114

TOPS–10 / TOPS–20

FORTRAN
Reference Card

digital
software

TOPS–10/TOPS–20 FORTRAN Reference Card

AV–P529A–TK

March 1983

This reference card is a convenient, pocket-sized booklet that summarizes the major elements of the FORTRAN language. The information in this booklet is a subset of the information in the *TOPS–10/TOPS–20 FORTRAN Language Manual*.

OPERATING SYSTEM:	TOPS–10 V7.01A TOPS–20 V4.1 TOPS–20 V5.1
SOFTWARE:	FORTRAN–10 V7 FORTRAN–20 V7

digital equipment corporation • marlboro, massachusetts

Contents

Contents (Cont.)

GENERAL INFORMATION

PREFACE

The FORTRAN–10/20 Reference Card reflects the software as of Version 7 of the FORTRAN–10/20 compiler. Version 7 of the FORTRAN–10/20 Object Time System (FOROTS), and Version 7 of the FORTRAN–10/20 debugging program (FORDDT).

This reference card describes the FORTRAN language as implemented for the TOPS–10 operating system (FORTRAN–10) and the TOPS–20 operating system (FORTRAN–20). Any differences between FORTRAN–10 and FORTRAN–20 are noted.

The following conventions are used throughout the reference card:

Braces { }	indicate that a choice must be made from one of the enclosed lines.
Brackets [ ]	indicate an optional feature.
Ellipsis . . . or . : .	indicate the omission of information from a programming example or that items in a command line can be optionally repeated.
Lowercase letters	indicate variable information you supply in a command string.
UPPERCASE LETTERS	indicate fixed (or literal information that you must enter as shown in a command string.

The standard for FORTRAN is the American National Standards Institute (ANSI) FORTRAN, x3.9–1978 (also known as FORTRAN–77). FORTRAN–10/20 extensions and additions to ANSI FORTRAN are in blue print.

Compiling, Executing and Debugging Programs

The COMPILE–class commands are:

COMPILE	EXECUTE
LOAD	DEBUG

Running the FORTRAN–10 Compiler

On TOPS–10, the command to run the FORTRAN compiler directly is:

```
.R FORTRA
```

The compiler responds with an asterisk (*), and is then ready to accept a command string. The form of the FORTRAN compiler command string is:

object filespec, listing filespec = source filespec(s)

FORTRAN–10 Compiler Switches

Switch	Meaning
/CROSSREF	Generates a file with extension .CRF that can be input to the CREF program.
/DEBUG	Includes debugging information in your program.
/EXPAND	Includes the octal-formatted version of the object file in the listing.
/F66	The FORTRAN–66 standard rules apply for DO loops and EXTERNAL statements. (Same function as the /NOF77 switch.)
/F77	The FORTRAN–77 standard rules apply for DO loops and EXTERNAL statements.
/INCLUDE	Compiles a D in card column 1 as space.
/LNMAP	Produces a line number/octal location map in the listing only if /MACROCODE was not specified.
/MACROCODE	Adds the mnemonic translation of the object code to the listing file.
/NOF77	The FORTRAN–66 standard rules apply for DO loops and EXTERNAL statements. (Same function as the /F66 switch.)
/NOERRORS	Does not print error messages on the terminal.

/NOWARN	Suppresses compiler warning messages.
/OPTIMIZE	Performs global optimization.
/SYNTAX	Performs syntax check only.

Running the FORTRAN–20 Compiler

On TOPS–20, the command to run the FORTRAN compiler directly is:

```
@FORTRA
```

The compiler responds with the following prompt:

```
FORTRAN>
```

and is then ready to accept a command string.

You should enter a command string in one of the following forms:

1. <source-file-spec> [switches]
2. <source-file-spec> + <source-file-spec> + ... [switches]
3. /TAKE:<file-spec> [/ECHO]
4. /RUN:<file-spec> [/OFFSET:<integer>]
5. /HELP
6. /EXIT

FORTRAN–20 Compiler Switches

Switch	Meaning
/ABORT	Causes the compiler to exit at the end of a compilation that contains errors.
/BINARY[:relfile]	Indicates that a relocatable binary file is generated. You can optionally specify the file specification.
/CROSSREF	Generates a file with extension .CRF that can be input to the CREF program.
/DEBUG	Includes debugging information in your program.
/DFLOATING	Indicates that double-precision numbers are stored in D–floating format.
/ECHO–OPTION	Echo switches selected from the SWITCH.INI file.
/EXPAND	Includes the octal-formatted version of the object file in the listing.
/F66	The FORTRAN–66 standard rules apply for DO loops and EXTERNAL statements. (Same function as the /NOF77 switch.)
/F77	The FORTRAN–77 standard rules apply for DO loops and EXTERNAL statements.

FORTRAN–20 Compiler Switches (Cont.)

Switch	Meaning
/GFLOATING	Indicates that double-precision numbers are stored in G–floating format. (TOPS–20 KL model B only.)
/INCLUDE	Compiles a D in card column 1 as space.
/LISTING[:listfile]	Indicates a list file will be generated. You can optionally specify the file specification.
/LNMAP	Produces a line number/octal location map in the listing only if /MACHINE–CODE was not specified.
/MACHINE–CODE	Adds the mnemonic translation of the object code to the listing file. This command will cause a default /LISTING.
/NOBINARY	Indicates that no relocatable binary file is generated.
/NOF77	The FORTRAN–66 standard rules apply for DO loops and EXTERNAL statements. (Same function as the /F66 switch.)
/NOERRORS	Does not print error messages on the terminal.
/NOWARN	Suppresses warning messages.

/OPTIMIZE	Performs global optimization.
/OPTION[:option]	Only read lines from the SWITCH.INI file that start with FORTRA:option.
/SYNTAX	Performs syntax check only.

Arguments to /DEBUG Switch

Arguments	Meaning
DIMENSIONS	Includes dimension information in .REL file for FORDDT.
TRACE	Generates references to FORDDT required for its trace features (automatically activates LABELS).
LABELS	Generates a label for each statement of the form <line-number>L. (This option may be used without FORDDT.)
INDEX	Forces DO loop indexes to be stored at the beginning of each iteration rather than held in a register for the duration of the loop. In addition, this switch forces all function values to be stored in memory prior to return from the function. If this switch is specified, you can set a FORDDT pause on the RETURN statement and then examine the value to be returned.
BOUNDS	Generates the bounds checking code for all array references and substring references. Bounds violations will produce run-time error messages. Note that the technique of specifying dimensions of 1 for subroutine arrays will cause bounds check errors. (You may use this option without FORDDT.)

ARGUMENTS	Generates type checking information at load time for actual argument types and associated dummy argument types. Type violations will produce non-fatal load-time error messages. This switch also performs type checking at compile-time for statement functions.
NONE	Do not include any debug features.
ALL	Enable all debugging aids.

The following formulas may be used to determine the increases in program size that will occur as a result of the addition of various /DEBUG options.

DIMENSIONS	For each array, 3+3*N words where N is the number of dimensions, and up to three constants for each dimension.
TRACE	One instruction per executable statement.
LABELS	No increase.
INDEX	One instruction per inner loop plus one instruction for some of the references to the index of the loop. Also, one instruction per subprogram.
BOUNDS	For each array, the formula is the same as DIMENSIONS. For each reference to an array element, use 5+N words; where N is the number of dimensions in the array. If you do not specify BOUNDS, approximately 1+3*(N−1) words will be used. For each reference to a substring, add 5 words.
ARGUMENTS	No increase.

Arguments to /NOWARN Switch

Arguments	Meaning
ALL	Suppress all warning messages.
NONE	Do not suppress warning messages.
xxx	Where xxx is the three character error mnemonic for the error message to be suppressed.

Hierarchy of FORTRAN Operators

Class	Level	Symbol or Mnemonic
EXPONENTIAL	First	** or ^
	Second	−(negation) and + (identity)
ARITHMETIC	Third	*,/
	Fourth	+,−
RELATIONAL	Fifth	.GT.,.GE.,.LT.,.LE.,.EQ.,.NE. or >,>=,<,<=,==,#
	Sixth	.NOT.
	Seventh	.AND.
LOGICAL	Eighth	.OR.
	Ninth	.EQV.,.NEQV.

FORTRAN Character Set

Letters

Uppercase: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Lowercase: a b c d e f g h i j k l m n o p q r s t u v w x y z

Digits

0 1 2 3 4 5 6 7 8 9

Symbols

!	Exclamation Point	,	Comma
"	Quotation Mark	-	Hyphen (Minus)
#	Number Sign	.	Period (Decimal Point)
$	Dollar Sign	/	Slant (Slash)
&	Ampersand	:	Colon
'	Apostrophe	;	Semicolon
(	Left Parenthesis	<	Less Than
)	Right Parenthesis	=	Equal To
*	Asterisk	>	Greater Than
+	Plus	^	Circumflex

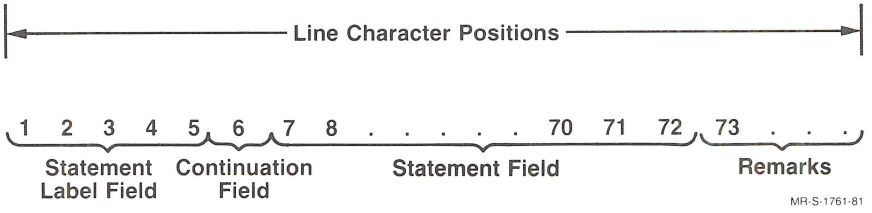
Line Termination Characters

Line Feed (LF), Form Feed (FF), Vertical Tab (VT)

Line Formatting Characters

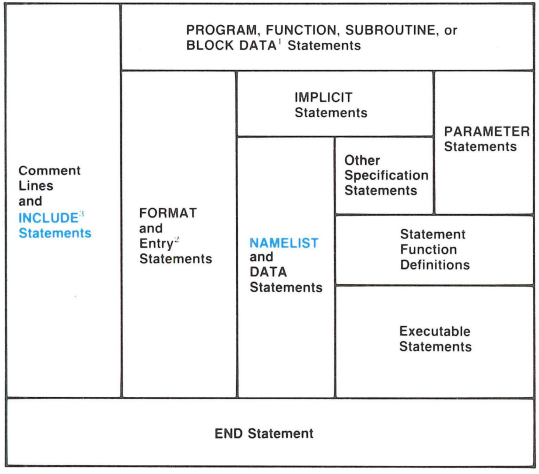
Carriage Return (RET), Horizontal Tab (TAB), Blank

Fields Within a FORTRAN Line



FORTRAN STATEMENTS

Ordering of FORTRAN Statements



[1] BLOCK DATA subroutines cannot contain any executable statements, statement functions, FORMAT statements, EXTERNAL statements, INTRINSIC statements, or NAMELIST statements (See Section 13.1).

[2] The ENTRY statement is allowed only in functions or subroutines. All executable statements which reference any dummy parameters must physically follow the ENTRY statement unless the references appear in the FUNCTION statement, the SUBROUTINE statement, or in a preceding ENTRY statement.

[3] The placement of an INCLUDE statement is dictated by the types of statements to be included.

MR-S-1763-81

Summary of FORTRAN Statements

Form

ACCEPT(FMT = f[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
ACCEPT(FMT = *[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
ACCEPT f[,iolist]
ACCEPT *[,iolist]

ASSIGN s to i

BACKFILE un
BACKFILE (UNIT = un[,ERR =][,IOSTAT = ios])
BACKFILE (un[,ERR = s][,IOSTAT = ios])

BACKSPACE un
BACKSPACE (UNIT = un[,ERR = s][,IOSTAT = ios])
BACKSPACE (un[,ERR = s][,IOSTAT = ios])

BLOCK DATA [sub]

CALL sub [([a1 [,a2]...])]

CHARACTER [*len[,]] v[*len] [,v[*len]]

CLOSE (closelist)

COMMON [/[cb]/]nlist[[,]/[cb]/nlist]...

COMPLEX v [,v...]

CONTINUE

DATA nlist/clist/ [[,]nlist/clist/]...

DECODE(c,f,a[,ERR=s][,IOSTAT=ios])[iolist]

DIMENSION a(d) [,a(d)...]

DO [s[,]] i=e1,e2[,e3]

DO [s[,]] WHILE (e)

DOUBLE PRECISION v [,v...]

ELSE

ELSE IF (e) THEN

ENCODE(c,f,a[,ERR=s][,IOSTAT=ios])[iolist]

END

END DO

Summary of FORTRAN Statements (Cont.)

Form

END IF

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

ENTRY en [(d1 [,d2...])]

EQUIVALENCE (nlist) [,(nlist)...]

EXTERNAL proc [,proc]...

FIND (UNIT = un,REC = rn[,ERR = s][,IOSTAT = ios])
FIND (un'rn[,ERR = s][,IOSTAT = ios])

FORMAT (fs)

fun ([arg1,arg2,...argn])

[typ] FUNCTION fun ([arg1 [,arg2],...,argn])

GO TO i [[,](s [,s]...)]

GO TO s

GO TO (s [,s]...)[,] e

INCLUDE filespec/switch

IF (e) st

IF (e) s1, s2, s3

IF (e) s1, s2

IF (e) THEN

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

INTEGER v [,v...]

INTRINSIC fun[,fun]

LOGICAL v [,v...]

NAMELIST /name/list[/name/list]...

OPEN (openlist)

PARAMETER (p = c[,p = c...])
PARAMETER p = c [,p = c...]

PAUSE [n]

Summary of FORTRAN Statements (Cont.)

Form

PRINT(FMT = f[,ERR = s][,IOSTAT = ios])[iolist]
PRINT(FMT = *[,ERR = s][,IOSTAT = ios])[iolist]
PRINT f[,iolist]
PRINT *[,iolist]

PROGRAM name

PUNCH(FMT = f[,ERR = s][,IOSTAT = ios])[iolist]
PUNCH(FMT = *[,ERR = s][,IOSTAT = ios])[iolist]
PUNCH f[,iolist]
PUNCH *[,iolist]

READ(UNIT = un,FMT = f[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
READ(un,FMT = f[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
READ(un, f[,END = s][,ERR = s][,IOSTAT = ios])[iolist]

READ(UNIT = un,FMT = *[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
READ(un,FMT = *[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
READ(un, *[,END = s][,ERR = s][,IOSTAT = ios])[iolist]

```
READ(UNIT = un,FMT = name[,END = s][,ERR = s][,IOSTAT = ios])
READ(         un,FMT = name[,END = s][,ERR = s][,IOSTAT = ios])
READ(         un,        name[,END = s][,ERR = s][,IOSTAT = ios])

READ f[,iolist]
READ *[,iolist]
READ(UNIT = *,FMT = f[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
READ(UNIT = *,FMT = *[,END = s][,ERR = s][,IOSTAT = ios])[iolist]

READ(UNIT = un[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
READ(         un[,END = s][,ERR = s][,IOSTAT = ios])[iolist]

READ(UNIT = un,FMT = f,REC = rn[,ERR = s][,IOSTAT = ios])[iolist]
READ(         un,FMT = f,REC = rn[,ERR = s][,IOSTAT = ios])[iolist]
READ(         un,        f,REC = rn[,ERR = s][,IOSTAT = ios])[iolist]
READ(         un'rn,FMT = f       [,ERR = s][,IOSTAT = ios])[iolist]
READ(         un'rn,      f       [,ERR = s][,IOSTAT = ios])[iolist]

READ(UNIT = un,REC = rn[,ERR = s][,IOSTAT = ios])[iolist]
READ(         un,REC = rn[,ERR = s][,IOSTAT = ios])[iolist]
READ(         un'rn       [,ERR = s][,IOSTAT = ios])[iolist]

REAL v [,v...]
```

Summary of FORTRAN Statements (Cont.)

Form

REREAD(FMT = f[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
REREAD(FMT = *[,END = s][,ERR = s][,IOSTAT = ios])[iolist]
REREAD f[,iolist]
REREAD *[,iolist]

RETURN [e]

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

SAVE [a[,a]...]

SKIPFILE un
SKIPFILE (UNIT = un[,ERR = s][,IOSTAT = ios])
SKIPFILE (un[,ERR = s][,IOSTAT = ios])

SKIPRECORD un
SKIPRECORD (UNIT = un[,ERR = s][,IOSTAT = ios])
SKIPRECORD (un[,ERR = s][,IOSTAT = ios])

STOP [n]

SUBROUTINE sub [([d1 [,d2]...])]

TYPE(FMT = f[,ERR = s][,IOSTAT = ios])[iolist]
TYPE(FMT = *[,ERR = s][,IOSTAT = ios])[iolist]
TYPE f[,iolist]
TYPE *[,iolist]

v = e

UNLOAD un
UNLOAD (UNIT = un[,ERR = s][,IOSTAT = ios])
UNLOAD (un[,ERR = s][,IOSTAT = ios])

WRITE(UNIT = un,FMT = f[,ERR = s][,IOSTAT = ios])[iolist]
WRITE(un,FMT = f[,ERR = s][,IOSTAT = ios])[iolist]
WRITE(un, f[,ERR = s][,IOSTAT = ios])[iolist]

WRITE(UNIT = un,FMT = *[,ERR = s][,IOSTAT = ios])[iolist]
WRITE(un,FMT = *[,ERR = s][,IOSTAT = ios])[iolist]
WRITE(un, *[,ERR = s][,IOSTAT = ios])[iolist]

Summary of FORTRAN Statements (Cont.)

Form

```
WRITE(UNIT=un,FMT=name[,ERR=s][,IOSTAT=ios])
WRITE(      un,FMT=name[,ERR=s][,IOSTAT=ios])
WRITE(      un,        name[,ERR=s][,IOSTAT=ios])

WRITE f[,iolist]
WRITE *[,iolist]
WRITE(UNIT=*,FMT=f[,ERR=s][,IOSTAT=ios])[iolist]
WRITE(UNIT=*,FMT=*[,ERR=s][,IOSTAT=ios])[iolist]

WRITE(UNIT=un[,ERR=s][,IOSTAT=ios])[iolist]
WRITE(      un[,ERR=s][,IOSTAT=ios])[iolist]

WRITE(UNIT=un,FMT=f,REC=rn[,ERR=s][,IOSTAT=ios])[iolist]
WRITE(      un,FMT=f,REC=rn[,ERR=s][,IOSTAT=ios])[iolist]
WRITE(      un,      f,REC=rn[,ERR=s][,IOSTAT=ios])[iolist]
WRITE(      un'rn,FMT=f,     [,ERR=s][,IOSTAT=ios])[iolist]
WRITE(      un'rn,      f    [,ERR=s][,IOSTAT=ios])[iolist]

WRITE(UNIT=un,REC=rn[,ERR=s][,IOSTAT=ios])[iolist]
WRITE(      un,REC=rn[,ERR=s][,IOSTAT=ios])[iolist]
WRITE(      un'rn    [,ERR=s][,IOSTAT=ios])[iolist]
```

OPEN Statement Specifiers

Argument	Possible Value
ACCESS=	Character expression with one of the following values: 'SEQIN', 'SEQOUT', 'SEQINOUT', 'SEQUENTIAL', 'DIRECT', 'RANDOM', 'RANDIN', 'APPEND'
ASSOCIATEVARIABLE=	Integer variable or integer array element
BLANK=	Character expression with one of the following values: 'NULL', 'ZERO'
BLOCKSIZE=	Integer expression
BUFFERCOUNT=	Integer expression
CARRIAGECONTROL=	Character expression with one of the following values: 'FORTRAN', 'LIST', 'DEVICE'
DENSITY=	Character expression with one of the following values: '200', '556', '800', '1600', '6250', 'SYSTEM'
DEVICE=	Character expression
DIALOG	
DIALOG=	Character expression
DIRECTORY=	Character expression
DISPOSE=	Character expression with one of the following values: 'SAVE', 'DELETE', 'PRINT', 'KEEP', 'LIST', 'PUNCH', 'EXPUNGE'
ERR=	Statement number
FILE=	Character expression
FILESIZE=	Integer expression
INITIALIZE=	

OPEN Statement Specifiers (Cont.)

Argument	Possible Value
FORM =	Character expression with one of the following values: 'FORMATTED', 'UNFORMATTED'
IOSTAT =	Integer variable or integer array element
MODE =	Character expression with one of the following values: 'ASCII', 'LINED', 'BINARY', 'IMAGE', 'DUMP'
NAME =	Character expression
PADCHAR =	A character expression in which the first character is used
PARITY =	Character expression with one of the following values: 'ODD', 'EVEN'
PROTECTION = (TOPS–10)	3–digit octal constant, integer variable, or integer array element
PROTECTION = (TOPS–20)	6–digit octal constant, integer variable, or integer array element
READONLY	
RECL =	Integer expression
RECORDSIZE =	
STATUS =	Character expression with one of the following values:
TYPE =	'OLD', 'NEW', 'SCRATCH', 'EXPUNGE', 'UNKNOWN', 'KEEP', 'DELETE'
UNIT =	Integer expression
VERSION =	Octal constant, integer variable, or integer array element

CLOSE Statement Specifiers

Argument	Possible Value
DEVICE=	Character expression
DIALOG	
DIALOG=	Character expression
DIRECTORY=	Character expression
DISPOSE=	Character expression with one of the following values: 'SAVE', 'DELETE', 'PRINT', 'KEEP', 'LIST', 'PUNCH', 'RENAME', 'EXPUNGE'
ERR=	Statement number
FILE=	Character expression
IOSTAT=	Integer variable or integer array element
NAME=	Character expression
PROTECTION= (TOPS–10)	3–digit octal constant, integer variable, or array element
PROTECTION= (TOPS–20)	6–digit octal constant, integer variable, or array element
STATUS=	Character expression with one of the following values:
TYPE=	'KEEP', 'DELETE', 'EXPUNGE'
UNIT=	Integer expression

FORMATTING

Repeatable Edit Descriptors

Edit Descriptor	Descriptor Type	Default Field Width
[r]I[w[.m]]	Integer	I15
[r]F[w.d]	Floating Point	Single prec. *F15.7/double prec. *F25.18
[r]E[w.d[Ee]]	Scientific Notation	Single prec. E15.7/double prec. E25.18
[r]D[w.d[Ee]]	Scientific Notation	Single prec. D15.7/double prec. D25.18
[r]G[w.d[Ee]]	General Conversion	Single prec. G15.7/double prec. G25.18
F,E,D,G (Two successive)	Complex	
[r]O[w[.m]]	Octal	Single prec. O15/double prec. O25
[r]Z[w[.m]]	Hexadecimal	Single prec. Z15/double prec. Z25
[r]L[w]	Logical	L15
[r]A[w]	Character or Hollerith	Single prec. A5/double prec. A10
[r]R[w]	Hollerith	Single prec. R5/double prec. R10

Key:

r is a nonzero, unsigned, integer constant called a repeat specification.

w is a nonzero, unsigned, integer constant which is equal to the total number of characters in the numeric field being described.

.m is an unsigned, integer constant which specifies the minimum number of digits to be output to the field being described. If necessary, leading zeros are output. The value of m must not exceed the value of w.
If m is zero and the value of the internal data item is zero, the output field consists of only blank characters, regardless of the sign control in effect.

.d is a nonzero, unsigned, integer constant which specifies the total number of digits to the right of the decimal point in the numeric field being described. If .d is specified, w must also be specified.

e is a nonzero, unsigned, integer constant which is equal to the total number of digits in the exponent field of the numeric field being described.

* If the default field width for F format is too small for the data, the field width expands to fit the data.

Nonrepeatable Edit Descriptors

Edit Descriptor	Function
'h1...hn'	Character Data
nHh	Hollerith Data
Tc	In–Record Positioning
TLc	In–Record Positioning
TRc	In–Record Positioning
[n]X	In–Record Positioning
$	(Dollar sign) Prevents record from terminating with END OF LINE
/	(Slash) Record Delimiter
:	(Colon) Format–Control Termination
S	Plus Sign Control for Output of Positive Numeric Fields
SP	Plus Sign Control for Output of Positive Numeric Fields
SS	Plus Sign Control for Output of Positive Numeric Fields
kP	Scaling Factor for Numeric Fields
BN	Specifies the handling of blanks during the input of Numeric Fields
BZ	Specifies the handling of blanks during the input of Numeric Fields
Q	Input Only Descriptor — returns the number of characters left in the current record.

Key:

n is a nonzero, unsigned, integer constant which is equal to a number of spaces (X descriptor) or the total number of characters (H descriptor).

h is a character capable of representation by the processor.

c is a nonzero, unsigned, integer constant which is equal to a number of characters within a record relative to the current position.

k is an optionally signed integer constant which declares the scaling factor for the field being described.

Carriage–Control Specifiers

Specifier	Format List Form	Printer Character	Octal Value	Effect on Carriage Control
blank	' '	LF	012	Skip to next line (form feed after 60 lines on printer).
plus	'+'			Suppress line feed; overprint the line.
zero	'0'	LF,LF	012,012	Skip a line.
one	'1'	FF	014	Form feed to top of next page.
two*	'2'	DLE	020	Space to next half page.
three*	'3'	VT	013	Space to next one-third of a page.
minus	'–'	LF,LF,LF	012,012,012	Skip two lines.
asterisk*	'*'	DC3	023	Skip to next line; suppress form feed. (Continuous print)
period*	'.'	DC2	022	Triple space, with a form feed after every 20 lines printed.
comma*	','	DC1	021	Double space, with a form feed after every 30 lines printed.
slash*	'/'	DC4	024	Space to next one-sixth of a page.

* Indicates carriage-control specifiers for which the effect on carriage control is device dependent. The effect described is for a line printer with a standard form setup.

Note: This table assumes a standard form setup for your line printer (or other output device).

Effect of Data Magnitude on G–Format Output Conversions

Data Magnitude (m)	Effective Conversion
m .LT. 0.1	Ew.d
0.1 .LE. m .LT. 1.0	F(w–n).d,n(x)
1.0 .LE. m .LT. 10.0	F(w–n).(d–1),n(x)
.	.
.	.
.	.
10**d–2 .LE. m .LT. 10**d–1	F(w–n).1,n(x)
10**d–1 .LE. m .LT. 10**d	F(w–n).0,n(x)
m .GE. 10**d	Ew.d

where:

x is a blank
n is 4 for Gw.d and e + 2 for Gw.dEe

FUNCTIONS AND SUBROUTINES

FORTRAN Intrinsic Functions

Name	Definition	Argument Type	Function Type
		Exponential	
EXP*	y = e**x	Real	Real
DEXP	y = e**x	Double	Double
CEXP	w = e**z	Complex	Complex
		Logarithm	
ALOG*	y = log(x)	Real	Real
DLOG	y = log(x)	Double	Double
CLOG	w = log(z)	Complex	Complex
ALOG10*	y = log(x) [base 10]	Real	Real
DLOG10	y = log(x) [base 10]	Double	Double

Square Root

SQRT*	y = SQRT(x) = x**1/2	Real	Real
DSQRT	y = SQRT(x) = x**1/2	Double	Double
CSQRT	w = SQRT(z) = z**1/2	Complex	Complex

Trigonometric

SIN*	y = sin(x)	Real	Real
SIND	y = sin(x) (degrees)	Real	Real
DSIN	y = sin(x)	Double	Double
CSIN	w = sin(z)	Complex	Complex
COS*	y = cos(x)	Real	Real
COSD	y = cos(x) (degrees)	Real	Real
DCOS	y = cos(x)	Double	Double
CCOS	w = cos(z)	Complex	Complex
TAN*	y = tan(x)	Real	Real
DTAN	y = tan(x)	Real	Double
COTAN	y = cot(x)	Real	Real
DCOTAN	y = cot(x)	Double	Double

FORTRAN Intrinsic Functions (Cont.)

Name	Definition	Argument Type	Function Type
	Inverse Trigonometric		
ASIN*	y = arcsin(x)	Real	Real
DASIN	y = arcsin(x)	Double	Double
ACOS*	y = arccos(x)	Real	Real
DACOS	y = arccos(x)	Double	Double
ATAN*	y = arctan(x)	Real	Real
DATAN	y = arctan(x)	Double	Double
ATAN2*	y = arctan(arg1/arg2)	Real	Real
DATAN2	y = arctan(arg1/arg2)	Double	Double
	Hyperbolic		
SINH*	y = sinh(x)	Real	Real
DSINH	y = sinh(x)	Double	Double
COSH*	y = cosh(x)	Real	Double
DCOSH	y = cosh(x)	Double	Double
TANH*	y = tanh(x)	Real	Real
DTANH	y = tanh(x)	Double	Double

Absolute Value

ABS*	y = |x|	Real	Real
IABS	y = |i|	Integer	Integer
DABS	y = |x|	Double	Double
CABS	y = |z|	Complex	Real

Truncation

AINT*	Sign of arg * largest integer .LT. |arg|	Real	Real
INT*	Sign of arg * largest integer .LT. |arg|	Real	Integer
IDINT	Sign of arg * largest integer .LT. |arg|	Double	Integer
DINT	Sign of arg * largest integer .LT. |arg|	Double	Double

Nearest Whole Number

ANINT*	y = int(x + 5) if x .GE. 0 else y = int(x − .5)	Real	Real
DNINT*	y = int(x + .5) if x .GE. 0 else y = int(x − .5)	Double	Double

FORTRAN Intrinsic Functions (Cont.)

Name	Definition	Argument Type	Function Type
	Nearest Integer		
NINT*	y = int(x + .5)if x .GE. 0 else y = int(x −.5)	Real	Integer
IDNINT	y = int(x + .5) if x .GE. 0 else y = int(x −.5)	Double	Integer
	Remaindering		
AMOD	Remainder when arg1 is divided by arg2	Real	Real
MOD*	Remainder when arg1 is divided by arg2	Integer	Integer
DMOD	Remainder when arg1 is divided by arg2	Double	Double

Maximum Value (MAX = Generic Function)

Maximum Value (MAX = Generic Function)

AMAX0	Argument with greatest value	Integer	Real
AMAX1*	Argument with greatest value	Real	Real
MAX0	Argument with greatest value	Integer	Integer
MAX1	Argument with greatest value	Real	Integer
DMAX1	Argument with greatest value	Double	Double

Minimum Value (MIN = Generic Function)

AMIN0	Argument with least value	Integer	Real
AMIN1*	Argument with least value	Real	Real
MIN0	Argument with least value	Integer	Integer
MIN1	Argument with least value	Real	Integer
DMIN1	Argument with least value	Double	Double

Transfer of Sign

SIGN*	If arg2 .GE. 0 then |arg1| else −|arg1|	Real	Real
ISIGN	If arg2 .GE. 0 then |arg1| else −|arg1|	Integer	Real
DSIGN	If arg2 .GE. 0 then |arg1| else −|arg1|	Double	Double

FORTRAN Intrinsic Functions (Cont.)

Name	Definition	Argument Type	Function Type
	Positive Difference		
DIM*	If arg1 .GT. arg2 then arg1 − arg2 else 0	Real	Real
IDIM	If arg1 .GT. arg2 then arg1 − arg2 else 0	Integer	Integer
DDIM	If arg1 .GT. arg2 then arg1 − arg2 else 0	Double	Double
	Double–Precision Product		
DPROD	arg1*arg2	Real	Double
	Conversion Routines		
CONJG	arg = x + i*y, CONJG = x − i*y	Complex	Complex
REAL*	arg = x + i*y returns x	Complex	Real
AIMAG	arg = x + i*y returns y	Complex	Real
CMPLX*	Returns arg1 + i*arg2	Real	Complex
DFLOAT	Integer to double-precision	Integer	Double
DBLE*	Real to double-precision	Real	Double
SNGL	Double-precision to real	Double	Real

FLOAT	Integer to real	Integer	Real
IFIX	Real to integer	Real	Integer
ICHAR	Character to Integer	Character	Integer
CHAR	Integer to Character	Integer	Character
	Length		
LEN	Length of character entity	Character	Integer
	Index of a Substring		
INDEX	Return location of arg2 within arg1 if not found return 0	Character	Integer
	Character Comparisons		
LGE	arg1 .GE. arg2	Character	Logical
LGT	arg1 .GT. arg2	Character	Logical
LLE	arg1 .LE. arg2	Character	Logical
LLT	arg1 .LT. arg2	Character	Logical

* Generic function

FORTRAN–Supplied External Functions

The following are the FORTRAN–supplied external functions:

x = DTOG(y)	returns a G–floating double-precision number in the range 1.47 × 10**–39 to 1.70 × 10** + 38. The argument y is a D–floating double-precision number.
x = GTOD(y)	returns a D–floating double-precision number in the range 1.47 × 10**–39 to 1.70 × 10** + 38. The argument y is a G–floating double-precision number.
x = LSNGET(unit)	Returns the last line number read in a line sequenced file. LSNGET returns a positive integer if the last line has a valid line number; returns zero if the last line is a page mark; or returns –1 if the last line number is invalid (such as, AAAA with bit 35 set). It also returns –1 if the file contains no line number, or was opened with a mode other than LINED.
x = RAN(0)	Returns a pseudo random floating-point number in the range of 0.LT.x.LT.1. The argument is a dummy (not used) and may be any number. Refer to the related subroutines SETRAN and SAVRAN.
x = RANS(0)	Returns a pseudo random floating-point number in the range of 0.LT.x.LT.1. RANS is a prime-modulus random number generator with shuffling capability. It calls RAN to generate its initial table of random deviates. Refer to the related subroutines SETRAN and SAVRAN.

x = TIM2G0(0)

Returns the number of seconds remaining in the job's run-time limit. The time limit is set by the /TIME switch when submitting the batch job. The argument is a dummy (not used) and may be any number.
You may also specify a time limit for an interactive job by using the SET TIME–LIMIT command on TOPS–20, or the SET TIME command on TOPS–10.

FORTRAN–Supplied Subroutines

The following are the FORTRAN–supplied subroutines:

dpres = CDABS(dparg)	Returns the double-precision absolute value of the specified double-precision complex number.
CALL CDCOS(dparg,dpres)	Finds the complex cosine of the specified double-precision complex number.
CALL CDEXP(dparg,dpres)	Finds the complex exponential of the double-precision complex number you specify.
CALL CDLOG(dparg,dpres)	Returns the complex logarithm of a specified double-precision complex number.
CALL CDSIN(dparg,dpres)	Returns the complex sine of the double-precision complex number specified.
CALL CDSQRT(dparg,dpres)	Returns the complex square root of the double-precision complex number specified.
CALL CHKDIV(unitvar)	Returns the number of the unit to which error messages are being written. Returns the value −1 if the messages are being sent to the terminal.
CALL CLRFMT(arrayname)	Discards the FORMAT statement saved by the execution of the SAVFMT subroutine.
CALL DATE(name)	Places the current date, left-justified, in a character variable.
CALL DIVERT(un)	Enables you to redirect error messages from the current device to an open file on a specified device.

Effect of Data Magnitude on C Format Output Conversions

Note: This table assumes a standard form setup for your line printer (or other output device).

CALL DTOGA(sname,dname,n)	Converts elements of double-precision arrays from D–floating double-precision format to G–floating double-precision format.
CALL DUMP(LB1,UB1,format1 [...,LBn,UBn,formatn])	Causes specified portions of memory to be dumped to the line printer (LPT:).
CALL ERRSET(n) CALL ERRSET(n,i) CALL ERRSET(n,i,subr) CALL ERRSNS(I)	Controls the output of arithmetic error messages during program execution.
CALL ERRSNS (I,J) CALL ERRSNS (I,J,MSG)	Determines the reason for an error trapped by ERR= on an OPEN, CLOSE, or data transfer operation. Returns integer values that describe the status of the last I/O operation.
CALL EXIT	Terminates the program and returns control to the monitor.
CALL FFUNIT(iunit)	Returns the number of the first available FORTRAN logical unit.
CALL GTODA (sname,dname,n)	Converts elements of double-precision arrays from G–floating double-precision format to D–floating double-precision format.
CALL ILL	Sets the ILLEG flag. If this flag is set and an illegal character is encountered in floating-point/double-precision input, the corresponding 36–bit value is set to zero.
CALL LEGAL	Clears the ILLEG flag set by the ILL subroutine.

FORTRAN–Supplied Subroutines (Cont.)

CALL OVERFL(IANS)	Returns information about overflow, underflow, and divide check.
CALL PDUMP(LB1,UB1,format1 [...,LBn,UBn,formatn])	Functions exactly like the DUMP subroutine except that control returns to the calling program after the dump has been executed.
CALL SAVFMT(name,arraysize)	Directs FOROTS to encode FORMAT specifications contained in the specified character variable or array.
CALL SAVRAN(n)	Saves the last internal integer seed value generated by the RAN function. The RAN function returns a random number each time it is called.
CALL SETRAN(n)	Specifies the internal integer seed value for the RAN function. If the SETRAN argument is zero, RAN uses its own default starting value.
CALL SORT('sort string')	Sorts one or more files using the SORT program. You can successfully use this subroutine only if the SORT software has been installed.
CALL TIME(x) CALL TIME(x,y)	Returns the current time of day in left-justified ASCII.
CALL TRACE	Generates a list of active subprograms on the terminal. An active subprogram is one that has been called but has not yet returned. The main program is always active.

Key:

Loading and Starting FORDDT

On TOPS–10, the simplest method of debugging with FORDDT is:

.DEBUG filespec(DEBUG)

On TOPS–20, the corresponding command is:

@DEBUG filespec /DEBUG

On both systems, FORDDT responds with:

```
STARTING FORTRAN DDT

Program name:
```

When FORDDT prompts you for a program name, type the same name specified in the PROGRAM statement of the program being debugged. If the PROGRAM statement is not used in the program being debugged, FORDDT uses MAIN., and will not prompt for a program name.

FORDDT next prints its command prompt:

```
>>
```

The angle brackets indicate that FORDDT is ready to receive a command.

Loading and Starting FORDDT (Cont.)

You may wish to load your compiled program and FORDDT directly with the linking loader. (Loading with LINK was accomplished implicitly in the DEBUG command string.) The command sequence is as follows:

On TOPS–10, to start LINK, type:

.R LINK

On TOPS–20, to start LINK, type:

@LINK

On both systems, when LINK prompts you with an asterisk, you can type a command string in any of the following forms:

*filespec /DEB/G	(loads DDT)
*filespec /DEB:{FORDDT}/G {FORTRA}	(loads FORDDT)

```
*filespec /DEB:(DDT,{FORDDT})/G          (loads DDT and FORDDT)
          {FORTRA}

*filespec /DEB:({FORDDT},DDT)/G          (loads FORDDT and DDT)
          {FORTRA}
```

FORDDT Commands

The following are FORDDT Commands:

ACCEPT — Allows you to change the contents of a FORTRAN variable, array, array element, array element range, or FORMAT statement. The command format is:

ACCEPT name[/mode] value

CHARACTER — Defines the dimensions of a character array. The command format is:

CHARACTER array name ([L1:] U1[,[L2:]U2,...])

CONTINUE — Allows the program to resume execution after a FORDDT pause. After a CONTINUE is executed, the program either runs to completion or until another pause is encountered. The command format is:

CONTINUE [n]

DDT — Transfers control of the program to DDT, the standard system debugging program. Any files currently opened by FOROTS are unaffected, and a return to FORDDT is possible so that program execution may be resumed.

%FDDT is the global symbol used to return control to FORDDT. The command format is:

%FDDT<ESC>G

DIMENSION

Sets, displays, or removes the user-defined dimensions of an array for FORDDT access purposes. These dimensions need not agree with those declared to the compiler in the source code. FORDDT allows you to redimension an array to have a larger scope than that of the source program. If this is done, a warning is given. The command format is:

DIMENSION name ([L1:]U1[,[L2:]U2, . . .])

The DIMENSION command cannot be used to declare double-precision, complex, or character arrays (see the CHARACTER and DOUBLE commands).

DOUBLE

Defines the dimensions of a double-precision or complex array. The command format is:

DOUBLE arrayname ([L1:]U1[,[L2:]U2, . . .])

GOTO

Allows you to continue your program from a point other than the one at which it last paused. The command format is:

GOTO n

GROUP

Sets up a string of text for input to a TYPE command. You can store TYPE statements as a list of variables identified by the numbers 1 through 8. The command format is:

GROUP [n list]

FORDDT Commands (Cont.)

LOCATE — Lists the program unit names in which a given symbol is defined. The command format is:

LOCATE n

MODE — Defines the display format for succeeding FORDDT TYPE commands. The command format is:

MODE list

NEXT — Allows you to cause FORDDT to trace source lines, statement labels, and entry point names during execution of your program. The command format is:

NEXT [n][/sw]

OPEN — Allows you to open a particular program unit of the loaded program so that the variables are accessible to FORDDT. The command format is:

OPEN name

PAUSE

Allows you to place a FORDDT breakpoint at a statement number, source line number, or subroutine entry point. The command formats include:

```
PAUSE
PAUSE p
PAUSE p AFTER n
PAUSE p IF condition
PAUSE p TYPING /g
PAUSE p AFTER n TYPING /g
PAUSE p IF condition TYPING /g
```

REMOVE

Removes the previously set FORDDT breakpoints. The command format is:

```
REMOVE [p]
```

START

Starts your program at the normal FORTRAN main program entry point. The command format is:

```
START
```

FORDDT Commands (Cont.)

STOP	Terminates the program, closes all files opened by FOROTS, and causes an exit to the monitor. The usual command format is: STOP STOP /RETURN
STRACE	Displays a subprogram level traceback of the current state of the program. The command format is: STRACE
TYPE	Displays FORTRAN defined variables, arrays, or array elements on your terminal. The command format is: TYPE list
WHAT	Displays on your terminal the name of the currently open program unit, any currently active breakpoints, any group specifications and, any user-set array dimensions. The command format is: WHAT

Logical Device Assignments

Default Devices (inaccessible to the user)

Device	Default Filename	Logical Unit Number	Use
PLT	FORPLT.DAT	-7	For use by FORPLT
Device last read	File last read	-6	REREAD statement
CDR	FORCDR.DAT	-5	READ statement
TTY	FORTTY.DAT	-4	ACCEPT statement
LPT	FORLPT.DAT	-3	PRINT statement
PTP	FORPTP.DAT	-2	PUNCH statement
TTY	FORTTY.DAT	-1	TYPE statement

Logical Device Assignments (Cont.)

Standard Devices*

Device	Default Filename	Logical Unit Number	Use
DSK	FOR00.DAT	00	Disk
DSK	FOR01.DAT	01	Disk
CDR	.	02	Card Reader
LPT	.	03	Line Printer
CTY	.	04	Console Teletype
TTY	.	05	User's Teletype
PTR	.	06	Paper Tape Reader
PTP	.	07	Paper Tape Punch
DIS	.	08	Display
DTA1	.	09	DECtape
DTA2	.	10	.
DTA3	.	11	.
DTA4	.	12	.
DTA5	.	13	.
DTA6	.	14	.
DTA7	.	15	DECtape
MTA0	.	16	Magnetic Tape

MTA1	.	17	.
MTA2	.	18	.
FORTR	.	19	Assignable Device
DSK	.	20	Disk
DSK	.	21	.
DSK	.	22	.
DSK	.	23	.
DSK	.	24	.
DEV1	FOR25.DAT	25	Assignable Devices
DEV2	.	26	.
DEV3	.	27	.
DEV4	.	28	.
DEV5	.	29	.
DSK	.	30	Disk
DSK	.	31	.
.	.	.	.
.	.	.	.
.	.	.	.
DSK	FOR99.DAT	99	Disk

* The device table can be altered when FOROTS is installed or by the system administrator. The supplied options are either values in the default table pictured above, or all positive logical unit numbers default to disk. Check to see which device table is being used at your installation.

DEVICE and MODE Cross–Table

	MODE=				
Device	**'ASCII'**	**'LINED'**	**'BINARY'**	**'IMAGE'**	**TOPS–10 'DUMP'**
Disk (sequential)	X	X	X	X	X
Disk (direct)	X		X	X	
DECtape	X	X	X	X	X
Terminal	X				
Magtape	X	X	X	X	X
Line Printer	X			X	
Card Reader	X		X	X	
Card Punch	X		X	X	
Paper Tape Reader	X		X	X	
Paper Tape Punch	X		X	X	

Argument Types and Type Codes

Type Code (Octal)	Description	
	FORTRAN Use	COBOL Use
0	Unspecified	Unspecified
1	FORTRAN Logical	Not applicable
2	Integer	1–word COMP
3	Reserved	Reserved
4	Real	COMP–1
5	Reserved	Reserved
6	Octal	Reserved
7	Label	Procedure address
10	Double real (D–floating)	Not applicable
11	Not applicable	2–word COMP
12	Double octal	Not applicable
13	Double real (G–floating)	Not applicable
14	Complex	Not applicable
15	Character	Byte string descriptor
16	Reserved	Reserved
17	Hollerith	Not applicable

Argument Types and Type Codes (Cont.)

Literal arguments are permitted, but they must reside in a writable segment. This is because the FORTRAN compiler makes a local copy of all non-array elements and may copy dummy arguments back to the actual arguments. All unused type codes are reserved for future DIGITAL development.

FOROTS Error Codes and ERRSNS Values

1st Value	2nd Value[5]	Code[1]	Meaning
0			No error detected
	0		No error detected
1			Arithmetic trap
	0	IDC[3]	Integer divide check
2			Input Conversion Error
	0	IOV[3]	Integer overflow
3			Input Conversion Error
	0	FOV[3]	Floating overflow
4			Arithmetic trap
	0	FOV[3]	Floating overflow
5			Arithmetic trap
	0	FDC[3]	Floating divide check
6			Arithmetic trap
	0	FUN[3]	Floating underflow

FOROTS Error Codes and ERRSNS Values (Cont.)

1st Value	2nd Value[5]	Code[1]	Meaning
7			Input Conversion Error
	0	FUN[3]	Floating underflow
9			Output Conversion Error
	0	FTS[3]	Output field width too small
21			FORLIB errors and warnings
	104	IDU	DIVERT: illegal to divert to unit
	105	UNO	DIVERT: unit not open
	106	NOF	DIVERT: unit not open for formatted I/O
	107	CWU	DIVERT: can't write to unit
	108	CLE	Concatenation result longer than expected
	109	ICE	Illegal length character expression
	110	NCS	No character stack allocated
	111	NCA	No memory available for character stack
	112	AQS	First argument of SORT must be a quoted string
	115	TMA	Too many arguments in call to SORT
	116	CGP	Can't get pages 600:677 for SORT
	117	CRP[1]	Can't return pages 600:677 after call to SORT

	118	NSS[2]	No free section available for SORT
	119	CFS[2]	Can't find SYS:SORT.EXE
22			I/O warnings
	120	CGS[2]	Can't get SYS:SORT.EXE
	509	ETL[3]	Attempt to WRITE beyond fixed-length record
	583	FVM[3]	Format and variable type do not match
	584	RIF[3]	Reading into FORMAT non-standard
23			FORLIB bounds check warnings
	113	SSE[3]	Substring range error
	114	SRE[3]	Subscript range error
24			End of file
	-1	EOF	End of file
25			Record or record number error
	302	BBF	Bad format binary file
	510	RNR	Attempt to read a record that has not been written
	512	IRN	Illegal record number
	536	CBI	Cannot backspace image file with no RECORDSIZE
	572	RSM	Record size different from that specified in OPEN
	573	FCL	Unexpected continuation LSCW found
	576	WBA	Attempt to WRITE beyond variable or array
	577	SLN	Record length negative or zero

FOROTS Error Codes and ERRSNS Values (Cont.)

1st Value	2nd Value[5]	Code[1]	Meaning
26			OPEN/CLOSE warnings
	J	UMO[2,3]	Cannot set tape parameter
	535	BSI[3]	BLOCKSIZE ignored: device is not a magnetic tape
	541	UOA[3]	Unknown OPEN keyword, ignored
	542	NCK[3]	OPEN–only keyword specified in CLOSE, ignored
	543	RND[3]	No filename specified — DISPOSE = 'RENAME' ignored
	549	DSS[3]	DISPOSE = 'SAVE' assumed — device is not disk
	550	CQF[1,3]	Cannot QUEUE file
	588	IAU[3]	Illegal attribute for unformatted file
28			CLOSE error
	J	CLF[2]	Cannot CLOSE file
	J	RNM[2]	Cannot RENAME file
	250 + n	CLS[1]	"Close" FILOP. error n[4]
	250 + n	DEL[1]	"Delete" FILOP. error n[4]
	250 + n	RNM[1]	"Rename" FILOP. error n[4]
	527	FD1	File to rename is not on DISK or DEC tape
	528	FD2	File to rename to is not on DISK or DEC tape

		OPEN error
J	OPE[2]	Cannot OPEN file
J	UFS[2]	Cannot switch file to UNFORMATTED
J	APP[2]	Cannot set up to append to magnetic tape file
240	RR1	Random I/O requires RECORDSIZE specifier in OPEN statement
240	RRR	Random I/O requires /RECORDSIZE
242	NFC[1]	Too many OPEN units
245	NSD	No such device
248	IAC	Specified ACCESS is illegal for this device
249	IDM	Specified MODE is illegal for this device
250 + n	OPN[1]	Cannot OPEN file
405	PPN[2]	JSYS error — PPN cannot be translated
506	ICA	Incompatible attributes
523	SNM	No file specification information allowed for SCRATCH files
540	SDO	Same device open on another unit with conflicting specifiers
585	IAV	Illegal value for OPEN specifier
589	IGN[2]	Illegal generation number

FOROTS Error Codes and ERRSNS Values (Cont.)

1st Value	2nd Value[5]	Code[1]	Meaning
31			Mixed access modes
	315	CDI	Cannot do SEQUENTIAL access to a RANDOM file
			Cannot do RANDOM access to a SEQUENTIAL file
	593	POI	Illegal for DIRECT (RANDOM) files
	594	CDF	Can't determine whether formatted or unformatted
	595	IMO	Industry tapes must be opened MODE = 'IMAGE'
32			Illegal logical unit number
	239	IUN	Illegal unit number
39			REREAD error
	310	RBR	REREAD not proceeded by READ
45			OPEN/CLOSE statement syntax error
	J	JSE/JSA[2]	Parse error in DIALOG
	241	ESV	Unknown or ambiguous keyword
	241	USW[1]	Unknown switch
	241	ASW[1]	Ambiguous switch

	533	DLT	Dialog string too long
	539	EDS/EDA[2]	Error parsing DIALOG string
	544	NDI[1]	No device specified with ":"
	545	IPP[1]	Illegal PPN
	546	TMF[1]	Too many SFDs
	547	NSI[1]	Null SFD
	548	IDD[1]	Illegal character in DIALOG string
	551	NQS	PADCHAR must be single character in double quotes
47			WRITE on READ–only file
	263	CDT	Cannot WRITE to READ–only file
	554	CWL	Cannot write a file with MODE = LINED
62			Syntax error in FORMAT
	301	ILF	Illegal character in FORMAT
	306	DLF	Data in I/O list but not in FORMAT
	524	RIC	Reading into character format illegal
	532	ARC[3]	Ambiguous repeat count
	538	IRC	Illegal repeat count
	552	IHC	Illegal Hollerith constant
	553	IFW	Illegal field width
	583	FVF	Format and variable type do not match

FOROTS Error Codes and ERRSNS Values (Cont.)

1st Value	2nd Value[5]	Code[1]	Meaning
64			Input conversion error
	307	ILC	Illegal character in data
81			FOROTS calling error
	501	UNS	Unit not specified
	504	WNA	Wrong number of arguments
	508	IOL	Bad I/O list
	574	IMV	Illegal MTOP value
	579	IDI	Illegal DUMP mode I/O list
	581	DLL	Dump mode I/O list too long
	582	IWI	Illegal to initiate another I/O statement
	586	IKV	Illegal keyword value
	599	ICE	Illegal length for character expression

96			Error in magnetic tape operation
	J	ILM[2]	Unexpected MTOPR% error[2]
	530	UTE[1]	Unexpected TAPOP. error
	531	UME[1]	Unexpected MTCHR. error
	537	UTO[1]	Unexpected TAPOP. error trying to set parameters
97			Unclassifiable data error
	309	VNN	Variable not in namelist
	513	NEQ	"=" not found in namelist data
	514	NRP	Missing right paren
	515	ILN	Variable or namelist does not start with letter
	516	ILS	Illegal subscript
	519	CCC	Cannot convert constant to correct type
	520	STL	Alpha string too long
	521	RPE	Illegal repeat count
	522	SNV	Sign with null value
	580	NLS	Null string illegal
	596	NEC	Found character when expecting ":"
	597	ISS	Substring descriptor illegal
	598	SNQ	String not within single quotes

FOROTS Error Codes and ERRSNS Values (Cont.)

1st Value	2nd Value[5]	Code[1]	Meaning
98			Unclassifiable device errors
	J	OSW[2]	Cannot switch to output
	J	ISW[2]	Cannot switch to input
	J	IOE[2]	General purpose I/O error
	250 + n	ISW[1]	Cannot switch to input
	250 + n	OSW[1]	Cannot switch to output
	400	IOE[1]	General-purpose I/O error
	590	DQE	Disk full or quota exceeded
	591	ETC	Please EXPUNGE, then type CONTINUE
		CCP[1,6]	Cannot create page
		CDP[1,6]	Cannot destroy page
		CGD[6]	Can't get DBMS
		DBM[6]	DBMS not loaded
		DST[1,6]	Error in dialog string
		EFS[6]	Enter correct file specs
		FFX[6]	FOROP function code exceeds range

IEM[6]	Error in memory management
IJE[2,6]	"Impossible" JSYS error
INI[6]	INQUIRE not implemented
MFU[6]	Memory full
NOR[3,6]	Error number out of range
POV[6]	PDL overflow
SHN[6]	Internal FOROTS error
TDT[6]	Trap occurred during trap processing

1 TOPS–10 only
2 TOPS–20 only
3 This is a warning, not an error. The error cannot be trapped with an ERR= branch, but IOSTAT and ERRSNS will be set.
4 See the TOPS–10 Monitor Calls Manual for the list of FILOP. error codes and their meanings.
5 "J" means the TOPS–20 JSYS error code. This number will be between 600000 and 610000 (octal).
6 No ERRSNS values

Comparison of Real, D–floating, and G–floating Numbers

	Bits of Exponent	Bits of Mantissa	Range	Digits of Precision
Real	8	27	1.47 x 10**–39 to 1.70 x 10** + 38	8.1
D–floating	8	62	1.47 x 10**–39 to 1.70 x 10** + 38	18.6
G–floating	11	59	2.78 x 10**–309 to 8.99 x 10** + 307	17.7

Legal Dummy and Actual Argument Associations

Dummy Argument Type \ Actual Argument Type	Alternate Return Label	Logical	Integer	Real	D–floating	G–floating	Complex	Character	Octal	Hollerith	Double Octal
Alternate Return Label	X										
Logical		X							X	X	
Integer			X						X	X	
Real				X					X	X	
D–floating					X					X	X
G–floating						X				X	X
Complex							X			X	X
Character								X			

X indicates legal associations. All others will cause a warning to be issued if /DEBUG:ARGUMENTS is specified.

ASCII Character Codes

Even Parity Bit	7–Bit Decimal	7–Bit Octal	Character	Class[1]	Remarks
0	000	000	NUL		Null, tape feed. Control @(control shift P[2]).
1	001	001	SOH	CC	Start of heading [SOM, start of message]. Control A.
1	002	002	STX	CC	Start of text [EOA, end of address]. Control B.
0	003	003	ETX	CC	End of text [EOM, end of message]. Control C.
1	004	004	EOT	CC	End of transmission; shuts of TWX machines and disconnects some data sets. Control D.
0	005	005	ENQ	CC	Enquiry [WRU, "Who are you?"]. Triggers identification ("Here is ...") at remote station if so equipped. Control E.
0	006	006	ACK	CC	Acknowledge [RU, "Are you ...?"]. Control F.

1	007	007	BEL		Bell (audible or attention signal). Control G.
1	008	010	BS	FE	Backspace. Control H.
0	009	011	HT	FE	Horizontal tabulation. Control I.
0	010	012	LF[3]	FE	Line feed. Control J.
1	011	013	VT[3]	FE	Vertical tabulation. Control K.
0	012	014	FF[3]	FE	Form feed (to top of next page). Control L.
1	013	015	CR	FE	Carriage return (to beginning of line). Control M.
1	014	016	SO		Shift out; change character set or change ribbon color to red. Control N.
0	015	017	SI		Shift in; return to standard character set or color. Control 0.
1	016	020	DLE	CC	Data link escape [DC0]. Control P.
0	017	021	DC1		Device control 1, turns transmitter (reader) on. Control Q (X ON).
0	018	022	DC2		Device control 2 turns punch or auxiliary on. Control R (TAPE, AUX ON).
1	019	023	DC3		Device control 3, turns transmitter (reader) off. Control S (X OFF).

ASCII Character Codes (Cont.)

Even Parity Bit	7–Bit Decimal	7–Bit Octal	Character	Class[1]	Remarks
0	020	024	DC4		Device control 4 (stop), turns punch or auxiliary off. Control T (TAPE,AUX OFF).
1	021	025	NAK	CC	Negative acknowledge [ERR, error]. Control U.
1	022	026	SYN	CC	Synchronous idle [SYNC]. Control V.
0	023	027	ETB	CC	End of transmission block [LEM, logical end of medium]. Control W.
0	024	030	CAN		Cancel [S_0]. Control X.
1	025	031	EM		End of medium [S_1]. Control Y.
1	026	032	SUB		Substitute [S_2]. Control Z.
0	027	033	ESC		Escape, prefix [S_3]. Control [(control shift K[2]).
1	028	034	FS	IS	File separator [S_4]. Control \ (control shift L[2]).

0	029	035	GS	IS	Group separator [S_5]. Control] (control shift M[2]).
0	030	036	RS	IS	Record separator [S_6]. Control ^ (control shift N[2]).
1	031	037	US	IS	Unit separator [S_7]. Control – (control shift O[2]).

1 CC communication control, FE format effector, IS information separator.
2 On LT33, LT35 and similar units.
3 Includes a carriage return on some equipment, but not on standard DIGITAL units.

Graphic Characters

Figures				Upper Case				Lowercase			
Even Parity Bit	7–Bit Decimal	7–Bit Octal	Character	Even Parity Bit	7–Bit Decimal	7–Bit Octal	Character	Even Parity Bit	7–Bit Decimal	7–Bit Octal	Character
1	032	040	SP	1	064	100	@	0	096	140	`[2]
0	033	041	!	0	065	101	A	1	097	141	a
0	034	042	"	0	066	102	B	1	098	142	b
1	035	043	#	1	067	103	C	0	099	143	c
0	036	044	$	0	068	104	D	1	100	144	d
1	037	045	%	1	069	105	E	0	101	145	e
1	038	046	&	1	070	106	F	0	102	146	f
0	039	047	'	0	071	107	G	1	103	147	g
0	040	050	(	0	072	110	H	1	104	150	h
1	041	051	)	1	073	111	I	0	105	151	i
1	042	052	*	1	074	112	J	0	106	152	j
0	043	053	+	0	075	113	K	1	107	153	k
1	044	054	,	1	076	114	L	0	108	154	l
0	045	055	−	0	077	115	M	1	109	155	m

0	046	056	.	0	078	116	N	1	110	156	n
1	047	057	/	1	079	117	O	0	111	157	o
0	048	060	Ø[1]	0	080	120	P	1	112	160	p
1	049	061	1	1	081	121	Q	0	113	161	q
1	050	062	2	1	082	122	R	0	114	162	r
0	051	063	3	0	083	123	S	1	115	163	s
1	052	064	4	1	084	124	T	0	116	164	t
0	053	065	5	0	085	125	U	1	117	165	u
0	054	066	6	0	086	126	V	1	118	166	v
1	055	067	7	1	087	127	W	0	119	167	w
1	056	070	8	1	088	130	X	0	120	170	x
0	057	071	9	0	089	131	Y	1	121	171	y
0	058	072	:	0	090	132	Z	1	122	172	z
1	059	073	;	1	091	133	[	0	123	173	{
0	060	074	<	0	092	134	\[2]	1	124	174	|
1	061	075	=	1	093	135	]	0	125	175	}[4]
1	062	076	>	1	094	136	^[2]	0	126	176	~[2,5]
0	063	077	?	0	095	137	—	1	127	177	DEL[6]

Graphic Characters (Cont.)

1 Zero — slash absent on many units.

2 Under study by responsible American National Standards Committee for possible change at next revision of ASCII (ca. 1982).

3 Codes 140–173 first defined in 1965. For a full ASCII character set the operating system accepts codes 140–176 as lower case. For a program requiring a character set that lacks lower case, the operating system translates input codes 140–174 into the corresponding upper case codes (100–134) and translates both 175 and 176 into 033, escape. Early versions of the DECsystem–10 Monitor used 175 as the escape code and translated both 176 and 033 to it.

4 Unassigned control character (usually ALT MODE) before 1965. Code generated by ALT MODE key on some DIGITAL units, especially earlier ones; on some more recent units, the ALT key generates the standard escape code, 033.

5 Control character ESC before 1965; code generated by ESC key on some DIGITAL units designed at that time.

6 Delete, rub out (not part of lower case set).

Remarks on Special Graphic Characters

SP Space — normally nonprinting.
! Exclamation point.
" Quotation mark, diaeresis.
Number sign. £ on some (non–DIGITAL) units.
$ Dollar sign.
% Percent.
& Ampersand.
' Apostrophe, closing single quotation mark, acute accent.
  ′ in appearance on some DIGITAL units.
(Opening parenthesis.
) Closing parenthesis.
* Asterisk.
+ Plus.
, Comma, cedilla.
− Hyphen, minus.
. Period, decimal point.
/ Slant, slash, solid us.
: Colon.
; Semicolon.
< Less than.
= Equals.
> Greater than.
? Question mark.

Remarks on Special Graphic Characters (Cont.)

@ Commercial at. ` 1965–67, but never on DIGITAL units.

[Opening bracket. Shift K on LT33, LT35 and similar units.

\ Reverse slant. ~ 1965–67, but never on DIGITAL units. Shift L on LT33, LT35 and similar units.

] Closing bracket. Shift M on LT33, LT35 and similar units.

^ Circumflex, upward arrow head. ↑ before 1965, but used until 1972 on DIGITAL units.

— Underline, underscore. ← before 1965, but used until 1972 on DIGITAL units.

` Grave accent, opening single quotation mark. @ 1965–67, but never on DIGITAL units.

{ Opening brace.

| Vertical line. Control character ACK before 1965; 1965–67, but never on DIGITAL units; ¦ in appearance 1968–1977, but generally not on DIGITAL units.

} Closing brace. Unassigned control character (usually ALT MODE) before 1965.

~ Overline, tilde, general accent. Control character ESC before 1965; | 1965–67, but never on DIGITAL units.

$$y = \ln(x+1)$$

NUNU*

Printed in U.S.A.