VS FORTRAN Version 2

Reference Summary

Release 6

SX26-3751-07

VS FORTRAN Version 2

Reference Summary

Release 6

SX26-3751-07

Note!

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

| **Eighth Edition (November 1993)**

| This edition replaces and makes obsolete the previous edition, SX26-3751-06.

| This edition applies to VS FORTRAN Version 2 Release 6, Program Numbers 5668-805, 5668-087, 5667-806, and to any subsequent releases until otherwise indicated in new editions or technical newsletters.

Specific changes for this edition are indicated by a vertical bar to the left of the change. A vertical bar to the left of a figure caption indicates that the figure has changed. Editorial changes that have no technical significance are not noted.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, 4300*, and *9370 Processors Bibliography*, GC20-0001, for the editions that are applicable and current.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Canada Ltd. Laboratory
Information Development, 2G/345/1150/TOR
1150 Eglinton Avenue East, North York
Ontario, Canada M3C 1H7

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

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

Programming Interface Information

This reference is intended to help you create application programs using the VS FORTRAN Version 2 licensed program. This reference documents General-Use Programming Interface and Associated Guidance Information provided by VS FORTRAN Version 2.

General-Use programming interfaces allow the customer to write programs that obtain the services of VS FORTRAN Version 2.

Trademarks and Service Marks

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

IBM

SAA

Systems Application Architecture

3090

Compiler and Library Information

Arithmetic Expressions

Arithmetic Operator	Definition
**	Exponentiation
*	Multiplication
/	Division
+	Addition (or unary plus)
–	Subtraction (or unary minus)

Relational Expressions

Standard Relational Operator	VS Fortran Relational Operator	Definition
.GT.	>	Greater than
.GE.	>=	Greater than or equal to
.LT.	<	Less than
.LE.	<=	Less than or equal to
.EQ.	==	Equal to
.NE.	/= <>	Not equal to

Logical Expressions

Logical Operator	Use	Meaning
.NOT.	.NOT.A	If A is true, then .NOT.A is false; if A is false, then .NOT.A is true.
.AND.	A.AND.B	If A and B are both true, then A.AND.B is true; if either A or B or both are false, then A.AND.B is false.
.OR.	A.OR.B	If either A or B or both are true, then A.OR.B is true; if both A and B are false, then A.OR.B is false.
.XOR.	A.XOR.B	If either A or B is true, the A.OR.B is true; if both A and B are false, then A.OR.B is false.
.EQV.	A.EQV.B	If A and B are both true or both false, then A.EQV.B is true; otherwise it is false.
.NEQV.	A.NEQV.B	If A and B are both true or both false, then A.NEQV.B is false; otherwise it is true.

Hierarchy of Operations

Operations	Hierarchy
Evaluation of functions	1st (highest)
Exponentiation (**)	2nd
Multiplication and division (* and /)	3rd
Unary addition and subtraction (+ or -)	4th
Binary addition and subtraction (+ and -)	5th
Concatenation (//)	6th
Relationals (.GT.,.GE.,.LT.,.LE.,.EQ.,.NE.)	7th
.NOT.	8th
.AND.	9th
.OR.	10th
.EQV. or .NEQV. or .XOR.	11th

Required Order of Statements and Comments

Figure 1. Order of Statements and Comment Lines

<table>
<tr><td rowspan="6">Comment Lines</td><td colspan="3">PROGRAM, FUNCTION, SUBROUTINE, or BLOCK DATA Statement</td></tr>
<tr><td rowspan="5">FORMAT and ENTRY Statements</td><td colspan="2">IMPLICIT NONE Statement</td></tr>
<tr><td rowspan="2">PARAMETER Statements</td><td>IMPLICIT Statements</td></tr>
<tr><td>Other Specification Statements</td></tr>
<tr><td rowspan="2">DATA Statements</td><td>Statement Function Statements</td></tr>
<tr><td>Executable Statements and Parallel Statements</td></tr>
<tr><td colspan="3">END Statement</td></tr>
</table>

Language Statement Categories

In the following statements, extensions to the Fortran standard language are printed in color.

Assignment Statements

- Arithmetic
- Character
- Logical
- ASSIGN

Control Statements

- CALL
- CONTINUE
- DO
- DO WHILE
- END
- END DO
- GO TO
- IF
  - (ELSE,ELSE IF, END IF)
- PAUSE
- RETURN
- STOP

Data Statements

- DATA

Static Debug Statements

- AT
- DEBUG
- DISPLAY
- END DEBUG
- TRACE OFF
- TRACE ON

Input/Output Statements

- BACKSPACE
- CLOSE
- DELETE
- ENDFILE

Input/Output Statements (*continued*)

- FORMAT
- INQUIRE
- OPEN
- PRINT
- READ
- REWIND
- REWRITE
- WAIT
- WRITE

Program Statement

- PROGRAM

Specification Statements

- AUTOMATIC
- COMMON
- DIMENSION
- EQUIVALENCE
- Explicit type:
  - CHARACTER,
  - COMPLEX,
  - DOUBLE PRECISION,
  - INTEGER,
  - LOGICAL, and REAL
  - DOUBLE COMPLEX
  - UNSIGNED
  - BYTE
- EXTERNAL
- IMPLICIT
- INTRINSIC
- NAMELIST
- PARAMETER
- POINTER
- SAVE
- STATIC

Subprogram Statements

BLOCK DATA
ENTRY
FUNCTION
SUBROUTINE

Allocation Statements

ALLOCATE
DEALLOCATE
NULLIFY

Parallel Statements

See page 26.

Statement Syntax

In the following statements, extensions to the Fortran standard language are printed in color.

ALLOCATE Statement

ALLOCATE (*pointee1* [, *pointee2* ...], STAT=*stat*)

Obtains space for a pointee array.

ASSIGN Statement

ASSIGN *stl* TO *i*

Assigns a number (statement label) to an integer variable.

Assignment Statement

a = b

Evaluates the expression to the right of the equal sign, replacing the current value of the variable, array element, character substring, or character variable to the left of the equal sign with the expression's value.

AT Statement

AT *stl*

Identifies the beginning of a debug packet and indicates the point in the program at which debugging statements are to be inserted.

AUTOMATIC Statement

AUTOMATIC *name1* [, *name2* ...]

Names variables and arrays to be in the automatic storage class.

BACKSPACE Statement

BACKSPACE *un*

BACKSPACE

([UNIT=] *un*
[,IOSTAT=*ios*]
[,ERR=*stl*])

Positions a sequentially-accessed file to the beginning of the Fortran record last written or read, or repositions the file to the beginning of the preceding record.

BLOCK DATA Statement

BLOCK DATA [*name*]

Initializes values for variables and array elements in named common blocks.

CALL Statement

CALL *name* [([*arg1* [,*arg2*]...])]

Evaluates actual arguments that are expressions, passes actual arguments that will be associated with dummy arguments defined in the subroutine, and transfers control to the subroutine.

CLOSE Statement

CLOSE
([UNIT=] *un* [,ERR=*stl*]
[,STATUS=*sta*]
[,IOSTAT=*ios*])

Disconnects a unit.

COMMON Statement

COMMON [/[*name1*]/] *list1* [[,]/[*name2*]/*list2*...]

Allows two or more program units to share storage and to specify the names of variables and arrays that are to occupy the area.

CONTINUE Statement

CONTINUE

Labels a position in a program (can designate the end of a DO loop).

Statement Syntax

DATA Statement

DATA *list1* /*clist1*/ [[,]*list2* /*clist2*/...]

Defines initial values of variables, array elements, arrays, and substrings.

DEALLOCATE Statement

DEALLOCATE (*pointee1*[,*pointee2*...] STAT=*stat*)

Releases space used for a pointee array and resets the associated pointer variable to the unassigned state.

DEBUG Statement

DEBUG *option1* [,*option2*...]

Sets the conditions for operation of the debug facility and designates debugging operations that apply to the entire program unit.

DELETE Statement

DELETE *un*

DELETE
([UNIT=] *un*
[,ERR=*stl*]
[,IOSTAT=*ios*])

Removes a record from a file connected for keyed access.

DIMENSION Statement

DIMENSION *a1* (*dim1*) [,*a2* (*dim2*)...]

Specifies the name and dimensions of an array.

DISPLAY Statement

DISPLAY *list*

Displays data in NAMELIST output format.

Statement Syntax

DO Statement

DO [*stl* [,]] *i*=*e1*, *e2* [,*e3*]

Controls the processing of the statements that follow it, up to and including the statement that denotes the end of the DO loop.

DO WHILE Statement

DO [*stl* [,]] WHILE (*m*)

Controls the processing of the statements that follow it, up to and including the terminating statement.

END Statement

END

Terminates a main program or a function, subroutine or block data subprogram.

END DEBUG Statement

END DEBUG

Terminates the last debug packet for the program.

END DO Statement

END DO

Terminates the range of a DO WHILE loop (may be used to terminate the range of a DO loop).

ENDFILE Statement

ENDFILE *un*

ENDFILE
([UNIT=] *un*
[,ERR=*stl*]
[,IOSTAT=*ios*])

Writes an end-of-file record on a sequentially accessed external file.

Statement Syntax

ENTRY Statement

ENTRY *name* [([*arg1* [,*arg2*]...])]

Names a place in a subroutine or function subprogram that can be used in a CALL statement or as a function reference.

EQUIVALENCE Statement

EQUIVALENCE (*list1*) [,(*list2*)...]

Permits the sharing of data storage within a single program unit.

Explicit Type Statement

type name1 [,*name2*...]

*type*******len name1*******len*(*dim*) [,*name2*******len*(*dim*)...]

Specifies the type and length of variables, arrays, and user-supplied functions, specifies the dimensions of an array, and assigns initial data values for variables and arrays.

EXTERNAL Statement

EXTERNAL *name1* [,*name2*...]

Identifies a user-supplied subprogram name and permits such a name to be used as an actual argument.

FORMAT Statement

FORMAT (f1 [,f2...])

Specifies the structure of Fortran records and the form of the data fields within the records.

Format Codes

I	Integer data editing
F	Real data editing
D	Real, complex, or double-precision data editing
E	Real, complex, double- or extended-precision data editing
Q	Extended-precision data editing
G	Real, integer, or logical data transmission
P	Scale factor specification
Z	Hexadecimal data transmission
L	Logical variable transmission
A	Character data transmission
H	Character constant transmission
X	Skipping characters
T	Data transmission, start
TL	Data transmission, starts number of characters to left
TR	Data transmission, starts number of characters to right
Group	Repeats a set of format codes
S	Plus character control restoration
SP	Plus character control production
SS	Plus character control cessation
BN	Blanks ignored
BZ	Blanks treated as zeros
Slash	Record termination
Colon	Format control termination
B	Binary data transmission
O	Octal data transmission
$	End-of-record suppression

FUNCTION Statement

[*type*] FUNCTION *name* ([*arg1* [,*arg2*]...])

[*type***len*] FUNCTION *name***len* ([*arg1* [,*arg2*]...])

Identifies a function subprogram.

Assigned GO TO Statement

GO TO *i* [[,](*stl1* [,*stl2*] [,*stl3*]...)]

Transfers control to a statement label depending on the current assignment of *i*.

Statement Syntax

Computed GO TO Statement

GO TO (*stl1* [,*stl2*] [,*stl3*]...) [,] *m*

Transfers control to a statement label depending on the current value of *m*.

Unconditional GO TO Statement

GO TO *stl*

Transfers control to the statement specified by the statement label.

Arithmetic IF Statement

IF (*m*) *stl1*,*stl2*,*stl3*

Transfers control to a statement label depending on the value of the arithmetic expression *m*.

Block IF Statement

IF (*m*) THEN

Controls processing sequence.

ELSE Statement

ELSE

Indicates statements that are processed if the preceding block IF or ELSE IF condition is evaluated to be false.

ELSE IF Statement

ELSE IF (*m*) THEN

Indicates statements that are processed if the preceding block IF condition is evaluated to be false.

END IF Statement

END IF

Concludes an IF-block.

Logical IF Statement

IF (*m*) *st*

Evaluates a logical expression and processes or skips a statement, depending on whether the value of the expression is true or false.

IMPLICIT Statement

IMPLICIT *type* (*a* [,*a*]...) [,*type*(*a* [,*a*]...)...]

IMPLICIT *type***len* (*a* [,*a*]...) [,*type***len*(*a* [,*a*]...)...]

IMPLICIT NONE

Confirms or changes the default implied types, or voids implied typing altogether; *type* can be AUTOMATIC or STATIC.

INQUIRE by File

INQUIRE
(FILE=*fn*
[,*specifier*[,*specifier*]...])

Allows you to determine file existence, connection status and other properties of a named file. The list of possible specifiers follows INQUIRE by Unnamed File statement on page 14.

INQUIRE by Unit

INQUIRE
([UNIT=] *un*
[,*specifier*[,*specifier*]...])

Allows you to determine the existence of a unit, whether the unit is connected to a file, and, if the unit is connected, what the properties are of the unit and file connection. The list of possible specifiers follows INQUIRE by Unnamed File statement on page 14.

Statement Syntax

INQUIRE by Unnamed File

INQUIRE
([UNIT=] *un*,
FILE=*fn*
[,*specifier*[,*specifier*]...])

Allows you to determine the file existence and connection status for an unnamed file, as well as other properties of the file. Following is the list of possible specifiers.

ACCESS=*acc*
ACTION=*act*
BLANK=*blk*
CHAR=*chr*
| DELIM=*dlm*
DIRECT=*dir*
ERR=*stl*
| EXIST=*exs*
FORM=*frm*
| FORMATTED=*fmt*
IOSTAT=*ios*
KEYED=*kyd*
KEYEND=*ken*
KEYID=*kid*
KEYLENGTH=*kle*
KEYSTART=*kst*
LASTKEY=*lky*
LASTRECL=*lrl*
NAME=*nam*
NAMED=*nmd*
NEXTREC=*nxr*
NUMBER=*num*
OPENED=*opn*
PAD=*pad*
PASSWORD=*pwd*
POSITION=*pos*
READ=*ron*
READWRITE=*rwr*
RECL=*rcl*
SEQUENTIAL=*seq*
UNFORMATTED=*unf*
WRITE=*wri*

INTRINSIC Statement

INTRINSIC *name*1 [,*name*2...]

Identifies a name as representing a procedure supplied by VS FORTRAN Version 2, and permits a specific intrinsic function name to be used as an actual argument.

NAMELIST Statement

NAMELIST /*name*1/ *list*1 [/*name*2/ *list*2...]

Specifies one or more lists or names for use in READ and WRITE statements.

| **NULLIFY Statement**

| **NULLIFY (***pointer1* **[,** *pointer2* **...])**

| Disassociates a pointer variable from an addressed pointee variable and
| sets the value of the pointer variable to the unassigned state.

OPEN Statement

OPEN
([UNIT=] *un* [,ERR=*stl*] [,STATUS=*sta*]
[,FILE=*fn*] [,ACCESS=*acc*] [,BLANK=*blk*]
[,CHAR=*chr*]
[,FORM=*frm*] [,IOSTAT=*ios*]
[,RECL=*rcl*]
[,ACTION=*act*] [,PASSWORD=*pwd*]
| [,POSITION=*pos*]
| [,PAD=*pad*]
| [,DELIM=*dlm*]
[,KEYS=(*start* : *end* [,*start* : *end*]...)])

Connects an existing file to a unit, creates a file that is preconnected, creates a file and connects it to a unit, or changes certain specifiers of a connection between a file and a unit.

PARAMETER Statement

PARAMETER (*name1=constant1* [,*name2=constant2*]...)

Assigns a name to a constant.

PAUSE Statement

PAUSE [*n*]
PAUSE ['*message*']

Temporarily halts the processing of the program and displays a message.

| **POINTER Statement**

| **POINTER[*** *len*](*ptr1[** *len*],*ptee1*[(*dim*)])[,(*ptr2[** *len*],*ptee2*[(*dim*)])...]

| Specifies a pointer variable and associates it with a target variable, called a
| pointee variable. The value of the pointer variable is the storage address of
| the pointee variable.

Statement Syntax

PRINT Statement—Formatted with Sequential Access

PRINT *fmt* [,*list*]

Transfers data from internal storage to an external device.

PRINT Statement—List-Directed to External Devices

PRINT ***** [,*list*]

Transfers data from internal storage to an external device.

PRINT Statement—NAMELIST with External Devices

PRINT *name*

Transfers data from internal storage to an external device.

PROGRAM Statement

PROGRAM *name*

Assigns a name to a main program.

READ Statement—Asynchronous

READ
([UNIT=] *un*,
ID=*id*)
[*list*]

Transmits unformatted data from a direct-access or tape device using sequential access.

READ Statement—Formatted with Direct Access

READ
([UNIT=] *un*, [FMT=] *fmt*,
REC=*rec* [,ERR=*stl*]
[,IOSTAT=*ios*]) [*list*]

Transfers data from an external direct-access device into internal storage.

READ Statement—Formatted with Keyed Access (Direct Retrieval)

READ
([UNIT=] *un*, [FMT=] *fmt* [,ERR=*stl*]
[,IOSTAT=*ios*] [,KEYID=*kid*] [,NOTFOUND=*stl*]
{,KEY=*key* | ,KEYGE=*kgel* | ,KEYGT=*kgt*}) [*list*]

Transfers data from an external direct-access device into internal storage.

READ Statement—Formatted with Keyed Access (Sequential Retrieval)

READ
([UNIT=] *un*, [FMT=] *fmt* [,ERR=*stl*]
[,IOSTAT=*ios*] [,NOTFOUND=*stl* | ,END=*stl*])
[*list*]

Transfers data from an external direct-access device into internal storage.

READ Statement—Formatted with Sequential Access

READ *fmt* [,*list*]

READ
([UNIT=] *un*, [FMT=] *fmt*
[,ERR=*stl*] [,END=*stl*]
[,IOSTAT=*ios*]) [*list*]

Transfers data from an external I/O device to storage.

READ Statement—Formatted with Sequential Access to Internal Files

READ
([UNIT=] *un*, [FMT=] *fmt*
[,ERR=*stl*] [,END=*stl*]
[,IOSTAT=*ios*]) [*list*]

Transfers data from one area of internal storage into another area of internal storage.

Statement Syntax

READ Statement—List-Directed from External Devices

READ ***** [,*list*]

READ
([UNIT=] *un*, [FMT=] *****
[,ERR=*stl*] [,END=*stl*]
[,IOSTAT=*ios*]) [*list*]

Transfers data from an external device into internal storage.

READ Statement—List-Directed with Internal Files

READ
([UNIT=] *un*, [FMT=] *****
[,ERR=*stl*] [,END=*stl*]
[,IOSTAT=*ios*]) [*list*]

Transfers data from one area of internal storage to one or more other areas of internal storage.

READ Statement—NAMELIST with External Devices

READ name

READ
([UNIT=] *un*
{[,FMT=]*name* | [,NML=]*name*}
[,ERR=*stl*] [,END=*stl*]
[,IOSTAT=*ios*])

Transfers data from an external I/O device into storage.

READ Statement—NAMELIST with Internal Files

READ
([UNIT=] *un*
{[,FMT=]*name* | [,NML=]*name*}
[,ERR=*stl*] [,END=*stl*]
[,IOSTAT=*ios*])

Transfers data from one area of internal storage to one or more other areas of internal storage.

READ Statement—Unformatted with Direct Access

READ
([UNIT=] *un*, REC=*rec*
[,ERR=*stl*] [,IOSTAT=*ios*]
[,NUM=*n*]) [*list*]

Transfers data without conversion from an external direct-access device into internal storage.

READ Statement—Unformatted with Keyed Access (Direct Retrieval)

READ
([UNIT=] *un* [,ERR=*stl*] [,IOSTAT=*ios*] [,KEYID=*kid*]
{,KEY=*key* | ,KEYGE=*kge* | ,KEYGT=*kgt*}
[,NOTFOUND=*stl*] [,NUM=*n*]) [*list*]

Transfers data without conversion from an external direct-access I/O device into internal storage.

READ Statement—Unformatted with Keyed Access (Sequential Retrieval)

READ
([UNIT=] *un* [,ERR=*stl*] [,IOSTAT=*ios*]
[,NOTFOUND=*stl* | ,END=*stl*]
[,NUM=*n*]) [*list*]

Transfers data without conversion from an external direct-access I/O device into internal storage.

READ Statement—Unformatted with Sequential Access

READ
([UNIT=] *un* [,ERR=*stl*]
[,END=*stl*] [,IOSTAT=*ios*]
[,NUM=*n*]) [*list*]

Transfers data without conversion from an external I/O device into internal storage.

Statement Syntax

RETURN Statement in Function Subprogram

RETURN

Returns control to the calling program.

RETURN Statement in Subroutine Subprogram

RETURN [*m*]

Returns control to the calling program.

REWIND Statement

REWIND *un*

REWIND
([UNIT=] *un* [,ERR=*err*]
[,IOSTAT=*ios*])

Repositions a sequentially-accessed file at the beginning of the first record of the file.

REWRITE Statement—Formatted with Keyed Access

REWRITE
([UNIT=] *un*, [FMT=] *fmt*
[,ERR=*stl*] [,IOSTAT=*ios*]
[,DUPKEY=*stl*]) *list*

Replaces a record in a keyed file.

REWRITE Statement—Unformatted with Keyed Access

REWRITE
([UNIT=] *un* [,ERR=*stl*]
[,IOSTAT=*ios*] [,DUPKEY=*stl*]
[,NUM=*num*]) *list*

Replaces a record in a keyed file.

SAVE Statement

SAVE [*name1* [,*name2*]...]

Retains the definition status of the name of a named common block, variable or array after the processing of a RETURN or END statement in a subprogram.

Statement Function Statement

name ([*arg1* [,*arg2*]...]) = *m*

Specifies operations to be performed whenever that statement function name appears as a function reference in another statement in the same program.

STATIC Statement

STATIC *name1* **[***/clist1/***] [,** *name2***[***/clist2/***] ...]**

Identifies the variables and arrays to be assigned the static storage class.

STOP Statement

STOP [*n*]
STOP ['*message*']

Ends the processing of the object program and displays a message.

SUBROUTINE Statement

SUBROUTINE *name* [([*arg1* [,*arg2*]...])]

Identifies a subroutine subprogram.

TRACE OFF Statement

TRACE OFF

Stops the display of program flow by statement label.

Statement Syntax

TRACE ON Statement

TRACE ON

Initiates the display of program flow by statement label.

WAIT Statement

WAIT
([UNIT=]*un*, ID=*id*
[,COND=*i1*] [,NUM=*i2*])
[*list*]

Synchronizes the completion of the data transmission begun by the corresponding asynchronous READ or WRITE statement.

WRITE Statement—Asynchronous

WRITE
([UNIT=] *un*,
ID=*id*)
list

Transmits data from an array in main storage to an external file.

WRITE Statement—Formatted with Direct Access

WRITE
([UNIT=] *un*, [FMT=] *fmt*,
REC=*rec* [,ERR=*stl*]
[,IOSTAT=*ios*]) [*list*]

Transfers data from internal storage onto an external device.

WRITE Statement—Formatted with Keyed Access

WRITE
([UNIT=] *un*, [FMT=] *fmt*
[,ERR=*stl*] [,IOSTAT=*ios*]
[,DUPKEY=*stl*]) *list*

Transfers data from internal storage onto an external device.

WRITE Statement—Formatted with Sequential Access

WRITE
([UNIT=] *un*, [FMT=] *fmt*
[,ERR=*stl*] [,IOSTAT=*ios*])
[*list*]

Transfers data from internal storage to a file.

WRITE Statement—Formatted with Sequential Access to Internal Files

WRITE
([UNIT=] *un*, [FMT=] *fmt*
[,ERR=*stl*] [,IOSTAT=*ios*])
[*list*]

Transfers data from one or more areas of internal storage to another area in internal storage.

WRITE Statement—List-Directed to External Devices

WRITE
([UNIT=] *un*, [FMT=] *****
[,ERR=*stl*] [,IOSTAT=*ios*])
[*list*]

Transfers data from internal storage to a file.

WRITE Statement—List-Directed with Internal Files

WRITE
([UNIT=] *un*, [FMT=] *****
[,ERR=*stl*] [,IOSTAT=*ios*])
[*list*]

Transfers data from one or more areas of internal storage to another area of internal storage.

Statement Syntax

WRITE Statement—NAMELIST with External Devices

```
WRITE
      ([UNIT=] un,
      {[,FMT=]name | [,NML=]name}
      [,ERR=stl]
      [,IOSTAT=ios])
```

Transfers data from internal storage to a file.

WRITE Statement—NAMELIST with Internal Files

```
WRITE
      ([UNIT=] un,
      {[,FMT=]name | [,NML=]name}
      [,ERR=stl]
      [,IOSTAT=ios])
```

Transfers data from one or more areas of internal storage to another area of internal storage.

WRITE Statement—Unformatted with Direct Access

```
WRITE
      ([UNIT=] un, REC=rec
      [,ERR=stl] [,IOSTAT=ios]
      [,NUM=n]) [list]
```

Transfers data without conversion from internal storage to a file.

WRITE Statement—Unformatted with Keyed Access

```
WRITE
      ([UNIT=] un [,ERR=stl]
      [,IOSTAT=ios] [,NUM=n]
      [,DUPKEY=stl]) list
```

Transfers data without conversion from internal storage to a file.

WRITE Statement—Unformatted with Sequential Access

WRITE
([UNIT=] *un* [,ERR=*stl*]
[,IOSTAT=*ios*] [,NUM=*n*])
[*list*]

Transfers data without conversion from internal storage to a file.

Parallel Statement Categories

In the following statements, extensions to the Fortran standard language are printed in color.

Parallel Task Management Statements

- ORIGINATE
- SCHEDULE
- TERMINATE
- WAIT FOR
  - ALL TASKS
  - ANY TASK
  - TASK

Parallel Loop Statements

- EXIT
- LOCAL
- PARALLEL DO
  - DOAFTER
  - DOBEFORE
  - DOEVERY

Parallel Sections Statements

- END SECTIONS
- LOCAL
- PARALLEL SECTIONS
- SECTION

Parallel Call Statements

- PARALLEL CALL
- WAIT FOR ALL CALLS

Parallel Statement Syntax

In the following statements, extensions to the Fortran standard language are printed in color.

Parallel Task Management Statements

ORIGINATE Statement

ORIGINATE TASK *ptaskid* | ORIGINATE ANY TASK *rtaskid*

Creates a new parallel task.

SCHEDULE Statement

{SCHEDULE TASK *ptaskid* | SCHEDULE ANY TASK *rtaskid*}
[,SHARING (*shrcom* [,*shrcom*]...)]
[,COPYING (*cpcom* [,*cpcom*]...)]
[,COPYINGI (*cpicom* [,*cpicom*]...)]
[,COPYINGO (*cpocom* [,*cpocom*]...)]
,CALLING *subx* [([*arg*[,*arg*]...])]

Assigns a subroutine to an originated task for parallel processing. Must have a matching WAIT FOR ALL TASKS, WAIT FOR ANY TASK, or WAIT FOR TASK statement.

TERMINATE Statement

TERMINATE TASK *taskid*

Deletes a parallel task created by ORIGINATE.

WAIT FOR Statements

One of the following WAIT FOR statements is required for each SCHEDULE statement in a parallel task.

WAIT FOR ALL TASKS Statement

WAIT FOR ALL TASKS

Causes the scheduling routine to wait for all originated tasks, owned by the scheduling routine, to finish processing.

Parallel Statement Syntax

WAIT FOR ANY TASK Statement

WAIT FOR ANY TASK *rtaskid*

Causes the scheduling routine to wait for any originated task, owned by the scheduling routine, to finish processing.

WAIT FOR TASK Statement

WAIT FOR TASK *ptaskid*

Causes the scheduling routine to wait for a specified originated task, owned by the scheduling routine, to finish processing.

Parallel Loop Statements

EXIT Statement

EXIT *stl*

Stops processing of the **DOEVERY** block of a parallel loop, whether or not all the iterations have finished running.

LOCAL Statement (for parallel loops)

LOCAL *var* [,*var*]...

Specifies that an instance of each variable and array listed is provided to each virtual processor participating in execution of the parallel loop.

PARALLEL DO Statement

PARALLEL DO [*stl* [,]] *i* = *e1*, *e2* [,*e3*]

Similar to the **DO** statement except each iteration of the loop can be processed concurrently; permits parallelism of the loops to be explicitly stated.

DOAFTER Statement

DOAFTER [LOCK]

Indicates the beginning of a block of statements that each virtual processor participating in the execution of the PARALLEL DO processes *after* the loop is run.

DOBEFORE Statement

DOBEFORE [LOCK]

Indicates the beginning of a block of statements that each virtual processor participating in the execution of the PARALLEL DO processes *before* the loop is run.

DOEVERY Statement

DOEVERY

Indicates the beginning of a block of statements with processing shared by the virtual processors assigned to the loop.

Parallel Sections Statements

END SECTIONS Statement

END SECTIONS

Terminates a group of parallel sections.

LOCAL Statement (for parallel sections)

LOCAL *var* [,*var*]...

Specifies that an instance of each variable and array listed is provided to each virtual processor participating in execution of the parallel loop.

PARALLEL SECTIONS Statement

PARALLEL SECTIONS

Indicates the beginning of a group of sections that can be run in parallel with other sections in the group.

SECTION Statement

SECTION [*m*][,WAITING (*n1* [,*n2*]...)]

Indicates the beginning of a block of statements to be processed as a parallel thread.

Parallel Call Statements

PARALLEL CALL Statement

PARALLEL CALL *name* [([*arg1*[,*arg2*]...])]

Assigns a subroutine to run as a parallel thread.

WAIT FOR ALL CALLS Statement

WAIT FOR ALL CALLS

Causes the calling routine to wait until all subroutines invoked with PARALLEL CALL within the same parallel thread have completed.

Compile-Time Options

Note: To specify compile-time options on a program-by-program basis, use the @PROCESS compiler directive; for example: @PROCESS LIST TEST.

AUTODBL (NONE | DBL | DBL4 | DBL8 | DBLPAD | DBLPAD4 | DBLPAD8 | *value*)
Provides an automatic means of converting single-precision floating-point calculations to double-precision, and double-precision calculations to extended-precision.

CHARLEN (*number* | 500)
Specifies the maximum length permitted for any character variable, character array element, or character function.

CI (*number1*, *number2*,...)
Specifies the identification numbers of the INCLUDE statements to be processed.

DBCS | NODBCS
Specifies whether the source file may contain double-byte characters.

DC (* | *name1*, *name2*,...)
Defines the names of common blocks to be allocated at run time.

DDIM | NODDIM
Indicates that the pointee arrays that specify object-time dimensions are to have those dimensions evaluated dynamically at each element reference.

DECK | NODECK
Specifies whether the compiler is to write the object module to the data set defined by the ddname SYSPUNCH.

DIRECTIVE (*trigger-constant*) | NODIRECTIVE [(*trigger-constant*)]
Specifies whether selected comments containing compiler directive statements are to be processed.

DYNAMIC (name1,name2...)
Provides dynamic loading of user subroutines or functions during program execution.

EC (* | *name1*, *name2*,...)
Defines the names of common blocks to be dynamically allocated as extended common blocks.

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

Compile-Time Options

FIPS (S | F) | NOFIPS
Specifies whether standard language flagging is to be performed, and, if it is, the standard language flagging level: subset or full.

Items not defined in the current American National Standard are flagged.

FLAG (I | W | E | S)
Specifies the level of diagnostic messages to be written: I (information) or higher, W (warning) or higher, E (error) or higher, or S (severe) or higher.

FREE | FIXED
Indicates whether the input source program is in free format or in fixed format.

GOSTMT | NOGOSTMT
Specifies whether internal statement numbers (for run-time error debugging information) are to be generated for a calling sequence to a subprogram or to the run-time library from the compiler-generated code.

HALT (I|W|E|S)
Causes termination of the compile after any phase if the compiler return code is at or above the specified level.

ICA [(
[USE (*name1*,*name2*,**...)]**
[UPDATE (*name***)]**
[DEF (nameA, nameB,...)]
[MXREF (S | L) | NOMXREF]
[CLEN | NOCLEN]
[CVAR | NOCVAR]
[MSG ({ NEW | NONE | ALL })]
[MSGON (*number1***,** *number2***, ...) | MSGOFF (***number1***,** *number2***, ...)]**
[RCHECK | NORCHECK]
)]
| NOICA
Specifies whether intercompilation analysis is to be performed, specifies the files containing intercompilation analysis information to be used or updated, and controls output from intercompilation analysis.

IL (DIM | NODIM)
Specifies whether the code for adjustably-dimensioned arrays is to be placed inline, IL(DIM) or called from the library, IL(NODIM).

LANGLVL (66 | 77)
Specifies the language level in which the input source program is written: the FORTRAN 66 language level or the FORTRAN 77 language level.

LINECOUNT (*number* **| 60)**
Specifies the maximum number of lines on each page of the printed source listing.

Compile-Time Options

LIST | NOLIST
Specifies whether the object module listing is to be written.

MAP | NOMAP
Specifies whether a table of source program variable names, named constants, and statement labels and their displacements is to be produced.

NAME (*name* | MAIN#)
Specifies the name of the control section (CSECT) generated in the object module of the main program (valid only when LNGLVL(66) is specified).

OBJECT | NOOBJECT
Under CMS, specifies whether the compiler is to write the object module to the file associated with the ddname TEXT.

Under MVS, specifies whether the compiler is to write the object module to the data set associated with the ddname SYSLIN.

OPTIMIZE (0 | 1 | 2 | 3) | NOOPTIMIZE
Specifies the optimizing level to be used during compilation:

OPTIMIZE (0) or NOOPTIMIZE specifies no optimization.
OPTIMIZE (1) specifies partial optimization.
OPTIMIZE (2) specifies full optimization with interruption localizing.
OPTIMIZE (3) specifies full optimization without interruption localizing.

PARALLEL [(
[REPORT [(*optionlist*)] | NOREPORT]
[LANGUAGE | NOLANGUAGE]
[AUTOMATIC | NOAUTOMATIC]
[REDUCTION | NOREDUCTION]
[TRACE | NOTRACE]
[ANZCALL | NOANZCALL]
)]
| NOPARALLEL
Specifies suboptions to the compiler for generating code for DO loops, the PARALLEL DO, PARALLEL SECTIONS, and PARALLEL CALL constructs, and the task management statements.

PTRSIZE (4|8)
Sets the default length for pointer variables.

RENT | NORENT
Specifies whether the object module generated is suitable for use in a shareable area.

Compile-Time Options

SAA | NOSAA
Specifies whether flagging of language elements that are not part of the Systems Application Architecture* (SAA*) is to be performed.

SC (* | *name1*, *name2*,...)
Defines the names of common blocks to be compiled as static common blocks.

SDUMP [(ISN | SEQ)] | NOSDUMP
Specifies whether symbolic dump information is to be generated, and if so, whether internal statement numbers or sequence numbers will be used. The -g flag on the **fvs** command is equivalent to this option.

SOURCE | NOSOURCE
Specifies whether the source listing is to be produced.

SRCFLG | NOSRCFLG
Controls the insertion of error messages in the source listing.

SXM | NOSXM
Formats XREF or MAP listing output to a 72-character width.

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.

TERMINAL | NOTERMINAL
Specifies whether error messages and compiler diagnostics are to be written on the SYSTERM data set and whether a summary of messages for all compilations is to be written at the end of the listing.

TEST | NOTEST
TEST overrides any optimization level about OPTIMIZE(0).

TRMFLG | NOTRMFLG
Controls the display of error messages on the terminal.

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

---

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

| **[CMPLXOPT | NOCMPLXOPT]**
)]
| NOVECTOR
Specifies whether to invoke the vectorization process, which produces programs that can utilize the speed of the IBM* 3090* vector facility.

XREF | NOXREF
Specifies whether a cross-reference listing is to be produced.

* IBM and 3090 are trademarks of the International Business Machines Corporation.

Conflicting Compile-Time Options

Conflicting Compile-Time Options		Options Assumed	
DBCS	FIPS	DBCS	NOFIPS
DBCS	SAA	DBCS	NOSAA
DC	EC	**Note 1**	**Note 1**
DC	SC	**Note 1**	**Note 1**
EC	SC	**Note 1**	**Note 1**
EC	NOEMODE	EC	EMODE
FIPS	FLAG¬=I	FIPS	FLAG=I
FIPS	SAA	Installation default	Installation default
FREE	FIPS	FREE	NOFIPS
FREE	SAA	FREE	NOSAA
FREE	SDUMP(SEQ)	FREE	SDUMP(ISN)
LANGLVL(66)	DBCS	LANGLVL(66)	NODBCS
LANGLVL(66)	FIPS	LANGLVL(66)	NOFIPS
LANGLVL(66)	SAA	LANGLVL(66)	NOSAA
LANGLVL(77)	NAME	LANGLVL(77)	Ignore NAME
NODECK	SYM	NODECK	NOSYM
NOOBJ	SYM	NOOBJ	NOSYM
NOTRMFLG	VEC(REP(TERM...))	NOTRMFLG	VEC(REP(NOTERM...))
NOTRMFLG	PAR(REP(TERM...))	NOTRMFLG	PAR(REP(NOTERM...))
PAR	OPT(0) or OPT(1)	PAR	OPT(3)
PAR	TEST	NOPAR	TEST
PAR(NOREP)	VEC(REP(LIST))	PAR(REP(LIST))	VEC(REP(LIST))
PAR(ANZCALL)	VEC(NOANZCALL)	PAR(NOANZCALL)	VEC(NOANZCALL)
PAR(NOANZCALL)	VEC(ANZCALL)	PAR(ANZCALL)	VEC(ANZCALL)
PAR(NOREP)	VEC(REP(XLIST))	PAR(REP(XLIST))	VEC(REP(XLIST))
PAR(NOREP)	VEC(REP(STAT))	PAR(REP(STAT))	VEC(REP(STAT))
PAR(REP(LIST))	VEC(NOREP)	PAR(REP(LIST))	VEC(REP(LIST))
PAR(REP(XLIST))	VEC(NOREP)	PAR(REP(XLIST))	VEC(REP(XLIST))
PAR(REP(STAT))	VEC(NOREP)	PAR(REP(STAT))	VEC(REP(STAT))
PAR(REP)	VEC(REP(SLIST))	PAR(REP(SLIST XLIST))	VEC(REP(SLIST XLIST))
TEST	OPT(1), OPT(2), OPT(3)	TEST	OPT(0)
TEST	NOSDUMP	TEST	SDUMP(ISN)
VEC	OPT(0) or OPT(1)	VEC	OPT(3)
VEC(IVA)	NOSDUMP	VEC(IVA)	SDUMP(ISN)
VEC(IVA)	PAR	VEC(NOIVA)	PAR

Note:

1. The SC, DC, and EC compile-time options use the last option indicated to resolve conflicts between them.

Compiler Directives

@PROCESS

Provides compile-time options that override the corresponding default options or those specified at compiler invocation.

EJECT

EJECT

Starts a new full page of the source listing.

INCLUDE

INCLUDE (*member*) [*n*]

INCLUDE '*filename* [*filetype* [*filemode*]] [(*member*)]'

INCLUDE '*dsn* [(*member*)]'

INCLUDE '*filename*'

Inserts a specified statement or group of statements into a program unit.

Parallel and Vector Directives

ASSUME COUNT

Local Directive:
ASSUME COUNT (*val*)

Global Directive:
ASSUME COUNT (*{val|var=val[,var=val]...}*) ON

ASSUME COUNT OFF

Specifies the value to be used for vector or parallel cost analysis when a loop iteration count cannot be determined at compile time.

IGNORE

IGNORE [RECRDEPS [(*array-list*)]] [CALLDEPS [(*name*[,*name*...])]]

Instructs the compiler to ignore specified dependences in a loop.

Warning: Use with extra caution. Incorrectly specifying IGNORE can produce erroneous program results.

PREFER

Local PREFER:
PREFER [SCALAR | VECTOR]
[SERIAL | PARALLEL]
[CHUNK({*n*|*n:*|*:m*.|*n::m*})]

Global PREFER:
PREFER [SCALAR | SERIAL] ON | OFF

Requests that particular loops be run in vector or scalar and/or parallel or serial modes.

Run-Time Options

ABSDUMP | NOABSDUMP
Specifies whether the post-ABEND symbolic dump information is to be printed in the event of an abnormal termination.

AUTOTASK (*loadmod*,*ntasks*) | NOAUTOTASK
Specifies whether the multitasking facility (MTF) is enabled for your program. This option available on MVS only.

CNVIOERR | NOCNVIOERR
Specifies whether to treat input conversion errors as I/O errors.

DEBUG | NODEBUG
Specifies whether to call VS FORTRAN Version 2 interactive debug.

DEBUNIT(*s1* [, *s2*...]) | NODEBUNIT (MVS format)
DEBUNIT(*s1* [*s2 s3*]) | NODEBUNIT (CMS format)
Identifies Fortran units considered to be connected to a terminal, so that interactive debug can handle their I/O in batch mode.

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

ERRUNIT(*number*)
Identifies the unit number to which run-time error information is to be directed.

FAIL (ABEND | RC | ABENDRC)
Indicates how to terminate unsuccessful programs. This option is not supported for parallel processing.

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

INQPCOPN | NOINQPCOPN
Controls using the OPENED specifier on an INQUIRE by unit to determine whether a preconnected unit had any I/O statements directed to it.

IOINIT | NOIOINIT
Specifies whether the normal initialization for I/O processing occurs during initialization of the run-time environment.

OCSTATUS | NOOCSTATUS
Specifies whether to verify the OPEN and CLOSE status specifiers.

PARALLEL [(*numprocs*)] | NOPARALLEL
Specifies whether the program runs in the parallel processing environment.

Run-Time Options

PRTUNIT(*number***)**
Identifies the unit number that is to be used for PRINT or WRITE statements that do not specify a unit number.

PTRACE[(*options***)]**
Enables the Paralllel Trace Facility and causes the Trace File to be initialized.

PUNUNIT(*number***)**
Identifies the unit number that is to be used for PUNCH statements that do not specify a unit number.

RDRUNIT(*number***)**
Identifies the unit number that is to be used for READ statements that do not specify a unit number.

RECPAD | NORECPAD
Specifies whether a formatted input record is padded with blanks when an input list and format specification require more data from the record than the record contains.

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

STAE | NOSTAE
Specifies whether the run-time environment takes control in the event of an abnormal termination.

XUFLOW | NOXUFLOW
Specifies whether an exponent underflow causes a program interrupt.

Service Subroutines

ARGSTR Subroutine

CALL ARGSTR(*string*,*rc*)

Retrieves the user-supplied parameters from the command line.

ASSIGNM Subroutine

CALL ASSIGNM (*input*, *output*, *rcode*, *rsncode*)

Moves a character string containing double-byte data to a character variable, substring, or array element, preserving balanced shift codes.

CLOCK Subroutine

CALL CLOCK (*cpuclk*, [,*count* [,*max*]])

Returns the value of the processor clock as a positive integer.

CLOCKX Subroutine

CALL CLOCKX (*cpuclk* [,*xcount* [,*xmax*]])

Returns an abbreviated version of the processor clock in a REAL*8 variable.

CDUMP/CPDUMP Subroutines

CALL {CDUMP | CPDUMP} (*a1*,*b1*,*a2*,*b2* ...)

Provides a symbolic dump of a specified area of storage containing character data.

CPUTIME Subroutine

CALL CPUTIME (*accumcpu*, *rcode*)

Lets you determine the amount of processor time used by a program or portion of a program. This subroutine is not allowed in a parallel program.

DATIM Subroutine

CALL DATIM (*now*)

Provides information about the date, time of day, and processor clock.

Service Subroutines

DATIMX Subroutine

CALL DATIMX (*now*)

Provides the date and time in a form that can be used to produce printable or formatted data.

DUMP/PDUMP Subroutines

CALL {DUMP | PDUMP} (*a1*,*b1*,*k1*,*a2*,*b2*,*k2* ...)

Provides a symbolic dump of a specified area of storage.

DVCHK Subroutine

CALL DVCHK (*k*)

Tests for divide-check exception.

EXIT Subroutine

CALL EXIT

Ends processing of the program. This subroutine can be called only in serial parts of the root task.

FILEINF Subroutine

CALL FILEINF [(*rcode* [,*param1*, *value1*, *param2*, *value2* ...])]

Sets up file characteristics to be used by an OPEN or an INQUIRE statement.

MVBITS Subroutine

CALL MVBITS(*arg1*,*arg2*,*arg3*,*arg4*,*arg5*)

Allows a bit subfield of one integer value to be assigned to a bit subfield of another integer value.

Service Subroutines

OVERFL Subroutine

CALL OVERFL (*k*)

Tests for exponent overflow or underflow.

PFAFFS Routine

CALL PFAFFS

Switches the virtual processor the parallel thread is running on to the same one that the parallel program first started running on; allows use of system services.

PFAFFC Routine

CALL PFAFFC

Releases any processor affinity previously set by calling PFAFFS.

PYIELD Routine

CALL PYIELD

Causes the VS FORTRAN Version 2 library to interrupt the execution of the current thread and to attempt to execute any threads now waiting to execute.

PTPARM Routine

CALL PTPARM (*argstring*)

Provides a means for your program to dynamically control the tracing activity of the Parallel Trace Facility during program execution.

PTWRIT Routine

CALL PTWRIT(*category*,*type*,*user_data* [,*user_data_len*])

Allows you to generate your own trace records for events you determine are of significance in the execution of your program.

Service Subroutines

SDUMP Subroutine

CALL SDUMP [(*rtn1* [,*rtn2*]...)]

Provides a symbolic dump of all variables in a program unit.

SYSABD Subroutine

CALL SYSABD (*compl-code*)

Causes abnormal termination of your job with a dump.

SYSABN Subroutine

CALL SYSABN (*compl-code*)

Causes abnormal termination of your job without a dump.

SYSRCS Subroutine

CALL SYSRCS (*n*)

Saves a return code value for future termination.

SYSRCT

CALL SYSRCT (*m*)

Obtains the value of the currently saved return code.

SYSRCX Subroutine

CALL SYSRCX [(*k*)]

Ends program processing using either the saved return code or a supplied return code.

UNTANY Subroutine

CALL UNTANY (*rcode*, *startnum*, *endnum*, *unitnum*)

Identifies the lowest Fortran unit number that is available, within a range of unit numbers, regardless of the file definitions in effect.

UNTNOFD Subroutine

CALL UNTNOFD (*rcode*, *startnum*, *endnum*, *unitnum*)

Identifies the lowest Fortran unit number that is available, within a range of unit numbers, that does not have a user-specified file definition associated with it.

XUFLOW Subroutine

CALL XUFLOW (*k*)

Allows or suppresses a program interrupt caused by exponent underflow.

Parallel Library Event Service Subroutines

PEORIG Subroutine

CALL PEORIG (*eventid* [,*postcount* [,*waitcount* [,*unique*]]])

Creates and initializes an event and returns an identifier for the event.

PEPOST Subroutine

CALL PEPOST (*eventid*)

Posts the specified event.

PETERM Subroutine

CALL PETERM (*eventid*)

Deletes the specified event.

PEWAIT Subroutine

CALL PEWAIT (*eventid*)

Causes the calling parallel thread to wait until the event's *post-count* (or *wait-count* if the current post-count equal to the initial value) is reached.

Parallel Library Lock Service Subroutines and Function

PLCOND Function

PLCOND (*lockid* [,*mode* [,*var* [,*var*]...]])

Conditionally obtains the specified lock.

PLFREE Subroutine

CALL PLFREE (*lockid* [,*var* [,*var* [*var*]...]])

Releases the specified lock.

PLLOCK Subroutine

CALL PLLOCK (*lockid*[,*mode*[,*var*[,*var*]...]])

Obtains the specified lock. If the lock is currently owned by another parallel thread, waits until lock is available.

PLORIG Subroutine

CALL PLORIG (*lockid*)

Creates and initializes a lock and returns an identifier for the lock.

PLTERM Subroutine

CALL PLTERM (*lockid*)

Deletes the specified lock.

Parallel Function

NPROCS Function

NPROCS ([*n*])

Allows the program to determine the number of virtual processors specified at run time.

Data-in-Virtual Subroutines

The data-in-virtual subroutines are not available for parallel processing.

DIVCML Subroutine

CALL DIVCML (*rcode, dyncom, length*)

Obtains the length of a dynamic or extended common.

DIVINF Subroutine

CALL DIVINF (*rcode, dyncom, objsize_commons, divobj, type, access*)

Allows you to associate a data object with a dynamic or extended common for reading or for reading and writing (fixed-view).

DIVINV Subroutine

CALL DIVINV (*rcode, obj-id, objsize_pages, divobj, type, access*)

Allows you to associate a data object with a data object ID for reading or for reading and writing (varying-view).

DIVRES Subroutine

CALL DIVRES (*rcode, dyncom*)

Resets the data in the dynamic or extended common to the values in the mapped part of the data object, eliminating any changes that have been made in the dynamic or extended common, either initially or since the last DIVSAV.

DIVSAV Subroutine

CALL DIVSAV (*rcode, dyncom*)

Saves changes made in the dynamic or extended common to the data object that has been accessed for READWRITE.

Data-in-Virtual Subroutines

DIVTRF Subroutine

CALL DIVTRF (*rcode, dyncom*)

Terminates the association of the data object to the dynamic or extended common (fixed-view).

DIVTRV Subroutine

CALL DIVTRV (*rcode, obj-id*)

Terminates the association between the data object ID and the data object (varying-view).

DIVVWF Subroutine

CALL DIVVWF (*rcode, dyncom, mapnum*)

Establishes the part of the data object the dynamic or extended common maps (fixed-view).

DIVVWV Subroutine

CALL DIVVWV (*rcode, dyncom, offset, obj-id*)

Establishes the part of the data object the dynamic or extended common maps (varying-view).

Multitasking Facility (MTF) Subroutines

DSPTCH Subroutine

CALL DSPTCH (*subrname* [, (*arg1*) [, (*arg2*)]...])

Schedules a parallel subroutine for processing in a subtask.

NTASKS Subroutine

CALL NTASKS (*n*)

Returns the number of subtasks specified with the AUTOTASK keyword in the PARM parameter of the EXEC statement for the job step. Returns a value of zero when the AUTOTASK keyword is not in effect.

SHRCOM Subroutine

CALL SHRCOM (*dyncom*)

Designates a dynamic common as shareable among the main task program and the parallel subroutines.

SYNCRO Subroutine

CALL SYNCRO

Causes the main task program to wait until all scheduled parallel subroutines finish processing.

Error-Handling Subroutines

Note that each parallel task has its own error option table. A parallel thread uses the error option table associated with the parallel task in which it runs.

ERRMON Subroutine

CALL ERRMON (*imes*, *iretcd*, *ierno* [, *data1*] [, *data2*, ...])

Calls the error monitor routine.

ERRSAV Subroutine

CALL ERRSAV (*ierno*, *tabent*)

Copies an option table entry into an 8-byte storage area accessible to the Fortran programmer.

ERRSET Subroutine

CALL ERRSET (*ierno*, *inoal* [, *inomes*] [, *itrace*]
[, *iusadr*] [, *irange*])

Permits the user to control processing when error conditions occur.

ERRSTR Subroutine

CALL ERRSTR (*ierno*, *tabent*)

Stores an entry in the option table.

ERRTRA Subroutine

CALL ERRTRA

Dynamically requests a traceback and continued processing.

Intrinsic Functions

Intrinsic functions are procedures supplied in VS FORTRAN Version 2 for standard mathematical computations, character manipulations, and bit manipulations.

The intrinsic functions provided by VS FORTRAN Version 2 are described in the following figure.

Figure 2 (Page 1 of 6). Intrinsic Functions

Intrinsic Function	Generic Name	Specific Name	No. of Arguments	Type of Argument	Type and Range of Function
		Exponential and Logarithmic Functions			
Exponential	EXP	EXP[2] DEXP[2] QEXP CEXP CDEXP CQEXP	1	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32
Natural logarithm	LOG	ALOG[2] DLOG[2] QLOG CLOG CDLOG CQLOG	1	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32
Common logarithm	LOG10	ALOG10[2] DLOG10[2] QLOG10	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
		Trigonometric Functions			
Sine	SIN	SIN[2] DSIN[2] QSIN CSIN CDSIN CQSIN	1	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32
Cosine	COS	COS[2] DCOS[2] QCOS CCOS CDCOS CQCOS	1	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16 COMPLEX*32

Figure 2 (Page 2 of 6). Intrinsic Functions

Intrinsic Function	Generic Name	Specific Name	No. of Arguments	Type of Argument	Type and Range of Function
Tangent	TAN	TAN[2] DTAN[2] QTAN	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Cotangent	COTAN	COTAN[2] DCOTAN[2] QCOTAN	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arcsine	ASIN	ASIN[2] DASIN[2] QARSIN	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arccosine	ACOS	ACOS[2] DACOS[2] QARCOS	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arctangent	ATAN	ATAN[2] DATAN[2] QATAN	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
	ATAN2	ATAN2[2] DATAN2[2] QATAN2	2	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
		Hyperbolic Functions			
Hyperbolic sine	SINH	SINH[2] DSINH QSINH	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Hyperbolic cosine	COSH	COSH[2] DCOSH QCOSH	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Hyperbolic tangent	TANH	TANH[2] DTANH QTANH	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
		Miscellaneous Mathematical Functions			
Truncation	AINT	AINT DINT QINT	1	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Nearest whole number	ANINT	ANINT DNINT	1	REAL*4 REAL*8	REAL*4 REAL*8
Nearest integer	NINT	NINT IDNINT	1	REAL*4 REAL*8	INTEGER*4 INTEGER*4

Intrinsic Functions

Figure 2 (Page 3 of 6). Intrinsic Functions

Intrinsic Function	Generic Name	Specific Name	No. of Arguments	Type of Argument	Type and Range of Function
Absolute value	ABS	IABS[5]	1	any integer	same as arg
		ABS		REAL*4	REAL*4
		DABS		REAL*8	REAL*8
		QABS		REAL*16	REAL*16
		CABS[2]		COMPLEX*8	REAL*4
		CDABS[2]		COMPLEX*16	REAL*8
		CQABS		COMPLEX*32	REAL*16
Error function	ERF	ERF	1	REAL*4	REAL*4
		DERF		REAL*8	REAL*8
		QERF	1	REAL*16	REAL*16
					REAL*4
		ERFC		REAL*4	REAL*8
	ERFC	DERFC		REAL*8	REAL*16
		QERFC		REAL*16	
Gamma and log gamma	GAMMA	GAMMA	1	REAL*4	REAL*4
		DGAMMA		REAL*8	REAL*8
	LGAMMA	ALGAMA		REAL*4	REAL*4
		DLGAMA		REAL*8	REAL*8
Remaindering	MOD	MOD[5]	2	any integer	same as arg
		AMOD		REAL*4	REAL*4
		DMOD		REAL*8	REAL*8
		QMOD		REAL*16	REAL*16
Transfer of sign	SIGN	ISIGN[5]	2	any integer	same as arg
		SIGN		REAL*4	REAL*4
		DSIGN		REAL*8	REAL*8
		QSIGN		REAL*16	REAL*16
Positive difference	DIM	IDIM[5]	2	any integer	same as arg
		DIM		REAL*4	REAL*4
		DDIM		REAL*8	REAL*8
		QDIM		REAL*16	REAL*16
Double precision product		DPROD	2	REAL*4	REAL*8
Imaginary part of a complex argument	IMAG	AIMAG	1	COMPLEX*8	REAL*4
		DIMAG		COMPLEX*16	REAL*8
		QIMAG		COMPLEX*32	REAL*16
Complex conjugate	CONJG	CONJG	1	COMPLEX*8	COMPLEX*8
		DCONJG		COMPLEX*16	COMPLEX*16
		QCONJG		COMPLEX*32	COMPLEX*32
Square root	SQRT	SQRT[2]	1	REAL*4	REAL*4
		DSQRT[2]		REAL*8	REAL*8
		QSQRT		REAL*16	REAL*16
		CSQRT		COMPLEX*8	COMPLEX*8
		CDSQRT		COMPLEX*16	COMPLEX*16
		CQSQRT		COMPLEX*32	COMPLEX*32

Figure 2 (Page 4 of 6). Intrinsic Functions

Intrinsic Function	Generic Name	Specific Name	No. of Arguments	Type of Argument	Type and Range of Function
		Conversion and Maximum/Minimum Functions[1]			
Conversion to type integer	INT	—	1	any integer	same as arg
		IFIX		REAL*4	INTEGER*4
		IDINT		REAL*8	INTEGER*4
		IQINT		REAL*16	INTEGER*4
		—		COMPLEX*8	INTEGER*4
		HFIX[3]		REAL*4	INTEGER*2
Conversion to type real	REAL	FLOAT[5]	1	any integer	REAL*4
		—		REAL*4	REAL*4
		SNGL		REAL*8	REAL*4
		SNGLQ		REAL*16	REAL*4
		—		COMPLEX*8	REAL*4
		DREAL		COMPLEX*16	REAL*8
		QREAL		COMPLEX*32	REAL*16
Conversion to type double precision	DBLE	DFLOAT[5]	1	any integer	REAL*8
		DBLE		REAL*4	REAL*8
		—		REAL*8	REAL*8
		DBLEQ		REAL*16	REAL*8
		—		COMPLEX*8	REAL*8
Conversion to type extended precision	QEXT	QFLOAT[5]	1	any integer	REAL*16
		QEXT		REAL*4	REAL*16
		QEXTD		REAL*8	REAL*16
Conversion to type complex	CMPLX	—	1 or 2	any integer	COMPLEX*8
		CMPLX		REAL*4	COMPLEX*8
		—		REAL*8	COMPLEX*8
		QCMPLX		REAL*16	COMPLEX*32
		—		COMPLEX*8	COMPLEX*8
		DCMPLX[3]		REAL*8	COMPLEX*16
Choosing largest value	MAX	MAX0[5]	≥ 2	any integer	same as arg
		AMAX1		REAL*4	REAL*4
		DMAX1		REAL*8	REAL*8
		QMAX1		REAL*16	REAL*16
		AMAX0[3,5]		any integer	REAL*4
		MAX1[3]		REAL*4	INTEGER*4

Intrinsic Functions

Figure 2 (Page 5 of 6). Intrinsic Functions

Intrinsic Function	Generic Name	Specific Name	No. of Arguments	Type of Argument	Type and Range of Function
Choosing smallest value	MIN	MIN0[9] AMIN1 DMIN1 QMIN1	>= 2	any integer REAL*4 REAL*8 REAL*16	same as arg REAL*4 REAL*8 REAL*16
		AMIN0[3,5] MIN1[3]		any integer REAL*4	REAL*4 INTEGER*4
		Storage Functions			
Allocation status		ALLOCATED	1	any	LOGICAL*4
Location of variable		LOC	1	any	POINTER
		Character Manipulation Functions			
Conversion to type integer		ICHAR	1	CHARACTER*1	INTEGER*4
Conversion to type character		CHAR[5]	1	any integer	CHARACTER*1
Length		LEN	1	CHARACTER	INTEGER*4
Index of a substring		INDEX	2	CHARACTER	INTEGER*4
Lexically greater than or equal		LGE	2	CHARACTER	LOGICAL*4
Lexically greater than		LGT	2	CHARACTER	LOGICAL*4
Lexically less than or equal		LLE	2	CHARACTER	LOGICAL*4
Lexically less than		LLT	2	CHARACTER	LOGICAL*4
		Bit Manipulation Functions			
Inclusive or		IOR[5]	2	any integer or unsigned	same as arg

Figure 2 (Page 6 of 6). Intrinsic Functions

Intrinsic Function	Generic Name	Specific Name	No. of Arguments	Type of Argument	Type and Range of Function
Logical and		IAND[5]	2	any integer or unsigned	same as arg
Logical complement		NOT[5]	1	any integer or unsigned	same as arg
Exclusive or		IEOR[5] XOR[5]	2	any integer or unsigned	same as arg
Shift bits		ISHFT[5]	2	any integer or unsigned	same as arg
Shift left		LSHIFT[5]	2	any integer or unsigned	same as arg
Shift right		RSHIFT[5]	2	any integer or unsigned	same as arg
Shift circularly		ISHFTC[5]	2 or 3	any integer or unsigned	same as arg
Bit test		BTEST[4,5]	2	any integer or unsigned	LOGICAL*4
Bit set		IBSET[4,5]	2	any integer or unsigned	same as arg
Bit clear		IBCLR[4,5]	2	any integer or unsigned	same as arg
Bit extraction		IBITS[4,5]	3	any integer or unsigned	same as arg

Notes:

1. The generic name must be used for conversion functions when no specific name is supplied.
2. Also available in the alternate mathematical library, which provides alternative functions that provide results compatible with VS FORTRAN Version 1.
3. The specific name must be used to obtain a function value of this type.
4. The bits in bit-manipulation functions are numbered from right to left, beginning at zero.
5. This specific name is also a generic name when used with integer argument(s); it is specific only with INTEGER*4 argument(s). When there is more than one argument, all arguments must agree in length.

Interactive Debug Commands

Note that for parallel programs, you can only use the interactive debug to debug the root task and Fortran code that is not parallel. New types and other features introduced in VS FORTRAN Version 2 Release 6 are not supported by the Interactive Debug. See the summary of changes in the front of the *VS FORTRAN Version 2 Language and Library Reference* for a list of these features.

When using interactive debug commands:

- A statement number is either an ISN or a sequence number (columns 73-80), depending on how the program was compiled.
- Precede a statement label by a slash (/) when it is used in place of a statement number.
- Separate list items with commas or blanks (except for command lists).

Interactive Debug Command Categories

Controlling Program Processing

- AT
- ENDDEBUG
- GO
- HALT
- LISTBRKS
- NEXT
- OFF
- OFFWN
- RESTART
- STEP
- WHEN

Monitoring and Modifying Variables

- AUTOLIST
- DESCRIBE
- IF
- LIST
- QUALIFY
- SET

Processing Sequential Files

- BACKSPACE
- CLOSE
- ENDFILE
- RECONNECT
- REWIND

Controlling Full Screen Display

- COLOR
- DOWN
- LEFT
- LISTINGS
- MOVECURS
- POSITION
- PREVDISP
- PROFILE
- REFRESH

Controlling Full Screen Display (*continued*)

- RESTORE
- RETRIEVE
- RIGHT
- SEARCH
- SIZE
- UP
- WINDOW
- ZOOM

Handling Run Time Library Errors

- ERROR
- FIXUP

Gathering Vector Tuning Information

- LISTVEC
- VECSTAT

Tracing and Timing

- ANNOTATE
- LISTFREQ
- LISTSAMP
- LISTSUBS
- LISTTIME
- TIMER
- TRACE
- WHERE

General Commands

- * or " (Comments)
- DBCS
- HELP
- PURGE
- QUIT
- SYSCMD
- TERMIO

Interactive Debug Command Syntax

* or ' (Comments)

{***** | "}
[*comment*]

Inserts comments in the debugging log.

ANNOTATE Command—Copying Source Listings to a Print File

ANNOTATE
{*unit* | (*unit-list*) | ***** }
[**SAMPLING** [**DIRECT** | CALLED | ALL] | FREQUENCY]

Shows sampling or frequency data as a source listing to the AFFPRINT file.

ANNOTATE Command—Providing a Bar Chart in the Source Window

ANNOTATE
{ON | OFF | TOGGLE}
[**SAMPLING** [**DIRECT** | CALLED | ALL]
| FREQUENCY | MESSAGE]

Shows sampling or frequency data as a bar chart overlay on the source listing window.

Note: Valid in full screen mode only.

ANNOTATE Command—Querying the Settings

ANNOTATE

Queries the ANNOTATE settings.

Interactive Debug Command Syntax

AT Command

AT
[*qual.*]
{*number*[:[*qual.*]*number*] | ENTRY | EXIT}
| (*number*/ENTRY/EXIT *list*)
[(*command-list*)]
[COUNT(*n*)]
[**NOTIFY** | NONOTIFY]

Sets breakpoints.

Note: Separate individual commands in the command list with percent signs (%).

AUTOLIST Command

AUTOLIST
[{[*qual.*] *name* [:[*qual.*] *name*]
| * | '*string*' | *number* | (*list*)}
[FORMAT [(*code*)] | DUMP [(*code*)]]]

Automatically displays values of variables in the monitor window.

Notes:

1. Valid in full screen mode only.
2. Check the format and dump codes table on page 74.

BACKSPACE Command

BACKSPACE
{*number* | [*qual.*] *integer-variable* |
[*qual.*]*integer-array-element*}

Positions a sequentially accessed external file at the beginning of the previous record.

CLOSE Command

CLOSE
{*number* | [*qual.*] *integer-variable* |
[*qual.*]*integer-array-element*}

Disconnects a sequential external file from a unit.

Interactive Debug Command Syntax

COLOR Command

COLOR

Customizes color, highlighting, and intensity on the main debugging panel.

Note: Valid in full screen mode only.

DBCS Command

DBCS
    [YES | **NO**]

Specifies whether X'0F' and X'0E' are interpreted as the double-byte character set shift characters in input and output.

DESCRIBE Command

DESCRIBE
    {[*qual.*] *name* | ***** | (*name-list*)}
    [PRINT]

Displays data types of scalar variables and arrays, and dimension information for arrays.

DOWN Command

DOWN
    [*number* | PAGE | HALF | CSR | DATA | MAX]

Scrolls the contents of a window so that lines below those currently displayed in the window can be seen.

Note: Valid in full screen mode only.

ENDDEBUG Command

ENDDEBUG
    [SAMPLE[(*msecs*)]
    [MAXSAMP(*n*[,STOP])]
    [CALLED]]

Discontinues debugging and continues program processing. Also initiates program sampling.

ENDFILE Command

ENDFILE
{*number* | [*qual.*] *integer-variable* |
[*qual.*]*integer-array-element*}

Writes an end-of-file record on a sequentially accessed external file.

ERROR Command

ERROR
{*error* | *error*.*error* | (*error-list*)}
[**MSG** | NOMSG]
[**EXIT** | NOEXIT]

Selects diagnostic options for run-time errors.

FIXUP Command

FIXUP
[ARG1(*value*)]
[ARG2(*value*)]

Specifies corrective action.

GO Command

GO
[[*qual.*] {*number* | EXIT}]

Resumes processing.

HALT Command

HALT
[OFF | STMT | GOTO | ENTRY | **IMMED**]

Causes processing to be suspended for every statement of a given class, or at a specific point in a command list.

Interactive Debug Command Syntax

HELP Command—CMS or TSO Full Screen Mode

HELP
[*command* | *vecmsg-id*]

Requests online information about interactive debug commands, common tasks, and vector messages contained in the vector report listing, as well as a task-oriented tutorial.

HELP Command—CMS Line Mode

HELP
[*command* [**ALL** | (DESC | (PARM | (FORM]
| *vecmsg-id*]

Requests online information about interactive debug commands, common tasks, and vector messages contained in the vector report listing, as well as a task-oriented tutorial.

HELP Command—TSO Line Mode

HELP
[*command* [**ALL** | FUNCTION | SYNTAX
| OPERANDS [(*keyword-list*)]] | *vecmsg-id*]

Requests online information about interactive debug commands, common tasks, and vector messages contained in the vector report listing, as well as a task-oriented tutorial.

IF Command

IF
(*condition*) *command*

Tests a condition.

LEFT Command

LEFT
[*number* | PAGE | HALF | CSR | DATA | MAX]

Scrolls the contents of a window so that columns to the left of those currently displayed in the window can be seen.

Note: Valid in full screen mode only.

Interactive Debug Command Syntax

LIST Command

LIST
{[*qual.*] *name* [: [*qual.*] *name*]
| * | '*string*' | *number* | (*list*)}
[PRINT]
[FORMAT [(*code*)] | DUMP [(*code*)]]

Displays values of variables.

Note: Check the format and dump codes table on page 74.

LISTBRKS Command

LISTBRKS [PRINT]

Lists all breakpoints and WHEN conditions currently set, and the current HALT status.

LISTFREQ Command

LISTFREQ
[[*qual.*]
{*number* [: [*qual.*] *number*] | ENTRY | EXIT}
| (number/ENTRY/EXIT list)]
[ZEROFREQ] [PRINT]

Lists the number of times statements processed.

LISTINGS Command

LISTINGS

Displays the interactive debug listings panel.

Note: Valid in full screen mode only.

Interactive Debug Command Syntax

LISTSAMP Command—Statement

LISTSAMP
{{*qual.*] *number*[:[*qual.*]*number*]
| [*qual.*]ENTRY | [*qual.*]***** | (*list*) | ***.***}
[**DIRECT**][CALLED][ALL]
[TOP[(*n*)]][PRINT]

Lists sampling counts by statement.

LISTSAMP Command—Program Unit

LISTSAMP
{*unit-name* | (*unit-name-list*) | ***** } SUMMARY
[**DIRECT** | CALLED | ALL]
[TOP[(*n*)]][PRINT]

Lists sampling counts by program unit.

LISTSAMP Command—DO Loop

LISTSAMP
{[*qual.*] *number* [:[*qual.*] *number* |
[*qual.*] ***** | (*list*) | ***.***}
DOLOOP [**DIRECT** | CALLED | ALL]
[TOP [(*n*)]] [PRINT]

Lists sampling counts by DO loop.

LISTSUBS Command

LISTSUBS
[PRINT]

Lists information about all debuggable program units in the running load module.

LISTTIME Command—Program Unit

LISTTIME
[PRINT]

Displays timing information for program units.

LISTTIME Command—DO Loop

LISTTIME
{ [*qual.*] *number* [:[*qual.*] *number*] |
[*qual.*] * | (*list*) | *.* }
DOLOOP [PRINT]

Displays timing information for analyzable DO loops.

LISTVEC Command

LISTVEC
{ [*qual.*] *number* [:[*qual.*] *number*] |
[*qual.*] * | (*list*) | *.* }
[TOP [(*n*)] [PRINT]

Displays DO loop length and stride information.

MOVECURS Command

MOVECURS

Toggles the cursor between the command line and its most recent position in the main debugging panel.

Note: Valid in full screen mode only.

NEXT Command

NEXT

Suspends program execution at the next statement, entry, or exit with a debugging hook.

OFF Command

OFF
[*qual.*]{ *number* [: [*qual.*] *number*] |
ENTRY | EXIT} | * |
(*number*/ENTRY/EXIT *list*)

Removes breakpoints in the currently qualified or specified program unit.

Interactive Debug Command Syntax

OFFWN Command

OFFWN
condition name | ***** | (*condition-name-list*)

Turns off WHEN condition monitoring.

POSITION Command

POSITION
number

Positions the cursor in the log window at a specified log line, in the source window at a given ISN or sequence number, or in the monitor window at a specified monitor line.

Note: Valid in full screen mode only.

PREVDISP Command

PREVDISP

Redisplays the previous panel displayed by the application program.

Note: Valid in full screen mode only.

PROFILE Command

PROFILE

Displays a profile panel to change current conditions or profile settings.

Note: Valid in full screen mode only.

PURGE Command

PURGE

Purges output.

QUALIFY Command

QUALIFY
[*program*]

Changes or displays the current qualification.

QUIT Command

QUIT

Ends the debugging session.

RECONNECT Command

RECONNECT
{*number* | [*qual.*]*integer-variable* |
[*qual.*]*integer-array-element*}

Resets a file to its original (preconnected) condition.

REFRESH Command

REFRESH
[**ON** | OFF]

Controls whether the IAD panel is refreshed.

Note: Valid in full screen mode only.

RESTART Command

RESTART

Restarts the debugging session while maintaining the log file.

Note: Valid in full screen mode only.

RESTORE Command

RESTORE

Restores the source window to the last point of execution.

Note: Valid in full screen mode only.

RETRIEVE Command

RETRIEVE

Redisplays the last command specified on the command line.

Note: Valid in full screen mode only.

Interactive Debug Command Syntax

REWIND Command

REWIND
{*number* | [*qual.*] *integer-variable*
| [*qual.*]*integer-array-element*}

Positions a sequentially accessed external file at the beginning of its first record.

RIGHT Command

RIGHT
[*number* | PAGE | HALF | CSR | DATA | MAX]

Scrolls the contents of a window so that columns to the right of those currently displayed in the window can be seen.

SEARCH Command

SEARCH
[/*string*[/]]

Searches either the source, monitor, or log window for a given character string.

Note: Valid in full screen mode only.

SET Command

SET
[*qual.*] *name*=*value* [,*value*...]

Assigns values to variables.

SIZE Command

SIZE
[SOURCE | MONITOR | LOG]

Resizes the windows on the main debugging panel.

Note: Valid in full screen mode only.

Interactive Debug Command Syntax

STEP Command

STEP
[*number*]

Processes one or more statements, then suspends processing. In full screen mode, processing is animated.

SYSCMD Command

SYSCMD
[*system-command*]

Issues system commands during debugging.

Note: The abbreviations CMS and TSO are recognized and executed, but the command entered does not appear in the session log.

TERMIO Command

TERMIO
[**IAD** | LIBRARY]
[MSG [(*userid*)] | **NOMSG**]

Selects I/O routines for terminal I/O from the VS FORTRAN program.

TIMER Command—Program Unit

TIMER
{ ***** | *program-unit-name* | (*program-unit-name-list*) }
[**ON** | OFF | RESET]

Controls timing by program unit.

TIMER Command—DO Loop

TIMER
{[*qual.*] *number* [:*qual.*] *number*
| [*qual.*] ***** | *(list)* | *****.*****}
DOLOOP [**ON** | OFF | RESET]

Controls timing by DO loop.

Interactive Debug Command Syntax

TRACE Command

TRACE
    [GOTO | ENTRY | **OFF**]
    [PRINT]

Traces statement branches and subprogram calls.

UP Command

UP
    [*number* | PAGE | HALF | CSR | DATA | MAX]

Scrolls the contents of a window so that lines above those currently displayed in the window can be seen.

Note: Valid in full screen mode only.

VECSTAT Command

VECSTAT
    {[*qual.*] *number* [:*qual.*] *number*
    | [*qual.*] ***** | *(list)* | *****.*****}
    [**ON** | OFF | RESET]

Activates, deactivates, or resets DO loop length and stride recording.

WHEN Command

WHEN
    condition-name [(*condition*) | *variable*]

Sets up monitoring of a condition.

WHERE Command

WHERE
    [TRBACK] [FLOW] [PRINT]

Displays statement at which processing is suspended.

Interactive Debug Command Syntax

WINDOW Command for Changing Configuration

WINDOW

Changes the window configuration of the debugging session.

Note: Valid in full screen mode only.

WINDOW Command for Saving Configuration

WINDOW
    SAVE

Saves the window configuration of the debugging session.

Note: Valid in full screen mode only.

WINDOW Command for Opening or Closing

WINDOW
    {OPEN | CLOSE}
    {SOURCE | MONITOR | LOG}]

Opens or closes a specified window of the debugging session.

Note: Valid in full screen mode only.

ZOOM Command

ZOOM
    [SOURCE | MONITOR | LOG]

Toggles between displaying one window and a configuration of windows.

Format and Dump Codes for the AUTOLIST and LIST Commands

Code	Output
L1	LOGICAL*1
L4	LOGICAL*4
I2	INTEGER*2
I4	INTEGER*4
R4	REAL*4
R8	REAL*8
R16	REAL*16
C8	COMPLEX*8
C16	COMPLEX*16
C32	COMPLEX*32
L	LOGICAL with size closest to internal data size
I	INTEGER with size closest to internal data size
R	REAL with size closest to internal data size
C	COMPLEX with size closest to internal data size
X[nnn]	Hexadecimal with nnn bytes per data item (default to internal data size)
Z[nnn]	Hexadecimal with nnn bytes per data item (default to Z4)
A[nnn]	CHARACTER with nnn bytes per data item (default to internal data size)
H[nnn]	CHARACTER with nnn bytes per data item (default to continuous full line output)

Valid SET Command Assignments

Note that values must be those allowable for the type of variable being set.

Variable	Type of Value Set to	Example
Scalar (REAL, INTEGER, or COMPLEX)	Another scalar variable	ALPHA=BETA ALPHA=–BETA
	An array element	ALPHA=A(3) ALPHA=–A(3)
	A numeric constant	NUM=7
LOGICAL	Another logical variable An array element	LOG1=LOG2
	A logical constant	LOG1=LOG(2) LOG=.TRUE.
CHARACTER	Another character variable An array element	CHAR1=CHAR2
	A character constant	CHAR1=CHAR(2)
	A substring	MSG='HELLO' A(1:3)='ABC'
Array element	Another array element	A(4)=B(1) AR(2,2)=–AR(5,5)
	A scalar variable	C(7)=RATE C(8)=–TIME
	A constant	D(I,J)=0.0
Contiguous array elements	Value, value,... (Values can be variables, array elements, or constants; multiple assignments of a value can be entered as n*value.)	A=3*1.0,4*0.0,7.2, 5.,ACCL,8.5E9 B(J,K)='C', 3*'Q','X'

We'd Like to Hear from You

VS FORTRAN Version 2
Reference Summary
Release 6

Publication No. SX26-3751-07

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

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

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

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

IBM may use or distribute your comments without obligation.

Readers' Comments

VS FORTRAN Version 2
Reference Summary
Release 6

Publication No. SX26-3751-07

How satisfied are you with the information in this book?

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

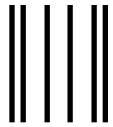
Please tell us how we can improve this book:

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

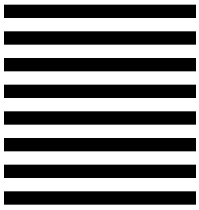
Name _____ Address _____

Company or Organization

Phone No.



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

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

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

Readers' Comments
SX26-3751-07

File Number: S370-40
Program Number: 5668-805
5668-806
5688-087

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

SX26-3751-07

Spine information:

VS FORTRAN Version 2 Reference Summary *Release 6*