File Number S360-25
Form C28-6596-4

OS
44PS
DOS

IBM Systems Reference Library

IBM System/360

FORTRAN IV Library Subprograms

Program Number 360S-LM-501
360F-LM-619
360N-LM-480

This publication describes the library subprograms supplied with Basic FORTRAN IV (E) and FORTRAN IV (G, H, DOS, and MODEL 44) and tells how to use the subprograms in either a FORTRAN or an assembler language program.

Preface

The purpose of this publication is to describe the FORTRAN library subprograms and their use in either a FORTRAN or an assembler language program. The body of the publication describes the mathematical subprograms (which perform computations) and the service subprograms (which perform testing and utility functions). This information is intended primarily for the FORTRAN programmer; Appendix E is intended for the assembler language programmer. Additional appendixes contain algorithms (the method by which a mathematical function is computed), performance statistics, descriptions of interruption and error procedures, storage estimates, and sample storage printouts.

The reader should be familiar with one of the following publications:

IBM System/360 FORTRAN IV Language, Form C28-6515

IBM System/360 Basic FORTRAN IV Language, Form C28-6629

IBM System/360 Operating System: Assembler Language, Form C28-6514

IBM System/360 Model 44 Programming System: Assembler Language, Form C28-6811

IBM System/360 Disk and Tape Operating Systems: Assembler Language, Form C24-3414

In addition, references are made within this publication to information contained in the following publications:

IBM System/360 Principles of Operation, Form A22-6521

IBM System/360 Operating Systems Supervisor and Data Management Macro-Instructions, Form C28-6647

IBM System/360 Model 44 Programming System: Guide to System Use, Form C28-6812

Standard mathematical notation is used in this publication. The reader is expected to be familiar with this notation and with common mathematical terminology.

Appendix E provides information about the use of these library routines by assembler language programmers. Corresponding information for DOS is in the publication *IBM System/360 Disk Operating System, FORTRAN IV Programmer's Guide*, Form C28-6397.

FIFTH EDITION (October 1968)

This is a revision of, and makes obsolete, C28-6596-3. Additions have been made to support the IBM System/360 Disk Operating System. All changes to the text or tables are indicated by a vertical line to the left of the change; revised illustrations are denoted by a • to the left of the caption.

Specifications contained herein are periodically subject to change; any such changes will be reported in subsequent revisions or Technical Newsletters. Before using this publication in connection with the operation of IBM systems, refer to the latest SRL Newsletter, Form N20-0360 for the editions that are applicable and current.

A form for readers' comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Systems Publications, 1271 Avenue of the Americas, New York, N.Y. 10020.

Contents

Illustrations

Tables

Figures

The FORTRAN IV library for the System/360 Operating System, the Model 44 Programming System and the Disk Operating System comprises two types of relocatable subprograms: mathematical subprograms and service subprograms. The mathematical subprograms correspond to a subprogram defined by a FUNCTION statement in a FORTRAN source module. These subprograms always return one answer (function value) to the calling module. The service subprograms correspond to a subprogram defined by a SUBROUTINE statement in a FORTRAN source module. These subprograms may or may not return a value to the calling module.

Calls to the library subprograms are either at the programmer's request or in response to program requirements. Under the System/360 Operating System, all calls are processed by the linkage editor, which takes the subprograms from the library. The library subprograms are then combined by the linkage editor with the calling module (either an object or a load module) into another load module which is ready for execution. Under the Model 44 Programming System and the Disk Operating System, the linkage editor takes the subprograms from the library and combines them with the calling module into an executable phase.

The library subprograms may be called in either a FORTRAN or an assembler language program. The next two sections of this publication contain calling information for the FORTRAN programmer; Appendix E contains calling information for the assembler language programmer.

Subprogram Names

Except for a three-character prefix, the names of subprograms described in this publication are the same for all systems. The prefix is uniformly IHC for the Operating System library, BOA for the Model 44 Programming System, and ILF for the Disk Operating System. These characters occupy the portion of the subprogram name shown as *xxx* in the text and tables. Thus, the subprogram named here as XXXCSLOG has the name IHCCSLOG under the Operating System, BOACSLOG under the Model 44 Programming System, and ILFCSLOG under the Disk Operating System.

Mathematical Subprograms

The mathematical subprograms supplied in the FORTRAN library perform computations frequently needed by the applications programmer. The mathematical subprograms are called in two ways: explicitly, when the programmer includes the appropriate entry name in a source language statement (see Table 1); and implicitly, when certain notation (e.g., raising a number to a power) appears within a source language statement (see Table 6).

The following text describes the individual mathematical subprograms and explains their use in a FORTRAN program. Detailed information about the actual method of computation used in each subprogram, the performance of the subprogram, interruption and error procedures, and storage estimates can be found in the appendixes of this publication.

Explicitly Called Subprograms

Each explicitly called subprogram performs one or more mathematical functions. Each mathematical function is identified by a unique entry name that differs from the name of the subprogram.

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

For example, the source statement

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

causes the IHCSSCN 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 IHCSSQRT 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 the tables that make up the rest of this section. These tables show the general function, subprogram name, the FORTRAN library that contains the subprogram, definition, entry name(s), argument information, type of function value returned, and assembler requirements. The following column headings are used in the tables:

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

Subprogram Name: This column gives the module name of the subprogram. The first three characters are uniformly IHC for subprograms in the System/360 Operating System library, BOA for those in the Model 44 Programming System library, and ILF for those in the Disk Operating System library. These three characters occupy the portion of the subprogram name that is shown as *xxx* in the text and tables.

Subset: This column indicates those subprograms that belong to the Basic FORTRAN IV (E) library. Unless otherwise indicated, all subprograms that belong to the E library also belong to the FORTRAN IV (G, H, MODEL 44, and DOS) library.

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 $y = x^{1/2}$.) The definition for those subprograms that accept complex arguments contains the notation $z = x_1 + x_2 i$.

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 upon the computation to be performed. For example, the IHCSSCN subprogram has two entry names: COS and SIN. If the cosine is to be computed, entry name COS is used; if the sine is to be computed, entry name SIN is used.

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

Argument Type: This column describes the mode and length of the argument. INTEGER, REAL, and COMPLEX represent the type of number; the notation *4, *8, and *16 represent the size of the argument in storage locations.

NOTE: In Basic FORTRAN IV, a *real* argument corresponds to the REAL *4 argument, and a *double-precision* argument corresponds to the REAL *8 argument. Complex arguments cannot be used with a Basic FORTRAN IV compiler.

Argument Range: This column gives the valid range for an argument. If the argument is not within this range, an error message is issued and execution of this load module is terminated. Appendix C contains a description of the error messages.

Function Value Type: This column describes the type of function value returned by the subprogram. The notation used is the same as that used for the argument type.

Table 1. Explicitly Called Mathematical Subprograms

General Function	Specific Function	Subprogram Name	Entry Name(s)
Logarithmic and exponential subprograms (described in Table 2)	Common and natural logarithm	xxxCLLOG* xxxCSLOG* xxxLLOG xxxSLOG	CDLOG CLOG DLOG, DLOG10 ALOG, ALOG10
	Exponential	xxxCLEXP* xxxCSEXP* xxxLEXP xxxSEXP	CDEXP CEXP DEXP EXP
	Square root	xxxCLSQT xxxCSSQT* xxxLSQRT xxxSSQRT	CDSQRT CSQRT DSQRT SQRT
Trigonometric subprograms (described in Table 3)	Arcsine and arccosine	xxxLASCN* xxxSASCN*	DARSIN, DARCOS ARSIN, ARCOS
	Arctangent	IHCLATAN xxxLATN2* IHCSATAN xxxSATN2*	DATAN DATAN, DATAN2 ATAN ATAN, ATAN2
	Sine and cosine	xxxCLSCN* xxxCSSCN* xxxLSCN xxxSSCN	CDSIN, CDCOS CSIN, CCOS DSIN, DCOS SIN, COS
	Tangent and cotangent	xxxLTNCT* xxxSTNCT*	DTAN, DCOTAN TAN, COTAN
Hyperbolic function subprograms (described in Table 4)	Hyperbolic sine and cosine	xxxLSCNH* xxxSSCNH*	DSINH, DCOSH SINH, COSH
	Hyperbolic tangent	xxxLTANH xxxSTANH	DTANH TANH
Miscellaneous subprograms (described in Table 5)	Absolute value	xxxCLABS* xxxCSABS*	CDABS CABS
	Error function	xxxLERF* xxxSERF*	DERF, DERFC ERF, ERFC
	Gamma and log-gamma	xxxLGAMA* xxxSGAMA*	DGAMMA, DLGAMA GAMMA, ALGAMA
	Maximum and minimum value	xxxFMAXD xxxFMAXI xxxFMAXR	DMAX1, DMIN1 AMAX0, AMIN0, MAX0, MIN0 AMAX1, AMIN1, MAX1, MIN1
	Modular arithmetic	IHCFMODI IHCFMODR	MOD AMOD, DMOD
	Truncation	IHCFAINT IHCFIFIX	AINT IDINT, INT

*Not available in FORTRAN IV (E)

Assembler Requirements: This column gives the registers used by the subprogram and the minimum save area that the assembler language programmer must supply. For example, the assembler requirements for the *xxx*CSSQT subprogram are:

registers 0, 2(4)
save area 9F

This information specifies that:

1. The function value is found in floating-point registers 0 and 2.
2. Floating-point register 4 is used for intermediate computation.
3. The save area must be at least nine full-words in length.

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

NOTE: In the following tables, the approximate value of $2^{18} \cdot \pi$ is .82354966406249996D+06; the approximate value of $2^{50} \cdot \pi$ is .35371188737802239D+16.

• Table 2. Logarithmic and Exponential Subprograms

General Function	Subprogram Name	Sub-set	Definition	Entry Name	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1]	Assembler Require-ments
Common and natural logarithm	xxxCLLOG	No	y=PV $\log_e$ (z) See Note 2	CDLOG	1	complex *16	$z \neq 0 + 0i$ See Note 3	complex *16	registers 0, 2 save area 8F
	xxxCSLOG	No	y=PV $\log_e$ (z) See Note 2	CLOG	1	complex *8	$z \neq 0 + 0i$ See Note 3	complex *8	registers 0, 2 save area 8F
	xxxLLOG	Yes	$y=\log_e x$ or $y=\ln x$	DLOG	1	real *8	$x > 0$	real *8	registers 0 (2) save area 9F
			$y=\log_{10} x$	DLOG10	1	real *8	$x > 0$	real *8	registers 0 (2) save area 9F
	xxxSLOG	Yes	$y=\log_e x$ or $y=\ln x$	ALOG	1	real *4	$x > 0$	real *4	registers 0 (2) save area 5F†
			$y=\log_{10} x$	ALOG10	1	real *4	$x > 0$	real *4	registers 0 (2) save area 5F†
Exponential	xxxCLEXP	No	$y=e^z$ See Note 4	CDEXP	1	complex *16	$x_1 \leqq 174.673$ $\lvert x_2 \rvert < (2^{50} \cdot \pi)$	complex *16	registers 0, 2 save area 8F
	xxxCSEXP	No	$y=e^z$ See Note 4	CEXP	1	complex *8	$x_1 \leqq 174.673$ $\lvert x_2 \rvert < (2^{50} \cdot \pi)$	complex *8	registers 0, 2 save area 8F
	xxxLEXP	Yes	$y=e^x$	DEXP	1	real *8	$x \leqq 174.673$	real *8	registers 0 (2) save area 9F
	xxxSEXP	Yes	$y=e^x$	EXP	1	real *4	$x \leqq 174.673$	real *4	register 0 save area 12F
Square root	xxxCLSQT	No	$y=\sqrt{z}$ or $y=z^{1/2}$	CDSQRT	1	complex *16	any complex argument See Note 3	complex *16	registers 0, 2 (4) save area 9F
	xxxCSSQT	No	$y=\sqrt{z}$ or $y=z^{1/2}$	CSQRT	1	complex *8	any complex argument See Note 3	complex *8	registers 0, 2 (4) save area 9F
	xxxLSQRT	Yes	$y=\sqrt{x}$ or $y=x^{1/2}$	DSQRT	1	real *8	$x \geqq 0$	real *8	registers 0 (2, 4) save area 5F†
	xxxSSQRT	Yes	$y=\sqrt{x}$ or $y=x^{1/2}$	SQRT	1	real *4	$x \geqq 0$	real *4	registers 0 (4) save area 5F†

NOTES:
1. In FORTRAN IV (E), a real argument corresponds to the REAL *4 argument, and a double-precision argument corresponds to the REAL *8 argument.
2. PV = principal value. The answer given is from that point where the 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. Floating-point overflow can occur.
4. Where z is a complex number of the form x_1+x_2i.

†For OS/360, seven words are needed.

Table 3. Trigonometric Subprograms

General Function	Subprogram Name	Subset	Definition	Entry Name	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1]	Assembler Requirements
Arcsine and arccosine	xxxLASCN	No	$y = \arcsin(x)$	DARSIN	1	real *8	$\|x\| \leq 1$	real *8 (in radians)	registers 0 (2, 4) save area 13F
			$y = \arccos(x)$	DARCOS	1	real *8	$\|x\| \leq 1$	real *8 (in radians)	registers 0 (2, 4) save area 13F
	xxxSASCN	No	$y = \arcsin(x)$	ARSIN	1	real *4	$\|x\| \leq 1$	real *4 (in radians)	registers 0 (2, 4) save area 10F
			$y = \arccos(x)$	ARCOS	1	real *4	$\|x\| \leq 1$	real *4 (in radians)	registers 0 (2, 4) save area 10F
Arctangent	IHCLATAN	See Note 2	$y = \arctan(x)$	DATAN	1	real *8	any real argument	real *8 (in radians)	registers 0 (2, 4, 6) save area 5F
	xxxLATN2	See Note 2	$y = \arctan(x)$	DATAN	1	real *8	any real argument	real *8 (in radians)	registers 0 (2, 4, 6) save area 5F†
			$y = \arctan\left(\frac{x_1}{x_2}\right)$	DATAN2	2	real *8	any real arguments (except 0, 0)	real *8 (in radians)	registers 0 (2, 4, 6) save area 5F†
	IHCSATAN	See Note 2	$y = \arctan(x)$	ATAN	1	real *4	any real argument	real *4 (in radians)	registers 0 (2, 4, 6) save area 5F
	xxxSATN2	See Note 2	$y = \arctan(x)$	ATAN	1	real *4	any real argument	real *4 (in radians)	registers 0 (2, 4, 6) save area 5F†
			$y = \arctan\left(\frac{x_1}{x_2}\right)$	ATAN2	2	real *4	any real arguments (except 0, 0)	real *4 (in radians)	registers 0 (2, 4, 6) save area 5F†
Sine and cosine	xxxCLSCN	No	$y = \sin(z)$ See Note 4	CDSIN	1	complex *16 (in radians)	$\|x_1\| < (2^{50} \cdot \pi)$ $\|x_2\| \leq 174.673$	complex *16	registers 0, 2 (4) save area 9F
			$y = \cos(z)$ See Note 4	CDCOS	1	complex *16 (in radians)	$\|x_1\| < (2^{50} \cdot \pi)$ $\|x_2\| \leq 174.673$	complex *16	registers 0, 2 (4) save area 9F
	xxxCSSCN	No	$y = \sin(z)$ See Note 4	CSIN	1	complex *8 (in radians)	$\|x_1\| < (2^{18} \cdot \pi)$ $\|x_2\| \leq 174.673$	complex *8	registers 0, 2 (4) save area 9F
			$y = \cos(z)$ See Note 4	CCOS	1	complex *8 (in radians)	$\|x_1\| < (2^{18} \cdot \pi)$ $\|x_2\| \leq 174.673$	complex *8	registers 0, 2 (4) save area 9F

Table 3. Trigonometric Subprograms (Continued)

General Function	Subprogram Name	Sub-set	Definition	Entry Name	Argument(s)			Function Value Type[1]	Assembler Require-ments
					No.	Type[1]	Range		
Sine and cosine (continued)	xxxLSCN	Yes	y = sin (x)	DSIN	1	real *8 (in radians)	$\|x\|<(2^{50}\cdot\pi)$	real *8	registers 0 (2, 4) save area 5F†
			y = cos (x)	DCOS	1	real *8 (in radians)	$\|x\|<(2^{50}\cdot\pi)$	real *8	registers 0 (2, 4) save area 5F†
	xxxSSCN	Yes	y = sin (x)	SIN	1	real *4 (in radians)	$\|x\|<(2^{18}\cdot\pi)$	real *4	registers 0 (2, 4) save area 5F†
			y = cos (x)	COS	1	real *4 (in radians)	$\|x\|<(2^{18}\cdot\pi)$	real *4	registers 0 (2, 4) save area 5F†
Tangent and cotangent	xxxLTNCT	No	y = tan (x)	DTAN	1	real *8 (in radians)	$\|x\|<(2^{50}\cdot\pi)$ See Note 3	real *8	registers 0 (2, 4, 6) save area 5F†
			y = cotan (x)	DCOTAN	1	real *8 (in radians)	$\|x\|<(2^{50}\cdot\pi)$ See Note 3	real *8	registers 0 (2, 4, 6) save area 5F†
	xxxSTNCT	No	y = tan (x)	TAN	1	real *4 (in radians)	$\|x\|<(2^{18}\cdot\pi)$ See Note 3	real *4	registers 0 (2, 4) save area 5F
			y = cotan (x)	COTAN	1	real *4 (in radians)	$\|x\|<(2^{18}\cdot\pi)$ See Note 3	real *4	registers 0 (2, 4) save area 5F

NOTES:

1. In FORTRAN IV (E), a real argument corresponds to the REAL *4 argument, and a double-precision argument corresponds to the REAL *8 argument.
2. Instead of the IHCLATAN and IHCSATAN subprograms contained in the FORTRAN IV (E) library, the FORTRAN IV library contains the xxxLATN2 and xxxSATN2 subprograms.
3. The argument for the cotangent functions may not be near a multiple of π; the argument for the tangent functions may not be near an odd multiple of $\pi/2$.
4. Where z is a complex number of the form x_1+x_2i.

†For OS/360, seven words are needed.

Table 4. Hyperbolic Function Subprograms

General Function	Subprogram Name	Subset	Definition	Entry Name	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1]	Assembler Requirements
Hyperbolic sine and cosine	xxxLSCNH	No	$y=\frac{e^x-e^{-x}}{2}$	DSINH	1	real *8	$\|x\|<174.673$	real *8	registers 0 (2, 4) save area 9F
			$y=\frac{e^x+e^{-x}}{2}$	DCOSH	1	real *8	$\|x\|<174.673$	real *8	registers 0 (2, 4) save area 9F
	xxxSSCNH	No	$y=\frac{e^x-e^{-x}}{2}$	SINH	1	real *4	$\|x\|<174.673$	real *4	registers 0 (2, 4) save area 9F
			$y=\frac{e^x+e^{-x}}{2}$	COSH	1	real *4	$\|x\|<174.673$	real *4	registers 0 (2, 4) save area 9F
Hyperbolic tangent	xxxLTANH	Yes	$y=\frac{e^x-e^{-x}}{e^x+e^{-x}}$	DTANH	1	real *8	any real argument	real *8	registers 0 (2) save area 5F
	xxxSTANH	Yes	$y=\frac{e^x-e^{-x}}{e^x+e^{-x}}$	TANH	1	real *4	any real argument	real *4	registers 0 (2) save area 5F

NOTE:
1. In FORTRAN IV (E), a real argument corresponds to the REAL *4 argument, and a double-precision argument corresponds to the REAL *8 argument.

Table 5. Miscellaneous Mathematical Subprograms

General Function	Subprogram Name	Subset	Definition	Entry Name	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1]	Assembler Requirements
Absolute value	xxxCLABS	No	$y=\|z\|=(x_1^2+x_2^2)^{1/2}$	CDABS	1	complex *16	any complex argument See Note 2	real *8	registers 0, 2 (4) save area 8F
	xxxCSABS	No	$y=\|z\|=(x_1^2+x_2^2)^{1/2}$	CABS	1	complex *8	any complex argument See Note 2	real *4	registers 0, 2 (4) save area 8F
Error function	xxxLERF	No	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}\,du$	DERF	1	real *8	any real argument	real *8	registers 0 (2, 4, 6) save area 11F
			$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}\,du$ $y=1-\text{erf}(x)$	DERFC	1	real *8	any real argument	real *8	registers 0 (2, 4, 6) save area 11F

Table 5. Miscellaneous Mathematical Subprograms (Continued)

General Function	Subprogram Name	Subset	Definition	Entry Name	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1]	Assembler Requirements
Error function (continued)	xxxSERF	No	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}\,du$	ERF	1	real *4	any real argument	real *4	registers 0 (2, 4, 6) save area 11F
			$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}\,du$ $y=1-\text{erf}(x)$	ERFC	1	real *4	any real argument	real *4	registers 0 (2, 4, 6) save area 11F
Gamma and log-gamma	xxxLGAMA	No	$y=\int_0^\infty u^{x-1}e^{-u}\,du$	DGAMMA	1	real *8	$x > 2^{-252}$ and $x < 57.5744$	real *8	registers 0 (2, 4, 6) save area 11F
			$y=\log_e \Gamma(x)$ or $y=\log_e \int_0^\infty u^{x-1}e^{-u}\,du$	DLGAMA	1	real *8	$x > 0$ and $x < 4.2913 \cdot 10^{73}$	real *8	registers 0 (2, 4, 6) save area 11F
	xxxSGAMA	No	$y=\int_0^\infty u^{x-1}e^{-u}\,du$	GAMMA	1	real *4	$x > 2^{-252}$ and $x < 57.5744$	real *4	registers 0 (2, 4, 6) save area 11F
			$y=\log_e \Gamma(x)$ or $y=\log_e \int_0^\infty u^{x-1}e^{-u}\,du$	ALGAMA	1	real *4	$x > 0$ and $x < 4.2913 \cdot 10^{73}$	real *4	registers 0 (2, 4, 6) save area 11F
Maximum and minimum values	xxxFMAXD	Yes	$y=\max(x_1, \ldots, x_n)$	DMAX1	≥ 2	real *8	any real arguments	real *8	register 0 save area 9F
			$y=\min(x_1, \ldots, x_n)$	DMIN1	≥ 2	real *8	any real arguments	real *8	register 0 save area 9F
	xxxFMAXI	Yes	$y=\max(x_1, \ldots, x_n)$	AMAX0	≥ 2	integer *4	any integer arguments	real *4	register 0 save area 9F
				MAX0	≥ 2	integer *4	any integer arguments	integer *4	register See Note 3 save area 9F
			$y=\min(x_1, \ldots, x_n)$	AMIN0	≥ 2	integer *4	any integer arguments	real *4	register 0 save area 9F
				MIN0	≥ 2	integer *4	any integer arguments	integer *4	register See Note 3 save area 9F

Table 5. Miscellaneous Mathematical Subprograms (Continued)

General Function	Subprogram Name	Subset	Definition	Entry Name	Argument(s) No.	Argument(s) Type[1]	Argument(s) Range	Function Value Type[1]	Assembler Requirements
Maximum and Minimum Values (continued)	xxxFMAXR	Yes	$y=\max(x_1,\ldots,x_n)$	AMAX1	≥ 2	real *4	any real arguments	real *4	register 0 save area 9F
				MAX1	≥ 2	real *4	any real arguments	integer *4	register See Note 3 save area 9F
			$y=\min(x_1,\ldots,x_n)$	AMIN1	≥ 2	real *4	any real arguments	real *4	register 0 save area 9F
				MIN1	≥ 2	real *4	any real arguments	integer *4	register See Note 3 save area 9F
Modular arithmetic	IHCFMODI	See Note 4	$y = x_1$ (modulo x_2) See Note 5	MOD	2	integer	$x_2 \neq 0$ See Note 6	integer *4	register See Note 3 save area 9F
	IHCFMODR	See Note 4	$y = x_1$ (modulo x_2) See Note 5	AMOD	2	real *4	$x_2 \neq 0$ See Note 6	real *4	register 0 save area 9F
			$y = x_1$ (modulo x_2) See Note 5	DMOD	2	real *8	$x_2 \neq 0$ See Note 6	real *8	register 0 save area 9F
Truncation	IHCFAINT	See Note 4	y = (sign x) · n where n is the largest integer $\leq \lvert x \rvert$	AINT	1	real *4	any real argument	real *4	register 0 save area 9F
	IHCFIFIX	See Note 4	y = (sign x) · n where n is the largest integer $\leq \lvert x \rvert$	IDINT	1	real *8	any real argument	integer	register 0 See Note 3 save area 9F
				INT	1	real *4	any real argument	integer	register See Note 3 save area 9F

NOTES:

1. In FORTRAN IV (E), a real argument corresponds to the REAL *4 argument, and a double-precision argument corresponds to the REAL *8 argument.
2. Floating-point overflow can occur.
3. The result is stored in *general* register 0.
4. The coding that performs this function is out-of-line in FORTRAN IV (E) and in-line in FORTRAN IV. Out-of-line coding is taken from the FORTRAN library by the linkage editor and processed with the calling module. In-line coding is inserted by the FORTRAN compiler at the point in the source module where the function is referenced. This means that the in-line functions are available in FORTRAN IV by using the appropriate entry name but that they are not part of the library.
5. The expression x_1 (modulo x_2) is defined as $x_1 - \left[\frac{x_1}{x_2}\right] \cdot x_2$, 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 $\frac{x_1}{x_2}$.
6. 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.)

Implicitly Called Subprograms

The implicitly called subprograms perform operations required by the appearance of certain notation in a FORTRAN source statement. When a number is to be raised to a power or when multiplication and division of complex numbers are to be performed, the 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 values of the form REAL *4, the *xxx*FRXPR subprogram is called by the FORTRAN compiler.

The implicitly called subprograms in the FORTRAN library are described in Table 6. This table shows the general function, subprogram name, the FORTRAN library that contains the subprogram, implicit function reference, entry name, argument information, type of function value returned, and assembler requirements. The column headed "Implicit Function Reference" gives a sample source statement that might appear in a FORTRAN source module and cause the subprogram to be called. The rest of the column headings in Table 6 have the same meaning as those used with the explicitly called subprograms.

The action taken within the subprogram depends upon the type of base and exponent used. Tables 7 through 10 show the result of an exponentiation performed with the different combinations and values of base and exponent. In these tables, I and J are integers; A and B are real numbers; C is a complex number.

Table 6. Implicitly Called Mathematical Subprograms

General Function	Subprogram Name	Subset	Implicit Function Reference[1]	Entry[2] Name	Argument(s) No.	Argument(s) Type[3]	Function Value Type[3]	Assembler Requirements
Multiply and divide complex numbers	xxxCLAS	No	$y = z_1 * z_2$	CDMPY#	2	complex *16	complex *16	registers 0, 2 (4, 6) save area 5F
			$y = z_1/z_2$	CDDVD#	2	complex *16	complex *16	registers 0, 2 (4, 6) save area 5F
	xxxCSAS	No	$y = z_1 * z_2$	CMPY#	2	complex *8	complex *8	registers 0, 2 (4, 6) save area 5F
			$y = z_1/z_2$	CDVD#	2	complex *8	complex *8	registers 0, 2 (4, 6) save area 5F
Raise an integer to an integral power	xxxFIXPI	Yes	y = i**j	FIXPI#	2	i = integer*4 j = integer*4	integer *4	register 0 See Note 4 save area 18F
Raise a real number to an integral power	xxxFRXPI	Yes	y = a**j	FRXPI#	2	a = real *4 j = integer	real *4	register 0 save area 18F
	xxxFDXPI	Yes	y = a**j	FDXPI#	2	a = real *8 j = integer*4	real *8	register 0 save area 18F
Raise a real number to a real power	xxxFRXPR	Yes	y = a**b	FRXPR#	2	a = real *4 b = real *4	real *4	register 0 save area 18F
	xxxFDXPD	Yes	y = a**b	FDXPD#	2	a = real *8 b = real *8	real *8	register 0 save area 18F
Raise a complex number to an integral power	xxxFCDXI	No	y = z**j	FCDXI#	2	z = complex *16 j = integer	complex *16	register 0, 2 save area 18F
	xxxFCXPI	No	y = z**j	FCXPI#	2	z = complex *8 j = integer	complex *8	register 0, 2 save area 18F

NOTES:

1. This is only a *representation* of a FORTRAN statement; it is not the only way the subprogram may be called.
2. 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.
3. In FORTRAN IV (E), a real argument corresponds to the REAL *4 argument and a double precision argument corresponds to the REAL *8 argument.
4. The result is stored in *general* register 0.

• Table 7. Exponentiation With Integer Base and Exponent

Base (I)	Exponent (J)		
	J > 0	J = 0	J < 0
I > 0	Compute the function value	Function value = 1	Function value = 1 if I = 1 Otherwise, function value = 0
I = 0	Function value = 0	Error message xxx241I; or OA241I	Error message xxx241I; or OA241I
I < 0	Compute the function value	Function value = 1	Function value = −1 if I = −1 and if J is an odd number Function value = 1 if I = −1 and if J is an even number Otherwise, function value = 0

• Table 8. Exponentiation with Real Base and Integer 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 xxx242I or xxx243I; or OA242I or OA243I	Error message xxx242I or xxx243I; or OA242I or OA243I
A < 0	Compute the function value	Function value = 1	Compute the function value

• Table 9. Exponentiation with Real Base and 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 xxx244I or xxx245I; or OA244I or OA245I	Error message xxx244I or xxx245I; or OA244I or OA245I
A < 0	Error message xxx253I or xxx263I	Function value = 1	Error message xxx253I or xxx263I

Optional Extended Error Message Facility[1]

When the FORTRAN library has been created by system generation, the extended error message facility may be selected. This facility provides the following features:

1. Execution can continue after an error occurs in a mathematical function, with a standard result supplied as the functional value.

2. A user-written routine can be supplied to take corrective action.

[1]For OS/360 Full-Language FORTRAN only.

• Table 10. Exponentiation with Complex Base and Integer Exponent

Base (C) C = R + Ri	Exponent (J)		
	J > 0	J = 0	J < 0
R > 0 and Ri > 0	Compute the function value	Function value=1 + 0i	Compute the function value
R > 0 and Ri = 0	Compute the function value	Function value=1 + 0i	Compute the function value
R > 0 and Ri < 0	Compute the function value	Function value=1 + 0i	Compute the function value
R = 0 and Ri > 0	Compute the function value	Function value=1 + 0i	Compute the function value
R = 0 and Ri = 0	Function value 0 + 0i	Error message xxx246I or xxx247I; or OA246I or OA247I	Error message xxx246I or xxx247I; or OA246I or OA247I
R = 0 and Ri < 0	Compute the function value	Function value=1 + 0i	Compute the function value
R < 0 and Ri > 0	Compute the function value	Function value=1 + 0i	Compute the function value
R < 0 and Ri = 0	Compute the function value	Function value=1 + 0i	Compute the function value
R < 0 and Ri < 0	Compute the function value	Function value=1 + 0i	Compute the function value

3. Installation-written function subprograms may use the FORTRAN extended error message facility for the control and printing of error messages comparable to those put out by the FORTRAN library.

Table 11 describes the function values supplied as standard corrective actions and the parameters upon which recognition is based when user-supplied corrective action is chosen. Table 12 shows corrective action after interrupt occurrence.

User-Supplied Corrective Action

If the user chooses to write his own routine for corrective action, the user-supplied routine is called as follows:

```
CALL NAME (IRETCD, IERNO, DATA1, DATA2, ...)
```

Parameter	Description	Value on entry to subroutine NAME	Value upon entry from NAME
IRETCD	Return code	1	1 or 0
IERNO	Error Code	Cannot be changed	Cannot be changed
DATA1 DATA2	Data	Data in error upon entry	*For IRETCD=1,* the new values to be used in recomputing the function are placed in DATA1, DATA2. *For IRETCD=0,* these parameters may not be altered.

The subprogram NAME may do the following for these parameters:

• Examine the values in DATA1,DATA2.

• If the function is to be recomputed, new data is placed in DATA1,DATA2.

• If standard corrective action is desired, IRETCD is changed to 0.

The parameters listed in "User-Supplied Corrective Action" in Table 11 correspond to DATA1,DATA2. When writing the subprogram NAME in FORTRAN, declaration must be made for the type of constant being processed. For example, for error 246, DATA1 (C) is COMPLEX*8, and DATA2 (I) is INTEGER*4.

Table 11. Optional Service for Error Situations

Error Code	FORTRAN Reference	Invalid Argument Range	Options: Standard Corrective Action	Options: User-Supplied Corrective Action (See Note 1)
For errors 207-210, see Table 12				
216	CALL SLITE (I)	I>4	The call is treated as a no operation	I
216	CALL SLITET (I, J)	I>4	J=2	I
241	K=I**J	I=0, J≦0	K=0	I, J
242	Y=X**I	X=0, I≦0	Y=0	X, I
243	DA=D**I	D=0, I≦0	DA=0	D, I
244	XA=X**Y	X=0, Y≦0	XA=0	X, Y
245	DA=D**DB	D=0, DB≦0	DA=0	D, DB
246	CA=C**I	C=0+0*i*, I≦0	CA=0+0*i*	C, I
247	CDA=CD*I	C=0+0*i*, I≦0	CA=0+0*i*	CD, I
251	Y=SQRT (X)	X<0	$Y=\lvert X\rvert^{1/2}$	X
252	Y=EXP (X)	X>174.673	Y=*	X
253	Y=ALOG (X)	X=0	Y=−*	X
		X<0	Y=log |X|	X
	Y=ALOG10 (X)	X=0	Y=−*	X
		X<0	$Y=\log_{10}\lvert X\rvert$	X
254	Y=COS (X) Y=SIN (X)	$\lvert X\rvert\geqq 2^{18}*\pi$	$Y=\sqrt{2}/2$	X
255	Y=ATAN2 (X, XA)	X=0, XA=0	Y=0	X, XA
256	Y=SINH (X) Y=COSH (X)	|X|≧174.673	Y=*	X
257	Y=ARSIN (X) Y=ARCOS (X)	|X|>1	Y=0	X
258	Y=TAN (X) Y=COTAN (X)	$\lvert X\rvert\geqq(2^{18})*\pi$	Y=1	X
259	Y=TAN (X)	X is too close to an odd multiple of $\frac{\pi}{2}$	Y=*	X
	Y=COTAN (X)	X is too close to a multiple of π	Y=*	X
261	DA=DSQRT (D)	D≦0	$DA=\lvert D\rvert^{1/2}$	D
262	DA=DEXP (D)	D>174.673	DA=*	D
263	DA=DLOG (D)	D=0	DA=−*	D
		D<0	DA=log|X|	D
	DA=DLOG10 (D)	D=0	DA=−*	D
		D<0	$DA=\log_{10}\lvert X\rvert$	D
264	DA=DSIN (D) DA=DCOS (D)	$\lvert D\rvert\geqq 2^{50}*\pi$	$DA=\sqrt{2}/2$	D
265	DA=DATAN2 (D, DB)	D=0, DB=0	DA=0	D, DB

Variable	Type
I, J	Variables of INTEGER*4
X, XA, Y	Variables of REAL*4
D, DA, DB	Variables of REAL*8
C, CA	Variables of COMPLEX*8
Z, X_1, X_2	Complex variables to be given the length of the functioned argument when they appear
CD	Variables of COMPLEX*16

NOTES:

1. The user-supplied answer is obtained by recomputation of the function using the value set by the user routine for the parameters listed. (See "User-Supplied Corrective Action.")
2. The largest number that can be represented in floating point is indicated by *.

Table 11. Optional Service for Error Situations (Continued)

Error Code	FORTRAN Reference	Invalid Argument Range	Options: Standard Corrective Action	Options: User-Supplied Corrective Action (See Note 1)
266	DA=DSINH (D) DA=DCOSH (D)	$\|D\| \geqq 174.673$	DA=*	D
267	DA=DARSIN (D) DA=DARCOS (D)	$\|D\| > 1$	DA=0	D
268	DA=DTAN (D) DA=DCOTAN (D)	$\|D\| \geqq 2^{50} * \pi$	DA=1	D
269	DA=DTAN (D)	D is too close to an odd multiple of $\frac{\pi}{2}$	DA=*	D
	DA=DCOTAN (D)	D is too close to a multiple of π	DA=*	D
For errors 271-275, let $C=X_1+iX_2$				
271	Z=CEXP (C)	$X_1 > 174.673$	$Z=*(\cos X_2+\sin X_2)$	C
272	Z=CEXP (C)	$\|X_2\| \geqq 2^{18} * \pi$	$Z=0+0i$	C
273	Z=CLOG (C)	$C=0+0i$	$Z=-*+0i$	C
274	Z=CSIN (C) Z=CCOS (C)	$\|X_1\| \geqq 2^{18} * \pi$	$Z=0+0i$	C
275	Z=CSIN (C)	$X_2 > 174.673$	$Z=\frac{*}{2}(\sin X_1+i\cos X_1)$	C
	Z=CCOS (C)		$Z=\frac{*}{2}(\cos X_1-i\sin X_1)$	C
	Z=CSIN (C)	$X_2 < -174.673$	$Z=\frac{*}{2}(\sin X_1-i\cos X_1)$	C
	Z=CCOS (C)		$Z=\frac{*}{2}(\cos X_1+i\sin X_1)$	C
For errors 281-285, let $CD=X_1+iX_2$				
281	Z=CDEXP (CD)	$X_1 > 174.673$	$Z=*(\cos X_2+i\sin X_2)$	CD
282	Z=CDEXP (CD)	$\|X_2\| \geqq 2^{50} * \pi$	$Z=0+0i$	CD
283	Z=CDLOG (CD)	$CD=0+0i$	$Z=-*+0i$	CD
284	Z=CDSIN (CD) Z=CDCOS (CD)	$\|X_1\| \geqq 2^{50} * \pi$	$Z=0+0i$	CD
285	Z=CDSIN (CD)	$X_2 > 174.673$	$Z=\frac{*}{2}(\sin X_1+i\cos X_1)$	CD
	Z=CDCOS (CD)		$Z=\frac{*}{2}(\cos X_1-i\sin X_1)$	CD
	Z=CDSIN (CD)	$X_2 < -174.673$	$Z=\frac{*}{2}(\sin X_1-i\cos X_1)$	CD
	Z=CDCOS (CD)		$Z=\frac{*}{2}(\cos X_1+i\sin X_1)$	CD
290	Y=GAMMA (X)	$X \leqq 2^{-252}$ or $X \geqq 57.5744$	Y=*	X
291	Y=ALGAMA (X)	$X \leqq 0$ or $X \geqq 4.2937 * 10^{73}$	Y=*	X
300	DA=DGAMMA (D)	$D \leqq 2^{-252}$ or $D \geqq 57.5774$	DA=*	D
301	DA=DLGAMA (D)	$D \leqq 0$ or $D \geqq 4.2937 * 10^{73}$	DA=*	D

Variable	Type
I, J	Variables of INTEGER*4
X, XA, Y	Variables of REAL*4
D, DA, DB	Variables of REAL*8
C, CA	Variables of COMPLEX*8
Z, X_1, X_2	Complex variables to be given the length of the functioned argument when they appear
CD	Variables of COMPLEX*16

NOTES:

1. The user-supplied answer is obtained by recomputation of the function using the value set by the user routine for the parameters listed. (See "User-Supplied Corrective Action.")
2. The largest number that can be represented in floating point is indicated by *.

Table 12. Corrective Action After Program Interrupt Occurrence

Program Interrupt Messages			Options	
Error Code	Parameters Passed to User Exit	Reason for Interrupt (See Note 1)	Standard Corrective Action	User-Supplied Corrective Action
207	D, I	Exponent overflow (Interrupt Code 12)	Result register set to the largest possible floating-point number. The sign of the result register is not altered.	User may alter D. See Note 2.
208	D, I	Exponent underflow (Interrupt Code 13)	The result register is set to zero.	User may alter D. See Note 2.
209	None	Divide check, Integer divide (Interrupt Code 9), Decimal divide (Interrupt Code 11), Floating point divide (Interrupt Code 15). See Note 4.	There is no standard fixup. Result registers are not touched.	See Note 3.
210	None	Specification interrupt (Interrupt Code 6) occurs for boundary misalignment. Other interrupts occur during boundary alignment adjustment. They will be shown with this error code, and the PSW portion of the message will identify the interrupt.	No special corrective action other than correcting boundary misalignments.	See Note 3.

Variable	Type	Description
D	A variable REAL*8	This variable contains the contents of the result register after the interrupt.
I	A variable INTEGER*4	This variable contains the "exponent" as an integer value for the number in D. It may be used to determine the amount of the underflow or overflow. The value in I is not the true exponent, but what was left in the exponent field of a floating-point number after the interrupt.

Notes to Table 12

Note 1: A program interrupt occurs asynchronously. Interrupts are described in *IBM System/360 Operating System: Principles of Operation*, Form A22-6821.

Note 2: The user-exit routine may supply an alternate answer for the setting of the result register. This is accomplished by placing a value for D in the user-exit routine. Although the interrupt may be caused by a long or short floating-point operation, the user-exit routine need not be concerned with this. The user-exit routine should always set a REAL*8 variable, and the FORTRAN library will load short or long depending upon the floating-point operation that caused the interrupt.

Note 3: The user-exit routine does not have the ability to change result registers after a divide check. The boundary alignment adjustments are informational messages and there is nothing to alter before execution continues.

Note 4: For floating-point divide check, the contents of the result register is shown in the message.

The service subprograms supplied in the FORTRAN library are divided into two groups: one group tests machine indicators and the other group performs utility functions. Service subprograms are called by using the appropriate entry name in a FORTRAN source language CALL statement.

Machine Indicator Test Subprograms

The machine indicator subprograms (*xxx*FSLIT, *xxx*FOVER, and *xxx*FDVCH) test the status of pseudo indicators and may return a value to the calling program. When the indicator is zero, it is off; when the indicator is other than zero, it is on. In the following descriptions of the subprograms, *i* represents an integer expression and *j* represents an integer variable.

xxxFSLIT Subprogram

The *xxx*FSLIT subprogram is used to alter, test, and/or record the status of pseudo sense lights. Either of two entry names (SLITE or SLITET) is used to call the subprogram. The particular entry name used in the CALL statement depends upon the operation to be performed.

If the *four* sense lights are to be turned OFF or *one* sense light is to be turned ON, entry name SLITE is used. The source language statement is:

CALL SLITE(*i*)

where *i* has a value of 0, 1, 2, 3, or 4.

If the value of *i* is 0, the four sense lights are turned off; if the value of *i* is 1, 2, 3, or 4, the corresponding sense light is turned on. If the value of *i* is not 0, 1, 2, 3, or 4, an error message is issued and execution of this module (or phase) is terminated. (This error message is explained in Appendix C.)

If a sense light is to be tested and its status recorded, entry name SLITET is used. After a sense light is tested, it is set to off. The source language statement is:

CALL SLITET (*i*,*j*)

where:

i has a value of 1, 2, 3, or 4, and indicates which sense light to test.

j is set to 1 if the sense light was on; or to 2 if the sense light was off.

If the value of *i* is not 1, 2, 3, or 4, an error message is issued and execution of this module (or phase) is terminated. (This error message is explained in Appendix C.)

xxxFOVER Subprogram

The *xxx*FOVER 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*)

where:

j is set to 1 if a floating-point overflow condition exists; to 2 if no overflow or underflow condition exists; or to 3 if a floating-point underflow condition exists. A detailed description of each exception is given in Appendix C.

xxxFDVCH Subprogram

The *xxx*FDVCH 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 1 if the divide-check indicator was on; or to 2 if the indicator was off. A detailed description of the divide-check exception is given in Appendix C.

Utility Subprograms

The utility subprograms perform two operations for the FORTRAN programmer: they either terminate execution (*xxx*FEXIT) or dump a specified area of storage (*xxx*FDUMP).

xxxFEXIT Subprogram

The *xxx*FEXIT subprogram terminates execution of this load module (or phase) and returns control to the operating system. (This subprogram performs a function similar to that performed by the STOP statement.) The *xxx*FEXIT subprogram is called by using the entry name EXIT in a CALL statement. The source language statement is:

CALL EXIT

xxxFDUMP Subprogram

The *xxx*FDUMP subprogram dumps 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 this load module (or phase) is to be terminated after the dump is taken, entry name DUMP is used. The source language statement is:

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

where:

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 Table 13. The formats available depend upon the compiler in use. A sample printout for each format is given in Appendix F.

Table 13. The xxxFDUMP Subprogram Format Specifications

FORTRAN IV (E)	FORTRAN IV
0 specifies hexadecimal 4 specifies INTEGER 5 specifies REAL 6 specifies DOUBLE PRECISION	0 specifies hexademical 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

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 explained previously.

Programming Considerations

A load module (or, in the Model 44 Programming System, a member of the phase library) 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.

NOTE: In the following examples, A is a variable in COMMON, B is a real number, and the array TABLE is dimensioned as:

```
DIMENSION TABLE (20)
```

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, the following call to the IHCFDUMP subprogram could be used to dump TABLE and B in 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 *xxx*FDUMP 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. 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 *xxx*FDUMP 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 (X, Y)
DIMENSION X(10)
```

and the call to SUBI within the source module is:

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

then the following statement in the subroutine should be used 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 A(1) and Y is dumped, due to the method of transmitting arguments.

Appendix A. Algorithms

Appendix A contains information about the computations used in the explicitly called mathematical subprograms. This information is arranged in alphabetical order, according to the module (or phase) name of the subprogram. The entry names associated with each subprogram are given in parentheses after the module (or phase) name.

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. 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 any one who has an understanding of higher mathematics and access to the common texts on numerical analysis.[1]

The accuracy of an answer produced by these algorithms is influenced by two factors: the performance of the subprogram (see Appendix B) 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 appendix to describe the effect of an argument error upon the accuracy of the answer:

SYMBOL	EXPLANATION
$q(x)$	The result given by the subprogram.
$f(x)$	The correct result.
ϵ	$\left\lvert \frac{f(x) - g(x)}{f(x)} \right\rvert$ The relative error of the result given by the subprogram.
δ	The relative error of the argument.
E	$\lvert f(x) - g(x) \rvert$ 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.[2]

---

[1] Any of the common numerical analysis texts may be used as a reference. One such text is F. B. Hildebrand's *Introduction to Numerical Analysis* (McGraw-Hill Book Company, Inc., New York, N. Y., 1956). Background information for algorithms that use continued fractions may be found in H. S. Wall's *Analytic Theory of Continued Fractions* (D. VanNostrand Co., Inc., Princeton, N. J., 1948).

[2] 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.

xxxCLABS (CDABS) and xxxCSABS (CABS) Subprograms

1. Write $|x + iy| = a + ib$.
2. If $x = y = 0$, then $a = 0$ and $b = 0$.
3. Let $v_1 = \max(|x|, |y|)$, and
   $v_2 = \min(|x|, |y|)$.

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

The algorithms for both complex absolute value subprograms are identical. Each subprogram uses the appropriate real square root subprogram (*xxx*LSQRT or *xxx*SSQRT).

xxxCLEXP (CDEXP) and xxxCSEXP (CEXP) Subprograms

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 (*xxx*LEXP or *xxx*SEXP) and the appropriate real sine/cosine subprogram (*xxx*LSCN or *xxx*SSCN).

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 $\epsilon(R) \sim \Delta(x)$.

xxxCLLOG (CDLOG) and xxxCSLOG (CLOG) Subprograms

Algorithm

1. Write $\log_e(x + iy) = a + ib$.
2. Then, $a = \log_e |x + iy|$ and $b =$ the principle value of $\arctan \frac{y}{x}$.

The algorithms for both complex natural logarithm subprograms are identical. Each subprogram uses the appropriate complex absolute value subprogram (*xxx*CLABS or *xxx*CSABS), the appropriate real natural logarithm subprogram (*xxx*LLOG or *xxx*SLOG), and the appropriate arctangent subprogram (*xxx*LATN2 or *xxx*SATN2).

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 $E(a) = \delta(r)$.

xxxCLSQT (CDSQRT) and xxxCSSQT (CSQRT) Subprograms

Algorithm

1. Write $\sqrt{x + iy} = a + ib$.
2. If $x = y = 0$, then $a = 0$ and $b = 0$.
3. If $x \geqq 0$, then $a = \sqrt{\frac{|x| + |x + iy|}{2}}$

   and $b = \frac{y}{2a}$.

4. If $x < 0$, then $b = (\text{sign } y) \cdot \sqrt{\frac{|x| + |x + iy|}{2}}$

$$\text{and } a = \frac{y}{2b}.$$

The algorithms for both complex square root subprograms are identical. Each subprogram uses the appropriate real square root subprogram (*xxx*LSQRT or *xxx*SSQRT).

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 $\epsilon(H) \sim \delta(h)$.

xxxCLSCN Subprogram (CDSIN and 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. If $x < 0$, then $\sinh(-x) = -\sinh(x)$.
3. If $x > 0.3465736$, then $\sinh(x) = \frac{e^x - \frac{1}{e^x}}{2}$.
4. If $0 \leqq x \leqq 0.3465736$, then compute sinh (x) by use of the polynomial:
$$\frac{\sinh(x)}{x} \cong a_0 + a_1x^2 + a_2x^4 + \ldots + a_5x^{10}.$$
The coefficients are obtained by expanding the polynomial with respect to the Chebyshev polynomials over the range $0 \leqq x^2 \leqq 0.120113$. The relative error of this approximation is less than $2^{-21.8}$.
5. The value of $\cosh(x)$ is computed as $\cosh(x) = \sinh|x| + \frac{1}{e^{|x|}}$.
This computation uses the real exponential subprogram (*xxx*LEXP) and the real sine/cosine subprogram (*xxx*LSCN).

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 *xxx*LSCN, *xxx*LEXP, and *xxx*LSNH subprograms.

xxxCSSCN Subprogram (CSIN and 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. If $x < 0$, then $\sinh(-x) = -\sinh(x)$.
3. If $x > 0.3465736$, then $\sinh(x) = \frac{e^x - \frac{1}{e^x}}{2}$.

4. If $0 \leqq x \leqq 0.3465736$, then compute sinh (x) by use of the polynomial:

$$\frac{\sinh(x)}{x} \cong a_0 + a_1x^2 + a_2x^4.$$

The coefficients are obtained by expanding the polynomial with respect to the Chebyshev polynomials over the range $0 \leqq x^2 \leqq 0.120113$. The relative error of this approximation is less than $2^{-26.4}$.

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

This computation uses the real exponential subprogram (*xxx*SEXP) and the real sine/cosine subprogram (*xxx*SSCN).

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 *xxx*SSCN, *xxx*SEXP, and *xxx*SSCNH subprograms.

xxxLASCN Subprogram (DARSIN and DARCOS)

Algorithm

1. If $0 \leqq x \leqq \frac{1}{2}$, then compute $\arccos(x)$ as:

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

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

$$\arcsin(x) = x + c_1x^3 + c_2x^5 + \ldots + c_{12}x^{25}.$$

The coefficients are obtained by expanding the function $f(z) = \frac{\arcsin(x)}{x}$, $z = x^2$, with respect to the Chebyshev polynomials over the range, $0 \leqq x \leqq \frac{1}{4}$. The relative error of this approximation is less than $2^{-55.7}$.

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

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

If $\frac{1}{2} < x \leqq 1$, then compute arccos (x) 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 (*xxx*LSQRT).

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.

IHCLATAN Subprogram (DATAN)

Algorithm

1. Reduce the computation of arctan (x) to the case $0 \leqq x \leqq 1$ by using
$$\arctan(-x) = -\arctan(x) \text{ or}$$
$$\arctan\frac{1}{|x|} = \frac{\pi}{2} - \arctan|x|$$

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 $(\sqrt{3} - 1)\, x - \frac{1}{2} - \frac{1}{2} + x$ to avoid the loss of significant digits.

3. For $|x| \leqq \tan 15°$, use a continued fraction of the form:
$$\frac{\arctan(x)}{x} \cong 1 + \frac{a_1 x^2}{(b_1 + x^2) +} \frac{a_2}{(b_2 + x^2) +} \frac{a_3}{(b_3 + x^2) +} \frac{a_4}{(b_4 + x^2) +} \cdots$$
The relative error of this approximation is less than $2^{-57.9}$. The coefficients of this formula were derived by transforming the continued fraction:
$$\frac{\arctan(x)}{x} = 1 + \frac{-\frac{1}{3}}{\left(\frac{3}{5} + x^{-2}\right) -} \frac{\frac{3 \cdot 4}{25 \cdot 7}}{\left(\frac{23}{5 \cdot 9} + x^{-2}\right) -} \frac{\frac{16 \cdot 25}{7 \cdot 81 \cdot 11}}{\left(\frac{59}{9 \cdot 13} + x^{-2}\right) -} \frac{\frac{4 \cdot 3 \cdot 9}{5 \cdot 11 \cdot 169}}{\left(\frac{3 \cdot 37}{13 \cdot 17} + x^{-2}\right) - w}$$
where w has an approximate value of $\frac{2}{5 \cdot 11 \cdot 13 \cdot 17}(-x^{-2} + 40)$ but the true value of w is $\frac{\frac{64 \cdot 27}{5 \cdot 289 \cdot 19}}{\left(\frac{179}{3 \cdot 7 \cdot 17} + x^{-2}\right) +} \cdots$

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.

xxxLATN2 Subprogram (DATAN and DATAN2)

Algorithm

1. For $\arctan(x_1, x_2)$, if either $x_2 = 0$ or $\left|\frac{x_1}{x_2}\right| > 2^{56}$, the answer $= (\text{sign } x_1) \cdot \frac{\pi}{2}$.
Otherwise, if $x_2 > 0$, the answer $= \arctan\left(\frac{x_1}{x_2}\right)$, and
if $x_2 < 0$, the answer $= \arctan\left(\frac{x_1}{x_2}\right) + (\text{sign } x_1) \cdot \pi$.
The rest of the computation is identical for either one or two arguments.

2. Reduce the computation of $\arctan(x)$ to the case $0 \leqq x \leqq 1$, by using
$$\arctan(-x) = -\arctan(x), \text{ or}$$
$$\arctan\left(\frac{1}{|x|}\right) = \frac{\pi}{2} - \arctan|x|.$$

3. If necessary, reduce the computation further to the case $|x| \leqq \tan 15°$ by using
$$\arctan(x) = 30° + \arctan\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}.$$
The value of $\left|\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}\right| \leqq \tan 15°$ if the value x is within the range, $\tan 15° < x \leqq 1$. The value of $(\sqrt{3} \cdot x - 1)$ is computed as $(\sqrt{3} - 1)\, x - 1$ to avoid the loss of significant digits.

4. For $|x| \leqq \tan 15°$, use a continued fraction of the form:
$$\frac{\arctan(x)}{x} \cong 1 + \frac{a_1 x^2}{(b_1 + x^2) +} \; \frac{a_2}{(b_2 + x^2) +} \; \frac{a_3}{(b_3 + x^2) +} \; \frac{a_4}{(b_4 + x^2) +} \cdots$$
The relative error of this approximation is less than $2^{-57.9}$. The coefficients of this formula were derived by transforming the continued fraction:
$$\frac{\arctan(x)}{x} = 1 + \frac{\frac{-1}{3}}{\left(\frac{3}{5} + x^{-2}\right) -} \; \frac{\frac{3 \cdot 4}{25 \cdot 7}}{\left(\frac{23}{5 \cdot 9} + x^{-2}\right) -} \; \frac{\frac{16 \cdot 25}{7 \cdot 81 \cdot 11}}{\left(\frac{59}{9 \cdot 13} + x^{-2}\right) -} \; \frac{\frac{4 \cdot 3 \cdot 9}{5 \cdot 11 \cdot 169}}{\left(\frac{179}{13 \cdot 17} + x^{-2}\right) - w}$$
where w has an approximate value of $\frac{2}{5 \cdot 11 \cdot 13 \cdot 17}(-x^{-2} + 40)$ but the true value of w is $\frac{\frac{64 \cdot 27}{5 \cdot 289 \cdot 19}}{\left(\frac{179}{3 \cdot 7 \cdot 17} + x^{-2}\right) +} \cdots$

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.

xxxLERF Subprogram (DERF and DERFC)

Algorithm

1. If $0 \leqq x < 1$, then compute the error function by the following approximation:
$$\operatorname{erf}(x) \cong x\,(a_0 + a_1 x^2 + a_2 x^4 + \ldots + a_{11} x^{22}).$$
The coefficients were obtained by expanding the function $f(z) = \frac{\operatorname{erf}(x)}{x}$, $z = x^2$, with respect to the Chebyshev polynomials over the range, $0 \leqq x < 1$. The relative error of this approximation is less than $1.07 \cdot 2^{-57}$. The value of the complemented error function is computed as $\operatorname{erfc}(x) = 1 - \operatorname{erf}(x)$ and is greater than $\frac{1}{16}$.

2. If $1 \leqq x < 2.0400009$, then compute the complemented error function by the following approximation:
$$\operatorname{erfc}(x) \cong b_0 + b_1 z + b_2 z^2 + \ldots + b_{18} z^{18}$$
where $z = x - T_0$ and $T_0 \cong 1.999999$. The coefficients were obtained by ex-

panding the function $f(z) = \text{erfc}(z + T_0)$ with respect to the Chebyshev polynomials over the range $-1 \leqq x \leqq 0.04$. The absolute error of this approximation is less than $1.5 \cdot 2^{-61}$. The limits of this range and the base value for T_0 were used to minimize the hexadecimal truncation error. The value of the complemented error function within this range is greater than $\frac{1}{256}$. The value of the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

3. If $2.0400009 \leqq x \leqq 13.306$, then compute the complemented error function by the following approximation:

$$\text{erfc}\ (x) \cong \frac{(c_0 + c_1x^{-2} + c_2x^{-4} + \ldots + c_{20}x^{-40})\ e^{-x^2}}{x}$$

The coefficients were obtained by expanding the function $f(z) = \text{erfc}\ (x) \cdot x \cdot e^{x^2}$, $z = x^{-2}$, with respect to the Chebyshev polynomials over the range $2.04^{-2} > z \geqq 13.306^{-2}$. The relative error of this approximation ranges from 2^{-53} at 2.04 to 2^{-51} at 13.306. This computation uses the real exponential subprogram (*xxx*LEXP).

If $x \leqq 6.092$, then the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

If $x > 6.092$, then the error function is $\cong 1$.

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

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

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 $\epsilon \sim e^{-x^2} \cdot \Delta$.

xxxLEXP Subprogram (DEXP)

Algorithm

1. If $x < -180.2183$, then 0 is given as the answer.
2. Divide x by $\log_e 2$ and write

$$y = \frac{x}{\log_e 2} = (4a - b - \frac{c}{16} - d)$$

where a, b, and c are integers, $0 \leqq b \leqq 3$, $0 \leqq c \leqq 15$, and d is within the range $0 \leqq d < \frac{1}{16}$. Then, $e^x = 2^y = 16^a \cdot 2^{-b} \cdot 2^{-c/16} \cdot 2^{-d}$.

3. Compute 2^{-d} by using the Chebyshev interpolation of degree 6 over the range, $0 \leqq d < \frac{1}{16}$. The maximum relative error of this computation is 2^{-57}.
4. If $c > 0$, then multiply 2^{-d} by $2^{-c/16}$. (The 15 values of $2^{-c/16}$ for $1 \leqq c \leqq 15$ are included in the subprogram.)
5. If $b > 0$, then halve the result b times.
6. Finally, add the hexadecimal exponent a to the characteristic of the answer.

Effect of an Argument Error

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

xxxLGAMA Subprogram (DGAMMA and DLGAMA)

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 (*xxx*LLOG).
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 following approximation:
   $$\Gamma(x) \cong a_0 + a_1 z + a_2 z^2 + \ldots + a_{22} z^{22}$$
   where $z = x - 1.5$. The coefficients were obtained by expanding the function $f(z) = \Gamma(x)$ with respect to the Chebyshev polynomial for $|z| \leqq 0.5$. The absolute error of this approximation is less than $1.5 \cdot 2^{-58}$.
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:
   $$\log_e\Gamma(x) \cong x(\log_e(x) - 1) - \frac{1}{2}\log_e(x) + \frac{1}{2}\log_e(2\pi) + G(x).$$
   The modifier term $G(x)$ is computed as
   $$G(x) \cong b_1 x^{-1} + b_2 x^{-3} + b_3 x^{-5} + b_4 x^{-7} + b_5 x^{-9}.$$
   The coefficients were obtained by expanding the function $f(z) = \dfrac{G(x)}{x}$, $z = x^{-2}$, with respect to the Chebyshev polynomials over the range $0 < z < 8^{-2}$. The absolute error of the approximation for $G(x)$ is less than $x \cdot 2^{-56}$. Because, in this range, $x < \log_e\Gamma(x)$, the contribution of this error to the relative error of the value for log-gamma is less than 2^{-56}. This computation uses the real logarithm subprogram (*xxx*LLOG).

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

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, $\epsilon \sim \delta$.

xxxLLOG Subprogram (DLOG and 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. Finally, $\log_e\left(\frac{1+z}{1-z}\right)$ is computed by using the Chebyshev interpolation of degree 7 in z^2 over the range, $0 \leqq z^2 \leqq 0.02944$. The maximum relative error of this approximation is $2^{-59.6}$.
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.

xxxLSCN Subprogram (DSIN and 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 Chebyshev interpolation of degree 6 in r_1^2 for the sine, and of degree 7 in r_1^2 for the cosine. 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, $-\frac{\pi}{2} \leqq x \leqq +\frac{\pi}{2}$.

xxxLSCNH Subprogram (DSINH and DCOSH)

Algorithm

1. If $|x| < 0.3465736$, then compute $\sinh(x)$ as:
   $$\sinh(x) \cong x + c_1x^3 + c_2x^5 + c_3x^7 + c_4x^9 + c_5x^{11}.$$
   The coefficients are obtained by expanding the function $f(z) = \frac{\sinh(x)}{x}$, $z = x^2$, with respect to the Chebyshev polynomials over the range, $0 \leqq z < 0.12011326$. The relative error of this approximation is less than $2^{-61.9}$.
2. If either $|x| \geqq 0.3465736$ or the $\cosh(x)$ is desired, obtain $w = e^{|x|}$. Then, $\cosh(x) = \frac{w + w^{-1}}{2}$, and $\sinh(x) = (\text{sign } x) \cdot \frac{w - w^{-1}}{2}$. The real exponential subprogram (*xxx*LEXP) is used to compute the value of w.

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

xxxLSQRT Subprogram (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, $\sqrt{x} = 16^p \cdot 2^{-2q} \cdot \sqrt{m}$.
4. For the first approximation of $\sqrt{x}$, compute the following:
   $$y_0 = 2^{-2q} \cdot 16^p \cdot \left(\frac{2}{9} + \frac{8}{9} \cdot m\right).$$
   The maximum relative error of this approximation is $\frac{1}{9}$.
5. Apply the Newton-Raphson iteration
   $$y_{n+1} = \frac{1}{2}\left(y_n + \frac{x}{y_n}\right)$$
   four times to y_0 (the first two times in the short form and the last two times in the long form). The final step is performed as
   $$y_4 = y_3 + \frac{1}{2}\left(\frac{x}{y_3} - y_3\right)$$
   to minimize the computational truncation error. The maximum relative error of the final result is theoretically $2^{-65.70}$.

Effect of an Argument Error

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

xxxLTANH Subprogram (DTANH)

Algorithm

1. If $|x| < 0.54931$, then use the following fractional approximation:

$$\frac{\tanh(x)}{x} \cong 1 - \frac{a_1x^2 + a_2x^4 + a_3x^6 + x^8}{b_0 + b_1x^2 + b_2x^4 + b_3x^6 + x^8}$$

where:

$a_1 = 676440.765$ $\quad$ $b_0 = 2029322.295$
$a_2 = 45092.124$ $\quad$ $b_1 = 947005.29$
$a_3 = 594.459$ $\quad$ $b_2 = 52028.55$
$\quad$ $b_3 = 630.476$

The maximum relative error of this approximation is $2^{-64.5}$. The formula was obtained by transforming the continued fraction

$$\frac{\tanh(x)}{x} = 1 + \frac{x^2}{3 +}\ \frac{x^2}{5 +} \cdots \frac{x^2}{15 + w}$$

where w has an approximate value of 0.017, but the true value of w is

$$\frac{x^2}{17 +}\ \frac{x^2}{19 +} \cdots$$

2. 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 (*xxx*LEXP).
3. If $x \geqq 20.101$, then $\tanh(x) \cong 1$.
4. 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.

xxxLTNCT Subprogram (DTAN and DCOTAN)

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 $q_0 = q \bmod 4$.

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 values 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 $P(w^2)$ is of degree 3 and $Q(w^2)$ is of degree 4 in w^2. The coefficients of P and Q are obtained by economizing the continued fraction

$$\frac{\tan(z)}{z} = 1 - \frac{z^2}{3-}\,\frac{z^2}{5-}\,\frac{z^2}{7-}\cdots$$

in the following way.

Write: $\frac{\tan(z)}{z} \cong 1 - \frac{z^2}{3-}\,\frac{z^2}{5-}\,\frac{z^2}{7-}\,\frac{z^2}{9-}\,\frac{z^2}{(11+d_1)-}\,\frac{z^2}{(13+d_2)-}\,\frac{z^2}{(15+d_3)}$

and determine the values for d_1, d_2, and d_3 so that the right-hand expression gives the exact answers for $z^2 = 0.395$, 0.542, and 0.607. Then the maximum relative error of this formula over the range $0 \leqq z \leqq \frac{\pi}{4}$ is $3.4 \cdot 10^{-19}$.

Change the variable from z to $w = \frac{\pi}{4} \cdot z$ and rewrite the formula to obtain $P(w^2)$ and $Q(w^2)$.

5. 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 $\epsilon \sim \frac{2}{\sin(2x)}$ 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 $\operatorname{cotan}(x)$ for values of x near $k\pi$, where k is an integer.

xxxSASCN Subprogram (ARSIN and ARCOS)

Algorithm

1. If $0 \leqq x \leqq \frac{1}{2}$, then compute $\arccos(x)$ as:

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

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

$$\arcsin(x) \cong x + c_1x^3 + c_2x^5 + c_3x^7 + c_4x^9 + c_5x^{11}.$$

The coefficients are obtained by expanding the function $f(z) = \frac{\arcsin(x)}{x}$, $z = x^2$, with respect to the Chebyshev polynomials over the range $0 \leqq z \leqq \frac{1}{4}$. The relative error of this approximation is less than $2^{-27.5}$.

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

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

If $\frac{1}{2} < x \leqq 1$, then compute $\arccos(x)$ 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 (*xxx*SSQRT).

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.

IHCSATAN Subprogram (ATAN)

Algorithm

1. Reduce the computation of arctan (x) to the case $0 \leqq x \leqq 1$, by using
   $$\arctan(-x) = -\arctan(x), \text{ or}$$
   $$\arctan\left(\frac{1}{|x|}\right) = \frac{\pi}{2} - \arctan|x|.$$
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
   $$\frac{\arctan(x)}{x} = 1 - \frac{x^2}{3 + \dfrac{\frac{x^2}{5}}{\left(\frac{5}{7} + x^{-2}\right) - w}}$$
   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 $\dfrac{\frac{4 \cdot 5}{7 \cdot 7 \cdot 9}}{\left(\frac{43}{7 \cdot 11} + x^{-2}\right) + \cdots}$
   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.

xxxSATN2 Subprogram (ATAN and ATAN2)

Algorithm

1. For $\arctan(x_1, x_2)$, if either $x_2 = 0$ or $\left|\frac{x_1}{x_2}\right| > 2^{24}$, the answer $= (\text{sign } x_1) \cdot \frac{\pi}{2}$.
   Otherwise, if $x_2 > 0$, the answer $= \arctan\left(\frac{x_1}{x_2}\right)$, and
   $$\text{if } x_2 < 0, \text{ the answer} = \arctan\left(\frac{x_1}{x_2}\right) + (\text{sign } x_1)\,\pi.$$
   The rest of the computation is identical for either one or two arguments.

2. Reduce the computation of $\arctan(x)$ to the case $0 \leqq x \leqq 1$, by using
$$\arctan(-x) = -\arctan(x) \text{ or}$$
$$\arctan\left(\frac{1}{|x|}\right) = \frac{\pi}{2} - \arctan|x|.$$

3. If necessary, reduce the computation further to the case $|x| \leqq \tan 15°$ by using
$$\arctan(x) = 30° + \arctan\frac{\sqrt{3} \cdot x - 1}{x + \sqrt{3}}.$$
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.

4. For $|x| \leqq \tan 15°$, use the approximation formula:
$$\frac{\arctan(x)}{x} \cong 0.60310579 - 0.0516045x^2 + \frac{0.55913709}{x^2 + 1.4087812}$$
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 $\dfrac{\dfrac{4 \cdot 5}{7 \cdot 7 \cdot 9}}{\left(\dfrac{43}{7 \cdot 11} + x^{-2}\right) + \ldots}$
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.

xxxSERF Subprogram (ERF and ERFC)

Algorithm

1. If $0 \leqq x \leqq 1.317$, then compute the error function by the following approximation:
$$\text{erf}(x) \cong x(a_0 + a_1x^2 + a_2x^4 + \ldots + a_6x^{12}).$$
The coefficients were obtained by expanding the function $f(z) = \frac{\text{erf}(x)}{x}$, $z = x^2$, with respect to the Chebyshev polynomials over the range $0 \leqq x \leqq 1.317$. The relative error of this approximation is less than 2^{-24}. The value of the complemented error function is computed as $\text{erfc}(x) = 1 - \text{erf}(x)$ and is greater than $\frac{1}{16}$.

2. If $1.317 < x \leqq 2.0400009$, then compute the complemented error function by the following approximation:
$$\text{erfc}(x) \cong b_0 + b_1z + b_2z^2 + \ldots + b_7z^7$$
where $z = x - T_0$ and $T_0 \cong 2.0400009$. The coefficients were obtained by expanding the function $f(z) = \text{erfc}(x + T_0)$ with respect to the Chebyshev

polynomials over the range $(1.317 - T_0) < z \leqq 0$. The absolute error of this approximation is less than $1.3 \cdot 2^{-30}$. The value of the complemented error function within the range $1.317 < x \leqq T_0$ is greater than $\frac{1}{256}$. The value of the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

3. If $T_0 < x \leqq 13.306$, then compute the complemented error function by the following approximation:

$$\text{erfc}(x) \cong \frac{(c_0 + c_1x^{-2} + c_2x^{-4} + \ldots + c_6x^{-12})\, e^{-x^2}}{x}$$

The coefficients were obtained by expanding the function $f(z) = \text{erfc}(x) \cdot x \cdot e^{x^2}$, $z = x^{-2}$, with respect to the Chebyshev polynomials over the range $T_0^{-2} > z \geqq 13.306^{-2}$. The relative error of this approximation is less than $1.2 \cdot 2^{-23}$. This computation uses the real exponential subprogram (*xxx*SEXP).

If $x \leqq 3.9192$, then the error function is computed as $\text{erf}(x) = 1 - \text{erfc}(x)$.

If $x > 3.9192$, then the error function is $\cong 1$.

4. If $13.306 < x$, then the error function is $\cong 1$, and the complemented error function is $\cong 0$.

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

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

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 $\epsilon \sim e^{-x^2} \cdot \Delta$.

xxxSEXP Subprogram (EXP)

Algorithm

1. If $x < -180.218$, then 0 is given as the answer.
2. If $|x| < 2^{-28}$, then 1 is given as the answer.
3. 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 $0 \leqq d < 1$.
Then, $e^x = 2^y = (16^a \cdot 2^{-b} \cdot 2^{-d})$.

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

$$2^{-d} \cong 1 - \left(\frac{2d}{0.034657359\, d^2 + d + 9.9545958 - \frac{617.97227}{d^2 + 87.417497}} \right)$$

This formula can be obtained by transforming the Gaussian-type 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}.

5. Multiply 2^{-d} by 2^{-b}.
6. 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 round-off error of the argument causes a substantial relative error in the answer because $\Delta = \delta \cdot x$.

xxxSGAMA Subprogram (GAMMA and ALGAMA)

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 (*xxx*SLOG).
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 following approximation:
   $$\Gamma(x) \cong a_0 + a_1 z + a_2 z^2 + \ldots + a_9 z^9$$
   where $z = x - 1.5$. The coefficients were obtained by expanding the function $f(z) = \Gamma(x)$ with respect to the Chebyshev polynomials for $|z| \leqq 0.5$. The absolute error of this approximation is less than $1.5 \cdot 2^{-25}$.
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:
   $$\log_e \Gamma(x) \cong x(\log_e(x) - 1) - \frac{1}{2}\log_e(x) + \frac{1}{2}\log_e(2\pi) + G(x).$$
   The modifier term $G(x)$ is computed as
   $$G(x) = b_1 x^{-1} + b_2 x^{-2}.$$
   The absolute error of the approximation for $G(x)$ is $1.4 \cdot 2^{-23}$. This computation uses the real logarithm subprogram (*xxx*SLOG).

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

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 round-off error of the argument contributes greatly to the relative error of the answer. For log-gamma with large values of x, $\epsilon \sim \delta$.

xxxSLOG Subprogram (ALOG and ALOG10)

Algorithm

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

   If $\frac{1}{16} \leqq m < \frac{1}{8}$, then $a = \frac{1}{16}$ and $b = 4$.

   If $\frac{1}{8} \leqq m < \frac{1}{2}$, then $a = \frac{1}{4}$ and $b = 2$.

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

3. Write $z = \frac{m-a}{m+a}$. Then, $m = a \cdot \frac{1+z}{1-z}$, and $|z| \leqq \frac{1}{3}$.

4. Now, $x = 2^{4p-b} \cdot \frac{1+z}{1-z}$, and $\log_e x = (4p - b)\log_e 2 + log_e\left(\frac{1+z}{1-z}\right)$.

5. Finally, $\log_e\left(\frac{1+z}{1-z}\right)$ is evaluated using the Chebyshev interpolation of degree 4 in z^2 over the range, $0 \leqq z^2 \leqq \frac{1}{9}$. The maximum relative error of this approximation is $2^{-27.8}$.

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 round-off error of the argument, e.g., $\delta \sim 6 \cdot 10^{-8}$, then $E \sim 6 \cdot 10^{-8}$. Therefore, if the argument is close to 1, the relative error can be very large because the value of the function is very small.

xxxSSCN Subprogram (SIN and 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. Finally, the computation for either the sine or the cosine is performed, using the Chebyshev interpolation of degree 3 in r_1^2. The maximum relative error of the sine polynomial is $2^{-28.1}$ and that of the cosine polynomial is $2^{-24.6}$.

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, $-\frac{\pi}{2} \leqq x \leqq +\frac{\pi}{2}$.

xxxSSCNH Subprogram (SINH and COSH)

Algorithm

1. If $|x| < 0.3465736$, then compute $\sinh(x)$ as:
$$\sinh(x) \cong x + 0.16666505x^3 + 0.00836915x^5.$$
The coefficients were obtained by expanding the function $f(z) = \frac{\sinh(x)}{x}$, $z = x^2$, with respect to the Chebyshev polynomials over the range $0 < z < 0.12011326$. The relative error of this approximation is less than $2^{-26.5}$.
2. If either $|x| \geqq 0.3465736$ or the $\cosh(x)$ is desired, obtain $w = e^{|x|}$. Then, $\cosh(x) = \frac{w + w^{-1}}{2}$, and $\sinh(x) = (\text{sign } x).\ \frac{w - w^{-1}}{2}$. The real exponential subprogram (*xxx*SEXP) is used to compute the value of w.

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.

xxxSSQRT Subprogram (SQRT)

Algorithm

1. If $x = 0$, then the answer is 0.
2. Write $x = 16^{2p} \cdot m$, where p is an integer and m is within the range, $\frac{1}{256} \leqq m < 1$.
3. Then, $\sqrt{x} = 16^p \cdot \sqrt{m}$, where p is the exponent of the answer and m is the mantissa of the answer.
4. For the first approximation of $\sqrt{m}$, take hyperbolic approximations of the form $a + \frac{b}{c + x}$ where the values of a, b, and c depend upon the value of m as follows:
   a. If $\frac{1}{16} \leqq m < 1$, then $a = 1.80713$, $b = -1.57727$, $c = 0.954182$

      These values minimize the maximum relative error (ϵ_0) over the range, while making an exact fit at $m = 1$. The exact fit at $m = 1$ minimizes the computational loss of the last hexadecimal digit for the values of m slightly less than 1. The relative error of this approximation is less than $2^{-5.44}$.

b. If $\frac{1}{256} \leqq m < \frac{1}{16}$, then $a = 0.428795$
$b = -0.0214398$
$c = 0.0548470$

These values minimize $m^{1/8} \cdot \epsilon_0$ over this range of m where ϵ_0 denotes the relative error of this approximation. ϵ_0 is less than $2^{-6.5} \cdot m^{-1/8}$.

5. Multiply the result by 16^p to obtain the first approximation (y_0) of the answer.
6. To obtain the final answer, the Newton-Raphson iteration

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

must be applied twice to y_0. For $\frac{1}{16} \leqq m < 1$, the final relative error is theoretically less than $2^{-24.7}$; for $\frac{1}{256} \leqq m < \frac{1}{16}$, the final absolute error is theoretically less than $2^{-29} \cdot 16^p$.

Effect of an Argument Error

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

xxxSTANH Subprogram (TANH)

Algorithm

1. If $|x| \leqq 2^{-12}$, then $\tanh(x) \cong x$.
2. If $2^{-12} < |x| < 0.54931$, use the following fractional approximation:

$$\frac{\tanh(x)}{x} \cong 1 - \frac{x^2 + 35.1535}{x^2 + 45.1842 + \frac{105.4605}{x^2}}$$

This approximation has a relative error less than 2^{-27}. The formula can be obtained by transforming the continued fraction

$$\frac{\tanh(x)}{x} = 1 + \frac{x^2}{3+}\,\frac{x^2}{5+}\,\frac{x^2}{7+w}$$

where w has an approximate value of 0.0307, but the true value of w is

$$\frac{x^2}{9+}\,\frac{x^2}{11+}\cdots$$

3. If $0.54931 \leqq x < 9.011$, then use the identity $\tanh(x) = 1 - \frac{2}{e^{2x}+1}$. The computation for this case uses the real exponential subprogram (*xxx*SEXP).
4. If $x \geqq 9.011$, 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, $\epsilon \sim \delta$, and as the value of x increases, the effect of δ upon ϵ diminishes.

xxxSTNCT Subprogram (TAN and 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 values 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 $P(w^2) = 212.58037 - 12.559912w^2$
   $Q(w^2) = 270.665736 - 71.645273w^2 + w^4$

   This approximation is obtained by economizing the continued fraction

   $$\frac{\tan(z)}{z} = 1 - \frac{z^2}{3-}\,\frac{z^2}{5-}\,\frac{z^2}{7-}\cdots$$

   in the following way:

   Write: $$\frac{\tan(z)}{z} \cong 1 - \frac{z^2}{(3 + d_1)-}\,\frac{z^2}{(5 + d_2)-}\,\frac{z^2}{(7 + d_3)}$$

   and determine values for d_1, d_2, and d_3 so that the right-hand expression gives the exact answers for $z^2 = 0.19$, 0.432, and 0.594. Then the maximum relative error of this formula over the range $0 \leqq z \leqq \frac{\pi}{4}$ is $1.74 \cdot 10^{-8}$.

   Change the variable from z to $w = \frac{4}{\pi} \cdot z$ and rewrite the formula to obtain $P(w^2)$ and $Q(w^2)$.

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

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 $\operatorname{cotan}(x)$ for x near $k\pi$, where k is an integer.

Appendix B. Performance Statistics

Appendix B contains accuracy and timing statistics for the explicitly called mathematical subprograms. These statistics are presented in Table 14 and are arranged in alphabetical order, according to the entry names. The following column headings are used in Table 14:

Entry Name: This column gives the entry name that must be 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 argument 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, 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. For example, absolute error is given for $\sin(x)$ for values of x near π.

Sample: This column indicates the type of sample used for the accuracy figures. The type of sample depends upon the function and range under consideration. The statistics may be based either upon an exponentially (E) distributed argument sample or a uniformly (U) distributed 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 upon the assumption that the arguments are perfect (i.e., without error and, therefore, having no error propagation effect upon the answers). The only error in the answers are those introduced by the subprograms. Appendix A contains a description of some of the symbols used in this appendix; the following additional symbols are used in the presentation of accuracy figures:

$$M(\epsilon) = \text{Max}\left|\frac{f(x) - g(x)}{f(x)}\right|$$

The maximum relative error produced during testing.

$$\sigma(\epsilon) = \sqrt{\frac{1}{N}\sum_i \left|\frac{f(x_i) - g(x_i)}{f(x_i)}\right|^2}$$

The standard deviation (root-mean-square) of the relative error.

$$M(E) = \text{Max}\,|f(x) - g(x)|$$

The maximum absolute error produced during testing.

$$\sigma(E) = \sqrt{\frac{1}{N}\sum_i \left|f(x_i) - g(x_i)\right|^2}$$

The standard deviation (root-mean-square) of the absolute error.

In the formulas for the standard deviation, N represents the total number of arguments in the sample; i is a subscript that varies from 1 to N.

Accuracy for the Model 44 is based on performance with the FLOATING-POINT PRECISION switch in the "14" (vertical) position. This position selects the highest (56) of the four long-precision increments permitted.

Average Speed: This column gives the timing statistics. These statistics represent the average speed in microseconds for the various System/360 models. Statistics are supplied for Models 30, 40, 44, 50, 65, and 75. Statistics for the Model 75 are based upon two-way interleaving.

Table 14. Performance Statistics

Entry Name	Argument Range	Sample E/U	Accuracy Figures				Average Speed (Microseconds)						
			M (ϵ)	σ (ϵ)	M (E)	σ (E)	30	40	50	65 (See Note 8)	75	44 (See Note 9)	
ALGAMA	$0 < x \leqq 0.5$	U	1.09×10^{-6}	3.35×10^{-7}			10500	2800	865	221	131	727	553
	$0.5 < x < 3$	U			9.65×10^{-7}	3.74×10^{-7}	10500	2820	884	225	133	734	560
	$3 \leqq x < 8$	U	1.21×10^{-6}	2.86×10^{-7}			12100	3250	1020	259	151	820	638
	$8 \leqq x < 16$	U	1.25×10^{-6}	3.89×10^{-7}			7600	2010	617	162	97.3	470	356
	$16 \leqq x < 500$	U	1.04×10^{-6}	2.03×10^{-7}			7600	2010	617	162	97.3	477	362
ALOG	$0.5 \leqq x \leqq 1.5$	U			3.46×10^{-7}	8.62×10^{-8}	4481	1178	361	91.9	52.3	229	176
	$x < 0.5, x > 1.5$	E	8.32×10^{-7}	1.20×10^{-7}			4481	1178	361	91.9	52.3	229	176
ALOG10	$0.5 \leqq x \leqq 1.5$	U			1.64×10^{-7}	4.78×10^{-8}	4847	1278	388	98.1	56.1	248	193
	$x < 0.5, x > 1.5$	E	1.05×10^{-6}	2.17×10^{-7}			4847	1278	388	98.1	56.1	248	193
ARCOS	$-1 \leqq x \leqq +1$	U	1.80×10^{-7}	3.29×10^{-6}			5340	1460	451	118	70.6	325	238
ARSIN	$-1 \leqq x \leqq +1$	U	8.56×10^{-7}	2.38×10^{-7}			5270	1450	445	114	68.4	312	228
ATAN	The full range (Mod. 44: tan y, y in $-\frac{\pi}{2}, \frac{\pi}{2}$)	Note 7	9.75×10^{-7}	4.54×10^{-7}			For ATAN in FORTRAN IV (E) 3602 913 255 68.8 40.6 Additional time for ATAN in FORTRAN IV 104 47 24 8 5.9					165	125
ATAN2	The full range	Note 7	9.75×10^{-7}	4.54×10^{-7}			4874	1288	375	103	19.7	—	—
CABS	The full range	Note 1	1.87×10^{-6}	7.65×10^{-7}			5194	1421	396	108	68.4	299	205
CCOS	$\lvert x_1 \rvert \leqq 10, \lvert x_2 \rvert \leqq 20$	U	1.79×10^{-6} See Note 2	7.21×10^{-7}			15783	4433	1329	334	203	1089	848
CDABS	The full range	Note 1	3.32×10^{-15}	5.16×10^{-16}			14021	3061	638	150	89.0	733	656
CDCOS	$\lvert x_1 \rvert \leqq 10, \lvert x_2 \rvert \leqq 1$	U	5.16×10^{-15} See Note 3	3.42×10^{-16}			48343	11705	2335	507	301	3020	2807
CDEXP	$\lvert x_1 \rvert \leqq 1, \lvert x_2 \rvert \leqq \frac{\pi}{2}$	U	4.04×10^{-16}	1.39×10^{-16}			41737	10144	2048	456	260	2616	2419
	$\lvert x_1 \rvert \leqq 20, \lvert x_2 \rvert \leqq 20$	U	3.63×10^{-15}	1.29×10^{-16}			41737	10144	2048	456	260	2611	2415
CDLOG	The full range	Note 1	8.73×10^{-15}	6.38×10^{-17}			53362	11940	2393	542	317	3105	2860
CDSIN	$\lvert x_1 \rvert \leqq 10, \lvert x_2 \rvert \leqq 1$	U	3.72×10^{-15} See Note 4	3.49×10^{-16}			48250	11668	2322	503	298	3018	2808
CDSQRT	The full range	Note 1	9.86×10^{-16}	1.91×10^{-16}			27951	5906	1282	301	181	1543	1392
CEXP	$\lvert x_1 \rvert \leqq 170, \lvert x_2 \rvert \leqq \frac{\pi}{2}$	U	1.18×10^{-6}	2.34×10^{-7}			13731	3888	1166	291	177	958	732
	$\lvert x_1 \rvert \leqq 170, \frac{\pi}{2} < \lvert x_2 \rvert \leqq 20$	U	1.06×10^{-6}	2.51×10^{-7}			13899	3930	1180	294	178	971	746
CLOG	The full range except (1 + 0i)	Note 1	2.00×10^{-6}	1.56×10^{-7}			15504	4246	1261	338	216	1014	777

Table 14. Performance Statistics (Continued)

Entry Name	Argument Range	Sample E/U	Accuracy Figures				Average Speed (Microseconds)						
			M (ϵ)	σ (ϵ)	M (E)	σ (E)	30	40	50	65 (See Note 8)	75	44 (See Note 9)	
COS	$0 \leqq x \leqq \pi$	U			1.47×10^{-7}	5.48×10^{-8}	3934	1047	298	74.2	44.0	196	157
	$-10 \leqq x < 0$ $\pi < x \leqq 10$	U			1.42×10^{-7}	5.67×10^{-8}	3990	1061	303	75.4	44.5	198	159
	$10 < \lvert x \rvert \leqq 100$	U			1.35×10^{-7}	5.61×10^{-8}	3990	1061	303	75.4	44.5	200	161
COSH	$-5 \leqq x \leqq +5$	U	1.31×10^{-6}	3.40×10^{-6}			6110	1810	570	145	89.2	486	312
COTAN	$\lvert x \rvert \leqq \frac{\pi}{4}$	U	1.29×10^{-6}	3.68×10^{-7}			4420	1180	341	86.7	56.0	227	180
	$\frac{\pi}{4} < \lvert x \rvert \leqq \frac{\pi}{2}$	U	3.80×10^{-4} See Note 5	7.70×10^{-4}			4610	1220	351	89.3	55.5	233	188
	$\frac{\pi}{2} < \lvert x \rvert \leqq 10$	U	1.13×10^{-5} See Note 5	6.03×10^{-7}			4580	1210	348	88.1	55.0	233	188
	$10 < \lvert x \rvert \leqq 100$	U	1.67×10^{-5} See Note 5	6.67×10^{-7}			4580	1210	348	88.1	55.0	233	187
CSIN	$\lvert x_1 \rvert \leqq 10, \lvert x_2 \rvert \leqq 1$	U	1.97×10^{-6} See Note 6	7.09×10^{-7}			15690	4397	1316	331	200	1081	843
CSQRT	The full range	Note 1	1.61×10^{-6}	4.58×10^{-7}			10408	2870	805	219	140	676	493
DARCOS	$-1 \leqq x \leqq +1$	U	2.72×10^{-16}	9.35×10^{-17}			22600	5100	1100	246	143	1439	1289
DARSIN	$-1 \leqq x \leqq +1$	U	2.40×10^{-16}	6.00×10^{-17}			22400	5060	1090	243	140	1440	1292
DATAN	The full range (Mod. 44: tan y, y in $-\frac{\pi}{2}, \frac{\pi}{2}$)	Note 7	2.08×10^{-16}	6.64×10^{-17}			For DATAN in FORTRAN IV (E) 19056	4000	715	153	83.6	1010	966
							Additional time for DATAN in FORTRAN IV 104	47	24	8	5.9		
DATAN2	The full range	Note 7	2.08×10^{-16}	6.64×10^{-17}			22281	4734	886	195	22.9	—	—
DCOS	$0 \leqq x \leqq \pi$	U			1.79×10^{-16}	6.40×10^{-17}	13133	3146	605	132	74.2	761	713
	$-10 \leqq x < 0$ $\pi < x \leqq 10$	U			1.76×10^{-16}	5.93×10^{-17}	13133	3146	605	132	74.2	762	714
	$10 < \lvert x \rvert \leqq 100$	U			2.65×10^{-15}	1.01×10^{-15}	13133	3146	605	132	74.2	759	712
DCOSH	$-5 \leqq x \leqq +5$	U	4.81×10^{-16}	1.34×10^{-16}			18300	4260	898	206	119	1050	923
DCOTAN	$\lvert x \rvert \leqq \frac{\pi}{4}$	U	3.46×10^{-16}	8.38×10^{-17}			15300	3590	675	150	89.2	869	819
	$\frac{\pi}{4} < \lvert x \rvert \leqq \frac{\pi}{2}$	U	1.72×10^{-13} See Note 5	5.00×10^{-15}			15900	3640	715	156	89.8	906	856
	$\frac{\pi}{2} < \lvert x \rvert \leqq 10$	U	5.33×10^{-13} See Note 5	1.09×10^{-14}			15800	3690	706	155	89.1	901	852
	$10 < \lvert x \rvert \leqq 100$	U	8.61×10^{-13} See Note 5	4.61×10^{-14}			15800	3690	706	155	89.1	899	849

Table 14. Performance Statistics (Continued)

Entry Name	Argument Range	Sample E/U	Accuracy Figures				Average Speed (Microseconds)						
			M (ϵ)	σ (ϵ)	M (E)	σ (E)	30	40	50	65 (See Note 8)	75	44 (See Note 9)	
DERF	$\|x\| \leqq 1.317$	U	1.70×10^{-16}	2.71×10^{-17}			18400	4610	879	190	109	1208	1083
	$1.317 < \|x\| \leqq 2.04$	U	2.91×10^{-17}	1.17×10^{-17}			24300	6060	1150	250	141	1635	1467
	$2.04 < \|x\| < 6.092$	U	1.70×10^{-17}	8.03×10^{-18}			45200	10900	2080	449	259	2868	2626
DERFC	$-6 < x < 0$	U	1.88×10^{-16}	6.84×10^{-17}			36700	8920	1700	369	218	2355	2147
	$0 \leqq x \leqq 1.317$	U	3.52×10^{-16}	7.62×10^{-17}			18600	4650	891	193	111	1213	1086
	$1.317 < x \leqq 2.04$	U	4.45×10^{-16}	1.27×10^{-16}			24100	6000	1130	244	139	1623	1454
	$2.04 < x < 4$	U	4.02×10^{-15}	1.24×10^{-15}			45000	10800	2060	444	258	2854	2612
	$4 \leqq x < 13.3$	U	5.02×10^{-15}	1.40×10^{-15}			45200	10900	2090	451	263	2871	2628
DEXP	$\|x\| \leqq 1$	U	2.27×10^{-16}	7.49×10^{-17}			12145	2907	607	138	75.5	720	648
	$1 < \|x\| \leqq 20$	U	2.31×10^{-15}	8.69×10^{-16}			12145	2907	607	138	75.5	716	644
	$20 < \|x\| \leqq 170$	U	2.33×10^{-15}	9.33×10^{-16}			12145	2907	607	138	75.5	715	643
DGAMMA	$0 < x < 1$	U	2.18×10^{-16}	7.93×10^{-17}			30700	7500	1400	304	175	2058	1853
	$1 \leqq x \leqq 2$	U	3.12×10^{-17}	8.45×10^{-18}			28200	7050	1340	292	169	1921	1714
	$2 < x \leqq 4$	U	1.69×10^{-15}	4.60×10^{-16}			30100	7520	1430	312	180	2034	1827
	$4 < x < 8$	U	2.85×10^{-15}	9.46×10^{-16}			33900	8470	1610	352	205	2268	2054
	$8 \leqq x < 16$	U	6.42×10^{-15}	2.01×10^{-15}			40600	9560	1940	434	241	2498	2294
	$16 \leqq x < 57$	U	6.2×10^{-14}	2.96×10^{-14}			40600	9560	1940	434	241	2503	2298
DLGAMA	$0 < x \leqq 0.5$	U	4.11×10^{-16}	1.60×10^{-16}			46900	11300	2160	471	267	3056	2783
	$0.5 < x < 3$	U			2.86×10^{-16}	1.16×10^{-16}	44900	11100	2130	466	264	2983	2709
	$3 \leqq x < 8$	U	2.38×10^{-15}	3.99×10^{-16}			45900	12200	2330	512	292	3228	2947
	$8 \leqq x < 16$	U	3.36×10^{-16}	1.18×10^{-16}			28200	6580	1310	296	161	1712	1572
	$16 \leqq x < 500$	U	1.62×10^{-15}	2.43×10^{-16}			28200	6580	1310	296	161	1725	1585
DLOG	$0.5 \leqq x \leqq 1.5$	U			1.85×10^{-16}	7.29×10^{-17}	16044	3769	734	161	86.5	929	855
	$x < 0.5, x > 1.5$	E	3.31×10^{-16}	5.46×10^{-17}			16041	3765	733	161	86.5	929	855
DLOG10	$0.5 \leqq x \leqq 1.5$	U			8.23×10^{-17}	3.09×10^{-17}	17149	4048	778	171	92.3	997	920
	$x < 0.5, x > 1.5$	E	6.14×10^{-16}	9.96×10^{-17}			17147	4044	777	170	92.0	996	920
DSIN	$\|x\| \leqq \frac{\pi}{2}$	U	4.08×10^{-16}	4.85×10^{-17}	9.10×10^{-17}	2.17×10^{-17}	13145	3148	609	133	75.5	762	714
	$\frac{\pi}{2} < \|x\| \leqq 10$	U			1.64×10^{-16}	6.35×10^{-17}	13145	3148	609	133	75.5	763	715
	$10 < \|x\| \leqq 100$	U			2.69×10^{-15}	1.03×10^{-15}	13145	3148	609	133	75.5	761	714
DSINH	$\|x\| \leqq 0.34657$	U	2.10×10^{-16}	5.29×10^{-17}			8650	2170	408	88.5	52.4	505	457
	$0.34657 < \|x\| \leqq 5$	U	3.59×10^{-16}	8.73×10^{-17}			18400	4280	901	207	119	1049	925

Table 14. Performance Statistics (Continued)

Entry Name	Argument Range	Sample E/U	Accuracy Figures				Average Speed (Microseconds)						
			M (ϵ)	σ (ϵ)	M (E)	σ (E)	30	40	50	65 (See Note 8)	75	44 (See Note 9)	
DSQRT	The full range	E	1.08×10^{-16}	2.17×10^{-17}			8173	1684	355	85.3	49.2	370	334
DTAN	$\lvert x\rvert \leqq \frac{\pi}{4}$	U	5.25×10^{-16}	9.26×10^{-17}			15100	3500	647	142	84.1	852	806
	$\frac{\pi}{4} < \lvert x\rvert \leqq \frac{\pi}{2}$	U	1.67×10^{-12} See Note 5	3.69×10^{-14}			15700	3660	696	151	86.8	902	854
	$\frac{\pi}{2} < \lvert x\rvert \leqq 10$	U	1.57×10^{-13} See Note 5	4.51×10^{-15}			15600	3640	688	150	86.1	895	848
	$10 < \lvert x\rvert \leqq 100$	U	3.79×10^{-12} See Note 5	9.50×10^{-14}			15600	3640	688	150	86.1	893	846
DTANH	$\lvert x\rvert \leqq 0.54931$	U	2.00×10^{-16}	4.45×10^{-17}			12299	2850	477	106	55.5	668	641
	$0.54931 < \lvert x\rvert \leqq 5$	U	1.99×10^{-16}	2.54×10^{-17}			16078	3778	833	192	110	979	874
EXP	$\lvert x\rvert \leqq 1$	U	4.65×10^{-7}	1.28×10^{-7}			4173	1250	388	95.2	53.4	311	192
	$\lvert x\rvert \leqq 170$	U	4.69×10^{-7}	1.17×10^{-7}			4183	1250	387	94.8	53.4	308	189
ERF	$\lvert x\rvert \leqq 1.317$	U	9.26×10^{-7}	1.43×10^{-7}			4140	1120	363	88.5	52.7	276	189
	$1.317 < \lvert x\rvert \leqq 2.04$	U	9.02×10^{-8}	3.42×10^{-8}			4500	1230	397	99.1	58.8	322	220
	$2.04 < \lvert x\rvert \leqq 3.9192$	U	6.07×10^{-8}	3.42×10^{-8}			10000	2800	845	213	127	740	525
ERFC	$-3.8 < x < 0$	U	9.10×10^{-7}	2.97×10^{-7}			7040	1960	607	153	91.6	521	367
	$0 \leqq x \leqq 1.317$	U	3.90×10^{-6}	5.65×10^{-7}			4250	1150	374	92.0	54.3	284	195
	$1.317 < x \leqq 2.04$	U	1.02×10^{-6}	2.13×10^{-7}			4410	1210	387	96.0	58.0	319	216
	$2.04 < x < 4$	U	1.20×10^{-6}	3.60×10^{-7}			9950	2780	835	210	126	737	521
	$4 \leqq x \leqq 13.3$	U	1.52×10^{-5}	8.45×10^{-6}			10100	2840	859	216	131	749	532
GAMMA	$0 < x < 1$	U	9.86×10^{-7}	3.45×10^{-7}			5840	1560	484	123	74.0	433	306
	$1 \leqq x \leqq 2$	U	1.00×10^{-7}	3.74×10^{-8}			5620	1520	489	123	73.5	428	301
	$2 < x \leqq 4$	U	9.29×10^{-7}	3.63×10^{-7}			6330	1700	546	137	81.1	460	330
	$4 < x < 8$	U	2.25×10^{-6}	8.14×10^{-7}			7740	2070	659	166	96.2	538	402
	$8 \leqq x < 16$	U	2.29×10^{-5}	7.67×10^{-6}			12000	3320	1020	263	155	845	618
	$16 \leqq x < 57$	U	4.36×10^{-5}	1.45×10^{-5}			12000	3320	1020	263	155	842	616
SIN	$\lvert x\rvert \leqq \frac{\pi}{2}$	U	1.59×10^{-6}	2.02×10^{-7}	1.31×10^{-7}	5.55×10^{-8}	3876	1036	298	74.2	44.3	194	155
	$\frac{\pi}{2} < \lvert x\rvert \leqq 10$	U			1.41×10^{-7}	5.53×10^{-8}	3989	1064	307	76.4	45.3	201	163
	$10 < \lvert x\rvert \leqq 100$	U			1.46×10^{-7}	5.61×10^{-8}	3989	1064	307	76.4	45.3	200	162
SINH	$-5 \leqq x \leqq +5$	U	1.20×10^{-6}	3.20×10^{-7}			5890	1740	545	139	85.3	461	297
SQRT	The full range	E	8.70×10^{-7}	1.68×10^{-7}			2965	801	210	59.1	35.8	142	90

Table 14. Performance Statistics (Continued)

Entry Name	Argument Range	Sample E/U	Accuracy Figures				Average Speed (Microseconds)						
			M (ϵ)	σ (ϵ)	M (E)	σ (E)	30	40	50	65 (See Note 8)	75	44 (See Note 9)	
TAN	$\lvert x\rvert \leqq \frac{\pi}{4}$	U	1.56×10^{-6}	3.22×10^{-7}			4220	1120	319	79.9	51.3	209	166
	$\frac{\pi}{4} < \lvert x\rvert \leqq \frac{\pi}{2}$	U	6.58×10^{-5} See Note 5	1.67×10^{-6}			4500	1184	338	85.3	52.8	232	188
	$\frac{\pi}{2} < \lvert x\rvert \leqq 10$	U	4.92×10^{-5} See Note 5	1.28×10^{-6}			4460	1170	335	84.1	52.4	226	183
	$10 < \lvert x\rvert \leqq 100$	U	3.35×10^{-5} See Note 5	1.02×10^{-6}			4460	1170	335	84.1	52.4	227	184
TANH	$\lvert x\rvert \leqq 0.54931$	U	8.12×10^{-7}	1.66×10^{-7}			2581	649	173	46.1	29.0	89	63
	$0.54931 < \lvert x\rvert \leqq 5$	U	5.74×10^{-7}	7.53×10^{-8}			5952	1774	551	142	86.3	446	294

Notes to Table 14

These notes are associated with Table 14 and contain more detailed information about samples and relative errors for certain functions.

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

Note 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.

Note 3: The maximum relative error cited for the CDCOS function is based upon a set of 1500 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.

Note 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.

Note 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 Appendix A for a description of the behavior of errors when the argument is near a singularity or a zero of the function.

Note 6: The maximum relative error cited for the CSIN function is based upon a set of 2000 randon 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.

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

Note 8: The statistics for the Model 75 are based upon two-way interleaving.

Note 9: The second column of speeds for the Model 44 applies to that machine with high-speed registers.

Appendix C. Interruption and Error Procedures

Appendix C contains descriptions of the procedures followed when the execution of a load module is discontinued. Execution may be discontinued due to one of two reasons: an interruption or an error. After an interruption is processed, execution of this load module or phase continues; after an error is processed, execution of this load module or phase is terminated. The following text explains the procedure used to handle each case.

Interruption Procedures

A program interruption is a computer-originated break in the flow of processing. (For a full description of program interrupts, see the publication *IBM System/360 Principles of Operation,* Form A22-6621.) The FORTRAN library processes those interrupts that are described below; all others are handled directly by the system Supervisor and will cause job termination.

The following services are provided by the FORTRAN support for interrupt occurrence:

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 "Service Subprograms."

2. A message is printed on the object error unit when each interrupt occurs. The 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, below) occur. (For a description of the format of floating-point numbers, see the publication *IBM System/360 Principles of Operation,* Form A22-6621.) Result registers are also set when a floating-point instruction is referenced by an assembler language execute (EX) 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.

The program interrupt message contains the old program status word (PSW), which indicates the cause of the interrupt. Figure 1 shows the format of the message as it is issued by the operating system. Figure 2 shows the format as issued by the Model 44 system, and Figure 2.1 shows the DOS format. If the letter A appears in parentheses in the program interrupt message, boundary adjustment has taken place. The letter P in the message indicates that the interruption was precise. This will always be the case for nonspecification interrupt messages in FORTRAN except when using machines with special hardware on which imprecise interruptions may occur. The eighth character in the PSW represents the code number associated with the type of interruption. These interruptions are described in the following paragraphs. (For more information on the PSW, see the publication *IBM System/360 Principles of Operation,* Form A22-6621.)

Specification Exception (Code 6): The specification exception (code 6) is recognized when a data address does not specify an integral boundary for that unit of information. A specification error would occur, for example, during the execution of the following program segment:

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

Fixed-Point Divide Exception (Code 9): The fixed-point divide exception (code 9) is recognized when division of a fixed-point number by zero is attempted.

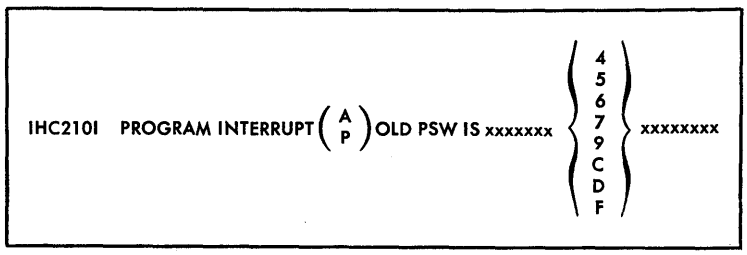

Figure 1. Format of Program Interrupt Message, Operating System

A fixed-point divide exception would occur during execution of the following statements:

```
J = 0
I = 7
K = I/J
```

Exponent-Overflow Exception (Code C): The exponent-overflow exception (code C) is recognized when the absolute value of the result of a floating-point addition, subtraction, multiplication, or division is greater than or equal to 16^{63} (approximately 7.2 x 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 and 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.

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

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

Before program execution continues, the FORTRAN library sets the result register to the largest possible floating-point number that can be represented in short precision [16^{63} * (1-16^{-6})] or in long precision [16^{63} * (1-16^{-14})], but the sign of the result is not changed. The condition code is not altered.

Exponent-Underflow Exception (Code D): The exponent-underflow exception (code D) is recognized when the absolute value of the result of a floating-point addition, subtraction, multiplication, or division, is less than 16^{-65} (approximately 5.4 x 10^{-79}) but not equal to 0. An exponent-underflow exception would occur during execution of the statement:

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

Although exponent underflows can be masked, FORTRAN jobs are executed without the mask so that the library will handle such interrupts.

When the interrupt occurs, the result register contains a floating-point number whose fraction and 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.

Exponent underflow 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 (TE) may be computed, as follows:

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

Before program execution continues, the FORTRAN library sets the result register 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. NOTE: The Sysem/360 Operating System FORTRAN programmer who wishes to take advantage of the "exponent wraparound" feature and handle the interrupt in his own program must call an assembly language subroutine to issue a SPIE macro-instruction, which will override the FORTRAN interruption routine.

Floating-Point Divide Exception (Code F): The floating-point divide exception (code F) is recognized when division of a floating-point number by zero is attempted. A floating-point divide exception would occur during execution of the following statements:

```
B = 0.0
A = 1.0
C = A/B
```

System/360 Operating System

The interrupt message is written in the system output data set.

A specification exception program interrupt message (code 6) is issued only if the BOUNDRY=ALIGN option was specified in the FORTLIB macro-instruction during system generation and a boundary alignment error occurs. Then the boundary alignment routine is invoked to correct the boundary misalignment. 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). (In these cases, the letter A appears in parentheses in the program inter-

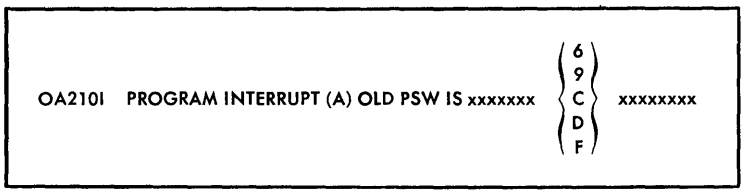

Figure 2. Format of Program Interrupt Message, Model 44 System

rupt message.) Then the job will terminate because both a specification error and a protection, addressing, or data error have been detected. The completion code in the dump will specify that the job terminated because of the specification error.

The number of warning messages printed is limited to ten. After ten boundary alignment adjustments have been made, the message is suppressed, but boundary alignment violations continue to be corrected.

Protection Exception (Code 4): 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 without a message.

Addressing Exception (Code 5): The addressing exception (code 5) is recognized when the address of the data is outside of available storage for the particular installation. A message is issued only if a specification exception (code 6) has already been recognized in the same instruction. Otherwise, the job terminates abnormally without a message.

Data Exception (Code 7): The data exception (code 7) is recognized when the sign or digit codes for a CONVERT TO BINARY instruction are incorrect. A message is issued only if a specification exception (code 6) has already been recognized in the same instruction. Otherwise, the job terminates abnormally without a message.

Model 44 Programming System

The interrupt message is written in SYSOPT.

The program interrupt message (with code 6, as described in "Specification Exception") that results when the boundary specification convention is violated will contain the "(A)" only if the &FIX option has been turned on (SETA 1) in BOAUOPT. With that option on (as it is in the distributed version of the system), a routine that adjusts for the misalignment is executed each time such a violation occurs, and processing continues.

With the &FIX option on, the number of program interrupt messages put out as the result of boundary violations is limited to a message for each of the first *n* violations per execution, where *n* is the operand of the SETA instruction for the &PRNTMES option. In the system as distributed, this operand is equal to 0. Only the message, and not the alignment correction, is inhibited.

Disk Operating System

The interrupt message is written on SYSLST.

The ILF225I message is issued for arithmetic exceptions, such as fixed point divide. The REGISTER CONTAINED portion is issued only for exponent overflow and underflow conditions. Processing resumes after the error condition is listed.

The ILF226I message appears for other forms of program interrupt. The A character appears only for boundary alignment errors. In such cases, it is for information only since an attempt is made to correct the error and resume processing. If, however, an instruction containing a boundary alignment violation also contains a protection, addressing, or data error, the message is reissued with the appropriate code, 4, 5, or 7. The job then terminates.

The interruption codes and their explanations are identical to those for the Operating System, as discussed previously in this section.

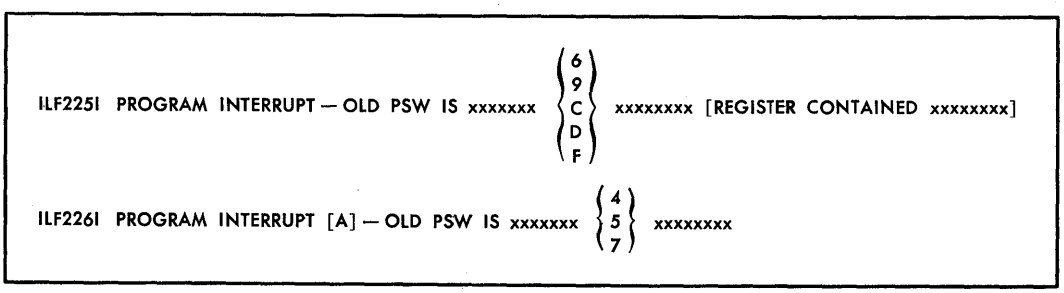

Figure 2.1. Format of DOS Program Interrupt Messages

Error Procedures

During execution, the mathematical subprograms assume that the argument(s) is the correct type. No checking is done for erroneous arguments (i.e., 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 Tables 2 through 6, an error message is written on the object 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.

The error message that is issued has the format:

IHCyyyI [Message text]
TRACEBACK FOLLOWS . . .
or
OAyyyI
or
ILFyyyI
TRACEBACK FOLLOWS . . .

where *yyy* is a numeric code that identifies the error detected.

The first message is issued by the Operating System, the second by the Model 44 Programming System, and the last by the Disk Operating System. Traceback is a diagnostic tool for FORTRAN under the Operating System and the Disk Operating System. It is a list of routines in the direct line of call to the routine in which the error occurred. It is described in the publications *IBM System/360 Operating System FORTRAN IV Programmer's Guide: FORTRAN IV (G),* Form C28-6639, or *FORTRAN IV (H),* Form C28-6602, and *IBM System/360 Disk Operating System: FORTRAN IV Programmer's Guide,* Form C28-6397. The following text lists the error messages in numeric order, explains the error, and indicates what action the system takes. In these explanations, *x* represents the argument supplied by the programmer and * represents the largest possible value that can be represented in floating-point notation.

IHC216I SLITE-SLITET X IS AN ILLEGAL VALUE
OA216I
ILF224I

Explanation: In the *xxx*FSLIT subprogram, a value of *i* that is not 0, 1, 2, 3, or 4 is an error; for *xxx*SLITE, a value for *i* that is not 1, 2, 3, or 4 is an error.

System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of no action for SLITE, and returning an OFF indication (i.e., J = 2) for SLITET.

IHC241I FIXPI INTEGER BASE = 0, INTEGER EXPONENT = X, LE 0
OA241I
ILF241I

Explanation: In the *xxx*FIXPI subprogram, a base number of zero and an exponent $\leqq 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC242I FRXPI REAL*4 BASE = 0.0, INTEGER EXPONENT = X, LE 0
OA242I
ILF242I

Explanation: In the *xxx*FRXPI subprogram, a base number of zero and an exponent $\leqq 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC243I FDXPI REAL*8 BASE = 0.0, INTEGER EXPONENT = X, LE 0
OA243I
ILF243I

Explanation: In the *xxx*FDXPI subprogram, a base number of zero and an exponent $\leqq 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC244I FRXPR REAL*4 BASE = 0.0, REAL*4 EXPONENT = X.X, LE 0
OA244I
ILF244I

Explanation: In the *xxx*FRXPR subprogram, a base number of zero and an exponent $\leqq 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC245I FDXPD REAL*8 BASE = 0.0 REAL*8 EXPONENT = X.X, LE 0
OA245I
ILF245I

Explanation: In the *xxx*FDXPD subprogram, a base number of zero and an exponent $\leqq 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC246I FCXPI COMPLEX*8 BASE = 0.0 + 0.0I, INTEGER EXPONENT = X, LE 0
OA246I
ILF246I

Explanation: In the *xxx*FCXPI subprogram, a base number of zero and an exponent $\leqq 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC247I FCDXI COMPLEX*16 BASE = 0.0 + 0.0I, INTEGER EXPONENT = X, LE 0
OA247I
ILF247I

Explanation: In the *xxx*FCDXI subprogram, a base number of zero and an exponent $\leqq 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC251I SQRT NEGATIVE ARGUMENT = X
OA251I
ILF251I

Explanation: In the *xxx*SSQRT subprogram, a value of $x < 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $|x|^{1/2}$.

IHC252I EXP ARG = X.X, GT 174.673
OA252I
ILF252I

Explanation: In the *xxx*SEXP subprogram, a value of $x > 174.673$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC253I ALOG-ALOG10 ARG = X.X, LE = 0
OA253I
ILF253I

Explanation: In the *xxx*SLOG subprogram, a value of $x \leqq 0$ is an error. Because this subprogram is also called by an exponentiation subprogram, this message also indicates that an attempt has been made to raise a negative real base to a power.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to * if $x = 0$, and to $\log |x|$ or $\log_{10} |x|$ (depending on the subprogram called) if x is less than 0.

IHC254I SIN-COS /ARG/ = /X.X (HEX = X)/, GE PI*2**18
OA254I
ILF254I

Explanation: In the *xxx*SSCN subprogram, a value of $|x| \geqq 2^{18} \cdot \pi$ is an error.

System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $\sqrt{2}/2$.

IHC255I ATAN2 ARGUMENTS = 0.0
OA255I
ILF255I

Explanation: In the *xxx*SATN2 subprogram when entry name ATAN2 is used, a value of $x_1 = x_2 = 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC256I SINH-COSH /ARG/ = X.X/, GE 174.673
OA256I
ILF256I

Explanation: In the *xxx*SSCNH subprogram, a value of $|x| \geqq 174.673$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC257I ARSIN-ARCOS /ARG/ = /X.X/ GT 1
OA257I
ILF257I

Explanation: In the *xxx*SASCN subprogram, a value of $|x| > 1$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC258I TAN-COTAN /ARG/ = /X.X (HEX = X)/, GE PI*218**
OA258I
ILF258I

Explanation: In the *xxx*STNCT subprogram, a value of $|x| \geqq 2^{18} \cdot \pi$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 1.

IHC259I TAN-COTAN /ARG/ = /X.X (HEX = X)/, APPROACHES SINGULARITY
OA259I
ILF259I

Explanation: In the *xxx*STNCT subprogram, a value of x too close to one of the singularities ($\pm\frac{\pi}{2}, \pm\frac{3\pi}{2}, \ldots$ for the tangent; $\pm\pi, \pm 2\pi, \ldots$ for the cotangent) is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution. continues and consists of setting the functional value to *.

IHC261I DSQRT NEGATIVE ARGUMENT = X.X
OA261I
ILF261I

Explanation: In the *xxx*SQRT subprogram, a value of $x < 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $|x|^{1/2}$.

IHC262I DEXP ARG = X.X, GT 174.673
OA262I
ILF262I

Explanation: In the *xxx*LEXP subprogram, a value of $x > 174.673$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC263I DLOG-DLOG10 ARG = X.X, LE ZERO
OA263I
ILF263I

Explanation: In the *xxx*LLOG subprogram, a value of $x \leqq 0$ is an error. Because this subprogram is also called by an exponentiation subprogram, this message also indicates that an attempt has been made to raise a negative real number to a power.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $-$ * if $x = 0$ and to $\log |x|$ or $\log_{10} |x|$ (depending on the subprogram called) if x is less than 0.

IHC264I DSIN-DCOS /ARG/ = /X.X (HEX = X)/, GE PI*2**50
OA264I
ILF264I

Explanation: In the *xxx*LSCN subprogram, a value of $|x| \geqq 2^{50} \cdot \pi$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $\sqrt{2}/2$.

IHC265I DATAN2 ARGUMENTS = 0.0
OA265I
ILF265I

Explanation: In the *xxx*LATN2 subprogram when entry name DATAN2 is used, a value of $x_1 = x_2 = 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC266I DSINH-DCOSH /ARG/ = /X.X/, GE 174. 673
OA266I
ILF266I

Explanation: In the *xxx*LSCNH subprogram, a value of $|x| \geqq 174.673$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC267I DARSIN-DARCOS /ARG/ = /X.X/, GT 1
OA267I
ILF267I

Explanation: In the *xxx*LASCN subprogram, a value of $|x| > 1$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 0.

IHC268I DTAN-DCOTAN /ARG/ = /X.X (HEX = X)/ GE PI* (2**50)
OA268I
ILF268I

Explanation: In the *xxx*LTNCT subprogram, a value of $|x| \geqq 2^{50} \cdot \pi$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to 1.

IHC269I DTAN-DCOTAN /ARG/ = /X.X (HEX = X)/, APPROACHES SINGULARITY
OA269I
ILF269I

Explanation: In the *xxx*LTNCT subprogram, a value of x too close to one of the singularities ($\pm\frac{\pi}{2}, \pm\frac{3\pi}{2}, \ldots$ for the tangent; $\pm\pi, \pm 2\pi, \ldots$ for the cotangent) is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC271I CEXP REAL ARG = X.X (HEX =X), GT 174.673
OA271I
ILF271I

Explanation: In the *xxx*CSEXP subprogram, a value of $x_1 > 174.673$ is an error.

System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to * ($\cos x - i\sin x$) where x is the imaginary part of the argument.

IHC272I CEXP IMAG ARG = X (HEX = X), ABS VALUE GE PI*218**
OA272I
ILF272I

Explanation: In the *xxx*CSEXP subprogram, a value of $|x_2| \geqq 2^{18} \cdot \pi$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $0 + 0i$.

IHC273I CLOG ARGUMENT = 0.0 + 0.0I
OA273I
ILF273I

Explanation: In the *xxx*CSLOG subprogram, a value of $x_1 = x_2 = 0$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $-$ * $+ 0i$.

IHC274I CSIN-CCOS/REAL ARG/ = /X.X (HEX = X), GE PI*218**
OA274I
ILF274I

Explanation: In the *xxx*CSSCN subprogram, a value of $|x_1| \geqq 2^{18} \cdot \pi$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $0 + 0i$.

IHC275I CSIN-CCOS/IMAG ARG/ = X.X (HEX = X)/, GT 174.673
OA275I
ILF275I

Explanation: In the *xxx*CSSCN subprogram, a value of $|x_2| > 174.673$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value as follows: If imaginary part > 174.673 (x is real part of argument): for sine, result $= \frac{*}{2}$ ($\sin x + i\cos x$); for cosine, result $= \frac{*}{2}$ ($\cos x - i\sin x$). If imaginary part < -174.673 (x is real part of argument): for sine, result $= \frac{*}{2}$ ($\sin x - i\cos x$); for cosine, result $= \frac{*}{2}$ ($\cos x + i\sin x$).

IHC281I CDEXP REAL ARG = X.X (HEX = X) GT 174.673
OA281I
ILF281I

Explanation: In the *xxx*CLEXP subprogram, a value of $x_1 > 174.673$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to * ($\cos x + i\sin x$) where x is the imaginary part of the argument.

IHC282I CDEXP IMAG ARG = X.X (HEX = X) ABS VALUE GE PI*250**
OA282I
ILF282I

Explanation: In the *xxx*CLEXP subprogram, a value of $|x_2| \geqq 2^{50} \cdot \pi$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated. *System Action With Extended Error Message Facility:* Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $0 + 0i$.

IHC283I CDLOG ARGUMENT = 0.D0 + 0.D0I
OA283I
ILF283I

Explanation: In the *xxx*CLLOG subprogram, a value of $x_1 = x_2 = 0$ is an error.

System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $- * + 0i$.

IHC284I CDSIN-CDCOS /REAL ARG/ = /X.X (HEX = X)/, GE PI*250**
OA284I
ILF284I

Explanation: In the *xxx*CLSCN subprogram, a value of $|x_1| \geqq 2^{50} \cdot \pi$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to $0 + 0i$.

IHC285I CDSIN-CDCOS /IMAG ARG/ = /X.X (HEX = X)/, GT 174.673
OA285I
ILF285I

Explanation: In the *xxx*CLSCN subprogram, a value of $|x_2| > 174.673$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value as follows: If imaginary part > 174.673 (x is real part of argument): for sine, result $= \frac{*}{2}$ (SIN x + iCOS x); for cosine, result $= \frac{*}{2}$ (COS x − iSIN x). If imaginary part < -174.673 (x is real part of argument): for sine, result $= \frac{*}{2}$ (SIN x − iCOS x); for cosine, result $= \frac{*}{2}$ (COS x + iSIN x).

IHC290I GAMMA ARG = X.X (HEX = X), LE 2 −252 OR GE 57.5744**
OA290I
ILF290I

Explanation: In the *xxx*SGAMA subprogram for the gamma function, a value of $x \leqq 2^{-252}$ or $x \geqq 57.5744$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC291I ALGAMA ARG = X.X (HEX = X), LE 0 OR GE 4.2937*1073**
OA291I
ILF291I

Explanation: In the *xxx*SGAMA subprogram for the log-gamma function, a value of $x \leqq 0$ or $x \geqq 4.2937 \cdot 10^{73}$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC300I DGAMMA ARG = X.X (HEX = X), LE 2 −252 OR GE 57.5744**
OA300I
ILF300I

Explanation: In the *xxx*LGAMA subprogram for the gamma function, a value of $x \leqq 2^{-252}$ or $x \geqq 57.5744$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

IHC301I DLGAMA ARG = X.X (HEX = X), LE 0 OR GE 4.2937*1073**
OA301I
ILF301I

Explanation: In the *xxx*LGAMA subprogram for the log-gamma function, a value of $x \leqq 0$ or $x \geqq 4.2937 \cdot 10^{73}$ is an error.
System Action Without Extended Error Facility: Execution of this load module or phase is terminated.
System Action With Extended Error Message Facility: Execution continues contingent upon the error occurrence counts kept in the Option Table. Standard corrective action is taken before execution continues and consists of setting the functional value to *.

Appendix D. Storage Estimates

Appendix D contains decimal storage estimates (in bytes) for the library subprograms. The estimate given does not include any additional library subprograms or FORTRAN execution-time routines that the subprogram needs during execution. The names of any additional library subprograms needed are given in Table 15 in the column headed "Additional Subprograms."

Some library subprograms also require execution-time routines for input/output, interruption, and error procedures.

If the programmer has not made allowances for the storage required by any of these additional routines (see Tables 17, 18 and 18.1), the amount of available storage may be exceeded and execution cannot begin. The programmer must add the estimates for all subprograms and routines needed to determine the amount of storage required.

System/360 Operating System

Note: The names of execution-time routines sometimes vary according to whether or not the Extended Error Message facility is in effect. In the following discussion, the names that are used are those when the facility has not been specified. Table 19 presents a cross listing of names for both circumstances.

The IHCFIOSH routine performs input/output procedures for both FORTRAN IV (E) and FORTRAN IV. [This routine refers to a table (IHCUATBL) for information about the input/output devices used during execution.] The IHCFCOME routine performs interruption and error procedures for FORTRAN IV (E) library subprograms; the IHCFCOMH, IHCFCVTH, IHCFINTH, IHCTRCH, and IHCUOPT routines perform the procedures for FORTRAN IV library subprograms. If a system contains both compilers, the IHCFCOMH-IHCFCVTH routines are used. Tables 15 and 16 indicate which library subprograms require these execution-time routines.

In addition, several other execution-time routines may be needed to resolve external references in a FORTRAN IV object module.

1. If a source module specifies direct access input/output operations, the compiler generates a call to the IHCDIOSE routine.
2. At the point that errors are encouraged during compilation, the compiler generates a call to an error

Table 15. Mathematical Subprogram Storage Estimates

Subprogram Name	Decimal Estimates OS	44PS	DOS	Additional Subprograms	Uses Input/Output and Interruption Routines
xxxCLABS	192	200	144	xxxLSQRT	Yes
xxxCLAS	240	220	208		No
xxxCLEXP	640	280	208	xxxLEXP, xxxLSCN	Yes
xxxCLLOG	464	310	224	xxxCLABS, xxxLSQRT, xxxLLOG, xxxLATN2	Yes
xxxCLSCN	840	500	408	xxxLEXP, xxxLSCN	Yes
xxxCLSQT	224	240	168	xxxLSQRT	Yes
xxxCSABS	184	190	136	xxxSSQRT	Yes
xxxCSAS	216	200	208		No
xxxCSEXP	600	280	208	xxxSEXP, xxxSSCN	Yes
xxxCSLOG	432	290	216	xxxSABS, xxxSSQRT, xxxSLOG, xxxSATN2	Yes
xxxCSSCN	760	440	344	xxxSEXP, xxxSSCN	Yes
xxxCSSQT	208	230	168	xxxSSQRT	Yes
IHCFAINT	80	—	—		No
xxxFCDXI	560	370	288	xxxCLAS	Yes
xxxFCXPI	536	350	272	xxxCSAS	Yes
xxxFDXPD	464	240	176	xxxLLOG, xxxLEXP	Yes
xxxFDXPI	368	200	168		Yes
IHCFIFIX	136	—	—		No
xxxFIXPI	368	230	184		Yes
xxxFMAXD	120	170	120		No
xxxFMAXI	224	290	224		No
xxxFMAXR	224	290	216		No
IHCFMODR	120	—	—		No
IHCFMODI	56	—	—		No
xxxFRXPI	360	190	168		Yes
xxxFRXPR	432	250	168	xxxSLOG, xxxSEXP	Yes
xxxLASCN	664	520	400	xxxLSQRT	Yes
IHCLATAN	352	—	—		No
xxxLATN2	680	520	528		Yes
xxxLERF	864	920	800	xxxLEXP	Yes
xxxLEXP	680	480	480		Yes
xxxLGAMA	1088	820	728	xxxLLOG, xxxLEXP	Yes
xxxLLOG	616	430	392		Yes
xxxLSCN	680	390	416		Yes
xxxLSCNH	584	400	304	xxxLEXP	Yes
xxxLSQRT	360	160	160		Yes
xxxLTANH	376	350	312	xxxLEXP	Yes
xxxLTNCT	776	400	416		Yes
xxxSASCN	536	380	312	xxxSSQRT	Yes
IHCSATAN	216	—	—		No
xxxSATN2	520	380	384		Yes
xxxSERF	504	560	432	xxxSEXP	Yes
xxxSEXP	480	340	304		Yes
xxxSGAMA	840	600	488	xxxSLOG, xxxSEXP	Yes
xxxSLOG	488	280	288		Yes
xxxSSCN	536	270	304		Yes
xxxSSCNH	504	340	248	xxxSEXP	Yes
xxxSSQRT	368	180	184		Yes
xxxSTANH	296	280	224	xxxSEXP	Yes
xxxSTNCT	672	310	320		Yes

routine (IHCIBERR for FORTRAN IV (E) and IHCIBERH for FORTRAN IV). If execution of the load module is attempted, the error routine is called, a message is issued, and the execution is terminated.

3. If a FORTRAN IV (E) source module contains a computed GO TO, the compiler generates a call to the HCCGOTO routine.
4. If a FORTRAN IV source module contains any input/output operations that refer to a NAMELIST name, compiler generates a call to the IHCNAMEL routine.
5. If a FORTRAN IV source module uses the debug facility, the compiler generates a call to the IHCDBUG routine.
6. If boundary alignment was specified during system generation, the IHCADJST routine will be loaded if a boundary-alignment error occurs.

Model 44 Programming System

In the FORTRAN library of the Model 44 Programming System, the BOAFIOCS routine is the interface with the system input/output services. This routine refers to a table, BOAUNITB, for information about the input/output devices used during execution. The BOAIBCOM routine performs interruption and error procedures. If a source program contains any input/output operation(s) referring to a NAMELIST name, the compiler generates a call to the BOANAMEL routine.

Disk Operating System

The ILFFIOCS routine performs input/output procedures for FORTRAN IV under the Disk Operating System. This routine refers to a table, either ILFUNTAB or ILFGHTAB, for information about the input/output devices used during execution. The ILFIBCOM, ILFADCOM, and ILFFINT routines perform input/output preparation, conversion, and interrupt processing for the library subprograms.

The ILFACOM routine is used, when necessary, for communication with DOS Basic FORTRAN modules. ILFTRBK provides traceback diagnostic messages.

In addition, several other execution time routines may be needed to resolve external references.

1. If a FORTRAN module requires direct access input/output operations, the compiler generates a call to ILFDIOCS.
2. When errors are encountered during compilation, the compiler generates a call to ILFIBERR. If execution of an incorrect statement is attempted, this routine is called, a message is issued, and execution is terminated.
3. If a FORTRAN source module specifies a NAMELIST operation, the compiler generates a call ILFNAMEL routine.
4. If a FORTRAN source module uses the debug facility, the compiler generates a call to the ILFDEBUG routine.

• Table 16. Service Subprogram Storage Estimates

Subprogram Name	Decimal Estimates			Uses Input/Output and Interruption Routines
	OS	44PS	DOS	
xxxFDVCH	80	120	80	Yes
xxxFDUMP	544	760	496	Yes
xxxFEXIT	32	40	32	Yes
xxxFOVER	88	130	88	Yes
xxxFSLIT	384	280	208	Yes

Table 17. Execution-Time Routine Storage Estimates, Operating System

Routine Name	Decimal Estimate	Used By
IHCADJST	1,156	FORTRAN IV
IHCCGOTO	60	FORTRAN IV (E)
IHCDBUG	2,152	FORTRAN IV
IHCDIOSE	2,688 (See Note 1)	Both
IHCFCOME	6,196	FORTRAN IV (E)
IHCFCOMH	4,168	FORTRAN IV
IHCCOMH2	520	FORTRAN IV
IHCFCVTH	4,688	FORTRAN IV
IHCFIOSH	3,744 + IHCUATBL (See Notes 2 and 3)	Both
IHCIBERH	224	FORTRAN IV
IHCIBERR	136	FORTRAN IV (E)
IHCNAMEL	2,880	FORTRAN IV
IHCTRCH	792	FORTRAN IV
IHCUOPT	8	FORTRAN IV

NOTE 1: This module also acquires dynamic storage. Its amount, in bytes, may be computed by the formula

184 + buffer size

Each buffer value should be 800, except for a card read punch which is 80 and a printer which is 133 FORTRAN utilizes double buffering.

NOTE 2: This module also acquires dynamic storage. Its amount, in bytes, may be computed by the formula

128 + buffer size

Buffer lengths are listed in Note 1.

NOTE 3: The number of bytes in table IHCUATBL may be computed by the formula

$12n + 8$

where n is the number of data set reference numbers requested during system generation.

- Table 18. Execution-Time Routine Storage Estimate, Model 44 System

Routine Name	Decimal Estimate
BOADIOCS	688
BOAFIOCS	1,776
BOAIBCOM	10,432
BOANAMEL	2,432
BOAUNITB	(see Note 1)
BOAUOPT	8
BNXADJST	1,000 (Note 2)

Note 1: The number of bytes in CSECT BOAUNITB may be computed by the formula

$$8n + 8$$

where n is the number of data set reference numbers requested during system construction.

Note 2: BNXADJST does not reside in the library of relocatable modules, but in the absolute phase library.

- Table 18.1. Execution-Time Storage Estimates, Disk Operating System

Routine Name	Decimal Estimate
ILFACOM	1168
ILFADCON	4309
ILFDEBUG	1592
ILFDIOCS	648
ILFFINT	1392
ILFFIOCS	3722
ILFGHTAB	272
ILFIBCOM	4336
ILFIBERR	200
ILFNAMEL	2222
ILFTRBK	592
ILFUNTAB	272

Table 19. Execution-Time Routines Storage Estimates With Extended Error Message Facility, Operating System

Routine Name	Corresponding Module With No Error Message Facility	Storage Estimate
IHCADJST	IHCADJST	1,156
IHCDBUG	IHCDBUG	2,152
IHCECOMH	IHCFCOMH	5,368
IHCCOMH2	IHCCOMH2	1,120
IHCEDIOS	IHCDIOSE	3,848 (See Note 1)
IHCEFIOS	IHCFIOSH	4,584 + IHCUATBL (See Notes 2 and 3)
IHCEFNTH	IHCFINTH	1,368
IHCERRM	—	1,512
IHCETRCH	IHCTRCH	706
IHCFCVTH	IHCFCVTH	4,688
IHCFOPT	—	824
IHCIBERH	IHCIBERH	224
IHCNAMEL	IHCNAMEL	2,880
IHCUOPT	IHCUOPT	800 + 8n (See Note 4)

NOTE 1: This module also acquires dynamic storage. Its amount, in bytes, may be computed by the formula

$$184 + \text{buffer size}$$

Each buffer value should be 800, except for a card read punch which is 80 and a printer which is 133. FORTRAN utilizes double buffering.

NOTE 2: This module also acquires dynamic storage. Its amount, in bytes, may be computed by the formula

$$128 + \text{buffer size}$$

Buffer lengths are listed in Note 1.

NOTE 3: The number of bytes in table IHCUATBL may be computed by the formula

$$12n + 8$$

where n is the number of data set reference numbers requested during system generation.

NOTE 4: The number of additional entries supplied in the Option Table during system generation is represented by n.

Appendix E. Assembler Language Information

The mathematical and service subprograms in the FORTRAN IV library are available to the assembler language programmer. The following text explains the method of calling a library subprogram in an assembler language program, and then gives additional information necessary to use each type of subprogram. (The assembler language programmer should also be familiar with the information contained in Appendix D.)

Calling Sequences

To call either type of library subprogram, the assembler language programmer supplies an entry name, an argument list, and an area used by the subprogram to store information (i.e., a save area). The following conventions must be observed when calling a library subprogram in an assembler language program:

1. The address of the entry name must be in general register 15.
2. The address of the point of return to the calling program must be in general register 14.
3. The address of the argument list must be in general register 1.
4. The argument list must be assembled on a full-word boundary; it consists of one 4-byte address constant for each argument. The last argument must have a 1 in its high order bit.
5. The address of the save area must be in general register 13.
6. The save area must be assembled on a full-word boundary. Although the minimum size of the save area depends upon the subprogram, the programmer is advised to use a save area of 18 full-words for all library subprograms. The minimum save area sizes are given in Tables 2 through 6 for the mathematical subprograms, and in Table 17 for the service subprograms.
7. If the information in a floating-point register is to be retained, the programmer must save and restore the contents of the register. The subprograms that make use of the floating-point registers contain no provisions for saving the information.
8. If a main program in assembler language contains any calls to those library subprograms that use the FORTRAN execution-time routines (see Appendix D), the following instructions must be included before the call to the subprogram is issued:

OS	44PS
L 15,=V(IBCOM#) BAL 14,64(15)	EXTRN IBCOM# ..... L 15,=A(IBCOM#) BAL 14,64(15)

These instructions cause the initialization of return coding and the interruption exceptions described in Appendix C. If these instructions are omitted, the occurrence of an interruption or an error causes unpredictable termination of the execution of this load module.

NOTE: In an assembler language program, a decimal-divide exception may occur. This causes the character B to appear in the program interruption message described in Appendix C.

The user of System/360 Operating System may use several methods to call a FORTRAN library subprogram: the appropriate macro-instructions described in the publication *IBM System/360 Operating System: Supervisor and Data Management Macro-Instructions*, Form C28-6647 or the general assembler language calling sequence (given in Figure 3). If the macro-instructions are used, the address of the save area must be placed in general register 13 before using a macro-instruction to give control to the subprogram. For example, if the square root of the value in AMNT is to be computed and SAVE is the address of the same area, the following statements could be included in an assembler language program to call the IHCSSQRT subprogram:

```
            L       15,=V(IBCOM#)
            BAL     14,64(15)
              .
            LA      13,SAVE
            CALL    SQRT,(AMNT),VL
              .
              .
              .
SAVE        DS      18F
```

If the general assembler language calling sequence shown in Figure 3 is used, the programmer must ensure that all of the conventions discussed previously are followed. For example, to call the IHCSSQRT subprogram to compute the square root of the number in AMNT, the following statements would be included in the source program:

```
            LA      13,SAVE
            LA      1,ARG
            L       15,ENTRY
            BALR    14,15
              .
              .
              .
ENTRY       DC      V(SQRT)
              .
              .
              .
SAVE        DS      18F
              .
              .
              .
ARG         DC      X'80'
            DC      AL3(AMNT)
```

	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 the binary calling sequence identifier. 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	*xx*F	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 full-words for all subprograms. (The minimum save area requirements are given in Tables 2 through 6 for the mathematical subprograms and in Table 16 for the service subprograms.)
	* * * *		
	CNOP		Aligns the argument list at a full-word boundary.
arglist	DC	X'80'	Indicates the first byte of the only argument.
	DC	AL3 (arg)	Contains the address of the argument.
	or 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'	Indicates the first byte of the last argument.
	DC	AL3 (arg_n)	Contains the address of the last argument.

Figure 3. General Assembler Language Calling Sequence

When the load module is executed, the IHCSQRT subprogram is called to compute the square root of the number in AMNT; the result is stored in floating-point register 0. The binary calling sequence identifier is not used.

The assembler language user of the Model 44 Programming System will use the calling sequence given in Figure 3. He will, however, use only the A-type address constant where the choice between that and the V-type is given. As the note in the figure states, the label in the operand portion of the address constant must be made the object of an EXTRN statement to obtain proper linkage.

Mathematical Subprograms

The assembler language programmer supplies one or more arguments for each mathematical subprogram. The arguments may be either integer values or normalized floating-point real or complex values.

An integer argument is four bytes in length and starts on a full-word boundary. A real argument is either four or eight bytes in length. The four-byte argument starts on a full-word boundary. The eight-byte argument starts on a double-word boundary and occupies two adjacent words. The first word contains the most significant digits. This word is also the address of the entire argument; the second word contains the least significant digits.

A complex argument is either eight or sixteen bytes in length and starts on a double-word boundary. The first half of the argument contains the real part of the complex argument; the second half contains the imaginary part. The address of the real part of the argument is the address of the entire argument.

Each mathematical subprogram returns a single answer. This answer is either an integer value or a normalized floating-point real or complex value. An integer answer is stored in general register 0, a real answer is stored in floating-point register 0, and a complex answer is stored in floating-point registers 0 and 2.

Tables 2 through 6 contain additional information for using the mathematical subprograms in an assembler language program. These tables give the floating-point

Table 20. Assembler Information for the Service Subprograms

Subprogram Name	Entry Name(s)	Save Area (Full Words)
xxxFDUMP	DUMP	18
	PDUMP	18
xxxFDVCH	DVCHK	10
xxxFEXIT	EXIT	5
xxxFOVER	OVERFL	10
xxxFSLIT	SLITE	9
	SLITET	10

registers that are used by the subprogram and the save area required by the subprogram.

Service Subprograms

The service subprograms do not use the floating-point registers during execution; however, each service subprogram requires a save area. The minimum size of the save area depends upon the subprogram to be used and is given in Table 20.

Appendix F. Sample Storage Printouts

A sample printout is given below for each dump format that can be specified for the *xxx*FDUMP 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 4). Note that the headings on the printouts are not generated by the system, but were obtained by using FORMAT statements.

The output of the *xxx*FDUMP subprogram (for both entry names, DUMP 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     42B00000  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
```

Figure 4. Sample Storage Printouts

Index

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

Title: IBM System/360
FORTRAN IV
Library Subprograms

Form: C28-6596-4

Your comments assist us in improving the usefulness of our publications; they are an important part of the input used for technical newsletters and revisions.

Please do not use this form for technical questions about the system; it only delays the response. Instead, direct your technical questions to your local IBM representative.

Corrections or clarifications needed:

Page	Comment

Please indicate in the space below if you wish a reply:

Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

8-6596-4

cut along this line

'OUR COMMENTS, PLEASE . . .

This publication is one of a series that serves as a reference source for systems analysts programmers, and operators of IBM systems. Your answers to the questions on the back of his form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

PLEASE NOTE: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

old fold

FIRST CLASS
PERMIT NO. 33504
NEW YORK, N.Y.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION

1271 AVENUE OF THE AMERICAS
NEW YORK, N.Y. 10020

Attention: PUBLICATIONS

fold fold

IBM®

nternational Business Machines Corporation
Data Processing Division
12 East Post Road, White Plains, N.Y. 10601
USA Only]

BM World Trade Corporation
121 United Nations Plaza, New York, New York 10017
International]