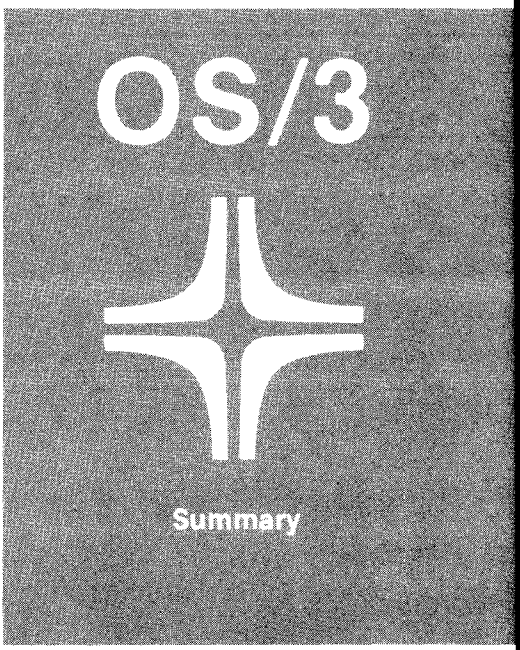
FORTRAN



Environment: 90/25, 30, 30B, 40 Systems

SPERRY UNIVAC

UP-8348
Rev. 2

H

This document contains the latest information available at the time of preparation. Therefore, it may contain descriptions of functions not implemented at manual distribution time. To ensure that you have the latest information regarding levels of implementation and functional availability, please consult the appropriate release documentation or contact your local Sperry Univac representative.

Sperry Univac reserves the right to modify or revise the content of this document. No contractual obligation by Sperry Univac regarding level, scope, or timing of functional implementation is either expressed or implied in this document. It is further understood that in consideration of the receipt or purchase of this document, the recipient or purchaser agrees not to reproduce or copy it by any means whatsoever, nor to permit such action by others, for any purpose without prior written permission from Sperry Univac.

Sperry Univac is a division of the Sperry Corporation.

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: UP-8348 Rev. 2
RELEASE LEVEL: 7.0 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		
PSS	1	
Preface	1	
Contents	1	
Text	1 thru 74	
User Comment Sheet		

Preface

This summary presents both Basic FORTRAN and FORTRAN IV in a 90/25, 30, 30B, and 40 environment. The functions that pertain only to FORTRAN IV are shaded.

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: $-2,147,483,648 \leq$ integer $\leq 2,147,483,647$. $-32768 \leq$ integer*2 ≤ 32767.
Real	4	8	A number with a fractional portion. Maximum value is $+ 7.237005 * 10^{75}$. Minimum value is $0.5397605 * 10^{-78}$. 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.
Literal ①	(0 < X ≤ 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)

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 1—6 character alphanumeric string, beginning with alphabetic character

② 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.

③ Length is not allowed with DOUBLE PRECISION type.

④ Variable-names used as arguments may be enclosed in slashes to indicate call by name.

SUBROUTINE subroutine-name [(({ variable-name-1 ① / array-name-1 / proc-name-1 / * ② } ... [,{ variable-name-n ① / 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)

Specification Statements

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

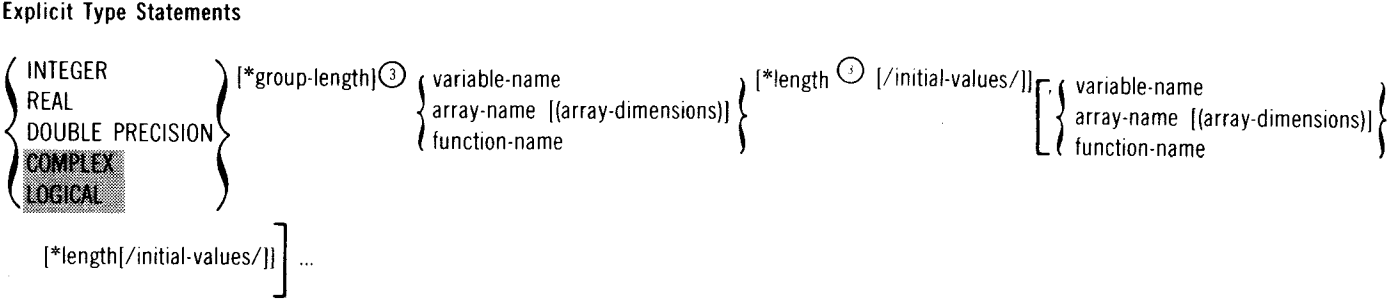
COMMON [/block-name/] { variable-name / array-name [(array-dimensions)] } [, { variable-name / array-name [(array-dimensions)] } ...]

[/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] ...)] ... ②

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-function-name } [, { external-subroutine-name / external-function-name }]...

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

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 but may not be part of a range.

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

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:

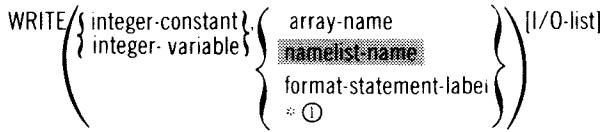
① Indicates the use of list-directed I/O.

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

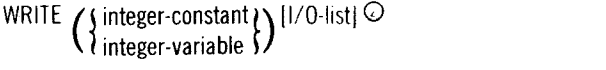
REWIND { integer-constant / integer-variable }

Sequential Input/Output Statements (cont)

Formatted WRITE Statement



Unformatted WRITE Statement



NOTE

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

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

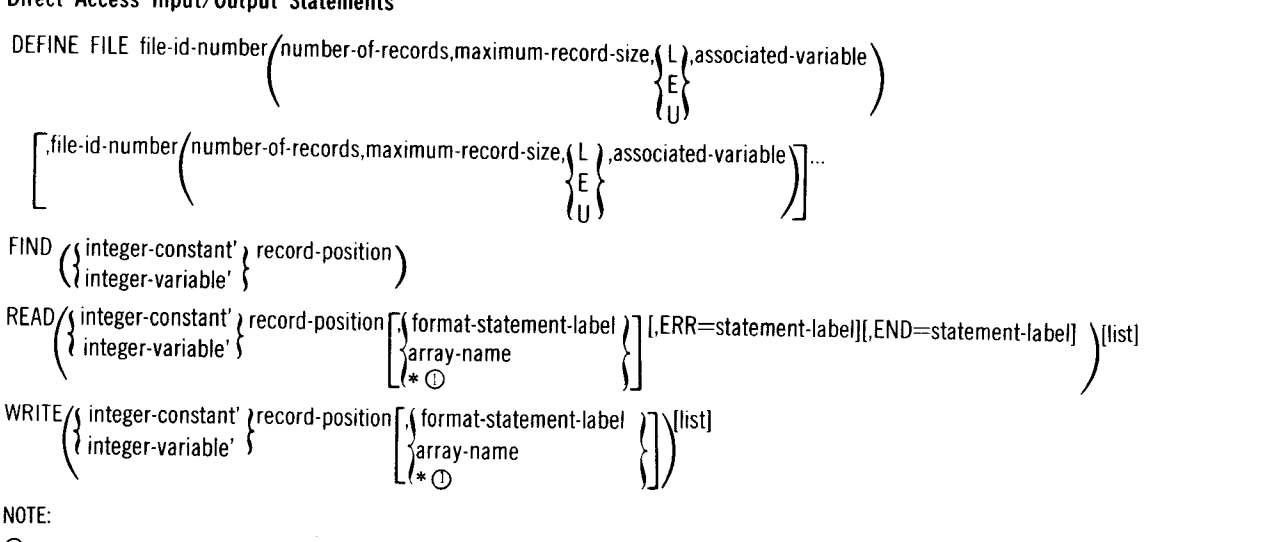
Other Sequential Input/Output Statements

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



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 / MIN0}	Select the smallest value	≥2	JMIN0 ①	Integer*2	Integer*2
			AMIN0 ②	Integer*4	Real*4
			AMIN1 ①	Real*4	Real*4
			{MIN ① / MIN0 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
			DHFLOT 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	JISIGN 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	1	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 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	$\|x_1\| \leq 174.673$ $\|x_2\| < 2^{18}\pi$	complex*8 $-M \leq y_1, y_2 \leq M$
		CDCOSH		1	complex*16	$\|x_1\| < 174.673$ $\|x_2\| < 2^{50}\pi$	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 $-1 \leqslant y \leqslant 1$
		DTANH		1	real*8	any real argument	real*8 $-1 \leqslant y \leqslant 1$
Exponential	EXP	EXP	$y=e^{x}$	1	real*4	$x \geqslant -180.218$ $x \leqslant 174.673$	real*4 $0 \leqslant y \leqslant M$
		DEXP		1	real*8	$x \geqslant -180.218$ $x \leqslant 174.673$	real*8 $0 \leqslant y \leqslant M$
		CEXP	$y=e^{z}$	1	complex*8	$x_1 \leqslant 174.673$ $\lvert x_2 \rvert \leqslant (2^{18}\pi)$	complex*8 $-M \leqslant y_1, y_2 \leqslant M$
		CDEXP		1	complex*16	$x_1 \leqslant 174.673$ $\lvert x_2 \rvert < (2^{50}\pi)$	complex*16 $-M \leqslant y_1, y_2 \leqslant M$

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 $0 < y \leq M$
		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 $0 \leq y_1 \leq 1.0987\ (M^{1/2})$ $\lvert y_2 \rvert \leq 1.0987\ (M^{1/2})$
		CDSQRT		1	complex*16	any complex argument	complex*16 $0 \leq y_1 \leq 1.0987\ (M^{1/2})$ $\lvert y_2 \rvert \leq 1.0987\ (M^{1/2})$
Cube root	CBRT	CBRT	$y=x^{1/3}$	1	real*4	any real argument	real*4 $-M^{1/3} \leq y \leq M^{1/3}$
		DCBRT		1	real*8	any real argument	real*8 $-M^{1/3} \leq y \leq M^{1/3}$

Distribution							
ERF	ERF	$y=\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2}du$	1	real*4	any real argument	real*4 $-1<y<1$
	DERF		3	real*8	any real argument	real*8 $-1<y<1$
ERFC	ERFC	$y=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-u^2}du$	1	real*4	any real argument	real*4 $0<y<2$
	DERFC	$y=1-\text{erf}(x)$	1	real*8	any real argument	real*8 $0<y<2$
GAMMA	GAMMA	$y=\int_0^\infty u^{x-1}e^{-u}du$	1	real*4	$x\geq 2^{-253}$ and $x<57.5744$	real*4 $0.88560<y<\infty$
	DGAMMA		1	real*8	$x\geq 2^{-252}$ and $x<57.5744$	real*8 $0.88560<y<\infty$

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	$y=\log_e \int^{\infty} (x)$ 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 – Basic FORTRAN

Parameter/Meaning
LST=k Indicates type of output to be printed by the compiler, where k is the sum of any of the following options: 1 – Source code listing 2 – Diagnostic listing 4 – Storage allocation map 8 – Object code listing Default value is 7.
IN=module-name[/filename] Specifies that source module is in the named file. If filename is omitted, default filename is YSRC.

PARAM Statement Parameters — Basic FORTRAN (cont)

Parameter/Meaning
OUT= { **(filename)** / **NO** } Specifies whether an object module is to be generated, and, if so, where it is to be stored. **(filename)** Is the name of the file where the generated object module is to be stored. Default value is YRUN. **NO** Generate no object module.
SUBCHK=ALL Specifies that all array element addresses are to be checked.

TRACE=ALL

ALL

Specifies that labels of executable statements in the entire program unit are to be traced.

X

Specifies that statements with an X in column 1 are to be compiled. If not specified, statements are treated as comments.

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 — FORTRAN IV

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 – FORTRAN IV (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.

Unit References – Basic FORTRAN

- Control Module FP$IO

Unit	Device	Notes
1	Card read	Cards in control stream
3	Printer	121 characters
5	Card read	Equivalent to unit 1
6	Printer	Equivalent to unit 3
29	Reread	

The LFD filename FORT03 is required for the printer.

Unit References – FORTRAN IV

- 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

Basic FORTRAN

```
10      17
FUNTAB
```

```
10      16
UNIT    FDEVICE=REREAD,
        FUNIT=k
```

FORTRAN IV DTF

```
 10      17
FUNTAB  SYS=FOR
```

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

FORTRAN IV CDI

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

Basic FORTRAN

```
10      16
UNIT    FDEVICE=PRINTER,
        FUNIT=k
        [,FNUMBUF=2]
        [,FRECSIZE=k]
```

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=PRINTER,
        FUNIT={k      }
              {PRINT  }
              {PUNCH  }
        [,FFILEID=                                  ]
        [        {filename                          }]
        [        {FORTk;if FUNIT=k                  }]
        [        {PRNTR;if FUNIT=PRINT              }]
        [        {PUNCH;if FUNIT=PUNCH              }]
        [,FRECSIZE={k  }]
        [          {121}]
        [,FNUMBUF={1}]
        [         {2}]
        [,FDIAGNOS=YES]
        [,FPRINTOV={SKIP  }]
        [          {NOSKIP}]
        [,FCHAR={OFF}]
        [       {ON }]
        [,FOPTION=YES]
```

Basic FORTRAN

Not Available

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=CARDIN,
        FUNIT={k      }
              {READ   }
        [,FFILEID=                                    ]
        [   {filename                                 }]
        [   {FORTk ; if FUNIT=k                       }]
        [   {READER ; if FUNIT=READ                   }]
        [,FREREAD=YES]
        [,FBUFPOOL=YES]
        [,FNUMBUF={1}]
        [         {2}]
        [,FWORKA={YES ; if FNUMBUF=1}]
        [        {NO  ; if FNUMBUF=2}]
        [,FRECSIZE={k  }]
        [          {80 }]
        [,FSTUB={51}]
        [       {66}]
        [,FOPTION=YES]
        [,FAUE=YES]
        [,FBKSZ={k        }]
        [       {FRECSIZE }]
```

Basic FORTRAN

```
10      16
UNIT    FDEVICE=CARDOUT,
        FUNIT=k
        [,FNUMBUF=2]
        [,FCRDERR=RETRY]
        [,FRECSIZE=k]
```

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=CARDOUT,
        FUNIT={k      }
              {PUNCH  }
        [,FFILEID=                                  ]
          [    {filename                    }      ]
          [    {FORTk; if FUNIT=k           }      ]
          [    {PUNCH; if FUNIT=PUNCH       }      ]
        [,FBUFPOOL=YES]
        [,FNUMBUF={1}]
                  {2}
        [,FWORKA={YES; if FNUMBUF=1}]
                 {NO; if FNUMBUF=2 }
        [,FRECSIZE={k  }]
                   {80 }
        [,FCRDERR=RETRY]
        [,FOPTION=YES]
        [,FBKSZ={k       }]
                {FRECSIZE}
```

FORTRAN IV CDI

This configuration is the System 80 equivalent to the two preceding configurations of Basic FORTRAN and FORTRAN IV.

```
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]
       [,FRECSZE=k]
       [{,FSPOOLIN=YES}]
       [{,FGETJCS=YES }]
       [,FTRANS=ASCII]
       [,FDIAGNOS=YES]
```

Basic FORTRAN

10	16
UNIT	FDEVICE=TAPE, FUNIT=k [,FTYPEFLE={INPUT / OUTPUT ① / WORK}] [,FNUMBUF=2] ① [,FBKSZ=k] [,FBKNO=YES] [,FERROPT={IGNORE / SKIP}] [,FFILABL={STD / NO}] [,FCKPT=YES] [,FOPTION=YES]

NOTE:

① Backspace not permitted if this option is chosen.

FORTRAN IV DTF

```
10      16
UNIT  FDEVICE=TAPE,
      FUNIT={k      }
            {READ   }
            {PUNCH  }
     [,FFILEID=                                          ]
     [   {filename                                       }]
     [   {FORTk ; if FUNIT=k                             }]
     [   {READER ; if FUNIT=READ                         }]
     [   {PUNCH ; if FUNIT=PUNCH                         }]
     [,FTYPEFLE=                                         ]
     [   {INPUT                                          }]
     [   {WORK ; if FUNIT=k                              }]
     [   {INPUT ; if FUNIT=READ                          }]
     [   {OUTPUT (1) ; if FUNIT=                         }]
     [   {  PUNCH                                        }]
     [,FRECFORM={VARUNB   }                              ]
     [          {VARBLK (1)}                             ]
     [          {FIXUNB (1)}                             ]
     [          {FIXBLK (1)}                             ]
     [,FNUMBUF={1    }                                   ]
     [         {2 (1)}                                   ]
     [,FWORKA=                                           ]
     [   {YES (1) ; if FNUMBUF=1                         }]
     [   {NO ; if FNUMBUF=2                              }]
     [,FBUFPOOL=YES]
     [,FRECSIZE={k   }                                   ]
     [          {508 }                                   ]
```

NOTE:

(1) Backspace not permitted if this option is chosen.

FORTRAN IV DTF (cont)

```
10        16
UNIT
(cont)  [,FBKSZ=
          ( k                                          )
          ( FRECSIZE ;if FRECFORM=                     )
          (    FIXUNB                                  )
          ( FRECSIZE+4 ;if FRECFORM=                   )
          (    VARUNB                                  )
          ( FRECSIZE*4 ;otherwise                      )]
        [,FREREAD=YES]
        [,FDIAGNOS=YES]
        [,FBKNO=YES]
        [,FERROPT={IGNORE}]
                  {SKIP  }
        [,FRECERR=YES]
        [,FCKPT=YES]
        [,FFILABL={STD}]
                  {NO }
        [,FCLRW={RWD   }]
                {NORWD }
                {UNLOAD}
        [,FOPRW=NORWD]
        [,FOPTION=YES]
```

FORTRAN IV CDI

```
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)]
```

FORTRAN IV CDI (cont)

```
10        16
UNIT      [,FREREAD=YES]
(cont)    [,FDIAGNOS=YES]
          [,FBKNO=YES]
          [,FERROPT={IGNORE}]
                    {SKIP  }
          [,FFILABL={STD}]
                    {NO }
          [,FCKPTREC=YES]
```

Basic FORTRAN

```
10        16
UNIT      FDEVICE=SDISC,
          FUNIT=k
          [,FTYPEFLE={INPUT }]
                     {OUTPUT}
                     {WORK  }
          [,FNUMBUF=2]
          [,FBKSZ=k]
          [,FERROPT={IGNORE}]
                    {SKIP  }
          [,FOPTION=YES]
          [,FVERIFY=YES]
```

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=SDISC,
        FUNIT={ k      }
              { READ   }
              { PUNCH  }
        [,FSECTOR={ NO  }]
                  { YES }
        [,FFILEID=                                    ]
          [  { filename                               }  ]
          [  { FORTk;if FUNIT=k                       }  ]
          [  { READER;if FUNIT=READ                   }  ]
          [  { PUNCH;if FUNIT=PUNCH                   }  ]
        [,FTYPEFLE=                                   ]
          [  { INPUT                                  }  ]
          [  { WORK;if FUNIT=k                        }  ]
          [  { INPUT;if FUNIT=READ                    }  ]
          [  { OUTPUT; if FUNIT=PUNCH                 }  ]
        [,FRECFORM={ VARUNB }  ]
                   { VARBLK }
                   { FIXUNB }
                   { FIXBLK }
        [,FNUMBUF={1}]
                  {2}
        [,FBUFPOOL=YES]
        [,FWORKA={YES;if FNUMBUF=1}]
                 {NO;if FNUMBUF=2 }
        [,FRECSIZE={k   }]
                   {508 }
```

FORTRAN IV DTF (cont)

```
10      16
UNIT    [,FBKSZ=                                          ]
(cont)  [       {k                                        }]
        [       {|FRECSIZE|;if FRECFORM=FIXUNB            }]
        [       {|FRECSIZE+4|;if FRECFORM=VARUNB          }]
        [       {|FRECSIZE*4|;otherwise                   }]
        [,FREREAD=YES]
        [,FDIAGNOS=YES]
        [,FERROPT={IGNORE}]
        [         {SKIP  }]
        [,FRECERR=YES]
        [,FOPTION=YES]
        [,FVERIFY=YES]
```

Basic FORTRAN

```
10      16
UNIT    FDEVICE=DISC,
        FUNIT=k
        [,FRECSIZE=k]
        [,FTYPEFLE={INPUT }]
        [          {OUTPUT}]
        [,FVERIFY=YES]
```

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=DISC,
        FUNIT=k
        [,FSECTOR={NO  }]
                  {YES }
        [,FFILEID=                          ]
          {filename                         }
          {FORTk;where k=FUNIT              }
        [,FTYPEFLE={INPUT }]
                   {OUTPUT}
        [,FBUFPOOL=YES]
        [,FRECSIZE={k   }]
                   {512 }
        [,FRECERR=YES]
        [,FREREAD=YES]
        [,FVERIFY=YES]
```

FORTRAN IV CDI

```
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             }]
```

FORTRAN IV CDI (cont)

```
10        16
UNIT      [,FTYPEFLE=                                          ]
(cont)    [    {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]
```

Basic FORTRAN

Not Available

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=MIDISC,
        FUNIT=(k
               READ
               PUNCH)
        [,FFILEID=
           (filename
            FORTk;if FUNIT=k
            READER;if FUNIT=READ
            PUNCH;if FUNIT=PUNCH)]
        [,FTYPEFLE=
           (WORK;if FUNIT=k
            INPUT;if FUNIT=READ
            OUTPUT;if FUNIT=PUNCH)]
        [,FRECFORM=(FIX)]
                   (VAR)
        [,FBUFPOOL=YES]
        [,FRECSIZE=
           (k
            255;if FRECFORM=FIX
            252;if FRECFORM=VAR)]
        [,FREREAD=YES]
        [,FRECERR=YES]
        [,FOPTION=YES]
        [,FVERIFY=YES]
```

FORTRAN IV CDI

```
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]
```

Basic FORTRAN

```
10      16
UNIT    FDEVICE=SPOOLIN,
        FUNIT=k
        [,FRECSIZE=k]
```

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=SPOOLIN,
        FUNIT={k   }
              {READ}
        [,FREREAD=YES]
        [,FBKSZ={k   }]
               {400 }
        [,FBUFPOOL=YES]
        [,FRECSIZE={k  }]
                   {80 }
```

FORTRAN CDI

Not Available

Basic FORTRAN

```
10      16
UNIT    FDEVICE=EQUIV,
        FUNIT=K,
        FEQUIV=j
```

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=EQUIV,
        FUNIT={k      },
              {READ   }
              {PRINT  },
              {PUNCH  }
        FEQUIV={k     }
               {READ  }
               {PRINT }
               {PUNCH }
```

FORTRAN CDI

```
10      16
UNIT    FDEVICE=EQUIV,
        FUNIT={k      }
              {READ   }
              {PRINT  }
              {PUNCH  }
        FEQUIV={k     }
               {READ  }
               {PRINT }
               {PUNCH }
```

Basic FORTRAN

Not Available

FORTRAN IV DTF

```
10      16
UNIT    FDEVICE=DMS,
        FUNIT=k,
        FWORKA=YES
        [,FFILEID={filename}]
                  {FORTk  }
        [,FRECFORM={VARUNB }]
                   {VARBLK }
                   {FIXUNB }
                   {FIXBLK }
        [,FRECSIZE={k   }]
                   {508 }
        [,FREREAD=YES]
```

FORTRAN CDI

Not Available

Basic FORTRAN

Not Available

FORTRAN IV DTF

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

FORTRAN IV DTF (cont)

```
10        17
ERRDEF [,FREAD=(  {i  },{j  })]
(cont)         {ALL}  {ALL}
                      {10 }
       [,FDATA=(  {i  },{j  })]
                 {ALL}  {ALL}
                        {10 }
       [,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≥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≤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, and Execute – Basic FORTRAN

```
1         10        16
// JOB    FORTJOB1                    Name of job
// FORT                               Call Basic FORTRAN
/$                                    }
      program data                    } Card input to compiler
/*                                    }
// ASM    LST=NC                      Call assembler
/$                                    }
MYIO      START                       }
          FUNTAB                      }
          UNIT                        }
            .                         } Unit table information
            .                         }
            .                         }
          FUNEND                      }
          END                         }
/*                                    }
```

Example of Compile, Assemble, Link, Execute – Basic FORTRAN (cont)

```
1          10      16
// LINK                                  Call link editor
// DVC---  // LFD---                     Optional devices required for execution
// EXEC   LNKLOD,$Y$RUN                  Execute program
/$                                       } Optional spoolin data file
     data                                }
/*                                       } End of job
/&                                       }
// FIN                                   }
```

Example of Compile, Link, Execute – Disk Input – Basic FORTRAN

```
// JOB    FORTJOB2                                  Name of job
//TEST1   FORT   LST=15,IN=(MYDISK,SOURCE)          Call Basic FORTRAN with
// PARAM  IN=TEST2/IN                               PARAM's — 2nd program
// PARAM  IN=TEST3/IN                               3rd program

//TEST    LINK   OUT=(MYDISK,LOADFILE)              Link module TEST to MYDISK

// DVC--- // LFD---                                 Optional device for execution

// EXEC   TEST,OUT                                  Execute TEST
/$
    data                                            Optional spoolin data
/*
/&                                                  End of job
// FIN
```

Example of Compile, Assemble, Link, Execute – FORTRAN IV

```
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
/$
MYIO      START
          FUNTAB  SYS=FOR
          UNIT    FDEVICE=---
           :                          Unit table information
          UNIT    FDEVICE=---
          FUNEND
          ERRDEF
          END
```

```
/*
// 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 – FORTRAN IV

```
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
```