SC26-3989-0
File No. S370-25

Program Product

VS FORTRAN Application Programming: Library Reference

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

Release 1

IBM

First Edition (February 1981)

This edition, as amended by technical newsletter SN26-0852, applies to Release 1 of VS FORTRAN, Program Products 5748-FO3 (Compiler and Library) and 5748-LM3 (Library only), and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Information concerning the IBM 3375 and 3380 direct access devices is for planning purposes only until the availability of the devices.

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

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

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

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

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

Preface

This publication describes the mathematical and service subprograms in the VS FORTRAN libraries supplied with VS FORTRAN. To aid the programmer in the use of this publication, a brief description of each chapter follows:

1. "Introduction" describes the four types of subprograms in the VS FORTAN library (VSFORTL) and defines their use in either a VS FORTRAN or an assembler language program.
2. "Mathematical and Character Subprograms" describes the subprograms which perform computations and conversions frequently needed by the programmer. A mathematical or character subprogram is invoked *explicitly* whenever one of its entry names appears in a source statement, or *implicitly* through use of certain notations in the source statements.
3. "Service Subprograms" contains information about those subprograms which perform utility functions.
4. "Algorithms" contains information about the method used in the library to compute a mathematical function and describes the effect of an argument error upon the accuracy of the answer returned.
5. "Accuracy Statistics" gives accuracy statistics for the explicitly called mathematical subprograms.
6. "Appendixes" provides a list of diagnostic messages, a list of module names, a sample storage printout, storage estimates, and information for the assembler language programmer.

Standard and mathematical notation is used in this manual. The reader should be familiar with this notation and with common mathematical terminology.

Industry Standards

The VS FORTRAN Compiler and Library program product is designed according to the specifications of the following industry standards, as understood and interpreted by IBM as of June, 1980:

American National Standard Programming Language FORTRAN, ANSI X3.9-1978 (also known as FORTRAN 77)

International Organization for Standardization ISO 1539-1980 Programming Languages-FORTRAN

These two standards are technically equivalent. In this manual, references to the *current standard* are references to the above two standards.

American Standard FORTRAN, X3.9-1966

International Organization for Standardization ISO R 1539-1972 Programming Languages-FORTRAN

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

Both the FORTRAN 77 and the FORTRAN 66 standard languages include IBM extensions. In this book, references to *current FORTRAN* are references to the FORTRAN 77 standard plus the IBM extensions valid with it; references to *old FORTRAN* are references to the FORTRAN 66 standard plus the IBM extensions valid with it.

VS Fortran Publications

The VS FORTRAN publications are designed to help you develop your programs with a minimum of wasted effort. This book, the *VS FORTRAN Application Programming: Library Reference,* gives you detailed information about the execution-time library subroutines.

A series of related publications give you detailed reference documentation you can use when you are actually performing the tasks this manual describes:

VS FORTRAN Application Programming:

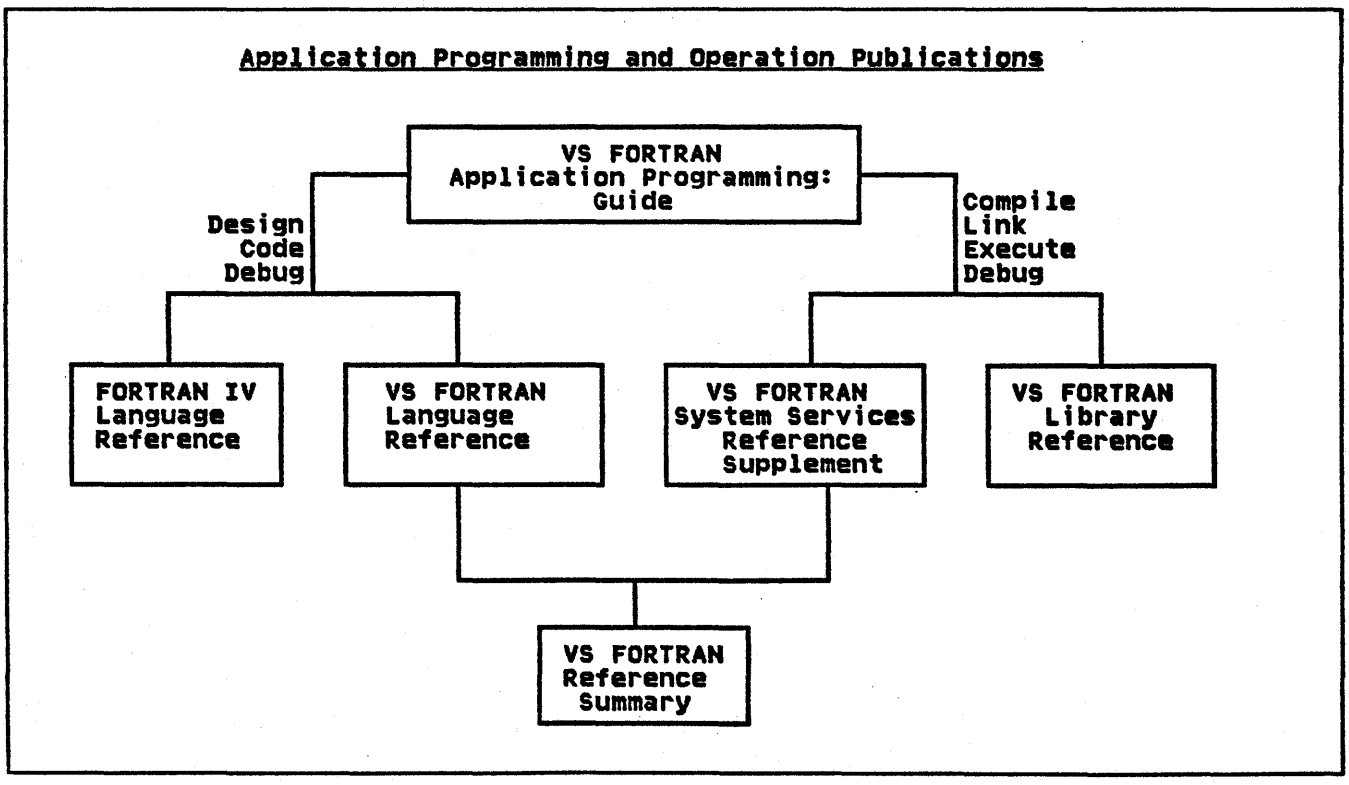
Application Programming Guide, SC26-3985, contains guidance information on designing, coding, debugging, testing, and executing VS FORTRAN programs written at the *current language* level.

Language Reference, GC26-3986, gives you the semantic rules for coding VS FORTRAN programs when you're using current FORTRAN.

System Services Reference Supplement, SC26-3988, gives you FORTRAN-specific reference documentation for the system your programs will operate under.

Source-Time Reference Summary, SX26-3731, is a pocket-sized reference card containing current FORTRAN syntax and brief descriptions of the compiler options.

System/360 and System/370 FORTRAN IV Language manual, GC28-6515, gives you the rules for writing VS FORTRAN programs when you're using old FORTRAN.



Related Publications

The FORTRAN programmer using the VS FORTRAN compiler and library should be familiar with the information in the following publications:

- *VS FORTRAN Language Reference,* GC26-3986
- *VS FORTRAN Application Programming Guide,* SC26-3985
- *VS FORTRAN System Services Reference Supplement,* SC26-3988
- *VS FORTRAN Compiler and Library Diagnosis,* SC26-3990

Information about IBM-supplied utility programs can be found in the following publications:

- *OS/VS1 Utilities,* GC26-3901
- *OS/VS2 Utilities Manual,* GC26-3902
- *DOS/VSE Utilities Manual,* GC33-5381

Information about the linkage editor and loader programs can be found in the following publications:

- *OS/VS Linkage Editor and Loader,* GC26-3813
- *Guide to the DOS/VSE Assember,* GC33-6077

Information about data management can be found in the following publications:

- *OS/VS1 Supervisor Services and Macro Instructions,* GC28-6670
- *DOS/VSE Data Management Concepts,* GC24-5138
- *DOS/VSE Macro User's Guide,* GC24-5139

Information about assembler language programming can be found in the following publications:

- *OS/VS and DOS/VS, VM/370 Assembler Language,* GC33-4010
- *OS/VS, VM/370 Assembler Programmer's Guide,* GC33-4021

Information about System/370 machine characteristics can be found in the following publication:

- *IBM System/370 Principles of Operation,* GA22-7000

SUMMARY OF AMENDMENTS

3 JUNE 1981

MISCELLANEOUS CHANGES

- "Compare of complex numbers" has been added to the chart Implicitly Called Mathematical Subprogams.
- Storage estimates have been updated.
- VSAM execution error messages have been included; other messages have been changes.
- A reentrant library module list has been added.
- A list of figures has been added.

Contents

FIGURES

Introduction

The VS FORTRAN library contains the following types of subprograms: (1) mathematical and character functions, (2) service subprograms, (3) input/output subprograms, and, (4) error handling subprograms. The mathematical and service subprograms handle single, double and extended-precision arguments. Character subprograms handle conversation between character and integer data.

The library subprograms may be used in either a FORTRAN or an assembler language program. (Appendix B contains calling information for the assembler language programmer.) In VS FORTRAN, calls to the library subprograms are either at the programmer's request or in response to the program requirements. Subprograms required by the program being compiled are provided by the linkage editor or loader, which takes the subprograms from the library.

Mathematical and Character Functions

These routines provide commonly used mathematical and character functions. When VS FORTRAN requests a function, the routine is either:

Inline Inserted into the program during compilation, or

Out-of-Line Included in the load module as a called subprogram during link editing.

This publication discusses both out-of-line and inline routines. The American National Standard Institute (ANSI) defines several arithmetic functions, such as absolute value (ABS), positive difference (DIM), and transfer of sign (SIGN) as *intrinsic functions.* For the most part, code for these functions is inserted inline by the FORTRAN compiler at the point in the source module where the function's name is used. However, the following ANSI-defined *intrinsic functions* have been implemented as part of the VS FORTRAN library and are provided out-of-line for all systems:

- INDEX
- LGE/LGT
- LLE/LLT
- MAX/MIN
- MAX0/MIN0
- AMAX0/AMIN0
- MAX1/MIN1
- AMAX1/AMIN1
- DMAX1/DMIN1
- QMAX1/QMIN1
- MOD/AMOD/AINT
- IFIX/INT/IDINT

Service Subroutine Subprograms

Each of the service subprograms corresponds to a subroutine form as defined by a SUBROUTINE statement in a FORTRAN source module. These subprograms perform mathematical exception test and utility functions. They may or may not return a value to the calling module.

Input/Output and Error Processing Routines

The library contains certain input/output and error processing routines that act as interfaces with the compiled program and operating system. Frequently, the mathematical and service functions require assistance from these routines for input/output, interruption, and error processing. Storage estimates for these routines are included in Appendix B.

Mathematical and Character Subprograms

The mathematical and character subprograms supplied in the VS FORTRAN library perform computations and conversions needed by the programmer. Mathematical and character subprograms are called in two ways: explicitly, when the programmer includes the appropriate entry name in a source language statement (see Figure 1); and implicitly, when certain notation (for example, raising a number to a power) appears within a source language statement (see Figure 1).

The following text describes the individual mathematical and character subprograms and explains their use in a VS FORTRAN program. Detailed information about the actual method of computations used in each subprogram, error messages, and storage estimates are discussed later in this publication.

General Function	Specific Function	Entry Name(s)
Logarithmic and exponential subprograms (described in Figure 2)	Exponential	EXP DEXP CEXP CDEXP QEXP CQEXP
	Logarithmic, common and natural	ALOG, ALOG10 DLOG, DLOG10 CLOG CDLOG QLOG, QLOG10 CQLOG
	Square root	SQRT DSQRT CSORT CDSQRT QSQRT CQSQRT
Trigonometric subprograms (described in Figure 3)	Arcsine and arccosine	ARSIN, ARCOS DARSIN, DARCOS ASIN, ACOS DASIN, DACOS QARSIN, QARCOS
	Arctangent	ATAN, ATAN2 DATAN, DATAN2 QATAN QATAN2
	Sine and cosine	SIN, COS DSIN, DCOS CSIN, CCOS CDSIN, CDCOS QSIN, QCOS CQSIN, CQCOS
	Tangent and cotangent	TAN, COTAN DTAN, DCOTAN QTAN, QCOTAN
Hyperbolic function subprograms (described in Figure 4)	Hyperbolic sine and cosine	SINH, COSH DSINH, DCOSH QSINH, QCOSH
	Hyperbolic tangent	TANH DTANH QTANH
Character Manipulation Routines (see Figure 6)	Convert integer to character	CHAR
	Convert character to integer	ICHAR
	Length of character item	LEN
	Index of character item	INDEX
	Alphamerically greater than or equal	LGE
	Alphamerically greater than	LGT
	Alphamerically less than or equal	LLE
	Alphamerically less than	LLT

Figure 1. *(Part 1 of 2)* Explicitly Called Mathematical and Character Subprograms

General Function	Specific Function	Entry Name(s)
Miscellaneous subprograms (described in Figure 7)	Absolute value	CABS CDABS CQABS IABS ABS DABS QABS
	Error function	ERF, ERFC DERF, DERFC QERF, QERFC
	Gamma and log-gamma	GAMMA, ALGAMA DGAMMA, DLGAMA LGAMMA
	Maximum and minimum value	AMAX0, AMIN0, MAX0, MIN0 AMAX1, AMIN1, MAX1, MIN1 DMAX1, DMIN1 MAX, MIN
	Modular arithmetic	MOD AMOD, DMOD QMOD
	Truncation	AINT INT, IDINT
	Imaginary Part of Complex Argument	IMAG, AIMAG DIMAG QIMAG
	Conjugate of a Complex Number	CONJG DCONJG, QCONJG
	Obtain Nearest Whole Number	ANINT DNINT
	Obtain Nearest Integer	NINT IDNINT
	Obtain Positive Difference	DIM, IDIM DDIM, QDIM
	Transfer of Sign	SIGN, ISIGN DSIGN QSIGN
	Obtain Double Precision Product	DPROD

Figure 1. ***(Part 2 of 2)*** **Explicity Called Mathematical and Character Subprograms**

Explicitly Called Subprograms

Each explicitly called subprogram performs one or more mathematical or character functions. Each mathematical and character function is identified by a unique entry name.

A subprogram is called whenever the appropriate entry name is included in a VS FORTRAN arithmetic or character expression. The programmer must supply one or more arguments. These arguments are separated by commas, the list of arguments is enclosed in parentheses, following the entry name.

For example, the source statement:

```
RESULT = SIN (RADIAN)
```

causes the sine subprogram to be called. The sine of the value in RADIAN is computed and the function value is stored in RESULT.

In the following example, the square root subprogram is called to compute the square root of the value in AMNT. The function value is then added to the value in STOCK and the result is stored in ANS.

```
ANS = STOCK + SQRT (AMNT)
```

The explicitly called subprograms are described in Figures 2 through 5. The following information is provided:

General Function: This column states the nature of the computation performed by the subprogram.

Entry Name: This column gives the entry name that the programmer must use to call the subprogram. A subprogram may have more than one entry name; the particular entry name used depends on the computation to be performed. For example, the sine and cosine subprogram has two entry names: SIN and COS. If the sine is to be computed, entry name SIN is used; if the cosine is to be computed, entry name COS is used.

Definition: This column gives a mathematical equation that represents the computation. An alternate equation is given in those cases where there is another way of representing the computation in mathematical notation. For example, the square root can be represented either as: $y = \sqrt{x}$ or $y = x^{1/2}$.

Argument Numbers: This column gives the number of arguments that the programmer must supply.

Argument Type: This column describes the type and length of each of the argument(s). INTEGER, REAL, COMPLEX, and Character represent the type; the notations *4, *8, *16, *32, and *n represent the size of the argument in number of storage locations. (The notation *n describes character data.)

IBM EXTENSION

Argument Range: This column gives the valid range for arguments. If an argument is not within this range, an error message is issued. For a description of the error messages see Appendix C of this publication.

Function Value Type and Range: This column describes the type and range of the function value returned by the subprogram. Type notation used is the same as that used for the argument type. The range symbol $\gamma = 16^{63}(1 - 16^{-6})$ for regular precision routines, $16^{63}(1 - 16^{-14})$ for double-precision and $\gamma = 16^{63}(1 - 16^{-28})$ for extended-precision.

END OF IBM EXTENSION

Error Code: This column gives the number of the message issued when an error occurs. Appendix C contains descriptions of the error messages.

Throughout this publication, the following approximate values are represented by $2^{18} \cdot \pi$ and $2^{50} \cdot \pi$:

$$2^{18} \cdot \pi = .8235496645826428D + 06$$
$$2^{50} \cdot \pi = .3537118876014220D + 16$$

Detailed information for the assembler language programmer is given in Appendix A.

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[4]	Error Code
			No.	Type[1]	Range		
Common and natural logarithm	ALOG	$y=\log_e x$ or $y=\ln x$	1	REAL *4	$x > 0$	REAL *4 $y \geqq -180.218$ $y \leqq 174.673$	253
	ALOG10	$y=\log_{10} x$	1	REAL *4	$x > 0$	REAL *4 $y \geqq -78.268$ $y \leqq 75.859$	253
	DLOG	$y=\log_e x$ or $y=\ln x$	1	REAL *8	$x > 0$	REAL *8 $y \geqq -180.218$ $y \leqq 174.673$	263
	DLOG10	$y=\log_{10} x$	1	REAL *8	$x > 0$	REAL *8 $y \geqq -78.268$ $y \leqq 75.859$	263
	CLOG	$y=PV \log_e (z)$ See Note 2	1	COMPLEX *8	$z \neq 0 + 0i$	COMPLEX *8 $y_1 \geqq -180.218$ $y_1 \leqq 175.021$ $-\pi \leqq y_2 \leqq \pi$	273
	CDLOG	$y=PV \log_e (z)$ See Note 2	1	COMPLEX *16	$z \neq 0 + 0i$	COMPLEX *16 $y_1 \geqq -180.218$ $y_1 \leqq 175.021$ $-\pi \leqq y_2 \leqq \pi$	283
	QLOG	$y=\log_e x$ or $y=\ln x$	1	REAL *16	$x>0$	REAL *16 $y \geqq -180.218$ $y \leqq 174.673$	293
	QLOG10	$y=\log_{10} x$	1	REAL *16	$x>0$	REAL *16 $y \geqq -78.268$ $y \leqq 175.859$	293
	CQLOG	$y=PV \log_e (z)$ See Note 2	1	COMPLEX *32	$z \neq 0 + 0i$	COMPLEX *32 $y_1 \geqq -180.218$ $y_1 \leqq 175.021$ $-\pi \leqq y_2 \leqq \pi$	278
Exponential	EXP	$y=e^x$	1	REAL *4	$x \leqq 174.673$	REAL *4 $0 \leqq y \leqq \gamma$	252
	DEXP	$y=e^x$	1	REAL *8	$x \leqq 174.673$	REAL *8 $0 \leqq y \leqq \gamma$	262
	CEXP	$y=e^z$ See Note 3	1	COMPLEX *8	$x_1 \leqq 174.673$ $\lvert x_2 \rvert < (2^{18} \cdot \pi)$	COMPLEX *8 $-\gamma \leqq y_1, y_2 \leqq \gamma$	271, 272
	CDEXP	$y=e^z$ See Note 3	1	COMPLEX *16	$x_1 \leqq 174.673$ $\lvert x_2 \rvert < (2^{50} \cdot \pi)$	COMPLEX *16 $-\gamma \leqq y_1, y_2 \leqq \gamma$	281, 282
	QEXP	$y=e^x$	1	REAL *16	$x \geqq -180.218$ $x \leqq 174.673$	REAL *16 $0 \leqq y \leqq \gamma$	292
	CQEXP	$y=e^x$ See Note 3	1	COMPLEX *32	$x_1 \leqq 174.673$ $x_2 \leqq 2^{100}$	COMPLEX *32 $-\gamma \leqq y_1, y_2 \leqq \gamma$	276, 277
NOTES: (See end of figure.)							

Figure 2. *(Part 1 of 2)* Logarithmic and Exponential Subprograms

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[4]	Error Code
			No.	Type[1]	Range		
Square root	SQRT	$y=\sqrt{x}$ or $y=x^{1/2}$	1	REAL *4	$x \geqq 0$	REAL *4 $0 \leqq y \leqq \gamma^{1/2}$	251
	DSQRT	$y=\sqrt{x}$ or $y=x^{1/2}$	1	REAL *8	$x \geqq 0$	REAL *8 $0 \leqq y \leqq \gamma^{1/2}$	261
	CSQRT	$y=\sqrt{z}$ or $y=z^{1/2}$ See Note 3	1	COMPLEX *8	any COMPLEX argument	COMPLEX *8 $0 \leqq y_1 \leqq 1.0987\,(\gamma^{1/2})$ $\lvert y_2\rvert \leqq 1.0987\,(\gamma^{1/2})$	—
	CDSQRT	$y=\sqrt{z}$ or $y=z^{1/2}$ See Note 3	1	COMPLEX *16	any COMPLEX argument	COMPLEX *16 $0 \leqq y_1 \leqq 1.0987\,(\gamma^{1/2})$ $\lvert y_2\rvert \leqq 1.0987\,(\gamma^{1/2})$	—
	QSQRT	$y=\sqrt{x}$ or $y=x^{1/2}$	1	REAL *16	$x \geqq 0$	REAL *16 $0 \leqq y \leqq \gamma^{1/2}$	289
	CQSQRT	$y=\sqrt{z}$ or $y=z^{1/2}$ See Note[3]	1	COMPLEX *32	any COMPLEX argument	COMPLEX *32 $0 \leqq y_1 \leqq 1.0987\,(\gamma^{1/2})$ $y_2 \leqq 1.0987\,(\gamma^{1/2})$	

NOTES:

[1] REAL *4, REAL *8, and REAL *16 arguments correspond to REAL DOUBLE PRECISION and EXTENDED PRECISION arguments, respectively, in VS FORTRAN.

[2] PV = principal value. The answer given $(y_1 + y_2 i)$ is that one whose imaginary part (y_2) lies between $-\pi$ and $+\pi$. More specifically: $-\pi < y_2 \leqq \pi$, unless $x_1 < 0$ and $x_2 = -0$, in which case, $y_2 = -\pi$.

[3] z is a complex number of the form $x_1 + x_2 i$.

[4] $\gamma = 16^{63}\,(1 - 16^{-6})$ for regular precision routines, $16^{63}\,(1 - 16^{-14})$ for double precision routines, and $16^{63}\,(1 - 16^{-28})$ for extended precision.

Figure 2 *(Part 2 of 2)* Logarithmic and Exponential Subprograms

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[4]	Error Code
			No.	Type[1]	Range		
Arcsine and arccosine	ASIN	See ARSIN, DARSIN or QARSIN					
	ACOS	See ARCOS, DARCOS or QARCOS					
	ARSIN	y=arcsin (x)	1	REAL *4	$\|x\| \leqq 1$	REAL *4 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	257
	ARCOS	y=arccos (x)	1	REAL *4	$\|x\| \leqq 1$	REAL *4 (in radians) $0 \leqq y \leqq \pi$	257
	DASIN	See DARSIN					
	DACOS	See DARCOS					
	DARSIN	y=arcsin (x)	1	REAL *8	$\|x\| \leqq 1$	REAL *8 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	267
	DARCOS	y=arccos (x)	1	REAL *8	$\|x\| \leqq 1$	REAL *8 (in radians) $0 \leqq y \leqq \pi$	267
	QARSIN	y=arcsin(x)	1	REAL *16	$\|x\| \leqq 1$	REAL *16 $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	297
	QARCOS	y=arcos(x)	1	REAL *16	$\|x\| \leqq 1$	REAL *16 $0 \leqq y \leqq \pi$	297
Arctangent	ATAN	y=arctan (x)	1	REAL *4	any REAL argument	REAL *4 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	—
	ATAN2	$y=\arctan\left(\frac{x_1}{x_2}\right)$	2	REAL *4	any REAL arguments (except 0, 0)	REAL *4 (in radians) $-\pi < y \leqq \pi$	255
	DATAN	y=arctan (x)	1	REAL *8	any REAL argument	REAL *8 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	—
	DATAN2	$y=\arctan\left(\frac{x_1}{x_2}\right)$	2	REAL *8	any REAL arguments (except 0, 0)	REAL *8 (in radians) $-\pi < y \leqq \pi$	265
	QATAN	y=arctan(x)	1	REAL *16	any REAL argument	REAL *16 (in radians) $-\frac{\pi}{2} \leqq y \leqq \frac{\pi}{2}$	
	QATAN2	$y=\arctan\left(\frac{x_1}{x_2}\right)$	2	REAL *16	any REAL arguments (except 0,0)	REAL *16 (in radians) $-\pi < y \leqq \pi$	295
NOTES: (See end of table.)							

Figure 3. *(Part 1 of 2)* Trigonometric Subprograms.

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[4]	Error Code
			No.	Type[1]	Range		
Sine and cosine	SIN	y=sin (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$	REAL *4 $-1 \leqq y \leqq 1$	254
	COS	y=cos (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$	REAL *4 $-1 \leqq y \leqq 1$	254
	DSIN	y=sin (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$	REAL *8 $-1 \leqq y \leqq 1$	264
	DCOS	y=cos (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$	REAL *8 $-1 \leqq y \leqq 1$	264
	CSIN	y=sin (z) See Note 2	1	COMPLEX *8 (in radians)	$\|x_1\| < (2^{18} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *8 $-\gamma \leqq y_1, y_2 \leqq \gamma$	274, 275
	CCOS	y=cos (z) See Note 2	1	COMPLEX *8 (in radians)	$\|x_1\| < (2^{18} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *8 $-\gamma \leqq y_1, y_2 \leqq \gamma$	274, 275
	CDSIN	y=sin (z) See Note 2	1	COMPLEX *16 (in radians)	$\|x_1\| < (2^{50} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *16 $-\gamma \leqq y_1, y_2 \leqq \gamma$	284, 285
	CDCOS	y=cos (z) See Note 2	1	COMPLEX *16 (in radians)	$\|x_1\| < (2^{50} \cdot \pi)$ $\|x_2\| \leqq 174.673$	COMPLEX *16 $-\gamma \leqq y_1, y_2 \leqq \gamma$	284, 285
	QSIN	y=sin(x)	1	REAL *16 (in radians)	$\|x\| < 2^{100}$	REAL *16 $-1 \leqq y \leqq 1$	294
	QCOS	y=cos(x)	1	REAL *16 (in radians)	$\|x\| < 2^{100}$	REAL *16 $-1 \leqq y \leqq 1$	294
	CQSIN	y=sin(z) See Note 2	1	COMPLEX *32 (in radians)	$\|x_1\| < 2^{100}$ $\|x_2\| \leqq 174.673$	COMPLEX *32 $-\gamma \leqq y_1, y_2 \leqq \gamma$	279, 280
	CQCOS	y=cos(z) See Note 2	1	COMPLEX *32 (in radians)	$\|x_1\| < 2^{100}$ $\|x_2\| \leqq 174.673$	COMPLEX *32 $-\gamma \leqq y_1, y_2 \leqq \gamma$	279, 280
Tangent and cotangent	TAN	y=tan (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$ See Note 4	REAL *4 $-\gamma \leqq y \leqq \gamma$	258, 259
	COTAN	y=cotan (x)	1	REAL *4 (in radians)	$\|x\| < (2^{18} \cdot \pi)$ See Note 4	REAL *4 $-\gamma \leqq y \leqq \gamma$	258, 259
	DTAN	y=tan (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$ See Note 4	REAL *8 $-\gamma \leqq y \leqq \gamma$	268, 269
	DCOTAN	y=cotan (x)	1	REAL *8 (in radians)	$\|x\| < (2^{50} \cdot \pi)$ See Note 4	REAL *8 $-\gamma \leqq y \leqq \gamma$	268, 269
	QTAN	y=tan(x)	1	REAL *16 (in radians)	$\|x\| < 2^{100}$ See Note 3	REAL *16 $-\gamma \leqq y \leqq \gamma$	298, 299
	QCOTAN	y=cotan(x)	1	REAL *16 (in radians)	$\|x\| < 2^{100}$ $\|x\| \geqq 16^{-63}$ See Note 3	REAL *16 $-\gamma \leqq y \leqq \gamma$	298, 299

NOTES:

[1] REAL *4, REAL *8, and REAL *16 correspond to REAL, DOUBLE and EXTENDED PRECISION arguments, respectively, in VS FORTRAN.

[2] z is a complex number of the form $x_1 + x_2 i$.

[3] x may not be such that one can find a singularity within 8 units of the last digit value of the floating-point representation of x. Singularities are $\pm (2n + 1)\frac{\pi}{2}$, n = 0, 1, 2, . . . for tangent, and $\pm n\,\pi$, n = 0, 1, 2, . . . for cotangent.

[4] The argument for the cotangent functions may not approach a multiple of π; the argument for the tangent functions may not approach an odd multiple of $\pi/2$.

[5] $\gamma = 16^{63}(1 - 16^{-6})$ for regular precision routines, $16^{63}(1 - 16^{-14})$ for double-precision routines and $16^{63}(1 - 16^{-28})$ for extended precision.

Figure 3. *(Part 2 of 2)* Trigonometric Subprograms.

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[2]	Error Code
			No.	Type[1]	Range		
Hyperbolic sine and cosine	SINH	$y = \frac{e^x - e^{-x}}{2}$	1	REAL *4	$\|x\| < 175.366$	REAL *4 $-\gamma \leqq y \leqq \gamma$	256
	COSH	$y = \frac{e^x + e^{-x}}{2}$	1	REAL *4	$\|x\| < 175.366$	REAL *4 $1 \leqq y \leqq \gamma$	256
	DSINH	$y = \frac{e^x - e^{-x}}{2}$	1	REAL *8	$\|x\| < 175.366$	REAL *8 $-\gamma \leqq y \leqq \gamma$	266
	DCOSH	$y = \frac{e^x + e^{-x}}{2}$	1	REAL *8	$\|x\| < 175.366$	REAL *8 $1 \leqq y \leqq \gamma$	266
	QSINH	$y = \frac{e^x - e^{-x}}{2}$	1	REAL *16	$\|x\| \leqq 175.366$	REAL *16 $-\gamma \leqq y \leqq \gamma$	296
	QCOSH	$y = \frac{e^x + e^{-x}}{2}$	1	REAL *16	$\|x\| \leqq 175.366$	REAL *16 $1 \leqq y \leqq \gamma$	296
Hyperbolic tangent	TANH	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	1	REAL *4	any REAL argument	REAL *4 $-1 \leqq y \leqq 1$	—
	DTANH	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	1	REAL *8	any REAL argument	REAL *8 $-1 \leqq y \leqq 1$	—
	QTANH	$y = \frac{e^x - e^{-x}}{ex + e^{-x}}$	1	REAL *16	any REAL argument	REAL *16 $-1 \leqq y \leqq 1$	—

NOTES:

[1] REAL *4, REAL *8, and REAL *16 arguments correspond to REAL, DOUBLE and EXTENDED PRECISION arguments, respectively, in VS FORTRAN.

[2] $\gamma = 16^{63}(1 - 16^{-6})$ for regular precision routines, $16^{63}(1 - 16^{-16})$ for double-precision routines, and $16^{63}(1 - 16^{-28})$ for extended precision.

Figure 4. Hyperbolic Function Subprograms

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[2]	Error Code
			No.	Type[1]	Range		
Absolute value	CABS	$y=\|z\|=(x_1^2+x_2^2)^{1/2}$	1	COMPLEX *8	any COMPLEX argument See Note 2	REAL *4 $0 \leqq y_1 \leqq \gamma$ $y_2 = 0$	—
	CDABS	$y=\|z\|=(x_1^2+x_2^2)^{1/2}$	1	COMPLEX *16	any COMPLEX argument See Note 2	REAL *8 $0 \leqq y_1 \leqq \gamma$ $y_2 = 0$	—
	CQABS	$y=\|z\|=(x_1^2+x_2^2)^{1/2}$	1	COMPLEX *32	any COMPLEX argument See Note 1	REAL *16 $0 \leqq y_1 \leqq \gamma$ $y_2 = 0$	—
	IABS	$y=\|x\|$	1	INTEGER *4	any INTEGER argument	INTEGER *4 $0 \leqq y \leqq \gamma$	—
	ABS	$y=\|x\|$	1	REAL *4	any REAL argument	REAL *4 $0 \leqq y \leqq \gamma$	—
	DABS	$y=\|x\|$	1	REAL *8	any REAL argument	REAL *8 $0 \leqq y_1 \leqq \gamma$	—
	QABS	$y=\|x\|$	1	REAL *16	any REAL argument	REAL *16 $0 \leqq y_1 \leqq \gamma$	—
Error function	ERF	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$	1	REAL *4	any REAL argument	REAL *4 $-1 \leqq y \leqq 1$	—
	ERFC	$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}du$ $y=1-\text{erf}(x)$	1	REAL *4	any REAL argument	REAL *4 $0 \leqq y \leqq 2$	—
	DERF	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$	1	REAL *8	any REAL argument	REAL *8 $-1 \leqq y \leqq 1$	—
	DERFC	$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}du$ $y=1-\text{erf}(x)$	1	REAL *8	any REAL argument	REAL *8 $0 \leqq y \leqq 2$	—
	QERF	$y=\frac{2}{\pi}\int_0^x e^{-u^2}du$	1	REAL *16	any REAL argument	REAL *16 $-1 \leqq y \leqq 1$	—
	QERFC	$y=\frac{2}{\pi}\int_x^x e^{-u^2}du$ $y=1-\text{erf}(x)$	1	REAL *16	any REAL argument	REAL *16 $0 \leqq y \leqq 2$	—

NOTES: (See end of figure.)

Figure 5. *(Part 1 of 3)* Miscellaneous Mathematical Subprograms

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[2]	Error Code
			No.	Type[1]	Range		
Gamma and log-gamma	GAMMA	$y=\int_0^\infty u^{x-1}e^{-u}\,du$	1	REAL *4	$x > 2^{-252}$ and $x < 57.5744$	REAL *4 $0.88560 \leqq y \leqq \gamma$	290
	ALGAMA	$y=\log_e \Gamma(x)$ or $y=\log_e \int_0^\infty u^{x-1}e^{-u}\,du$	1	REAL *4	$x > 0$ and $x < 4.2913 \cdot 10^{73}$	REAL *4 $-0.12149 \leqq y \leqq \gamma$	291
	DGAMMA	$y=\int_0^\infty u^{x-1}e^{-u}\,du$	1	REAL *8	$x > 2^{-252}$ and $x < 57.5744$	REAL *8 $0.88560 \leqq y \leqq \gamma$	300
	DLGAMA	$y=\log_e \Gamma(x)$ or $y=\log_e \int_0^\infty u^{x-1}e^{-u}\,du$	1	REAL *8	$x > 0$ and $x < 4.2913 \cdot 10^{73}$	REAL *8 $-0.12149 \leqq y \leqq \gamma$	301
	LGAMMA	$y=\log_e \Gamma(x)$ or $y=\log_e \int_0^\infty u^{x-1}e^{-u}\,du$	1	REAL *4	$x > 2^{-252}$ and $x < 57.5744$	REAL *4 $0.88560 \leqq y \leqq \gamma$	290
Maximum and minimum values	MAX0	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	INTEGER *4	—
	MIN0	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	INTEGER *4	—
	AMAX0	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	REAL *4	—
	AMIN0	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	INTEGER *4	any INTEGER arguments	REAL *4	—
	MAX1	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	INTEGER *4	—
	MIN1	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	INTEGER *4	—
	AMAX1	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	REAL *4	—
	AMIN1	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	REAL *4	any REAL arguments	REAL *4	—
	DMAX1	$y=\max(x_1,\ldots,x_n)$	$\geqq 2$	REAL *8	any REAL arguments	REAL *8	—
	DMIN1	$y=\min(x_1,\ldots,x_n)$	$\geqq 2$	REAL *8	any REAL arguments	REAL *8	—
NOTES: (See end of figure.)							

Figure 5. *(Part 2 of 3)* Miscellaneous Mathematical Subprograms

General Function	Entry Name	Definition	Argument(s)			Function Value Type[1] and Range[5]	Error Code
			No.	Type[1]	Range		
Maximum and minimum values (continued)	MAX	$y=\max(x_1,\ldots,x_n)$	≥ 2	INTEGER *4	any INTEGER arguments	INTEGER *4	--
	MIN	$y=\min(x_1,\ldots,x_n)$	≥ 2	INTEGER *4	any INTEGER arguments	INTEGER *4	--
	QMAX1	$y=\max(x_1,\ldots,x_n)$	≥ 2	REAL *16	any REAL argument	REAL *16	--
	QMIN1	$y=\min(x_1,\ldots,x_n)$	≥ 2	REAL *16	any REAL argument	REAL *16	--
Modular arithmetic	MOD	$y=x_1$ (modulo x_2) See Note 3	2	INTEGER	$x_2 \neq 0$ See Note 4	INTEGER *4	--
	AMOD	$y=x_1$ (modulo x_2) See Note 3	2	REAL *4	$x_2 \neq 0$ See Note 4	REAL *4	--
	DMOD	$y=x_1$ (modulo x_2) See Note 3	2	REAL *8	$x_2 \neq 0$ See Note 4	REAL *8	--
	QMOD	$y=x_1$ (modulo x) See Note 3	2	REAL *16	$x_2 \neq 0$ See Note 4	REAL *16	--
Truncation	AINT	y = (sign of x)•n where $n = \lfloor \lvert x \rvert \rfloor$ See Note 6	1	REAL *4	any REAL argument	REAL *4	--
	INT	y = (sign of x)•n where $n = \lfloor \lvert x \rvert \rfloor$ See Note 6	1	REAL *4	any REAL argument	INTEGER *4	--
	IDINT	y = (sign of x)•n where $n = \lfloor \lvert x \rvert \rfloor$ See Note 6	1	REAL *8	any REAL argument	INTEGER *4	--

NOTES:

[1] REAL *4, REAL *8, and REAL *16 arguments correspond to REAL, DOUBLE and EXTENDED PRECISION arguments, respectively, in VS FORTRAN.

[2] Floating-point overflow can occur.

[3] The expression x_1 (modulo x_2) is defined as $x_1 - \left[\frac{x_1}{x_2}\right] \bullet x_2$, where the brackets indicate that an integer is used. The largest integer whose magnitude does not exceed the magnitude of $\frac{x_1}{x_2}$ is used. The sign of the integer is the same as the sign of $\frac{x_1}{x_2}$.

[4] If $x_2 = 0$, then the modulus function is mathematically undefined. In addition, a divide exception is recognized and an interruption occurs. (A detailed description of the interruption procedure is given in Appendix C.)

[5] $\gamma = 16^{63}(1 - 16^{-6})$ for regular precision routines, $16^{63}(1 - 16^{-14})$ for double-precision routines and $16^{63}(1 - 16^{-28})$ for extended precision routines.

For example, $n = \lfloor m \rfloor$ where m is the greatest integer satisfying the relationship $\lvert m \rvert \leq \lvert x \rvert$.

Figure 5. *(Part 3 of 3)* Miscellaneous Mathematical Subprograms

General Function	Entry Name	Definition	Argument(s)			Function Value Type and Range	Error Code
			No.	Type	Function		
Convert character to integer	ICHAR	Conversion to integer	1	Character	Integer	INTEGER *4	–
Convert integer to character	CHAR	Conversion to character	1	Integer	Character	INTEGER *4	188
Length of character item	LEN	Length of character entity	1	Character	Integer	INTEGER *4	–
Index of characte item	INDEX	Location of substring a_2 in string a_1	2	Character	Integer	INTEGER *4	189, 190
Alphamerically greater than or equal	LGE	$a_1 \geq a_2$	2	Character	Logical	LOGICAL *4	191, 192
Alphamerically greater than	LGT	$a_1 > a_2$	2	Character	Logical	LOGICAL *4	191, 192
Alphamerically less than or equal	LLE	$a_1 \leq a_2$	2	Character	Logical	LOGICAL *4	191, 192
Alphamerically less than	LLT	$a_1 < a_2$	2	Character	Logical	LOGICAL *4	191, 192

Figure 6. Character Manipulation Routines

Implicitly Called Subprograms

The implicitly called subprograms are executed as a result of certain notations appearing in a VS FORTRAN source statement. When a number is raised to a power or when a multiplication or division of complex numbers is to be performed, the VS FORTRAN compiler generates the instructions necessary to call the appropriate subprogram. For example, if the following source statement appears in a source module,

```
ANS = BASE**EXPON
```

where BASE and EXPON are REAL*4 variables, the VS FORTRAN compiler generates a reference to FRXPR#, the entry name for a subprogram that raises a real number to a real power.

The implicitly called subprograms in the VS FORTRAN library are described in Figure 6. The column headed "Implicit Function Reference" gives a representation of a source statement that might appear in a VS FORTRAN source module and cause the subprogram to be called. The rest of the column headings in Figure 6 have the same meaning as those used with the explicitly called subprograms. Algorithms for implicitly called subprograms are given in the chapter "Algorithms." Additional information for assembler language programmers is given in Appendix A.

For subprograms that involve exponentiation, the action taken within a subprogram depends upon the types of the base and exponent used. Figures 7 through 11 show the result of an exponentiation performed with the different combinations and values of base and exponent. In these figures, I and J are integers; A and B are real numbers; C is a complex number.

General Function	Entry[1] Name	Implicit Function Reference[2]	Argument(s) No.	Argument(s) Type[3]	Function Value Type[3]	Error Code
Multiply and divide complex numbers	CDMPY#	$y = z_1 * z_2$	2	COMPLEX *16	COMPLEX *16	—
	CDDVD#	$y = z_1/z_2$	2	COMPLEX *16	COMPLEX *16	—
	CMPY#	$y = z_1 * z_2$	2	COMPLEX *8	COMPLEX *8	—
	CDVD#	$y = z_1/z_2$	2	COMPLEX *8	COMPLEX *8	—
	CQMPY#	$y = z_1 * z_2$	2	COMPLEX *32	COMPLEX *32	—
	CQDVD#	$y = z_1/z_2$	2	COMPLEX *32	COMPLEX *32	—
Compare of Complex numbers	CXMPR # (See Note 4)	$y = z_1$ compop z_2 (See Note 5)	2	COMPLEX (of all lengths)	LOGICAL*4	—
Raise an integer to an integer power	FIXPI#	y = i**j	2	i = INTEGER *4 j = INTEGER *4	INTEGER *4	241
Raise a real number to an integer power	FRXPI#	y = a**j	2	a = REAL *4 j = INTEGER *4	REAL *4	242
	FDXPI#	y = a**j	2	a = REAL *8 j = INTEGER *4	REAL *8	243
	FQXPI#	y = a**j	2	a = REAL *16 j = INTEGER *4	REAL*16	248
Raise a real number to a real power	FRXPR#	y = a**b	2	a = REAL *4 b = REAL *4	REAL *4	244
	FDXPD#	y = a**b	2	a = REAL *8 b = REAL 8*	REAL *8	245
	FQXPQ#	y = a**b	2	a = REAL *16 b = REAL *16	REAL *16	249, 250
Raise 2 to a real power	FQXP2#	y = 2**b	1	b = REAL *16	REAL *16	260
Raise a complex number to an integer power	FCDXI#	y = z**j	2	z = COMPLEX *16 j = INTEGER *4	COMPLEX *16	247
	FCXPI#	y = z**j	2	z = COMPLEX *8 j = INTEGER *4	COMPLEX *8	246
	FCQXI#	y = z**j	2	z = COMPLEX *32 j = INTEGER *4	COMPLEX *32	270

Figure 7. Implicitly Called Mathematical Subprograms (Part 1 of 2)

NOTES:
1. This name must be used in an assembler language program to call the subprogram; the character # is a part of the name and must be included.
2. This is only a *representation* of a FORTRAN statement; it is not the only way the subprogram may be called.
3. REAL *4, REAL *8, and REAL *16 arguments correspond to REAL, DOUBLE PRECISION, and EXTENDED PRECISION arguments, respectively, in VS FORTRAN.
4. CXMPR# is an entry name in the library module IFYCCMPR, which is also used for a compare of character arguments.
5. compop is one of the following relational operators: equal or not equal.

Figure 7. Implicitly Called Mathematical Subprograms (Part 2 of 2)

Entry Name	Implicit Function Reference	Argument(s)		Function Value Type	Error Code
		No.	Type		
CCMPR#	$y = x_1$ compop x_2 (See Note)	6	Character	Any character argument	193 194
CMOVE#	$y = x$	4	Character	Any character argument	195 196 197
CNCAT#	$y = x_1 // x_2 \ldots // x_n$	≥ 2	Character	Any character argument	198 199

NOTE: Where compop is one of the following relational operators:
equal
not equal
greater than
less than
greater than or equal
less than or equal

Figure 8. Implicitly Called Character Subprograms

Base (I)	Exponent (J)		
	J > 0	J = 0	J < 0
I > 1	Compute the function value	Function value = 1	Function value = 0
I = 1	Compute the function value	Function value = 1	Function value = 1
I = 0	Function value = 0	Error message 241	Error message 241
I = −1	Compute the function value	Function value = 1	If J is an odd number, function value = −1. If J is an even number, function value = 1.
I < −1	Compute the function value	Function value = 1	Function value = 0

Figure 9. Exponentiation with Integer Base and Exponent

Base (A)	Exponent (J)		
	J > 0	J = 0	J < 0
A > 0	Compute the function value	Function value = 1	Compute the function value
A = 0	Function value = 0	Error message 242 or 243	Error message 242 or 243
A < 0	Compute the function value	Function value = 1	Compute the function value

Figure 10. Exponentiation with Real Base and Integer Exponent

Base (A)	Exponent (B)		
	B > 0	B = 0	B < 0
A > 0	Compute the function value	Function value = 1	Compute the function value
A = 0	Function value = 0	Error message 244 or 245	Error message 244 or 245
A < 0	Error message 253 or 263	Function value = 1	Error message 253 or 263

Figure 10.1. Exponentiation with Real Base and Exponent

Base (C) C = P + Qi	Exponent (J)		
	J > 0	J = 0	J < 0
P > 0 and Q > 0	Compute the function value	Function value=1 + 0*i*	Compute the function value
P > 0 and Q = 0	Compute the function value	Function value=1 + 0*i*	Compute the function value
P > 0 and Q < 0	Compute the function value	Function value=1 + 0*i*	Compute the function value
P = 0 and Q > 0	Compute the function value	Function value=1 + 0*i*	Compute the function value
P = 0 and Q = 0	Function value 0 + 0*i*	Error message 246 or 247	Error message 246 or 247
P = 0 and Q < 0	Compute the function value	Function value=1 + 0*i*	Compute the function value
P < 0 and Q > 0	Compute the function value	Function value=1 + 0*i*	Compute the function value
P < 0 and Q = 0	Compute the function value	Function value=1 + 0*i*	Compute the function value
P < 0 and Q < 0	Compute the function value	Function value=1 + 0*i*	Compute the function value

Figure 10.2. Exponentiation with Complex Base and Integer Exponent

Service Subroutine Subprograms

The service subprograms supplied in the VS FORTRAN library are divided into two groups: One group tests for mathematical exceptions and the other groups perform utility functions. Service subprograms are called by the appropriate entry name in a VS FORTRAN language CALL statement.

Mathematical Exception Test Subprograms

These subprograms test the status of indicators and may return a value to the calling program. In the following descriptions of the subprograms, *j* represents an integer value.

Overflow Indicator Subprogram
Entry Name: OVERFL

This subprogram tests for an exponent overflow or underflow exception and returns a value that indicates the existing condition. After testing, the overflow indicator is turned off. This subprogram is called by using the entry name OVERFL in a CALL statement. The source language statement is:

```
CALL OVERFL (j)
```

The value of *j* is returned by the subprogram to indicate the following:

1 = floating-point overflow condition has occurred last.
2 = no overflow or underflow condition has occurred.
3 = a floating-point underflow condition has occurred last.

Note: A value for *j* of 1 or 3 indicates that condition was the last one to occur. An overflow followed by an underflow in the same statement would be recorded as condition 3—"underflow occurred last."

Divide Check Subprogram
Entry Name: DVCHK

This subprogram tests for a divide-check exception and returns a value that indicates the existing condition. After testing, the divide-check indicator is turned off. This subprogram is called by using entry name DVCHK in a CALL statement. The source language statement is:

```
CALL DVCHK (j)
```

where:
j is set to one if the divide-check indicator was on; or to 2 if the indicator was off.

Utility Subprograms

The utility subprograms perform two operations for the FORTRAN programmer: they either terminate execution (EXIT) or dump a specified area of storage (DUMP/PDUMP).

End Execution Subprogram
Entry Name: EXIT

The end execution subprogram terminates execution of the load module and returns control to the operating system. (Except that no operator message is produced, EXIT performs a function similar to that performed by the STOP statement.) This subprogram is called by using the entry name EXIT in a CALL statement. The source language statement is:

```
CALL EXIT
```

Storage Dump Subprogram
Entry Names: DUMP/PDUMP, CDUMP/CPDUMP

These subprograms dump character data.

Entry names DUMP/PDUMP dump a specified area of storage. Either of two entry names (DUMP or PDUMP) can be used to call the subprogram. The entry name is followed by the limits of the area to be dumped and the format specification. The entry name used in the CALL statement depends upon the nature of the dump to be taken.

If execution of the load module is to be terminated after the dump is taken, entry name DUMP is used. The source language statement is:
where:

CALL DUMP ($a_1, b_1, f_1, \ldots, a_n, b_n, f_n$)

a and *b* are variables that indicate the limits of storage to be dumped (either *a* or *b* may represent the upper or lower limits of storage).

f indicates the dump format and may be one of the integers given in Figure 11. A sample printout for each format is given in Appendix E.

VS FORTRAN
0 specifies hexadecimal
1 specifies LOGICAL *1
2 specifies LOGICAL *4
3 specifies INTEGER *2
4 specifies INTEGER *4
5 specifies REAL *4
6 specifies REAL *8
7 specifies COMPLEX *8
8 specifies COMPLEX *16
9 specifies literal
10 specifies REAL *16
11 specifies COMPLEX *32

Figure 11. DUMP/PDUMP Format Specifications

If execution is to be resumed after the dump is taken, entry name PDUMP is used. The source language statement is:

CALL PDUMP ($a_1, b_1, f_1, \ldots, a_n, b_n, f_n$)

where *a*, *b*, and *f* have the same meaning as for DUMP.

Programming Considerations

A load module or phase may occupy a different area of storage each time it is executed. To ensure that the appropriate areas of storage are dumped, the following conventions should be observed.

If an array and a variable are to be dumped at the same time, a separate set of arguments should be used for the array and for the variable. The specification of limits for the array should be from the first element in the array to the last element. For example, assume that A is a variable in COMMON, B is a REAL number, and TABLE is an array of 20 elements. The following call to the storate dump subprogram could be used to dump TABLE and B in the hexadecimal format and terminate execution after the dump is taken:

```
CALL DUMP (TABLE(1), TABLE(20),0,B,B,0)
```

If an area of storage in COMMON is to be dumped at the same time as an area of storage not in COMMON, the arguments for the area in COMMON should be given separately. For example, the following call to the storage dump subprogram could be used to dump the variables A and B in REAL*8 format without terminating execution:

```
CALL PDUMP (A,A,6,B,B,6)
```

If variables not in COMMON are to be dumped, each variable must be listed separately in the argument list. For example, if R, P, and Q are defined implicitly in the program, the statement

```
CALL PDUMP (R,R,5,P,P,5,Q,Q,5)
```

should be used to dump the three variables in REAL *4 format. If the statement

```
CALL PDUMP (R,Q,5)
```

is used, all main storage between R and Q is dumped, which may or may not include P, and may include other variables.

If an array and a variable are passed to a subroutine as arguments, the arguments in the call to the storage dump subprogram in the subroutine should specify the parameters used in the definition of the subroutine. For example, if the subroutine SUBI is defined as:

```
SUBROUTINE SUBI (C,Y)
DIMENSION X (10)
```

and to call to SUBI within the source module is:

```
DIMENSION A (10)
      .
      .
      .
CALL SUBI (A, B)
```

then the following statement should be used in SUBI to dump the variables in hexadecimal format without terminating execution:

```
CALL PDUMP (X(1), X(10), 0, Y, Y, 0)
```

If the statement

```
CALL PDUMP (X(1), Y, 0)
```

is used, all storage between (1) and Y is dumped because of the method of transmitting arguments.

When hexadecimal (0) or literal (9) is specified, the programmer should realize that the upper limit is assumed to be of length 4.

Storage Dump Subprogram
Entry Names: CDUMP/CPDUMP

This subprogram dumps a specified area of storage, which contains character data only. Either of two entry names (CDUMP or CPDUMP) can be used to call the subprogram. The entry name is followed by the limits of the area to be dumped. The entry name used in the CALL statement depends upon the nature of the dump to be taken.

If execution of the load module is to be terminated after the dump is taken, entry name CDUMP is used. The source language statement is:

CALL CDUMP ($a_1, b_1, \ldots, a_n, b_n$)

where:
a and *b* are variables that indicate the limits of storage to be dumped.

A sample printout for each format is given in Appendix E.

If execution is to be resumed after the dump is taken, entry name CPDUMP is used. The source language statement is:

CALL CPDUMP ($a_1, b_1, \ldots, a_n, b_n$)

where *a*, and *b*, have the same meaning as for CDUMP.

If c_1 is a character variable of length 8 and c_2 is a character array of dimension 10 and length 15 then the statement:

CALL CPDUMP (c_1, c_2 *(1)*, c_2 *(10)*)

will dump the variable c_1 and all the array elements of array c_2 in character format and continue execution.

Programming Considerations

A load module may occupy a different area of storage each time it is executed. To ensure that the appropriate areas of storage are dumped, the following conventions should be observed.

If an array and a variable are to be dumped at the same time, a separate set of arguments should be used for the array and for the variable. The specification of limits for the array should be from the first element in the array to the last element. For example, assume that A is a variable in COMMON, B is a REAL number, and TABLE is an array of 20 elements. The following call to the storate dump subprogram could be used to dump TABLE and B in the hexadecimal format and terminate execution after the dump is taken:

CALL CDUMP ($a_1, b_1, \ldots, a_n, b_n$)

If an area of storage in COMMON is to be dumped at the same time as an area of storage not in COMMON, the arguments for the area in COMMON should be given separately. For example, the following call to the storage dump subprogram could be used to dump the variables A and B in REAL*8 format without terminating execution:

CALL PDUMP (A,A,6,B,B,6)

Algorithms

This chapter contains information about the method by which each mathematical function is computed. The information for explicitly called subprograms is arranged alphabetically according to the specific function of each subprogram (i.e., absolute value, exponentiation, logarithmic, etc.). The individual entry names associated with each subprogram are arranged logically from simple to complex within each function. For example, the heading "Square Root Subprograms" will have algorithms arranged in the following order by entry name: SQRT, DSQRT, CSQRT, CDSQRT.

Information for the implicity called subprograms is arranged alphabetically according to function, and alphabetically by entry name within that function. For example, the heading "Complex Multiply and Divide Subprograms" will have algorithms arranged in the following order: CDDVD#/CDMPY#, CDVD#/CMPY#.

The information for each subprogram is divided into two parts. The first part describes the algorithm used; the second part describes the effect of an argument error upon the accuracy of the answer returned.

The presentation of each algorithm is divided into its major computational steps; the formulas necessary for each step are supplied. For the sake of brevity, the needed constants are normally given only symbolically. (The actual values can be found in the assembly listing of the subprograms.) Some of the formulas are widely known; those that are not so widely known are derived from more common formulas. The process leading from the common formula to the computational formula is sketched in enough detail so that the derivation can be reconstructed by anyone who has an understanding of college mathematics and access to the common texts on numerical analysis. Many approximations were derived by the so-called "minimax" methods. The approximation sought by these methods can be characterized as follows. Given a function $f(x)$, an interval I, the form of the approximation (such as the rational form with specified degrees), and the type of error to be minimized (such as the relative error), there is normally a unique approximation to $f(x)$ whose maximum error over I is the smallest among all possible approximations of the given form. Details of the theory and the various methods of deriving such approximation are provided in standard reference. The accuracy figures cited in the algorithm sections are theoretical, and they do not take round-off errors into account. Minor programming techniques used to minimize round-off errors are not necessarily described here.

The accuracy of an answer provided by these algorithms is influenced by two factors: the performance of the subprogram (see the chapter, "Accuracy Statistics") and the accuracy of the argument. The effect of an argument error upon the accuracy of an answer depends solely upon the mathematical function involved and not upon the particular coding used in the subprogram.

A guide to the propagational effect of argument errors is provided because argument errors always influence the accuracy of answers whether the errors are accumulated prior to use of the subprogram or introduced by newly converted data. This guide (expressed as a simple formula where possible) is intended to assist users in assessing the effect of an argument error.

The following symbols are used in this chapter to describe the effect of an argument error upon the accuracy of the answer:

SYMBOL	EXPLANATION
$g(x)$	The result given by the subprogram.
$f(x)$	The correct result.
ϵ	$\left\|\frac{f(x) - g(x)}{f(x)}\right\|$ The relative error of the result given by the subprogram.
δ	The relative error of the argument.
E	$\|f(x) - g(x)\|$ The absolute error of the result given by the subprogram.
Δ	The absolute error of the argument.

The notation used for the continued fractions complies with the specifications set by the National Bureau of Standards.[1]

Although it is not specifically stated below for each subroutine, the algorithms is the chapter were programmed to conform to the following standards governing floating-point overflow/underflow.

- Intermediate underflow and overflows are not permitted to occur. This prevents the printing of irrelevant messages.
- Those arguments for which the answer can overflow are excluded from the permitted range of the subroutine. This rule does not apply to CDABS and CABS.
- When the magnitude of the answer is less than 16^{-65}, zero is given as the answer. If the floating-point underflow exception mask is on at the time, the underflow message will be printed.

Control of Program Exceptions in Mathematical Functions

The VS FORTRAN mathematical functions have been coded with careful control of error situations. A result is provided whenever the answer is within the range representable in the floating-point form. In order to be consistent with VS FORTRAN control of exponent overflow/underflow exceptions, the following types of conditions are recognized and handled separately.

When the magnitude of the function value is too large to be represented in the floating-point form, the condition is called a terminal overflow; when the magnitude is too small to be represented, a terminal underflow. On the other hand, if the function value is representable, but if execution of the chosen algorithm causes an overflow or underflow in the process, this condition is called an intermediate overflow or underflow.

Function subroutines in the VS FORTRAN library have been coded to observe the following rules for these conditions:

1. Algorithms which can cause an intermediate overflow have been avoided. Therefore an intermediate overflow should occur only rarely during the execution of a function subroutine of the library.
2. Intermediate underflows are generally detected and not allowed to cause an interrupt. In other words, spurious underflow signals are not allowed to be given. Computation of the function value is successfully carried out.
3. Terminal overflow conditions are screened out by the subroutine. The argument is considered out of range for computation and an error diagnostic is given.

[1] For more information, see Milton Abramowitz and Irene A. Stegun (editors), *Handbook of Mathematical Functions*, Applied Mathematics Series-55 (National Bureau of Standards, Washington, D.C., 1965).

4. Terminal underflow conditions are handled by forcing a floating-point underflow exception. This provides for the detection of underflow in the same manner as for an arithmetic statement. Terminal underflows can occur in the following function subroutines: EXP, DEXP, ATAN2, DATAN2, ERFC, and DERFC.

For implicit arithmetic subroutines, these rules do not apply. In this case, both terminal overflows and terminal underflows will cause respective floating-point exceptions. In addition, in the case of complex arithmetic (implicit multiply and divide), premature overflow/underflow is possible when the result of arithmetic is very close to an overflow or underflow condition.

Explicitly Called Subprograms

Absolute Value Subprograms

ABS/IABS

Algorithm

If $x < 0$, $|x| = -x$. Otherwise $|x| = x$.

CABS/CDABS

Algorithm

1. Write $|x + iy| = a + ib$.
2. Let $v_1 = \max(|x|, |y|)$, and $v_2 = \min(|x|, |y|)$.
3. If characteristics of v_1 and v_2 differ by 7 (15 for CDABS) or more, or if $v_2 = 0$, then $a = v_1$, $b = 0$.
4. Otherwise,

$$a = 2 \cdot v_1 \cdot \sqrt{1/4 + 1/4\left(\frac{v_2}{v_1}\right)^2}, \text{ and } b = 0.$$

If the answer is greater than 16^{63}, the floating-point overflow interruption will take place (see Appendix C). The algorithms for both complex absolute value subprograms are identical. Each subprogram uses the appropriate real square root subprogram (SQRT or DSQRT).

CQABS

Algorithm

1. Write $|x + iy| = a + ib$.
2. Let $v_1 = \max(|x|, |y|)$, and $v_2 = \min(|x|, |y|)$.
   Let $16^{p-1} \leq v_1 < 16^p$.
3. If characteristics of v_1 and v_2 differ by 15 or more, or if $v_2 = 0$, then $a = v_1$, $b = 0$.
4. Otherwise, let $w_1 = 16^{1-p} \cdot v_1$, and $w_2 = 16^{1-p} \cdot v_2$.
5. Compute $w = \sqrt{w_1^2 + w_2^2}$. Then $a = 16^{p-1} w$ and $b = 0$.
6. The scaling factor 16^{p-1} is easy to construct. Scaling is carried out by short precision divisions, and the restoration is carried out by extended precision multiplication.

Effect of an Argument Error

$\epsilon \sim \frac{x^2}{a^2}\delta(x) + \frac{y^2}{a^2}\delta(y)$ where $\delta(x)$ and $\delta(y)$ are relative erros inherent in the real part and the imaginary part of the argument, respectively.

Arcsine and Arccosine Subprograms

ARSIN/ARCOS

Algorithm

1. If $0 \leqq x \leqq \frac{1}{2}$, then compute arcsin (x) by a continued fraction of the form:

   $$\arcsin(x) \cong x + x^3 \cdot F \text{ where}$$

   $$F = \frac{d_1}{(x^2 + c_1) + \dfrac{d_2}{(x^2 + c_2)}}.$$

   The coefficients of this formula were derived by transforming the minimax rational approximation (in relative error, over the range $0 \leqq x^2 \leqq \frac{1}{4}$) for arcsin $(x)/x$ of the following form:

   $$\frac{\arcsin(x)}{x} \cong a_0 + x^2 \cdot \left[\frac{a_1 + a_2 x^2}{b_0 + b_1 x^2 + x^4}\right].$$

   Minimax was taken under the constraint that $a_0 = 1$ exactly. The relative error of this approximation is less than $2^{-28.3}$.
   If $0 \leqq x \leqq \frac{1}{2}$, $\arccos(x)$ is computed as:

   $$\arccos(x) = \frac{\pi}{2} - \arcsin(x).$$

2. If $\frac{1}{2} < x \leqq 1$, then compute $\arccos(x)$ essentially as:

   $$\arccos(x) = 2 \cdot \arcsin\left(\sqrt{\frac{1-x}{2}}\right).$$

   This case is now reduced to the first case because within these limits,

   $$0 \leqq \sqrt{\frac{1-x}{2}} \leqq \frac{1}{2}.$$

   This computation uses the real square root subprogram (SQRT).
   If $\frac{1}{2} < x \leqq 1$, $\arcsin(x)$ is computed as:

   $$\arcsin(x) = \frac{\pi}{2} - \arccos(x).$$

   Implementation of the above algorithms (steps 1 and 2) was carried out with care to minimize the round off errors.

3. If $-1 \leqq x < 0$, then $\arcsin(x) = -\arcsin|x|$
   and $\arccos(x) = \pi - \arccos|x|$.

   This reduces these cases to one of the two positive cases.

Effect of an Argument Error

$E \sim \frac{\Delta}{\sqrt{1-x^2}}$. For small values of x, $E \sim \Delta$. Toward the limits (± 1) of the range, a small Δ causes a substantial error in the answer. For the arcsine, $\epsilon \sim \delta$ if the value of x is small.

ASIN/ACOS

Algorithm

If x is R*4, then ASIN(x) = ARSIN(x) and ACOS(x) = ARCOS(x)
If x is R*8, then DASIN(x) = DARSIN(x) and DACOS(x) = DARCOS(x)
If x is R*16, then QASIN(x) = QARSIN(x) and QACOS(x) = QARCOS(x)

DARSIN/DARCOS

Algorithm

1. If $0 \leqq x \leqq \frac{1}{2}$, then compute $\arcsin(x)$ by a continued fraction of the form:

$$\arcsin(x) \cong x + x^3 \cdot F \text{ where}$$

$$F = c_1 + \frac{d_1}{(x^2 + c_2) + \dfrac{d_2}{(x^2 + c_3) + \dfrac{d_3}{(x^2 + c_4) + \dfrac{d_4}{(x^2 + c_5)}}}}.$$

The relative error of this approximation is less than $2^{-57.2}$.
The coefficients of this formula were derived by transforming the minimax rational approximation (in relative error, over the range $0 \leqq x^2 \leqq \frac{1}{4}$) for $\arcsin(x)/x$ of the following form:

$$\frac{\arcsin(x)}{x} \cong a_0 + x^2\left[\frac{a_1 + a_2x^2 + a_3x^4 + a_4x^6 + a_5x^8}{b_0 + b_1x^2 + b_2x^4 + b_3x^6 + x^8}\right].$$

Minimax was taken under the constraint that $a_0 = 1$ exactly.
If $0 \leqq x \leqq \frac{1}{2}$, $\arccos(x)$ is computed as:

$$\arccos(x) = \frac{\pi}{2} - \arcsin(x).$$

2. If $\frac{1}{2} < x \leqq 1$, then compute $\arccos(x)$ essentially as:

$$\arccos(x) = 2 \cdot \arcsin\left(\sqrt{\frac{1 - x}{2}}\right).$$

This case is now reduced to the first case because within these limits,

$$0 \leqq \sqrt{\frac{1 - x}{2}} \leqq \frac{1}{2}.$$

This computation uses the real square root subprogram (DSQRT).
If $\frac{1}{2} < x \leqq 1$, $\arcsin(x)$ is computed as:

$$\arcsin(x) = \frac{\pi}{2} - \arccos(x).$$

Implementation of the above algorithms (steps 1 and 2) was carried out with care to minimize the round-off errors.

3. If $-1 \leqq x < 0$, then $\arcsin(x) = -\arcsin|x|$, and $\arccos(x) = \pi - \arccos|x|$. This reduces these cases to one of the two positive cases.

Effect of an Argument Error

$E \sim \dfrac{\Delta}{\sqrt{1 - x^2}}$. For small values of x, $E \sim \Delta$. Toward the limits ($\pm$ 1) of the range a small Δ causes a substantial error in the answer. For the arcsine, $\epsilon \sim \delta$ if the value of x is small.

DASIN/DACOS

These names are aliases for DARSIN and DARCOS.

QARSIN/QARCOS

Algorithm

1. If $0 \leqq x \leqq \frac{1}{2}$, then compute arcsin (x) by a minimax rational approximation of the following form:

   $w = 2x^2$, and

   $$\arcsin(x) \cong x + x \cdot w\left[\frac{a_0 + w[a_1 + a_2 w + \ldots + a_9 w^8]}{b_0 + b_1 w + \ldots + b_8 w^8 + w^9}\right]$$

   Coefficients $\{a_i, b_i\}$ were determined by a minimax technique and the relative error of this approximation is less than 16^{-28}. The order of evaluating this rational form was so chosen as to reduce round-off errors.

   If $0 \leqq x \leqq \frac{1}{2}$, arccos (x) is computed as:

   $$\arccos(x) = \frac{\pi}{2} - \arcsin(x).$$

2. If $\frac{1}{2} < x \leqq 1$, then compute arccos (x) essentially as:

   $$\arccos(x) = 2 \cdot \arcsin\left(\sqrt{\frac{1-x}{2}}\right)$$

   Or more specifically, $w = 1 - x$, $z = \sqrt{2(1-x)}$, and
   arccos $(x) \cong z + z\ w\ [a_0 +$ the above rational form].

This case is now reduced to the first case because, within these limits,

$$0 \leqq \sqrt{\frac{1-x}{2}} \leqq \frac{1}{2}.$$

This computation uses the square root subroutine (QSQRT).

If $\frac{1}{2} < x \leqq 1$, arcsin (x) is computed as:

$$\arcsin(x) = \frac{\pi}{2} - \arccos(x).$$

3. If $-1 \leqq x < 0$, then $\arcsin(x) = -\arcsin|x|$
   and $\arccos(x) = \pi - \arccos|x|$.
   This reduces these cases to one of the two positive cases.

Effect of an Argument Error

$E \sim \frac{\Delta}{\sqrt{1-x^2}}$. For a small value of x, $E \sim \Delta$. Towards the limits (± 1) of the range, a small Δ causes a substantial error in the answer. For arcsin, $\epsilon \sim \delta$ if the value of x is small.

Arctangent Subprograms

ATAN

Algorithm

1. Reduce the computation of arctan (x) to the case $0 \leqq x \leqq 1$, by using

   $\arctan(-x) = -\arctan(x)$, or

   $$\arctan\left(\frac{1}{|x|}\right) = \frac{\pi}{2} - \arctan|x|.$$

2. If necessary, reduce the computation further to the case $|x| \leqq \tan 15°$ by using

   $$\arctan(x) = 30° + \arctan\left(\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}\right).$$

   The value of $\left|\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}\right| \leqq \tan 15°$ if the value of x is within the range, $\tan 15° < x \leqq 1$. The value of $(\sqrt{3} \cdot x - 1)$ is computed as $(\sqrt{3} - 1)\,x - 1 + x$ to avoid the loss of significant digits.

3. For $|x| \leqq \tan 15°$, use the approximation formula:

   $$\frac{\arctan(x)}{x} \cong 0.60310579 - 0.05160454x^2 + \frac{0.55913709}{x^2 + 1.4087812}.$$

   This formula has a relative error less than $2^{-27.1}$ and can be obtained by transforming the continued fraction

   $$\frac{\arctan(x)}{x} = 1 - \frac{x^2}{3 + \dfrac{\dfrac{x^2}{5}}{\left(\dfrac{5}{7} + x^{-2}\right) - w}}$$

   where w has an approximate value of $\left(-\frac{75}{77}x^{-2} + \frac{3375}{77}\right) 10^{-4}$, but the true value of w is $\dfrac{\dfrac{4 \cdot 5}{7 \cdot 7 \cdot 9}}{\left(\dfrac{43}{7 \cdot 11} + x^{-2}\right) + \cdots}$.

   The original continued fraction can be obtained by transforming the Taylor series into continued fraction form.

Effect of an Argument Error

$E \sim \frac{\Delta}{1 + x^2}$. For small values of x, $\epsilon \sim \delta$; as the value of x increases, the effect of δ upon ϵ diminishes.

ATAN/ATAN2

Algorithm

1. For arctan (x_1, x_2):
   If $x_1 < 0$, use the identity $\arctan(x_1, x_2) = -\arctan(-x_1, x_2)$.
   Hence we may assume that $x_1 \geqq 0$. Then:

   If either $x_2 = 0$ or $\left|\frac{x_1}{x_2}\right| > 2^{24}$, the answer $= \frac{\pi}{2}$.

   If $x_2 < 0$ and $\left|\frac{x_1}{x_2}\right| < 2^{-24}$, the answer $= \pi$.

For the general case, if $x_2 > 0$, the answer = $\arctan\left(\left|\frac{x_1}{x_2}\right|\right)$, and

if $x_2 < 0$, the answer = $\pi - \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$.

2. The computation of $\arctan\left(\left|\frac{x_1}{x_2}\right|\right)$ above, or of $\arctan(x)$ for the single argument case, follows the algorithm given for the subprogram ATAN with a single argument.

Effect of an Argument Error

$E \sim \frac{\Delta}{1 + x^2}$. For small values of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of ϵ upon δ diminishes.

DATAN

Algorithm

1. Reduce the computation of $\arctan(x)$ to the case $0 \leqq x \leqq 1$ by using

   $\arctan(-x) = -\arctan(x)$ and

   $\arctan\frac{1}{|x|} = \frac{\pi}{2} - \arctan|x|$.

2. If necessary, reduce the computation further to the case $|x| \leqq \tan 15°$ by using

   $\arctan(x) = 30° + \arctan\left(\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}\right)$.

   The value of $\left|\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}\right| \leqq \tan 15°$, if the value of x is within the range $\tan 15° < x \leqq 1$. The value of $(\sqrt{3} \cdot x - 1)$ is computed as $(\sqrt{3} - 1)\,x - 1 + x$ to avoid the loss of significant digits.

3. For $|x| \leqq \tan 15°$, use a continued fraction of the form:

   $$\frac{\arctan(x)}{x} \cong 1 + x^2\left[b_0 - \frac{a_1}{(b_1 + x^2) -} \frac{a_2}{(b_2 + x^2) -} \frac{a_3}{(b_3 + x^2)}\right].$$

   The relative error of this approximation is less than $2^{-60.7}$.
   The coefficients of this formula were derived by transforming a minimax rational approximation (in relative error, over the range $0 \leqq x^2 \leqq 0.071797$) for $\arctan(x)/x$ of the following form:

   $$\frac{\arctan(x)}{x} \cong a_0 + x^2\left[\frac{c_0 + c_1x^2 + c_2x^4 + c_3x^6}{d_0 + d_1x^2 + d_2x^4 + x^6}\right].$$

   Minimax was taken under the constraint that $a_0 = 1$ exactly.

Effect of an Argument Error

$E \sim \frac{\Delta}{1 + x^2}$. For small values of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of ϵ upon δ diminishes.

DATAN/DATAN2

Algorithm

1. For $\arctan(x_1, x_2)$:
   If $x_1 < 0$, use the identity $\arctan(x_1, x_2) = -\arctan(-x_1, x_2)$.
   Hence we may assume that $x_1 \geqq 0$. Then:

   If either $x_2 = 0$ or $\left|\frac{x_1}{x_2}\right| > 2^{56}$, the answer = $\frac{\pi}{2}$.

If $x_2 < 0$ and $\left|\frac{x_1}{x_2}\right| < 2^{-56}$, the answer $= \pi$.

For the general case, if $x_2 > 0$, the answer $= \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$, and

if $x_2 < 0$, the answer $= \pi - \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$.

2. The computation of $\arctan\left(\left|\frac{x_1}{x_2}\right|\right)$ above, or of $\arctan(x)$ for the single argument case, follows the algorithm given for the subprogram DATAN with a single argument.

Effect of an Argument Error

$E \sim \frac{\Delta}{1+x^2}$. For small values of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of ϵ upon δ diminishes.

QATAN/QATAN2

Algorithm

1. For $\arctan(x)$, if $x < 0$, then $\arctan(x) = -\arctan(|x|)$. So assume $x \geqq 0$.
2. Define break points β_i, $i = 0, 1, 2, \ldots, 8$ as $\beta_i = \tan\left(\frac{2i-1}{32}\pi\right)$.

   Define origins θ_i to be approximately $\frac{i}{16}\pi$, $i = 0, 1, 2, \ldots, 8$ in such a way that $\tan\theta_i$ are exact short form numbers. $\theta_8 = \frac{\pi}{2}$ exactly.
3. $\beta_i \leqq x < \beta_{i+1}$ for $i = 0, 1, 2, \ldots, 7$, then use the following reduction:

   $$\arctan(x) = \theta_i + \arctan\left(\frac{x - \tan\theta_i}{1 + x\tan\theta_i}\right)$$

   If $\beta_8 \leqq x < \infty$, use the reduction:

   $$\arctan(x) = \frac{\pi}{2} + \arctan\left(\frac{-1}{x}\right).$$

   Note the quantity within the parentheses on the right is in either case within the basic range (β_0, β_1), that is, is less than $\frac{\pi}{32}$ in magnitude.
4. Within the basic range $-\frac{\pi}{32} \leqq x \leqq \frac{\pi}{32}$, a minimax approximation of the following form is used to compute $\arctan(x)$:

   $$\arctan(x) \cong x + a_1 x^3 + a_2 x^5 + \ldots + a_{12} x^{25}$$

   The relative error of this approximation is less than 2^{-112}.
   It is sufficient to compute the last three terms in double precision.
5. For $\arctan(x_1, x_2)$:
   If $x_1 < 0$, use the identity $\arctan(x_1, x_2) = -\arctan(|x_1|, x_2)$.
   Hence we may assume that $x_1 \geqq 0$. Then:

   if either $x_2 = 0$ or $\left|\frac{x_1}{x_2}\right| > 2^{112}$, the answer $\cong \frac{\pi}{2}$.

   If $x_2 < 0$ and $\left|\frac{x_1}{x_2}\right| < 2^{-112}$, the answer $\cong \pi$.

   For the general case, if $x_2 > 0$, the answer $= \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$, and

   if $x_2 < 0$, the answer $= \pi - \arctan\left(\left|\frac{x_1}{x_2}\right|\right)$.

Here arctan $\left(\left|\frac{x_1}{x_2}\right|\right)$ is computed as described in steps 1 through 4 above, except for the following simplification for the case $\beta_8 \leqq \left|\frac{x_1}{x_2}\right| < \infty$:

$$\arctan\left(\left|\frac{x_1}{x_2}\right|\right) = \frac{\pi}{2} + \arctan\left(\frac{-|x_2|}{|x_1|}\right).$$

This combines two needed extended precision divisions into one for this case.

Effect of an Argument Error

$E \sim \frac{\Delta}{1 + x^2}$. For a small value of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of δ upon ϵ diminishes.

Error Functions Subprograms

ERF/ERFC

Algorithm

1. If $0 \leqq x \leqq 1$, then compute the error function by the following approximation:

   $$\text{erf}(x) \cong x(a_0 + a_1x^2 + a_2x^4 + \ldots + a_5x^{10}).$$

   The coefficients were obtained by the minimax approximation (in relative error) of $\text{erf}(x)/x$ as a function of x^2 over the range $0 \leqq x^2 \leqq 1$. The relative error of this approximation is less than $2^{-24.6}$. The value of the complemented error function is computed as $\text{erfc}(x) = 1 - \text{erf}(x)$.
2. If $1 < x < 2.040452$, then compute the complemented error function by the following approximation:

   $$\text{erfc}\,(x) \cong b_0 + b_1z + b_2z^2 + \ldots + b_9z^9$$

   where $z = x - T_0$ and $T_0 \cong 1.709472$. The coefficients were obtained by the minimax approximation (in absolute error) of the function $f(z) = \text{erfc}(z + T_0)$ over the range $-0.709472 \leqq z \leqq 0.33098$. The absolute error of this approximation is less than $2^{-31.5}$. The limits of this range and the value of the origin T_0 were chosen to minimize the hexadecimal round-off errors. The value of the complemented error function within this range is between $\frac{1}{256}$ and 0.1573. The value of the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.
3. If $2.040452 \leqq x < 13.306$, then compute the complemented error function by the following approximation:

   $$\text{erfc}(x) \cong e^{-z} \cdot F/x \text{ where } z = x^2 \text{ and}$$

   $$F = c_0 + \frac{c_1 + c_2z + c_3z^2}{d_1z + d_2z^2 + z^3}.$$

   The coefficients for F were obtained by transforming a minimax rational approximation (in absolute errors, over the range $13.306^{-2} \leqq w \leqq 2.040452^{-2}$) of the function $f(w) = \text{erfc}(x) \cdot x \cdot e^{x^2}$, $w = x^{-2}$, of the following form:

   $$f(w) \cong \frac{a_0 + a_1w + a_2w^2 + a_3w^3}{b_0 + b_1w + w^2}.$$

   The absolute error of this approximation is less than $2^{-26.1}$. This computation uses the real exponential subprogram (EXP).

If $2.040452 \leqq x < 3.919206$, then the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.
If $3.919206 \leqq x$, then the error function is $\cong 1$.

4. If $13.306 \leqq x$, then the error function is $\cong 1$, and the complemented error function is $\cong 0$ (underflow).
5. If $x < 0$, then reduce to a case involving a positive argument by the use of the following formulas:

$$\text{erf}(-x) = -\text{erf}(x), \text{ and erfc}(-x) = 2 - \text{erfc}(x).$$

Effect of an Argument Error

$E \sim e^{-x^2} \cdot \Delta$. For the error function, as the magnitude of the argument exceeds 1, the effect of an argument error upon the final accuracy diminishes rapidly. For small values of x, $\epsilon \sim \delta$. For the complemented error function, if the value of x is greater than 1, $\text{erfc}(x) \sim \frac{e^{-x^2}}{2x}$. Therefore, $\epsilon \sim 2x^2 \cdot \delta$. If the value of x is negative or less than 1, then $\epsilon \sim e^{-x^2} \cdot \Delta$.

DERF/DERFC

Algorithm

1. If $0 \leqq x < 1$, then compute the error function by the following approximation:

$$\text{erf}(x) \cong x(a_0 + a_1x^2 + a_2x^4 + \ldots + a_{11}x^{22}).$$

The coefficients were obtained by the minimax approximation (in relative error) of $\text{erf}(x)/x$ as a function of x^2 over the range $0 \leqq x^2 \leqq 1$. The relative error of this approximation is less than $2^{-56.9}$. The value of the complemented error function is computed as $\text{erfc}(x) = 1 - \text{erf}(x)$.

2. If $1 \leqq x < 2.040452$, then compute the complemented error function by the following approximation:

$$\text{erfc}(x) \cong b_0 + b_1z + b_2z^2 + \ldots b_{18}z^{18}$$

where $z = x - T_0$ and $T_0 \cong 1.709472$. The coefficients were obtained by the minimax approximation (in absolute error) of the function $f(z) = \text{erfc}(z + T_0)$ over the range $-0.709472 \leqq z \leqq 0.33098$. The absolute error of this approximation is less than $2^{-60.3}$. The limits of this range and the value of the origin T_0 were chosen to minimize the hexadecimal round-off errors. The value of the complemented error function within this range is between $\frac{1}{256}$ and 0.1573. The value of the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

3. If $2.040452 \leqq x < 13.306$, then compute the complemented error function by the following approximation:

$$\text{erfc}(x) \cong e^{-z} \cdot F/x \text{ where } z = x^2 \text{ and}$$

$$F = c_0 + \frac{d_1}{(z + c_1) +} \frac{d_2}{(z + c_2) +} \cdots \frac{d_6}{(z + c_6) +} \frac{d_7}{(z + c_7)}.$$

The coefficients for F were derived by transforming a minimax rational approximation (in absolute errors, over the range $13.306^{-2} \leqq w \leqq 2.040452^{-2}$) of the function $f(w) = \text{erfc}(x) \cdot x \cdot e^{x^2}$, $w = x^{-2}$, of the following form:

$$f(w) \cong \frac{a_0 + a_1w + a_2w^2 + \ldots + a_7w^7}{b_0 + b_1w + b_2w^2 + \ldots + b_6w^6 + w^7}.$$

The absolute error of this approximation is less than $2^{-57.9}$. This computation uses the real exponential subprogram (DEXP). If $2.040452 \leqq x < 6.092368$, then the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.
If $6.092368 \leqq x$, then the error function is $\cong 1$.

4. If $13.306 \leqq x$, then the error function is $\cong 1$, and the complemented error function $\cong 0$ (underflow).

5. If $x < 0$, then reduce to a case involving a positive argument by the use of the following formulas:

$$\text{erf}(-x) = -\text{erf}(x), \text{ and erfc}(-x) = 2 - \text{erfc}(x).$$

Effect of an Argument Error

$E \sim e^{-x^2} \cdot \Delta$. For the error function, as the magnitude of the argument exceeds 1, the effect of an argument error upon the final accuracy diminishes rapidly. For small values of x, $\epsilon \sim \delta$. For the complemented error function, if the value of x is greater than 1, $\text{erfc}(x) \sim \frac{e^{-x^2}}{2x}$. Therefore, $\epsilon \sim 2x^2 \cdot \delta$. If the value of x is negative or less than 1, then $\epsilon \sim e^{-x^2} \cdot \Delta$.

QERF/QERFC

Algorithm

1. If $0 \leqq x \leqq 1$, then:

Write $a(z) = \frac{\sqrt{\pi}}{2x}\text{erf}(x)$ where $z = x^2$

Then $x.2xa' + a = \frac{\sqrt{\pi}}{2}\frac{d}{dx}(\text{erf}(x))$ where $a' = \frac{da}{dz} = e^{-x^2}$

that is $2za' + a = e^{-z}$

Then $2za'' + 3a' = -e^{-z} = -(2za' + a)$

so that $2za'' + (2z + 3)a' + a = 0$

Now integrate twice

$$2za' + 2za + a - \int_0^z a\,dz - A \qquad \text{where } A \text{ is a constant}$$

But if $z = 0$ then $x = 0$ and $a = 1$ so that $A = 1$

Hence

$$2a' + 2a + \frac{a-1}{z} - \frac{\int_0^z a\,dz}{z} = 0$$

and

$$2a + \int_0^z \left\{2a + \frac{a-1}{z} - \frac{\int_0^z a\,dz}{z}\right\} dz = B = 2$$

Now write $\tilde{a} = 1 + \sum_{i=1}^{m} a_i z^i$ as an approximation to a and solve

$$2\tilde{a} + \int_0^z \left\{2\tilde{a} + \frac{\tilde{a}-1}{z} - \frac{\int_0^z \tilde{a}\,dz}{z}\right\} d_s = \beta + {}_{\tau}T^*_{m+1},$$

Where $T^*_m = \sum_{i=0}^{m} T^*_{m,i} \times x^i$ is the Chebyshev polynomial over the appropriate range.

Equating coefficients of powers of z and multiplying the coefficient f of z^i by i^2 we have:

$$3a_1 + 1 = {}_{\tau}T^*_{m+1,1}$$

$$10a_2 + 3a_1 = 4\,T^*_{m+1,2}$$

$$\vdots$$

$$m(2m+1)a_m + (2m-1)\,am - 1 = m^2\,{}_{\tau}T^*_{m+1,m}$$

$$(2m+1)\,am = (m+1)^2\,{}_{\tau}T^*_{m+1,m+1}$$

and we can solve these equations to obtain

$$\frac{2}{\sqrt{\pi}}a_1, \frac{2}{\sqrt{\pi}}a_2, \ldots \frac{2}{\sqrt{\pi}}a_m \text{ and } \tau.$$

Then $\operatorname{erf}(x) = x + \left(\frac{2}{\sqrt{\pi}} - 1\right)x + x\sum_{i=1}^{m}\left(\frac{2}{\sqrt{\pi}}a_i\right)(x^2)^i$

$$= x + x\left\{\left(\frac{2}{\sqrt{\pi}} - 1\right) + \sum_{i=1}^{m}\left(\frac{2}{\sqrt{\pi}}a_i\right)(x^2)^i\right\}$$

2. If $1 \leqq x \leqq 2.84375$, then:

   Write $b(z) = \operatorname{erfc}(x)$ where $z = x - t_0$

   Then $b' = -\frac{2}{\sqrt{\pi}}e^{-x^2}$

   and $b' = \frac{2}{\sqrt{\pi}}2xe^{-x^2}$

   so that $b'' + 2(z + t_0)\,b' = 0$

   Now integrate twice

   $$b' + 2(z + t_0)\,b - 2\int b\,dz = A$$
   $$b + 2(z + t_0)\int b\,dz - 4\iint b\,dz\,dz = Az + B$$

   Let $\tilde{b} = \sum_{i=0}^{m} b_i z^i$ be an approximation to b and solve the equations:

   $$\tilde{b} + 2(z + t_0)\int_0^z \tilde{b}\,dz - 4\int_0^z\int_0^z \tilde{b}\,dz\,dz = Az + B + \tau T^*_{m+2} + \sigma T^*_{m+1}$$

   Now $z = 0 \geqq A = b'(0) + 2t_0 b(0) = \operatorname{erfc}'(t_0) + 2t_0\operatorname{erfc}(t_0)$

   and $B = b(0) = \operatorname{erfc}(t_0)$

   Hence the equations solved are:

   $$b_0 = \tau T^*_{m,0} + \sigma T^*_{m+1,0} + \operatorname{erfc}(t_0)$$
   $$b_1 + 2t_0 b_0 = \tau T^*_{m,1} + \sigma T^*_{m+1,1} + \operatorname{erfc}'(t_0) + 2t_0\operatorname{erfc}(t_0)$$
   $$b_2 + t_0 b_1 = \tau T^*_{m,2} + \sigma T^*_{m+1,2}$$
   $$b_3 + \frac{1}{3}b_1 + \frac{2}{3}t_0 b_2 = \tau T^*_{m,3} + \sigma T^*_{m+1,3}$$
   $$\vdots$$
   $$b_m + \frac{2m-4}{m(m-1)}b_{m-2} + \frac{2}{m}t_0 b_{m-1} = \tau T^*_{m,m} + \sigma T^*_{m+1,m}$$
   $$\frac{2m-2}{(m+1)m}b_{m-1} + \frac{2}{m+1}t_0 b_m = \tau T^*_{m,m+1} + \sigma T^*_{mm,m-1}$$
   $$\frac{2m}{(m+2)(m+1)}b_m = \tau T^*_{m+1,m+2}$$

   Finally

   $\operatorname{erfc}(x) - b(z)$ where $z = x - t_0$

3. If $2.84375 \leqq x \leqq 13.306$, then:

   Write $c(z) = xe^{x^2}\operatorname{erfc}(x)$ where $z = 1/x^2$

   Then $c' = \left\{\frac{c}{x} + 2xc - \frac{2}{\sqrt{\pi}}x\right\}\frac{x^3}{-2}$

   i.e. $2z^2c' + (z + 2)\,c = \frac{2}{\sqrt{\pi}}$

   Let $\tilde{c} = \sum_{i=0}^{m+1} c_i z^i$ be an approximation to c and solve the equations:

   $$2z^2c + (z + 2)\,\tilde{c} = \frac{2}{\sqrt{\pi}} + \tau T^*_{m+2}$$

Then use the approximation

$$T^*_{m+1} \cong 0$$

to approximate $c_{m+1} z^{m+1}$ by a polynomial of degree m.
The equations solved in (A) are:

$$2c_0 = \frac{2}{\sqrt{\pi}} + T^*_{m+2,0}$$

$$c_0 + 2c_1 = \tau\, T^*_{m+2,1}$$

$$\vdots$$

$$(2m+1)c_m + 2c_{m+1} = \tau\, T^*_{m+2,m+1}$$

$$(2m+3)c_{m+1} = \tau\, T^*_{m+2,m+1}$$

Finally:

$$\operatorname{erfc}(x) = c(z) \cdot \frac{e^{-x^2}}{2X} \quad \text{where } z = 1/x^2$$

4. If $13.306 \leqq x$, then $\operatorname{erf}(x) = 1$ and $\operatorname{erfc}(x) = 0$.
5. If $0 > x$, then $\operatorname{erf}(x) = -\operatorname{erf}(-x)$, and $\operatorname{erfc}(x) = 2 - \operatorname{erfc}(-x)$.

Effect of an Argument Error

$E \sim e^{-x^2} \cdot \Delta$. For the error function, as the magnitude of the argument exceeds 1, the effect of an argument error upon the final accuracy diminishes rapidly. For small values of X, $\epsilon \sim \delta$. For the complicated error function, if the value of X is greater than 1, $\operatorname{erfc}(x) \sim \frac{e^{-x^2}}{2X}$.

Therefore, $\epsilon \sim 2X^2 \cdot \delta$. If the value of x is negative or less than 1, then $\epsilon \sim e^{-x^2} \cdot \Delta$.

Exponential Subprograms

EXP

Algorithm

1. If $x < -180.218$, then 0 is given as the answer via floating-point underflow.
2. Otherwise, divide x by $\log_e 2$ and write

$$y = \frac{x}{\log_e 2} = 4a - b - d$$

where a and b are integers, $0 \leqq b \leqq 3$ and $0 \leqq d < 1$.

3. Compute 2^{-d} by the following fractional approximation:

$$2^{-d} \cong 1 - \frac{2d}{0.034657359\, d^2 + d + 9.9545948 - \dfrac{617.97227}{d^2 + 87.417497}}.$$

This formula can be obtained by transforming the Gaussian continued fraction

$$e^{-z} = 1 - \frac{z}{1+}\ \frac{z}{2-}\ \frac{z}{3+}\ \frac{z}{2-}\ \frac{z}{5+}\ \frac{z}{2-}\ \frac{z}{7+}\ \frac{z}{2}.$$

The maximum relative error of this approximation is 2^{-29}.
Multiply 2^{-d} by 2^{-b}.
Finally, add the hexadecimal exponent a to the characteristic of the answer.

Effect of an Argument Error

$\epsilon \sim \Delta$. If the magnitude of x is large, even the roundoff error of the argument causes a substantial relative error in the answer because $\Delta = \delta \cdot x$.

DEXP

Algorithm

1. If $x < -180.2187$, then 0 is given as the answer via floating-point underflow.
2. Divide x by $\log_e 2$ and write

$$x = \left(4a - b - \frac{c}{16}\right) \cdot \log_e 2 - r$$

where a, b, and c are integers, $0 \leqq b \leqq 3$, $0 \leqq c \leqq 15$, and the remainder r is within the range $0 \leqq r < \frac{1}{16} \cdot \log_e 2$. This reduction is carried out in an extra precision to ensure accuracy. Then $e^x = 16^a \cdot 2^{-b} \cdot 2^{-c/16} \cdot e^{-r}$.

3. Compute e^{-r} by using a minimax polynomial approximation of degree 6 over the range $0 \leqq r < \frac{1}{16} \cdot \log_e 2$. In obtaining coefficients of this approximation, the minimax of relative errors was taken under the constraint that the constant term a_0 shall be exactly 1. The relative error is less than $2^{-56.87}$.
4. Multiply e^{-r} by $2^{-c/16}$. The 16 values of $2^{-c/16}$ for $0 \leqq c \leqq 15$ are included in the subprogram. Then halve the result b times.
5. Finally, add the hexadecimal exponent of a to the characteristic of the answer.

Effect of an Argument Error

$\epsilon \sim \Delta$. If the magnitude of x is large, even the roundoff error of the argument causes a substantial relative error in the answer because $\Delta = \delta \cdot x$.

CEXP/CDEXP

Algorithm

The value of e^{x+iy} is computed as $e^x \cdot \cos(y) + i \cdot e^x \cdot \sin(y)$. The algorithms for both complex exponential subprograms are identical. Each subprogram uses the appropriate real exponential subprogram (EXP or DEXP) and the appropriate real sine/cosine subprogram (COS/SIN or DCOS/DSIN).

Effect of an Argument Error

The effect of an argument error depends upon the accuracy of the individual parts of the argument. If $e^{x+iy} = R \cdot e^{iH}$, then $H = y$ and $\epsilon(R) \sim \Delta(x)$.

QEXP

Algorithm

1. Basic computation is that of 2^x. For QEXP entry, multiply x by $\log_e$ in a 31 hexadecimal digit arithmetic, and raise the result to the power of 2.
2. Decompose x as $x = 4p - q - r$ where p is an integer, $q = 0, 1, 2$, or 3, and $0 \leqq r < 1$.
3. Find two indices i, j, $0 \leqq i \leqq 8, 0 \leqq j \leqq 3$ such that $4i + j$ is the integer nearest to $32r$.

   Using these indices, select two encoded constants α_i, β_j where $\alpha_i = [2^{-i/8}]$, $\beta_j = [2^{-j/32}]$.

   Here the bracket indicates rounding to the nearest 17 binary digit number. Obtain the product $\Psi_{ij} = \alpha_i \beta_j$.

4. Obtain the reduced argument $s = -r - \log_2(\Psi_{ij})$ accurately by subtracting $\log_2(\Psi_{ij}) = \log_2 \alpha_i + \log_2 \beta_j +$ in an extra precision. Constants $\log_2 \alpha_i$ and $\log_2 \beta_j$ are encoded in 31 hexadecimal digits of accuracy. Then s is approximately bounded by $\pm\frac{1}{64}$.
5. Compute 2^s by a minimax approximation of the form:

$$2^s \cong 1 + \frac{2sP(s^2)}{Q(s^2) - sp(s^2)}$$

where P and Q are polynomials of degree 2.
6. Then $2^x = 16^p \cdot (2^{-q}\Psi_{ij}) \cdot 2^8$. In assembling this product a virtual rounding is applied.
7. The limited use of extra precision arithmetic in the above computation enhances accuracy of both *QEXP* and A**B application (see note below).

Effect of an Argument Error

$\epsilon \sim \Delta$. If the magnitude of x is large, even the roundoff error of the argument causes a substantial relative error in the answer because $\Delta = \delta \cdot x$.

CQEXP

Algorithm

The value of e^{x+iy} is computed as $e^x \cdot \cos(y) + i \cdot e^x \cdot \sin(y)$. The algorithms for both complex exponential subprograms are identical. Each subprogram uses the appropriate real exponential subprogram (QEXP) and the appropriate real sine/cosine subprogram (QCOS/QSIN).

Effect of an Argument Error

The effect of the argument error depends upon the accuracy of the individual parts of the agreement. If $e^{x+iy} = R \cdot e^{iH}$, then $H = y$ and $\epsilon(R) \sim \Delta(x)$.

Gamma and Log Gamma Subprograms

GAMMA/ALGAMA/LGAMMA

Algorithm

1. If $0 < x \leqq 2^{-252}$, then compute log-gamma as $\log_e \Gamma(x) \cong -\log_e(x)$. This computation uses the real logarithm subprogram (ALOG).
2. If $2^{-252} < x < 8$, then compute log-gamma by taking the natural logarithm of the value obtained for gamma. The computation of gamma depends upon the range into which the argument falls.
3. If $2^{-252} < x < 1$, then use $\Gamma(x) = \frac{\Gamma(x+1)}{x}$ to reduce to the next case.
4. If $1 \leqq x \leqq 2$, then compute gamma by the minimax rational approximation (in absolute error) of the following form:

$$\Gamma(x) \cong c_0 + \frac{z[a_0 + a_1 z + a_2 z^2 + a_3 z^3]}{b_0 + b_1 z + b_2 z^2 + z^3}$$

where $z = x - 1.5$. The absolute error of this approximation is less than $2^{-25.9}$.
5. If $2 < x < 8$, then use $\Gamma(x) = (x-1)\,\Gamma(x-1)$ to reduce step by step to the preceding case.
6. If $8 \leqq x$, then compute log-gamma by the use of Stirling's formula:

$$\log_e\Gamma(x) \cong x(\log_e(x) - 1) - \tfrac{1}{2}\log_e(x) + \tfrac{1}{2}\log_e(2\pi) + G(x).$$

The modifier term $G(x)$ is computed as

$$G(x) \cong d_0x^{-1} + d_1x^{-2}.$$

These coefficients were obtained by a form of minimax approximation minimizing the ratio of the absolute error to the value of x. The absolute error is less than $x \cdot 2^{-26.2}$. Remembering the fact that $x < \log_e\Gamma(x)$ in this range, the contribution of this error to the relative error of the value for log-gamma is less than $2^{-26.2}$. This computation uses the real logarithm subprogram (ALOG).

For gamma, compute $\Gamma(x) = e^y$, where y is the value obtained for log-gamma. This computation uses the real exponential subprogram (EXP).

Effect of an Argument Error

$\epsilon \sim \psi(x) \cdot \Delta$ for gamma, and $E \sim \psi(x) \cdot \Delta$ for log-gamma, where ψ is the digamma function.

If $\frac{1}{2} < x < 3$, then $-2 < \psi(x) < 1$. Therefore, $E \sim \Delta$ for log-gamma. However, because $x = 1$ and $x = 2$ are zeros of the log-gamma function, even a small δ can cause a substantial ϵ in this range.

If the value of x is large, then $\psi(x) \sim \log_e(x)$. Therefore, for gamma, $\epsilon \sim \delta x \cdot \log_e(x)$. In this case, even the roundoff error of the argument contributes greatly to the relative error of the answer. For log-gamma with large values of x, $\epsilon \sim \delta$.

DGAMMA/DLGAMA/LGAMMA

Algorithm

1. If $0 < x \leqq 2^{-252}$, then compute log-gamma as $\log_e\Gamma(x) \cong -\log_e(x)$.
   This computation uses the real logarithm subprogram (DLOG).
2. If $2^{-252} < x < 8$, then compute log-gamma by taking the natural logarithm of the value obtained for gamma. The computation of gamma depends upon the range into which the argument falls.
3. If $2^{-252} < x < 1$, then use $\Gamma(x) = \dfrac{\Gamma(x+1)}{x}$ to reduce to the next case.
4. If $1 \leqq x \leqq 2$, then compute gamma by the minimax rational approximation (in absolute error) of the following form:
   $$\Gamma(x) \cong c_0 + \frac{z[a_0 + a_1z + \ldots + a_6z^6]}{b_0 + b_1z + \ldots + b_6z^6 + z^7}$$
   where $z = x - 1.5$. The absolute error of this approximation is less than $2^{-59.3}$.
5. If $2 < x < 8$, then use $\Gamma(x) = (x-1)\,\Gamma(x-1)$ to reduce to the preceding case.
6. If $8 \leqq x$, then compute log-gamma by the use of Stirling's formula:
   $$\log_e\Gamma(x) \cong x(\log_e(x) - 1) - \tfrac{1}{2}\log_e(x) + \tfrac{1}{2}\log_e(2\pi) + G(x).$$
   The modifier term $G(x)$ is computed as
   $$G(x) \cong d_0x^{-1} + d_1x^{-3} + d_2x^{-5} + d_3x^{-7} + d_4x^{-9}.$$
   These coefficients were obtained by a form of minimax approximation minimizing the ratio of the absolute error to the value of x. The absolute error is less than $x \cdot 2^{-56.1}$. Remembering the fact that $x < \log_e\Gamma(x)$ in this range, the contribution of this error to the relative error of the value for log-gamma is less than $2^{-56.1}$. This computation uses the real logarithm subprogram (DLOG). For gamma, compute $\Gamma(x) = e^y$, where y is the value obtained for log-gamma. This computation uses the real exponential subprogram (DEXP).

Effect of an Argument Error

$\epsilon \sim \psi(x) \cdot \Delta$ for gamma, and $E \sim \psi(x) \cdot \Delta$ for log-gamma, where ψ is the digamma function.

If $\frac{1}{2} < x < 3$, then $-2 < \psi(x) < 1$. Therefore, $E \sim \Delta$ for log-gamma. However, because $x = 1$ and $x = 2$ are zeros of the log-gamma function, even a small δ can cause a substantial ϵ in this range.

If the value of x is large, then $\psi(x) \sim \log_e(x)$. Therefore, for gamma, $\epsilon \sim \delta \cdot x \cdot \log_e(x)$. In this case, even the round-off error of the argument contributes greatly to the relative error of the answer. For log-gamma with large values of x, $\epsilon \sim \delta$.

LGAMMA

Algorithm

If x is R*4, then LGAMMA (x) = ALGAMA (x).
If x is R*8, then LGAMMA (x) = DLGAMA (x).

Hyperbolic Sine and Cosine Subprograms

SINH/COSH

Algorithm

1. If $|x| < 1.0$, then compute $\sinh(x)$ as:
   $$\sinh(x) \cong x + c_1x^3 + c_2x^5 + c_3x^7.$$
   The coefficient c_i were obtained by the minimax approximation (in relative error) of $\frac{\sinh(x)}{x}$ as a function of x^2. The maximum relative error of this approximation is $2^{-25.6}$.
2. If $x \geqq 1.0$, then $\sinh(x)$ is computed as:
   $$\sinh(x) = (1 + \delta)\,[e^{x+\log_e v} - v^2/e^{x+\log_e v}].$$
   Here, $1 + \delta = \frac{1}{2v}$, so that this expression is theoretically equivalent to $[e^x - e^{-x}]/2$. The value of v (and consequently those of $\log_e v$ and δ) was so chosen as to satisfy the following conditions:
   a) v is slightly less than ½, so that $\delta > 0$ and small.
   b) $\log_e v$ is an exact multiple of 2^{-16}.
   The condition *b)* ensures that the addition $x + \log_e v$ is carried out exactly. This maneuver was designed to reduce the roundoff errors and also to enlarge the limits of acceptable arguments. This computation uses the real exponential subprogram (EXP).
3. If $x \leqq -1.0$, use $\sinh(x) = -\sinh(|x|)$ to reduce to case 2 above.
4. If cosh (x) is desired, then for all valid values of arguments use the identity: $\cosh(x) = (1 + \delta)\,[e^{x+\log_e v} + v^2/e^{x+\log_e v}]$. Here the notation and the consideration are identical to case 2 above. This computation uses the real exponential subprogram (EXP).

Effect of an Argument Error

For the hyperbolic sine, $E \sim \Delta \cdot \cosh(x)$ and $\epsilon \sim \Delta \cdot \coth(x)$.

For the hyperbolic cosine, $E \sim \Delta \cdot \sinh(x)$ and $\epsilon \sim \delta \cdot \tanh(x)$.

Specifically, for the cosine, $\epsilon \sim \Delta$ over the entire range; for the sine, $\epsilon \sim \delta$ for small values of x.

DSINH/DCOSH

Algorithm

1. If $|x| < 0.881374$, then compute $\sinh(x)$ as:
   $$\sinh(x) \cong c_0x + c_1x^3 + c_2x^5 + \ldots + c_6x^{13}.$$
   The coefficients c_i were obtained by the minimax approximation (in relative error) of $\frac{\sinh(x)}{x}$ as the function of x^2. Minimax was taken under the constraint that $c_0 = 1$ exactly. The maximum relative error of this approximation is $2^{-55.7}$.
2. If $x \geqq 0.881374$, then $\sinh(x)$ is computed as:
   $$\sinh(x) = (1 + \delta)\,[e^{x+\log_e v} - v^2/e^{x+\log_e v}].$$
   Here, $1 + \delta = \frac{1}{2v}$, so that this expression is theoretically equivalent to $[e^x - e^{-x}]/2$. The value of v (and consequently those of $\log_e v$ and δ) was so chosen as to satisfy the following conditions:
   a) v is slightly less than ½, so that $\delta > 0$ and small.
   b) $\log_e v$ is an exact multiple of 2^{-16}.
   The condition *b*) insures that the addition $x + \log_e v$ is carried out exactly. This maneuver was designed to reduce the round-off errors and also to enlarge the limits of acceptable arguments. This computation uses the real exponential subprogram (DEXP).
3. If $x \leqq -0.881374$, then use $\sinh(x) = -\sinh(|x|)$ to reduce to case 2 above.
4. If $\cosh(x)$ is desired, then, for all valid arguments use the identity:
   $\cosh(x) = (1 + \delta)\,[e^{x+\log_e v} + v^2/e^{x+\log_e v}]$. Here the notation and the consideration are identical to case 2 above. This computation uses the real exponential subprogram (DEXP).

Effect of an Argument Error

For the hyperbolic sine, $E \sim \Delta \cdot \cosh(x)$ and $\epsilon \sim \Delta \cdot \coth(x)$.
For the hyperbolic cosine, $E \sim \Delta \cdot \sinh(x)$ and $\epsilon \sim \Delta \cdot \tanh(x)$.

Specifically, for the cosine, $\epsilon \sim \Delta$ over the entire range; for the sine, $\epsilon \sim \delta$ for the small values of x.

QSINH/QCOSH

Algorithm

1. If $|x| < 1$ then compute $\sinh(x)$ as:
   $$\sinh(x) \cong c_0x + c_1x^3 + c_2x^5 + \ldots + c_{12}x^{25}.$$
   The coefficients c_i were obtained by the minimax approximation (in relative error) of $\frac{\sinh(x)}{x}$ as the function of x^2. Minimax was taken under the constraint that $c_0 = 1$ exactly. The maximum relative error of this approximation is less than 2^{-112}.
2. If $x \geqq 1$ then $\sinh(x)$ is computed as:
   $$\sinh(x) = (1 + \delta)\,[e^{x+\log_e v} - v^2/e^{x+\log_e v}].$$
   Here, $1 + \delta = \frac{1}{2v}$, so that this expression is theoretically equivalent to $[e^x - e^{-x}]/2$. The value of v (and consequently those of $\log_e v$ and δ) was so chosen as to satisfy the following conditions.

a) v is slightly less than $\frac{1}{2}$, so that $\delta > 0$ and small.

b) $\log_e v$ is an exact multiple of 2^{-16}.

The condition b) insures that the addition $x + \log_e v$ is carried out exactly. This maneuver was designed to reduce the round-off errors and also to enlarge the limits of acceptable arguments. This computation uses the exponential subprogram. Accuracy of the quotient $v^2/e^{x+\log_e v}$ is not critical if x is large. For $x >$ 21.85, a double precision division yields a sufficiently accurate result.

3. If $x \leqq -1$ then use $\sinh(x) = -\sinh(|x|)$ to reduce the case to 2 above.
4. If $\cosh(x)$ is desired, for all allowable arguments use the identity: $\cosh(x) = (1 + \delta)\,[e^{x+\log_e v} + v^2/e^{x+\log_e v}]$. Here the notation and the consideration are identical to the case 2 above.

Effect of an Argument Error

For hyperbolic sine, $E \sim \Delta \cdot \cosh(x)$ and $\epsilon \sim \Delta \cdot \coth(x)$. For hyperbolic cosine, $E \sim \Delta \cdot \sinh(x)$ and $\epsilon \sim \delta \cdot \tanh(x)$. In other words, for cosine, $\epsilon \sim \Delta$ over the the entire range; for sine $\epsilon \sim \delta$ for small values of x.

Hyperbolic Tangent Subprograms

TANH

Algorithm

1. If $|x| \leqq 2^{-12}$, then $\tanh(x) \cong x$.
2. If $2^{-12} < |x| \leqq 0.7$, use the following fractional approximation:

$$\frac{\tanh(x)}{x} \cong 1 - x^2\left[0.0037828 + \frac{0.3145651}{x^2 + 2.471749}\right].$$

The coefficients of this approximation were obtained by taking the minimax of relative error, over the range $x^2 < 0.49$, of approximations of this form under the constraint that the first term shall be exactly 1.0. The maximum relative error of this approximation is $2^{-26.4}$.

3. If $0.7 < x < 9.011$, then use the identity $\tanh(x) = 1 - \frac{2}{(e^x)^2 + 1}$.

The computation for this case uses the real exponential subprogram (EXP).

4. If $x \geqq 9.011$, than $\tanh(x) \cong 1$.
5. If $x < -0.7$, then use the identity $\tanh(x) = -\tanh(-x)$.

Effect of an Argument Error

$E \sim (1 - \tanh^2 x)\,\Delta$, and $\epsilon \sim \frac{2\Delta}{\sinh(2x)}$. For small values of x, $\epsilon \sim \delta$, and as the value of x increases, the effect of δ upon ϵ diminishes.

DTANH

Algorithm

1. If $|x| \leqq 2^{-28}$, then $\tanh(x) \cong x$.
2. If $2^{-28} < |x| < 0.54931$, use the following fractional approximation:

$$\frac{\tanh(x)}{x} \cong c_0 + \frac{d_1 x^2}{x^2 + c_1 +} + \frac{d_2}{x^2 + c_2 +} \frac{d_3}{x^2 + c_3}.$$

This approximation was obtained by rewriting a minimax approximation of the following form:

$$\frac{\tanh(x)}{x} \cong c_0 + x^2 \cdot \frac{a_0 + a_1 x^2 + a_2 x^4}{b_0 + b_1 x^2 + b_2 x^4 + x^6}.$$

Here the minimax of relative error, over the range $x^2 \leqq 0.30174$, was taken under the constraint that c_0 shall be exactly 1.0. The maximum relative error of the above is 2^{-63}.

3. If $0.54931 \leqq x < 20.101$, then use the identity $\tanh(x) = 1 - \frac{2}{e^{2x} + 1}$.

   This computation uses the double precision exponential subprogram (DEXP).
4. If $x \geqq 20.101$, then $\tanh(x) \cong 1$.
5. If $x \leqq -0.54931$, then use the identity $\tanh(x) = -\tanh(-x)$.

Effect of an Argument Error

$E \sim (1 - \tanh^2 x)\,\Delta$, and $\epsilon \sim \frac{2\Delta}{\sinh(2x)}$. For small values of x, $\epsilon \sim \delta$. As the value of x increases, the effect of δ upon ϵ diminishes.

QTANH

Algorithm

1. If $|x| \leqq 0.54931$, use a minimax fractional approximation of the following form:

   $$\tanh(x) \cong x + \frac{x^3(a_0 + a_1x^2 + a_2x^4 + a_3x^6 + a_4x^8)}{b_0 + b_1x^2 + b_2x^4 + b_3x^6 + b_4x^8 + x^{10}}$$

   Approximation of this form attains accuracy better than 2^{-112} for x in the above range.
2. If $0.54931 < x \leqq 39.1628$, compute $\tanh(x)$ with the aid of the exponential subroutines as follows:

   $$\tanh(x) = 1 - \frac{2}{e^{2x}+1}.$$

   Here if $x > 21.14$, the division is carried out in double precision to save execution time. The quotient term is so small relative to 1 that double precision is accurate enough.
3. If $x > 39.1628$, then $\tanh(x) \cong 1$.
4. If $x \leqq -0.54931$, then use the identity $\tanh(x) = -\tanh(-x)$ to reduce the case to either 3. or 4. above.

Effect of an Argument Error

$E \sim (1 - \tanh^2 x)\,\Delta$, and $\epsilon \sim \frac{2\Delta}{\sinh(2x)}$. For small values of x, $\epsilon \sim \delta$. As the value of x increases, the effect of δ upon ϵ diminishes.

Logarithmic Subprograms (Common and Natural)

ALOG/ALOG10

Algorithm

1. Write $x = 16^p \cdot 2^{-q} \cdot m$ where p is the exponent, q is an integer, $0 \leqq q \leqq 3$, and m is within the range, $\frac{1}{2} \leqq m < 1$.
2. Define two constants, a and b (where a = base point and $2^{-b} = a$), as follows:

   If $\frac{1}{2} \leqq m < \frac{1}{\sqrt{2}}$, then $a = \frac{1}{2}$ and $b = 1$.

   If $\frac{1}{\sqrt{2}} \leqq m < 1$, then $a = 1$ and $b = 0$.
3. Write $z = \frac{m-a}{m+a}$. Then, $m = a \cdot \frac{1+z}{1-z}$ and $|z| < 0.1716$.
4. Now, $x = 2^{4p-q-b} \cdot \frac{1+z}{1-z}$, and $\log_e(x) = (4p - q - b)\log_e 2 + \log_e\left(\frac{1+z}{1-z}\right)$.

5. To obtain $\log_e\left(\frac{1+z}{1-z}\right)$, first compute $w = 2z = \frac{m-a}{0.5m+0.5a}$ (which is represented with slightly more significant digits than z itself), and apply an approximation of the following form:

$$\log_e\left(\frac{1+z}{1-z}\right) \cong w\left[c_0 + \frac{c_1 w^2}{c_2 - w^2}\right].$$

These coefficients were obtained by the minimax rational approximation of $\frac{1}{2z}\log_e\left(\frac{1+z}{1-z}\right)$ over the range $z^2 \epsilon (0, 0.02944)$ under the constraint that c_0 shall be exactly 1.0. The maximum relative error of this approximation is less than $2^{-25.33}$.

6. If the common logarithm is desired, then $\log_{10}x = \log_{10}e \cdot \log_e x$.

Effect of an Argument Error

$E \sim \delta$. Specifically, if δ is the roundoff error of the argument, for example, $\delta \sim 6 \cdot 10^{-8}$, then $E \sim \delta\, 6 \cdot 10^{-8}$. Therefore, if the argument is close to 1, the relative error can be very large because the value of the function is very small.

DLOG/DLOG10

Algorithm

1. Write $x = 16^p \cdot 2^{-q} \cdot m$ where p is the exponent, q is an integer, $0 \leqq q \leqq 3$, and m is within the range $\frac{1}{2} \leqq m < 1$.
2. Define two constants, a and b (where a = base point and $2^{-b} = a$), as follows:

   If $\frac{1}{2} \leqq m < \frac{1}{\sqrt{2}}$, then $a = \frac{1}{2}$ and $b = 1$.

   If $\frac{1}{\sqrt{2}} \leqq m < 1$, then $a = 1$ and $b = 0$.

3. Write $z = \frac{m-a}{m+a}$. Then, $m = a \cdot \frac{1+z}{1-z}$ and $|z| < 0.1716$.
4. Now, $x = 2^{4p-q-b} \cdot \frac{1+z}{1-z}$, and $\log_e x = (4p - q - b)\log_e 2 + \log_e\left(\frac{1+z}{1-z}\right)$.
5. To obtain $\log_e\left(\frac{1+z}{1-z}\right)$, first compute $w = 2z = \frac{m-a}{0.5m+0.5a}$ (which is represented with slightly more significant digits than z itself), and apply an approximation of the following form:

$$\log_e\left(\frac{1+z}{1-z}\right) \cong w\left[c_0 + c_1 w^2\left(w^2 + c_2 + \frac{c_3}{w^2 + c_4 + \frac{c_5}{w^2 + c_6}}\right)\right].$$

These coefficients were obtained by the minimax rational approximation of $\frac{1}{2z}\log_e\left(\frac{1+z}{1-z}\right)$ over the range $z^2 \epsilon (0, 0.02944)$ under the constraint that c_0 shall be exactly 1.0. The maximum relative error of this approximation is less than $2^{-60.55}$.

6. If the common logarithm is desired, then $\log_{10}x = \log_{10}e \cdot \log_e x$.

Effect of an Argument Error

$E \sim \delta$. Therefore, if the value of the argument is close to 1, the relative error can be very large because the value of the function is very small.

CLOG/CDLOG

Algorithm

1. Write $\log_e(x + iy) = a + ib$.
2. Then, $a = \log_e|x + iy|$ and $b =$ the principal value of arctan (y, x).
3. $\log_e|x + iy|$ is computed as follows:
   Let $v_1 = \max(|x|, |y|)$, and $v_2 = \min(|x|, |y|)$.

   Let t be the exponent of v_1, i.e., $v_1 = m \cdot 16^t$, $\frac{1}{16} \leqq m < 1$.

   Finally, let $t_1 = \begin{cases} t \text{ if } t \leqq 0 \\ t - 1 \text{ if } t > 0 \end{cases}$,

   and $s = 16^{t_1}$.

   Then, $\log_e|x + iy| = 4t_1 \cdot \log_e(2) + \frac{1}{2}\log_e\left[\left(\frac{v_1}{s}\right)^2 + \left(\frac{v_2}{s}\right)^2\right]$.

Computation of v_1/s and v_2/s are carried out by manipulation of the characteristics of v_1 *and* v_2. In particular, if v_2/s is very small, it is taken to be 0. The algorithms for both complex logarithm subprograms are identical. Each subprogram uses the appropriate real natural logarithm subprogram (ALOC or DLOC) and the appropriate arctangent subprogram (ATAN2 or DATAN2).

Effect of an Argument Error

The effect of an argument error depends upon the accuracy of the individual parts of the argument. If $x + iy = r \cdot e^{ih}$ and $\log_e(x + iy) = a + ib$, then $h = b$ and $E(a) = \delta(r)$.

QLOG/QLOG10

Algorithm

1. Decompose x as $x = 16^p \cdot 2^{-q} \cdot m$, where $\frac{1}{2} \leqq m < 1$.
2. Make an estimate of $\log_2 m$ and define three indices $0 \leqq i \leqq 8$, $0 \leqq i \leqq 3$, $0 \leqq k \leqq 4$ so that $20i + 5i + k$ is the nearest integer to $-160 \cdot \log_2 m$. Using these indices, select three constants $\alpha_i, \beta_j, \gamma_k$ where
   $$\alpha_i = [2^{-i/8}],\ \beta_j = [2^{-j/32}],\ \gamma_k = [2^{-k/160}].$$
   Here the bracket indicates rounding to the nearest 17 digit binary number. Obtain the exact product $\varphi_{ijk} = \alpha_i \beta_j \gamma_k$ by use of *ME* and *MXD* instructions. The 18 short constants α_i, β_j, and γ_k are encoded in the subroutine.
3. Denote $z = (m - \varphi_{ijk})/(m + \varphi_{ijk})$.
   Compute $w = 2z/\log_e(2) = (m - \varphi_{ijk})/[0.5 \cdot \log_e(2) \cdot (m + \varphi_{ijk})]$.

   The computed w is bounded approximately by $\pm \frac{1}{320}$, and it has 112 bit accuracy.
4. Compute $\log_2\left(\frac{1+z}{1-z}\right) = \log_2(m) - \log_2(\varphi_{ijk})$ as follows:
   $$\log_2\left(\frac{1+z}{1-z}\right) \cong w + a_1w^3 + a_2w^5 + \ldots + a_5w^{11}$$
   where coefficients $\{a_n\}$ have been obtained by the minimax technique. $\log_2\left(\frac{1+z}{1-z}\right)$ is approximately bounded by $\pm\frac{1}{300}$. This value is computed with full 28 hexadecimal digit accuracy, and the absolute error is at most 16^{-30}.
5. Now $\log_2(x) = 4p - q + \log_2\alpha_i + \log_2\beta_j + \log_2\gamma_k + \log_2\left(\frac{1+z}{1-z}\right)$.
   $\log_2\alpha_i$, $\log_2\beta_j$, and $\log_2\gamma_k$ are encoded with 31 hexadecimal digits of accuracy. Combine these components in such a way that the maximum absolute error is

still 16^{-30} approximately. This is done to improve accuracy of *A**B* application (see Note below).

6. Truncate $\log_2(x)$ at the 28th hexadecimal digit, and multiply by $\log_e(2)$ or by $\log_{10}(2)$ to obtain $\log_e(x)$ or $\log_{10}(x)$ as desired.

Effect of an Argument Error

$E \sim \delta$. Therefore, if the value of the argument is close to 1, the relative error can be very large, because the value of the function is very small.

LOG/LOG 10

Algorithm

If x is R*4, then LOG (x) = ALOG(x) and LOG10(x) = ALOG10(x).
If x is R*8, then LOG(x) = DLOG(x) and LOG10(x) = DLOG10(x).
If x is R*16, then LOG(x) = QLOG(x) and LOG10(x) = QLOG10(x).

CQLOG

Algorithm

1. Write $\log_e(x + iy) = a + ib$
2. Then, $a = \log_e |x + iy|$ and b = the principal value of $\arctan(y,x)$.
3. $\log_e |x + iy|$ is computed as follows:

   Let $v_1 = \max(|x|, |y|)$, and $v_2 = \min(|x|, |y|)$.

   Let t be the exponent of v_1, i.e., $v_1 = m \cdot 16^t$, $\frac{1}{16} \leqq m < 1$.

   Finally, let $t_1 = \begin{cases} t & \text{if } t \leqq 0 \\ t - 1 & \text{if } t > 0, \end{cases}$
   and $s = 16^{t_1}$.

   Then, $\log_e |x + iy| = 4t_1 \cdot \log_e(2) + \frac{1}{2} \log_e \left[\left(\frac{v_1}{s} \right)^2 + \left(\frac{v_2}{s} \right)^2 \right]$.

   Computation of v_1/s and v_2/s are carried out by manipulation of the characteristics of v_1 and v_2. In particular, if v_2/s is very small, it is given the exponent of -16 to avoid characteristic wrap-around.

Effect of an Argument Error

$E \sim \delta$. Therefore, if the argument is close to 1, the relative error can be very large because the value of the function is very small.

Sine and Cosine Subprograms

SIN/COS

Algorithm

1. Define $z = \frac{4}{\pi} \cdot |x|$ and separate z into its integer part (q) and its fraction part (r). Then $z = q + r$, and $|x| = \left(\frac{\pi}{4} \cdot q\right) + \left(\frac{\pi}{4} \cdot r\right)$.

2. If the cosine is desired, add 2 to q. If the sine is desired and if x is negative, add 4 to q. This adjustment of q reduces the general case to the computation of $\sin(x)$ for $x \geqq 0$ because

$$\cos(\pm x) = \sin\left(\frac{\pi}{2} + x\right), \text{ and}$$
$$\sin(-x) = \sin(\pi + x).$$

3. Let $q_0 \equiv q \bmod 8$.

Then, for $q_0 = 0, \sin(x) = \sin\left(\frac{\pi}{4} \cdot r\right)$,

$$q_0 = 1, \sin(x) = \cos\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 2, \sin(x) = \cos\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 3, \sin(x) = \sin\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 4, \sin(x) = -\sin\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 5, \sin(x) = -\cos\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 6, \sin(x) = -\cos\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 7, \sin(x) = -\sin\left(\frac{\pi}{4}(1 - r)\right).$$

These formulas reduce each case to the computation of either $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ or $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ where r_1 is either r or $(1 - r)$ and is within the range, $0 \leqq r_1 \leqq 1$.

4. If $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ is needed, it is computed by a polynomial of the following form:

$$\sin\left(\frac{\pi}{4} \cdot r_1\right) \cong r_1 (a_0 + a_1 r_1^2 + a_2 r_1^4 + a_3 r_1^6).$$

The coefficients were obtained by interpolation at the roots of the Chebyshev polynomial of degree 4. The relative error is less than $2^{-28.1}$ for the range.

5. If $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ is needed, it is computed by a polynomial of the following form:

$$\cos\left(\frac{\pi}{4} \cdot r_1\right) \cong 1 + b_1 r_1^2 + b_2 r_1^4 + b_3 r_1^6.$$

Coefficients were obtained by a variation of the minimax approximation which provides a partial rounding for the short precision computation. The absolute error of this approximation is less than $2^{-24.57}$.

Effect of an Argument Error

$E \sim \Delta$. As the value of x increases, Δ increases. Because the function value diminishes periodically, no consistent relative error control can be maintained outside the principal range, $-\frac{\pi}{2} \leqq x \leqq +\frac{\pi}{2}$.

DSIN/DCOS

Algorithm

1. Divide $|x|$ by $\frac{\pi}{4}$ and separate the quotient (z) into its integer part (q) and its fraction part (r). Then, $z = |x| \cdot \frac{4}{\pi} = q + r$, where q is an integer and r is within the range, $0 \leqq r < 1$.
2. If the cosine is desired, add 2 to q. If the sine is desired and if x is negative, add 4 to q. This adjustment of q reduces the general case to the computation of $\sin(x)$ for $x \geqq 0$, because

$$\cos(\pm x) = \sin\left(|x| + \frac{\pi}{2}\right), \text{ and}$$
$$\sin(-x) = \sin(|x| + \pi).$$

3. Let $q_0 \equiv q \bmod 8$.

Then, for $q_0 = 0, \sin(x) = \sin\left(\frac{\pi}{4} \cdot r\right)$,

$$q_0 = 1, \sin(x) = \cos\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 2, \sin(x) = \cos\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 3, \sin(x) = \sin\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 4, \sin(x) = -\sin\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 5, \sin(x) = -\cos\left(\frac{\pi}{4}(1 - r)\right),$$
$$q_0 = 6, \sin(x) = -\cos\left(\frac{\pi}{4} \cdot r\right),$$
$$q_0 = 7, \sin(x) = -\sin\left(\frac{\pi}{4}(1 - r)\right).$$

These formulas reduce each case to the computation of either $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ or $\cos\left(\frac{\pi}{4} \cdot r_1\right)$, where r_1 is either r or $(1 - r)$, and is within the range, $0 \leqq r_1 \leqq 1$.

4. Finally, either $\sin\left(\frac{\pi}{4} \cdot r_1\right)$ or $\cos\left(\frac{\pi}{4} \cdot r_1\right)$ is computed, using the polynomial interpolations of degree 6 in r_1^2 for the sine, and of degree 7 in r_1^2 for the cosine. In either case, the interpolation points were the roots of the Chebyshev poly-

nomial of one higher degree. The maximum relative error of the sine polynomial is 2^{-58} and that of the cosine polynomial is $2^{-64.3}$.

Effect of an Argument Error

$E \sim \Delta$. As the value of the argument increases, Δ increases. Because the function value diminishes periodically, no consistent relative error control can be maintained outside of the principal range, $-\frac{\pi}{2} \leqq x \leqq +\frac{\pi}{2}$.

CSIN/CCOS

Algorithm

1. If the sine is desired, then
   $$\sin(x + iy) = \sin(x) \cdot \cosh(y) + i \cdot \cos(x) \cdot \sinh(y).$$
   If the cosine is desired, then
   $$\cos(x + iy) = \cos(x) \cdot \cosh(y) - i \cdot \sin(x) \cdot \sinh(y).$$
2. The value of $\sinh(x)$ is computed within the subprogram as follows. Assume $x \geqq 0$ for this, since $\sinh(-x) = -\sinh(x)$.
3. If $x \geqq 0.346574$, then use $\sinh(x) = \frac{1}{2}\left(e^x - \frac{1}{e^x}\right)$.
4. If $0 \leqq x < 0.346574$, then compute $\sinh(x)$ by use of a polynomial:
   $$\frac{\sinh(x)}{x} \cong a_0 + a_1x^2 + a_2x^4.$$
   The coefficients were obtained by the minimax approximation (in relative error) of $\sinh(x)/x$ over the range $0 \leqq x^2 \leqq 0.12011$ under the constraint that a_0 shall be exactly 1.0. The relative error of this approximation is less than $2^{-26.18}$.
5. The value of $\cosh(x)$ is computed as $\cosh(x) = \sinh|x| + \frac{1}{e^{|x|}}$.
   This computation uses the real exponential subprogram (EXP) and the real sine/cosine subprogram (SIN/COS).

Effect of an Argument Error

To understand the effect of an argument error upon the accuracy of the answer, the programmer must understand the effect of an argument in the SIN/COS, EXP, and SINH/COSH subprograms.

CDSIN/CDCOS

Algorithm

1. If the sine is desired, then
   $$\sin(x+iy) = \sin(x) \cdot \cosh(y) + i \cdot \cos(x) \cdot \sinh(y).$$
   If the cosine is desired, then
   $$\cos(x + iy) = \cos(x) \cdot \cosh(y) - i \cdot \sin(x) \cdot \sinh(y).$$
2. The value of $\sinh(x)$ is computed within the subprogram as follows. Assume $x \geqq 0$ for this, since $\sinh(-x) = -\sinh(x)$.
3. If $x \geqq 0.481212$, then use $\sinh(x) = \frac{1}{2}\left(e^x - \frac{1}{e^x}\right)$.
4. If $0 \leqq x < 0.481212$, then compute $\sinh(x)$ by use of a polynomial:
   $$\frac{\sinh(x)}{x} \cong a_0 + a_1x^2 + a_2x^4 + a_3x^6 + a_4x^8 + a_5x^{10}.$$

The coefficients were obtained by the minimax approximation (in relative error) of $\sinh(x)/x$ over the range $0 \leqq x^2 \leqq 0.23156$ under the constraint that a_0 shall be exactly 1.0. The relative error of this approximation is less than $2^{-56.07}$.

5. The value of $\cosh(x)$ is computed as $\cosh(x) = \sinh|x| + \frac{1}{e^{|x|}}$.

   This computation uses the real exponential subprogram (DEXP) and the real sine/cosine subprogram (DSIN/DCOS).

Effect of an Argument Error

To understand the effect of an argument error upon the accuracy of the answer, the programmer must understand the effect of an argument error in the DSIN/DCOS, DEXP, and DSINH/DCOSH subprograms.

QSIN/QCOS

Algorithm

1. Separate the argument into an integral multiple of $\frac{\pi}{2}$ and the remainder part:

   $$|x| = \frac{\pi}{2} \cdot q + r \text{ where } q \text{ is an integer, and } -\frac{\pi}{4} \leqq r < \frac{\pi}{4}.$$

   In this decomposition, after q is estimated in the working precision, r is accurately computed as $r = |x| - \frac{\pi}{2} \cdot q$ with the aid of approximately 10 hexadecimal guard digits.

2. Add 1 to q if cosine is desired, since $\cos(\pm x) = \sin\left(\frac{\pi}{2} + x\right)$.

   Add 2 to q if sine is desired and x is negative, since $\sin(-x) = \sin(\pi + x)$. These adjustments reduce the general case to computation of $\sin(x)$ for $x \geqq 0$.

3. Let $q_0 \equiv q \bmod 4$. Then,

   if $q_0 = 0, \sin(|x|) = \sin(r)$
   $q_0 = 1, \sin(|x|) = \cos(r)$
   $q_0 = 2, \sin(|x|) = -\sin(r)$
   $q_0 = 3, \sin(|x|) = -\cos(r)$

4. Compute $\sin(r)$ or $\cos(r)$ as follows:

   $$\sin(r) \cong r + a_1 r^3 + a_2 r^5 + \ldots + a_{11} r^{23}$$
   $$\cos(r) \cong 1 + b_1 r^2 + b_2 r^4 + \ldots + b_{12} r^{24}$$

   Coefficients $\{a_j\}$, $\{b_j\}$ are determined by the minimax technique as applied to the range $0 \leqq r \leqq \frac{\pi}{4}$. The relative errors of these approximations are less than 2^{-112}.

Effect of an Argument Error

$E \sim \Delta$. As the value of x increases, Δ increases. Because the function value diminishes periodically, no consistent relative error control can be normally maintained outside the principal range $-\frac{\pi}{2} \leqq x \leqq +\frac{\pi}{2}$.

CQSIN/CQCOS

Algorithm

1. If the sine is desired, then

   $$\sin(x + iy) = \sin(x) \cdot \cosh(y) + i \cdot \cos(x) \cdot \sinh(y).$$

   If the cosine is desired, then

   $$\cos(x + iy) = \cos(x) \cdot \cosh(y) - i \cdot \sin(x) \cdot \sinh(y).$$

2. The value of $\sinh(x)$ is computed within the subprogram as follows. Assume $x \geqq 0$ for this, since $\sinh(-x) = -\sinh(x)$.
3. If $x \geqq 0.481212$, then use $\sinh(x) = \frac{1}{2}\left(e^x - \frac{1}{e^x}\right)$.
4. If $0 \leqq x < 0.481212$, then compute $\sinh(x)$ by the use of the polynomial:
$$\frac{\sinh(x)}{x} \cong a_0 + a_1x^2 + a_2x^4 + \ldots + a_{10}x^{20}$$
The coefficients were obtained by the minimax approximation (in relative error) of $\sinh(x)/x$ over the range $0 \leqq x^2 \leqq 0.23156$ under the constraint that a_0 shall be exactly 1.0. The relative error of this approximation is less than 2^{-112}. The highest three terms of this polynomial need only be evaluated in double precision.
5. The value of $\cosh(x)$ is computed as $\cosh(x) = \sinh|x| + \frac{1}{e^{|x|}}$.

Effect of an Argument Error

Combine such effects on sine/cosine/hyperbolic-sine/hyperbolic-cosine functions according to the formula in step 1 of the algorithm.

Square Root Subprograms

SQRT

Algorithm

1. If $x = 0$, then the answer is 0.
2. Write $x = 16^{2p-q} \cdot m$, where $2p - q$ is the exponent and q equals either 0 or 1; m is the mantissa and is within the range $\frac{1}{16} \leqq m < 1$.
3. Then, $\sqrt{x} = 16^p \cdot 4^{-q}\sqrt{m}$.
4. For the first approximation of $\sqrt{x}$, compute the following:
$$y_0 = 16^p \cdot 4^{-q} \cdot \left(1.681595 - \frac{1.288973}{0.8408065 + m}\right).$$
This approximation attains the minimax relative error for hyperbolic fits of $\sqrt{x}$. The maximum relative error is $2^{-5.748}$.
5. Apply the Newton-Raphson iteration
$$y_{n+1} = \tfrac{1}{2}\left(y_n + \frac{x}{y_n}\right)$$
twice. The second iteration is performed as
$$y_2 = \tfrac{1}{2}\left(y_1 - \frac{x}{y_1}\right) + \frac{x}{y_1},$$
with a partial rounding. The maximum relative error of y_2 is theoretically $2^{-25.9}$.

Effect of an Argument Error

$$\epsilon \sim \frac{1}{2}\delta.$$

DSQRT

Algorithm

1. If $x = 0$, then the answer is 0.
2. Write $x = 16^{2p-q} \cdot m$, where $2p - q$ is the exponent and q equals either 0 or 1; m is the mantissa and is within the range $\frac{1}{16} \leqq m < 1$.

3. Then, $\sqrt{x} = 16^p \cdot 4^{-q} \sqrt{m}$.
4. For the first approximation of $\sqrt{x}$, compute the following:

$$y_0 = 16^p \cdot 4^{1-q} \cdot 0.2202\,(m + 0.2587).$$

The extrema of relative errors of this approximation for $q = 0$ are $2^{-3.202}$ at $m = 1$, $2^{-3.265}$ at $m = 0.2587$, and $2^{-2.925}$ at $m = \frac{1}{16}$. This approximation, rather than the minimax approximation, was chosen so that the quantity $\frac{x}{y_3} - y_3$ below becomes less than 16^{p-8} in magnitude. This arrangement allows us to substitute short form counterparts for some of the long form instructions in the final iteration.

5. Apply the Newton Raphson iteration

$$y_{n+1} = \tfrac{1}{2}\left(y_n + \frac{x}{y_n}\right)$$

four times to y_0, twice in the short form and twice in the long form. The final step is performed as

$$y_4 = y_3 + \tfrac{1}{2}\left(\frac{x}{y_3} - y_3\right)$$

with an appropriate truncation maneuver to obtain a virtual rounding. The maximum relative error of the final result is theoretically $2^{-63.23}$.

Effect of an Argument Error

$$\epsilon \sim \frac{1}{2}\delta$$

CSQRT/CDSQRT

Algorithm

1. Write $\sqrt{x + iy} = a + ib$.
2. Compute the value $z = \sqrt{\frac{|x| + |x + iy|}{2}}$ as $k \cdot \sqrt{w_1 + w_2}$ where k, w_1 and w_2 are defined in 3 or 4, below. In any case let $v_1 = \max(|x|, |y|)$ and $v_2 = \min(|x|, |y|)$.
3. In the special case when either $v_2 = 0$ or v_1 greatly exceeds v_2, let $\omega_1 = v_2$ and $\omega_2 = v_1$ so that $\omega_1 + \omega_2$ is effectively equal to v_1.
   Also let $k = 1$ if $v_1 = |x|$ and
   $k = 1/\sqrt{2}$ if $v_1 = |y|$.
4. In the general case, compute $F = \sqrt{\tfrac{1}{4} + \tfrac{1}{4}\left(\frac{v_2}{v_1}\right)^2}$.
   If $|x|$ is near the underflow threshold, then take
   $w_1 = |x|$, $w_2 = v_1 \cdot 2F$, and $k = 1/\sqrt{2}$.
   If $v_1 \cdot F$ is near the overflow threshold, then take
   $w_1 = |x|/4$, $w_2 = v_1 \cdot F/2$, and $k = \sqrt{2}$.
   In all other cases, take $w_1 = |x|/2$, $w_2 = v_1 \cdot F$, and $k = 1$.
5. If $z = 0$, then $a = 0$ and $b = 0$.
   If $z \neq 0$ and $x \geqq 0$, then $a = z$, and

$$b = \frac{y}{2z}.$$

   If $z \neq 0$ and $x < 0$, then $a = \left|\frac{y}{2z}\right|$, and

$$b = (\operatorname{sign} y) \cdot z.$$

The algorithms for both complex square root subprograms are identical. Each subprogram uses the appropriate real square root subprogram (SQRT or DSQRT).

Effect of an Argument Error

The effect of an argument error depends upon the accuracy of the individual parts of the argument. If $x + iy = r \cdot e^{ih}$ and $\sqrt{x + iy} = R \cdot e^{iH}$,

then $\epsilon(R) \sim \frac{1}{2}\,\delta(r)$, and $\epsilon(H) \sim \delta(h)$.

QSQRT

Algorithm

1. Let $x = 16^{2p+q} \cdot m$, where p is an integer, $q = 0$ or 1, and
   $\frac{1}{16} \leqq m < 1$. Let $x_1 = 16^{32-q} \cdot m$
   This scaling by 16^{32} is made to avoid intermediate underflows.
2. Compute the first approximation y_0 to $\sqrt{x_1}$ as follows:
   $$y_0 = 16^{16} \cdot 4^{-q} \left\{ 1.807018 - \frac{1.576942}{0.9540356 + m} \right\}$$
   These coefficients were determined to minimize the relative error of the approximation while being exact at $m = 1$. The maximum relative error is $2^{-5.48}$.
3. Apply Newton Raphson iteration three times – twice in short form and once in long form.
   $$y_i = \frac{1}{2}\left(y_{i-1} + \frac{x_1}{y_{i-1}} \right) \qquad i = 1, 2, 3.$$
   At the end of the third iteration, the relative error ϵ_3 of y_3 is at most 2^{-41}.
4. Apply to y_3 the following cubic refinement in extended precision:
   $$y_4 = y_3 - 2y_3 \cdot \frac{y_3^2 - x_1}{3y_3^2 + x_1}.$$
   The relative error ϵ_4 of y_4 is $\frac{1}{4}(\epsilon_3)^3$ or 2^{-125}.
   Since the right hand term is only a correctional term, a simplified extended division suffices. In the process of assembling y_4, a virtual rounding is given.
5. Replace the exponent of y_4 with the correct exponent $p + q$.

Effect of an Argument Error

$$\epsilon \sim \frac{1}{2}\,\delta$$

CQSQRT

Algorithm

1. Write $\sqrt{x + iy} = a + ib$
2. Let $16^{2p+q-1} \leqq \max(|x|, |y|) < 16^{2p+q}$, $q = 0$, or 1
   Let $x_1 = x \cdot 16^{-2p}$, and $y_1 = y \cdot 16^{-2p}$.
   This scaling operation is carried out by manipulation of the characteristic fields of x and y. In doing this necessary precaution is exercised to avoid the anomaly of characteristic wrap-around.
3. Compute $z_1 = \sqrt{\dfrac{|x_1| + |x_1 + iy_1|}{2}}$
   Restore scaling: $z = 16^p \cdot z_1$
4. If $z = 0$, then $a = 0$ and $b = 0$.
   If $z \neq 0$, and $x \geqq 0$, then $a = z$, and

$$b = \frac{y}{2z}.$$

If $z \neq 0$ and $x < 0$, then $a = \left|\frac{y}{2z}\right|$, and

$$b = (\text{sign } y) \cdot z.$$

Effect of an Argument Error

Using polar coordinate, write $x + iy = r \cdot e^{ih}$ and $\sqrt{x + iy} = R \cdot e^{iH}$.

Then $\epsilon(R) \sim \frac{1}{2}\delta(r)$, and $\epsilon(H) \sim \delta(h)$.

Tangent and Cotangent Subprograms

TAN/COTAN

Algorithm

1. Divide $|x|$ by $\frac{\pi}{4}$ and separate the result into the integer part (q) and the fraction part (r). Then $|x| = \frac{\pi}{4}(q + r)$.

2. Obtain the reduced argument (w) as follows:

   if q is even, then $w = r$

   if q is odd, then $w = 1 - r$.

   The range of the reduced argument is $0 \leqq w \leqq 1$.

3. Let $q_0 \equiv q \bmod 4$.

   Then for $q_0 = 0$, $\tan |x| = \tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot |x| = \cot\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 1$, $\tan |x| = \cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot |x| = \tan\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 2$, $\tan |x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot |x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 3$, $\tan |x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot |x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$.

4. The value of $\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot\left(\frac{\pi}{4} \cdot w\right)$ are computed as the ratio of two polynomials:

   $$\tan\left(\frac{\pi}{4} \cdot w\right) \cong \frac{w \cdot P(u)}{Q(u)}, \quad \cot\left(\frac{\pi}{4} \cdot w\right) \cong \frac{Q(u)}{w \cdot P(u)}$$

   where $u = \frac{1}{2}w^2$ and

   $P(u) = -8.460901 + u$

   $Q(u) = -10.772754 + 5.703366 \cdot u - 0.159321 \cdot u^2$.

   These coefficients were obtained by the minimax rational approximation (in relative error) of the indicated form. The maximum relative error of this approximation is 2^{-26}. Choice of u rather than w^2 as the variable for P and Q is to improve the roundoff quality of the coefficients.

5. If $x < 0$, then $\tan(x) = -\tan |x|$, and $\cot(x) = -\cot |x|$.

6. This program is provided with two kinds of error controls. One is for arguments whose magnitude is greater than $2^{18} \cdot \pi$. The other is for arguments which are very close to a singularity of the function. In either case, the precision of the argument is deemed insufficient for obtaining a reliable result. More specifically, the second control screens out the following arguments:

   a) $|x| \leqq 16^{-63}$ for COTAN (the result would overflow).

   b) x is such that one can find a singularity within eight units of the last digit

value of the floating-point representation of the sum $q + r$. Singularities are cases when the cotangent ratio is to be taken and $w = 0$.

Effect of an Argument Error

$E \sim \frac{\Delta}{\cos^2(x)}$, and $\epsilon \sim \frac{2}{\sin(2x)}$ for $\tan(x)$. Therefore, near the singularities $x = \left(k + \frac{1}{2}\right)\pi$, where k is an integer, no error control can be maintained. This is also true for $\text{cotan}(x)$ for x near $k\pi$, where k is an integer.

DTAN/DCOTAN

Algorithm

1. Divide $|x|$ by $\frac{\pi}{4}$ and separate the result into integer part (q) and the fraction part (r). Then $|x| = \frac{\pi}{4}(q + r)$.
2. Obtain the reduced argument (w) as follows:
   if q is even, then $w = r$
   if q is odd, then $w = 1 - r$.
   The range of the reduced argument is $0 \leqq w \leqq 1$.
3. Let $q_0 \equiv q \bmod 4$.

   Then for $q_0 = 0$, $\tan|x| = \tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = \cot\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 1$, $\tan|x| = \cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = \tan\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 2$, $\tan|x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$,

   $q_0 = 3$, $\tan|x| = -\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot|x| = -\cot\left(\frac{\pi}{4} \cdot w\right)$.

4. The value of $\tan\left(\frac{\pi}{4} \cdot w\right)$ and $\cot\left(\frac{\pi}{4} \cdot w\right)$ are computed as the ratio of two polynomials:

   $$\tan\left(\frac{\pi}{4} \cdot w\right) \cong \frac{w \cdot P(w^2)}{Q(w^2)}, \text{ and } \cot\left(\frac{\pi}{4} \cdot w\right) \cong \frac{Q(w^2)}{w \cdot P(w^2)}.$$

   where both P and Q are polynomials of degree 3 in w^2. The coefficients of P and Q were obtained by the minimax rational approximation (in relative error) of $\frac{1}{w}\tan\left(\frac{\pi}{4}w\right)$ of the indicated form. The maximum relative error of this approximation is $2^{-55.6}$.
5. If $x < 0$, then $\tan(x) = -\tan|x|$, and $\cot(x) = -\cot|x|$.
6. This program is provided with two kinds of error controls. One is for arguments whose magnitude is greater than $2^{50} \cdot \pi$. The other is for arguments which are very close to a singularity of the function. In either case, the precision of the argument is deemed insufficient for obtaining a reliable result. More specifically, the second control screens out the following arguments:
   a) $|x| \leqq 16^{-63}$ for COTAN (the result would overflow).
   b) x is such that one can find a singularity within eight units of the last digit value of the floating-point representation of the sum $q + r$. Singularities are cases when the cotangent ratio is to be taken and $w = 0$.

Effect of an Argument Error

$E \sim \frac{\Delta}{\cos^2(x)}$, and $\epsilon \sim \frac{2}{\sin(2x)}$ for $\tan(x)$. Therefore, near the singularities of $x = \left(k + \frac{1}{2}\right)\pi$, where k is an integer, no error control can be maintained. This is also true for $\text{cotan}(x)$ for values of x near $k\pi$, where k is an integer.

QTAN/QCOTAN

Algorithm

1. Separate argument into an integral multiple of $\frac{\pi}{2}$ and the remainder part:

   $$|x| = \frac{\pi}{2} \cdot q + r \quad \text{where } q \text{ is an integer, and } -\frac{\pi}{4} \leqq r < \frac{\pi}{4}.$$

   In this decomposition, after q is estimated in the working precision, r is accurately computed as $r = |x| - \frac{\pi}{2} \cdot q$ with the aid of approximately 10 hexadecimal guard digits.
2. If $\cot(x)$ is desired, add 1 to q, and remember to change the sign of the answer. Since $\cot(x) = -\tan\left(x + \frac{\pi}{2}\right)$, this reduces the case to computation of tangent.
3. If q is even, $\tan(|x|) = \tan(r)$, and the latter is obtained by a minimax approximation of the form:

   $$\tan(r) \cong \frac{rP(r^2)}{Q(r^2)}$$

   where P and Q are polynomials of degree 6 and 5 respectively.

   If q is odd, $\tan(|x|) = -\cot(r)$, and the latter is computed as

   $$\cot(r) \cong \frac{Q(r^2)}{rP(r^2)}$$

   using the same polynomials as the former case.

   The relative errors of these approximations are less than 2^{-111}. In evaluating these rational approximations, an exponent scaling is used to avoid intermediate partial underflows, which can result in a loss of accuracy.
4. If $x < 0$, then $\tan(x) = -\tan(|x|)$, and $\cot(x) = -\cot(|x|)$.

Effect of an Argument Error

$E \sim \frac{\Delta}{\cos^2(x)}$, and $\epsilon \sim \frac{2\Delta}{\sin(2x)}$ for $\tan(x)$. Therefore near the singularities $x = \left(k + \frac{1}{2}\right)\pi$, where k stands for integers, no error control can be maintained. This is also true for $\cot(x)$ for x near $k\pi$, where k is an integer.

Implicitly Called Subprograms

The entry point names of the following implicitly called subprograms are generated by the compiler.

Complex Multiply and Divide Subprograms

CDVD#/CMPY# (Divide/Multiply for COMPLEX*8 Arguments)

CDDVD#/CDMPY# (Divide/Multiply for COMPLEX*16 Arguments)

Algorithm

Multiply: $(A + Bi)\,(C + Di) = (AC - BD) + (AD + BC)i$

Divide: $(A + Bi)/(C + Di)$

1. If $|C| \leqq |D|$, set
   $A = B, B = -A, C = D, D = -C$, since
   $$\frac{A + Bi}{C + Di} = \frac{B - Ai}{D - Ci}$$ before step 2.
2. Set $A' = \frac{A}{C}, B' = \frac{B}{C}, D' = \frac{D}{C}$;
   then compute
   $$\frac{A + Bi}{C + Di} = \frac{A' + B'i}{1 + D'i} = \frac{A' + B'D}{1 + D'D'} + \frac{B' - A'D'}{1 + D'D'}\,i.$$

Error Conditions

Partial underflows can occur in preparing the answer.

CQMPY#/CQDVD# (Multiply/Divide for COMPLEX*32 Arguments)

Algorithm

Multiply: $(a+bi)\,(c+di) = (ac-bd) + (ad+bc)\,i$

Divide: $(a+bi)/(c+di)$

1. Let $a + bi$ and $c + di$ be the first and the second operands respectively.
2. Find exponents p_1, p_2 which satisfy the following:
   $$16^{p_1-1} \leqq \max(|a|, |b|) < 16^{p_1},\ 16^{p_2-1} \leqq \max(|c|, |d|) < 16^{p_2}.$$
   Choose $q = -3$ if $p_1 \geqq 0$
   $q = 31$ if $p_1 < 0$
3. Scale c and d by 16^{p_2-q} and change sign of d if CQDVD#:
   $$c_1 = c \cdot 16^{q-p_2}$$
   $$d_1 = \begin{cases} d \cdot 16^{q-p_2} & \text{if CQMPY\#} \\ -d \cdot 16^{q-p_2} & \text{if CQDVD\#.} \end{cases}$$
   Here if the exponent adjustment results in underflow, replace the affected quantity with 0.
4. Compute $u_1 + v_1 i = (ac_1 - bd_1) + (ad_1 + bc_1)\,i$
5. If CQMPY#, restore the scaling to obtain the answer $u + vi$:
   $u = u_1 \cdot 16^{p_2-q}$ and $v = v_1 \cdot 16^{p_2-q}$.
6. If CQDVD#, compute the denominator as follows:
   $$w_1 = (c_1^2 + d_1^2) \cdot 16^{-2q}$$
   Note that $16^{-2} \leqq w_1 < 2$.
   Then divide: $u_2 = u_1/w_1$ and $v_2 = v_1/w_1$
   Finally, restore the scaling to obtain the answer $u + vi$:
   $u = u_2 \cdot 16^{-q-p_2}$ and $v = v_2 \cdot 16^{-q-p_2}$.

Effect of an Argument Error

In terms of complex vector relative errors, $\epsilon \sim \delta x + \delta y$ where δx is the relative error of the first operand and δy is the relative error of the second operand.

Complex Exponentiation Subprograms

(Exponentiation of a Complex Base to an Integer Power)

FCDXI# (COMPLEX*16 Arguments)

FCXPI# (COMPLEX*8 Arguments)

Algorithm

The value of $y_1 + y_2 i = (z_1 + z_2 i)^j$ is computed as follows.

Let $|j| = \sum_{k=0}^{K} r_k \cdot 2^k$ where $r_k = 0$ or 1 for $k = 0, 1, \ldots, K$.

Then $z^{|j|} = \Pi\, z^{2^k}$, and the factors z^{2^k}, can be obtained by successive squaring.

More specifically:

1. Initially: $k = 0$, $n^{(0)} = |j|$, $y_1^{(0)} + y_2^{(0)} i = 1 + 0i$, $z_1^{(0)} + z_2^{(0)} i = z_1 + z_2 i$.
2. Raise the index k by 1, and let $n^{(k-1)} = 2q + r$, where q is the integer quotient and $r = 0$ or 1.
3. Let $n^{(k)} = q$.
4. If $r = 0$, then $y_1^{(k)} + y_2^{(k)} i = y_1^{(k-1)} + y_2^{(k-1)} i$.
   If $r = 1$, then $y_1^{(k)} + y_2^{(k)} i = (y_1^{(k-1)} + y_2^{(k-1)} i)\ (z_1^{(k-1)} + z_2^{(k-1)} i)$.
5. If $n^{(k)} \neq 0$, then $z_1^{(k)} + z_2^{(k)} i = (z_1^{(k-1)} + z_2^{(k-1)})^2$, and steps 2 through 5 are repeated until $n^{(k)} = 0$.
6. When $n^{(k)} = 0$, and $j \geqq 0$, then $y_1 + y_2 i = y_1^{(k)} + y_2^{(k)} i$.
   If $j < 0$, then $y_1 + y_2 i = (1 + 0i) \,/\, (y_1^{(k)} + y_2^{(k)} i)$.

(Exponentiation of a Complex Base to a Complex Power)

FCQXQ# (COMPLEX*32 Arguments)
FCDXD# (COMPLEX*16 Arguments)
FCXPC# (COMPLEX*8 Arguments)

Algorithm

z_1 ** z_2 = exp (z_2 * log z_1), where the functions 'exp' and 'log' are CEXP and CLOG, CDEXP and CDLOG, or CQEXP and CQLOG respectively as the arguments are C*8, C*16, or C*32.

Effect of an Argument Error

If $z_1 = x_1 + iy$, and $z_2 = x_2 + iy_2$, then
z_1 ** z_2 = exp (a) * (cos(b) + i sin(b)), where
$a = x_2 * \log|\, x_1 + iy_1 \,| - y_2 *$ arctan (y_1/x_1) and
$b = y_2 * \log|\, x_2 + iy_2 \,| + x_2 *$ arctan (y_1/x_1).

The function z z_2 is calculated using the FORTRAN routines for sin, cos, exp, log, and arctan.**

Therefore the effect of an argument error upon the accuracy of the result depends upon its effect in the functions SIN, COS, EXP, LOG, and ATAN.

Exponentiation of a Real Base to a Real Power Subprograms

FDXPD# (REAL*8 Arguments)

FRXPR# (REAL*4 Arguments)

Algorithm

Assume the desired answer is a^b.

1. If $a = 0$ and $b \leqq 0$, error return.
   If $a = 0$ and $b > 0$, the answer is 0.
2. If $a \neq 0$ and $b = 0$, the answer is 1.
3. All other cases, compute a^b as $e^{b \cdot \log a}$. In this computation the exponential subroutine and the natural logarithm subroutine are used. If a is negative or if $b \cdot \log a$ is too large, an error return is given by one of these subroutines.

Error estimate

The relative error of the answer can be expressed as $(\epsilon_1 + \epsilon_2)\, b \cdot \log(a) + \epsilon_3$ where ϵ_1, ϵ_2, and ϵ_3 are relative errors of the logarithmic routine, machine multiplication, and the exponential routine, respectively.

For FDXPD#, $\epsilon_1 \leqq 3.5x10^{-16}$, $\epsilon_2 \leqq 2.2x10^{-16}$, and $\epsilon_3 \leqq 2.0x10^{-16}$. Hence the relative error $\leqq 5.7x10^{-16}x \mid b \cdot \log a \mid + 2.0x10^{-16}$. Note that $b \cdot \log a$ is the natural logarithm of the answer.

For FRXPR#, $\epsilon_1 \leqq 8.3x10^{-7}$, $\epsilon_2 \leqq 9.5x10^{-7}$, and $\epsilon_3 \leqq 4.7x10^{-7}$. Hence the relative error $\leqq 1.8x10^{-6}\, x \mid b \cdot \log a \mid + 4.7x10^{-7}$.

Effect of an Argument Error

$[a(1 + \delta_1)]\, b(1 + \delta_2) \cong a^b(1 + \delta_2 b \cdot \log a + b\delta_1)$. Note that if the answer does not overflow, $|b \cdot \log a| < 175$. On the other hand b can be very large without causing an overflow of a^b if $\log a$ is very small. Thus, if $a \cong 1$ and if b is very large, then the effect of the perturbation δ_1 of a shows very heavily in the relative error of the answer.

Exponentiation of a Real Base to an Integer Power Subprograms

FDXPI# (REAL*8 Arguments)

FRXPI# (REAL*4 Arguments)

Algorithm

1. If $a = 0$ and $b \leqq 0$, error return.
   If $a = 0$ and $b > 0$, the answer is 0.
2. If $a \neq 0$ and $b = 0$, the answer is 1.
3. The value of $y = a^j$ is computed as follows: Let $|j| = \sum_{k=0}^{K} r_k 2^k$ where $r_k = 0$ or 1 for $k = 0, 1, \ldots, K$. Then $a^{|j|} = \prod_{r_k \neq 0} a^{2^k}$ and the factors a^{2^k} can be obtained by successive squaring.

   More specifically:
   1. Initially: $k = 0$, $n^{(0)} = |j|$, $y^{(0)} = 1$, and $z^{(0)} = a$.
   2. Raise the index k by 1, and decompose $n^{(k-1)} = 2q + r$, where q is the integer quotient and $r = 0$ or 1.
   3. Let $n^{(k)} = q$.
   4. If $r = 0$, then $y^{(k)} = y^{(k-1)}$.
      If $r = 1$, then $y^{(k)} = y^{(k-1)} z^{(k-1)}$.

5. If $n^{(k)} \neq 0$, then $z^{(k)} = z^{(k-1)}z^{(k1-)}$, and steps 2 through 5 are repeated until $n^{(k)} = 0$.
6. When $n^{(k)} = 0$, and $j \geqq 0$, then $y = y^{(k)}$. If $j < 0$, then $y = \frac{1}{y^{(k)}}$.

Note: The negative exponent is computed by taking the reciprocal of the positive power. Thus it is not possible to compute 16.0**—64 because there is a lack of symmetry for real floating-point numbers — i.e., 16.0**—64 can be represented, but 16.0**64 cannot. The result is obtained by successive multiplications and is exact only if the answer contains at most 14 significant hexadecimal digits.

Exponentiation of an Integer Base to an Integer Power Subprogram

FIXPI# (INTEGER*4 Arguments)

Algorithm

1. If $a = 0$ and $b \leqq 0$, error return.
   If $a = 0$ and $b > 0$, the answer is 0.
2. If $a \neq 0$ and $b = 0$, the answer is 1.
3. The value of $L = I^j$ is computed as follows: Let $j = \sum_{k=0}^{K_1} r_k \cdot 2^k$ where $r_k = 0$ or 1 for $k = 0, 1, \ldots, K$. Then $I^j = \prod_{r_k \neq 0} I^{2^k}$, and the factors I^{2^k} can be obtained by successive squaring.

   More specifically:
   1. Initially: $k = 0$, $n^{(0)} = j$, $y^{(0)} = 1$, and $m^{(0)} = I$.
   2. Raise the index k by 1, and decompose $n^{(k-1)} = 2q + r$, where q is the integer quotient and $r = 0$ or 1.
   3. Let $n^{(k)} = q$.
   4. If $r = 0$, then $y^{(k)} = y^{(k-1)}$.
      If $r = 1$, then $y^{(k)} = y^{(k-1)} \cdot m^{(k-1)}$.
   5. If $n^{(k)} \neq 0$, then $m^{(k)} = m^{(k-1)} \cdot m^{(k-1)}$, and steps 2 through 5 are repeated until $n^{(k)} = 0$.
   6. When $n^{(k)} = 0$, $L = L^{(k)}$.

Note: The result is obtained by successive multiplications. The result is exact only if it is less than (2**31) — 1. Results are meaningless when this limit is exceeded and may even be of changed sign. No tests for overflow are made.

Complex Exponentiation Subprogram

FCQXI# (COMPLEX*32 Arguments)

Algorithm

1. Write $(x + yi)^J = a + bi$.
2. If $x + yi = 0 + 0i$ and $J > 0$, then $a + bi = 0 + 0i$
3. If $J = 0$, $a + bi = 1.0 + 0i$. Assume now $J \neq 0$.
4. Let $|J| = \sum_{j=0}^{n} g_j 2^{n-j}$ where $g_j = 0$ or 1, $g_0 = 1$.
   Initialize $a_0 + b_0 i = x + yi$. If $|J| = 1$, skip the following.
   Do the following for $j = 1, 2, \ldots, n$:
   $$a_j + b_j i = \begin{cases} (a_{j-1} + b_{j-1}i)^2 & \text{if } g_j = 0 \\ (a_{j-1} + b_{j-1}i)^2(x + yi) & \text{if } g_j = 1 \end{cases}$$
   At the end of iteration $a_n + b_n i = (x + yi)^{|J|}$.
5. If $J < 0$, $(x + yi)^J = \dfrac{1.0 + 0i}{(x + yi)^{|J|}}$

Effect of an Argument Error

$|\epsilon| \sim J\,|\delta|$ where δ is the complex relative error of the base and ϵ is the complex relative error of the result due to this.

Exponentiation of a Base 2 Argument to a Real Power Subprogram

FQXP2# (REAL*16 Arguments)

Algorithm

This subprogram uses the same algorithm as the QEXP explicit subprogram.

Exponentiation of a Real Base to an Integer Power Subprogram

FQXPI# (REAL*16 Arguments)

Algorithm

1. Write $x^J = y$
2. If $x = 0$ and $J > 0$, then $y = 0$
3. If $x \neq 0$, and $J = 0$, then $y = 1.0$. Assume now $J \neq 0$.
4. Let $|J| = \sum_{j=0}^{n} g_j 2^{n-j}$ where $g_j = 0$ or 1, $g_0 = 1$.
   Initialize $y_0 = x$. If $|J| = 1$, skip the following.
   Do the following for $j = 1, 2, \ldots, n$:
   $$y_j = \begin{cases} y^2_{j-1} & \text{if } g_j = 0 \\ y^2_{j-1} \cdot x & \text{if } g_j = 1 \end{cases}$$
   At the end of iteration $y_n = x^{|J|}$.
5. If $J < 0$, $x^J = \dfrac{1}{x^{|J|}}$

Note: The negative power is computed by taking the reciprocal of the positive power. Thus it is not possible to compute 16.0**−64 because there is a lack of symmetry in real floating point numbers; i.e., 16.0**−64 can be represented, but 16.0**64 cannot.

Effect of an Argument Error

$\epsilon \sim J\,\delta$

Exponentiation of a Real Base to a Real Power Subprogram

FQXPQ# (REAL*16 Arguments)

Algorithm

1. Basically, $x^y = 2^{y \cdot \log_2(x)}$.
2. More specifically, $\log_2(x)$ is computed with aimed accuracy of 16^{-30} in absolute error, or 16^{-28} in relative error, whichever is smaller, by the algorithm of QLOG/QLOG10. The result is kept as two components; the high order part is represented by a single precision number; and the low order part, which is less than 16^{-2} in absolute value, is represented by an extended precision number.
3. The product $y \cdot \log_2(x)$ is obtained by a simulated multiplication to obtain up to 31 hexadecimal digits of accuracy.
4. Raise the result to the power of 2 by the algorithm of QEXP. As stated there, this includes a virtual final rounding with the result that one obtains the full 28 hexadecimal digit accuracy unless x is very close to 1.0.

Effect of an Argument Error

$\epsilon \sim y \cdot \delta_x + y \cdot \log(x) \cdot \delta_y$. The factor $y \cdot \log(x)$ is limited by 180 in magnitude. If beyond this, the result will overflow. However, the other factor y can be very large if x is close to 1. This is so because $\log(x)$ will then be very small.

Accuracy Statistics

This chapter contains accuracy statistics for explicitly and implicitly called mathematical subprograms. These statistics are presented in Figure 12. They are arranged in alphabetic order, according to the entry names. The following information is given in the two figures:

Entry Name: This column gives you the entry name that is used to call the subprogram.

Argument Range: This column gives the argument range used to obtain the accuracy figures. For each function, accuracy figures are given for one or more representative segments within the valid range. In each case, the figures given are the most meaningful to the function and range under consideration.

The maximum relative error and standard deviation of the relative error are generally useful and revealing statistics; however, they are useless for the range of a function where its value becomes 0. This is because the slightest error in the argument can cause an unpredictable fluctuation in the magnitude of the answer. When a small argument error would have this effect, the maximum absolute error and standard deviation of the absolute error are given for the range.

Sample: This column indicates the type of sample used for the accuracy figures. The type of sample depends on the function and range under consideration. The statistics may be based either upon an exponentially distributed (E) argument sample or a uniformally distributed (U) argument sample.

Accuracy Figures: This column gives accuracy figures for one or more representative segments within the valid argument range. The accuracy figures supplied are based on the assumption that the arguments are perfect, that is, without error and, therefore, have no error propagation effect upon the answers. The only errors in the answer are those introduced by the subprograms. The chapter "Algorithms" contains a description of some of the symbols used in the chapter; the following additional symbols are used in the presentation of accuracy figures:

$M(\epsilon) = \text{Max}\left\lvert\frac{f(x) - g(x)}{f(x)}\right\rvert$	The maximum relative error produced during testing.
$\sigma(\epsilon) = \sqrt{\frac{1}{N}\Sigma_i \left\lvert\frac{f(x_i) - g(x_i)}{f(x_i)}\right\rvert^2}$	The standard deviation (root-mean-square) of the relative error.
$M(E) = \text{Max}\lvert f(x) - g(x)\rvert$	The maximum absolute error produced during testing.
$\sigma(E) = \sqrt{\frac{1}{N}\Sigma_i \lvert f(x_i) - g(x_i)\rvert^2}$	The standard deviation (root-mean-square) of the absolute error.

In case of complex functions, the absolute value signs employed in the above definitions are to mean the complex absolute values. In the formulas for standard deviation, N represents the total number of arguments in the sample; i is a subscript that varies from 1 to N.

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (ϵ)	σ (ϵ)	M (E)	σ (E)
ALGAMA	$0 < X < 0.5$	U	1.16×10^{-6}	3.54×10^{-7}		
	$0.5 \leqq X < 3.0$	U			9.43×10^{-7}	3.42×10^{-7}
	$3.0 \leqq X < 8.0$	U	1.25×10^{-6}	3.04×10^{-7}		
	$8.0 \leqq X < 16.0$	U	1.18×10^{-6}	3.80×10^{-7}		
	$16.0 \leqq X < 500.0$	U	9.85×10^{-7}	1.90×10^{-7}		
ALOG	$0.5 \leqq X \leqq 1.5$	U			6.85×10^{-8}	2.33×10^{-8}
	$X < 0.5, X > 1.5$	E	8.32×10^{-7}	1.19×10^{-7}		
ALOG 10	$0.5 \leqq X \leqq 1.5$	U			7.13×10^{-8}	2.26×10^{-8}
	$X < 0.5, X > 1.5$	E	1.05×10^{-6}	2.17×10^{-7}		
ARCOS	$-1 \leqq X \leqq +1$	U	8.85×10^{-7}	3.19×10^{-7}		
ARSIN	$-1 \leqq X \leqq +1$	U	9.34×10^{-7}	2.06×10^{-7}		
ATAN	The full range	Note 7	1.01×10^{-6}	4.68×10^{-7}		
ATAN2	The full range	Note 7	1.01×10^{-6}	4.68×10^{-7}		
CABS	The full range	Note 1	9.15×10^{-7}	2.00×10^{-7}		
CCOS	$\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$	U	2.50×10^{-6} See Note 2	7.66×10^{-7}		
CDABS	The full range	Note 1	2.03×10^{-16}	4.83×10^{-17}		
CDCOS	$\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$	U	3.98×10^{-15} See Note 3	2.50×10^{-16}		
CDEXP	$\lvert X_1\rvert \leqq 1, \lvert X_2\rvert \leqq \pi/2$	U	3.76×10^{-16}	1.10×10^{-16}		
	$\lvert X_1\rvert \leqq 20, \lvert X_2\rvert \leqq 20$	U	2.74×10^{-15}	9.64×10^{-16}		
CDLOG	The full range except (1 + 0i)	Note 1	2.72×10^{-16}	5.38×10^{-17}		
CDSIN	$\lvert X_1\rvert \leqq 10, \lvert X_2\rvert \leqq 1$	U	2.35×10^{-15} See Note 4	2.25×10^{-16}		
CDSQRT	The full range	Note 1	1.76×10^{-16}	4.06×10^{-17}		
CEXP	$\lvert X_1\rvert \leqq 170, \lvert X_2\rvert \leqq \pi/2$	U	9.93×10^{-7}	2.67×10^{-7}		
	$\lvert X_1\rvert \leqq 170,$ $\pi/2 < \lvert X_2\rvert \leqq 20$	U	1.07×10^{-6}	2.73×10^{-7}		
CLOG	The full range except (1 + 0i)	Note 1	7.15×10^{-7}	1.36×10^{-7}		
COS	$0 \leqq X \leqq \pi$	U			1.19×10^{-7}	4.60×10^{-8}
	$-10 \leqq X < 0,$ $\pi < X \leqq 10$	U			1.28×10^{-7}	4.55×10^{-8}
	$10 < \lvert X\rvert \leqq 100$	U			1.14×10^{-7}	4.60×10^{-8}
COSH	$-5 \leqq X \leqq +5$	U	1.27×10^{-6}	2.63×10^{-7}		
COTAN	$\lvert X\rvert \leqq \pi/4$	U	1.07×10^{-6}	3.58×10^{-7}		
	$\pi/4 < \lvert X\rvert \leqq \pi/2$	U	1.40×10^{-6} (Note 5)	2.56×10^{-7}		
	$\pi/2 < \lvert X\rvert \leqq 10$	U	1.30×10^{-6} (Note 5)	3.11×10^{-7}		
	$10 < \lvert X\rvert \leqq 100$	U	1.49×10^{-6} (Note 5)	3.15×10^{-7}		
NOTES: (See end of figure.)						

Figure 12. *(Part 1 of 6)* Accuracy Figures

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (ϵ)	σ (ϵ)	M (E)	σ (E)
CQABS	Full range	Note 9	2.77×10^{-33}	5.45×10^{-34}		
CQCOS	$-10 < x < 10$ $-1 < y < 1$	U	6.87×10^{-33}	2.44×10^{-33}		
CQDVD#	*	Note 8	5.32×10^{-33}	1.42×10^{-33}		
CQEXP	$-170 < x < 170$ $-\frac{\pi}{2} < y < \frac{\pi}{2}$	U	3.82×10^{-33}	8.30×10^{-34}		
CQLOG	Full range	Note 9	4.53×10^{-33}	9.72×10^{-34}		
CQMPY#	*	Note 8	4.52×10^{-33}	1.27×10^{-33}		
CQSIN	$-10 < x < 10$ $-1 < y < 1$	U	7.26×10^{-33}	2.37×10^{-33}		
CQSQRT	Full range	Note 9	3.37×10^{-33}	7.27×10^{-34}		
CSIN	$\|X_1\| \leqq 10, \|X_2\| \leqq 1$	U	1.92×10^{-6} See Note 6	7.38×10^{-7}		
CSQRT	The full range	Note 1	7.00×10^{-7}	1.71×10^{-7}		
DARCOS	$\|X\| \leqq 1$	U	2.07×10^{-16}	7.05×10^{-17}		
DARSIN	$\|X\| \leqq 1$	U	2.04×10^{-16}	5.15×10^{-17}		
DATAN	The full range	Note 7	2.18×10^{-16}	7.04×10^{-17}		
DATAN2	The full range	Note 7	2.18×10^{-16}	7.04×10^{-17}		
DCOS	$0 \leqq X \leqq \pi$	U			1.79×10^{-16}	6.53×10^{-17}
	$-10 \leqq X < 0$, $\pi < X \leqq 10$	U			1.75×10^{-16}	5.93×10^{-17}
	$10 < X \leqq 100$	U			2.64×10^{-15}	1.01×10^{-15}
DCOSH	$\|X\| \leqq 5$	U	3.63×10^{-16}	9.05×10^{-17}		
DCOTAN	$\|X\| \leqq \pi/4$	U	2.46×10^{-16} (Note 5)	8.79×10^{-17}		
	$\pi/4 < \|X\| \leqq \pi/2$	U	2.78×10^{-13} (Note 5)	8.61×10^{-15}		
	$\pi/2 < \|X\| \leqq 10$	U	5.40×10^{-13} (Note 5)	1.13×10^{-14}		
	$10 < \|X\| \leqq 100$	U	8.61×10^{-13} (Note 5)	4.61×10^{-14}		
DERF	$\|X\| \leqq 1.0$	U	1.89×10^{-16}	2.60×10^{-17}		
	$1.0 < \|X\| \leqq 2.04$	U	2.87×10^{-17}	9.84×10^{-18}		
	$2.04 < \|X\| < 6.092$	U	1.39×10^{-17}	8.02×10^{-18}		
DERFC	$-6 < X < 0$	U	2.08×10^{-16}	6.52×10^{-17}		
	$0 \leqq X \leqq 1$	U	1.40×10^{-16}	2.59×10^{-17}		
	$1 < X \leqq 2.04$	U	4.11×10^{-16}	8.86×10^{-17}		
	$2.04 < X < 4$	U	3.26×10^{-16}	8.65×10^{-17}		
	$4 \leqq X < 13.3$	U	3.51×10^{-15}	1.96×10^{-15}		
NOTES: (See end of figure.)						

Figure 12. *(Part 2 of 6)* Accuracy Figures

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (ϵ)	σ (ϵ)	M (E)	σ (E)
DEXP	$\lvert X\rvert \leqq 1$	U	2.04×10^{-16}	5.43×10^{-17}		
	$1 < \lvert X\rvert \leqq 20$	U	2.03×10^{-16}	4.87×10^{-17}		
	$20 < \lvert X\rvert \leqq 170$	U	1.97×10^{-16}	4.98×10^{-17}		
DGAMMA	$0 < X < 1$	U	2.14×10^{-16}	7.84×10^{-17}		
	$1 \leqq X \leqq 2$	U	2.52×10^{-17}	6.07×10^{-18}		
	$2 < X < 4$	U	2.21×10^{-16}	8.49×10^{-17}		
	$4 \leqq X < 8$	U	5.05×10^{-16}	1.90×10^{-16}		
	$8 \leqq X < 16$	U	6.02×10^{-15}	1.78×10^{-15}		
	$16 \leqq X < 57$	U	1.16×10^{-14}	4.11×10^{-15}		
DLGAMA	$0 < X \leqq 0.5$	U	2.77×10^{-16}	9.75×10^{-17}		
	$0.5 < X < 3$	U			2.24×10^{-16}	7.77×10^{-17}
	$3 \leqq X < 8$	U	2.89×10^{-16}	8.80×10^{-17}		
	$8 \leqq X < 16$	U	2.86×10^{-16}	8.92×10^{-17}		
	$16 \leqq X < 500$	U	1.99×10^{-16}	3.93×10^{-17}		
DLOG	$0.5 \leqq X \leqq 1.5$	U			4.60×10^{-17}	2.09×10^{-17}
	$X < 0.5, X > 1.5$	E	3.32×10^{-16}	5.52×10^{-17}		
DLOG10	$0.5 \leqq X \leqq 1.5$	U			2.73×10^{-17}	1.07×10^{-17}
	$X < 0.5, X > 1.5$	E	3.02×10^{-16}	6.65×10^{-17}		
DSIN	$\lvert X\rvert \leqq \pi/2$	U	3.60×10^{-16}	4.82×10^{-17}	7.74×10^{-17}	1.98×10^{-17}
	$\pi/2 < \lvert X\rvert \leqq 10$	U			1.64×10^{-16}	6.49×10^{-17}
	$10 < \lvert X\rvert \leqq 100$	U			2.68×10^{-15}	1.03×10^{-15}
DSINH	$\lvert X\rvert \leqq 0.88137$	U	2.06×10^{-16}	3.74×10^{-17}		
	$0.88137 < \lvert X\rvert \leqq 5$	U	3.80×10^{-16}	9.21×10^{-17}		
DSQRT	The full range	E	1.06×10^{-16}	2.16×10^{-17}		
DTAN	$\lvert X\rvert \leqq \pi/4$	U	3.41×10^{-16}	6.27×10^{-17}		
	$\pi/4 < \lvert X\rvert \leqq \pi/2$	U	1.43×10^{-12} (Note 5)	2.95×10^{-14}		
	$\pi/2 < \lvert X\rvert \leqq 10$	U	2.78×10^{-13} (Note 5)	7.23×10^{-15}		
	$10 < \lvert X\rvert \leqq 100$	U	3.79×10^{-12} (Note 5)	9.50×10^{-14}		
DTANH	$\lvert X\rvert \leqq 0.54931$	U	1.91×10^{-16}	3.86×10^{-17}		
	$0.54931 < \lvert X\rvert \leqq 5$	U	1.54×10^{-16}	1.87×10^{-17}		
ERF	$\lvert X\rvert \leqq 1.0$	U	8.16×10^{-7}	1.10×10^{-7}		
	$1.0 < \lvert X\rvert \leqq 2.04$	U	1.13×10^{-7}	3.70×10^{-8}		
	$2.04 < \lvert X\rvert \leqq 3.9192$	U	5.95×10^{-8}	3.41×10^{-8}		
NOTES: (See end of figure.)						

Figure 12. *(Part 3 of 6)* Accuracy Figures

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M(ε)	σ(ε)	M(E)	σ(E)
ERFC	$-3.8 < X < 0$	U	9.10×10^{-7}	2.96×10^{-7}		
	$0 \leqq X \leqq 1.0$	U	7.42×10^{-7}	1.27×10^{-7}		
	$1.0 < X \leqq 2.04$	U	1.54×10^{-6}	3.78×10^{-7}		
	$2.04 < X \leqq 4.0$	U	2.28×10^{-6}	3.70×10^{-7}		
	$4.0 < X \leqq 13.3$	U	1.55×10^{-5}	8.57×10^{-6}		
EXP	$\|X\| \leqq 1$	U	4.65×10^{-7}	1.28×10^{-7}		
	$1 < \|X\| \leqq 170$	U	4.42×10^{-7}	1.15×10^{-7}		
FCQXI#	$2 \leqq J \leqq 160$, $10^{-75/J} < \|x + iy\| < 10^{75/J}$		$3.7 \times 10^{-33} \times J$	$10^{-33} \times (J - 1)$		
FQXPI#	$2 \leqq J \leqq 160$, $10^{-75/J} < x < 10^{75/J}$		$2.5 \times 10^{-33} \times (J - 1)$	$6.1 \times 10^{-34} \times (J - 1)$		
FQXPQ#	$0.99 < A < 1.01$ $-75 \log_A 10 < 8$ $< 75 \log_A 10$	U	5.68×10^{-31}	5.16×10^{-32}		
(A**B)	$0.5 < A \leqq 0.99$, or $1.01 \leqq A < 2$ $-75 \log_A 10 < B$ $< 75 \log_A 10$	U	5.65×10^{-32}	2.16×10^{-33}		
	$0 < A \leqq 0.5$, or $2 \leqq A < 10^{75}$	E	1.60×10^{-33}	3.87×10^{-34}		
	$-75 \log_A 10 < B$ $< 75 \log_A 10$	U	1.60×10^{-33}	3.87×10^{-34}		
FQXP2#	$-260 < x < 252$	U	1.52×10^{-33}	3.78×10^{-34}		
GAMMA	$0 < X < 1.0$	U	9.86×10^{-7}	3.66×10^{-7}		
	$1.0 \leqq X \leqq 2.0$	U	1.13×10^{-7}	3.22×10^{-8}		
	$2.0 < X \leqq 4.0$	U	9.47×10^{-7}	3.79×10^{-7}		
	$4.0 < X < 8.0$	U	2.26×10^{-6}	8.32×10^{-7}		
	$8.0 \leqq X \leqq 16.0$	U	2.20×10^{-5}	7.61×10^{-6}		
	$16.0 < X \leqq 57.0$	U	4.62×10^{-5}	1.51×10^{-5}		
QARCOS	$-1 \leqq x \leqq 1$	U	3.18×10^{-33}	9.81×10^{-34}		
QARSIN	$-1 \leqq x \leqq 1$	U	3.14×10^{-33}	7.89×10^{-34}		
QATAN	$-10^{75} < x < 10^{75}$	Note 10	2.92×10^{-33}	7.32×10^{-34}		
QATAN2	Full range	Note 9	3.53×10^{-33}	7.83×10^{-34}		
QCOS	$0 \leqq x \leqq \pi$	U	4.41×10^{-33}	6.58×10^{-34}	3.23×10^{-34}	1.48×10^{-34}
	$-10 < x < 0$, or $\pi \leqq x < 10$	U			3.43×10^{-34}	1.57×10^{-34}
	$-200 < x \leqq -10$, or $10 \leqq x < 200$	U			3.48×10^{-34}	1.57×10^{-34}
QCOSH	$-10 < x < 10$	U	5.83×10^{-33}	1.57×10^{-33}		
NOTES: (See end of figure.)						

Figure 12. *(Part 4 of 6)* Accuracy Figures

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M(ε)	σ(ε)	M(E)	σ(E)
QCOTAN	$-\frac{\pi}{4} < x < \frac{\pi}{4}$	U	3.02×10^{-33}	9.09×10^{-34}		
	$-\frac{\pi}{2} < x \leqq -\frac{\pi}{4}$, or $\frac{\pi}{4} \leqq x < \frac{\pi}{2}$	U	3.98×10^{-33}	1.09×10^{-33}		
	$-10 < x \leqq -\frac{\pi}{2}$, or $\frac{\pi}{2} \leqq x < 10$	U	4.55×10^{-33}	1.13×10^{-33}		
	$-200 < x \leqq -10$, or $10 \leqq x < 200$	U	3.98×10^{-33}	1.11×10^{-33}		
QERF	$\|x\| < 1$	U	3.0×10^{-33}	5.3×10^{-34}		
	$1 \leqq \|x\| < 2.84375$	U	9.2×10^{-34}	2.3×10^{-34}		
	$2.84375 \leqq \|x\| < 5$	U	1.9×10^{-34}	1.3×10^{-34}		
QERFC	$-5 < x < 0$	U	3.1×10^{-33}	1.2×10^{-33}		
	$0 \leqq x < 1$	U	3.3×10^{-33}	5.8×10^{-34}		
	$1 \leqq x < 2.84375$	U	7.7×10^{-33}	2.8×10^{-33}		
	$2.84375 \leqq x < 5$	U	4.88×10^{-32}	1.83×10^{-32}		
QEXP	$-1 < x < 1$	U	1.51×10^{-33}	4.27×10^{-34}		
	$-10 < x < 10$	U	1.53×10^{-33}	3.96×10^{-34}		
	$-180 < x < 174$	U	1.54×10^{-33}	3.82×10^{-34}		
QLOG	$0.99 < x < 1.01$	U	4.27×10^{-33}	1.51×10^{-33}	1.92×10^{-35}	8.36×10^{-36}
	$0.5 < x < 2$	U	4.06×10^{-33}	8.24×10^{-34}	3.17×10^{-34}	1.63×10^{-34}
	$10^{-78} < x < 10^{75}$	E	4.45×10^{-33}	8.77×10^{-34}		
QLOG10	$10^{-78} < x < 10^{75}$	E	3.59×10^{-33}	1.16×10^{-33}		
QSIN	$-\frac{\pi}{2} < x < \frac{\pi}{2}$	U	2.48×10^{-33}	3.12×10^{-34}	2.95×10^{-34}	1.17×10^{-34}
	$-10 < x \leqq -\frac{\pi}{2}$, or $\frac{\pi}{2} \leqq x < 10$	U			3.48×10^{-34}	1.60×10^{-34}
	$-200 < x \leqq -10$, or $10 \leqq x < 200$	U			3.50×10^{-34}	1.56×10^{-34}
QSINH	$-1 < x < 1$	U	2.91×10^{-33}	6.86×10^{-34}		
	$-10 < x \leqq -1$, or $1 \leqq x < 10$	U	6.71×10^{-33}	1.37×10^{-33}		
QSQRT	$10^{-50} < x < 10^{50}$	E	1.49×10^{-33}	2.95×10^{-34}		
	$10^{-78} < x < 10^{75}$	E	1.39×10^{-33}	2.76×10^{-34}		
QTAN	$-\frac{\pi}{4} < x < \frac{\pi}{4}$	U	3.75×10^{-33}	9.16×10^{-34}		
	$-\frac{\pi}{2} < x \leqq \frac{\pi}{4}$, or $\frac{\pi}{4} \leqq x < \frac{\pi}{2}$	U	2.77×10^{-33}	8.78×10^{-34}		
	$-10 < x \leqq -\frac{\pi}{2}$, or $\frac{\pi}{2} \leqq x < 10$	U	4.52×10^{-33}	9.16×10^{-33}		
	$-200 < x \leqq -10$, or $10 \leqq x < 200$	U	4.47×10^{-33}	9.12×10^{-33}		
QTANH	$-0.54931 < x < 0.54931$	U	2.41×10^{-33}	5.12×10^{-34}		
	$-5 < x \leqq -0.54931$, or $0.54931 \leqq x < 5$	U	2.09×10^{-33}	2.46×10^{-34}	1.04×10^{-33}	1.68×10^{-34}

NOTES: (See end of figure.)

Figure 12. *(Part 5 of 6)* Accuracy Figures

Entry Name	Argument Range	Sample E/U	Accuracy Figures			
			Relative		Absolute	
			M (e)	σ (e)	M (E)	σ (E)
SIN	$\|X\| \leqq \pi/2$	U	1.32×10^{-6}	1.82×10^{-7}	1.18×10^{-7}	4.55×10^{-8}
	$\pi/2 < \|X\| \leqq 10$	U			1.15×10^{-7}	4.64×10^{-8}
	$10 < \|X\| \leqq 100$	U			1.28×10^{-7}	4.52×10^{-8}
SINH	$-5 \leqq X \leqq +5$	U	1.26×10^{-6}	2.17×10^{-7}		
SQRT	The full range	E	4.45×10^{-7}	8.43×10^{-8}		
TAN	$\|X\| \leqq \pi/4$	U	1.71×10^{-6}	2.64×10^{-7}		
	$\pi/4 < \|X\| \leqq \pi/2$	U	1.05×10^{-6} (Note 5)	3.59×10^{-7}		
	$\pi/2 < \|X\| \leqq 10$	U	6.49×10^{-6} (Note 5)	3.38×10^{-7}		
	$10 < \|X\| \leqq 100$	U	1.57×10^{-6} (Note 5)	3.07×10^{-7}		
TANH	$\|X\| \leqq 0.7$	U	8.48×10^{-7}	1.48×10^{-7}		
	$0.7 < \|X\| \leqq 5$	U	2.44×10^{-7}	4.23×10^{-8}		

NOTES:

[1] The distribution of sample arguments upon which these statistics are based is exponential radially and is uniform around the origin.

[2] The maximum relative error cited for the CCOS function is based upon a set of 2000 random arguments within the range. In the immediate proximity of the points $\left(n + \frac{1}{2}\right)\pi + 0i$ (where $n = 0, \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[3] The maximum relative error cited for the CDCOS function is based upon a set of 1500 random arguments within the range. In the immediately proximity of the points $\left(n + \frac{1}{2}\right)\pi + 0i$ (where $n = 0, \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[4] The maximum relative error cited for the CDSIN function is based upon a set of 1500 random arguments within the range. In the immediate proximity of the points $n\pi + 0i$ (where $n = \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[5] The figures cited as the maximum relative errors are those encountered in a sample of 2500 random arguments within the respective ranges. See the appropriate section in the chapter "Algorithms" for a description of the behavior of errors when the argument is near a singularity or a zero of the function.

[6] The maximum relative error cited for the CSIN function is based upon a set of 2000 random arguments within the range. In the immediate proximity of the points $n\pi + 0i$ (where $n = \pm 1, \pm 2, \ldots,$) the relative error can be quite high, although the absolute error is small.

[7] The sample arguments were tangents of numbers uniformly distributed between $-\frac{\pi}{2}$ and $+\frac{\pi}{2}$.

[8] $x + iy = \delta e^{i\theta}$, where δ is exponentially distributed in $(0, 10^{35})$, and θ is uniformly distributed in $(-\pi, \pi)$.

[9] $x + iy = \delta e^{i\theta}$, where δ is exponentially distributed in $(0, 10^{75})$, and θ is uniformly distributed in $(-\pi, \pi)$.

[10] Tangents of linearly scaled random angles between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.

Figure 12. *(Part 6 of 6)* Accuracy Figures

Appendix A: Assembler Language Information

The mathematical and service subprograms in the VS FORTRAN library can be used by the assembler language programmer. Successful use depends on three things: (1) making the library available to the linkage editor; (2) setting up proper calling sequences, based on either a call macro instruction or a branch; and (3) supply correct parameters.

LIBRARY AVAILABILITY

The assembler language programmer must arrange for the desired subprograms (modules) to be taken from the VS FORTRAN library and brought into main storage, usually as a part of the programmer's load module. This can be done by employing the techniques described in the *OS/VS Linkage Editor and Loader,* and *Guide to the DOS/VSE Assembler,* publications.

For example, the VS FORTRAN library could be made part of the automatic call library by using these job control statements:

```
//jobname JOB desired operands
//stepname EXEC ASMFCLG,PARM.LKED='XREF,LIST,MAP'
//ASM.SYSIN DD*
  (assembler language program source deck)
/*
//LKED.SYSLIB DD DSNAME=data set name,DISP=SHR
/*
```

Subprograms requested in the source program would then be available to the linkage editor for inclusion in the load module.

CALLING SEQUENCES

Two general methods of calling are possible: (1) coding an appropriate macro instruction (for OS/VS and VM/370 CMS, see *OS/VS Supervisor Services and macro Instructions,* and for DOS/VSE see *DOS/VSE Data Management Concepts,* and *DOS/VSE Macro User's Guide*), such as CALL; or (2) coding assembler language branch instructions.

In all cases, a save area must be provided that:

- is aligned on a fullword boundary
- is at least as large as the size specified in Figures 13, 14, and 15, (but preferably the standard 18 words to ensure future compatibility)
- All extended precision mathematical subprograms (both explicit and implicit) use all 16 registers, and require their callers to supply a full 18 word save area.
- has its address in general register 13 at the time of the CALL macro instruction or branch

Notes:

1. For performance reasons, VS FORTRAN subprograms use certain floating-point registers (see Figure 14), but do not save and restore original register contents. If you wish floating-point information retained, you must save it before calling the subprogram and restore it on return.
2. From the DOS/VSE control program register 1 is not used but the execution parameters are passed as bit settings in the communications area.

If the called subprogram uses VS FORTRAN input/output, error processing, or interruption routines (see Figure 17), the calling program must initialize the execution environment by executing the following two instructions before the branch is made:

```
L    15,=V(VSCOM#)
BAL 14,64(15)
```

These instructions cause a branch into the VSCOM# subprogram, which initializes return coding and prepares routines to handle interruptions. If this initialization is omitted, an interruption or error may cause abnormal termination. (After initialization, VSCOM# returns to the instruction following the BAL.)

Note:

An initialization entry to VSCOM# is not required if the main program is written in FORTRAN, and the assembler language routine is an intermediate.

Entry Name(s)	Save Area (Fullwords)	Registers Used[1]	
		Result	Intermediate
AINT	9	0	2, 4, 6
ALGAMA, GAMMA	9	0	2, 4, 6
ALOG, ALOG10	7	0	2, 4, 6
AMAX0, AMIN0	6	0	
MAX0, MIN0	9	0*	
AMAX1, AMIN1	6	0	
MAX1, MIN1	9	0*	
AMOD, DMOD	9	0	2, 4, 6
ARCOS, ARSIN	10	0	2, 4
ATAN	5	0	2, 4, 6
ATAN, ATAN2	7	0	2, 4, 6
CABS	7	0, 2	4, 6
CCOS, CSIN	9	0, 2	4
CDABS	7	0, 2	4, 6
CDCOS, CDSIN	9	0, 2	4
CDEXP	8	0, 2	4, 6
CDLOG	8	0, 2	4, 6
CDSQRT	9	0, 2	4, 6
CEXP	8	0, 2	4, 6
CLOG	8	0, 2	4, 6
COS, SIN	7	0	2, 4
COSH, SINH	8	0	2, 4
COTAN, TAN	7	0	2, 4
CSQRT	9	0, 2	4, 6
DARCOS, DARSIN	13	0	2, 4
DATAN	5	0	2, 4, 6
DATAN, DATAN2	7	0	2, 4, 6
DCOS, DSIN	7	0	2, 4
DCOSH, DSINH	8	0	2, 4
DCOTAN, DTAN	7	0	2, 4, 6
DERF, DERFC	11	0	2, 4, 6
DEXP	7	0	2
DGAMMA, DLGAMA	11	0	2, 4, 6
DLOG, DLOG10	9	0	2, 4, 6
DMAX1, DMIN1	9	0	
DSQRT	7	0	2, 4
DTANH	5	0	2, 4, 6
EXP	11	0	
ERF, ERFC	11	0	2, 4, 6
IDINT, INT, IFIX	9	0*	
MOD	9	0*	
SQRT	7	0	2
TANH	5	0	2, 4, 6

[1]Floating-point; asterisk indicates general.

Figure 13. Assembler Information for the Explicitly Called Mathematical Subprograms

Entry Name(s)	Save Area (Fullwords)	Registers Used[1]	
		Result	Intermediate
CDMPY#, CDDVD#	5	0, 2	4, 6
CDVD#, CMPY#	5	0, 2	4, 6
FIXPI#	18	0*	
FRXPI#	18	0	
FDXPI#	18	0	
FRXPR#	18	0	
FDXPD#	18	0	
FCDXI#	18	0, 2	
FCXPI#	18	0, 2	

[1]Floating-point; asterisk indicates general.

Figure 14. Assembler Information for the Implicitly Called Mathematical Subprograms

Entry Name(s)	Save Area (Fullwords)
CCMPR#	18
CMOVE#	18
CNCAT#	18
LGE, LGT, LLE, LLT	18

Figure 15. Assembler Information for the Implicitly Called Character Subprograms.

Entry Name(s)	Save Area (Fullwords)
DUMP, PDUMP	18
DVCHK	10
EXIT	5
OVERFL	10

Figure 16. Assembler Information for the Service Subprograms

When a branch instruction rather than a call macro instruction is used to invoke a subprogram, several additional conventions must be observed:

- An argument (parameter) address list must be assembled on a fullword boundary. It consists of one 4-byte address constant for each argument, with the last address constant containing a 1 in its high order bit.
- The address of the first item in this argument address list must be in general register 1.
- From the DOS/VSE control program register 1 is not used but the execution parameters are passed as bit settings in the communications area.
- The address of the entry point of the called subprogram must be in general register 15.
- The address of the point of return to the calling program must be in general register 14.

The total requirements for an assembler language calling sequence are illustrated in Figure 14.

	L	15, = V(IBCOM#)	These two statements are necessary only if the called subprogram uses FORTRAN input/output, error, or interrupt routines (see Appendix B), and if the main program is not a FORTRAN routine.
	BAL	14, 64(15)	
	* *	* *	
	LA	13,area	General register 13 contains the address of the save area.
	LA	1,arglist	General register 1 contains the address of the argument list.
	L	15,entry	General register 15 contains the address of the subprogram.
	BALR	14,15	General register 14 contains the address of the point of return to the calling program.
	NOP	X'id''	This statement is optional. The id represents an identification number. This number is supplied by the programmer and may be any hexadecimal integer less than $2^{16} - 1$.
	* *	* *	
entry	DC	V (entry name)	
	or		
entry	DC	A (entry name)	NOTE: In this case, the entry name must be defined by an EXTRN instruction to obtain proper linkage.
	* *	* *	
area	DS	xxF	This statement defines the save area needed by the subprogram. The xx represents the minimum size of the save area required; however, the programmer is advised to use a save area of 18 fullwords for all subprograms. (The minimum save area requirements are given in Figures 13 and 14 for the mathematical subprograms and in Figure 16 for the service subprograms.)
	* *	* *	
For one argument			
	CNOP	DS0F	Aligns the argument list at a fullword boundary.
arglist	DC	X'80'	Places a 1 in the high order bit of the only argument.
	DC	AL3 (arg)	Contains the address of the argument.
For more than one argument			
arglist	DC	A (arg_1)	Contains the address of the first argument.
	DC	A (arg_2)	Contains the address of the second argument.
	•		
	•		
	•		
	DC	X'80'	Places a 1 in the high order bit of the last argument.
	DC	AL3 (arg_n)	Contains the address of the last argument.

Figure 17. General Assembler Language Calling Sequence

Supplying Correct Parameters

Arguments must be of the proper type, length, quantity, and in certain cases, within a specified range, for the subprogram called.

For mathematical and character subprograms, this information can be found in Figures 2 through 6. INTEGER *4 denotes a signed binary number four bytes long. REAL *4 and REAL *8 are normalized floating point numbers, 4 and 8 bytes long, respectively. COMPLEX *8 and COMPLEX *16 are complex numbers, 8 and 16 bytes long, respectively, whose first half contains the real part, and whose second half contains the imaginary part. Each part is a normalized floating-point number. Four-byte argument types must be aligned on fullword boundaries, and 8-byte and 16-byte types must be aligned on doubleword boundaries.

Argument information for nonmathematical subprograms can be found in "Service Subprograms."

Error messages resulting from incorrect arguments are explained in "Appendix C. Library Interruption Procedures, Error Procedures, and Messages."

Results

Each mathematical subprogram returns a single answer of a type listed in Figures 2 through 6 (see "Function Value Type"). The INTEGER answers are returned in general register 0, REAL answers are returned in floating-point register 0, and COMPLEX answers are returned in floating-point register 0 and 2. Result registers are listed by subprogram entry name in Figure 12.

For extended precision mathematical subprograms, results are always returned in the floating-point registers: 0 and 2 for REAL *16 results, and 0, 2, 4, and 6 for COMPLEX *32 results.

The location and form of the service subroutine results can be determined from the discussion in "Service Subprograms."

Example

To find the square root of the value in AMNT, the library square root subprogram (entry name SQRT) could be invoked, using the following statements (for assembler language MAIN programs only):

```
         L       15,=V(VSCOM#)
         BAL     14,64(15)
         .
         .
         .
         LA      13,SAVE
         CALL    SQRT,(AMNT),VL
         STE     0,ANSWER
         .
         .
         .
SAVE     DS      18F
         .
         ..
         .
AMNT     DC      E'144'
ANSWER   DC      E'0'
```

(The VL operand in CALL indicates that the macro expansion should flag the end of the parameter list.)

Employing only assembler language instructions, the sequence would be:

```
                L           15,=V(VSCOM#)
                BAL         14,64(15)
                .
                .
                .
                LA          13,SAVE
                LA          1,ARG
                L           15,ENTRY
                BALR        14,15
                STE         0,ANSWER
                .
                .
                .
ENTRY           DC          V(SQRT)
ANSWER          DC          E'0'
                .
                .
                .
SAVE            DS          18F
                .
                .
                .
                DS          0F,
ARG             DC          X'80',AL3(AMNT)
                .
                .
                .
AMNT            DC          E'144'
```

Note that, in both cases, a branch to VSCOM# is provided, a save area is set up, and AMNT meets argument specifications by being a four-byte non-negative normalized floating-point number, aligned on a fullword boundary.

In both cases, the answer is returned in floating-point register 0 as a four-byte floating-point number.

SPACE CONSIDERATIONS

Many of the mathematical subprograms require other mathematical subprograms for their calculations. In addition, most of the subprograms use the input/output, error processing, and interruption library subroutines. (This interdependence is outlined in "Appendix B. Storage Estimates."). Thus, although the programmer may request just one VS FORTRAN subprogram, the requirements of that subprogram may make the resultant load module quite large. The SQRT routine, for example, takes only 344 bytes of storage itself, but requires other subroutines that increase the load module size by approximately 20,000 bytes.

Appendix B: Storage Estimates

This Appendix contains decimal storage estimates (in bytes) for the library subprograms. The estimate given does not include any additional mathematical subprograms for VS FORTRAN routines that the subprograms may use during execution. The entry-names of any additional mathematical library subprograms used are given in Figure 18. Figures 18, 19, and 20 also indicate which mathematical and service subprograms require VS FORTRAN routines for input/output, interruption, and error procedures.

The programmer must add the estimates for all subprograms and routines needed to determine the amount of storage required. If the programmer has not made allowances for the storage required by any of these additional routines (see Figure 21), the amount of available storage may be exceeded and execution cannot begin, or may terminate abnormally.

Entry Name(s)	Decimal Estimates in Bytes	Additional Mathematical Subprograms Used	I/O, Error & Interrupt Routines
AINT	80		No
ALGAMA, GAMMA	848	ALOG, EXP	Yes
ALOG, ALOG10	464		Yes
AMAX0, AMIN0, MAX0, MIN0	224		No
AMAX1, AMIN1, MAX1, MIN1	224		No
AMOD, DMOD	120		No
ARCOS, ARSIN	496	SQRT	Yes
ATAN	200		No
ATAN, ATAN2	488		Yes
CABS	192	SQRT	Yes
CCOS, CSIN	760	EXP. SIN/COS	Yes
CDABS	200	DSQRT	Yes
CDCOS, CDSIN	832	DEXP, DSIN/DCOS	Yes
CDDVD#, CDMPY#	240		No
CDEXP	624	DEXP, DSIN/DCOS	Yes
CDLOG	488	DLOG, ATAN2	Yes
CDSQRT	328	DSQRT	Yes
CDVD#, CMPY#	216		No
CEXP	592	EXP, SIN/COS	Yes
CLOG	464	ALOG, ATAN2	Yes
COS, SIN	504		Yes
COSH, SINH	504	EXP	Yes
COTAN, TAN	648		Yes
QCABS	344	QSQRT	No*
CQCOS, CQSIN	1,100	QEXP, QSIN, QCOS	Yes
CQDVD#, CQMPY#	576		No
CQEXP	624	QEXP, QSIN, QCOS	Yes
CQLOG	584	QLOG, QATAN2	Yes
CQSQRT	504	QSQRT	No*
CSQRT	312	SQRT	Yes
CXMPR# (see Figure 20.)			
DARCOS, DARSIN	648	DSQRT	Yes
DATAN	312		No
DATAN, DATAN2	648		Yes
DCOS, DSIN	696		Yes
DCOSH, DSINH	592	DEXP	Yes
DCOTAN, DTAN	760		Yes
DERF, DERFC	808	DEXP	Yes
DEXP	704		Yes
DGAMMA. DLGAMMA	1056	DLOG, DEXP	Yes
DLOG, DLOG10	538		Yes

* (See notes at end of figure.)

Figure 18. Mathematical Subprogram Storage Estimates (Part 1 of 2)

Entry Name(s)	Decimal Estimates in Bytes	Additional Mathematical Subprograms Used	I/O, Error, & Interrupt Routines
DMAX1, DMIN1	120		No
DSQRT	352		Yes
DTANH	304	DEXP	Yes
EXP	424		Yes
ERF, ERFC	520	EXP	Yes
FCDXPI#	560	CDMPY#/CDDVD#	Yes
FCXPI#	536	CDVD#/CMPY#	Yes
FCQXI#	608	CQMPY#, CQDVD#	Yes
FDXPD#	464	DEXP, DLOG	Yes
FDXPI#	368		Yes
FQXPI#	384		Yes
FQXPQ#, FQXP2#†	2.880		Yes
FRXPR#	432	EXP, ALOG	Yes
FIXPI#	368		Yes
FRXPI#	360		Yes
IDINT, INT	136		No
MOD	56		No
QARCOS, QARSIN	1,104	QSQRT	Yes
QATAN, QATAN2	1,160		Yes
QCOS, QSIN	976		Yes
QCOSH, QSINH	896	QEXP	Yes
QCOTAN, QTAN	1,112		Yes
QERF, QERFC	1,200	QEXP‡	Yes
QEXP, QLOG, QLOG10†	2,880		Yes
QSQRT	520		Yes
QTANH	664	QEXP	No*
SQRT	344		Yes
TANH	256	EXP	Yes

* Note that although these mathematical subprograms do not themselves require the input/output, error or interruption routines, they use other mathematical subprograms which do.

† All share the same subroutine.

‡ When the argument falls between 2.84375 and 13.306, the module IFYQERF2 (size 1,300 bytes) is also used. IFYQERF2 in turn uses routine FQXPQ#.

Figure 18. Mathematical Subprogram Storage Estimates (Part 2 of 2)

Entry Name	Decimal Estimates (in bytes)	I/O, Error & Interrupt Routines
DUMP/PDUMP	870	Yes
DVCHK	60	Yes
EXIT	32	Yes
OVERFL	72	Yes

Figure 19. Service Subprogram Storage Estimates

Entry Name(s) [1]	Decimal Estimates (in bytes)	I/O, Error, and Interrupt Routines
CCMPR#, CXMPR# [2]	604	Yes[3]
CMOVE#	456	Yes
CNCAT#	480	Yes
LGE, LGT, LLE, LLT	1728	Yes

Notes to Figure:
1. No additional character subprograms are used.
2. The entry point CXMPR# is used for complex operands.
3. There is no I/O error or interrupt routine invoked for the CXMPR# entry name.

Figure 20. Character Subprogram Storage Estimates

Routine Name	Decimal Estimates (in bytes)
IFYUATBL[4]	Installation dependant
IFYUOPT	Installation dependent
IFYDIOCS	360
IFYIBCOM	2452
IFYLDFIO	4372
IFYNAMEL	3356
IFYOPSYS[5]	672
IFYVASU	2526
IFYVASYN[3]	1696
IFYVCLOS	596
IFYVCOMD	8799
IFYVCOMH	4640
IFYVCOM2	736
IFYVCONI	740
IFYVCOND	1316
IFYVCVTH	4216
IFYVDIOS[2]	2500
IFYVERRE	430
IFYVERRM	1124
IFYVFNTH	1414
IFYVIIOS	632
IFYVINQR	1792
IFYVMOPT	735
IFYVOPEN	1693
IFYVSCOM	2252
IFYVSERH	203
IFYVSIOS[1]	5724
IFYVSTAE	1671
IFYVTEN	680
IFYVTRCH	746
IFYVVIOS[1]	2304

NOTES TO FIGURE:
1. This module also requires dynamic storage. For each I/O unit used, the amount (in bytes) is 184 + buffer size(s).
2. This module also acquires dynamic storage. For each I/O unit used, the amount (in bytes) is 224 + buffer size(s).
3. This module also acquires dynamic storage. For each I/O unit used, the amount (in bytes) is 256 + buffer size(s).
4. The number of bytes in table IFYUATBL may be computed by the formula $16n + 8$, where n is the number of data set reference numbers requested at installation time.
5. This routine is for DOS/VSE only.

Figure 21. Library Execution-Time Routines Storage Estimates

APPENDIX C. LIBRARY INTERRUPTION PROCEDURES, ERROR PROCEDURES, AND MESSAGES

This appendix contains brief explanations of the program interruption and error procedures used by the FORTRAN library. The messages generated by the VS FORTRAN library are also given. A full description of program interrupts is given in the publication IBM System/370 Principles of Operation. For detailed information about error processing and message formats, see VS FORTRAN Application Programming: Guide.

LIBRARY INTERRUPTION PROCEDURES

The VS FORTRAN library processes those interrupts that are described below; all others are handled directly by the system Supervisor:

1. When an interrupt occurs, indicators are set to record exponent overflow, underflow, fixed-point, floating-point or decimal divide exceptions. These indicators can be interrogated dynamically by the subprograms described in the chapter, "Service Subroutine Subprograms."

2. A message is printed on the object program error unit when each interrupt occurs. The old program status word (PSW) printed in the message indicates the cause of each interrupt.

3. Result registers are changed when exponent overflow or exponent underflow (codes C and D) occur. Result registers are also set when a floating-point instruction is referenced by an assembler language execute (EXEC) instruction.

4. Condition codes set by floating-point addition or substraction instructions are altered for exponent underflow (code D).

5. After the foregoing services are performed, execution of the program continues from the instruction following the one that caused the interrupt.

LIBRARY ERROR PROCEDURES

During execution, the mathematical subprograms assume that the argument(s) is the correct type. However, some checking is done for erroneous arguments (for example, the wrong type, invalid characters, the wrong length, etc.); therefore, a computation performed with an erroneous argument has an unpredictable result. However, the nature of some mathematical functions requires that the input be within a certain range. For example, the square root of a negative number is not permitted. If the argument is not within the valid range given in Figures 2 through 6, an error messgae is written on the object program error unit data set defined by the installation during system generation. The execution of this load module or phase is terminated and control is returned to the operating system. However, execution can continue, with extended error handling for this program. This facility provides for standard corrective action by the user. For a full description of extended error handling, see VS FORTRAN Application Programming: Guide.

LIBRARY MESSAGES

The VS FORTRAN library generates three types of messages:

- Program interrupt messages
- Execution error messages
- Operator messages

All library messages are numbered. Program interrupt messages are written when an exception to a system restriction occurs, such as when an invalid storage address or an authorized access to protected storage is requested. Execution error messages are written when a FORTRAN library function or subroutine is misused or an I/O error occurs. Operator messages are written when a STOP n or PAUSE statement is executed.

Refer to "Section 1. Guide" of VS FORTRAN Diagnosis when a problem recurs after you have performed the specified programmer response for the message received.

PROGRAM INTERRUPT MESSAGES

Program interrupt messages are written with the old Program Status Word (PSW), which aids the programmer in determining the nature of the error.

Program interrupt messages consist of messages IFY207I, IFY208I, IFY209I, and IFY210I.

IFY207I VFNTH - PROGRAM INTERRUPT (P) - OVERFLOW PSW xxxxxxxxxxxxxxxx REGISTER CONTAINS nnnnnnnn

Explanation: The message indicates that an exponent-overflow exception, identified by the character C in the eighth position of the PSW, has occurred. This exception occurs when the result of a floating-point arithmetic operation is greater than or equal to 16^{63} (approximately 7.2×10^{75}).

Supplemental Data Provided: The floating point number (nnnnnnnn) before alteration.

Standard Corrective Action: Execution continues at the point of the interrupt with the result register set to the largest possible correctly-signed floating-point number that can be represented in short precision ($16^{63} \times (1-16^{-6})$), in long precision ($16^{63} \times (1-16^{-14})$, or in extended precision ($16^{63} \times (1-16^{-28})$).

Programmer Response: Make sure that a variable or variable expression does not exceed the allowable magnitude. Verify that all variables have been initialized correctly in previous source statements and have not been inadvertently modified.

IFY208I VFNTH - PROGRAM INTERRUPT (P) - UNDERFLOW PSW xxxxxxxxxxxxxxxx REGISTER CONTAINS nnnnnnnn

Explanation: The message indicates that an exponent-underflow exception, identified by a D in the eighth position of the PSW, has occurred. This exception occurs when the result of a floating-point arithmetic operation is less than 16^{-65} (approximately 5.4×10^{-79}).

Supplemental Data Provided: The floating point number (nnnnnnnn) before alteration.

Standard Corrective Action: Execution continues at the point of the interrupt with the result register set to a true zero of correct precision.

Programmer Response: Make sure that a variable or variable expression is not smaller than the allowable magnitude. Verify that all variables have been initialized correctly in previous source statements and have not been inadvertently modified.

IFY209I VFNTH - PROGRAM INTERRUPT (P) - DIVIDE CHECK PSW XXXXXXXXXXXXXXX REGISTER CONTAINS nnnnnnnn

Explanation: This message indicates that an attempt to divide by zero has occurred. A fixed-point-divide exception is identified by a 9 in the eighth position of the PSW; a floating-point-divide exception by an F.

Supplemental Data Provided: Floating-point number (nnnnnnnn) before alteration, for a floating-point interrupt.

Standard Corrective Action: For floating-point-divide, execution continues at the point of the interrupt with the result registers set to:

1. True zero of correct precision for case of n/0 where n=0.
2. Largest possible floating-point number of correct precision for case of n/0 where n≠0. For fixed-point-divide, leave registers unmodified and continue execution.

Programmer Response: Either correct the source where division by zero is occurring, or modify previous source statements to test for the possibilities, or bypass the invalid division.

IFY210I VFNTH - PROGRAM INTERRUPT (P/O) - CCCCCCCCCCCCC PSW XXXXXXXXXXXXXXX

Explanation: The operating system has detected a condition that causes a program interruption.

The letter P enclosed in parentheses indicates that the interruption was precise. This will always be the case for non-specification interrupt messages in FORTRAN except when using machines with special hardware on which imprecise interruptions may occur. The letter O enclosed in parentheses indicates that extended precision floating point simulation has taken place and a secondary interrupt occurred.

The eighth character position in the PSW (1, 4, 5, 6, 7, 9, C, D, or F) represents the hexadecimal code number associated with the type of interrupt. The following text describes those interrupts.

CODE 1—Operation Exception:

An operation exception is recognized when the processor encounters an instruction with an invalid operation code. The operation code may not be assigned, or the instruction with that operation code may not be available on the processor. (For the purpose of recognizing an operation exception, the first eight bits of an instruction, or, when the first eight bits have the hexadecimal value B2, the first 16 bits form the operation code.)

Supplemental Data Provided: The instruction-length code is 1, 2, or 3.

Standard Corrective Action: The operation is suppressed.

Programmer Response: Correct the operation code.

CODE 4—Protection Exception:

The protection exception (code 4) is recognized when the key of an operand in storage does not match the protection key in the PSW. A message is issued only if a specification exception (code

6) has already been recognized in the same instruction. Otherwise, the job terminates abnormally.

Supplemental Data Provided: None.

Standard Corrective Action: The interrupted instruction is ignored and execution continues at point of interrupt.

Programmer Response: If the job has been terminated with a completion code of SYSTEM=0C6 (Specification Interrupt), correct the source statements that are causing boundary misalignment.

CODE 5—Addressing Exception:

The addressing exception (code 5) is recognized when the address of the data is outside of the addressable storage for the particular system configuration or installation. A message is issued only if exception codes 5 or 6 have already been recognized in the same instruction. Otherwise, the job terminates abnormally.

Supplemental Data Provided: None.

Standard Corrective Action: The interrupted instruction is ignored and execution continues at point of interrupt.

Programmer Response: If the job has been terminated with a completion code of SYSTEM=0C6 (Specification Interrupt), correct the source statements that are causing boundary misalignment.

CODE 6—Specification Exception:

The specification exception (code 6) is recognized when a data address does not specify an integral boundary for that unit of information as required by the instruction, or an improper register is used in an instruction. For example, a specification error would occur during execution of the following instructions:

```
DOUBLE PRECISION D, E
COMMON A, B, C
EQUIVALENCE (B, D)
D = 3.0D02
```

Note: If an instruction causes a boundary violation, a specification interrupt will occur and the message will be issued with code 6. The boundary-adjustment routine will then be invoked if the BOUNDARY=ALIGN option was specified in the FORTLIB macro instruction during program installation. If an instruction that has been processed for boundary misalignment also contains a protection, addressing, or data error, the interrupt message will be reissued with the appropriate code (4, 5, or 7). The job will then terminate because both a specification error and a protection, addressing, or data error have been detected. The completion code will specify that the job terminated because of the specification error.

Supplemental Data Provided: None.

Standard Corrective Action: The interrupted instruction is ignored and execution continues at point of interrupt.

Programmer Response: Make sure that proper alignment of variables is guaranteed. Arrange variables in fixed descending order according to length, or force proper alignment with dummy variables. Construct COMMON blocks so that the displacement of each variable can be evenly divided by the element length associated with the variable. Use the MAP option for information on the relative address of each variable in the block. Make sure that EQUIVALENCE statements do not cause misalignment.

CODE 7—Data Exception:

The data exception (code 7) is recognized when the sign or digit codes for a CONVERT TO BINARY instruction are incorrect.

Supplemental Data Provided: None.

Standard Corrective Action: The interrupted instruction is ignored and execution continues at the point of interrupt.

Programmer Response: If the job has been terminated with a completion code of SYSTEM=0C6 (specification interrupt), correct the source statements that are causing boundary misalignment.

CODE 9—Fixed-Point-Divide Exception:

The fixed-point-divide exception (code 9) is recognized when division of a fixed-point number by zero is attempted. For example, a divide exception would occur during execution of the following statement:

```
K=I/J
```

where

J=0 and I=7

Supplemental Data Provided: None.

Standard Corrective Action: The interrupted instruction is ignored and execution continues at point of interrupt.

Programmer Response: Either correct the source where division by zero is occurring, or modify previous source statements to test for the possibility of, or to bypass, the invalid division.

CODE C—Exponent-Overflow Exception:

The exponent-overflow exception (code C) is recognized when the result of a floating-point addition, subtraction, multiplication, or division is greater than or equal to 16^{63} (approximately 7.2×10^{75}). For example, an exponent-overflow would occur during execution of the statement:

```
A=1.0E + 75 + 7.2E + 75
```

When the interrupt occurs, the result register contains a floating-point number whose fraction is normalized and whose sign is correct. However, the number is not usable for further computation since its characteristic field no longer reflects the true exponent. The content of the result register as it existed when the interrupt occurred is printed following the program interrupt message with the format:

```
REGISTER CONTAINED hhhhhhhhhhhhhhhh
```

where hhhhhhhhhhhhhhhh is the floating-point number in hexadecimal notation. (An additional 16 hexadecimal characters are printed for extended-precision numbers.)

Exponent overflow causes "exponent wraparound" - i.e., the characteristic field represents an exponent that is 128 smaller than the correct one. Treating bits 1 to 7 (the exponent characteristic field) of the of the floating-point number as a binary integer, the true exponent may be computed as follows:

```
TE = (Bits 1 to 7) + 128 - 64
```

Standard Corrective Action: The result register is set to the largest possible floating-point number that can be represented in short precision ($16^{63} \times (1-16^{-6})$) in long precision ($16^{63} \times (1-16^{-14})$), or in extended precision ($16^{63} \times (1-16^{-28})$), but the sign of the result is not changed. The condition code is not altered.

Programmer Response: Make sure that a variable expression does not exceed the allowable magnitude. Verify that all variables have been initialized correctly in previous source statements and have not been inadvertently modified in intermediate source.

CODE D—Exponent-Underflow Exception:

The exponent-underflow exception (code D) is recognized when the result of a floating-point addition, subtraction, multiplication, or division is less than 16^{-65} (approximately 5.4×10^{-79}). For example, an exponent-underflow exception would occur during execution of the statement:

```
A=-1.0E - 50 * 1.0E - 50
```

Although exponent underflows are maskable, FORTRAN jobs are executed with the mask enabled so that the library will handle such interrupts.

When the interrupt occurs, the result register contains a floating-point number whose fraction is normalized and whose sign is correct. However, the number is not usable for further computation since its characteristic field no longer reflects the true exponent. The content of the result register as it existed when the interrupt occurred is printed following the program interrupt message with the format:

```
REGISTER CONTAINS hhhhhhhhhhhhhhhh
```

where hhhhhhhhhhhhhhhh is the floating-point number in hexadecimal notation. (An additional 16 hexadecimal characters are printed for extended-precision numbers.) Exponent overflow causes "exponent wraparound" - i.e., the characteristic field represents an exponent that is 128 larger than the correct one. Treating bits 1 to 7 (the exponent characteristic field) of the floating-point number as a binary integer, the true exponent may be computed as follows:

```
TE = (Bits 1 to 7) - 128 - 64
```

Standard Corrective Action: The result register is set to a true zero of correct precision. If the interrupt resulted from a floating-point addition or subtraction operation, the condition code is set to zero to reflect the setting of the result register.

Programmer Response: Make sure that a variable or variable expression is not smaller than the allowable magnitude. Verify that all variables have been initialized correctly in previous source statements and have not been inadvertently modified in intermediate source. To take advantage of the 'exponent wraparound' feature and override the FORTRAN interruption routine, a programmer may handle the interrupt in his own program, but must call an assembly language subroutine to issue a SPIE macro instruction.

CODE F—Floating-Point-Divide Exception:

The floating-point-divide exception (code number F) is recognized when division of a floating-point number by zero is attempted. For example, a floating-point divide exception would occur during execution of the statement:

```
C=A/B
```

where

```
B=0.0 and A=1.0 or B=0.0 and A=0.0
```

Supplemental Data Provided: Registers before alteration.

Standard Corrective Action: The interrupted instruction is ignored and execution continues at point of interrupt with result registers set to:

1. True zero of the correct precision for the case of n/0 where n=0.

2. Largest possible floating point number of correct sign and precision for case of n/0 where n≠0.

Programmer Response: Either correct the source statement(s) where division by zero is occurring, or modify previous source statements to test for the possibility of, or to bypass, the invalid division.

EXECUTION ERROR MESSAGES

Execution error messages have the form:

IFYxxxI VFNTH [message text]

TRACEBACK FOLLOWS-ROUTINE ISN REG. 14,REG. 15,REG. 0,REG. 1

The description of each diagnostic message contains the error number, the abbreviated module name for the origin of the error, and an explanation describing the type of error. In addition, supplemental data is provided and standard corrective action to be taken to correct the error is described.

Variable information in the message is shown in lower case letters, and, in the corrective action descriptions, • denotes the largest possible number that can be represented for a floating-point value.

The abbreviated module name for the origin of the error is:

VASYN	IFYVASYN routine (performs asynchronous I/O processing).
VDIOS	IFYVDIOS routine (performs direct access I/O operations for FORTRAN load module execution).
VCVTH	IFYVCVTH routine (performs conversions).
VSIOS	IFYVSIOS routine (performs I/O operations for FORTRAN load module execution).
VCOMH	IFYVCOMH routine (performs interruption and error procedures).
VSERH	IFYVSERH routine (performs the processing of errors detected during compilation of the load modules).
LDFIO	IFYLDFIO routine (performs list-directed I/O processing).
NAMEL	IFYNAMEL routine (performs namelist processing).
other	Mathematical routine implicit or explicit entry points (perform mathematical calculations).

Note: Messages issued by the error handler contain no module or entry point name. For information on the error handling subroutines, refer to VS FORTRAN Application Programming Language Reference.

IFY160I VCOMH - FORMAT NESTED PARENTHESES TABLE OVERFLOW. REDUCE NUMBER OF NESTED PARENTHESES IN PROGRAM AND RERUN

Explanation: The format contains more nested parentheses than the library table can hold.

Supplemental Data Provided: None.

Standard Corrective Action: Parenthesis group is ignored. Processing continues. Results are unpredictable.

Programmer Response: Reduce the number of parenthesis groups to 50 or less.

IFY161I VASYN - ASYNCHRONOUS I/O NOT SUPPORTED ON THIS OPERATING SYSTEM (DOS OR CMS)

Explanation: A program called the asynchronous I/O scheduling routine while running in a DOS or CMS environment.

Supplemental Data Provided: TRACEBACK PATH is provided. If GOSTMT is used as a compiler option, then TRACEBACK provides the ISN of the I/O statement making the asynchronous I/O request.

Standard Corrective Action: The asynchronous I/O request is ignored and the ARRAY expected to be modified, if a READ (IN#) request, is unchanged. The ARRAY isn't saved or written if a WRITE (OUT#) request.

Programmer Response: Run the program on an OS system (MVS or VS1) or rewrite the program to use synchronous I/O (unformatted).

IFY162I VVIOS (CVIOS, DVIOS) - WRITE STATEMENT CANNOT BE ISSUED TO SEQUENTIALLY ACCESSED VSAM RRDS FILE filename

Explanation: An attempt was made to add a record to a sequentially accessed VSAM relative record file that was not empty when the file was opened.

Supplemental Data Provided: Name of the file upon which the request was made.

Standard Corrective Action: The execution is terminated.

Programmer Response: If a record must be added to a nonempty VSAM relative record file, use the access mode of DIRECT.

IFY163I VVIOS (CVIOS, DVIOS) - FILE POSITIONING INPUT/OUTPUT STATEMENT IS NOT ALLOWED IN THE DIRECT ACCESS METHOD

Explanation: A file positioning input/output statement (REWIND, BACKSPACE, or ENDFILE) was issued to a VSAM direct file.

Supplemental Data Provided: None.

Standard Corrective Action: The execution is terminated.

Programmer Response: Correct the program so that no file positioning input/output statements are issued for VSAM direct files.

IFY164I VVIOS (CVIOS, DVIOS) - RECORD LENGTH OF FILE filename IS LONGER THAN THE ONE DEFINED IN VSAM CATALOG

Explanation: The maximum record length for the file found in VSAM catalog (that is, the value specified in the RECORDSIZE parameter when the VSAM cluster is defined using Access Method Services) is less than the length of the record to be written.

Supplemental Data Provided: Name of the file upon which the request was made.

Standard Corrective Action: The execution is terminated.

Programmer Response: Either correct the program so that the length of the record to be written is not greater than the one in the VSAM catalog, or change the record length in the VSAM catalog by redefining the cluster.

IFY165I VVIOS (CVIOS, DVIOS) - FILE filename IS CONNECTED TO A KEY SEQUENCED DATA SET

Explanation: VS FORTRAN supports VSAM entry sequenced data sets (ESDS) and relative record data sets (RRDS). The file used was connected to VSAM key sequenced data sets (KSDS).

Supplemental Data Provided: Name of the file upon which the request was made.

Standard Corrective Action: The execution is terminated.

Programmer Response: Change the JCL so that the file is connected to ESDS or RRDS.

IFY166I VVIOS (CVIOS, DVIOS) - ENDFILE STATEMENT IS TREATED AS DOCUMENTATION FOR VSAM FILE filename

Explanation: A request was made to write an end-of-file mark on a VSAM file.

Supplemental Data Provided: Name of the file upon which the request was made.

Standard Corrective Action: The request is ignored.

Programmer Response: Remove the statement after carefully checking the effect of removing the statement.

IFY167I VVIOS (CVIOS, DVIOS) - ERROR ON VSAM FILE: WHEN ATTEMPTING TO PROCESS A(N) xxxxxxxxxx RC=yy ERROR CODE=zzz

Explanation: An error was detected by VSAM while an input or output statement indicated by xxxxxxxxxx was being processed. The return code and the error code returned by VSAM were yy and zzz, respectively.

Supplemental Data Provided: Name of the operation that caused the error and the return and error codes from VSAM.

Standard Corrective Action: The execution is terminated.

Programmer Response: Determine the cause for the error by examining the VSAM return and error codes.

IFY168I VVIOS (CVIOS, DVIOS) - xxxxxxxxxx OPERATION IS ISSUED TO UNOPENED VSAM FILE ON UNIT uuu

Explanation: An input or output request was made to an unopened VSAM file.

Supplemental Data Provided: Name of the operation issued to an unopened file.

Standard Corrective Action: The execution is terminated.

Programmer Response: Make sure that the OPEN statement for the file was successfully executed.

IFY169I DFNTH - EXTENDED PRECISION OPERATION NOT SUPPORTED IN DOS ENVIRONMENT, PSW, xxxxxxxxxx

Explanation: An extended precision machine operation was attempted in the DOS/VSE environment that is not supported by the machine instruction set. This is generally a divide operation.

Supplemental Data Provided: The program status word (PSW) at the point of interrupt. An IFY210I message with TRACEBACK or a dump of storage follows.

Standard Corrective Action: None.

Programmer Response: Change program to exclude the unsupported instruction.

IFY170I VSIOS - OPEN OR CLOSE STATEMENT NOT ALLOWED ON OBJECT PROGRAM ERROR UNIT, REQUEST FOR FILE filename

Explanation: An OPEN or CLOSE statement was directed to the unit upon which execution time error messages are being directed.

Supplemental Data Provided: Name of the file connected to the error message unit.

Standard Corrective Action: The request is ignored and the job terminated if an ERR= or IOSTAT parameter was not specified in the OPEN or CLOSE.

Programmer Response: Change the program to request I/O to a unit not being used for error messages.

IFY171I VSIOS - CLOSE STATUS OF KEEP IS NOT ALLOWED ON FILE OPENED WITH STATUS OF SCRATCH, FILE filename

Explanation: The file connected to the unit specified in the CLOSE statement was opened as a SCRATCH file and cannot be kept at close time.

Supplemental Data Provided: Name of the file connected to the unit specified in the CLOSE statement.

Standard Corrective Action: The CLOSE status is changed to DELETE and execution proceeds.

Programmer Response: Change either the OPEN or CLOSE STATUS parameter to agree with the file usage.

IFY172I VSIOS - FILE filename ALREADY CONNECTED TO A UNIT, OPEN REQUEST CANCELLED.

Explanation: The file whose name appears in the message already is connected to a unit which is different than the unit specified in the OPEN statement.

Supplemental Data Provided: Name of the file specified in the OPEN statement.

Standard Corrective Action: The OPEN request is ignored.

Programmer Response: Change the program to specify a different unit in the OPEN request or change the logic to use the current unit to which the file is connected.

IFY173I VSIOS - OPEN SPECIFYING UNFORMATTED I/O ATTEMPTED ON FORMATTED FILE filename

Explanation: FORMATTED and UNFORMATTED I/O request on the same file is not allowed.

Supplemental Data Provided: Name of the file upon which the request was made.

Standard Corrective Action: The I/O operation is ignored.

Programmer Response: Correct the program to direct FORMATTED and UNFORMATTED I/O to different files.

IFY174I VSIOS - OPEN SPECIFYING FORMATTED I/O ATTEMPTED ON UNFORMATTED FILE filename

Explanation: FORMATTED and UNFORMATTED I/O request on the same file is not allowed.

Supplemental Data Provided: Name of the file upon which the request was made.

Standard Corrective Action: The I/O operation is ignored.

Programmer Response: Correct the program to direct FORMATTED and UNFORMATTED I/O to different files.

IFY175I OPSYS - AN INVALID LITERAL PARAMETER WAS DETECTED ON THE CALL OPSYS STATEMENT

Explanation: The first parameter in the call to OPSYS did not specify a literal of FILEOPT or LOAD.

Supplemental Data Provided: None.

Standard Corrective Action: The request is ignored.

Programmer Response: Correct the program to specify the correct parameter value.

IFY176I OPSYS - THE FORTRAN LOGICAL UNIT NUMBER IS ASSIGNED TO SYSTEM USE, UNIT unit

Explanation: The unit specified in the call to OPSYS currently has a file connected and cannot be modified.

Supplemental Data Provided: Unit number specified in the call to OPSYS.

Standard Corrective Action: The request is ignored.

Programmer Response: Correct the program to process the I/O on a different unit.

IFY177I OPSYS - INVALID BLOCK SIZE SPECIFIED; ASCII (18-2048) OR EBCDIC (18-32767), UNIT unit

Explanation: An invalid block size was specified for the unit set up for ASCII or EBCDIC processing.

Supplemental Data Provided: Unit number specified in the call to OPSYS.

Standard Corrective Action: The request is ignored.

Programmer Response: Correct the program to specify a block size consistent with the file usage.

IFY178I OPSYS - INVALID BUFFER OFFSET SPECIFIED; GREATER THAN 99, EXCEEDS BLOCK SIZE OR IS NEGATIVE, UNIT unit

Explanation: The buffer offset specified was larger than the blocksize for the file, or was a negative value, or a value greater than 99.

Supplemental Data Provided: Unit number specified in the call to OPSYS.

Standard Corrective Action: The request is ignored.

Programmer Response: Correct the program to specify an offset consistent to the restrictions.

IFY179I OPSYS - AN I/O OPERATION HAS ALREADY BEEN PERFORMED ON THE UNIT, REQUEST IGNORED FOR UNIT unit

Explanation: An attempt was made to modify the parameters for a file which was already being used for I/O operations.

Supplemental Data Provided: Unit number specified in the call to OPSYS.

Standard Corrective Action: The request is ignored.

Programmer Response: Correct the program to process the I/O on a different unit.

IFY180I VOPEN - FILE PARAMETER IS NOT VALID FOR AN OPEN STATEMENT, UNIT unit

Explanation: The FILE= parameter on the OPEN statement did not specify a 7 character or less name and/or specified a name that did not start with an alphabetic character.

Supplemental Data Provided: Unit number for which the command was issued.

Standard Corrective Action: The OPEN statement is ignored.

Programmer Response: Correct the program to specify a correct filename.

IFY181I VOPEN - STATUS PARAMETER IS NOT VALID FOR AN OPEN STATEMENT, UNIT unit

Explanation: The STATUS= parameter did not specify NEW, OLD, SCRATCH, or UNKNOWN as the status of the file being opened on the unit.

Supplemental Data Provided: Unit number for which the command was issued.

Standard Corrective Action: STATUS is set to UNKNOWN and processing continues.

Programmer Response: Correct the program to specify a correct STATUS parameter.

IFY182I VOPEN - ACCESS PARAMETER IS NOT VALID FOR AN OPEN STATEMENT, UNIT unit

Explanation: The ACCESS= parameter did not specify SEQUENTIAL or DIRECT for the type of file access to be employed on the unit.

Supplemental Data Provided: Unit number for which the command was issued.

Standard Corrective Action: The OPEN request is ignored.

Programmer Response: Correct the program to specify a correct ACCESS parameter.

IFY183I VOPEN - BLANK PARAMETER IS NOT VALID FOR AN OPEN STATEMENT, UNIT unit

Explanation: The BLANK= parameter did not specify ZERO or NULL for the treatment of blanks on a FORMATTED I/O request.

Supplemental Data Provided: Unit number for which the command was issued.

Standard Corrective Action: The BLANK parameter is assigned the value NULL.

Programmer Response: Correct the program to specify a correct BLANK parameter.

IFY184I VOPEN - FORM PARAMETER IS NOT VALID FOR AN OPEN STATEMENT, UNIT unit

Explanation: The FORM= parameter did not specify FORMATTED or UNFORMATTED for the file.

Supplemental Data Provided: Unit number for which the command was issued.

Standard Corrective Action: The OPEN request is ignored.

Programmer Response: Correct the program to specify the correct formatting technique.

IFY185I VOPEN - STATUS OF SCRATCH NOT ALLOWED FOR A NAMED FILE OPEN STATEMENT, UNIT unit

Explanation: An OPEN requested FILE= and STATUS='SCRATCH' at the same time. The STATUS value is not allowed.

Supplemental Data Provided: Unit number for which the command was issued.

Standard Corrective Action: The STATUS value is set to UNKNOWN and processing continues.

Programmer Response: Correct the program to make the two parameters consistent with each other.

IFY186I VCLOS - STATUS PARAMETER IS NOT VALID FOR A CLOSE STATEMENT, UNIT unit

Explanation: The STATUS= parameter did not specify KEEP or DELETE, or a STATUS of KEEP was specified on the CLOSE statement for a file that was opened with a STATUS of SCRATCH.

Supplemental Data Provided: Unit number for which the command was issued.

Standard Corrective Action: The STATUS value is set to DELETE if the file was opened as SCRATCH; otherwise, the status is set to KEEP.

Programmer Response: Correct the program to specify the correct status values or make the status of the OPEN and CLOSE consistent with each other.

IFY187I DSPAN - LOWER BOUND OF ARRAY DIMN. GREATER THAN UPPER.

Explanation: For an array with adjustable dimensions the lower bound of an array dimension has been specified greater than the upper bound.

Supplemental Data Provided: None.

Standard Corrective Action: Span calculations are not completed for this array. Invalid results will probably generated from references to this array.

Programmer Response: Correct the calculation or specification of the dimensions.

IFY188I CITFN - ARGUMENT TO CHAR FUNCTION GREATER THAN 255

Explanation: A value greater than 255 (highest EBCDIC representation) has been specified for the CHAR function.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not evaluated and execution continues.

Programmer Response: Specify correct value.

IFY189I INDEX - INVALID LENGTH FOR INDEX - OP TWO.

Explanation: The length specified for the second operand of the INDEX function is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not evaluated and execution continues.

Programmer Response: Specify the correct length.

IFY190I INDEX - INVALID LENGTH FOR INDEX - OP ONE.

Explanation: The length specified for the first operand of the INDEX function is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not evaluated and execution continues.

Programmer Response: Specify the correct length.

IFY191I LXCMP - INVALID LENGTH FOR LEXICAL COMPARE - OPERAND TWO.

Explanation: The length specified for the second operand of the LGE, LGT LLE, or LLT function is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not evaluated and execution continues.

Programmer Response: Specify the correct length.

IFY192I LXCMP - INVALID LENGTH FOR LEXICAL COMPARE - OPERAND ONE.

Explanation: The length specified for the first operand of the LGE, LGT LLE, or LLT function is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not evaluated and execution continues.

Programmer Response: Specify the correct length.

IFY193I CCMPR - INVALID LENGTH FOR CHARACTER COMPARE - OP TWO.

Explanation: The length of the second operand of a Character relational compare (.eq., .lt., ...) is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not performed and execution continues.

Programmer Response: Specify the correct length.

IFY194I CCMPR - INVALID LENGTH FOR CHARACTER COMPARE - OP ONE.

Explanation: The length of the first operand of a Character relational compare (.eq., .lt., ...) is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not performed and execution continues.

Programmer Response: Specify the correct length.

IFY195I CMOVE - CHARACTER MOVE INVALID - TARGET AND SOURCE OVERLAP DESTRUCTIVELY.

Explanation: The storage locations assigned to the target and source are such that source data will be destroyed by the requested assignment.

Supplemental Data Provided: None.

Standard Corrective Action: The assignment is not performed and execution continues.

Programmer Response: Check storage MAP for storage assignments. Also check EQUIVALENCE statements.

IFY196I CMOVE - INVALID TARGET LENGTH FOR CHARACTER MOVE.

Explanation: The length of the target (left of equal variable) is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The assignment is not performed and execution continues.

Programmer Response: Specify the correct length.

IFY197I CMOVE - INVALID SOURCE LENGTH FOR CHARACTER MOVE.

Explanation: The length of the source (right of equal expression) is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The assignment is not performed and execution continues.

Programmer Response: Specify the correct length.

IFY198I CNCAT - CONCATENATED STRING LENGTH GREATER THAN TARGET

Explanation: The concatenation of the specified character strings will produce a string whose length is greater than 500 or greater than the length of the target (left of equal variable).

Supplemental Data Provided: None.

Standard Corrective Action: The concatenated string is truncated on the right.

Programmer Response: Specify the correct length.

IFY199I CNCAT - INVALID LENGTH FOR CONCATENATION OPERAND

Explanation: The length of one of the operands of a concatenation operation is less than or equal to zero or greater than 500.

Supplemental Data Provided: None.

Standard Corrective Action: The concatenation operation is not performed.

Programmer Response: Specify the correct length.

IFY200I VIIOS - END OF INTERNAL FILE, I/O PROCESSING ENDS

Explanation: The end of an internal file was reached before the completion of an internal I/O request.

Supplemental Data Provided: None.

Standard Corrective Action: Return to END= label if the request is a READ; otherwise, the job is terminated.

Programmer Response: Either keep a counter to avoid exceeding the end of record or file, or insert an END=n parameter on the READ statement for appropriate transfer of control on end of data set. Check all job control statements.

IFY201I VIIOS - REQUEST FOR INTERNAL FILE CONTROL, CLOSE OR LIST DIRECTED IS NOT ALLOWED

Explanation: A request for OPEN, CLOSE, list directed file input/output, or control operation has been requested for an internal file. Such operations are not supported for internal files.

Supplemental Data Provided: None.

Standard Corrective Action: The function is not performed, and execution continues.

Programmer Response: Change the source program, and rerun the job.

IFY203I IBCOM - INVALID COMBINATION OF INITIAL, TEST, AND INCREMENT VALUE FOR READ/WRITE IMPLIED DO, FILE filename

Explanation: A READ/WRITE statement with an implied DO had an invalid combination of initial, test, and increment values (I1, I2, and I3, respectively) for one of its levels of nesting:

1. I3=0, or
2. I2<I1 and I3≤I2-I1, or
3. I1<I2 and I3<0.

Supplemental Data Provided: Filename.

Standard Corrective Action: Processing is terminated.

Programmer Response: Check the statements which set the initial, test, and increment variables.

IFY204I LDFIO - ITEM SIZE EXCEEDS BUFFER LENGTH, FILE filename

Explanation: For a non-complex number, the number is longer than the buffer. For a complex number, half the length of the number plus one (for the comma) is longer than the buffer.

Supplemental Data Provided: Filename.

Standard Corrective Action: The remainder of the I/O list is ignored.

Programmer Response: Make sure that the record length specified is large enough to contain the longest item in the I/O list.

IFY205I VASYN - I/O SUBTASK ABENDED

Explanation: An asynchronous I/O subtask resulted in an abnormal termination.

Supplemental Data Provided: None.

Standard Corrective Action: Processing is terminated.

Programmer Response: Verify that all DD statements are coded correctly and refer to the appropriate data sets. Check all READ and WRITE statements and any END FILE, REWIND, and BACKSPACE statements. Check the system completion code for assistance in determining the type of error that caused abnormal termination.

IFY206I VCVTH - INTEGER VALUE OUT OF RANGE nnnnnnnn

Explanation: An integer was too large to be processed by the load module. (The largest integer that can be processed is 2**15-1 for INTEGER*2 and 2**31-1 for INTEGER*4.)

Supplemental Data Provided: Integer input for processing.

Standard Corrective Action: Specify as much of the lower order part of the given integer as will fit for the integer size (INTEGER*2 or INTEGER*4) specified.

Programmer Response: Make sure that all integer input data used is within the required range for the integer variable size.

IFY207I

Explanation: Refer to "Program Interrupt Messages" for information on this message.

IFY208I

Explanation: Refer to "Program Interrupt Messages" for information on this message.

IFY209I

Explanation: Refer to "Program Interrupt Messages" for information on this message.

IFY210I

Explanation: Refer to "Program Interrupt Messages" for information on this message.

IFY211I VCOMH - ILLEGAL field FORMAT CHARACTER SPECIFIED char FILE filename

Explanation: An invalid character has been detected in a FORMAT statement.

Supplemental Data Provided: The field containing the character in error, the character specified, and the filename.

Standard Corrective Action: Format field treated as an end of format.

Programmer Response: Make sure that all format specifications read in at object time are valid.

IFY212I VCOMH - FORMATTED I/O, END OF RECORD, FILE filename

Explanation: An attempt has been made to read or write a record, under FORMAT control, that exceeds the buffer length.

Supplemental Data Provided: Filename.

Standard Corrective Action: For a read, the remainder of the I/O list is ignored; for a write, a new record is started with no control character.

Programmer Response: If the error occurs on input, verify that a FORMAT statement does not define a FORTRAN record longer than the record supplied by the data set. No record to be punched should be specified as longer than 80 characters. For printed output, make sure than no specification is longer than the printer's line length.

IFY213I [VCOMH / LDFIO / VASYN] - READ, END OF RECORD, FILE filename

Explanation:

FOR VCOMH AND VASYN: The input list in an I/O statement without a FORMAT specification is larger than the logical record.

Supplemental Data Provided: Filename.

Standard Corrective Action: The remainder of the I/O list is ignored.

Programmer Response: Make sure the number of elements in the I/O list matches the number of items in the record.

FOR LDFIO: A FORTRAN list-directed READ statement attempted to read more items from a variable spanned logical record than were present in the record. (This message can be issued only when the record format is variable spanned.)

Supplemental Data Provided: Filename.

Standard Corrective Action: The remainder of the I/O list is ignored.

Programmer Response: Make sure that the records and the input data agree in number. Either delete extra variable names or supply additional logical records.

IFY214I [VSIOS / VASYN] - UNFORMATTED I/O, RECORD FORMAT NOT SPECIFIED AS VS OR VBS, FILE filename

Explanation:

FOR VSIOS: For unformatted records read or written in EBCDIC sequentially organized data sets, the record format specification must be variable spanned and can be blocked or unblocked. This message appears if the programmer has not specified variable spanned, or if an ASCII tape was specified.

Supplemental Data Provided: Filename.

Standard Corrective Action: For non-ASCII data sets, the read request is ignored; for a write request, the record form is changed to variable spanned.

Programmer Response: Correct the record format to variable spanned.

For VASYN: For unformatted records in an asynchronous I/O operation, the record format specification (RECFM) did not include the characters VS.

Supplemental Data Provided: Filename.

Standard Corrective Action: For an input operation, the read request is ignored; for an output operation, VS is assumed.

Programmer Response: Change the record format specification to VS.

IFY215I VCVTH - ILLEGAL DECIMAL CHARACTER char

Explanation: An invalid character was found in the decimal input corresponding to an I, E, F, or D format code.

Supplemental Data Provided: The record in which the character appeared.

Standard Corrective Action: 0 replaces the character encountered.

Programmer Response: If an IFY214I message has occurred previously, correct the source causing the error. Otherwise, make sure that all decimal input is valid. Correct any FORMAT statements specifying decimal input where character input should be indicated.

IFY216I VSIOS - INVALID USE OF I/O CONTROL COMMAND AT LOAD POINT filename

Explanation: The use of a BACKSPACE control command was recognized when the file was at the start of the first record.

Supplemental Data Provided: Filename for which command was issued.

Standard Corrective Action: The control command is ignored.

Programmer Response: Correct program to ensure that a BACKSPACE will not occur at the first command for a file.

IFY217I VSIOS - END OF DATA SET, FILE filename.

IFY217I VASYN - END OF DATA SET, FILE filename.

Explanation: An end of data set was sensed during a READ operation; that is, a program attempted to read beyond the data.

Supplemental Data Provided: filename.

Standard Corrective Action: The next file is read, that is, the data set sequence number is incremented by 1 in the OS environment. A permanent I/O error is set for the DOS environment.

Programmer Response: Either keep a counter to avoid exceeding the end of record or file, or insert an END=n parameter on the READ statement for appropriate transfer of control on end of data set. Check all job control statements.

IFY218I name - I/O ERROR, FILE filename, xxx...xxx

Explanation: VASYN, VSIOS or VDIOS - One of the following occurred:

- A permanent I/O error has been encountered.

- For sequential I/O, the length of a physical record is inconsistent with the default block size or the block size specified on the job control statement.
- An attempt has been made to read or write a record that is less than 18 bytes long on magnetic tape.

xxx...xxx is the character string specifying the type of I/O error.

Note: If a permanent I/O error has been detected while writing in the object error unit data set, the error message is written to the programmer either at the terminal or the OS SYSOUT data set, and job execution is terminated.

Supplemental Data Provided: Abbreviated module name and filename.

Standard Corrective Action: The interrupted instruction and the I/O request are ignored. After the traceback is completed, control is returned to the call routine statement designated in the ERR parameter of an I/O statement if that parameter was specified.

Note: ERR=parameter is honored.

Programmer Response: For sequential I/O, make sure that the length of the physical record is consistent with the default or specified block size. Check all job control statements. Make sure that no attempt has been made to read or write a magnetic tape record that is fewer than 18 bytes in length.

IFY219I [VSIOS / VDIOS / VASYN] **- OPEN FAILED, MISSING OR INVALID CONTROL STATEMENT, FILE filename**

Explanation:

FOR EBCDIC DATA SETS:

Either a data set is referred to in the load module and no job control statement is supplied for it, or a job control statement has an erroneous filename.

Supplemental Data Provided: Filename.

Standard Corrective Action: The interrupted instruction is ignored, and execution continues and the I/O request is ignored.

Note: If no job control statement has been supplied for the object error unit data set, the message is written either to the programmer at the terminal or console or to the OS SYSOUT data set, and the job is terminated.

Programmer Response: Either provide the missing job control statement, or correct any erroneous job control statement. Refer to VS FORTRAN Application Programming: Guide for more information.

FOR ASCII DATA SETS:

A data set may have been referred to in the load module but had no corresponding job control statement, or the job control statement may have had an erroneous filename.

Supplemental Data Provided: Filename.

Standard Corrective Action: The I/O request is ignored and execution continues.

Programmer Response: Either provide the missing job control statement, or correct any erroneous filename. Also, for OS files, be sure that the LABEL parameter on the DD statement specifies AL (or NL provided that the DCB subparameter OPTCD=Q is also specified). Also be sure that the operating system permits the use of ASCII data sets.

IFY220I name - UNIT NUMBER OUT OF RANGE, UNIT unit

Explanation: VSIOS or VASYN - A unit number exceeds the limit specified for unit numbers when the library was installed.

Supplemental Data Provided: Abbreviated module name and unit number.

Standard Corrective Action: The interrupted instruction is ignored, and execution continues.

Programmer Response: Correct the invalid unit number.

IFY221I NAMEL - NAME LARGER THAN EIGHT CHARACTERS. NAME=name

Explanation: An input variable name is longer than eight characters.

Supplemental Data Provided: First eight characters of the name specified.

Standard Corrective Action: The remainder of the NAMELIST request is ignored.

Programmer Response: Correct the invalid NAMELIST input variable, or provide any missing delimiters.

IFY222I NAMEL - NAME NOT IN NAMELIST DICTIONARY NAME=name

Explanation: An input variable name is not in the NAMELIST dictionary, or an array is specified with an insufficient amount of data.

Supplemental Data Provided: The name specified.

Standard Corrective Action: The remainder of the NAMELIST request is ignored.

Programmer Response: Make sure that a correct NAMELIST statement is included in the source module for all variable and array names read in using NAMELIST.

IFY223I NAMEL - END OF RECORD ENCOUNTERED BEFORE EQUAL SIGN. NAME=name

Explanation: An input variable name or a subscript has no delimiter.

Supplemental Data Provided: Name of item.

Standard Corrective Action: The remainder of the NAMELIST request is ignored.

Programmer Response: Make sure that all NAMELIST input data is correctly specified and all delimiters are correctly positioned. Check all delimiters. Make sure that sequence numbers are not present in columns 73 through 80.

IFY224I NAMEL - SUBSCRIPT FOR NON-DIMEMSIONED VARIABLE OR SUBSCRIPT OUT OF RANGE. NAME=name

Explanation: A subscript is encountered after an undimensioned input name, or the subscript is too large.

Supplemental Data Provided: Name of item.

Standard Corrective Action: The remainder of the NAMELIST request is ignored.

Programmer Response: Insert any missing DIMENSION statements, or correct the invalid array reference.

IFY225I VCVTH - ILLEGAL HEXADECIMAL CHARACTER char

Explanation: An invalid character is encountered on input for the Z format code.

Supplemental Data Provided: Display the record in which the character appeared.

Standard Corrective Action: 0 replaces the encountered character.

Programmer Response: Either correct the invalid character, or correct or delete the Z format code.

IFY226I VCVTH - REAL VALUE OUT OF RANGE chars

Explanation: A real number was too large or too small to be processed by the load module. (The largest number that can be processed is $16^{63}-1$; the smallest number that can be processed is 16^{-65}.)

Supplemental Data Provided: The field of input characters.

Standard Corrective Action: If the number was too large, the result is set to $16^{63}-1$. If the number was too small, the result is set to zero.

Programmer Response: Make sure that all real input is within the required range for the number specified.

IFY227I LDFIO - ERROR IN REPEAT COUNT, FILE filename

Explanation: A repeat count (k*---) was not followed by a blank, comma, or integer.

Supplemental Data Provided: Filename.

Standard Corrective Action: The remainder of the I/O list is ignored.

Programmer Response: Make sure that all repeat counts are followed by a valid character: a blank, a comma, or an integer.

IFY228I VASYN - LAST ELEMENT IN THE I/O LIST HAS A LOWER ADDRESS THAN THE FIRST ELEMENT. FILE filename

Explanation: An I/O list contained an element having a lower storage address than the first element in the list.

Supplemental Data Provided: Filename.

Standard Corrective Action: The interrupted instruction is ignored, and execution continues.

Programmer Response: Make sure that all elements in the I/O list are specified in the correct order.

IFY230I VSERH - SOURCE ERROR AT ISN ' ' EXECUTION TERMINATED. THE PROGRAM NAME IS 'program'.

Explanation: An attempt to run a program containing compile errors has been intercepted at the execution of the statement in error.

Supplemental Data Provided: ISN of statement in compiled program that is in error, and the name of the routine or subroutine in which the ISN is located.

Standard Corrective Action: Execution terminates with a return code of 16.

Programmer Response: Correct the source program statement, and rerun the job.

IFY231I VSIOS - SEQUENTIAL I/O ATTEMPTED ON A DIRECT ACCESS DATA SET, UNIT unit

IFY231I VDIOS - DIRECT ACCESS I/O ATTEMPTED BEFORE AN OPEN OR DEFINE FILE, UNIT unit

Explanation: (1) Direct access I/O statements are used for a sequential file, or I/O statements for a sequential file are used for a direct access file. (2) The same file cannot be opened in the same programming unit for both sequential and direct access processing.

Supplemental Data Provided: Unit number.

Standard Corrective Action: The I/O request is ignored.

Programmer Response:

For Cause 1: Either include the necessary DEFINE FILE statement for direct access or delete the DEFINE FILE for a sequential file. Make sure that all job control statements are correct. Verify that all data sets are referenced with valid FORTRAN statements for the file type.

For Cause 2: Make sure the same filename is not used twice within the same program unit for different types of access.

For Cause 3: For a file opened for direct access, the READ or WRITE statement must contain a record specification (REC= or u'r).

For a file opened for sequential access, the READ or WRITE statement must not contain a record specification (REC= or u'r).

IFY232I VDIOS - RECORD NUMBER nnnn OUT OF RANGE, FILE filename

Explanation: The relative position of a record is not a positive integer, or the relative position exceeds the number of records in the data set.

Supplemental Data Provided: Record number and filename.

Standard Corrective Action: The I/O request is ignored.

Programmer Response: Make sure that the relative position of the record on the data set has been specified correctly. Check all job control statements.

IFY233I VDIOS - RECORD LENGTH GREATER THAN 32767 SPECIFIED, FILE filename

Explanation: The record length specified in the DEFINE FILE or OPEN statement exceeds the capabilities of the system and the physical limitation of the volume assigned to the data set in the job control statement.

Supplemental Data Provided: Filename.

Standard Corrective Action: Record length is set to 32,000.

Programmer Response: Make sure that appropriate parameters of the job control statement conform to specifications in the DEFINE FILE or OPEN statement; the record length in both must be equivalent and within the capabilities of the system and the physical limitations of the assigned volume.

IFY234I VDIOS - ATTEMPT TO USE THE OBJECT ERROR UNIT AS A DIRECT ACCESS DATA SET, UNIT unit

Explanation: The data set assigned to print execution error messages cannot be a direct access data set.

Supplemental Data Provided: Unit number.

Standard Corrective Action: The request for direct I/O is ignored.

Programmer Response: Make sure that the object error unit specified is not direct access.

IFY235I VDIOS - ATTEMPT TO USE A UNIT FOR DIRECT ACCESS I/O WHICH IS CURRECTLY OPEN FOR SEQUENTIAL I/O, UNIT unit

Explanation: A unit number assigned to a direct access data set is used for a sequential data set.

Supplemental Data Provided: Unit number.

Standard Corrective Action: The request for direct I/O is ignored.

Programmer Response: Make sure that use of and/or reference to sequential data sets does not conflict with FORTRAN direct access data sets. Verify that device classes assigned by the installation do not conflict with the specification on the UNIT parameter of the job control statement. Make sure that the unit specified in a DEFINE FILE or OPEN statement defines a direct access data set. Check all job control statements.

IFY236I VDIOS - DIRECT ACCESS READ REQUESTED BEFORE DATASET WAS CREATED, FILE filename

Explanation: A READ is executed for a direct access data set that has not been created.

Supplemental Data Provided: Filename.

Standard Corrective Action: The I/O request is ignored.

Programmer Response: Make sure that either a data set utility program has been used, or appropriate parameters have been specified on the associated job control statement. For further information, refer to VS FORTRAN Application Programming: Guide.

IFY237I VDIOS - INCORRECT RECORD LENGTH SPECIFIED, FILE filename

Explanation: The length of the record did not correspond to the length of the record specified in the DEFINE FILE or the OPEN statement.

Supplemental Data Provided: Filename.

Standard Corrective Action: The I/O request is ignored.

Programmer Response: Make sure that the length of the records supplied matches the length specified in the DEFINE FILE or the OPEN statement. If necessary, change the statement to specify the correct record length.

IFY238I LDFIO - INCORRECT DELIMITER IN COMPLEX OR LITERAL INPUT, FILE filename

Explanation: A literal string in the input record(s) was not closed with a quotation mark (or was longer than 256 characters); alternatively, a complex number in the input record(s) contained embedded blanks, no internal comma, or no closing right parenthesis.

Supplemental Data Provided: Filename.

Standard Corrective Action: The remainder of the I/O list is ignored.

Programmer Response: Supply the missing quotation mark or amend the literal data to keep within the 256-character limit if the error was in the literal input. Check complex input numbers to see that they contain no embedded blanks, and that they contain an internal comma and a closing right parenthesis.

IFY239I VASYN - BLKSIZE IS NOT SPECIFIED FOR AN INPUT FILE, FILE filename

Explanation: The block size for an input file was not specified in the JCL or was specified as zero.

Supplemental Data Provided: Filename for which error occurred.

Standard Corrective Action: The I/O request is ignored.

Programmer Response: Make sure the block size is specified on the JCL for a new file.

IFY240I VSTAE - ABEND CODE IS: SYSTEM SSSS, USER UUUU, SCB/SDWA= HHHHHHHH.

IFY240I VSTAE - IO - NOT RESTORED. PSW IS XXXXXXXXXXXXXXXX.

IFY240I VSTAE - REGS 0-3 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

IFY240I VSTAE - REGS 4-7 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

IFY240I VSTAE - REGS 8-11 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

IFY240I VSTAE - REGS 12-15 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX

Explanation: An abnormal termination occurred. In some instances, pointers to subroutine entry points may have been destroyed, causing the traceback map to be incomplete. If an incomplete subroutine traceback map is printed, the following additional text appears between message IFY240I and the traceback map:

TRACEBACK MAY NOT BEGIN WITH ABENDING ROUTINE.

Supplemental Data Provided: SSSS represents the completion code if a system code caused termination; UUUU represents the completion code if a program code caused termination.

For specific explanations of the completion codes, see the messages and codes manual that applies to your operating system.

The SCB field (HHHHHHHH) gives the address of the STAE Control Block, which contains the old PSW and the contents of general registers at the time of abnormal termination. The PSW field gives the contents of the last FORTRAN program status word when abnormal termination occurred.

Input/output operations associated with the error are defined as NOT RESTORED, RESTORED, or NONE, as follows:

NOT RESTORED--Input/output has been halted and cannot be restored.

RESTORED--Input/output has been halted. FORTRAN will attempt to restart I/O and then close data sets.

NONE--No active I/O operations were present at abnormal termination time. Fortran will close data sets.

Standard Corrective Action: None.

Programmer Response: Use the abend code, the contents of the SCB and PSW, and any accompanying system messages, to determine the nature of the error.

IFY241I FIXPI INTEGER BASE=0, INTEGER EXPONENT=exp LE ZERO

Explanation: For an exponentiation operation (I**J) in the subprogram IFYFIXPI (FIXPI#) where I and J represent integer variables or integer constants, I is equal to zero and J is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action: Result = 0.

Programmer Response: Make sure that integer variables and/or interger constants for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during program execution, then either modify the operands or insert source code to test for the situation and make the appropriate compensation. Bypass the exponentiation operation if necessary.

IFY242I FRXPI - REAL*4 BASE=0.0, INTEGER EXPONENT=exp LE ZERO

Explanation: For an exponentiation operation (R**J) in the subprogram IFYFRXPI (FRXPI#), where R represents a REAL*4 variable or REAL*4 constant and J represents an integer variable or integer constant, R is equal to zero and J is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action:

If BASE=0,EXP<0,RESULT=•;
If BASE=0.0,EXP=0,RESULT=1.

Programmer Response: Make sure that both the real variable or constant base and the integer variable or constant exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during program execution, then either modify the operand(s), or insert source code to test for the situation and make the appropriate compensation. Bypass the exponentiation operation if necessary.

IFY243I FDXPI - REAL*8, BASE=0.0, INTEGER EXPONENT=exp LE ZERO

Explanation: For an exponentiation operation (D**J) in the subprogram IFYFDXPI (FDXPI#), where D representa a REAL*8 variable or REAL*8 constant and J represents an integer variable or integer constant, D is equal to zero and J is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action:

If BASE=0,EXP<0,RESULT=•;
If BASE=0.0,EXP=0,RESULT=1.

Programmer Response: Make sure that both the real variable or constant base and the integer variable or constant exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during execution, then either modify the operand(s), or insert source code to test for the situation and make the appropriate compensation. Bypass the exponentiation operation if necessary.

IFY244I FRXPR - REAL*4, BASE=0.0, REAL*4 EXPONENT=exp LE ZERO

Explanation: For an exponentiation operation (R**S) in the subprogram IFYFRXPR (FRXPR#), where R and S represent REAL*4 variables or REAL*4 constants, R is equal to zero and S is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action:

If BASE=0,EXP<0,RESULT=•;
If BASE=0.0,EXP=0,RESULT=1.

Programmer Response: Make sure that both the real variable or constant base and exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during program execution, then either modify the operand(s), or insert source code to test for the situation and make appropriate compensation. Bypass the exponentiation operation if necessary.

IFY245I FDXPD REAL*8 BASE=0.0, REAL*8 EXPONENT=exp, LE ZERO

Explanation: For an exponentiation operation (D**P) in the subprogram IFYFDXPD (FDXPD#), where D and P represent REAL*8 variables or REAL*8 constants, D is equal to zero and P is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action: Result=0.

Programmer Response: Make sure that both the real variable or constant base and exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during program execution, then either modify the operand(s), or insert source code to test for the situation and make appropriate compensation. Bypass the exponentiation operation if necessary.

IFY246I FCXPC COMPLEX*8 BASE=0.0+0.0I, EXPONENT=exp LE ZERO

IFY246I FCXPI COMPLEX*8 BASE=0.0+0.0I, INTEGER EXPONENT=exp, LE ZERO

Explanation: For an exponentiation operation (Z**J) in the subprograms IFYFCXPI (FCXPI#) and IFYFCXPC (FCXPC#), where Z represents a COMPLEX*8 variable or COMPLEX*8 constant and J represents an integer variable or integer constant, Z is equal to zero and J is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action:

If BASE=0,EXP<0,RESULT=•;
If BASE=0.0,EXP=0,RESULT=1

Programmer Response: Make sure that both the complex variable or constant base and the integer variable or constant exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during program execution, then either modify the operand(s), or insert source code to test for the situation and make the appropriate compensation. Bypass the exponentiation operation if necessary.

IFY247I FCDXI COMPLEX*16 BASE=0.0+0.0I, INTEGER EXPONENT=exp, LE ZERO

IFY247I FCDCD COMPLEX*16 BASE=0.0+0.0I, EXPONENT=exp, LE ZERO

Explanation: For an exponentiation operation (Z**J) in the subprograms IFYFCDXI (FCDXI#) and IFYFCDCD (FCDCD#), where Z represents a COMPLEX*16 variable or COMPLEX*16 constant and J represents an integer variable or integer constant, Z is equal to zero and J is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action:

If BASE=0,EXP<0,RESULT=•;
If BASE=0.0,EXP=0,RESULT=1

Programmer Response: Make sure that both the complex variable or constant base and the integer variable or constant exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during program execution, then either modify the operand(s), or insert source code to test for the situation and make the appropriate compensation. Bypass the exponentiation operation if necessary.

IFY248I FQXPI# REAL*16 BASE=0.0, INTEGER EXPONENT=exp, LE ZERO

Explanation: For an exponentiation operation (Q**J) in the subprogram IFYFQXPI (FQXPI#), where Q represents a REAL*16 variable or constant and J represents an integer variable or constant, Q is equal to zero and J is less than or equal to zero.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action:

If BASE=0,EXP<0,RESULT=•;
If BASE=0.0,EXP=0,RESULT=1

Programmer Response: Make sure that both the real variable or constant base and the integer variable or constant exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during execution, then either modify the operand(s), or insert source code to test for the situation and make the appropriate

compensation. Bypass the exponentiation operation if necessary.

IFY249I FQXPQ# REAL*16 BASE=base,REAL*16 EXP=exp, BASE=0.0, AND EXP LE ZERO OR BASE LT ZERO AND EXP NE ZERO

Explanation: For an exponentiation operation (X**Y) in the subprogram IFYFQXPQ(FQXPQ#), where X and Y represent REAL*16 variables or constants, if X equals zero, Y must be greater than zero; if X is less than zero, Y must equal zero. One of these conditions is violated.

Supplemental Data Provided: Base and exponent specified.

Standard Corrective Action:

If BASE=0 and EXP<0,RESULT=•;
If BASE=0.0 and EXP=0,RESULT=1;
If BASE=<0.0 and EXP≠0,RESULT=|X|**Y.

Programmer Response: Make sure that both the real variable or constant base and exponent for an exponentiation operation are within the allowable range. If the base and exponent may or will fall outside that range during program execution, then either modify the operand(s), or insert source code to test for the situation and make appropriate compensation. Bypass the exponentiation operation if necessary.

IFY250I FQXPQ# REAL*16 BASE=base, REAL*16 EXP=exp, ARGUMENT COMBINATION EXP.*LOG2 (BASE) GE 252

Explanation: For an exponentiation operation in the subprogram IFYFQXPQ, the argument combination of y*log2(x) generates a number greater than or equal to 252.

Supplemental Data Provided: The arguments specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the base and exponent are within the allowable range. If necessary, restructure arithmetic operations.

IFY251I SQRT ARGUMENT=arg LT ZERO

Explanation: In the subprogram IFYSSQRT (SQRT), the argument is less than 0.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=$|X|^{1/2}$.

Programmer Response: Make sure that the argument is within allowable range. Either modify the argument, or insert source code to test for a negative argument and make the necessary compensation. Bypass the function reference if necessary.

IFY252I EXP ARG=arg, GT 174.673

Explanation: In the subprogram IFYSEXP(EXP), the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument to the exponentiation function is within allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, If necessary, modify the argument or bypass the source referencing the

function subprogram.

IFY253I ALOG-ALOG10 ARG=arg, LE ZERO

Explanation: In the subprogram IFYSLOG (ALOG and ALOG10), the argument is less than or equal to zero. Because this subprogram is called by an exponential subprogram, this message may also indicate that an attempt has been made to raise a negative base to a real power.

Supplemental Data Provided: Argument specified.

Standard Corrective Action:

If X=0, RESULT=-•;
If X<0, RESULT=log|X| or $\log_{10}$ |X|.

Programmer Response: Make sure that the argument to the logarithmic function is within the allowable range. If the argument may or will be outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY254I SIN-COS ABS(ARG)=arg GE PI*(218)**

Explanation: In the subprogram IFYSSCN (SIN and COS), the absolute value of an argument is greater than or equal to 2^{18} * pi (2^{18} * pi=.8235496640624996D + 06).

Supplemental Data Provided: None.

Standard Corrective Action: Result=SQRT(2)/2.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131^{0}) to the trigonometric sine or cosine function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY255I ATAN2 ARGUMENTS=0.0

Explanation: In the subprogram IFYSATN2, when entry name ATAN2 is used, both arguments are equal to zero.

Supplemental Data Provided: None.

Standard Corrective Action: Result=0.

Programmer Response: Make sure that both arguments do not become 0 during program execution, or are not inadvertently initialized or modified to 0. Provide code to test for the situation and, if necessary, modify the arguments or bypass the source referencing the function subprogram.

IFY256I SINH-COSH/ARG/=/arg/, GE 175.366

Explanation: In the subprogram IFYSSCNH (SINH or COSH), the argument is greater than or equal to 175.366.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: SINH(X)=±•; COSH(X)=•

Programmer Response: Make sure that the argument to the hyperbolic sine or cosine function is within the allowable range. If the argument may or will exceed that range during

program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY257I ARSIN-ARCOS /ARG/=/arg/ GT 1

Explanation: In the subprogram IFYSASCN (ARSIN or ARCOS), the absolute value of the argument is greater than 1.

Supplemental Data Provided: Argument specified.

Standard Corrective Action:

If x>1.0,ARCOS(x)=0;
If <-1.0,ARCOS(x)=pi;
If x>1.0,ARSIN=pi/2;
If x<-1.0,ARSIN=-pi/2.

Programmer Response: Make sure that the argument to the arcsine or arccosine function is between -1 and +1, inclusive. If the argument may or will fall outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY258I TAN-COTAN /ARG/=/arg(HEX=hex)/, GE PI*218**

Explanation: In the subprogram IFYSTNCT (TAN or COTAN), the absolute value of the argument is greater than or equal to 2^{18}*pi (2^{18}*pi=.82354966406249996D+06).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=1.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131^{0}) to the trigonometric tangent or cotangent function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY259I TAN-COTAN /ARG/=/arg(HEX=hex)/, APPROACHES SINGULARITY

Explanation: In the subprogram IFYSTNCT (TAN or COTAN), the argument value is too close to one of the singularities (±pi/2, ±3pi/2, ... for the tangent or ±pi, ±2pi, ... for the cotangent).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131^{0}) to the trigonometric tangent or cotangent function is within the allowable range. If the argument may or will approach the corresponding singularities for the function during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY260I FQXP2# REAL*16 EXPONENT=exp, GE 252

Explanation: In the subprogram IFYFQXPR(FQXP2#), the exponent is beyond the range of 2^{252}.

Supplemental Data Provided: Exponent specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the exponent is within the allowable range.

IFY261I DSQRT ARGUMENT=arg LT ZERO

Explanation: In the subprogram IFYLSQRT(DSQRT), the argument is less than 0.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=$|X|^{1/2}$.

Programmer Response: Make sure that the argument is within the allowable range. Either modify the argument, or insert source code to test for a negative argument and make the necessary compensation. Bypass the function reference if necessary.

IFY262I DEXP ARG=arg, GT 174.673

Explanation: In the subprogram IFYLEXP(DEXP), the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument to the exponential function is within allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY263I DLOG-DLOG10 ARG=arg, LE ZERO

Explanation: In the subprogram IFYLLOG (DLOG and DLOG10), the argument is less than or equal to zero. Because the subprogram is called by an exponential subprogram, this message may also indicate that an attempt has been made to raise a negative base to a real power.

Supplemental Data Provided: Argument specified.

Standard Corrective Action:

If X=0,RESULT=-•;
If X<0,RESULT=log|X| or $\log_{10}|X|$.

Programmer Response: Make sure that the argument to the logarithmic function is within the allowable range. If the argument may or will be outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY264I DSIN-DCOS /ARG/=/arg(HEX=hex)/, GE PI250**

Explanation: In the subprogram IFYLSCN (DSIN and DCOS), the absolute value of the argument is greater than or equal to 2^{50}*pi (2^{50}*pi=.3537118873780239D+16).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=SQRT(2)/2.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131°) to the trigonometric sine or cosine function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY265I DATAN2 ARGUMENTS=0.0

Explanation: In subprogram IFYLATN2, when entry name DATAN2 is used, both arguments are equal to zero.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=0.

Programmer Response: Make sure that both arguments do not become zero during program execution, or are not inadvertently initialized or modified to zero. Provide code to test for the situation and, if necessary, modify the arguments or bypass the source referencing the function subprogram.

IFY266I DSINH-DCOSH /ARG/=arg/, GE 175.366

Explanation: In the subprogram IFYSCNH (DSINH or DCOSH), the absolute value of the argument is greater than or equal to 175.366.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: DSINH(X)=±•; DCOSH(X)=•

Programmer Response: Make sure that the argument to the hyperbolic sine or cosine function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY267I DARSIN-DARCOS /ARG/=/arg/ GT 1

Explanation: In the subprogram IFYLASCN (DARSIN or DARCOS), the absolute value of the argument is greater than 1.

Supplemental Data Provided: Argument specified.

Standard Corrective Action:

```
If x >  1.0 DARCOS(x) = 0;
If x < -1.0 DARCOS(X) = pi;
If x >  1.0 DARSIN = pi/2;
If x < -1.0 DARSIN = -pi/2.
```

Programmer Response: Make sure that the argument to the arcsine or arccosine function is between -1 and +1, inclusive. If the argument may or will fall outside that range during execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY268I DTAN-DCOTAN /ARG/=/arg(HEX=hex)/, GE PI*250**

Explanation: In the subprogram IFYLTNCT (DTAN or DCOTAN), the absolute value of the argument is greater than or equal to $2^{50} \times pi$ ($2^{50} \times pi$=.3537118876014220ID+16).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=1.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131°) to the trigonometric tangent or cotangent function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY269I DTAN-DCOTAN /ARG/=/arg(HEX=hex)/, APPROACHES SINGULARITY

Explanation: In the subprogram IFYLTNCT (DTAN or DCOTAN), the argument value is too close to one of the singularities (±pi/2, ±3pi/2, ... for the tangent; ±pi, ±2pi, ... for the cotangent).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131°) to the trigonometric tangent or cotangent function is within the allowable range. If the argument may or will approach the corresponding singularities for the function during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY270I FCQXI COMPLEX*32 BASE=0.0+0.0I, INTEGER EXPONENT=exp, LE ZERO

IFY270I FCQCG COMPLEX*32 BASE=0.0*0.0I, EXPONENT=exp, LE ZERO

Explanation: In the subprograms IFYFCQXI (FCQXI#) and IFYFCQCG (FCQCG#), a base 0 number has been raised to a power less than or equal to zero.

Supplemental Data Provided: Argument specified.

Standard Corrective Action:

If X=0+0i and J=0, RESULT=1+0i;
If X=0+0i and J<0, RESULT=•+0i.

(where J=exponent)

Programmer Response: Make sure the base is a non-zero number or raise the exponent to a non-zero value.

IFY271I CEXP REAL ARG=arg(HEX=hex), GT 174.673

Explanation: In the subprogram IFYCSEXP (CEXP), the value of the real part of the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•(COS X + iSIN X) where X is the imaginary portion of the argument.

Programmer Response: Make sure that the argument to the exponential function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY272I CEXP IMAG ARG=arg(HEX=hex), ABS VALUE GE PI*218**

Explanation: In the subprogram IFYCSEXP (CEXP), the absolute value of the imaginary part of the argument is greater than or equal to 2^{18}*pi (2^{18}*pi=.82354966406249996D+06).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=e^{X1}+0*i.

Programmer Response: Make sure that the argument to the exponential function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation, and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY273I CLOG ARGUMENT=0.0+0.0I

Explanation: In the subprogram IFYCSLOG (CLOG), the real and imaginary parts of the argument are equal to zero.

Supplemental Data Provided: None.

Standard Corrective Action: Result=-∞+0i.

Programmer Response: Make sure that both the real and imaginary parts of the argument do not become zero during program execution, or are not inadvertently initialized or modified to zero. Provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY274I CSIN-CCOS /REAL ARG/=/arg(HEX=hex)/, GE PI*218**

Explanation: In the subprogram IFYCSSCN (CSIN or CCOS), the absolute value of the real part of the argument is greater than or equal to 2^{18}*pi (2^{18}*pi=.82354966406249996D+06).

Supplemental Data Provided: Argument specified.

Standard Corrective Action:

Result=COSH(x_2)+0*i; CSIN=0+SINH(x_2)*i.

Programmer Response: Make sure that the real part of the argument (in radians where 1 radian is equivalent to 57.2957795131^0) to the trigonometric sine or cosine function is within the allowable range. If the real part of the argument may or will exceed the range during program execution, then provide code to test for the situation and, if necessary, modify the real part of the argument or bypass the source referencing the function subprogram.

IFY275I CSIN-CCOS /IMAG ARG=/arg(HEX=hex)/, GT 174.673

Explanation: In the subprogram IFYCSSCN (CSIN or CCOS), the absolute value of the imaginary part of the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: If imaginary part > 0, (X is real portion of argument):

For sine, result=∞/2(SIN X + iCOS X).
For cosine, result=∞/2(COS X - iSIN X).

If imaginary part < 0, (X is real portion of argument):

For sine, result=e/2(SIN X - iCOS X).
For cosine, result=e/2(COS X + iSIN X).

Programmer Response: Make sure that the imaginary part of the argument (in radians where 1 radian is equivalent to 57.2957795131^0) to the trigonometric sine or cosine function is within the allowable range. If the imaginary part of the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the imaginary part of the argument or bypass the source referencing the function subprogram.

IFY276I CQEXP REAL ARG=arg, GT 174.673

Explanation: In the subprogram IFYCQEXP (CQEXP), the value of the real part of the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result =e(COS X + iSIN X) where X is the imaginary portion of the argument.

Programmer Response: Make sure that the real part of the argument to the exponential function is within the allowable range. If the real part of the argument may or will exceed the range during program execution, then provide code to test for the situation, and, if necessary, modify the real part of the argument or bypass the source referencing the function subprogram.

IFY277I CQEXP IMAG ARG=arg, ABS VALUE GT PI*2100**

Explanation: In the subprogram IFYCQEXP (CQEXP), the absolute value of the imaginary part of the argument is greater than 2^{100}*pi (2^{100}*pi=.3982418129956973D + 31)

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=e $^{X1+0*i}$.

Programmer Response: Make sure that the imaginary part of the argument to the exponential function is within the allowable range. If the imaginary part of the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the imaginary part of the argument or bypass the source referencing the function subprogram.

IFY278I CQLOG ARGUMENT = 0.0+0.0I

Explanation: In the subprogram IFYCQLOG (CQLOG), the real and imaginary parts of the argument are equal to zero.

Supplemental Data Provided: None.

Standard Corrective Action: Result=-e+0i.

Programmer Response: Make sure that both the real and imaginary parts of the argument do not become zero during program execution, or are not inadvertently initialized or modified to zero. Provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY279I CQSIN-CQCOS /REAL ARG/=/arg/, GE 2100**

Explanation: In the subprogram IFYCQSCN (CQSIN or CQCOS), the absolute value of the real part of the argument is greater than or equal to 2^{100}.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: If the argument is X + iY, for CQSIN, result=0 + DSINH (Y)*i and, for CQCOS, result = DCOSH (Y)+0*i.

Programmer Response: Make sure that the real part of the argument (in radians where 1 radian is equivalent to 57.2957795131^0) to the trigonometric sine or cosine function is within the allowable range. If the part of the argument may or will exceed the range during program execution, then provide code to test for the situation and, if necessary, modify the real part of the argument or bypass the source referencing the function subprogram.

IFY280I CQSIN-CQCOS /IMAG ARG/=/arg/, GT 174.673

Explanation: In the subprogram IFYCQSCN(CQSIN or CQCOS), the absolute value of the imaginary part of the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: If imaginary part > 0, (X is real portion of argument):

For sine, result=•/2(SIN X + iCOS X).
For cosine, result=•/2(COS X - iSIN X).

If imaginary part <0, (X is real portion of argument):

For sine, result=•/2(SIN X - iCOS X).
For cosine, result=•/2(COS X + iSIN X).

Programmer Response: Make sure that the imaginary part of the argument (in radians where 1 radian is equivalent to 57.2957795131^0) to the trigonometric sine or cosine function is within the allowable range. If the imaginary part of the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the imaginary part of the argument or bypass the source referencing the function subprogram.

IFY281I CDEXP REAL ARG=arg(HEX=hex), GT 174.673

Explanation: In the subprogram IFYCLEXP (CDEXP), the value of the real part of the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•(COS X + iSIN X) where X is the imaginary portion of the argument.

Programmer Response: Make sure that the real part of the argument to the exponential function is within the allowable range. If the real part of the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the real part of the argument or bypass the source referencing the function subprogram.

IFY282I CDEXP IMAG ARG=arg(HEX=hex),ABS VALUE GE PI*250**

Explanation: In the subprogram IFYCLEXP(CDEXP), the absolute value of the imaginary part of the argument is greater than or equal to 2^{50}*pi (2^{50}*pi=.3537118876014220ID+16).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=e ^{1}+0*i.

Programmer Response: Make sure that the imaginary part of the argument to the exponential function is within the allowable range. If the imaginary part of the argument may or will exceed that range during program execution, then provide code to test for the situation, and, if necessary, modify the imaginary part of the argument or bypass the source referencing the function subprogram.

IFY283I CDLOG ARGUMENT=0.D0+0.D0I

Explanation: In the subprogram IFYCLLOG (CDLOG), the real and imaginary parts of the argument are equal to zero.

Supplemental Data Provided: None.

Standard Corrective Action: Result=-•+0i.

Programmer Response: Make sure that both the real and imaginary parts of the argument do not become zero during program execution, or are not inadvertently initialized or modified to zero. Provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY284I CDSIN-CDCOS /REAL ARG/ =/arg(HEX=hex)/, GE PI*250**

Explanation: In the subprogram IFYCLSCN (CDSIN or CDCOS), the absolute value of the real part of the argument is greater than or equal to 2^{50}×pi (2^{50}×pi = .3537118876014220ID+16).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: If the argument is X + iY, for CDSIN, the result=0 + DSINH (Y) +i; for CDCOS, the result=DCOSH (Y)+0×i.

Programmer Response: Make sure that the real part of the argument (in radians where 1 radian is equivalent to 57.2957795131^{0}) to the trigonometric sine or cosine function is within the allowable range. If the part of the argument may or will exceed the range during program execution, then provide code to test for the situation, and, if necessary, modify the real part of the argument or bypass the source referencing the function subprogram.

IFY285I CDSIN-CDCOS /IMAG ARG/=/arg(HEX=hex)/, GT 174.673

Explanation: In the subprogram IFYCLSCN (CDSIN or CDCOS), the absolute value of the imaginary part of the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: If imaginary part >0, (X is real portion of argument):

For sine, result=•/2(SIN X + iCOS X).
For cosine, result=•/2(COS X - iSIN X).

If imaginary part <0, (X is real portion of argument):

For sine, result=•/2(SIN X - iCOS X).
For cosine, result=•/2(COS X + iSIN X).

Programmer Response: Make sure that the imaginary part of the argument (in radians where 1 radian is equivalent to 57.2957795131^{0}) to the trigonometric sine or cosine function is within the allowable range. If the imaginary part of the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the imaginary part of the argument or bypass the source

referencing the function subprogram.

IFY286I VSIOS - ATTEMPT TO ISSUE SYNCHRONOUS AND ASYNHRONOUS I/O REQUESTS WITHOUT AN INTERVENING REWIND. FILE filename

Explanation: A data set that has been using one mode of I/O operations (that is, either synchronous or asynchronous) must be rewound before changing modes. An attempt was made to change the mode without rewinding the data set.

Supplemental Data Provided: Filename.

Standard Corrective Action: The I/O request is ignored and execution continues.

Programmer Response: Insert a REWIND statement at an appropriate point in the program.

IFY287I VASYN - A WAIT ISSUED WITH NO OUTSTANDING I/O REQUEST, FILE filename

Explanation: A WAIT statement was issued with no corresponding READ or WRITE request.

Supplemental Data Provided: Filename.

Standard Corrective Action: The WAIT statement is ignored and execution continues.

Programmer Response: Remove the WAIT statement or include a corresponding READ or WRITE statement.

IFY288I VASYN - NO WAIT ISSUED FOR AN OUTSTANDING I/O REQUEST FILE filename

Explanation: No WAIT statement was issued for an outstanding READ or WRITE request.

Supplemental Data Provided: Filename.

Standard Corrective Action: Execution continues with an implied WAIT.

Programmer Response: Include the WAIT statement or remove the READ or WRITE statement.

IFY289I QSQRT ARGUMENT=arg LT ZERO

Explanation: In the subprogram IFYQSQRT (QSQRT#), the argument is less than zero.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result = $|x|^{1/2}$

Programmer Response: Make sure that the argument is within the allowable range. Either modify the argument, or insert source code to test for a negative argument and make the necessary compensation. Bypass the function reference if necessary.

IFY290I GAMMA ARG=arg(HEX=hex),LE 2-252 OR GE 57.5744**

Explanation: In the subprogram IFYSGAMA (GAMMA), the value of the argument is outside the valid range ($2^{-252} < x < 57.5744$).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument to the gamma function is within the allowable range. If the argument may or will be outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY291I ALGAMA ARG=arg(HEX=hex),LE ZERO. OR GE 4.2937*1073**

Explanation: In the subprogram IFYSGAMA (ALGAMA), the value of the argument is outside the valid range ($0<x<4.2937 \times 10^{73}$).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument to the ALGAMA function is within the allowable range. If the argument may or will be outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY292I QEXP ARG=arg, GT 174.673

Explanation: In the subprogram IFYFQXPR (QEXP), the argument is greater than 174.673.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument to the exponential function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY293I QLOG-QLOG10 ARG=arg, LE ZERO

Explanation: In the subprogram IFYQLOG (QLOG and QLOG10), the argument is less than or equal to zero. Because the subprogram is called by an exponential subprogram, this message may also indicate that an attempt has been made to raise a negative base to a real power.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: If X=0, result=-•; if X<0, result=$\log_{10}|X|$ or $\log|X|$.

Programmer Response: Make sure that the argument to the logarithmic function is within the allowable range. If the argument may or will be outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY294I QSIN-QCOS /ARG/=/arg/, GE 2100**

Explanation: In the subprogram IFYQSCN (QSIN and QCOS), the absolute value of the argument is greater than or equal to 2^{100}.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=SQRT(2)/2

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131°) to the trigonometric sine or cosine function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY295I QATAN2 ARGUMENTS = 0.0

Explanation: In subprogram IFYQATN2, when entry name QATAN2 is used, both arguments are equal to zero.

Supplemental Data Provided: None.

Standard Corrective Action: Result=0.

Programmer Response: Make sure that both arguments do not become zero during program execution, or are not inadvertently initialized or modified to zero. Provide code to test for the situation and, if necessary, modify the arguments or bypass the source referencing the function subprogram.

IFY296I QSINH-QCOSH /ARG/=/arg/, GE 175.366

Explanation: In the subprogram IFYQSCNH (QSINH or QCOSH), the absolute value of the argument is greater than (or equal to) 175.366.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: QSINH(X)=±•; QCOSH(X)=•.

Programmer Response: Make sure that the argument to the hyperbolic sine or cosine function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY297I QARSIN-QARCOS /ARG/=/arg/, GT 1

Explanation: In the subprogram IFYQASCN (QARSIN or QARCOS), the absolute value of the argument is greater than 1.

Supplemental Data Provided: Argument specified.

Standard Corrective Action:

```
If X >  1.0 QARCOS(X) = 0;
If X < -1.0 QARCOS(X) = pi;
If X >  1.0 QARSIN = pi/2;
If X < -1.0 DARSIN = -pi/2.
```

Programmer Response: Make sure that the argument to the arcsine or arccosine function is between -1 and +1, inclusive. If the argument may or will fall outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY298I QTAN-QCOTAN /ARG/=/arg/, GE 2100**

Explanation: In the subprogram IFYQTNCT (QTAN or QCOTAN), the absolute value of the argument is greater than or equal to 2^{100}.

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=1.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131^0) to the trigonometric tangent or cotangent function is within the allowable range. If the argument may or will exceed that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY299I QTAN-QCOTAN /ARG/=/arg/, APPROACHES SINGULARITY

Explanation: In the subprogram IFYQTNCT (QTAN or QCOTAN), the argument value is too close to one of the singularities ($\pm pi/2$, $\pm 3pi/2$,... for the tangent; $\pm pi$, $\pm 2pi$,... for the cotangent).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument (in radians where 1 radian is equivalent to 57.2957795131^0) to the trigonometric tangent or cotengent function is within the allowable range. If the argument may or will approach the corresponding singularities for the function during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY300I DGAMMA ARG=arg(HEX=hex),LE 2-252 OR GE 57.5744**

Explanation: In the subprogram IFYLGAMA (DGAMMA), the value of the argument is outside the valid range ($2^{-252}<x<57.5744$).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument to the DGAMMA function is within the allowable range. If the argument may or will be outside the range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY301I DLGAMA ARG=arg(HEX=hex), LE ZERO. OR GE 4.2937*1073**

Explanation: In the subprogram IFYLGAMA (DLGAMA), the value of the argument is outside the valid range ($0<x<4.2937x10^{73}$).

Supplemental Data Provided: Argument specified.

Standard Corrective Action: Result=•.

Programmer Response: Make sure that the argument to the DLGAMA function is within the allowable range. If the argument may or will be outside that range during program execution, then provide code to test for the situation and, if necessary, modify the argument or bypass the source referencing the function subprogram.

IFY900I EXECUTION TERMINATING DUE TO ERROR COUNT FOR ERROR NUMBER nnnn

Explanation: This error has occurred frequently enough to reach the count specified as the number at which execution should be terminated.

Supplemental Data Provided: Error number.

Standard Corrective Action: No corrective action is implemented.

System Action: The job step is terminated with a completion code of 16.

Programmer Response: Make sure that occurrences of the error number indicated are eliminated.

IFY901I EXECUTION TERMINATING DUE TO SECONDARY ENTRY TO ERROR MONITOR FOR ERROR NUMBER nnnn WHILE PROCESSING ERROR NUMBER nnnn

Explanation: In a user's corrective action routine, an error has occurred that has called the error monitor before it has returned from processing a previously diagnosed error.

Supplemental Data Provided: Error numbers.

Standard Corrective Action: No corrective action is attempted.

System Action: The job step is terminated with a completion code of 16.

Note: If a traceback follows this message, it may be unreliable.

Programmer Response: Make sure that the error monitor is not called prior to processing the diagnosed error.

Example: A statement such as R=A**B (where A and B are REAL*4) cannot be used in the exit routine for error 252, because FRXPR# uses EXP, which detects error 252.

Refer to VS FORTRAN Application Programming Language Reference for information on the error handling subroutines.

IFY902I ERROR NUMBER nnnn OUT OF RANGE OF ERROR TABLE

Explanation: A request has been made to reference a non-existent Option Table entry.

Supplemental Data Provided: Error number.

System Action: The request is ignored and execution continues. IRETCD is set to 0.

Programmer Response: Make sure that the value assigned to an error condition is within the range of entries in the option table.

IFY903I VMOPT - ATTEMPT TO CHANGE UNMODIFIABLE TABLE ENTRY. MESSAGE NUMBER=nnnn

Explanation: The Option Table specifies that no changes may be made in this entry, but a change request has been made by use of CALL ERRSET or CALL ERRSTR.

Refer to VS FORTRAN Application Programming Language Reference for information on the error handling subroutines.

Supplemental Data Provided: Message number.

System Action: The request is ignored and execution continues.

Programmer Response: Make sure that no attempt has been made to alter dynamically an unmodifiable entry in the Option Table.

IFY904I ATTEMPT TO DO I/O DURING FIXUP ROUTINE FOR AN I/O TYPE ERROR, FILE filename

Explanation: When attempting to correct an I/O error, the user may not issue a READ, WRITE, BACKSPACE, ENDFILE, REWIND, CALL PDUMP, CALL DUMP, DEBUG, or CALL ERRTRA.

Refer to VS FORTRAN Application Programming Language Reference for information on the error handling subroutines.

Supplemental Data Provided: Filename.

System Action: The job step is terminated with a completion code of 16.

Programmer Response: Make sure that, if an I/O error is detected, the user exit routine does not attempt to execute any FORTRAN I/O statement.

OPERATOR MESSAGES

Operator messages for PAUSE and STOP statements may be generated during load module execution as follows:

yy IFY001A PAUSE x

Explanation: A FORTRAN PAUSE statement has been executed. The yy is an identification number assigned to the message by the operating system. The x can be:

- An unsigned 1- to 5-digit integer constant specified in the PAUSE statement.
- A literal constant specified in the PAUSE statement.
- A zero to indicate that the PAUSE statement contained no constant.

System Action: The program enters the wait state.

Operator Response: Follow the instructions given by the programmer when the program was submitted for execution; these instructions should indicate the action to be taken for any constant printed in the message text or for a PAUSE statement without a constant.

To resume execution, reply to the outstanding console message after performing the operations requested.

IFY002A STOP x

Explanation: A FORTRAN STOP statement has been executed. The x can be an unsigned 1- to 5-digit nonzero integer constant specified in the STOP statement.

System Action: The STOP statement caused the program to terminate.

Operator Response: None.

APPENDIX D. MODULE NAMES

Entry Name	Module Name
CHAR	IFYCITFN
ICHAR	IFYCITFN
INDEX	IFYINDEX
LEN	IFYCITFN
LGE	IFYLXCMP
LGT	IFYLXCMP
LLE	IFYLXCMP
LLT	IFYLXCMP

Figure 22. Character Subprogram Module Names

OS	CMS	DOS
IFYVCOMH	IFYVCOMH	IFYVCOMH
IFYVSIOS	IFYVSIOS	IFYDSIOS
IFYVDIOS	IFYVDIOS	IFYDDIOS
IFYVIIOS	IFYVIIOS	IFYVIIOS
IFYVVIOS	IFYCVIOS	IFYDVIOS
IFYVCVTH	IFYVCVTH	IFYVCVTH
IFYVCONI	IFYVCONI	IFYVCONI
IFYVCONO	IFYVCONO	IFYVCONO
IFYVTEN	IFYVTEN	IFYVTEN
IFYVERRM	IFYVERRM	IFYVERRM
IFYVERRE	IFYVERRE	IFYVERRE
IFYVTRCH	IFYVTRCH	IFYVTRCH

Figure 23. Reentrant Library Module Names

Entry Name	Module Name	Entry Name	Module Name
ACOS	IFYSASEN	DSIN	IFYLSIN
AINT	IFYFAINT	DSINH	IFYLSCNH
ALGAMA	IFYSGAMA	DSQRT	IFYLSQRT
ALOG	IFYSLGC	DTAN	IFYLTNCT
ALOG10	IFYSLGC	DTANH	IFYLTANH
AMAX0	IFYFMAXI		
AMAX1	IFYFMAXR	EXP	IFYSEXP
AMIN0	IFYFMAXI	ERF	IFYSERF
AMIN1	IFYFMAXR	ERFC	IFYSERF
AMOD	IFYFMODR		
ARCOS	IFYSASCN	FCDXI#	IFYFCDXI
ARSIN	IFYSASCN	FCQXI#	IFYFCQXI
ASIN	IFYSASCN	FCXPI#	IFYFCXPI
ATAN	IFYSATN2	FDXPD#	IFYFDXPD
ATAN2	IFYSATN2	FDXPI#	IFYFDXPI
		FIXPI#	IFYFIXPI
CABS	IFYCSABS	FQXPI#	IFYFQXPI
CCOS	IFYCSSCN	FQXPQ#	IFYFQXPQ
CDABS	IFYCLABS	FQXP2#	IFYFQXPQ
CDCOS	IFYCLSCN	FRXPI#	IFYFRXPI
CDDVD#	IFYCLAD	FRXPR#	IFYFRXPR
CDEXP	IFYCLEXP		
CDLOG	IFYCLLOG	GAMMA	IFYSGAMA
CDMPY#	IFYCLAM		
CDSIN	IFYCLSCN	IDINT	IFYFIFIX
CDSQRT	IFYCLSQT	IFIX	IFYFIFIX
CDVD#	IFYCSAD	INT	IFYFIFIX
CEXP	IFYCSEXP		
CLOG	IFYCSLOG	LGAMMA	IFYSGAMA
CMPY#	IFYCSAM	LOG	IFYSLGN
COS	IFYSCOS	LOG10	IFYSLGN
COSH	IFYSSCNH		
COTAN	IFYSTNCT	MAX0	IFYFMAXI
CQABS	IFYCQABS	MAXI	IFYFMAXR
CQCOS	IFYCQSCN	MIN0	IFYFMAXI
CQDVD#	IFYCQRIT	MIN1	IFYFMAXR
CQEXP	IFYCQEXP	MOD	IFYFMODI
CQLOG	IFYCQLOG		
CQMPY#	IFYCQRIT	QARCOS	IFYQASCN
CQSIN.	IFYCQSCN	QARSIN	IFYQASCN
CQSQRT	IFYCQSQT	QATAN	IFYQATN2
CSIN	IFYCSSCN	QATAN2	IFYQATN2
CSQRT	IFYCSSQT	QCOS	IFYQSCN
		QCOSH	IFYQSCNH
DARCOS	IFYLASCN	QCOTAN	IFYQTNCT
DARSIN	IFYLASCN	QERF	IFYQERF
DATAN	IFYLATN2	QERFC	IFYQERF
DATAN2	IFYLATN2	QEXP	IFYFQXPQ
DCOS	IFYLCOS	QLOG	IFYFQXPQ
DCOSH	IFYLSCNH	QLOG10	IFYFQXPQ
DCOTAN	IFYLTNCT	QSIN	IFYQSCN
DERF	IFYLERF	QSINH	IFYQSCNH
DERFC	IFYLERF	QSQRT	IFYQSQRT
DEXP	IFYLEXP	QTAN	IFYQTNCT
DGAMMA	IFYLGAMA	QTANH	IFYQTANH
DLGAMA	IFYLGAMA		
DLOG	IFYLLGN	SIN	IFYSSIN
DLOG10	IFYLLGC	SINH	IFYSSCNH
DMAX1	IFYFMAXD	SQRT	IFYSSQRT
DMIN1	IFYFMAXD		
DMOD	IFYFMODR	TAN	IFYSTNCT
		TANH	IFYSTANH

Figure 24. Mathematical Subprogram Module Names

Appendix E: Sample Storage Printouts

A sample printout is given below for each dump format that can be specified for the storage dump subprogram. The printouts are given in the following order: hexadecimal, LOGICAL *1, LOGICAL *4, INTEGER *2, INTEGER *4, REAL *4, REAL *8, COMPLEX *8, COMPLEX *16, and literal (see Figure 25). Note that the headings on the printouts are not generated by the system, but were obtained by using FORMAT statements. The number printed at the left of each output line is the storage location (in hexadecimal) of the first data item tabulated.

The output of the storage dump subprogram (for entry names DUMP, CPDUMP, and PDUMP) is placed on the object error unit data set defined by the installation during system generation.

```
CALL PDUMP WITH HEXADECIMAL FORMAT SPECIFIED
00A3E0     485F5E10  00000000  485F5E10  10000000  42100000
006DC8     42800000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000  00000000
006DF8     C0000000  00000000  41200000  41566666  0000000C  41100000
CALL PDUMP WITH LOGICAL*1 FORMAT SPECIFIED
006E1E     T  F
CALL PDUMP WITH LOGICAL*4 FORMAT SPECIFIED
006E10     F  T
CALL PDUMP WITH INTEGER*2 FORMAT SPECIFIED
006E18       10
006E1A      -100
006E1C       10
CALL PDUMP WITH INTEGER*4 FORMAT SPECIFIED
006E20            1          2          3          4          5          6          7          8          9         10
006E48           11         12
CALL PDUMP WITH REAL*4 FORMAT SPECIFIED
006E00      0.20000000E 01   0.53999996E 01
CALL PDUMP WITH REAL*8 FORMAT SPECIFIED
006DC8      0.17599999999999990 03
CALL PDUMP WITH COMPLEX*8 FORMAT SPECIFIED
006DD0               (3.0000000,4.0000000)             (4.0000000,8.0000000)
CALL PDUMP WITH COMPLEX*16 FORMAT SPECIFIED
006DE0            (0.99999999999999990,0.99999999999999990)         (-0.99999999999999990,-0.99999999999999990)
CALL PDUMP WITH LITERAL FORMAT SPECIFIED
006E5C     THIS ARRAY CONTAINS ALPHAMERIC DATA
CALL CPDUMP
008990     FILE READ ARGUMENT
```

Figure 25. Sample Storage Printouts

A

C

D

E

R

S

T

U

VS FORTRAN Application Programming Library Reference (File No. S370-25) Printed in U.S.A. SC26-3989-0

SC26-3989-0

VS FORTRAN Application
Programming Library Reference
SC26-3989-0

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

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

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

List TNLs here:

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

Last TNL _____

Previous TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

SC26-3989-0

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 Application
Programming Library Reference
SC26-3989-0

Reader's Comment Form

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. This form may be used to communicate your views about this publication. They will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

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

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

List TNLs here:

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

Last TNL _____

Previous TNL _____

Previous TNL _____

Fold on two lines, tape, and mail. No postage necessary if mailed in the U.S.A. (Elsewhere, any IBM representative will be happy to forward your comments.) Thank you for your cooperation.

SC26-3989-0

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 Application Programming Library: ...rence (File No. S370-25) Printed in U.S.A. SC26-3989-0

Technical Newsletter

This Newsletter No.	SN26-0852
Date	3 June 1981
Base Publication No.	SC26-3989-0
File No.	S370-25
Prerequisite Newsletters	None

VS FORTRAN Application Programming: Library Reference

This technical newsletter, a part of Release 1 of VS FORTRAN, Program Products 5748-FO3 (Compiler and Library) and 5748-LM3 (Library only), provides new and replacement pages for the subject publication. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and removed are:

cover-vi.2 (vi.1, vi.2 added)
13,14
23-26
85-96.4 (96.1-96.4 added)
101-112.1 (112.1 added)
131-134

Each technical change is marked by a vertical line to the left of the change.

Summary of Amendments

Changes included in this newsletter are summarized under "Summary of Amendments" following the preface.

Note: Please file this cover letter at the back of the publication to provide a record of changes.

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

Printed in U.S.A.