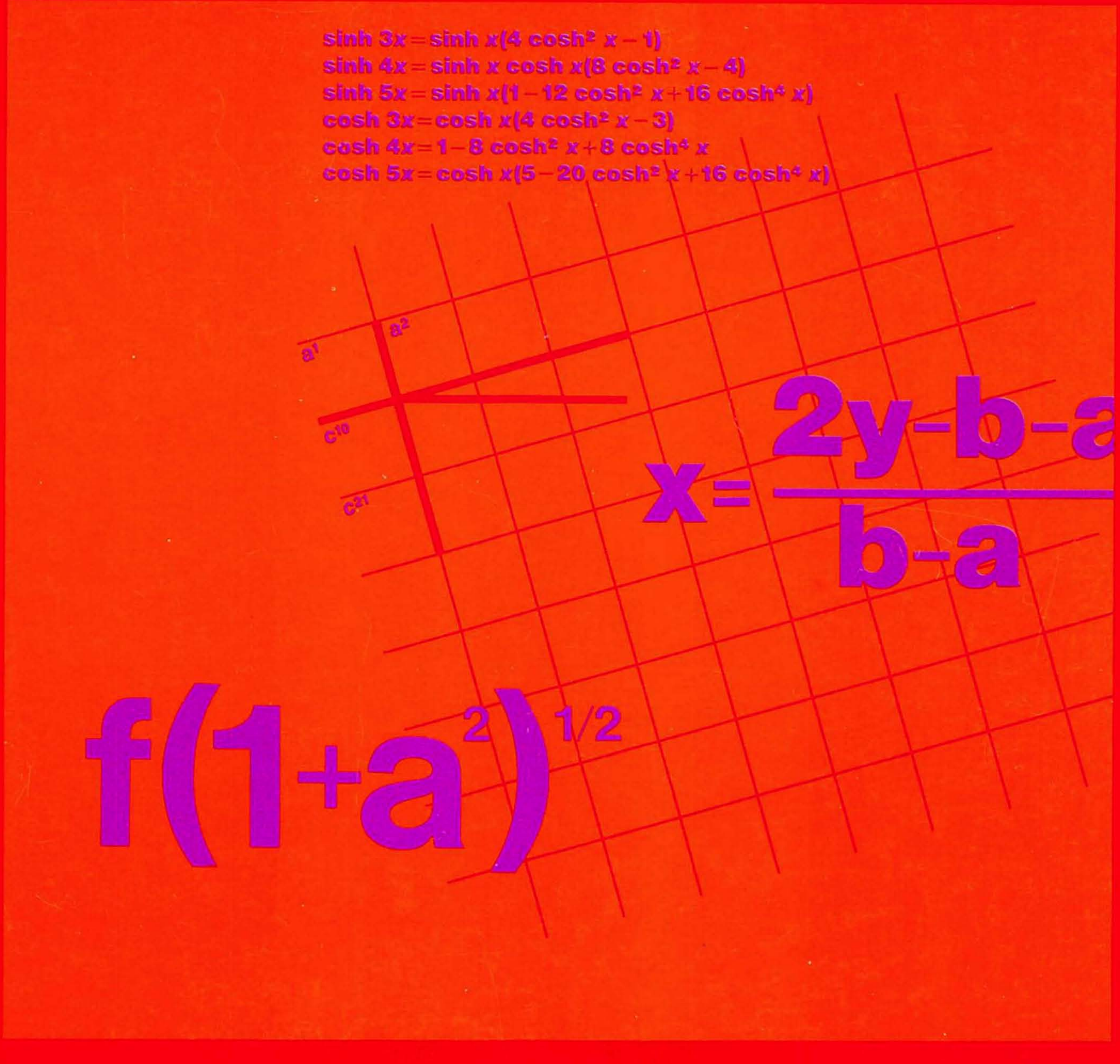
IBM

VS FORTRAN Version 2 Diagnosis Guide

Licensed Program



sinh 3x = sinh x(4 cosh² x − 1)
sinh 4x = sinh x cosh x(8 cosh² x − 4)
sinh 5x = sinh x(1 − 12 cosh² x + 16 cosh⁴ x)
cosh 3x = cosh x(4 cosh² x − 3)
cosh 4x = 1 − 8 cosh² x + 8 cosh⁴ x
cosh 5x = cosh x(5 − 20 cosh² x + 16 cosh⁴ x)
a1
a2
c10
c21
x = 2y-b-a / b-a
f(1+a2)1/2

Order Number LY27-9516-1
File Number S370-37

Program Numbers
5668-805
5668-806
Release 2

IBM

VS FORTRAN Version 2 Diagnosis Guide

Licensed Program

Order Number LY27-9516-1
File Number S370-37

Program Numbers
5668-805
5668-806
Release 2

| **Second Edition (June 1987)**

| This edition replaces and makes obsolete the previous edition, LY27-9516-0.

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

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

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

About This Book

This book helps you diagnose a failure in the VS FORTRAN Version 2 licensed program. The quickest way to resolve such a failure is to first determine if IBM already has a correction for it. If your problem has already been documented, reported and resolved, then all the information you need will be contained in an IBM data base such as the Software Support Facility (SSF). To efficiently determine that, you must describe the failure with a **keyword string**. This book assists you in the systematic development of keyword strings.

This book does not assist in diagnosing and debugging problems in your application programs. For such assistance, see the VS FORTRAN publications referred to below.

SSF is an IBM online data base that contains information about all current Authorized Program Analysis Reports (APARs) and Program Temporary Fixes (PTFs). Your keyword string serves as a **search argument** for the software support data base.

The software support data base can be searched either by yourself, if you have the Info Access Program, or by IBM Support Center personnel. IBM Support Center personnel have direct access to SSF and are responsible for using your keyword string as a search argument to retrieve the records that describe your problem and its correction. They can also help you improve your keyword string, thereby improving your search for a solution and reducing your downtime. Do not hesitate to utilize their expertise.

You may also find a problem similar to yours, and a correction for it, in the Early Warning System (EWS). EWS is a microfiche copy of the data contained in SSF, organized by component identification number and indexed by an APAR symptom code. EWS is published monthly and is available to IBM licensed program customers.

If you choose to perform the data base search yourself and fail to find a similar known problem, you can still use your keyword string to describe the failure to IBM Support Center personnel in an APAR. For additional information on keyword strings and APAR preparation, see *Field Engineering Programming System General Information*, G229-2228.

Diagnostic Preliminaries

To use this book effectively, you should first thoroughly check your VS FORTRAN Version 2 application program and defined functions for user errors. This book assumes that you have:

1. Used all compile-time and run-time options available to you, such as static debug and interactive debug, to assure that your application program is coded correctly.

2. Examined all error messages and error conditions produced by the program, and corrected them when possible.

3. Ensured, as far as possible, that the error is occurring in the VS FORTRAN Version 2 licensed program, not in your application program.

4. Noted the specific sequence of events preceding the error condition.

Summary of Changes

Release 2, June 1987

Major Changes to the Product

- Support for 31-character symbolic names, which can include the underscore (_) character.
- The ability to detect incompatibilities between separately-compiled program units using the new compile-time option ICA (intercompilation analyzer).
- Addition of the NONE keyword for the IMPLICIT statement.
- Enhancement of SDUMP when specified for programs vectorized at LEVEL(2), so that ISNs of vectorized statements and DO-loops appear in the object listing.
- The ability of run-time library error-handling routines to identify vectorized statements when a program interrupt occurs, and the ability under Interactive Debug to set breakpoints at vectorized statements.
- Under MVS, addition of a data set and an optional DD statement to be used during execution for loading library modules and Interactive Debug.
- Under VM, the option of creating a combined VSF2LINK TXTLIB during installation for use in link mode in place of VSF2LINK and VSF2FORT.
- The ability to sample CPU use within a program unit using the new Interactive Debug commands LISTSAMP and ANNOTATE.
- The ability to reset a closed unit to its original (preconnected) state using the new Interactive Debug command RECONNECT.
- The ability to automatically allocate data sets for viewing in the Interactive Debug source window.

- Change in the relative placement in storage of local items, that is, those that are not in common blocks. Programs which depend upon a given arrangement of items in storage may have to be re-coded. This change does not affect items in common blocks nor the association of equivalenced items.

- Change in semantics for OPEN, CLOSE, and INQUIRE statements and the addition of the execution-time options OCSTATUS and NOOCSTATUS.

  - The INQUIRE statement can be used to determine the properties of a file that has never been opened. INQUIRE specifiers SEQUENTIAL, DIRECT, KEYED, FORMATTED, and UNFORMATTED will return values dependent on how the files could potentially be connected.

  - If execution-time option OCSTATUS is in effect:

    - File existence is checked for consistency with the OPEN statement specifiers STATUS = 'OLD' and STATUS = 'NEW'.
    - File deletion occurs when the CLOSE statement specifier STATUS = 'DELETE' is given (on devices which allow deletion).
    - A preconnected file is disconnected when a CLOSE statement is given or when another file is opened on the same unit. It can be reconnected only by an OPEN statement when there is no other file currently connected to that unit.

  - If execution-time option NOOCSTATUS is in effect:

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

  - An OPEN statement cannot be issued for a currently open file except to change the BLANK specifier.

Major Changes to This Manual

Documentation of the above product enhancements has been added.

Release 1.1, September 1986

Major Changes to the Product

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

Major Changes to This Manual

Documentation of the above product enhancements has been added.

Contents

Figures

Figures

Chapter 1. Introduction

This chapter introduces fundamental diagnostic concepts and presents a general diagnosis procedure.

Keyword Strings and Keywords

VS FORTRAN Version 2 product failures can be described through the use of **keyword strings**. A keyword string is a set of **keywords**, in any order, that succinctly describes a product failure. Each keyword is an abbreviation or identifier related to a single aspect of a product or product failure.

Each keyword can be compared to an index of keywords in a data base that associates it with known, recorded problems. A keyword string narrows the data base search to a small number of problems that can be visually scanned for a match. A standard spelling for each keyword ensures that any two people describing the same failure will produce equivalent keyword strings.

Characteristics of Valid Keywords

A valid keyword is

- A single word with no blanks
- Not longer than 15 characters
- Constructed without any special characters (although $ and # are allowed)

Overview of Keyword String Structure and Construction

Figure 2 on page 3 provides a pictorial overview of the VS FORTRAN Version 2 keyword string structure.

As stated earlier, each keyword of a keyword string describes one aspect of a licensed program. For example, the first keyword of a keyword string identifies the licensed program by its **Component Identification Number**. A search of a software support data base with only this first keyword detects all reported problems for the entire licensed program.

Every subsequent keyword added to the keyword string, narrows your data base search, making it more specific and thereby reducing the number of problem descriptions you have to examine. Your choice of particular keywords from any of the keyword categories depends upon the type of failure that has occurred.

The following chart lists the types of keywords that together constitute a full keyword string, and provides both an example of each form of the possible types and a brief explanation:

Type of Keyword	**Form of Keyword**	**Explanation**
Component ID	**566880501** **566880601** **566880602**	Identifies the VS FORTRAN Version 2 product in the software support data base.
Type-of-Failure	**ABENDx** **ABENDUx** **MSGx** **LOOP** **WAIT** **INCORROUT** **PERFM** **DOC**	Identifies the external symptom of a program failure.
Module Name	**ILXxxxxx** **AFBxxxxx** **AFFxxxxx**	Replace the x's with a VS FORTRAN Version 2 module identifier; module identifiers may vary in length from three to five characters.
Modifier		Refer to the modifier keywords listed in the appendixes.
Release Level	**Rxxx**	Replace the x's to identify the VS FORTRAN Version 2 release level installed on your system; do not use decimal points in the keyword.

Figure 1. Types and Forms of Keywords

Examples of keyword strings in this manual generally show x's where you supply information that describes the problem. After you have replaced all x's with relevant information for each keyword category employed, you can then use the keyword string you have constructed either as a search argument in one of IBM's software support data bases or as a description of the product failure when communicating with Support Center personnel.

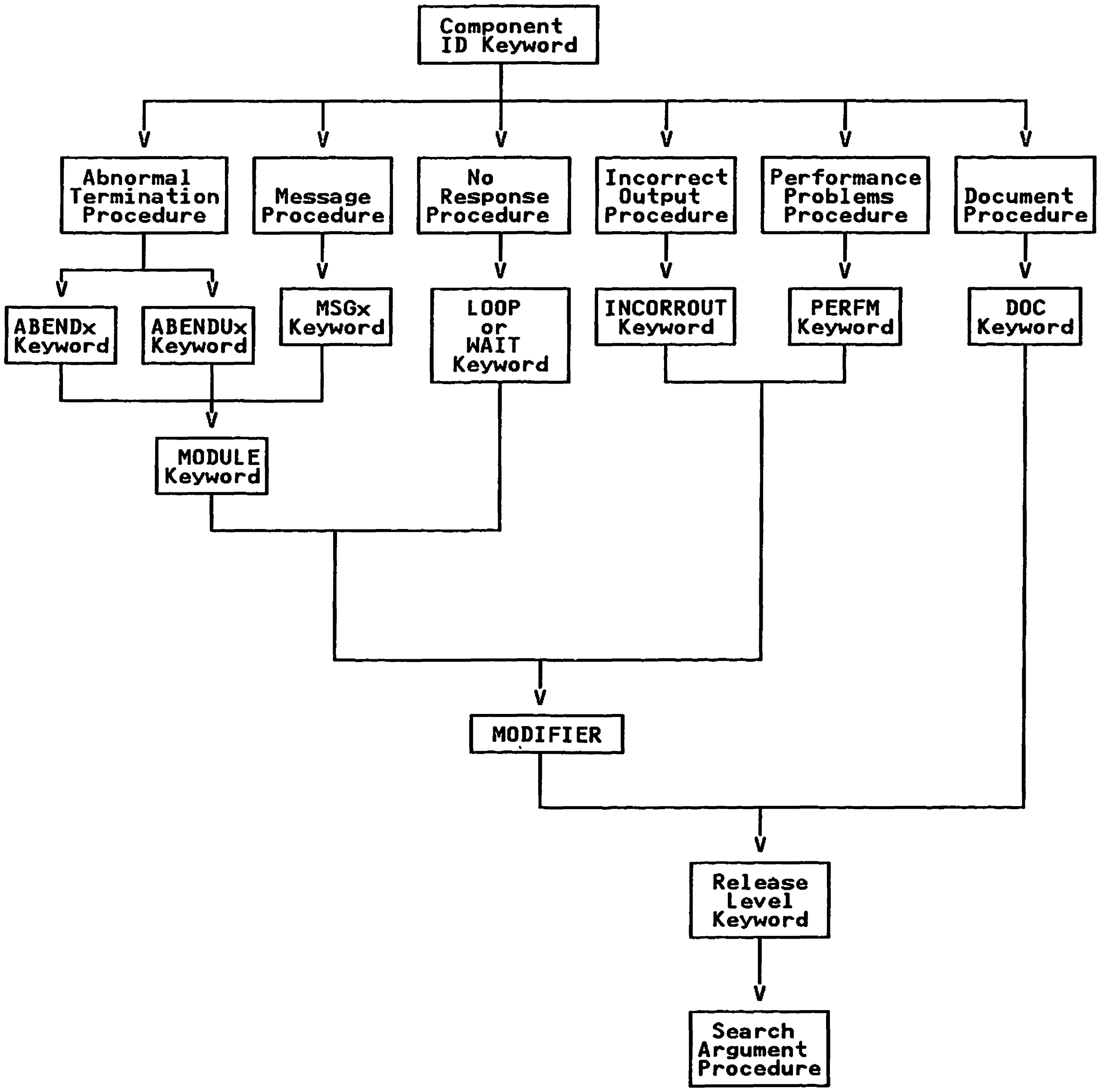


Figure 2. Structure of a VS FORTRAN Version 2 Keyword String

Overview of Keyword String Use

After you have constructed the keyword string, it can be used, as stated earlier, to search an IBM software support data base. If your problem is identified in the software support data base with a keyword string equivalent to the one you have constructed, the search yields a description of the licensed program failure and its resolution, as well as information on its current status.

In some cases, it might be possible to find a problem similar to yours by using a less specific keyword string. To some extent, then, each keyword after the component identification keyword is optional.

In general, however, when contacting the IBM Support Center for assistance, you should identify the program failure with a complete keyword string consisting of appropriate selections from each of the previously specified categories of keywords.

Diagnosis Procedure

Use these procedures to gather the diagnostic information needed to develop keyword search arguments for a software support data base.

1. Correct all problems diagnosed in compiler and library messages, and ensure that any messages previously generated are not associated with the immediate problem. The following table lists the VS FORTRAN Version 2 error message prefixes for the various components:

Component	**Prefix**
Compiler	ILX
Library	AFB
Interactive Debug (IAD)	AFF

Figure 3. Component Error Message Prefixes

2. If your program has been compiled at optimization level 0 with the NORENT option, use interactive debug or static debug statements to identify the problem that occurs at run time. You can use interactive debug for execution-time analysis of your program and its error. You should use the older static debug to isolate the problem and determine whether it is a user error, a compiler-produced code error, or an execution-time library error.

   For more information on interactive debug and static debug, see Appendix A, "Service Aids" on page 31.

3. If compilation fails at an optimization level higher than 0, try to recompile the program at a lower level. If compilation is successful, use the object code produced at that level to bypass the problem until a fix is found and applied. Remember to note the conditions and options in effect when the failure occurred.

4. Determine if the program has been changed since it last compiled or executed successfully. If it has, examine the changes. If the error is occurring in the changed code and cannot be corrected, note the change that caused the error. If possible, retain copies of both the original and the changed programs.

5. After the failure has been explored and identified, consider writing a small test case that re-creates the problem. This test case should help you to:

   - Distinguish between an error in the application program and an error in VS FORTRAN Version 2

   - Pinpoint the problem

   - Choose keywords that best describe the error

6. If the problem occurs during execution, it may be necessary to recompile the source program to produce maximum diagnostic information. Specify the following compile options, in addition to the options originally specified:

   LIST
   GOSTMT
   MAP
   SDUMP
   SOURCE
   SRCFLG
   XREF

   If the failure appears to be associated with VECTOR operations, specify the vector options you used and be sure to include:

   REPORT(LIST)

7. Note the sequence of events that led to the error condition. Also, note which options you were using. This information may be useful in developing a set of keywords, and will be needed if an APAR is required. Refer to Appendix C, "Compile-Time Options" on page 37 and Appendix D, "Execution-Time Options" on page 39 for detailed information.

8. If the user program is abending, direct the object error unit (usually unit 6) to a file, not a terminal, to allow the post-abend dump to be printed.

9. Begin developing the set of keywords, using the procedure in Chapter 2, "Component Identification Keyword" on page 7.

Chapter 2. Component Identification Keyword

The primary keyword of any keyword string is the component identification number (**COMPID**). The following table lists the **COMPIDs** for each component of the VS FORTRAN Version 2 licensed program:

Component	COMPID
Library	566880501
Compiler	566880601
Interactive Debug (IAD)	566880602

Figure 4. Component Identification Numbers (COMPIDs)

Continue the diagnosis procedure with Chapter 3, "Type of Failure Keyword" on page 9.

Chapter 3. Type of Failure Keyword

From the following list, select the symptom that best describes the problem. Then turn to the page referenced for that type of problem. If more than one keyword describes the problem, use the one that appears first in the list.

Symptom	Description	Reference
Abnormal Termination	A program exception has occurred, or the program has terminated abnormally.	"Abnormal Termination Problems" on page 10
Message Problems	A message indicates an internal program error, or seems itself to be an error.	"Message Problems" on page 13
No Response	The program seems to be doing nothing, or is doing something repetitively.	"No Response Problems" on page 16
Documentation Problems	The information in one of the VS FORTRAN Version 2 publications is incorrect or missing.	"Documentation Problems" on page 17
Incorrect Output	The output from the program is missing or invalid.	"Incorrect Output Problems" on page 18
Slow Response	The performance of the program is degraded.	"Performance Problems" on page 19

Figure 5. VS FORTRAN Version 2 Symptom Table

Abnormal Termination Problems

Use the ABENDx keyword procedure when a program exception occurs:

- Within VS FORTRAN Version 2 compiler, library or IAD
- Within a program compiled by VS FORTRAN Version 2
- Whenever a program compiled by VS FORTRAN Version 2 terminates without a message

Do not use this keyword if termination was forced because too much time was spent in a wait state or in an endless loop. For those situations, refer to the procedure for "No Response Problems" on page 16.

Procedure

Determine whether the abnormal termination problem occurs during compile time or during execution time.

- If the problem occurs during the compilation of your program, go to step 1 below.
- If the problem occurs during execution of your program, there is a strong probability that it is a user error. Before continuing, verify that the problem is not the result of incorrect coding.

A. Compilation Abends:

1. Use ABENDU0016 as your type of failure keyword, because all abnormal terminations in the compiler receive this user abend.

2. Recompile the program to produce a trace point 77 dump if the VS FORTRAN Version 2 compiler terminates abnormally.

   For information on how to create a trace point 77 dump, refer to "Compiler Trace Option TRACE" on page 32.

3. Use the trace point 77 dump to determine the module that failed, the location within the module at which the problem occurred, and the type of error exception.

   Figure 7 on page 33 shows a sample trace point 77 output. Areas discussed in the following procedure are in boldface and underscored.

4. Locate the program status word (PSW) in the first line under the heading of the trace point 77 dump. The last byte of the first word in the PSW is an identifier which is the interrupt code.

   Select the exception code associated with this interrupt code in your ABEND keyword from the following table:

Interrupt Code	Interrupt	Exception Code
1	operation	0C1
4	protection	0C4
5	addressing	0C5
6	specification	0C6
7	data	0C7
9	fixed-point divide	0C9
C	exponent overflow	0CC
D	exponent underflow	0CD
F	floating-point divide	0CF
Vector Only:		
6	misaligned data	0E0
C	exponent overflow	0E0
D	exponent underflow	0E0
F	floating-point divide	0E0
19	vector operation exception	09F
1D	unnormalized data exception	0E0

Now replace the xxx of ABENDxxx with the exception code to make a keyword.

For example, if you suspect that the compiler caused the abend and have the trace point 77 shown in Figure 7 on page 33, you would develop the following keyword string:

```
Component Identification:   566880601
Type of Failure:            ABENDU0016
Exception Code Modifier:    ABEND0C1
```

5. Use the trace point 77 dump to develop a modifier keyword from the module name at which the error occurred. This modifier can be found in the dump. Refer to Appendix A.

   Using the example shown in Figure 7, your set of keywords would look like this:

```
Component Identification:   566880601
Type of Failure:            ABENDU0016
Exception Code Modifier:    ABEND0C1
Module Modifier:            ILX2STAL
```

6. Determine at what levels of optimization the failure occurs.

   If it occurs only at some levels, indicate in the keyword string the levels of optimization at which it does occur. Select the appropriate modifier keyword from the lists shown in Appendix C, "Compile-Time Options" on page 37 and Appendix D, "Execution-Time Options" on page 39.

7. Continue with Chapter 4, "Release and Modification Level Keyword" on page 21.

B. Execution Abends:

Execution-time errors that cause the program to terminate abnormally produce message AFB240I and a traceback map.

The MSGAFB240I message has the following format:

```
AFB240I VSTAE:  ABEND CODE IS:  SYSTEM SSSS, USER UUUU,
          SCB/SDWA=HHHHHHHH.
```

1. In the ABENDx keyword, replace the x with the abend code, which is the SSSS coding shown inside the MSGAFB240I message.

   VS FORTRAN Version 2 does not issue user ABENDs during program execution. If you receive a message that indicates a user ABEND, contact your system programmer to determine the problem in your code.

2. Use message MSGAFB240I as a modifier in the keyword string. If you received an additional message besides MSGAFB240I, also include that message as a modifier. For further details on the MSGAFB240I message, refer to "Appendix I" in *VS FORTRAN Version 2: Language and Library Reference*.

3. Study the traceback map to discover which source statement is causing the problem. For a complete description of how to request and interpret a traceback map, refer to the section "Fixing Execution-Time Errors" in *VS FORTRAN Version 2: Programming Guide*. If you are unable to determine which statement is causing the failure, continue with Chapter 4, "Release and Modification Level Keyword" on page 21.

4. Use the FORTRAN, MTF and IAD statement names from Appendix B as modifier keywords to describe the statement you think is causing the error.

5. Determine at what levels of optimization the failure occurs.

   If it occurs only at some levels, indicate in the keyword string the levels of optimization at which it does occur. Select the appropriate modifier keyword from the list shown in Appendix C, "Compile-Time Options" on page 37.

6. Continue with Chapter 4, "Release and Modification Level Keyword" on page 21.

Message Problems

Use the MSGx keyword procedure for any of the following conditions:

- A message is issued indicating an internal program error.
- A message is issued under conditions that should not have caused it to be issued.
- A message contains invalid data or is missing data.

Do not use this procedure if you received a message as the result of an abnormal termination. In that case, see "Abnormal Termination Problems" on page 10.

Message Formats

As shown below, there are minor differences in the formats of the various types of messages.

Compiler Messages

Each VS FORTRAN Version 2 compiler message is identified by a string of eight characters, followed by a string of four characters:

ILXpnnns mmmm

where:

ILX	is the message prefix identifying all VS FORTRAN Version 2 compiler messages.
p	is the number of the compiler phase issuing the message.
nnn	is the message number.
s	indicates the severity level: A – Action required E – Error I – Information W – Warning
mmmm	represents the last four characters of the name of the module issuing the message.

Library Messages

Each VS FORTRAN Version 2 library message is identified by a string of seven characters, followed by a string of up to five characters:

```
AFBnnnx mmmmm
```

where:

AFB	is the message prefix identifying all VS FORTRAN Version 2 library messages.
nnn	is the message number.
x	is either an I for informational messages, or an S for severe error conditions.
mmmmm	represents the rightmost 3-5 characters of the name of the library module issuing the message.

IAD Messages

Each VS FORTRAN Version 2 Interactive Debug message is identified by a string of seven characters:

```
AFFnnns
```

where:

AFF	is the message prefix identifying all VS FORTRAN Version 2 Interactive Debug messages.
nnn	is the number of the message issued.
s	indicates the severity level:

A	–	Action required
E	–	Error
I	–	Information
W	–	Warning

For a list of Interactive Debug messages, see *VS FORTRAN Version 2: Interactive Debug Guide and Reference,* Appendix C.

Procedure

1. Replace the x of MSGx with the complete message identifier. For example, if the message identifier is ILX1YYYS, the MSGx keyword is MSGILX1YYYS.

2. Use the name of the module that caused the message to be issued as the module name keyword.

   a. If the message identifier begins with ILX, the compiler module name is prefixed with ILX and the phase number. For example

      ```
      ILX1060I IFAR
      ```

      results in the following keywords:

      ```
      Type of Failure:           MSGILX1060I
      Module Name:               ILX1IFAR
      ```

   b. If the message identifier begins with AFB, it is a library message. In this case, the module name is prefixed with AFB. For example

      ```
      AFB207I VFNTH
      ```

      results in the following keywords:

      ```
      Type of Failure:           MSGAFB207I
      Module Name:               AFBVFNTH
      ```

   c. If the message identifier begins with AFF, it is an IAD message. In this case, no module name is available on the message. For example

      ```
      AFF190E
      ```

      results in the following keyword:

      ```
      Type of Failure:           MSGAFF190E
      ```

3. Proceed with Chapter 4, "Release and Modification Level Keyword" on page 21.

No Response Problems

Use the LOOP keyword procedure when a program seems to be doing nothing or is doing something repetitively. However, if the problem looks like a WAIT, it is probably a system problem and you should follow your installation's procedures for resolution.

Procedure

1. If the failure occurs during execution, there is a strong probability that this is a user error. Carefully check your source program to be sure it does not contain an endless loop. Adding a DEBUG or a PRINT statement and recompiling may help to detect such a program error. For instructions on how to use static debug, see Appendix A, "Service Aids" on page 31.

2. If you are running in batch mode and the error is a system abend indicating not enough time, it is possible that inadequate time was allotted to compile the program. Increase the time allotment and recompile. If the problem is still not resolved, a set of keywords describing the failure would look something like this:

   ```
   Component Identification:   566880601
   Type of Failure:            LOOP
   ```

3. Determine at what levels of optimization the failure occurs.

   If it occurs at only some levels, indicate in the keyword string the levels of optimization at which it does occur. Select the appropriate modifier keyword from the lists shown in Appendix C, "Compile-Time Options" on page 37 and Appendix D, "Execution-Time Options" on page 39.

4. Continue with Chapter 4, "Release and Modification Level Keyword" on page 21.

Documentation Problems

Use the DOC keyword procedure when a problem appears to be caused by incorrect or missing information in one of the VS FORTRAN Version 2 publications.

Procedure

1. Locate the page where the problem occurs in the document, and prepare a description of the error and the problem it caused.

2. Decide whether this documentation problem is severe enough to cause lost time for other users.

   If the problem is not severe, fill out the Reader's Comment Form attached to the back of that manual giving your problem description. Be sure to include your name and return address so IBM can respond to your comments.

   If the problem is likely to cause lost time for other users, continue creating your keyword string to determine whether IBM has a record of the problem. Should it be a new problem, you will be asked to submit a severity 3 or 4 APAR.

3. Use the order number on the cover of the document, along with the DOC keyword as the type of failure keyword, but omit the hyphens. For example, instead of SC26-4222-1 (the *VS FORTRAN Version 2: Programming Guide*), enter SC2642221. Your set of keywords would look like this:

   ```
   Component Identification:   566880601
   Type of Failure:            DOC SC2642221
   ```

4. Continue the diagnostic procedure with Chapter 4, "Release and Modification Level Keyword" on page 21. If, after searching the IBM software support data base, you do not find a matching problem description, **return here to continue.**

5. Before assuming your search argument is not in the data base, you may want to try the search again, using the following format:

   ```
   Component Identification:   566880601
   Type of Failure:            DOC SC264222**
   ```

   This method (using the asterisks) allows you to search for all problems reported for that document number rather than a specific release number.

Incorrect Output Problems

Use the INCORROUT keyword procedure when output appears to be incorrect or missing, but the program otherwise terminates normally.

Procedure

1. The failure occurred during compilation.

   a. If you suspect incorrect or missing output from a compilation that otherwise compiled successfully, select a modifier keyword from the following list to describe the type of error in the output.

Modifier	Type of Incorrect Output
MISSING	Some expected output was missing.
DUPLICATE	Some records or data were duplicated, but not repeated endlessly (in that case, see "No Response Problems" on page 16).
INVALID	The proper amount of output appeared, but it was not what was expected.

   b. Select another modifier keyword from the following list to describe the portion of the output in which the error occurred.

Modifier	Portion of Output in Error
DEBUG	Static debug
DUMP	CDUMP, PDUMP, and SDUMP
MAP	Storage map
OBJECT	Machine-language object program
SOURCE	Source listing
SRCFLG	Source listing
STAT	Statistics and error listing
TRMFLG	Terminal screen error listing
VECREP	Vector option for report
XREF	Cross-reference listing

   For example, if you think the compiler has produced an incorrect cross-reference listing, your keyword string would look like this so far:

   ```
   Component Identification:    566880601
   Type of Failure:             INCORROUT
   Modifiers:                   INVALID XREF
   ```

   Continue the diagnostic procedure with Chapter 4, "Release and Modification Level Keyword" on page 21.

2. The failure occurred during execution.

   a. There is a strong probability that this is a user error. Check your VS FORTRAN Version 2 source program to be sure it is free of program errors. Adding a DEBUG or PRINT statement and recompiling can help you to detect such an error. See Appendix A, "Service Aids" on page 31 for instructions on how to use static debug statements.

   b. If you suspect incorrect or missing output from a program that otherwise executed successfully, select a modifier from the following list to describe the type of error in the output.

Modifier	**Type of Incorrect Output**
MISSING	Some expected output was missing.
DUPLICATE	Some data or records were duplicated, but not repeated endlessly (in that case, see "No Response Problems" on page 16).
INVALID	The proper amount of output appeared, but it was incorrect.

   c. Continue the diagnostic procedure with Chapter 4, "Release and Modification Level Keyword" on page 21.

Performance Problems

Use the PERFM keyword procedure when a performance problem cannot be corrected by system tuning, and performance is below expectations documented in an IBM product publication.

Procedure

1. Record actual performance and expected performance measurements for your system configuration. Note the order number and page of the IBM document that is the source of your performance expectations. You will be asked for this information if you contact the IBM support center; if you prepare materials for an APAR, you should also include this information in the error description.

2. Continue the diagnostic procedure with Chapter 4, "Release and Modification Level Keyword" on page 21.

Chapter 4. Release and Modification Level Keyword

Use the following procedure to identify the release of the VS FORTRAN Version 2 Compiler or Library and the current level of modification.

Procedure

1. Locate the 'Level n.n.n' line at the top of your latest compilation printout, or at the top of your trace point 77 dump. This line begins as follows:

   ```
   *LEVEL 2.2.0 (July 1987)              VS FORTRAN
   ```

   and also contains the date and time of processing and a page number.

2. Locate the release level keyword in the following chart by release number and operating system.

VS FORTRAN Version 2 Release and Mod Level	Operating System MVS	VM
2.2	R202	R820
2.1.1	R103	R811
2.1	R102	R210

Figure 6. Release and Modification Level

3. You now have the information necessary for an effective search of known problems documented in a software support data base. Continue with Chapter 5, "Using the Keyword String as a Search Argument" on page 23.

Chapter 5. Using the Keyword String as a Search Argument

Now that you have developed a keyword string to describe your program failure, this procedure explains how to use it as a search argument in an IBM software support data base.

Each keyword describes one aspect of a program failure. The component identification number, when used with the type-of-failure keyword as a search argument, should detect all APAR and PTF descriptions for VS FORTRAN Version 2 with that type of failure. The more precisely you describe the program failure by using additional keywords, the more selective is the resulting search, yielding fewer problem descriptions for you to review.

Follow these rules when using a set of keywords to search the IBM software support data base:

- Use only the keywords given in this manual.
- Spell keywords as they are spelled in this manual.
- Include all appropriate keywords in any discussion with IBM or in any APAR description.

  Given the following list:

  ```
  Component Identification:    566880601
  Type of Failure:             ABENDOC1
  Module Name:                 ILX2CNTL
  Release Level:               R202
  ```

  your keyword string would be:

  ```
  566880601 ABENDOC1 ILX2CNTL R202
  ```

Search the IBM software support data base, using the keywords you have developed. You can do the search yourself if you have any of the available search tools (such as EWS), or you can call the IBM Support Center. Compare each matching PTF or APAR problem description with your current failure symptoms.

- If you find an appropriate PTF or APAR problem description, correct the problem by applying the fix described.
- If you do not find an appropriate PTF or APAR problem description, vary the search argument by following the suggestions provided under "Techniques for Varying the Search Argument" below.

If you do not find a matching problem description even after you have tried varying the search argument, turn to Chapter 6, "Preparing an APAR" on page 27.

Techniques for Varying the Search Argument

Listed below are several techniques to vary your search argument and help you find a problem similar to your own.

1. If you used a complete set of keywords (as described in this manual) and were unable to find any problem descriptions to examine, drop one or more of the following keywords to broaden the search:

   Release level keyword
   Module keyword
   Modifier keyword

2. If you searched with less than a complete set of keywords and found too many problem descriptions to examine, continue to add keywords to make the search argument more specific. As the search argument becomes more specific, the number of problem descriptions will decrease.

3. Substitute 566880501 for the component identification keyword if the error has occurred while running within a module of the VS FORTRAN Version 2 library.

   The VS FORTRAN Version 2 library supports programs compiled with the VS FORTRAN Version 2 compiler, as well as any of the following compilers:

Product Name	**Component ID**
VS FORTRAN Version 1	5748FO300
FORTRAN IV H-Extended Compiler	5734FO301
FORTRAN IV G1 Compiler	5734FO201
FORTRAN IV F Compiler	360NFO479

   If you are using one of these compilers in conjunction with the VS FORTRAN Version 2 library, search the software support data base, using that compiler's component identification keyword along with the VS FORTRAN Version 2 library's identification number.

4. Consider using a synonym as a replacement for a modifier keyword. The problem may have been entered into the data base using a slightly different expression than the now-recommended format.

5. If you received an error message that documents an abend or a program check, and you used the **ABEND** type-of-failure keyword to describe this error, you can also try using the **MSGx** keyword to perform the search.

6. If your type-of-failure keyword is **WAIT**, **LOOP**, or **PERFM**, and if you did not find a matching problem description, try to replace the keyword you used with one of the others. Sometimes a problem that appears to be a performance problem may actually be a WAIT or a LOOP; likewise, a problem that seems to be a WAIT or a LOOP may actually be recorded as a performance problem.

7. If your type-of-failure keyword is **MSGx**, and you received more than one message when you encountered the problem, try replacing the message number in the keyword with a number from one of the other messages.

8. If your type-of-failure keyword is **MSGx**, **PERFM**, or **INCORROUT**, and if the problem occurred immediately after you performed some action that a VS FORTRAN Version 2 publication told you to perform, then the problem may be recorded as a **DOC** type of failure. In this case, try searching with **DOC** as your type-of-failure keyword.

Chapter 6. Preparing an APAR

An APAR should be prepared only after your keyword search proves unsuccessful. Contact IBM for assistance in completing the APAR.

1. Initiating an APAR

   Contact the IBM support center either directly or through an IBM Program Support Representative (PSR) for assistance. Be prepared to provide the following information:

   - Your customer number.
   - Your current release level for either VM/SP CMS or OS/VS2 MVS.
   - The set of keywords you used to search the IBM software support data base.
   - Your processor number (serial and model).
   - A description of how reproducible the error is:
     - Can it be reproduced each time?
     - Can it be reproduced only sometimes?
     - Have you been unable to reproduce it?

     If possible, reproduce the error in the most direct way you can. For example, reduce the number of statements within the failing function to the fewest needed to cause the error to occur.
   - A dump, if the error is a system error or an abnormal termination.

2. Gathering APAR Documentation

   You will be asked to supply various types of information that describe the failure. Any or all of the following items might be requested as documentation to be submitted with the APAR:

   - If your terminal has a hard-copy print device, save the hard-copy log of the events leading up to the failure. If your 3279 terminal has a PRINT key and associated printer, press the PRINT key for each display, and save the hard-copy log.
   - Save the machine-readable version of any dumps.

- Save a machine-readable version of the job control language job stream you submitted, or the commands you used for CMS.

- Save the system output (SYSOUT) associated with the TSO batch job, and the spool files associated with the CMS batch job.

- In TSO, if SMP is used to make all changes to the system, execute the LIST CDS and LIST PTFBY functions of SMP to obtain a list of the current maintenance from the SMP control data set (CDS).

  If any changes are made to the system without using SMP, execute the LISTIDR function of the AMBLIST service aid program to obtain a list of all members with a PTF or local fix, and save the output.

  Execute the program against the:

  - SYS1.LINKLIB data set
  - SYS1.SVCLIB data set
  - SYS1.LPALIB data set
  - Library containing the program that issued the message

- In TSO, save the associated volume; in CMS, save the associated minidisk.

- In TSO, execute the Access Method Services LISTCAT command to:

  - List the contents of the applicable catalog
  - List the catalog entries for the applicable objects and any related objects

- For TSO, refer to *OS/VS Message Library: VS2 Systems Messages*, GC38-1002, for more detail.

- For CMS, refer to *VM/SP Messages and Codes*, SC19-6204, for more detail.

3. Submitting APAR Documentation

   When submitting material for an APAR to IBM, carefully pack and clearly identify any magnetic tapes, printed output, or decks of punched cards that are supplied containing application source programs, job stream data, data sets, or libraries.

   a. Each magnetic tape submitted must have the following information attached and visible:

      - The APAR number assigned by IBM.

      - A list of data sets on the tape (application source program, JCL, data).

- A description of how the tape was made:
  - Exact JCL listing or the list of commands used
  - Recording mode and density
  - Tape labeling
  - Record format and block size used for each data set

b. Each card deck submitted must have the following information attached and visible:

- APAR number assigned by IBM
- Contents of the card deck (application source program, job control statements, or data)

c. Each dump and any other printed materials must show the APAR number.

Appendix A. Service Aids

Source Statements and Options

For more comprehensive information on the following statements and options, see "Debug Statements" in *VS FORTRAN Version 2: Language and Library Reference.*

Static DEBUG Statements

AT Statement: specifies the beginning of a debugging packet.

DEBUG and END DEBUG Statements: delimit the debugging portions of a program.

Debug Option	Description
INIT	displays a variable or an array name in the debug output when the variable or array element is assigned a new value.
SUBCHK	checks the validity of the subscripts used with the named arrays.
SUBTRACE	specifies that the name of this subprogram is to be produced whenever it is entered, and RETURN is printed when the subprogram is exited.
TRACE	specifies portions of a program to be traced.
UNIT	specifies a particular data set for all debugging output.

DISPLAY Statement: displays data within a debugging packet.

TRACE/ON and OFF Statements: specify when to begin and end tracing.

Note: Any FORTRAN statement can be used in conjunction with static debug statements. Refer to step 4 on page 12 for a list of VS FORTRAN Version 2 statements. These statements can only be used at optimization level 0 and with the NORENT option.

Interactive Debug

If you have VS FORTRAN Version 2 Interactive Debug at your installation, you can use it to help determine the statement causing an error, to suspend program execution, continue execution, skip sections of code, correct errors, set up expected input, display output, or debug optimized code (with some restrictions). For more information, see *VS FORTRAN Version 2: Interactive Debug Guide and Reference.*

DUMP and PDUMP Subroutines

DUMP is a service subroutine used to dump storage and terminate object program execution. PDUMP is a service subroutine used to dump storage and continue object program execution. The DUMP and PDUMP subroutines are invoked by the CALL statement.

CDUMP and CPDUMP Subroutines

CDUMP is a service subroutine used to dump storage in character format and terminate object program execution. CPDUMP is a service subroutine used to dump storage in character format and continue object program execution. The CDUMP and CPDUMP subroutines are invoked by the CALL statement.

SDUMP Subroutine

The Symbolic Dump (SDUMP) subroutine displays a symbolic dump of all the variables in one or more program units. The variables are displayed in a format determined by the variable types and sizes declared, or defaulted, in the source program. The SDUMP subroutine is invoked by the CALL statement.

Compiler Trace Option TRACE

The TRACE compiler option indicates that compiler diagnostic trace information is to be produced. The TRACE option is only valid as part of an @PROCESS statement. The format of the @PROCESS statement using the TRACE option is: @PROCESS beginning in column 1, followed by TRACE and any other options.

Multiple options are separated by one or more blanks. The @PROCESS statement must be the first statement in the source program. Further details on the @PROCESS statement are found in *VS FORTRAN Version 2: Programming Guide.*

When a program interrupt occurs within the FORTRAN Compiler and the TRACE option is on, a special trace point (known as trace point 77) is activated in the interrupt handler. It provides a printout of the PSW, all general registers, save areas for all active modules, and formatted printouts of all available tables. Other trace points are available but are used only at the request of the IBM product specialist. See the following page for sample trace point 77 output.

```
PSW AND GENERAL REGISTERS.   PHASE 2.   TRACE POINT 77.   PROGRAM  CHECK.
PSW:     FFE40001  4208B53A

GR 0-7:000F1B99  0008F978  00000000  00000000  0011F9E0  00000000  00090463  0008F464

GR 8-F:0008E465  0008D466  0008C467  000F1C00  0008B468  000F2180  4208B538  00000000

AREA AROUND INTERRUPT.

(0008B510) B2BC5040 D3685840 B2C05040 D36C5840  B2C45040 D3709101 B07B4770 C0D458F0

(0008B530) 76D04110 751405EF 00000000 58F076D0  4110751C 05EF5840 B1BC5040 B4F89640

(0008B550) B4C65840 B1BC5040 D198D201 B358762E  1F445040 D18C4340 76FF4240 D1CA4140

THE PROGRAM CHECK OCCURRED AT DISPLACEMENT (00D2) IN ILX2STAL

SAVE AREA CONTENTS: UNUSED, BKPTR, FWDPTR, GPR(14-12), CALLER ID, WORKING STORAGE.

SAVE AREA FOR ILX2STAL (000F2180).  CALLED BY ILX2CNTL AT DISPLACEMENT (0496).

(00000000) 00000000 000F2120 000F24F8 0036C3EA  0202724C 00000004 003FC808 00000000

(00000020) 00000000 0011F9E0 00000000 00090463  0008F464 0008E465 0008D466 0008C467

(00000040) 000F1C00 0008B468 C3D5E3D3 0002C54F  000F2178 001075F0 0002B527 000F1C00
                                 .
                                 .
                                 .

(00000340) 00000026 00000000 00000000 00000000  0011F680 00000000 00000000 00000000

(00000360) 0011F878 00000000 0011F908 00000000  0011F9E0 00000000

SAVE AREA FOR ILX2CNTL (000F2120).  CALLED BY ILX0CNTL AT DISPLACEMENT (0706).

(00000000) 00000000 000F1930 000F2180 4202C48E  0008B468 000F1B99 000F216C 000F217D

(00000020) 00000000 00000000 00000000 00000000  00000000 00000001 000F1930 00000000

(00000040) 000F1C00 0002BFF8 C3D5E3D3 0002C54F  000F2178 00000000 0000002C 00000000
```

Figure 7. Sample Trace Point 77 Output

Appendix B. Keyword Modifiers

This appendix contains lists of keyword modifiers for the source program, for MTF, and for Interactive Debug.

Source Program Modifiers

ASSIGN
AT
BACKSPACE
BLOCKDATA
CALL
CHARACTER
CLOSE
COMMON
COMPLEX
COMPLEX4
CONTINUE
DATA
DEBUG
DEFINE FILE[1]
DELETE
DIMENSION
DISPLAY
DO
DOUBLE PRECISION
EJECT
ELSE[2]
ELSEIF[2]
END
ENDFILE
ENDIF[2]
ENTRY
EQUIVALENCE
EXTERNAL
FIND[1]
FORMAT
FUNCTION
GENERIC[1]
GOTO[3]
IF[4]
IFTHEN[5]
IMPLICIT
INCLUDE
INQUIRE
INTEGER
INTEGER2
INTRINSIC
LOGICAL
NAMELIST
OPEN
PARAMETER
PAUSE
PRINT
PROGRAM
PUNCH[1]
READ[6]
REAL
REAL4
REAL8
REAL16
RETURN
REWIND
REWRITE
SAVE
STOP
SUBROUTINE
TRACE OFF
TRACE ON
WAIT
WRITE[6]

Notes:

[1] LANGLVL(66) only.
[2] Use IF THEN as keyword.
[3] Includes assigned GOTO, computed GOTO, and unconditional GOTO.
[4] Includes arithmetic IF and logical IF.
[5] Includes block IF, ELSE, ELSE IF, and END IF.
[6] Includes asynchronous, with list-directed I/O, with NAMELIST, with internal files, formatted or unformatted with direct or keyed access, or formatted or unformatted with sequential access.

MTF Modifiers

AUTOTASK (MVS only)
DSPTCH
MTF
NTASKS
SUBTASK
SYNCRO

Interactive Debug Modifiers

ANNOTATE
AT
AUTOLIST
BACKSPACE
BATCH
CLOSE
COLOR
DESCRIBE
ENDDEBUG
ENDFILE
ERROR
FIXUP
GO
HALT
HELP
IF
ISPF
LINEMODE
LIST
LISTBRKS
LISTFREQ
LISTINGS
LISTSAMP
LISTSUBS
LISTTIME
MOVECURS
NEXT
OFF
OFFWN
POSITION
PREVDISP
PROFILE
PURGE
QUALIFY
QUIT
RECONNECT
REFRESH
RESTART
REWIND
SEARCH
SET
STEP
SPLITSCREEN
SYSCMD
TERMIO
TIMER
TRACE
WHEN
WHERE
WINDOW

Appendix C. Compile-Time Options

Use the modifier keywords shown below to specify compiler options in the keyword string.

Option	Modifier Keywords
AUTODBL (value)	AUTODBL
CHARLEN (number)	CHARLEN
CI (number)	CI
DC (name)	DC
DECK or NODECK	DECK NODECK
DIRECTIVE(trigger-constant) or NODIRECTIVE(trigger-constant)	DIRECTIVE NODIRECTIVE
FIPS (S|F) or NOFIPS	FIPSF FIPSS NOFIPS
FLAG (I|W|E|S)	FLAGI FLAGW FLAGE FLAGS
FREE or FIXED	FREE FIXED
GOSTMT or NOGOSTMT	GOSTMT NOGOSTMT
ICA(USE(name-1,name-2,...,name-n) UPDATE(name)|UPD(name) MXREF|NOMXREF CLEN|NOCLEN CVAR|NOCVAR MSG(NEW|NONE|ALL)) or NOICA	ICA UPDATE UPD MXREF NOMXREF CLEN NOCLEN CVAR NOCVAR MSG NOICA
IL(DIM|NODIM)	DIM NODIM NOIL
LANGLVL (66|77)	LANGLVL66 LANGLVL77
LINECOUNT (number)	LINECOUNT
LIST or NOLIST	LIST NOLIST
MAP or NOMAP	MAP NOMAP
NAME (name)	NAME
OBJECT or NOOBJECT	OBJECT NOOBJECT
OPTIMIZE (0|1|2|3) or NOOPTIMIZE[1]	OPT0 OPT1 OPT2 OPT3
RENT or NORENT	RENT NORENT
SDUMP(ISN|SEQ) or NOSDUMP	SDUMP NOSDUMP ISN SEQ
SOURCE or NOSOURCE	SOURCE NOSOURCE
SRCFLG or NOSRCFLG	SRCFLG NOSRCFLG

Option	**Modifier Keywords**
SXM or NOSXM	SXM NOSXM
SYM or NOSYM	SYM NOSYM
TERMINAL or NOTERMINAL	TERM NOTERM
TEST or NOTEST	TEST NOTEST
TRMFLG or NOTRMFLG	TRMFLG NOTRMFLG
VECTOR LEVEL(0|1|2) REPORT(LIST|XLIST|TERM) |NOREPORT INTRINSIC|NOINTRINSIC REDUCTION|NOREDUCTION SIZE(ANY|LOCAL|number) or NOVECTOR	VEC0 VEC1 VEC2 REPORTLIST REPORTXLIST REPORTTERM NOREPORT INTRINSIC NOINTRINSIC REDUCTION NOREDUCTION SIZEANY SIZELOCAL SIZE128 SIZE256 NOVECTOR
XREF or NOXREF	XREF NOXREF

Note:

[1] The NOOPTIMIZE option is the same as OPTIMIZE 0.

Appendix D. Execution-Time Options

Use the modifier keywords shown below to specify execution-time options in the keyword string.

Option	**Modifier Keywords**
ABSDUMP or NOABSDUMP	ABSDUMP NOABSDUMP
AUTOTASK or NOAUTOTASK	AUTOTASK NOAUTOTASK
DEBUG or NODEBUG	DEBUG NODEBUG
DEBUNIT or NODEBUNIT	DEBUNIT NODEBUNIT
IOINIT or NOIOINIT	IOINIT NOIOINIT
OCSTATUS or NOOCSTATUS	OCSTATUS NOOCSTATUS
SPIE or NOSPIE	SPIE NOSPIE
STAE or NOSTAE	STAE NOSTAE
XUFLOW or NOXUFLOW	XUFLOW NOXUFLOW

Reader's Comment Form

VS FORTRAN Version 2: Diagnosis Guide

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

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

Note: Do not use this form to request IBM publications. If you do, your order will be delayed because publications are not stocked at the address printed on the reverse side. Instead, you should direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

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

Chapter/Section ____________

____________ Page No. ____________

Comments:

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

Name ____________ Phone No. (____) ____________

Company ____________

Address ____________

____

Thank you for your cooperation. 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.)

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

LY27-9516-1

Reader's Comment Form

Fold and tape Please do not staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

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

POSTAGE WILL BE PAID BY ADDRESSEE

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

Fold and tape Please do not staple Fold and tape

IBM®

VS FORTRAN Version 2: Diagnosis Guide (File No. S370-37) Printed in U.S.A. LY27-9516-1

VS FORTRAN Version 2
Diagnosis Guide

The VS FORTRAN Version 2 Library

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

LY27-9516-01

Printed in U.S.A.