UNISYS

OS/3 FORTRAN IV

Programming Quick-Reference Guide

Relative to Release Level 8.0

Priced Item

August 1987

Printed in U S America

UP-8815

UNISYS

OS/3 FORTRAN IV

Programming Quick-Reference Guide

Previous Title: OS/3 FORTRAN IV Summary

Relative to Release Level 8.0

August 1987

Printed in U S America

Priced Item

UP-8815

NO WARRANTIES OF ANY NATURE ARE EXTENDED BY THE DOCUMENT. Any product and related material disclosed herein are only furnished pursuant and subject to the terms and conditions of a duly executed Program Product License or Agreement to purchase or lease equipment. The only warranties made by Unisys, if any, with respect to the products described in this document are set forth in such License or Agreement. Unisys cannot accept any financial or other responsibility that may be the result of your use of the information in this document or software material, including direct, indirect, special or consequential damages.

You should be very careful to ensure that the use of this information and/or software material complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

PAGE STATUS SUMMARY

ISSUE: Update B - UP-8815
RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level
Cover		B
Title Page/Disclaimer		B*
PSS	1	B
Contents	1	Orig.
Text	1 thru 42 43 44 thru 56	Orig. A Orig.
User Comment Form		

*New page

Contents

Evaluation of FORTRAN Operations

Type	Operator	Meaning	Hierarchy
Functions		Evaluation of functions	1 (highest)
Arithmetic	**	Exponentiation	2
	*	Multiplication	3
	/	Division	3
	+	Addition	4
	–	Subtraction	4
Relational	.GT.	Greater than ($>$)	5
	.GE.	Greater than or equal to ($\geq$)	5
Relational (cont)	.LT.	Less than ($<$)	5
	.LE.	Less than or equal to ($\leq$)	5
	.EQ.	Equal to ($=$)	5
	.NE.	Not equal to ($\neq$)	5
Logical	.NOT.	Logical NOT	6
	.AND.	Logical AND	7
	.OR.	Logical inclusive OR	8

Classification of Data Types in FORTRAN

Type	Bytes Allocated		Definition
	Standard	Optional	
Integer	4	2	A number with no fractional part $-2{,}147{,}483{,}648 \leqslant \text{integer} \leqslant 2{,}147{,}483{,}647$, $-32768 \leqslant \text{integer}*2 \leqslant 32767$.
Real	4	8	A number with a fractional portion. Maximum value is $\pm\ 7{,}237005*10^{75}$. Minimum value is $\pm\ 0.5397605*10^{78}$. Accuracy is approximately 7 decimal digits for REAL*4 and 17 decimal digits for REAL*8.
Double precision	8	None	Same as REAL*8.
Complex	8	16	A pair of real numbers; the first is the real part and the second the imaginary part of the complex value.
Logical	4	1	The logical values FALSE or TRUE, or F or T, respectively. Zero represents false and nonzero true. Only the first byte is significant.
Literal①	$(0 < X \leqslant 255)$	None	Any symbols that can be represented as a Hollerith constant or as a character string enclosed in apostrophes.

① No variable type is associated with literal or hexadecimal data.

Syntax Conventions

Brackets, [ ], enclose words and phrases used or omitted as required.

Braces, { }, indicate a choice of two or more variant forms.

Ellipsis, ..., indicates where repetition may occur at programmer's option.

Underscore, ——, indicates FORTRAN IV default.

Determining the Mode of an Expression

		First Operand: Type (Length)					
		Integer (2)	Integer (4)	Real (4)	Real (8)	Complex (8)	Complex (16)
Second Operand: Type (Length)	Integer (2)	Integer (4)	Integer (4)	Real (4)	Real (8)	Complex (8)	Complex (16)
	Integer (4)	Integer (4)	Integer (4)	Real (4)	Real (8)	Complex (8)	Complex (16)
	Real (4)	Real (4)	Real (4)	Real (4)	Real (8)	Complex (8)	Complex (16)

	Real (8)	Real (8)	Real (8)	Real (8)	Real (8)	Complex (16)	Complex (16)
	Complex (8)	Complex (8)	Complex (8)	Complex (8)	Complex (16)	Complex (8)	Complex (16)
	Complex (16)	Complex (16)	Complex (16)	Complex (16)	Complex (16)	Complex (16)	Complex (16)

Organizational Statements

PROGRAM program-name ①

BLOCK DATA [subprog-name]

ENTRY entry-point name [(dummy-arg-1,...,dummy-arg-n)] ②

[{ INTEGER / REAL / DOUBLE PRECISION / COMPLEX / LOGICAL }] FUNCTION function-name [*length] ③ ({ variable-name-arg / array-name-arg / proc-name-arg } ...) ④

function-name (dummy-arg-name-1,... [,dummy-arg-name-n] ...) = { arith-expression / **logical-expression** }

NOTES:

○ 1—6 character alphanumeric string, beginning with alphabetic character

○ There need not be any arguments if the ENTRY is to a SUBROUTINE. An ENTRY into an external function subprogram must specify at least one argument.

○ Length is not allowed with DOUBLE PRECISION type.

○ Variable-names used as arguments may be enclosed in slashes to indicate call by name.

SUBROUTINE subroutine-name [((variable-name-1① | array-name-1 | proc-name-1 | *②) ... [,variable-name-n① | array-name-n | proc-name-n | *②])]

END

DATA variable-name | array-name | array-elem-name | DO-implied list [[variable-name | array-name | array-elem-name | DO-implied list] ...] / [integer-constant*] constant [, [integer-constant*] constant] .../

[variable-name | array-name | array-elem-name | DO-implied list / [integer-constant*] constant [, [integer-constant*] constant] .../] ...

NAMELIST/namelist-name/ { variable-name / array-name } [, { variable-name / array-name } ...] [/namelist-name/ { variable-name / array-name } [, { variable-name / array-name } ...] ...]

stmt-label FORMAT ([/...] field-descriptor-1 { / / , ③ } ... field-descriptor-n [/...])

NOTES:

① Variable-names may be enclosed in slashes to specify call-by-name.

② Represents dummy label arguments that correspond to CALL statement label arguments

③ Commas are not required when they follow fields described by blank, Hollerith, and literal descriptors.

Organizational Statements (cont)

ABNORMAL [function-name [,function-name] ...] ①

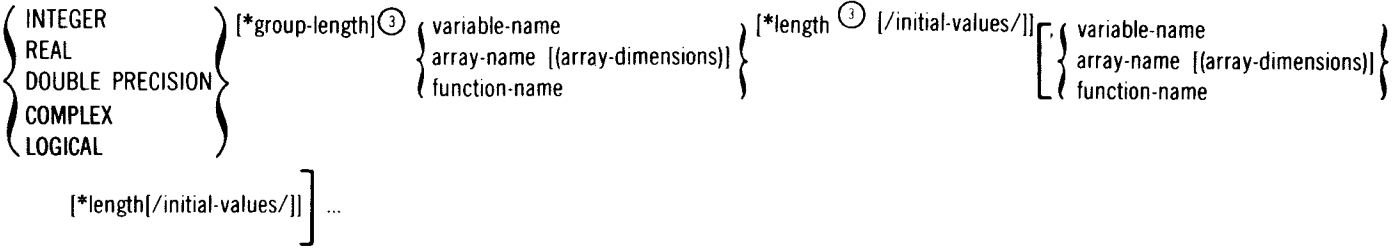
COMMON [/block-name/] { variable-name / array-name [(array-dimensions)] } [{ variable-name / array-name [(array-dimensions)] ... }]

[/block-name/] { variable-name / array-name [(array-dimensions)] } [, { variable-name / array-name [(array-dimensions)] } ...]...

DIMENSION array-name (array-dimensions) [,array-name (array-dimensions)] ...

EQUIVALENCE (variable,variable[,variable]...) [,(variable,variable[,variable] ...)] ... ○

Explicit Type Statements



NOTES:

(1) An ABNORMAL statement with no function-name indicates that all functions are to be considered normal.

(2) The variables may have no subscripts, a single subscript, or multiple subscripts. Variables may be variable-names, array-element-names, or array-names.

(3) Length is not allowed for DOUBLE PRECISION.

```
EXTERNAL { external-subroutine-name } [ , { external-subroutine-name } ] ...
         { external-function-name   }   [   { external-function-name   } ]

IMPLICIT { INTEGER          } [*length] (1) ( { letter,letter,... } ) [ , { INTEGER          } [*length] (1) ( { letter,letter,... } ) ] ... (2)
         { REAL             }                 { letter-letter     }       { REAL             }                 { letter-letter     }
         { DOUBLE PRECISION }                                             { DOUBLE PRECISION }
         { COMPLEX          }                                             { COMPLEX          }
         { LOGICAL          }                                             { LOGICAL          }
```

NOTES:

(1) Length is not allowed for DOUBLE PRECISION.

(2) In the IMPLICIT statement, the character $ is treated as a letter of a symbolic name and may be part of a range only as the last character.

Assignment Statements

```
variable = { logical-expression    }
           { arithmetic-expression }

ASSIGN statement-label TO integer*4-variable
```

Sequential Input/Output Statements

BACKSPACE { integer-constant / integer-variable }

END FILE { integer-constant / integer-variable }

Formatted READ Statements

READ ({ integer-constant / integer-variable } { array-name / namelist-name / format-stmt-label / * ① } [{ EOF=stmt-label-1 / END=stmt-label-1 }] [,ERR=label-2]) [I/O-list]

Unformatted READ Statements

READ({ integer-constant / integer-variable } [, { EOF=stmt-label-1 / END=stmt-label-1 }] [,ERR=stmt-label-2]) [I/O-list] ②

NOTES:

① Indicates the use of list-directed I/O.

② For unformatted data set formats, see the OS/3 FORTRAN IV programmer reference, UP-8474 (current version).

REWIND { integer-constant / integer-variable }

Sequential Input/Output Statements (cont)

Formatted WRITE Statement

WRITE({ integer-constant / integer- variable } , { array-name / namelist-name / format-statement-label / * ① }) [I/O-list]

Unformatted WRITE Statement

WRITE ({ integer-constant / integer-variable }) [I/O-list] ②

NOTES:

① Indicates the use of list-directed I/O.

② For unformatted data set formats, see the OS/3 FORTRAN IV programmer reference, UP-8474 (current version).

Other Sequential Input/Output Statements

PRINT { format-statement-label / format-statement-array-name / *① },I/O-list

PUNCH { format-statement-label / format-statement-array-name / *① },I/O-list

READ { format-statement-label / format-statement-array-name / *① },I/O-list

Direct Access Input/Output Statements

DEFINE FILE file-id-number(number-of-records,maximum-record-size,{L / E / U},associated-variable)

[,file-id-number(number-of-records,maximum-record-size,{L / E / U},associated-variable)]...

FIND({integer-constant' / integer-variable'} record-position)

READ({integer-constant' / integer-variable'} record-position[,{format-statement-label / array-name / * ①}] [,ERR=statement-label][,END=statement-label])[list]

WRITE({integer-constant' / integer-variable'} record-position[,{format-statement-label / array-name / * ①}])[list]

NOTE:

① Specifies list-directed I/O.

CALL { user-subroutine-name / std-lib-subroutine-name } [(arg-1,..., arg-n)]

CONTINUE

DO statement-label control-variable=initial-value, limit [,increment]

GO TO statement-label

GO TO (statement-label-1 [,statement-label-n] ...), integer-variable ①

GO TO integer * 4-variable [,(statement-label-1 [,statement-label-n] ...)] ①

IF (arithmetic-expression) ② [statement-label] [,statement-label] [,statement-label]

IF (logical-expression) executable-statement ③

NOTES:

① Maximum 127 statement labels permitted ② A complex expression is not permitted.

③ The executable-statement may not be a DO, END, or another logical IF statement.

Control Statements (cont)

PAUSE [integer-constant ① | 'literal-constant' ②]

RETURN [integer-constant ③ | integer-variable]

STOP [integer-constant ① | 'literal-constant' ②]

TRACE ON

TRACE OFF

NOTES:

① May be 1—5 digits

② May not be more than 243 characters

③ This integer value points to a label parameter (signaled by an ampersand) in the argument list of the CALL statement.

Format Codes

Code	Function	Format
G	To read or write integer, real, complex, or logical data	[scale-factor] [repetition-factor] G field-length .digit count
I	To read or write integer data	[repetition-factor] I field-length
F	To read or write real data	[scale-factor] [repetition-factor] F field-length .digit count
E	To read or write real data with an E decimal exponent	[scale-factor] [repetition-factor] E field-length .digit-count
D	To read or write real data with a D decimal exponent	[scale-factor] [repetition-factor] D field-length .digit-count
Z	To read or write hexadecimal data	[repetition-factor] Z field-length
L	To read or write logical variables	[repetition-factor] L field-length

Format Codes (cont)

Code	Function	Format
A	To read or write characters	[repetition-factor] A field-length
H	To read or write literals	field-length H character-string
'...'	To read or write literals	'character-string'
X	To insert blanks on output or to skip characters on input	field-length X
T	To indicate the record position where the transfer of data is to begin.	T transfer-position
P	To serve as a scale factor	integer P

Intrinsic Functions

Generic Name	Use	Number Arguments	Member Function Name	Member Argument Type	Member Function Type
ABS	Determine the absolute value of the argument	1	ABS IABS JABS DABS	Real*4 Integer*4 Integer*2 Double precision	Real*4 Integer*4 Integer*2 Double precision
CABS	Determine the absolute value of the argument	1	CABS ① CDABS ①	Complex*8 Complex*16	Real*4 Double precision
AINT	Truncation; eliminate the fractional portion of argument	1	AINT DINT	Real*4 Double precision	Real*4 Double precision
INT	Truncation; eliminate the fractional portion of argument	1	INT IDINT	Real*4 Double precision	Integer*4 Integer*4

Intrinsic Functions (cont)

Generic Name	Use	Number Arguments	Member Function Name	Member Argument Type	Member Function Type
MOD	Remaindering; defined as $a_1-[x]a_2$, where [x] is the greatest integer whose magnitude does not exceed the magnitude of a_1/a_2 and whose sign is the same as a_1/a_2	2 (Argument 2 must be non-zero.)	JMOD AMOD MOD DMOD	Integer*2 Real*4 Integer*4 Double precision	Integer*2 Real*4 Integer*4 Double precision
{MAX MAX0}	Select the largest value	≥2	JMAX0 ① AMAX0 ② AMAX1 ① {MAX ① MAX0 ①} MAX1 ② DMAX1 ①	Integer*2 Integer*4 Real*4 Integer*4 Real*4 Double precision	Integer*2 Real*4 Real*4 Integer*4 Integer*4 Double precision

{MIN / MINO}	Select the smallest value	≥2	JMINO ①	Integer*2	Integer*2
			AMINO ②	Integer*4	Real*4
			AMIN1 ①	Real*4	Real*4
			{MIN ① / MINO ①}	Integer*4	Integer*4
			MIN1 ②	Real*4	Integer*4
			DMIN1 ①	Double precision	Double precision
	Convert argument from integer to real or double precision	1	FLOAT ②	Integer*4	Real*4
			DFLOAT ②	Integer*4	Double precision
			HFLOAT ②	Integer*2	Real*4
			DFLOAT ②	Integer*2	Double precision
	Convert argument from real to integer	1	IFIX ②	Real*4	Integer*4
			HFIX ②	Real*4	Integer*2

Intrinsic Functions (cont)

Generic Name	Use	Number Arguments	Member Function Name	Member Argument Type	Member Function Type
SIGN	Replace the algebraic sign of the first argument with the sign of the second argument	2	JSIBN SIGN ISIGN DSIGN	Integer*2 Real*4 Integer*4 Double precision	Integer*2 Real*4 Integer*4 Double precision
DIM	Positive difference; subtract the smaller of the two arguments from the first argument	2	JDIM DIM IDIM DDIM	Integer*2 Real*4 Integer*4 Double precision	Integer*2 Real*4 Integer*4 Double precision
SNGL	Convert double precision to real	1	SNGL CSNGL	Double precision Complex*16	Real*4 Complex*8
REAL	Get real part of a complex number	1	REAL DREAL	Complex*8 Complex*16	Real*4 Double precision

AIMAG IMAG	Get imaginary part of a complex number		IMAG AIMAG DIMAG	Complex*8 Complex*16	Real*4 Double precision
DBLE	Convert from real to double precision	1	DBLE CDBLE	Real*4 Complex*8	Double precision Complex*16
CMPLX	Convert two real arguments to a complex number	2	CMPLX DCMPLX	Real*4 Double precision	Complex*8 Complex*16
CONJG	Get conjugate of a complex number	1	CONJG DCONJG	Complex*8 Complex*16	Complex*8 Complex*16

NOTES:

① This function is an external procedure supplied in the FORTRAN IV library.

② This function is accessible only through its member name.

Standard Library Functions

General Operation	Generic Name	Member Name	Mathematical Definition	Argument			Function Value Type and Range
				Number	Type	Range	
Trigonometric	SIN	SIN	y=sin(x)	1	real*4 (in radians)	$\|x\| < (2^{18}.\pi)$	real*4 $-1 \leq y \leq 1$
		DSIN		1	real*8 (in radians)	$\|x\| < (2^{50}.\pi)$	real*8 $-1 \leq y \leq 1$
		CSIN	y=sin(z)	1	complex*8 (in radians)	$\|x_1\| < (2^{18}.\pi)$ $\|x_2\| \leq 174.673$	complex*8 $-M \leq y_1, y_2 \leq M$
		CDSIN		1	complex*16 (in radians)	$\|x_1\| < (2^{50}.\pi)$ $\|x_2\| \leq 174.673$	complex*16 $-M \leq y_1, y_2 \leq M$

COS	COS	y=cos(x)	1	real*4 (in radians)	$\|x\| < (2^{18}.\pi)$	real*4 $-1 \leq y \leq 1$
	DCOS		1	real*8 (in radians)	$\|x\| < (2^{50}.\pi)$	real*8 $-1 \leq y \leq 1$
	CCOS	y=cos(z)	1	complex*8 (in radians)	$\|x_1\| < (2^{18}.\pi)$ $\|x_2\| \leq 174.673$	complex*8 $-M \leq y_1, y_2 \leq M$
	CDCOS		1	complex*16 (in radians)	$\|x_1\| < (2^{50}.\pi)$ $\|x_2\| \leq 174.673$	complex*16 $-M \leq y_1, y_2 \leq M$
TAN	TAN	y=tan(x)	1	real*4 (in radians)	$\|x\| < (2^{18}.\pi)$	real*4 $-M \leq y \leq M$
	DTAN		1	real*8 (in radians)	$\|x\| < (2^{50}.\pi)$	real*8 $-M \leq y \leq M$

Standard Library Functions (cont)

General Operation	Generic Name	Member Name	Mathematical Definition	Argument			Function Value Type and Range
				Number	Type	Range	
Trigonometric (cont)	{COTAN / COT}	{COTAN / COT}	y=cotan(x)	1	real*4 (in radians)	$\|x\| < (2^{18}.\pi)$	real*4 $-M \leq y \leq M$
		{DCOTAN / DCOT}		1	real*8 (in radians)	$\|x\| < (2^{50}.\pi)$	real*8 $-M \leq y \leq M$
	{ASIN / ARSIN}	{ASIN / ARSIN}	y=arcsin(x)	1	real*4	$\|x\| \leq 1$	real*4 (in radians) $-\pi/2 \leq y \leq \pi/2$
		{DASIN / DARSIN}		1	real*8	$\|x\| \leq 1$	real*8 (in radians) $-\pi/2 \leq y \leq \pi/2$
	{ACOS / ARCOS}	{ACOS / ARCOS}	y=arccos(x)	1	real*4	$\|x\| \leq 1$	real*4 (in radians) $0 \leq y \leq \pi$
		{DACOS / DARCOS}		1	real*8	$\|x\| \leq 1$	real*8 (in radians) $0 \leq y \leq \pi$

	ATAN	ATAN	y=arctan(x)	1	real*4	any real argument	real*4 (in radians) $-\pi/2 \leq y \leq \pi/2$
		DATAN		1	real*8	any real argument	real*8 (in radians) $-\pi/2 \leq y \leq \pi/2$
	ATAN2	ATAN2	$y=\arctan\left(\frac{x_1}{x_2}\right)$	2	real*4	any real arguments except (0,0)	real*4 (in radians) $-\pi \leq y \leq \pi$
		DATAN2		2	real*8	any real arguments except (0,0)	real*8 (in radians) $-\pi \leq y \leq \pi$
Hyperbolic	SINH	SINH	$y=\frac{e^x-e^{-x}}{2}$	1	real*4	$\|x\| < 175.366$	real*4 $-M \leq y \leq M$
		DSINH		1	real*8	$\|x\| < 175.366$	real*8 $-M \leq y \leq M$

Standard Library Functions (cont)

General Operation	Generic Name	Member Name	Mathematical Definition	Argument			Function Value Type and Range
				Number	Type	Range	
Hyperbolic (cont)	SINH (cont)	CSINH	$y=\frac{e^{z}-e^{-z}}{2}$	1	complex*8	$\|x_1\|\leq 174.673$ $\|x_2\|<2^{18}.\pi$	complex*8 $-M\leq y_1,y_2\leq M$
		CDSINH		1	complex*16	$\|x_1\|\leq 174.673$ $\|x_2\|<2^{50}.\pi$	complex*16 $-M\leq y_1,y_2\leq M$
	COSH	COSH	$y=\frac{e^{x}+e^{-x}}{2}$	1	real*4	$\|x\|<175.366$	real*4 $1\leq y\leq M$
		DCOSH		1	real*8	$\|x\|<175.366$	real*8 $1\leq y\leq M$

		CCOSH	$y=\frac{e^{z}+e^{-z}}{2}$	1	complex*8	$\lvert x_1\rvert \leq 174.673$ $\lvert x_2\rvert < 2^{18}.\pi$	complex*8 $-M \leq y_1, y_2 \leq M$
		CDCOSH		1	complex*16	$\lvert x_1\rvert < 174.673$ $\lvert x_2\rvert < 2^{50}.\pi$	complex*16 $-M \leq y_1, y_2 \leq M$
	TANH	TANH	$y=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	1	real*4	any real argument	real*4 $-1 \leq y \leq 1$
		DTANH		1	real*8	any real argument	real*8 $-1 \leq y \leq 1$
Exponential	EXP	EXP	$y=e^{x}$	1	real*4	$x \geq -180.218$ $x \leq 174.673$	real*4 $0 \leq y \leq M$
		DEXP		1	real*8	$x \geq -180.218$ $x \leq 174.673$	real*8 $0 \leq y \leq M$
		CEXP	$y=e^{z}$	1	complex*8	$x_1 \leq 174.673$ $\lvert x_2\rvert \leq (2^{18}.\pi)$	complex*8 $-M \leq y_1, y_2 \leq M$
		CDEXP		1	complex*16	$x_1 \leq 174.673$ $\lvert x_2\rvert < (2^{50}.\pi)$	complex*16 $-M \leq y_1, y_2 \leq M$

Standard Library Functions (cont)

General Operation	Generic Name	Member Name	Mathematical Definition	Argument			Function Value Type and Range
				Number	Type	Range	
Base 10 Exponential	DEXP10	EXP10	$y = 10^x$	1	real*4	$x \geq -180.216/\ln(10)$ $x \leq 174.673/\ln(10)$	real*4 $0 \leq y \leq M$
		EXP10		1	real*8	$x \geq -180.216/\ln(10)$ $x \leq 174.673/\ln(10)$	real*8 $0 \leq y \leq M$
Natural logarithm		ALOG LOG	$y=\log_e x$ or $y=\ln(x)$	1	real*4	$x > 0$	real*4 $y \geq -180.218$ $y \leq 174.673$
		DLOG		1	real*8	$x > 0$	real*8 $y \geq -180.218$ $y \leq 174.673$

	{ALOG / LOG}	CLOG	$y=PVlog_e(z)$ PV is principal value, which means y_2 between $-\pi$ and π.	1	complex*8	$z \neq 0+0i$	complex*8 $y=y_1+y_2i$ $y_1 \geq -180.218$ $y_1 \leq 175.021$ $-\pi \leq y_2 \leq \pi$
		CDLOG		1	complex*16	$z \neq 0+0i$	complex*16 $y=y_1+y_2i$ $y_1 \geq -180.218$ $y_1 \leq 175.021$ $-\pi \leq y_2 \leq \pi$
Common logarithm	{ALOG10 / LOG10}	{ALOG10 / LOG10}	$y=log_{10}x$	1	real*4	$x>0$	real*4 $y \geq -78.268$ $y \leq 75.859$
		DLOG10		1	real*8	$x>0$	real*8 $y \geq -78.268$ $y \leq 75.859$

Standard Library Functions (cont)

General Operation	Generic Name	Member Name	Mathematical Definition	Argument			Function Value Type and Range
				Number	Type	Range	
Square root	SQRT	SQRT	$y=\sqrt{x}$ or $y=x^{1/2}$	1	real*4	$x \geq 0$	real*4 $0 \leq y \leq M^{1/2}$
		DSQRT		1	real*8	$x \geq 0$	real*8 $0 \leq y \leq M^{1/2}$
		CSQRT	$y=\sqrt{z}$ or $y=z^{1/2}$	1	complex*8	any complex argument	complex*8 $0 \leq y_1 < 1.0987\ (M^{1/2})$ $\lvert y_2 \rvert < 1.0987\ (M^{1/2})$
		CDSQRT		1	complex*16	any complex argument	complex*16 $0 \leq y_1 < 1.0987\ (M^{1/2})$ $\lvert y_2 \rvert < 1.0987\ (M^{1/2})$
Cube root	CBRT	CBRT	$y=x^{1/3}$	1	real*4	any real argument	real*4 $-M^{1/3} < y < M^{1/3}$
		DCBRT		1	real*8	any real argument	real*8 $-M^{1/3} < y < M^{1/3}$

Distribution	ERF	ERF	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$	1	real*4	any real argument	real*4 $-1 \leq y \leq 1$
		DERF		1	real*8	any real argument	real*8 $-1 \leq y \leq 1$
	ERFC	ERFC	$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}du$	1	real*4	any real argument	real*4 $0 \leq y \leq 2$
		DERFC	y=1 – erf(x)	1	real*8	any real argument	real*8 $0 \leq y \leq 2$
	GAMMA	GAMMA	$y=\int_0^\infty u^{x-1}e^{-u}du$	1	real*4	$x > 2^{-252}$ and $x < 57.5744$	real*4 $0.88560 \leq y \leq M$
		DGAMMA		1	real*8	$x > 2^{-252}$ and $x < 57.5744$	real*8 $-0.88560 \leq y \leq M$

Standard Library Functions (cont)

General Operation	Generic Name	Member Name	Mathematical Definition	Argument			Function Value Type and Range
				Number	Type	Range	
Distribution (cont)	ALGAMMA LGAMMA	ALGAMA	$y=\log_e \Gamma(x)$ or	1	real*4	$x>0$ and $x<4.2913 \cdot 10^{73}$	real*4 $-0.12149 \leq y \leq M$
		DLGAMA	$y=\log_e \int_0^{\infty} u^{x-1} e^{-u} du$	1	real*8	$x>0$ and $x<4.2913 \cdot 10^{73}$	real*8 $-0.12149 \leq y \leq M$

NOTES:

1. $m = 16^{63} \cdot (1 - 16^{-6})$ for real*4 and $16^{63} \cdot (1 - 16^{-14})$ for real*8

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

Standard Library Subroutines

Subroutine	Format	Use
OVERFL	CALL OVERFL (i)	Tests for overflow or underflow.
DVCHK	CALL DVCHK (i)	Tests for invalid division.
ERROR	CALL ERROR (i)	Tests for function or I/O error conditions.
ERROR1	CALL ERROR1	Sets the function error indicator.
SLITE	CALL SLITE (e)	Sets the sense indicators specified by e.
SLITET	CALL SLITET (e,i)	Tests for the setting of specified sense indicators.
SSWTCH	CALL SSWTCH (e,i)	Tests the binary switch specified by the integer expression (e) and returns a value in the integer variable name (i).

Standard Library Subroutines (cont)

Subroutine	Format	Use
DUMP	CALL DUMP (list)	Dumps main storage assigned to the program; program execution terminates.
PDUMP	CALL PDUMP (list)	Dumps main storage assigned to the program; program execution continues.
EXIT	CALL EXIT	Terminates the program.
FETCH	CALL FETCH (s)	Loads and transfers control to the overlay specified by the phase name (s).
LOAD	CALL LOAD (s)	Loads subprogram overlays and transfers control to the program statement under the CALL statement.
OPSYS	CALL OPSYS ('LOAD',s)	Loads subprogram overlays and transfers control to the program statement under the CALL statement; equivalent to CALL statement for LOAD subroutine.

PARAM Statement Parameters

Parameter/Meaning
LST=option Indicates type of compiler output information required, where option may be: N – Compiler identification, parameters, and diagnostics. S – Source code listing plus N-option listings. M – Object summary and storage map with addresses of variables and arrays, plus S-option listings. Default option is S.

PARAM Statement Parameters (cont)

Parameter/Meaning
IN=module-name[/filename] Specifies name of source module being compiled. The module resides in the name of file. If filename is omitted, the name specified on the LIN parameter, or LIB1, is assumed.
OUT=filename Specifies name of object module file. Default action places object modules in temporary scratch file YRUN.

LIN=filename

Specifies name of default source module file.

Default value is LIB1.

PARAM Statement Parameters (cont)

Parameter/Meaning
OPT=(S,N,X,C,T) Specifies compiler options: S – Statement numbers are inserted into generated code as an aid to debugging. N – No object module is generated. Program units are compiled only. X – Compile all cards with an X in column 1. C – References to array elements are checked to see if they are outside declared array limits. T – Executable labels are traced.

MAP=(S,A,L)

Specifies type of map produced by compiler:

S – Object summary information, including module size and external subroutines called.

A – Alphabetical listing of addresses assigned to variables, arrays, and statement labels.

L – Listing of variables, arrays, and statement labels by storage allocations.

ERRFIL=module-name/lfdname

Specifies that an error file is created.

Module-name is the name of the module for a MIRAM source library file. Lfdname is the // LFD name of a MIRAM source library file.

Default is that no error file is written.

Unit References

- Control Module FL$IO

Unit	Device	Notes
1	Card read	Cards in control stream
3	Printer	Also used for diagnostics
5	Card read	Equivalent to unit 1
6	Printer	Equivalent to unit 3
29	Reread	

■ Control Module FL$IO1

Unit	Device	Notes
1	Card read	Cards in control stream
2	Card punch	
3	Printer	Also used for diagnostics
5	Card read	Equivalent to unit 1
6	Printer	Equivalent to unit 3
11,12	Tapes	508-byte variable unblocked records, no labels, workfile
29	Reread	To reread cards, but not tapes

```
10      16
UNIT    FDEVICE=REREAD
        FUNIT={k   }
              {READ}
```

```
10      16
UNIT    [FDEVICE=UNITREC,]
        FUNIT={k     }
              {PRINT }
              {PUNCH }
              {READ  }
        [,FFILEID=
             {filename                          }
             {FORTk; if FUNIT=k                 }
             {PRNTR; if FUNIT=PRINT             }
             {PUNCH; if FUNIT=PUNCH             }
             {READER; if FUNIT=READ             }]
        [,FNUMBUF={1}]
                  {2}
        [,FTYPEFLE=
             {INPUT; if FUNIT=READ              }
             {OUTPUT; if FUNIT=PRINT or PUNCH   }]
        [,FCHAR={OFF}]
                {ON }
        [,FOPTION=YES]
        [,FAUE=YES]
        [,FREREAD=YES]
        [,FCRDERR=RETRY]
        [,FRECSIZE=k]
        [{,FSPOOLIN=YES}]
        [{,FGETJCS=YES }]
        [,FTRANS=ASCII]
        [,FDIAGNOS=YES]
```

```
10        16
UNIT   [FDEVICE=TAPE,]
       FUNIT={k     }
             {READ  }
             {PUNCH }
             {PRINT }
       [,FFILEID=
            {filename                          }
            {FORTk; if FUNIT=k                 }
            {READER; if FUNIT=READ             }
            {PUNCH; if FUNIT=PUNCH             }
            {PRNTR; if FUNIT=PRINT             }]
       [,FTYPEFLE=
            {WORK                              }
            {INPUT; if FUNIT=READ              }
            {OUTPUT; if FUNIT=PRINT            }
            {   or PUNCH                       }]
       [,FRECFORM={VARUNB}]
                  {VARBLK}
                  {FIXUNB}
                  {FIXBLK}
       [,FNUMBUF={1}]
                 {2}
       [,FRECSIZE=k]
       [,FBFSZ=
            {k                                 }
            {FRECSIZE; if FRECFORM=FIXUNB      }
            {FRECSIZE+4; if FRECFORM=VARUNB    }
            {FRECSIZE*4; if blocked            }]
       [,FREREAD=YES]
       [,FDIAGNOS=YES]
       [,FBKNO=YES]
       [,FERROPT={IGNORE}]
                 {SKIP  }
       [,FFILABL={STD}]
                 {NO }
       [,FCKPTREC=YES]
```

```
10        16
UNIT   [FDEVICE=DISK,]
       FUNIT=(k
              READ
              PUNCH
              PRINT)
       [,FFILEID=
            (filename
             FORTk; if FUNIT=k
             READER; if FUNIT=READ
             PUNCH; if FUNIT=PUNCH
             PRNTR; if FUNIT=PRINT)]
       [,FTYPEFLE=
            (WORK
             INPUT; if FUNIT=READ
             OUTPUT; if FUNIT=PUNCH
                or PRINT)]
       [,FBFSZ=k]
       [,FRECFORM={VARUNB}]
                  {FIXUNB}
       [,FRECSIZE=k]
       [,FOPTION=YES]
       [,FVERIFY=YES]
       [,FDIAGNOS=YES]
       [{,FSPOOLIN=YES}]
        {,FGETJCS=YES }
       [,FREREAD=YES]
```

```
10        16
UNIT    [FDEVICE=WORKSTN,]
        FUNIT=(k       )
              (READ    )
              (PUNCH   )
              (PRINT   )

        [,FFILEID=                                    ]
        [     (filename                               )]
        [     (FORTk; if FUNIT=k                      )]
        [     (READER; if FUNIT=READ                  )]
        [     (PUNCH; if FUNIT=PUNCH                  )]
        [     (PRNTR; if FUNIT=PRINT                  )]

        [,FSCREND=(WRAP    )]
        [         (SCROLL  )]
        [         (NEWPAGE )]

        [,FTYPEFLE=                                   ]
        [     (WORK                                   )]
        [     (INPUT; if FUNIT=READ                   )]
        [     (OUTPUT; if FUNIT=PUNCH                 )]
        [        or PRINT                             )]

        [,FIOOPT=YES]

        [,FLINCNTL=YES]

        [,FNUMBUF=(1)]
        [         (2)]

        [,FOPTION=YES]

        [,FRECSIZE=k]

        [,FREREAD=YES]

        [(,FSPOOLIN=YES)]
        [(,FGETJCS=YES )]

        [,FDIAGNOS=YES]
```

```
10      16
UNIT    FDEVICE=EQUIV,
        FUNIT=(k
               READ
               PRINT
               PUNCH),
        FEQUIV=(k
                READ
                PRINT
                PUNCH)
```

```
10        17
ERRDEF  [,FPROG=((i|ALL|1),(j|ALL|1))]
        [,FARITH=((i|ALL),(j|ALL|10))]
        [,FARG=((i|ALL),(j|ALL|10))]
        [,FUNDFLO=YES]
        [,FALIGN=((i|ALL),(j|ALL|10))]
```

```
10        17
ERRDEF [,FREAD=((i,j))]
(cont) [,FDATA=((i,j))]
       [,FERROPT=((NONE
                   READ
                   READ,DATA
                   READ,UNREC
                   READ,DATA,UNREC
                   DATA
                   DATA,UNREC
                   UNREC))]
```

where:

i

Is a positive integer constant less than 32,768, with $i \geq j$; specifying the number of times the error is to be accepted before program termination. For a fatal error, j is assumed to be 1.

j

Is a positive integer constant less than 32,768, with $j \leq i$; specifying the number of diagnostic messages to be produced. For a fatal error, i is assumed to be 1.

ALL

Specifies that there is no limit on the number of times the error is to be accepted before program termination or that there is no limit on the number of diagnostic messages to be produced.

Special Characters Used in FORTRAN Source Language

Special Characters		Card Punches
(blank)	space	none
+	plus	12-6-8
–	minus	11
/	virgule	0-1
=	equal	6-8
.	decimal point	12-3-8
)	right parenthesis	11-5-8
*	asterisk	11-4-8
,	comma	0-3-8
(	left parenthesis	12-5-8
'	apostrophe	5-8
&	ampersand	12
;	semicolon	11-6-8

Example of Compile, Assemble, Link, Execute

```
1          10      16
// JOB    FORTJOB1                    Name of job
// FOR4                               Call FORTRAN IV
/$
      program data                    Card input to compiler
/*
// ASM    LST=NC                      Call assembler; no cross-reference
/$
MY10      START
          FUNIT=...
             .
             .
             .                        Unit table information
          FUNIT=...
          FUNEND
          ERRDEF
          END
```

Example of Compile, Assemble, Link, Execute (cont)

```
/*
// LINK                                  Call link editor
// DVC ---   // LFD ---
      :                                  Assign devices required
      :                                  by unit table
// DVC ---   // LFD ---
// EXEC  LNKLOD,$Y$RUN                   Execute the program
/$
    data                                 Optional spoolin data file
/*
/&                                       Terminate job stream
// FIN
```

Example of Compile, Link, Execute — Disk Input

```
1          10      16
// JOB    FORJOB2                          Name of job
//TEST1   FOR4   LST=M,IN=(MYDISK,SOURCE)  Call FORTRAN IV with
// PARAM  IN=TEST2                         PARAM's — 2nd program
// PARAM  IN=TEST3                         3rd program
      :
//TEST    LINK   OUT=(MYDISK,LOADFILE)     Link module TEST to MYDISK
// DVC ---       // LFD ---                 Optional devices for execution
// EXEC   TEST,OUT                         Execute TEST
/$                                        }
    data                                  } Optional data
/*                                        }
/&                                        } End of job
// FIN                                    }
```