error-handling subroutines supplied with VS FORTRAN. These subroutines are described under "Specifying Dynamic Control" on page 37.

CHANGING ERROR OPTION TABLE ENTRIES DURING CUSTOMIZATION

Before beginning to customize the error option table entries, you must do some planning. For example, you must assign error numbers to the error conditions you want detected, and you must plan the error option table entries for these conditions. Remember that these modifications will affect everyone at your installation who uses VS FORTRAN.

Planning for the Modifications

When planning your modifications, consider the following:

- You must plan the error condition numbers for your installation's programs. IBM-designated error conditions have reserved error codes from 120 to 301. The error codes you assign for installation-designated error conditions must be in the range 302 to 899. VS FORTRAN uses the error code to find the proper entry in the error option table.
- You must know the number of error conditions for your installation, so that appropriate entries will be provided in the error option table.
- The routine that uses the VS FORTRAN error monitor for error service should have a general-purpose function.

An Overview

The error option table is supplied by IBM, and installed as a module in the VS FORTRAN library. Using the VSFUOPT macro, you can:

- Add new error message numbers to the error option table.

  You can then write your own message text and call ERRMON in your program to write the message.

- Change the default values in the error option table.

  You can change the default values for IBM-supplied messages, or for new message numbers you have previously added.

The defaults for IBM-supplied error messages are documented in Chapter 10 of VS FORTRAN Language and Library Reference.

You must complete the following steps to customize the error option table. More detail is provided for each step in the sections that follow.

1. Code the VSFUOPT macro instructions.
2. Using the VSFUOPT macro instructions you have written in Step 1, assemble the module IFYUOPT.
3. Replace the object module IFYUOPT in VFORTLIB.
4. Rebuild the composite module IFYVLBCM.

Step 1: Coding the VSFUOPT Macro Instructions

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

CODING THE REQUIRED VSFUOPT MACRO INSTRUCTION: The first macro instruction has the following syntax:

```
VSFUOPT [ADDNTRY=n] [,ARCH=STD|XA]
```

where

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

ARCH
specifies whether you want a standard (non-MVS/XA) or MVS/XA error option table.

STD
specifies that you want a standard error option table. STD is the default.

XA
specifies that you want an MVS/XA error option table. You can specify XA in any operating system environment if Assembler H, Version 2, Release 1, is used to assemble the error option table. However, you must specify XA to execute under MVS/XA.

If you want to add additional error messages without modifying any existing entries in the error option table, follow your VSFUOPT instruction with an END statement, and go on to "Step 2: Assembling the Module IFYUOPT" on page 36.

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

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

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

where

MSGNO
specifies which error message numbers will be affected by the default changes.

ermsno
specifies the first error message number in a series of consecutive numbers.

qty
specifies the number of consecutive error message numbers, beginning with **ermsno**. If the defaults for only one error message number are to be changed, then **qty**, the preceding comma, and the surrounding parentheses may be omitted.

For example, if the parameter were coded MSGNO=(153,4), then the defaults for four error messages, beginning with number 153, will be changed as specified by the remaining parameters. Thus, the defaults for messages 153 through 156 will be changed.

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

errs
specifies the number of errors allowed. To specify an exact number of errors allowed, errs must be a positive integer with a maximum of 255. A zero, or any number greater than 255, means the error can occur an unlimited number of times.

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

If the message number is an IBM-supplied message number, the default value for this parameter is listed in the table in Chapter 10 of VS FORTRAN Language and Library Reference. If the message number has been added by your installation, the default value is 10.

PRINT
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 printed. To specify an exact number of times printed, prmsg must be a positive integer, with a maximum of 254. A zero means the message will not be printed. Specifying 255 means the message can be printed an unlimited number of times.

If the message number is an IBM-supplied message number, the default value for this parameter is listed in the table in Chapter 10 of VS FORTRAN Language and Library Reference. If the message number has been added by your installation, the default value is 5.

MODENT
specifies whether or not the ERRSET subroutine may be used to modify the error option table entry for this message.

YES
specifies that the entry may be modified.

NO
specifies that the entry may not be modified.

If you code a YES value for an IBM-supplied error message whose default is NO, and you subsequently modify this entry using the ERRSET subroutine, you may receive undesirable results. Check the table in Chapter 10 of VS FORTRAN Language and Library Reference to find out which message numbers have a "Modifiable Entry" value of NO.

See "Default Values for the Optional Macro Instruction Parameters" on page 35 for the default for this parameter.

PRTBUF
specifies whether or not the I/O buffer is to be printed following certain I/O errors.

YES
specifies that the contents of the buffer are to be printed.

NO
specifies that the contents of the buffer are not to be printed.

This option applies only to IBM-supplied error messages. Do not code YES unless the IBM-supplied default for this error message number already allows the buffer to be printed. Check the table in Chapter 10 of VS FORTRAN Language and Library Reference to find out which message numbers have a "Print Buffer" value of YES.

See "Default Values for the Optional Macro Instruction Parameters" on page 35 for the default for this parameter.

INFOMSG
specifies whether the message is an informational or an error message.

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

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

NO
specifies that the message is an error message.

See "Default Values for the Optional Macro Instruction Parameters" on page 35 for the default for this parameter.

TRACBAK
specifies whether or not a module traceback listing is to be printed following the error message.

YES
specifies that the traceback listing is to be printed.

NO
specifies that the traceback listing is not to be printed.

See "Default Values for the Optional Macro Instruction Parameters" for the default for this parameter.

USREXIT
specifies the user error exit routine that will be invoked following the printing of the error message.

exitname
specifies the entry point name of the user error exit routine. If the routine is specified here, instead of being specified as a parameter passed to the ERRSET subroutine, the routine will be 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, 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 IFYVLBCM.

If the user error exit routine must communicate with the VS FORTRAN program in which the error was detected, it must do so using a dynamic common area, not a static one.

There is no default value for this parameter.

DEFAULT VALUES FOR THE OPTIONAL MACRO INSTRUCTION PARAMETERS:
The default values for four parameters on the optional VSFUOPT macro instruction vary according to two conditions. These conditions and the default values are as follows:

1. The message number is an IBM-supplied message number, and **none** of the default values for MODENT, PRTBUF, INFOMSG, TRACBAK are being changed.

   For this condition, the default values are those found in the table in Chapter 10 of VS FORTRAN Language and Library Reference.

2. The message number is an IBM-supplied message number, and the default values for one or more of the following are being changed: MODENT, PRTBUF, INFOMSG, or TRACBAK.

   OR

   The message number has been added by your installation.

   For this condition, the default values for the unspecified parameters are as follows:

Parameter	Default
MODENT	YES
PRTBUF	NO
INFOMSG	NO
TRACBAK	YES

Step 2: Assembling the Module IFYUOPT

Using the VSFUOPT macro instructions you have written in Step 1, assemble the module IFYUOPT. To assemble the module, make the VS FORTRAN library that contains the VSFUOPT macro definition available to the assembler. If you are running under:

- MVS: You will find the VSFUOPT macro in SYS1.VSFLBS.

  Make this library available to the assembler using a SYSLIB DD statement.

- VSE: You will find the VSFUOPT macro in A5748F03.SYSSLB.VSRCLIB.

  Make this library available to the assembler using a LIBDEF command.

- VM: You will find the VSFUOPT macro in VFMACLIB MACLIB.

  Make this library available to the assembler using a GLOBAL MACLIB command.

Step 3: Replacing the Object Module IFYUOPT

Replace the existing module IFYUOPT in VFORTLIB with the new module IFYUOPT, assembled in Step 2. If you are running under:

- MVS: You must link-edit the module into VFORTLIB.

- VSE: You must catalog the module using the CATALR control statement of the MAINT program.

- VM: You must delete the existing module in VFORTLIB TXTLIB, and then add the new IFYUOPT using the CMS TXTLIB command.

Step 4: Rebuilding the Composite Module IFYVLBCM.

Because the module IFYUOPT is a required module in the composite module IFYVLBCM, your new IFYUOPT module must be added to IFYVLBCM. Any user exit error routines specified in USREXIT parameters must also be included in IFYVLBCM. For more detail on rebuilding IFYVLBCM, follow the instructions for "Building the Composite Modules" in the chapter on customization under your system.

CALLING ERRMON TO EXECUTE YOUR OWN ERROR HANDLING

If you have added your own error messages, any program can call the VS FORTRAN error monitor (ERRMON) routine to write them out. ERRMON examines the error option table for the appropriate error number and its associated entry, and takes the actions specified.

For information on how to use ERRMON, see VS FORTRAN Programming Guide.

CHANGING ERROR OPTION TABLE ENTRIES DYNAMICALLY

As a VS FORTRAN programmer, you can modify entries in the error option table at execution time for a specific program.

Planning for Your Own Messages

If you plan to print any of your own messages in the range 302 through 899, the installer must have assigned the error numbers for your use. IBM-designated error conditions have reserved error codes from 120 through 301.

Specifying Dynamic Control

You can specify dynamic error control during your program's execution by calling the following subroutines supplied with VS FORTRAN:

- ERRSAV—to obtain a copy of an error option table entry
- ERRSTR—to store an entry in the error option table
- ERRSET—to change parameters in the error option table
- ERRTRA—to request a trace of program execution
- ERRMON—to write out your own error message

For reference information on the extended error-handling subroutines, and information describing actions you can take to correct an error, see VS FORTRAN Language and Library Reference.

For example, a common application of dynamic error handling is to specify a user exit address in the error option table. When the table entry defined as the user exit address contains an address, the user exit is taken; otherwise, only the standard corrective action is taken. If you want to specify that no corrective action, either standard or your own, is to be taken, you must indicate in the table entry that only one error is to be allowed before termination of execution.

You can dynamically modify other error option table entries besides the user exit routine. For a complete list and description of these options, see the discussion of the ERRSET subroutine in VS FORTRAN Language and Library Reference.

CHAPTER 8. CUSTOMIZATION UNDER MVS

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

- Alternative mathematical library subroutines
- Cataloged procedures
- The separation tool
- Reentrant I/O library modules
- Execution-time loading of library modules

ALTERNATIVE MATHEMATICAL LIBRARY SUBROUTINES

You may choose to replace the supplied standard VS FORTRAN routines with the alternative mathematical library subroutines, or you may insert the alternative routines into a local user library so they can be made available when needed by the individual user.

The alternative mathematical library subroutines are link-edited in SYS1.VALTLIB by the installation process.

To make the alternative mathematical routines available to all users, you should change the cataloged procedures FORTVCL, FORTVCLG, etc., provided in SYS1.PROCLIB to concatenate SYS1.VALTLIB ahead of SYS1.VFORTLIB in the link-edit step SYSLIB DD statement. For example, use these statements in load mode:

```
//SYSLIB DD DSN=SYS1.VALTLIB,...
//       DD DSN=SYS1.VFORTLIB,...
```

Or use these statements in link mode:

```
//SYSLIB DD DSN=SYS1.VALTLIB,...
//       DD DSN=SYS1.VLNKMLIB,...
//       DD DSN=SYS1.VFORTLIB,...
```

CATALOGED PROCEDURES

Cataloged procedures are placed in the procedure library, SYS1.PROCLIB. You may want to edit the supplied procedures to fit your system's requirements. For additional information on writing and processing cataloged procedures under MVS, see VS FORTRAN Programming Guide.

A typical example of a cataloged procedure is FORTVCLG, which compiles, link-edits, and executes a VS FORTRAN program in load mode. To help you understand the various functions and statements, Figure 1 is a listing of this procedure for your reference during the discussion below.

```
//FORTVCLG PROC FVPGM=FORTVS,FVREGN=1200K,FVPDECK=NODECK,
//              FVPOLST=NOLIST,FVPOPT=0,FVTERM='SYSOUT=A',GOREGN=100K,
//              FVLNSPC='3200,(25,6)',
//              GOF5DD='DDNAME=SYSIN',GOF6DD='SYSOUT=A',
//              GOF7DD='SYSOUT=B'
//*
//*             PARAMETER  DEFAULT-VALUE     USAGE
//*
//*               FVPGM    FORTVS            COMPILER NAME
//*               FVREGN   1200K             FORT-STEP REGION
//*               FVPDECK  NODECK            COMPILER DECK OPTION
//*               FVPOLST  NOLIST            COMPILER LIST OPTION
//*               FVPOPT   0                 COMPILER OPTIMIZATION
//*               FVTERM   SYSOUT=A          FORT.SYSTERM OPERAND
//*               FVLNSPC  3200,(25,6)       FORT.SYSLIN SPACE
//*               GOREGN   100K              GO-STEP REGION
//*               GOF5DD   DDNAME=SYSIN      GO.FT05F001 OPERAND
//*               GOF6DD   SYSOUT=A          GO.FT06F001 OPERAND
//*               GOF7DD   SYSOUT=B          GO.FT07F001 OPERAND
//*
//FORT    EXEC  PGM=&FVPGM,REGION=&FVREGN,COND=(4,LT),
//              PARM='&FVPDECK,&FVPOLST,OPT(&FVPOPT)'
//SYSPRINT      DD SYSOUT=A,DCB=BLKSIZE=3429
//SYSTERM       DD &FVTERM
//SYSPUNCH      DD SYSOUT=B,DCB=BLKSIZE=3440
//SYSLIN        DD DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSDA,
//              SPACE=(&FVLNSPC),DCB=BLKSIZE=3200
//LKED    EXEC  PGM=IEWL,REGION=200K,COND=(4,LT),
//              PARM='LET,LIST,MAP,XREF'
//SYSPRINT      DD SYSOUT=A
//SYSLIB        DD DSN=SYS1.VFORTLIB,DISP=SHR
//SYSUT1        DD UNIT=SYSDA,SPACE=(1024,(200,20))
//SYSLMOD       DD DSN=&&GOSET(MAIN),DISP=(,PASS),UNIT=SYSDA,
//              SPACE=(TRK,(10,10,1),RLSE)
//SYSLIN        DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//              DD DDNAME=SYSIN
//GO      EXEC  PGM=*.LKED.SYSLMOD,REGION=&GOREGN,COND=(4,LT)
//FT05F001      DD &GOF5DD
//FT06F001      DD &GOF6DD
//FT07F001      DD &GOF7DD
//STEPLIB       DD DSN=SYS1.VFORTLIB,DISP=SHR
```

Figure 1. An example of a cataloged procedure (FORTVCLG)

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

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

The statements shown in Figure 1 have specific functions when compiling, link-editing, executing, and loading.

COMPILING

When compiling, the statements have the following functions:

The EXEC statement for the compilation step named FORT specifies the compiler as the program to be executed. It does this through the PGM parameter (PGM=FORTVS).

The DD statements describe data sets required by the compiler.

SYSLIN describes the output of the compilation step, an object module stored as a temporary data set named &&LOADSET.

The DISP parameter is coded (MOD,PASS). MOD permits more than one object module to be stored if many source modules are submitted for compilation, and PASS permits the data set to be used in later job steps.

You must specify the source module data set in a SYSIN DD statement as follows:

```
//FORT.SYSIN DD {*|data set name}
```

LINK-EDITING

When link-editing, the statements have the following functions:

The EXEC statement for the link-edit step named LKED specifies the linkage editor as the program to be executed (PGM=IEWL).

In FORTVCL and FORTVCLG, the EXEC statement COND parameter indicates that the program is to be executed only if the FORT step has returned a code less than or equal to 4.

The DD statements describe required data sets.

SYSLMOD describes the output of the link-edit step. This output is a load module named MAIN, which is stored as a member of a temporary library named &&GOSET.

SYSLIN is the input to the linkage editor.

SYSLIB describes the library module data files.

When the linkage editor is the first step to be executed, as in FORTVLG, SYSLIN points to the object module defined by a SYSIN DD statement. You must supply the following statement:

```
//LKED.SYSIN DD {*|data set name}
```

SYSLIB points to the location of the FORTRAN library routines. In link mode, use these statements:

```
//SYSLIB DD DSN=SYS1.VLNKMLIB,DISP=SHR
//       DD DSN=SYS1.VFORTLIB,DISP=SHR
```

In load mode, use this statement:

```
//SYSLIB DD DSN=SYS1.VFORTLIB,DISP=SHR
```

If you want to make the alternative mathematical library routines available, see "Alternative Mathematical Library Subroutines" on page 38. For information on choosing link mode or load mode, see "Selection of Load Mode or Link Mode" on page 44.

EXECUTING

When executing, the statements have the following functions:

The EXEC statement for the go step named GO specifies that the load module created in the link-edit step is the program to be executed (PGM=*.LKED.SYSLMOD).

The parameter COND indicates that the program is to be executed only if previous steps returned a code less than or equal to 4.

The DD statements describe required data sets. DD statement FT05F001 indicates that the input data set is to be defined by a SYSIN DD statement, which you must supply. FT06F001 defines a printer data set; FT07F001 defines a card punch data set.

You must specify input to a program using a SYSIN DD statement as follows:

```
//GO.SYSIN DD {*|data set name}
```

For load mode, you need a STEPLIB statement that provides the execution-time modules needed for proper execution of the program. The statement looks like this:

```
//STEPLIB DD DSN=SYS1.VFORTLIB,DISP=SHR
```

For information on choosing link mode or load mode, see "Selection of Load Mode or Link Mode" on page 44.

LOADING

Although the example supplied above does not have a loader statement, some cataloged procedures, such as FORTVCG, do.

When loading, the statements have the following functions:

The EXEC statement for the loader step is named GO, and specifies the loader as the program to be executed (PGM=LOADER).

The DD statements describe required data sets.

SYSLOUT describes printed output, such as a module map.

The other data sets are the same as those used by the linkage editor and the load module. Note that a SYSLMOD DD statement is not specified; the loader places the load module directly into storage for execution.

When the loader is the first step to be executed, as in FORTVL, you must define the object module in the following SYSLIN DD statement (not SYSIN):

```
//GO.SYSLIN  DD {*|data set name}
```

You must define input to the load module in a SYSIN DD statement coded as follows:

```
//GO.SYSIN  DD {*|data set name}
```

SYSLIB points to the location of the FORTRAN library routines. In link mode, use these statements:

```
//SYSLIB DD DSN=SYS1.VLNKMLIB,DISP=SHR
//       DD DSN=SYS1.VFORTLIB,DISP=SHR
```

In load mode, use these statements:

```
//SYSLIB DD DSN=SYS1.VFORTLIB,DISP=SHR
//STEPLIB DD DSN=SYS1.VFORTLIB,DISP=SHR
```

If you want to make the alternative mathematical library routines available, see "Alternative Mathematical Library Subroutines" on page 38. For information on choosing link mode or load mode, see "Selection of Load Mode or Link Mode" on page 44.

THE SEPARATION TOOL

The VS FORTRAN separation tool separates the compiler's object output from a compile with the RENT option into reentrant and nonreentrant portions. This allows you to build shared modules with the reentrant portions.

The separation tool uses the object output (object deck or text file) from the compiler as input. It generates a listing file indicating the activity that took place, and also generates two object output files. The first file (ddname=SYSUT1) contains the nonreentrant text files. The second file (ddname=SYSUT2) contains the reentrant CSECTs and the table generated by the separation tool to help locate these CSECTs.

The separation tool is distributed with the VS FORTRAN Library in the form of two load modules, IFYVSFST and IFYVSFIO. Some of the ways to reinstall and execute the separation tool are described below. Note that DD statements or allocate statements are not given here but are available in VS FORTRAN Programming Guide or in PROCLIB.

- One way to use the separation tool is to access it from SYS1.VFORTLIB.

```
//A    EXEC  PGM=IFYVSFST,PARM='RENTPART'
//STEPLIB DD DSN=SYS1.VFORTLIB,DISP=SHR
```

- Another way is to link-edit the modules into one load module and insert this module into SYS1.LINKLIB. Note that the separation tool is not intended to run as an authorized program; it is intended only for use in program state.

  The link-edit control statements you need are:

```
INCLUDE VFORTLIB(IFYVSFST)
INCLUDE VFORTLIB(IFYVSFIO)
ENTRY IFYVSFST
NAME IFYVSFST(R)
```

- Another way of accessing the separation tool is to put both modules into the link pack area (SYS1.LPALIB). Both modules are reentrant and reusable.

- In the MVS/XA environment, IFYVSFST may be put into the extended link pack area, and IFYVSFIO may be put into the link pack area.

As long as the modules of the separation tool are in SYS1.VFORTLIB, maintenance done with SMP or SMP/E is available immediately. If you choose to insert the separation tool in some other library or in the link pack area, then you must be sure the updated separation tool is also inserted in the appropriate place.

REENTRANT I/O LIBRARY MODULES—TRANSITIONAL SUPPORT

The previously supported facility for loading the reentrant I/O library modules has been replaced in Release 4.0 by more extensive loading of library modules during execution. In Release 4.0, an IFYVRENT module is installed in SYS1.VFORTLIB, and contains a version of these previous modules compatible with load modules created prior to Release 4.0. However, these modules contain no new Release 4.0 functions. After your load module contains any code compiled with Release 4.0 or any Release 4.0 library modules, then all library modules linked

into that load module must be at the Release 4.0 level. The former IFYVRENT mechanism will then no longer be used for that load module.

Load modules created with Release 1.0 or Release 1.1 are not compatible with the module IFYVRENT from Release 2.0 or later. If you have such load modules, they must be relink-edited using the Release 4.0 library.

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

- In the procedures that compile, link, and execute VS FORTRAN programs, add a STEPLIB DD statement for SYS1.VFORTLIB to the GO step for loading IFYVRENT. This is a change from the former requirement for referring to SYS1.VRENTLIB.
- You may choose to put the module IFYVRENT in the pageable link pack area by moving it to SYS1.LPALIB. In an MVS/XA system, this module will reside below 16 megabytes.

EXECUTION-TIME LOADING OF LIBRARY MODULES

When you link-edit a program using the Release 4.0 library, you can choose to have all library modules (other than the mathematical routines) either link-edited into the load module with compiler-generated code, or to have many of them loaded dynamically at execution time. Execution-time loading has several advantages. It reduces auxiliary storage requirements for load modules, speeds link-editing, and, in an MVS/XA environment, allows some library routines to be placed in the extended link pack area. Note that this new feature replaces the previous technique used to load the reentrant library.

COMPOSITE MODULES

If you choose the link-edit method, or link mode, no further loading is required at execution time. If you choose execution-time loading, or load mode, each module is loaded the first time it is used, unless it has been previously loaded. Because execution-time performance suffers if a large number of library modules are individually loaded, the modules to be loaded at execution time may be combined into composite load modules.

There are three of these composite modules installed in VFORTLIB for an MVS/XA system and two for other systems:

IFYVLBCM contains nonreentrant library modules, including the library common work area and the initialization module.

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

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

IFYVRENC includes all the loadable reentrant modules for systems other than MVS/XA. In an MVS/XA system, this module is not used.

As part of its initialization procedure in load mode, the library loads the composite modules listed above. The only modules that need to be loaded separately after initialization are those that are not contained in the composite modules. At installation time (or at any time thereafter), you may add or

delete library modules from the composite load modules to further tune your system. For example, if keyed access (VSAM KSDS) is not normally used at your installation, you may choose not to place the modules that perform these functions in the composite load modules. This reduces their size. You will then have to load the direct access and keyed access I/O modules individually if you ever need them. (These modules may reside in the link pack area so they don't need to be brought into your region, or they may be brought into your region from the library containing them.)

SELECTION OF LOAD MODE OR LINK MODE

After installation of the VS FORTRAN library, you may update the installation's cataloged procedures (for example, FORTVCLG for compile, link-edit, and go), to specify the libraries needed for use in link mode. All procedures provided with the product are set up for load mode. The procedures for specifying libraries in load mode or link mode are described below.

Specifying Libraries in Load Mode

For operation in load mode, provide SYS1.VFORTLIB but not SYS1.VLNKMLIB to the linkage editor to use when including VS FORTRAN library modules. Specify only SYS1.VFORTLIB in the DD statement for SYSLIB in the linkage editor step:

```
//SYSLIB   DD  DSN=SYS1.VFORTLIB,DISP=SHR
```

To execute a program that has been link-edited in load mode, make VFORTLIB available for the execution step by performing one of the following steps.

1. Concatenate SYS1.VFORTLIB to SYS1.LINKLIB in the system link list so that SYS1.VFORTLIB will be searched as part of the link library without JOBLIB or STEPLIB DD statements. The reentrant composite modules IFYVRENA (MVS/XA only), IFYVRENB (MVS/XA only), and IFYVRENC (non-XA only), as well as selected individual reentrant modules, may be placed in the link pack area (SYS1.LPALIB). The copy of the modules in the link pack area will be used without searching SYS1.VFORTLIB. (If maintenance affects any modules in the link pack area, the updated copies of the modules must be copied into the link pack area from SYS1.VFORTLIB.)

2. Place the following JOBLIB DD statement in the JCL for the job which executes the VS FORTRAN program:

   ```
   //JOBLIB   DD  DSN=SYS1.VFORTLIB,DISP=SHR
   ```

   or place the following STEPLIB DD statement in the JCL for the step which executes the VS FORTRAN program:

   ```
   //STEPLIB  DD  DSN=SYS1.VFORTLIB,DISP=SHR
   ```

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

3. If you want to use a step library or job library in addition to loading reentrant modules from the link pack area, you must do the following:

   a. After tailoring the composite modules, place the reentrant composite modules IFYVRENA (MVS/XA only), IFYVRENB (MVS/XA only), and IFYVRENC (non-XA only) in the link pack area (library SYS1.LPALIB).

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

c. Create a new library that contains all modules from SYS1.VFORTLIB minus the modules (either composite modules or individual modules) that have been placed in the link pack area. Make this library available as either a step library or as a job library for the execution of the VS FORTRAN program.

If maintenance affects any of the modules in the link pack area or your new library, the updated modules must be copied from SYS1.VFORTLIB.

Specifying Libraries in Link Mode

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

```
//SYSLIB  DD  DSN=SYS1.VLNKMLIB,DISP=SHR
//        DD  DSN=SYS1.VFORTLIB,DISP=SHR
```

A program which is link-edited in link mode does not require any VS FORTRAN libraries at execution time (although other load module libraries may be required for reentrant programs).

DECIDING WHAT TO INCLUDE IN COMPOSITE MODULES

You may update the composite modules IFYVLBCM, IFYVRENA (for MVS/XA only), IFYVRENB (for MVS/XA only), and IFYVRENC (non-XA only) to include only the library routines commonly used at your installation. You should base your choice to include or not include a module in the composite module upon the following considerations:

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

Each library module not in the applicable composite module is loaded from the VFORTLIB library when the module is first referenced at execution-time. However, these modules could be placed in a link pack area under their own module names, and then loaded individually.

BUILDING THE COMPOSITE MODULES

The following tables list the library modules you can include in the various composite modules. The "Size" column lists approximate module sizes, in hexadecimal. The "Default Set" column indicates which modules are placed into the composite modules during installation. Except for those modules that must be in the composite modules, you can subsequently add or delete modules in this set to match the needs at your installation.

If a module performs a function used frequently at your installation, you should consider including it in your composite module even if you are trying to limit the size of the composite module.

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

Composite Module IFYVLBCM

REQUIRED MODULES

Module	Size	Default Set	Function
IFYVLBC0	12D0	X	Library common work area
IFYVBLN$	34	X	Internal linkage routine
IFYVCOM$	34	X	Internal linkage routine
IFYVCOM2	BCC	X	Initialization/termination
IFYVCNO$	78	X	Internal linkage routine
IFYVCNI$	78	X	Internal linkage routine
IFYVCVT$	30C	X	Internal linkage routine
IFYVDIO$	3A	X	Internal linkage routine
IFYVEMG$	74	X	Internal linkage routine
IFYVERE$	34	X	Internal linkage routine
IFYVERS$	60	X	Internal linkage routine
IFYVFNTH	8EB	X	Program interrupt handler
IFYVIIO$	3A	X	Internal linkage routine
IFYVKIO$	3A	X	Internal linkage routine
IFYVLOAD	2B0	X	Loader
IFYVLOC$	34	X	Internal linkage routine
IFYVPARM	2BC	X	Execution time parameters
IFYVPOS$	34	X	Internal linkage routine
IFYVSPIE	134	X	Interrupt interceptor
IFYVSTA$	10C	X	Internal linkage routine
IFYVTRC$	34	X	Internal linkage routine
IFYVVIO$	88	X	Internal linkage routine
IFYUATBL	varies	X	Unit assignment table
IFYUOPT	5B8	X	Error option table
Total	3C29+	23	

OPTIONAL MODULES

Module	Note	Size	Default Set	Function
IFYDIOCP		1B0		Define file (LANGLVL 66)
IFYDSPAP	2	51C		Dimension calculator
IFYIBCOP	1	7A4		Pre-VS FORTRAN interface
IFYLDFIP	2	420		List-directed I/O
IFYNAMEP	2	384		Namelist I/O
IFYSDUMQ		2156		SDUMP subroutine
IFYVASYP		AC0		Asynchronous I/O
IFYVDBUP		1058		Debugging packet
IFYVDUMQ		6D4		DUMP/PDUMP subroutine
IFYVIONP		1026		Namelist I/O
IFYVLOCA		593		Statement number locator
IFYVMOPP		450		Extended error handling
IFYVPOSA		2D9E		Post ABEND processor
IFYVSCOP	3	600		Pre-Release 4.0 interface
IFYVSPAP		46C		Dimension calculator

Notes:

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

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

3. The module IFYVSCOP is used when running object decks produced by the VS FORTRAN compiler from releases 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.

BUILDING THE COMPOSITE MODULE IFYVLBCM: The composite module IFYVLBCM is created in a linkage editor step as follows:

```
//LKEDLBCM  EXEC PGM=IEWL,PARM='XREF,REUS'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=SYS1.VFORTLIB,DISP=OLD
//SYSLIB    DD   DSN=SYS1.VFORTLIB,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(IFYVLBC0)
 INCLUDE SYSLIB(IFYxxxxx)
 .
 .
 .
 ORDER IFYVLBC0
 ENTRY IFYLBCOM
 NAME  IFYVLBCM(R)
/*
```

The linkage editor step creates the load module IFYVLBCM in the library SYS1.VFORTLIB, and replaces a previous copy of the load module, if one exists. The inclusion of any of the optional modules in the composite module IFYVLBCM is controlled by the linkage editor INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module you decide to include. Except for the module IFYVLBC0, no INCLUDE statements should be provided for the modules listed above as "Required."

Composite Module IFYVRENC for MVS/SP only (non-XA)

REQUIRED MODULES

Module	Size	Default Set	Function
IFYVREN	FC	X	Internal linkage module
IFYVGMFM	1A9	X	GETMAIN/FREEMAIN
IFYVSIOS	22F8	X	Sequential I/O services
Total	259D	3	

OPTIONAL MODULES

Module	Size	Default Set	Function
IFYDDCMP	270		Dynamic common
IFYVASUB	D1C		Asynchronous I/O
IFYVBLNT	1EC	X	Implied DO in I/O
IFYVCLOP	230	X	CLOSE statement
IFYVCOMH	1502	X	Formatted I/O
IFYVCONI	2DC	X	Input floating-pt conversion
IFYVCONO	754	X	Output floating-pt conversion
IFYVCVTH	114C	X	Data conversion
IFYVEMGN	F00	X	Error message generator
IFYVERRE	21C	X	Error summary
IFYVDIOS	1744		Direct access I/O services
IFYVIIOS	27C		Internal file services
IFYVINQP	964		INQUIRE statement
IFYVIOCP	296		BACKSPACE, REWIND, ENDFILE
IFYVIOFP	6A8	X	Formatted I/O
IFYVIOLP	11E8	X	List-directed I/O
IFYVIOUP	ACA	X	Unformatted I/O
IFYVKIOS	2A70		Keyed access I/O services
IFYVLINP	234		Link to reentrant CSECT
IFYVMSKL	467F	X	Message skeletons
IFYVOPEP	688	X	OPEN statement
IFYVSTAE	82C		ABEND processor
IFYVTEN	2C0	X	Powers of ten table
IFYVTRCH	88C	X	Traceback generator
IFYVVIOS	19B4		Nonkeyed VSAM I/O services

BUILDING THE COMPOSITE MODULE IFYVRENC IN MVS/SP (NON-XA): For use in a non-XA version of an MVS/SP system, the composite module IFYVRENC is created in a linkage editor step as follows:

```
//LKEDRENC  EXEC PGM=IEWL,PARM='XREF,RENT'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=SYS1.VFORTLIB,DISP=OLD
//SYSLIB    DD   DSN=SYS1.VFORTLIB,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(IFYVREN)
 INCLUDE SYSLIB(IFYxxxxx)
  .
  .
  .
 ORDER IFYVREN
 ENTRY IFYVREN
 NAME  IFYVRENC(R)
/*
```

The linkage editor step creates the load module IFYVRENC in library SYS1.VFORTLIB; any previous copy of the load module is replaced. The inclusion of any of the optional reentrant modules in the composite module IFYVRENC is controlled by the

linkage editor INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module you decide to include. Except for the module IFYVREN, no INCLUDE statements should be provided for the modules listed above as "Required."

After the composite module IFYVRENC has been created, it may be placed in the pageable link pack area for shared use by all regions. If it is not placed in the link pack area, it is loaded from the library SYS1.VFORTLIB.

Composite Module IFYVRENA (MVS/XA only)

REQUIRED MODULES

Module	Size	Default Set	Function
IFYVAREN	D8	X	Internal linkage module
IFYVGMFM	1A9	X	GETMAIN/FREEMAIN
Total	281	2	

OPTIONAL MODULES

Module	Size	Default Set	Function
IFYDDCMP	270	X	Dynamic common
IFYVBLNT	1EC	X	Implied DO in I/O
IFYVCLOP	230	X	CLOSE statement
IFYVCOMH	1502	X	Formatted I/O
IFYVCONI	2DC	X	Input floating-pt conversion
IFYVCONO	754	X	Output floating-pt conversion
IFYVCVTH	114C	X	Data conversion
IFYVEMGN	F00	X	Error message generator
IFYVERRE	21C	X	Error summary
IFYVIIOS	27C	X	Internal file services
IFYVINQP	964	X	INQUIRE statement
IFYVIOCP	296	X	BACKSPACE, REWIND, ENDFILE
IFYVIOFP	6A8	X	Formatted I/O
IFYVIOLP	11E8	X	List-directed I/O
IFYVIOUP	ACA	X	Unformatted I/O
IFYVLINP	234	X	Link to reentrant CSECT
IFYVMSKL	467F	X	Message skeletons
IFYVOPEP	688	X	OPEN statement
IFYVSTAE	82C	X	ABEND processor
IFYVTEN	2C0	X	Powers of ten table
IFYVTRCH	88C	X	Traceback generator

BUILDING THE COMPOSITE MODULE IFYVRENA FOR MVS/XA: The composite module IFYVRENA is created for use in an MVS/XA system in a linkage editor step as follows:

```
//LKEDRENA  EXEC PGM=IEWL,PARM='XREF,RENT'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=SYS1.VFORTLIB,DISP=OLD
//SYSLIB    DD   DSN=SYS1.VFORTLIB,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(IFYVAREN)
 INCLUDE SYSLIB(IFYxxxxx)
  .
  .
  .
 ORDER IFYVAREN
 ENTRY IFYVAREN
 MODE  AMODE(31),RMODE(ANY)
 NAME  IFYVRENA(R)
/*
```

The linkage editor step creates the load module IFYVRENA in library SYS1.VFORTLIB; any previous copy of the load module is replaced. The module should have a residence mode of ANY so that it can reside above the 16-megabyte virtual storage line. The inclusion of any of the optional reentrant modules in the composite module IFYVRENA is controlled by the linkage editor INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module you decide to include. Except for the module IFYVAREN, no INCLUDE statements should be provided for the modules listed above as "Required."

After the composite module IFYVRENA has been created, it may be placed in the extended pageable link pack area (ELPA) for shared use by all regions. If it is not placed in the link pack area, it is loaded from the library SYS1.VFORTLIB.

Composite Module IFYVRENB for MVS/XA system

REQUIRED MODULES

Module	Size	Default Set	Function
IFYVBREN	FC	X	Internal linkage module
IFYVSIOS	22F8	X	Sequential I/O services
Total	23F4	2	

OPTIONAL MODULES

Module	Size	Default Set	Function
IFYVASUB	D1C		Asynchronous I/O
IFYVDIOS	1744		Direct access I/O services
IFYVKIOS	2A70		Keyed access I/O services
IFYVVIOS	19B4		Non-keyed VSAM I/O service

BUILDING THE COMPOSITE MODULE IFYVRENB IN MVS/XA: The composite module IFYVRENB is created for use in an MVS/XA system in a linkage editor step as follows:

```
//LKEDRENB  EXEC PGM=IEWL,PARM='XREF,RENT'
//SYSPRINT  DD   SYSOUT=A
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSLMOD   DD   DSN=SYS1.VFORTLIB,DISP=OLD
//SYSLIB    DD   DSN=SYS1.VFORTLIB,DISP=SHR
//SYSLIN    DD   *
 INCLUDE SYSLIB(IFYVBREN)
 INCLUDE SYSLIB(IFYxxxxx)
 .
 .
 .
 ORDER IFYVBREN
 ENTRY IFYVBREN
 MODE  AMODE(31),RMODE(24)
 NAME  IFYVRENB(R)
/*
```

The linkage editor step creates the load module IFYVRENB in library SYS1.VFORTLIB; any previous copy of the load module is replaced. The module must have a residence mode of 24 so that it resides below the 16-megabyte virtual storage line. The inclusion of any of the optional reentrant modules in the composite module IFYVRENB is controlled by the linkage editor INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module you decide to include. Except for the module IFYVBREN, no INCLUDE statements should be provided for the modules listed above as "Required."

After the composite module IFYVRENB has been created, it may be placed in the pageable link pack area for shared use by all regions. If it is not placed in the link pack area, it is loaded from the library SYS1.VFORTLIB.

CHAPTER 9. CUSTOMIZATION UNDER VSE

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

Alternative mathematical library subroutines

Cataloged procedures

Compiler and Library defaults

Execution-time logical units

Execution-time loading of library modules

ALTERNATIVE MATHEMATICAL LIBRARY SUBROUTINES

You may choose to replace the supplied standard VS FORTRAN routines with the alternative mathematical library subroutines, or you may insert the alternative routines into a local user library so they can be made available when needed by the individual user.

The INSTALL job places both the standard and alternative mathematical library subroutines into your VFORTLIB relocatable library. If you intend to use the alternatives, the standard routines must be deleted and the alternatives renamed using the MAINT program. The required statements are:

```
DELETR IFYFDXPD
DELETR IFYLCOS
DELETR IFYLEXP
DELETR IFYLSIN
DELETR IFYFRXPR
DELETR IFYSEXP
DELETR IFYLTNCT
RENAMR IFYWDXPD,IFYFDXPD
RENAMR IFYWLCOS,IFYLCOS
RENAMR IFYWLEXP,IFYLEXP
RENAMR IFYWLSIN,IFYLSIN
RENAMR IFYWRXPR,IFYFRXPR
RENAMR IFYWSEXP,IFYSEXP
RENAMR IFYWTNCT,IFYLTNCT
```

If you do not want to use the alternative routines, you can delete them using the following statements:

```
DELETR IFYWDXPD
DELETR IFYWLCOS
DELETR IFYWLEXP
DELETR IFYWLSIN
DELETR IFYWRXPR
DELETR IFYWSEXP
DELETR IFYWTNCT
```

If you want to keep both the standard and the alternative mathematical subroutines, you can copy the alternative subroutines (those beginning with IFYW) to a private relocatable library. To make the alternative routines available, you need to rename them to the standard subroutine names, and use a LIBDEF statement at link-edit time to define the relocatable library you want to use. The LIBDEF statement should look like this:

```
LIBDEF RL SEARCH=xxxx
```

where xxxx is the file name of the private relocatable library.

CATALOGED PROCEDURES

You may want to write and catalog procedures for users to compile, link, and execute FORTRAN jobs. To catalog a procedure in the procedure library, submit a CATALP statement specifying the procedure name. Rules for naming the procedures are given in VSE/Advanced Functions System Control Statements.

The statements to be included in the procedure follow the CATALP statement; they can be job control or linkage editor control statements, or both. The end of the control statements to be cataloged must be indicated by an end-of-procedure delimiter, which is usually /+.

Each procedure cataloged in the procedure library should have a unique identity. This identity is required if you want to modify the job stream at execution. Therefore, when cataloging, identify each control statement in columns 73 through 79 (blanks may be embedded).

Additional information on writing and modifying VSE/Advanced Functions cataloged procedures can be found in VS FORTRAN Programming Guide.

COMPILER AND LIBRARY DEFAULTS

MODIFYING COMPILER DEFAULT OPTIONS

The compiler default options provided by IFXOOPTS are described in Chapter 3, "The Installation Macros: VSFORTC and VSFORTL" on page 5. If you want to change the default options, you must code the VSFORTC macro with the desired new default options. When coding, column 1 must be blank. VSFORTC may appear anywhere before column 72 but must precede the operands by at least one blank. The operands are separated by commas and may be continued on any number of cards as long as column 72 contains a nonblank character and the data on the following card begins in column 16. You do not need to code all keyword parameters (PUNCH, SORLIST, STORMAP, and so forth).

Assemble the macro and relink-edit the compiler. The following example must be coded and adjusted for your specific system configuration.

The first job below assembles the VSFORTC macro and punches the deck to temporary space allocated on DASD. Then the job catalogs the new version of IFXOOPTS in the relocatable library, making it available for the relink-edit of the compiler as shown in the second job sample.

If you will be compiling with VS FORTRAN in a batch environment and will not be using a SYSTERM data set, specify the NOTRMFLG and NOTERMINAL options to avoid messages about having no terminal "online."

```
// JOB ASSEDCK
// DLBL IJSYSPH,'PCHFILE',0          ****NNNNNN=VOLID FOR DASD
// EXTENT SYSPCH,NNNNNN,,,nnnnn,100  **** nnnnn=starting track of file
ASSGN SYSPCH,X'DDD'                  **** DDD=DEVICE ADDRESS
// OPTION DECK
// EXEC ASSEMBLY
  PUNCH ' CATALR IFX00PTS '
  VSFORTC SYSTEM=DOS/VSE,.......OPTIONS AS DESIRED ..
  END
/*
CLOSE SYSPCH,X'00D'
// DLBL IJSYSIN,'PCHFILE'
// EXTENT SYSIPT
ASSGN SYSIPT,X'DDD'              **** DDD=DEVICE ADDRESS USED ABOVE
// EXEC MAINT
/&
CLOSE SYSIPT,X'00C'


// JOB FORTLKED VFORTRAN
* THIS JOB WILL LINK EDIT THE VS FORTRAN COMPILER
// PAUSE READY TO LINK-EDIT VS FORTRAN COMPILER
// OPTION CATAL,NODUMP,LOG
 ACTION MAP,NOAUTO,CANCEL
 INCLUDE VF03LINK
// EXEC LNKEDT
/&
```

MODIFYING LIBRARY OBJECT-TIME I/O OPTIONS

The VS FORTRAN object-time I/O options provided by IFYUATBL are described in Chapter 3, "The Installation Macros: VSFORTC and VSFORTL" on page 5 under VSFORTL. If you want to change the number of units in the VS FORTRAN unit assignment table (or the VS FORTRAN unit numbers that will be used as defaults for READ, PUNCH, and WRITE statements), you must code the VSFORTL macro, assemble it, and catalog the resulting module in the relocatable library. Then relink-edit the composite module IFYVLBCM. Note that the default number of units in the VS FORTRAN unit assignment table is 8.

The following example may be coded and adjusted for your specific system configuration.

```
// JOB ASSEDCK
// DLBL IJSYSPH,'PCHFILE',0          **** NNNNNN=VOLID ON DASD
// EXTENT SYSPCH,NNNNNN,,,nnnnn,100 **** nnnnn=starting track of file
ASSGN SYSPCH,X'DDD'                  **** DDD=DEVICE ADDRESS
// OPTION DECK
// EXEC ASSEMBLY
  PUNCH ' CATALR IFYUATBL '
  VSFORTL SYSTEM=DOS/VSE,..... OPTIONS AS DESIRED ..
  END
/*
CLOSE SYSPCH,X'00D'
// DLBL IJSYSIN,'PCHFILE'
// EXTENT SYSIPT
ASSGN SYSIPT,X'DDD'              **** DDD=DEVICE ADDRESS USED ABOVE
// EXEC MAINT
/&
CLOSE SYSIPT,X'00C'
```

EXECUTION-TIME LOGICAL UNITS

The following table lists the default use of the FORTRAN reference numbers 0 through 8, which are established when you install the product. Units 9 through 99 may be added by reassembling the VSFORTL macro and replacing the unit assignment table module (IFYUATBL). See "Modifying Library Object-Time I/O Options" on page 54 for more information.

FORTRAN Ref. No.	Logical Unit	DOS File Name	Function (Primary)	Device Type
0	SYS000	IJSYS00	Program data set	Unit record Magnetic tape Direct access
1	SYS001	IJSYS01	Program data set	Unit record Magnetic tape Direct access
2	SYS002	IJSYS02	Program data set	Unit record Magnetic tape Direct access
3	SYS003	IJSYS03	Program data set	Unit record Magnetic tape Direct access
4	SYS004	IJSYS04	Program data set	Unit record Magnetic tape Direct access
5	SYSIPT or SYSIN	IJSYSIP	Input data set to load module	Card reader Magnetic tape Direct access
6	SYSLST	IJSYSLS	Printed output data	Printer Magnetic tape Direct access
7	SYSPCH	IJSYSPH	Punched output data	Card punch Magnetic tape Direct access
8 thru 99	SYS005 thru SYS096	IJSYS05 thru IJSYS96	Program data set	Unit record Magnetic tape Direct access

EXECUTION-TIME LOADING OF LIBRARY MODULES

When you link-edit a program using the Release 4.0 library, you may choose to have all library modules (other than the mathematical routines) either link-edited into your phase with the compiler-generated code, or loaded dynamically at execution time. Execution-time loading has several advantages. It reduces auxiliary storage requirements for phases in the core image library and speeds link-editing.

COMPOSITE MODULES

If you choose the link-edit method, or link mode, no further loading is required at execution time. If you choose execution-time loading, or load mode, each module is loaded the first time it is used, unless it has been previously loaded. Because execution-time performance is not as good if a large number of library modules are individually loaded, the modules to be loaded at execution time may be combined into composite modules. There are two composite modules built during installation:

IFYVLBCM contains nonreentrant library modules, including the library common work area and the initialization module.

IFYVRENC contains the loadable reentrant modules.

As part of its initialization procedure in load mode, the library loads the composite modules listed above. The only modules that must be loaded separately after initialization are those not contained in the composite modules. At installation time (or at any time thereafter), you may add or delete library modules from the composite load modules to further tune your system. For example, if direct access or keyed access is not normally used at your installation, you may choose not to place the modules that perform these functions in the composite modules. You could thus reduce the size of these modules. The direct access and keyed access I/O modules would then have to be loaded individually should they ever be needed. (These modules may reside in the shared virtual area so they don't need to be brought into your partition, or they may be brought into your partition from the library containing them.)

SELECTION OF LOAD MODE OR LINK MODE

After installation of the VS FORTRAN library, you must update the operational procedures to specify the libraries needed for use in load mode or link mode. If you need to specify the libraries, do the following:

Specifying Libraries in Load Mode

- For operation in load mode, provide VFORTLIB but not VLNKMLIB to the linkage editor for its use when it includes VS FORTRAN library modules. Make only the relocatable library VFORTLIB available for the linkage editor step.

```
// DLBL VFORTLI,'A5748F03.SYSRLB.VFORTLIB'
// EXTENT SYSmmm,volser
// ASSGN SYSmmm,cuu
// LIBDEF RL,SEARCH=(VFORTLI),TEMP
```

- To make the relocatable library available to all linkage editor steps, put the DLBL and EXTENT statements in the standard label job, make permanent assignments, and specify PERM on the LIBDEF command instead of TEMP.

To execute a program that has been link-edited in load mode, make VFLODLIB available for the execution step.

1. Use the following statements in the step that executes the VS FORTRAN program:

```
// DLBL VFLODLI,'A5748F03.SYSCLB.VFLODLIB'
// EXTENT SYSnnn,volser
// ASSGN SYSnnn,cuu
// LIBDEF CL,SEARCH=(VFLODLI),TEMP
```

2. To make VFLODLIB available to all jobs, put the DLBL and EXTENT statements in the standard label area, make the

SYSnnn assignment permanent, and specify PERM on the LIBDEF command.

Specifying Libraries in Link Mode

- For operation in link mode, concatenate VLNKMLIB ahead of VFORTLIB for use by the linkage editor when it includes VS FORTRAN library modules. Make the relocatable libraries VLNKMLIB and VFORTLIB available for the linkage editor step:

```
// DLBL VFLKMLI,'A5748F03.SYSRLB.VLNKMLIB'
// EXTENT SYSnnn,volser
// ASSGN SYSnnn,cuu
// DLBL VFORTLI,'A5748F03.SYSRLB.VFORTLIB'
// EXTENT SYSmmm,volser
// ASSGN SYSmmm,cuu
// LIBDEF RL,SEARCH=(VFLKMLI,VFORTLI),TEMP
```

- Alternatively, put the DLBL and EXTENT statements in the standard label area, make permanent assignments, and specify PERM on the LIBDEF command. This will make the relocatable libraries available to all linkage editor steps.

- A program link-edited for execution in link mode requires no VS FORTRAN libraries at execution time.

DECIDING WHAT TO INCLUDE IN COMPOSITE MODULES

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

- Because IFYVLBCM contains the nonreentrant modules, it must be loaded into your partition for each execution of a VS FORTRAN program. Including all possible nonreentrant modules may require the partition size to be larger than would otherwise be necessary.

- If IFYVRENC is not in the SVA, it must be loaded into your partition. Including all possible reentrant modules may require the partition size to be larger than would otherwise be necessary.

- If IFYVRENC is in the SVA, including a large number of the reentrant modules in the composite module has no effect upon the partition size. However, the larger IFYVRENC does require additional virtual storage in the SVA.

Each library module not in the applicable composite module is loaded from the VFLODLIB library when the module is first referenced at execution time. These modules could be placed in an SVA as separate modules under their own module names and loaded individually.

BUILDING THE COMPOSITE MODULES

The following tables list the library modules you can include in the various composite modules. The "Size" column lists approximate module sizes, in hexadecimal. The "Default Set" column indicates which modules are placed into the composite modules during the installation process. Except for the modules that must be in the composite modules, you can subsequently add or delete modules in this set to match the needs at your installation.

If a module performs a function used frequently at your installation, you should consider including it in your composite module even if you are trying to limit the size of the composite module.

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

Composite Module IFYVLBCM

REQUIRED MODULES

Module	Size	Default Set	Function
IFYDLBC0	12D0	X	Library common work area
IFYVBLN$	34	X	Internal linkage routine
IFYVCOM$	34	X	Internal linkage routine
IFYDCOM2	780	X	Initialization/termination
IFYVCNO$	78	X	Internal linkage routine
IFYVCNI$	78	X	Internal linkage routine
IFYVCVT$	30C	X	Internal linkage routine
IFYDDIO$	3A	X	Internal linkage routine
IFYVEMG$	74	X	Internal linkage routine
IFYVERE$	34	X	Internal linkage routine
IFYVERS$	60	X	Internal linkage routine
IFYDFNTH	8E4	X	Program interrupt handler
IFYVIIO$	3A	X	Internal linkage routine
IFYDKIO$	3A	X	Internal linkage routine
IFYDLOAD	220	X	Loader
IFYVLOC$	34	X	Internal linkage routine
IFYVPARM	2BC	X	Execution time parameters
IFYVPOS$	34	X	Internal linkage routine
IFYDSPIE	FF	X	Interrupt interceptor
IFYVTRC$	34	X	Internal linkage routine
IFYDVIO$	88	X	Internal linkage routine
IFYUATBL	varies	X	Unit assignment table
IFYUOPT	5B8	X	Error option table
Total	3605+	22	

OPTIONAL MODULES

Module	Note	Size	Default Set	Function
IFYDIOCP		1B0		Define file (LANGLVL 66)
IFYDSPAP	2	51C		Dimension calculator
IFYIBCOP	1	7A4		Pre-VS FORTRAN interface
IFYLDFIP	2	420		List-directed I/O
IFYNAMEP	2	384		Namelist I/O
IFYOPSYP		2156	X	VSE system services
IFYSDUMQ		AC0		SDUMP subroutine
IFYVDBUP		1058		Debugging packet
IFYVDUMQ		6D4		DUMP/PDUMP subroutine
IFYVIONP		1026		Namelist I/O
IFYVLOCA		593		Statement number locator
IFYVMOPP		450		Extended error handling
IFYVPOSA		2D9E		Post ABEND processor
IFYVSCOP	3	600		Pre-Rel. 4 interface
IFYVSPAP		46C		Dimension calculator

Notes:

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

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

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

BUILDING THE COMPOSITE MODULE IFYVLBCM: Composite module IFYVLBCM is created in a linkage editor step as follows:

```
// DLBL IJSYSRL,'A5748F03.SYSRLB.VFORTLIB'
// EXTENT ,volser
// ASSGN SYSRLB,cuu
// DLBL VFLODLI,'A5748F03.SYSCLB.VFLODLIB'
// EXTENT SYSnnn,volser
// ASSGN SYSnnn,cuu
// LIBDEF CL,TO=VFLODLI,TEMP
// OPTION CATAL
 ACTION MAP
 PHASE IFYVLBCM,*+0
 INCLUDE IFYDLBC0
 INCLUDE IFYxxxxx
 .
 .
 .
 ENTRY IFYLBCOM
// EXEC LNKEDT
```

The linkage editor step creates the phase IFYVLBCM in the core image library A5748F03.SYSCLB.VFLODLIB; any previous copy of the phase is replaced. The inclusion of any of the optional modules in composite module IFYVLBCM is controlled by the INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module that is to be included. A separate INCLUDE statement is required for each optional module that is to be in the composite module. Except for module IFYDLBC0, no INCLUDE statements should be provided for the modules listed above as "Required."

Composite Module IFYVRENC

REQUIRED MODULES

Module	Size	Default Set	Function
IFYDREN	FC	X	Internal linkage module
IFYDGMFM	1E5	X	GETMAIN/FREEMAIN
IFYDSIOS	1E52	X	Sequential I/O services
Total	2133	3	

OPTIONAL MODULES

Module	Size	Default Set	Function
IFYVBLNT	1EC	X	Implied DO in I/O
IFYVCLOP	230	X	CLOSE statement
IFYVCOMH	1502	X	Formatted I/O
IFYVCONI	2DC	X	Input floating-pt conversion
IFYVCONO	754	X	Output floating-pt conversion
IFYVCVTH	114C	X	Data conversion
IFYDDCMP	270		Dynamic common
IFYVEMGN	F00	X	Error message generator
IFYVERRE	21C	X	Error summary
IFYDDIOS	12B0		Direct access I/O services
IFYVIIOS	27C		Internal file services
IFYVINQP	964		INQUIRE statement
IFYVIOCP	296		BACKSPACE, REWIND, ENDFILE
IFYVIOFP	6A8	X	Formatted I/O
IFYVIOLP	11E8	X	List-directed I/O
IFYVIOUP	ACA	X	Unformatted I/O
IFYDKIOS	3190		Keyed access I/O services
IFYVMSKL	467F	X	Message skeletons
IFYVOPEP	688	X	OPEN statement
IFYVTEN	2C0	X	Powers of ten table
IFYVTRCH	88C	X	Traceback generator
IFYDVIOS	1E9C		Non-keyed VSAM I/O services

BUILDING THE COMPOSITE MODULE IFYVRENC: The composite module IFYVRENC is created in a linkage editor step as follows:

```
// DLBL IJSYSRL,'A5748F03.SYSRLB.VFORTLIB'
// EXTENT ,volser
// ASSGN SYSRLB,cuu
// DLBL VFLODLI,'A5748F03.SYSCLB.VFLODLIB'
// EXTENT SYSnnn,volser
// ASSGN SYSnnn,cuu
// LIBDEF CL,TO=VFLODLI,TEMP
// OPTION CATAL
 ACTION MAP
 PHASE IFYVRENC,*+0,SVA
 INCLUDE IFYDREN
 INCLUDE IFYxxxxx
  .
  .
  .
 ENTRY IFYDREN
// EXEC LNKEDT
```

The linkage editor step creates the phase IFYVRENC in the core image library A5748F03.SYSCLB.VFLODLIB; any previous copy of the phase is replaced. The inclusion of any of the optional reentrant modules in composite module IFYVRENC is controlled by the INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module that is to be included. A separate INCLUDE statement is required for each optional module that is to be in the composite module. Except for module IFYDREN, no INCLUDE statements should be provided for the modules listed above as "Required."

After composite module IFYVRENC has been created, it may be placed in the shared virtual area for shared use by all partitions. If it is not placed in the shared virtual area, it is loaded from the core image library A5748F03.SYSCLB.VFLODLIB.

CHAPTER 10. CUSTOMIZATION UNDER VM

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

Alternative mathematical library subroutines

The compiler as a discontiguous shared segment

Extended precision operations

The separation tool

Execution-time loading of library modules

ALTERNATIVE MATHEMATICAL LIBRARY SUBROUTINES

You may choose to replace the supplied standard VS FORTRAN routines with the alternative mathematical library subroutines, or you may insert the alternative routines into a local user library so they can be made available when needed by the individual user.

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

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

```
GLOBAL TXTLIB VALTLIB VFORTLIB CMSLIB
```

Or this statement for use in link mode:

```
GLOBAL TXTLIB VALTLIB VLNKMLIB VFORTLIB CMSLIB
```

THE COMPILER AS A DISCONTIGUOUS SHARED SEGMENT

You must complete the following steps before executing the installation EXEC (I5748F03) when installing the compiler as a shared segment:

1. Allocate permanent space on a CP-owned DASD volume to contain the saved segment (256 pages). (Refer to VM/SP Planning Guide and Reference for information on the amount of disk space needed.)

2. Define the segment to be saved by adding a NAMESYS macro to your installation's DMKSNT ASSEMBLE module (see VM/SP Planning Guide and Reference and VM/SP System Programmer's Guide). Choose the load address, using the following guidelines:

   - The address must be greater than the largest virtual machine of any VS FORTRAN user.
   - The address should not be unnecessarily high; if it is, storage is wasted for unreferenced CP segment table entries.
   - The address must not allow the VS FORTRAN shared segment to overlap any other shared segment that may be used at the same time.

The following example of the NAMESYS macro defines DSSVFORT. The sample numbers given illustrate a possible set of numbers and are not intended as the only location for a DCSS.

```
SAMPLE NAMESYS SYSNAME=DSSVFORT,
               SYSSIZE=1024K,
               SYSHRSG=(64,65,66,67,68,69,70,71,
                      72,73,74,75,76,77,78,79),     See Note A
               SYSPGCT=256,
               SYSPGNM=(1024-1279),                 See Note B
               VSYSADR=IGNORE,
               SYSVOL=VM7RES,                       See Note C
               SYSSTRT=(049,1)                      See Note D
```

Note A. Must be adjusted to your installation requirements.

Note B. The page numbers to be saved. To calculate the page numbers, divide the load address by 4K and convert the result to decimal.

Note C. The serial number of the storage volume allocated.

Note D. The starting cylinder and page address for the saved segment.

The above example will require a hexadecimal origin address of 400000.

PROCEDURE

1. Assemble the new system name table, DMKSNT, by using the GENERATE EXEC procedure as described in VM/SP Planning Guide and Reference.
2. Redefine a virtual storage size that exceeds the entire shared segment; that is, if the DCSS starting address is 400000 hexadecimal, a virtual storage of 6 megabytes is needed.
3. Refer to the VS FORTRAN Program Directory for updated information and specific space data needed for installation of VS FORTRAN.
4. Reply YES to the prompt asking if you are installing the VS FORTRAN compiler as a discontiguous shared segment, and be prepared to give the starting address. Invoke the installation EXEC, I5748F03, from E class privilege in order to execute the SAVESYS command.

EXTENDED PRECISION OPERATIONS

If the hardware you are using with VS FORTRAN does not support one or more extended precision arithmetic operations (add, subtract, multiply, or divide), you must include CMSLIB in the GLOBAL TXTLIB statement in order to make the required simulation modules available (IEAXPSIM, IEAXPDXR, IEAXPALL). Failure to do this will cause an abend during execution.

In load mode, the GLOBAL statements you'll need at execution are:

```
GLOBAL TXTLIB VFORTLIB CMSLIB
GLOBAL LOADLIB VFLODLIB
```

In link mode, the statements are:

```
GLOBAL TXTLIB VLNKMLIB VFORTLIB CMSLIB
GLOBAL LOADLIB
```

The GLOBAL command for a LOADLIB is not required for operation in link mode unless you are using routines you have link-edited into a loadlib.

THE SEPARATION TOOL

The VS FORTRAN separation tool separates the compiler's object output from a compile with the RENT option into reentrant and nonreentrant portions. This allows you to build shared modules with the reentrant portions.

The separation tool uses the object output (object deck or text file) from the compiler as input. It generates a listing file indicating the activity that took place, and also generates two object output files. The first file (ddname=SYSUT1) contains the nonreentrant CSECTs. The second file (ddname=SYSUT2) contains the reentrant CSECTs, and the table generated by the separation tool to help locate them.

The separation tool is distributed with the VS FORTRAN Library in the form of two text files, IFYVSFST and IFYVSFIO. Some of the ways to reinstall and execute the separation tool are described below. (Note that FILEDEF statements are not given here but are available in VS FORTRAN Programming Guide.)

- One method of accessing the separation tool is from VFORTLIB. Here we are using "rentpart" as the reentrant module name.

```
GLOBAL TXTLIB VFORTLIB
LOAD IFYVSFST rentpart
START * rentpart
```

- Another method is to designate only IFYVSFST as a module, and to load IFYVSFIO at execution.

```
GLOBAL TXTLIB VFORTLIB
LOAD IFYVSFST (CLEAR
GENMOD IFYVSFST
```

  then

```
IFYVSFST rentpart
```

- Another method is to create one module containing both IFYVSFST and IFYVSFIO, and to load this module at execution. This method improves startup performance, but requires additional disk space to contain the module.

```
GLOBAL TXTLIB VFORTLIB
LOAD IFYVSFST IFYVSFIO (CLEAR
GENMOD IFYVSFST
```

  then

```
IFYVSFST rentpart
```

- Another method is to create a load module made up of IFYVSFST and IFYVSFIO. This load module should occupy a LOADLIB. You can then invoke the separation tool with the OSRUN program.

  1. Get IFYVSFST and IFYVSFIO from VFORTLIB TXTLIB

```
CP SPOOL PUNCH TO *
PUNCH VFORTLIB TXTLIB * (MEMBER IFYVSFST
PUNCH VFORTLIB TXTLIB * (MEMBER IFYVSFIO
READ *
READ *
XEDIT IFYVSFST TEXT A    (to remove leftover records
BOTTOM                    from VFORTLIB TXTLIB)
UP
DELETE 2
FILE
XEDIT IFYVSFIO TEXT A    (to remove leftover records
BOTTOM                    from VFORTLIB TXTLIB)
UP
DELETE 2
FILE
```

2. Make one text file

```
COPYFILE IFYVSFIO TEXT A IFYVSFST TEXT A (APPEND
```

3. Link-edit the text file into VSFSTLIB LOADLIB

```
LKED IFYVSFST (NAME IFYVSFST LIBE VSFSTLIB
```

4. Run the separation tool, remembering your FILEDEF statements.

```
GLOBAL LOADLIB VSFSTLIB
OSRUN IFYVSFST PARM='RENTPART'
```

EXECUTION-TIME LOADING OF LIBRARY MODULES

When you create an executable program using the Release 4.0 library, you may choose to have all library modules (other than the mathematical routines) either made a part of your executable program along with the compiler-generated code, or loaded dynamically at execution time. Execution-time loading has several advantages. It reduces auxiliary storage requirements for your executable programs and speeds link-editing.

COMPOSITE MODULES

If you choose to have all the library modules included as part of your executable program (link mode), no further loading is required at execution time. If you choose execution-time loading (load mode), each module is loaded the first time it is used, unless it has been previously loaded. Because execution-time performance is not as good if a large number of library modules are individually loaded, the modules to be loaded at execution time may be combined into composite modules. The two composite modules are installed in VFLODLIB.

IFYVLBCM contains nonreentrant library modules, including the library common work area and the initialization module.

IFYVRENC contains all the loadable reentrant modules. One or more copies of this composite module may be placed in a discontiguous shared segment.

As part of its initialization procedure in load mode, the library loads the composite modules listed above. The only modules that need to be loaded separately after initialization are those not contained in the composite modules. At installation time (or at any time thereafter), you may add or delete library modules from the composite load modules to further tune your system. For example, if direct access or keyed access is not normally used at your installation, you may choose not to place the modules that perform these functions in the composite modules. You could thus reduce the size of these modules. The direct access and keyed access I/O modules would then have to be loaded individually should they ever be needed.

SELECTION OF LOAD MODE OR LINK MODE

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

Specifying Libraries in Load Mode

Specify the VFORTLIB TXTLIB but not the VLNKMLIB TXTLIB in the CMS GLOBAL command for use by the LOAD command:

```
GLOBAL TXTLIB VFORTLIB CMSLIB
```

Library text files, that is, CMS files with a name beginning with IFY and with a file type of TEXT, must not be on any accessed disk during the execution of the LOAD command unless the option NOAUTO is specified. (During installation of the VS FORTRAN library, the library text files should be placed on a different minidisk than the text libraries in order to eliminate the problems that would occur because of the omission of the NOAUTO option on the LOAD command.)

Specify the VFORTLIB TXTLIB in a FILEDEF command for use by the LKED command:

```
FILEDEF SYSLIB DISK VFORTLIB TXTLIB fm
```

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

```
GLOBAL LOADLIB VFLODLIB
```

Specifying Libraries in Link Mode

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

Specify the TXTLIBs VLNKMLIB and VFORTLIB in the CMS GLOBAL command for use by the LOAD command:

```
GLOBAL TXTLIB VLNKMLIB VFORTLIB CMSLIB
```

Library text files, that is, CMS files with a name beginning with IFY and with a file type of TEXT, must not be on any accessed disk during the execution of the LOAD command unless the option NOAUTO is specified. (During installation of the VS FORTRAN library, the library text files should be placed on a different minidisk than the text libraries in order to eliminate the problems that would occur because of the omission of the NOAUTO option on the LOAD command.)

Do not use the LKED command to create an executable program that operates in link mode.

A program created for execution in link mode does not require any VS FORTRAN libraries at execution time.

Issue the following GLOBAL command to be sure that VFLODLIB LOADLIB is not available:

```
GLOBAL LOADLIB
```

DECIDING WHAT TO INCLUDE IN COMPOSITE MODULES

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

- Because IFYVLBCM contains the nonreentrant modules, it must be loaded into your virtual machine for each execution of a VS FORTRAN program. Including all possible nonreentrant modules may cause the storage required for the program to be larger than necessary.
- If IFYVRENC is not in the DCSS, it must be loaded into your virtual machine. Including all possible reentrant modules may cause the storage required for the program to be larger than necessary.
- If IFYVRENC is in the DCSS, including a large number of the reentrant modules in the composite module has no effect upon the storage required for the program. However, the larger IFYVRENC does require additional virtual storage for the DCSS.

Each library module not in the applicable composite module is loaded from the VFLODLIB library when the module is first referenced at execution time. The reentrant modules, that is, those that can be placed in the composite module IFYVRENC, could also be placed in a DCSS as separate modules under their own module names, and loaded individually.

BUILDING THE COMPOSITE MODULES

The following tables list the library modules you can include in the various composite modules. The "Size" column lists approximate module sizes in hexadecimal. The "Default Set" column indicates which modules are placed into the composite modules during the installation process. Except for the modules that must be in the composite modules, you can subsequently add or delete modules in this set to match the needs at your installation.

If a module performs a function used frequently at your installation, you should consider including it in your composite module even if you are trying to limit the size of the composite module.

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

Composite Module IFYVLBCM

REQUIRED MODULES

Module	Size	Default Set	Function
IFYCLBC0	12D0	X	Library common work area
IFYVBLN$	34	X	Internal linkage routine
IFYVCMSS	3B0	X	CMS interface
IFYVCOM$	34	X	Internal linkage routine
IFYVCOM2	BCC	X	Initialization/termination
IFYVCNO$	78	X	Internal linkage routine
IFYVCNI$	78	X	Internal linkage routine
IFYVCVT$	30C	X	Internal linkage routine
IFYVDIO$	3A	X	Internal linkage routine
IFYVEMG$	74	X	Internal linkage routine
IFYVERE$	34	X	Internal linkage routine
IFYVERS$	60	X	Internal linkage routine
IFYVFNTH	8EB	X	Program interrupt handler
IFYVIIO$	3A	X	Internal linkage routine
IFYVKIO$	3A	X	Internal linkage routine
IFYVLOAD	2B0	X	Loader
IFYVLOC$	34	X	Internal linkage routine
IFYVPARM	2BC	X	Execution time parameters
IFYVPOS$	34	X	Internal linkage routine
IFYVSPIE	134	X	Interrupt interceptor
IFYVSTA$	10C	X	Internal linkage routine
IFYVTRC$	34	X	Internal linkage routine
IFYCVIO$	88	X	Internal linkage routine
IFYUATBL	varies	X	Unit assignment table
IFYUOPT	5B8	X	Error option table
Total	3FD9+	24	

OPTIONAL MODULES

Module	Note	Size	Default Set	Function
IFYCRNAM	4			DCSS name list
IFYDIOCP		1B0		Define file (LANGLVL 66)
IFYDSPAP	2	51C		Dimension calculator
IFYIBCOP	1	7A4		Pre-VS FORTRAN interface
IFYLDFIP	2	420		List-directed I/O
IFYNAMEP	2	384		Namelist I/O
IFYSDUMQ		2156		SDUMP subroutine
IFYVDBUP		1058		Debugging packet
IFYVDUMQ		6D4		DUMP/PDUMP subroutine
IFYVIONP		1026		Namelist I/O
IFYVLOCA		593		Statement number locator
IFYVMOPP		450		Extended error handling
IFYVPOSA		2D9E		Post ABEND processor
IFYVSCOP	3	600		Pre-Rel. 4 interface
IFYVSPAP		46C		Dimension calculator

Notes:

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

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

3. Module IFYVSCOP is used when running object decks produced by the VS FORTRAN compiler from prior to Release 4.0. It is needed for formatted and unformatted I/O and for initialization from a main program or from a subroutine with characters arguments.

4. If you have placed IFYVRENC in a shared segment, you must assemble module IFYCRNAM by coding a VSFCRNAM macro instruction, described in item 4 under "IFYVRENC as a Discontiguous Shared Segment" on page 70. The macro instruction specifies the names of your shared segments that have copies of the reentrant composite module IFYVRENC. After assembling module IFYCRNAM, you must incorporate it into composite module IFYVLBCM, as described in "Building the composite module IFYVLBCM." If module IFYCRNAM is not in the composite module, only the shared segment name IFYVRENC will be accessed.

BUILDING THE COMPOSITE MODULE IFYVLBCM: Composite module IFYVLBCM is built using the following commands:

```
FILEDEF SYSLIB DISK VFORTLIB TXTLIB fm
LKED LKEDLBCM (NOTERM XREF LIBE VFLODLIB
```

The LKED command refers to a file whose file name is LKEDLBCM and whose file type is TEXT. You must create this TEXT file yourself in the following format. All these cards contain linkage editor control statements, so column 1 must be blank.

```
INCLUDE SYSLIB(IFYCLBC0)
INCLUDE SYSLIB(IFYxxxxx)
.
.
.
ORDER IFYCLBC0
ENTRY IFYLBCOM
NAME  IFYVLBCM(R)
```

The LKED command creates the load module IFYVLBCM in the LOADLIB called VFLODLIB; a previous copy of the load module, if any, is replaced. The inclusion of any of the optional modules in the composite module IFYVLBCM is controlled by the linkage editor INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module that is to be in the composite module. Except for the module IFYCLBC0, no INCLUDE statements should be provided for the modules listed above as "Required."

Composite Module IFYVRENC

REQUIRED MODULES

Module	Size	Default Set	Function
IFYCREN	FC	X	Internal linkage module
IFYVGMFM	1A9	X	GETMAIN/FREEMAIN
IFYVSIOS	22F8	X	Sequential I/O services
Total	259D	3	

OPTIONAL MODULES

Module	Size	Default Set	Function
IFYVBLNT	1EC	X	Implied DO in I/O
IFYVCLOP	230	X	CLOSE statement
IFYVCOMH	1502	X	Formatted I/O
IFYVCONI	2DC	X	Input floating-pt conversion
IFYVCONO	754	X	Output floating-pt conversion
IFYVCVTH	114C	X	Data conversion
IFYDDCMP	270		Dynamic common
IFYVEMGN	F00	X	Error message generator
IFYVERRE	21C	X	Error summary
IFYVDIOS	1744		Direct access I/O services
IFYVIIOS	27C		Internal file services
IFYVINQP	964		INQUIRE statement
IFYVIOCP	296		BACKSPACE, REWIND, ENDFILE
IFYVIOFP	6A8	X	Formatted I/O
IFYVIOLP	11E8	X	List-directed I/O
IFYVIOUP	ACA	X	Unformatted I/O
IFYVKIOS	2A70		Keyed access I/O services
IFYVLINP	234		Link to reentrant CSECT
IFYVMSKL	467F	X	Message skeletons
IFYVOPEP	688	X	OPEN statement
IFYVSTAE	82C		ABEND processor
IFYVTEN	2C0	X	Powers of ten table
IFYVTRCH	88C	X	Traceback generator
IFYCVIOS	1C04		Nonkeyed VSAM I/O services

BUILDING THE COMPOSITE MODULE IFYVRENC: Composite module IFYVRENC may be built and placed into a LOADLIB using the following commands:

```
FILEDEF SYSLIB DISK VFORTLIB TXTLIB A
LKED LKEDRENC (NOTERM XREF LIBE VFLODLIB
```

The LKED command refers to a file whose file name is LKEDRENC and whose file type is TEXT. This TEXT file must have the following format:

```
INCLUDE SYSLIB(IFYCREN)
INCLUDE SYSLIB(IFYxxxxx)
.
.
.
ORDER IFYCREN
ENTRY IFYCREN
NAME  IFYVRENC(R)
```

The LKED command creates the load module IFYVRENC in the LOADLIB called VFLODLIB; any previous copy of the load module is replaced. The inclusion of any of the optional reentrant modules in composite module IFYVRENC is controlled by the linkage editor INCLUDE statement, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module to be included. A separate INCLUDE statement is required for each optional module that is to be in the composite module. Except for the module IFYCREN, no INCLUDE statements should be provided for the modules listed above as "Required."

IFYVRENC AS A DISCONTIGUOUS SHARED SEGMENT

Composite module IFYVRENC may be built and placed into a discontiguous shared segment. The virtual storage address selected for the DCSS must be greater than the virtual machine size of anyone who accesses it. In order to accommodate different virtual machine sizes, a facility is available to save multiple copies of composite module IFYVRENC, each with a different name and virtual storage address.

To install IFYVRENC as a DCSS, take the following steps:

1. Your VM/SP systems programmer must update the system name table (DMKSNT).

   To update DMKSNT, an additional NAMESYS macro instruction must be included in the DMKSNT ASSEMBLE file. This macro instruction defines the DCSS that will contain the composite module. If more than one DCSS is to be built to hold copies of the composite module IFYVRENC at different addresses, there must be a NAMESYS macro instruction defining each DCSS. For more information, refer to VM/SP System Programmer's Guide.

   The following example of the NAMESYS macro in DMKSNT ASSEMBLE defines a sample DCSS called "FTNLIB40". The sample numbers given illustrate a possible set of numbers and are not intended as the only location for a DCSS.

```
LIBRTNS NAMESYS SYSNAME=FTNLIB40,
                SYSSIZE=64K,
                SYSHRSG=(64),
                SYSPGCT=16,
                SYSPGNM=(1024-1039),
                VSYSADR=IGNORE,
                SYSVOL=VMSRES,
                SYSSTRT=(072,1)
```

2. Install VS FORTRAN as usual.

3. Place IFYVRENC in the shared segment.

   A DCSS for the composite module is built with the following commands. In order to issue the SAVESYS command shown below, you must be a class E user.

```
CP DEFINE STORAGE mach-size
CP IPL CMS
.
.
.
GLOBAL  TXTLIB VFORTLIB
LOAD    IFYCREN (NOAUTO CLEAR ORIGIN seg-addr
INCLUDE IFYxxxxx (SAME
.
.
.
CP SAVESYS sys-name
ERASE   sys-name MAP A
RENAME  LOAD MAP A   sys-name = =
```

   where

   mach-size specifies a virtual machine size at least as large as the address at which the shared segment resides (seg-addr) plus the length of the composite module.

   seg-addr specifies the virtual storage address at which the shared segment is to reside, as defined in the NAMESYS macro instruction for the system name table.

sys-name specifies the name of the shared segment, as defined in the NAMESYS macro instruction for the system name table and in the VSFCRNAM macro instruction for assembling the module IFYCRNAM.

The SAVESYS command saves the composite module as a DCSS using the specified name; any previous copy of this DCSS is replaced. The inclusion of any of the optional reentrant modules in composite module IFYVRENC is controlled by the CMS INCLUDE command, which refers to IFYxxxxx, where IFYxxxxx is to be replaced by the name of the module to be included. No INCLUDE command should be provided for the modules listed as "Required" in the table above.

4. Use VSFCRNAM to assemble the IFYCRNAM text deck.

   The VSFCRNAM macro builds the CSECT IFYCRNAM, which supplies the shared segment names that will be available and initialized to hold the module IFYVRENC. None of the names supplied can be prefixed by the letters IFY. (However, IFYVRENC can be used as a valid shared segment name.)

   When coding the macro instruction, column 1 must be blank. VSFCRNAM may appear anywhere before column 72 but must precede the operands by at least one blank. The operands may be continued on any number of cards as long as column 72 contains a nonblank character and the data on the following card begins in column 16.

   Place your VSFCRNAM macro instruction in a file whose file name is IFYCRNAM, and whose file type is ASSEMBLE. This file must have the following format:

```
VSFCRNAM SYSNAME=(name1,name2,...)
END
```

   where **name1**, **name2**, and so on are the names of one or more shared segments that contain the IFYVRENC composite module. The names must be listed in increasing order of their virtual storage addresses. No name beginning with "IFY" should be listed.

   Assemble the module IFYCRNAM as follows:

```
GLOBAL   MACLIB VFORTMAC
ASSEMBLE IFYCRNAM
```

5. Insert the IFYCRNAM text deck in VFORTLIB TXTLIB.

   The TEXT file that results from the assembly of the DCSS name list must be placed in the VFORTLIB TXTLIB as follows:

```
TXTLIB DEL VFORTLIB IFYCRNAM
TXTLIB ADD VFORTLIB IFYCRNAM
```

6. Update composite module IFYVLBCM.

   Finally, the DCSS name list module, IFYCRNAM, must be placed in composite module IFYVLBCM. This composite module must be built as described above. Place the following linkage editor INCLUDE statement in the TEXT file LKEDLBCM:

```
INCLUDE SYSLIB(IFYCRNAM)
```

Locating a Usable Copy of the Composite Module

At the initialization of a VS FORTRAN program in load mode, the following search order is used to locate a usable copy of composite module IFYVRENC:

1. If the DCSS name list module, IFYCRNAM, is part of composite module IFYVLBCM, a saved segment is sought, using the names in the DCSS name list in the order that they are listed. The first such segment that does not overlap your virtual machine size is used as the composite module IFYVRENC.

2. If there is a DCSS with a name of "IFYVRENC" that does not overlap your virtual machine, it is used as the composite module IFYVRENC.

3. Finally, the module with a name of "IFYVRENC" is loaded from the LOADLIB made available in the GLOBAL LOADLIB command. (This LOADLIB should be VFLODLIB.) This module is used as composite module IFYVRENC.

APPENDIX A. PROGRAM SUPPORT

The VS FORTRAN Compiler and Library is classified as a licensed program (LP) with S G program services. S G program services provide corrective and preventive service for product defects and a support structure for product problem resolution consisting of the following facilities:

- Central Service, including the IBM Support Center
- Local program support available for a monthly charge under an agreement for IBM licensed programs

For details of these facilities and a list of all the products supported, refer to Field Engineering Programming Systems General Information.

Problem resolution and defect correction are handled as follows: If consulting VS FORTRAN Compiler and Library Diagnosis Guide confirms that a problem still exists, use the manual as a guide in reporting the problem to the IBM Support Center. There it will either be resolved, or accepted as an authorized programming analysis report (APAR) describing a probable product defect. An APAR is resolved by Central Service with either an explanation or a new corrective service program temporary fix (PTF) for the defect. PTFs consist of a replacement text module that is installed in the product to correct a defect. Collections of new PTFs for products are provided to all customers as preventive service program update tapes (PUTs). Problem resolution and defect correction assistance are available through Marketing Product Support.

APPENDIX B. COMPILER AND LIBRARY MODULES

The first list below names all compiler modules, regardless of system type or variable installation conditions. The subsequent lists name modules for particular systems or functions. The sizes of the modules can be found in VS FORTRAN Language and Library Reference.

COMPILER MODULES

Module Name	Module Name	Module Name	Module Name
IFX0ABNT	IFX1CONV	IFX1PARM	IFX3VPBZ
IFX0CMS	IFX1CONI	IFX1PRNS	IFX3VPFP
IFX0CNTL	IFX1CPLX	IFX1PROG	IFX3VPFT
IFX0DCTL	IFX1CSST	IFX1PTEX	IFX3VPGK
IFX0DSYI	IFX1DATA	IFX1P1ER	IFX3VPLP
IFX0DTTM	IFX1DBAT	IFX1RELS	IFX3VPPR
IFX0EMSG	IFX1DBUG	IFX1RPLC	IFX3VQBM
IFX0ESK0	IFX1DEFF	IFX1RTRN	IFX3VQBY
IFX0ESK1	IFX1DICP	IFX1SAVE	IFX3VQBN
IFX0ESK2	IFX1DIMN	IFX1STFC	IFX3VQBS
IFX0ESK3	IFX1DODO	IFX1STOR	IFX3VQBR
IFX0ESK4	IFX1ELSF	IFX1STPS	IFX3VQCA
IFX0INOT	IFX1ELSE	IFX1STRE	IFX3VQCB
IFX0MAPP	IFX1ENDO	IFX1SUBR	IFX3VQCX
IFX0OBJD	IFX1ENDD	IFX1SUBC	IFX3VQCZ
IFX0OPTN	IFX1ENIF	IFX1SUBP	IFX3VQCC
IFX0PACH	IFX1ENTY	IFX1SUBS	IFX3VQFF
IFX0PALC	IFX1EQUV	IFX1TEXT	IFX3VQFN
IFX0PREM	IFX1EQUZ	IFX1TOKN	IFX3VQFX
IFX0STOR	IFX1EXPN	IFX1TRDB	IFX3VQFA
IFX0SYIF	IFX1EXTR	IFX1TRON	IFX3VQFU
IFX0TRCE	IFX1FNLU	IFX1TROF	IFX3VQFV
IFX0TRCM	IFX1FORM	IFX1TYPE	IFX3VQGX
IFX0TRCD	IFX1FUNC	IFX1UNRY	IFX3VQGA
IFX0TRP1	IFX1FWCN	IFX1USAG	IFX3VQGQ
IFX0TRP2	IFX1GENR	IFX1VTEN	IFX3VQLA
IFX0TRP4	IFX1GOTO	IFX2BKCN	IFX3VQLF
IFX1AROP	IFX1GOTC	IFX2CNTL	IFX3VQLR
IFX1ARTH	IFX1GOTA	IFX2COMN	IFX3VQLS
IFX1ASGN	IFX1IFAR	IFX2DISQ	IFX3VQLB
IFX1ASSN	IFX1IFLG	IFX2DYCM	IFX3VQLI
IFX1BLDA	IFX1IFTH	IFX2EQUV	IFX3VQLT
IFX1CALL	IFX1IMPL	IFX2NAML	IFX3VQLJ
IFX1CATN	IFX1INCL	IFX2PACH	IFX3VQLC
IFX1CCNV	IFX1INIT	IFX2SORT	IFX3VQLK
IFX1CHST	IFX1INQR	IFX2STAL	IFX3VQLW
IFX1CKEQ	IFX1INTR	IFX2USAG	IFX3VQLV
IFX1CKLB	IFX1IOLS	IFX2XREF	IFX3VQMF
IFX1CKMD	IFX1IOMN	IFX3CNTL	IFX3VQPF
IFX1CLAS	IFX1IOST	IFX3PACH	IFX3VQPG
IFX1CLSE	IFX1LABU	IFX3PSTR	IFX3VQPT
IFX1CMXP	IFX1LOGL	IFX3QCTL	IFX3VQRX
IFX1CNTL	IFX1NAML	IFX3TRP3	IFX3VQRA
IFX1COMN	IFX1OPEN	IFX3VPBD	IFX3VQRB
IFX1CONT	IFX1PACH	IFX3VPBT	IFX3VQRF

Module Name	Module Name
IFX3VQRZ	IFX4CDE8
IFX3VQRE	IFX4CDE9
IFX3VQRW	IFX4CD10
IFX3VQSC	IFX4CD11
IFX3VQSX	IFX4CD12
IFX3VQSR	IFX4CD13
IFX3VQSM	IFX4CD14
IFX3VQSE	IFX4CD15
IFX3VQSI	IFX4CD16
IFX3VQTD	IFX4CD17
IFX3VQTE	IFX4CD18
IFX3VQTT	IFX4CD19
IFX3VQTM	IFX4CD20
IFX3VQTS	IFX4CD21
IFX3VQTX	IFX4CD22
IFX3VQTF	IFX4CD23
IFX3VQVS	IFX4CD24
IFX3VQWT	IFX4CD25
IFX3VQXM	IFX4CD26
IFX3VQXZ	IFX4CD27
IFX3VRAS	IFX4CD28
IFX3VRBK	IFX4CD29
IFX3VRBP	IFX4CD30
IFX3VRFP	IFX4CD31
IFX3VRFL	IFX4CG01
IFX3VRFR	IFX4CNTL
IFX3VRGB	IFX4DBUG
IFX3VRGR	IFX4NAML
IFX3VRGS	IFX4PACH
IFX3VRLL	IFX4RENT
IFX3VRRG	
IFX3VRRL	
IFX3VRSS	
IFX3VRSX	
IFX3VRSL	
IFX3VRTB	
IFX3VRTF	
IFX3VSBS	
IFX3VSBT	
IFX4ATAB	
IFX4ATAC	
IFX4ATEN	
IFX4ATEP	
IFX4ATPK	
IFX4ATPR	
IFX4ATRN	
IFX4AUEN	
IFX4AVFN	
IFX4CDE1	
IFX4CDE2	
IFX4CDE3	
IFX4CDE4	
IFX4CDE5	
IFX4CDE6	
IFX4CDE7	

MODULES FOR SPECIFIC SYSTEMS

VSE Systems Only	VM Systems Only	MVS and VM Only
IFX0DCTL	IFX0CMS	IFX0CNTL
IFX0DSYI		IFX0SYIF

LIBRARY MODULES

The first list below names all library modules, regardless of system type or variable installation conditions. The subsequent lists name modules for particular systems or functions. The sizes of the modules can be found in VS FORTRAN Language and Library Reference.

Module Name	Module Name	Module Name
IFYBLOGL	IFYDREN	IFYLSIN
IFYBTSHS	IFYDSIOS	IFYLSQRT
IFYCCMPR	IFYDSPAN	IFYLTANH
IFYCITFN	IFYDSPAP	IFYLTNCT
IFYCLABS	IFYDSPA1	IFYLXCMP
IFYCLAD	IFYDSPIE	IFYNAMEL
IFYCLAM	IFYDVIO$	IFYNAMEP
IFYCLBC0	IFYDVIOS	IFYNAME1
IFYCLBC1	IFYFABS	IFYOPSYP
IFYCLEXP	IFYFAINT	IFYOPSYS
IFYCLLOG	IFYFCDCD	IFYOPSY1
IFYCLSCN	IFYFCDXI	IFYQASCN
IFYCLSQT	IFYFCONJ	IFYQATN2
IFYCMOVE	IFYFCQCQ	IFYQERF
IFYCNCAT	IFYFCQXI	IFYQERF2
IFYCQABS	IFYFCXPC	IFYQSCN
IFYCQEXP	IFYFCXPI	IFYQSCNH
IFYCQLOG	IFYFDIM	IFYQSQRT
IFYCQRIT	IFYFDXPD	IFYQTANH
IFYCQSCN	IFYFDXPI	IFYQTNCT
IFYCQSQT	IFYFIFIX	IFYSASCN
IFYCREN	IFYFIMAG	IFYSATN2
IFYCSABS	IFYFIXPI	IFYSCOS
IFYCSAD	IFYFMAXD	IFYSDUMP
IFYCSAM	IFYFMAXI	IFYDUMQ
IFYCSEXP	IFYFMAXR	IFYDUM1
IFYCSLOG	IFYFMODI	IFYSERF
IFYCSSCN	IFYFMODR	IFYSEXP
IFYCSSQT	IFYFNINT	IFYSGAMA
IFYCVIO$	IFYFQXPI	IFYSLGC
IFYCVIOS	IFYFQXPQ	IFYSLGN
IFYDBDFT	IFYFRXPI	IFYSSCNH
IFYDCOM2	IFYFRXPR	IFYSSIN
IFYDDCMN	IFYFSIGN	IFYSSQRT
IFYDDCMP	IFYIBCOM	IFYSTANH
IFYDDCM1	IFYBCOP	IFYSTNCT
IFYDDIO$	IFYBCO1	IFYTFORT
IFYDDIOS	IFYINDEX	IFYUATBL
IFYDFNTH	IFYLASCN	IFYUOPT
IFYDGMFN	IFYLATN2	IFYVAREN
IFYDIOCP	IFYLCOS	IFYVASUB
IFYDIOCS	IFYLDFIO	IFYVASYN
IFYDIOC1	IFYDFIP	IFYVASYP
IFYDLBC0	IFYDFI1	IFYVASY1
IFYDLBC1	IFYLERF	IFYVBLN$
IFYDLCI0	IFYLEXP	IFYVBLNT
IFYDKIO$	IFYLGAMA	IFYVBREB
IFYDKIOS	IFYLLGC	IFYVCIAD
IFYDLOAD	IFYLLGN	IFYVCIA4
IFYDPROD	IFYLSCNH	IFYVCLOP

Module Name
IFYVCLOS
IFYCL01
IFYCLSI
IFYVCMSS
IFYCNI$
IFYCNO$
IFYCOM$
IFYVCOMH
IFYVCOM2
IFYVCONI
IFYVCONO
IFYVCVT$
IFYVCVTH
IFYVDBUG
IFYVDBUP
IFYVDBU1
IFYVDIO$
IFYVDCMN
IFYVDIOS
IFYVDUMP
IFYVDUMQ
IFYVDUM1
IFYVDVCH
IFYVEMG$
IFYVEMGN
IFYVERE$
IFYVERRE
IFYVERS$
IFYVEXIT
IFYVFNTH
IFYVGMFN
IFYVIIO$
IFYVIIOS
IFYVINQP
IFYVINQR
IFYVINQ1
IFYVINTE
IFYVIOCP
IFYVIOCT
IFYVIOC1
IFYVIOD0
IFYVIOD1
IFYVIOFM
IFYVIOFP
IFYVIOF1
IFYVIOI0
IFYVIOI1
IFYVIOK0
IFYVIOK1
IFYVIOLD
IFYVIOLP
IFYVIOL1
IFYVIONL
IFYVIONP
IFYVION1

Module Name
IFYVIOUF
IFYVIOUP
IFYVIOU1
IFYVKIO$
IFYVKIOS
IFYVLBC0
IFYVLBC1
IFYVLCI0
IFYVLCI1
IFYVLINK
IFYVLINP
IFYVLIN1
IFYVLOAD
IFYVLOC$
IFYVLOCA
IFYVMOPP
IFYVMOPT
IFYVMOP1
IFYVMSKL
IFYVOPEN
IFYVOPEP
IFYVOPE1
IFYVOVER
IFYVPARM
IFYVPOS$
IFYVPOSA
IFYVPOST
IFYVREN
IFYVSCOM
IFYVSCOP
IFYVSC01
IFYVSERH
IFYVSFIO
IFYVSFST
IFYVSIOS
IFYVSPAN
IFYVSPAP
IFYVSPA1
IFYVSPIE
IFYVSTA$
IFYVSTAE
IFYVTEN
IFYVTRC$
IFYVTRCH
IFYVVIO$
IFYVVIOS
IFYVXMSK
IFYWDXPD
IFYWLCOS
IFYWLEXP
IFYWLSIN
IFYWRXPR
IFYWSEXP
IFYWTNCT
IFY3COMH

Module Name
IFY3CVTH
IFY3CONI
IFY3CONO
IFY3DIOS
IFY3ERRE
IFY3ERRM
IFY3IIOS
IFY3LOCA
IFY3MOPT
IFY3RENT
IFY3SIOS
IFY3TRCH
IFY3VIOS

Note: IFYUATBL is supplied for VSE systems and generated on OS and CMS during installation.

MODULES FOR SPECIFIC SYSTEMS

MVS Systems Only	VSE Systems Only	VM Systems Only	MVS and VM Only
IFYVAREN	IFYDCOM2	IFYLBC0	IFYVCIAD
IFYVASUB	IFYDDIO$	IFYLBC1	IFYVCIA4
IFYVBREN	IFYDDIOS	IFYCREN	IFYVCOM2
IFYVLBC0	IFYDFNTH	IFYCRNAM	IFYVDIO$
IFYVLBC1	IFYDGMFM	IFYCVIO$	IFYVDIOS
IFYVREN	IFYDKIO$	IFYCVIOS	IFYVFNTH
IFYVVIO$	IFYDKIOS	IFYCMSS	IFYVKIO$
IFYVVIOS	IFYDLBC0		IFYVKIOS
IFY3COMH	IFYDLBC1		IFYVLCI0
IFY3CVTH	IFYDLCI0		IFYVLINP
IFY3CONI	IFYDLOAD		IFYVLIN1
IFY3CONO	IFYDREN		IFYVLOAD
IFY3DIOS	IFYDSIOS		IFYVSFIO
IFY3ERRE	IFYDVIO$		IFYVSFST
IFY3ERRM	IFYDVIOS		IFYVSIOS
IFY3IIOS	IFYOPSYP		IFYVSPIE
IFY3RENT	IFYOPSYS		IFYVSTA$
IFY3SIOS	IFYOPSY1		IFYVSTAE
IFY3VIOS			IFY3MOPT
IFY3TRCH			

REENTRANT LIBRARY MODULES

MVS Module Names	VSE Module Names	VM Module Names
IFYVAREN	IFYDDIOS	IFYCREN
IFYVASUB	IFYDGMFM	IFYCVIOS
IFYVBLNT	IFYDKIOS	IFYVCLOP
IFYVBREN	IFYDREN	IFYVCOMH
IFYVCLOP	IFYDSIOS	IFYVCONI
IFYVCOMH	IFYDVIOS	IFYVCONO
IFYVCONI	IFYVCLOP	IFYVCVTH
IFYVCONO	IFYVCOMH	IFYVDIOS
IFYVCVTH	IFYVCONI	IFYVEMGN
IFYVDIOS	IFYVCONO	IFYVERRE
IFYVEMGN	IFYVCVTH	IFYVGMFM
IFYVERRE	IFYVEMGN	IFYVIIOS
IFYVGMFM	IFYVERRE	IFYVINQP
IFYVIIOS	IFYVGMFM	IFYVIOCP
IFYVINQP	IFYVIIOS	IFYVIOFP
IFYVIOCP	IFYVINQP	IFYVIOUP
IFYVIOFP	IFYVIOCP	IFYVKIOS
IFYVIOUP	IFYVIOFP	IFYVLINP
IFYVKIOS	IFYVIOUP	IFYVMSKL
IFYVLINP	IFYVMSKL	IFYVOPEP
IFYVMSKL	IFYVOPEP	IFYVSFIO
IFYVOPEP	IFYVTEN	IFYVSFST
IFYVREN		IFYVSIOS
IFYVSFIO		IFYVSTAE
IFYVSFST		IFYVTEN
IFYVSIOS		
IFYVSTAE		
IFYVTEN		
IFYVVIOS		
IFY3COMH		
IFY3CVTH		
IFY3CONI		
IFY3CONO		
IFY3DIOS		
IFY3ERRE		
IFY3ERRM		
IFY3IIOS		
IFY3LOCA		
IFY3MOPT		
IFY3RENT		
IFY3SIOS		
IFY3TRCH		
IFY3VIOS		

MEMBERS OF VALTLIB, VLNKMLIB, VFLODLIB

VALTLIB Members	VLNKMLIB Members	VFLODLIB Members
IFYWDXPD	IFYCLBC1	IFYCVIOS
IFYWLCOS	IFYDDCM1	IFYDDCMP
IFYWLEXP	IFYDIOC1	IFYDDIOS
IFYWLSIN	IFYDLBC1	IFYDIOCP
IFYWRXPR	IFYDSPA1	IFYDKIOS
IFYWSEXP	IFYIBCO1	IFYDSIOS
IFYWTNCT	IFYLDFI1	IFYDSPAP
	IFYNAME1	IFYIBCOP
	IFYOPSY1	IFYLDFIP
	IFYSDUM1	IFYNAMEP
	IFYVASY1	IFYOPSYP
	IFYVCLO1	IFYSDUMQ
	IFYVDBU1	IFYVASUB
	IFYVDUM1	IFYVASYP
	IFYVINQ1	IFYVBLNT
	IFYVIOC1	IFYVCIAD
	IFYVIOD1	IFYVCIA4
	IFYVIOF1	IFYVCLOP
	IFYVIOI1	IFYVCOMH
	IFYVIOK1	IFYVCONI
	IFYVIOL1	IFYVCONO
	IFYVION1	IFYVCVTH
	IFYVIOU1	IFYVDBUP
	IFYVLBC1	IFYVDIOS
	IFYVLCI1	IFYVDUMQ
	IFYVLIN1	IFYVEMGN
	IFYVMOP1	IFYVERRE
	IFYVOPE1	IFYVIIOS
	IFYVSCO1	IFYVINQP
	IFYVSPA1	IFYVIOCP
		IFYVIOFP
		IFYVIOLP
		IFYVIONP
		IFYVIOUP
		IFYVKIOS
		IFYVLINP
		IFYVLOCA
		IFYVMOPP
		IFYVMSKL
		IFYVOPEP
		IFYVPOSA
		IFYVSCOP
		IFYVSPAP
		IFYVSTAE
		IFYVTEN
		IFYVTRCH
		IFYVVIOS

A

C

D

E

F

G

I

J

Reader's Comment Form

VS FORTRAN Compiler and Library
Installation and Customization
SC26-3987-3

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

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

List TNLs here:

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

Last TNL _____

Previous TNL _____

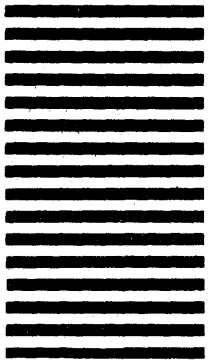
Previous TNL _____

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

Reader's Comment Form

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

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

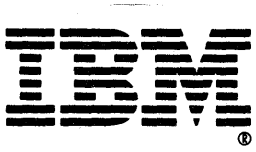


BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
P.O. Box 50020
Programming Publishing
San Jose, California 95150

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



VSE/Advanced Functions Computer and Library Installation and Customization (File No. S370-25) Printed in U.S.A. SC26-3987-3

VS FORTRAN
Compiler and Library
Installation
and Customization

SC26-3987-3

File No. S370-34

Printed in U.S.A.