VAX-11 FORTRAN Language Summary

software

October 1982

VAX-11 FORTRAN

Language Summary

Order No. AV-H768A-TE

digital equipment corporation · maynard, massachusetts

Contents

Symbols and Conventions

- Brackets [ ] enclose optional language constructs.
- Braces { } enclose a list of items from which one and only one item must be chosen.
- Ellipsis points (. . .) indicate that an item may be repeated one or more times. If the ellipses are preceded with a comma [, . . .] the items must be separated by commas.
- Lines that are not part of the language syntax (that are VAX-11 extensions to the FORTRAN-77 standard).

Where to Find More Information

The following documents contain more detailed information:

- The *VAX-11 FORTRAN Language Reference Manual* contains detailed information on the FORTRAN language elements mentioned in this booklet.
- The *VAX-11 FORTRAN User's Guide* describes how to compile, link, debug, and execute programs written in the VAX-11 FORTRAN language, using the facilities of the VAX/VMS operating system. It contains other information of interest to FORTRAN programmers, such as FORTRAN input/output processing, programming guidelines, compatibility between VAX-11 FORTRAN and VAX-11 FORTRAN 66, and compatibility between VAX-11 FORTRAN and PDP-11 FORTRAN.

For a list of other related VAX/VMS documents, see the *VAX-11 Information Directory and Index*.

1.0 VAX-11 FORTRAN COMMAND FORMATS

1.1 FORTRAN Command

FORTRAN [/qualifiers] file-spec-list [/qualifiers]

Qualifiers

/CHECK= { [NO]BOUNDS / [NO]OVERFLOW / [NO]UNDERFLOW / ALL / NONE }

/NOCHECK

Specifies whether the compiler generates code to perform run-time checks for the specified conditions. Default is /CHECK=(NOBOUNDS,NOOVERFLOW,NOUNDERFLOW)

/CONTINUATIONS=n

Specifies the maximum number of continuation lines allowed in a source statement. Default is n=19.

/[NO]CROSS_REFERENCE

Specifies whether a cross-reference listing is generated as part of the listing file. Default is /NOCROSS_REFERENCE.

/DEBUG= { [NO]SYMBOLS / [NO]TRACEBACK / ALL / NONE }

Specifies whether the compiler provides information for use by the VAX-11 Symbolic Debugger and run-time traceback mechanism. Default is /DEBUG=(NOSYMBOLS,TRACEBACK).

/[NO]D_LINES

Specifies whether the compiler reads and compiles lines that start with a D in column 1 in the source program. Default is /NOD_LINES.

/DML

Specifies that the DML preprocessor is to be invoked.

/[NO]F77

Specifies whether the FORTRAN 77 interpretation rules are used for those statements having a meaning incompatible with FORTRAN 66. Default is /F77.

/[NO]G_FLOATING

Specifies how the compiler interprets REAL*8, COMPLEX*16, DOUBLE PRECISION, and DOUBLE COMPLEX quantities. Default is /NOG_FLOATING.

/[NO]I4

Specifies how the compiler interprets INTEGER and LOGICAL declarations that do not specify a length. Default is /I4.

/LIBRARY

Indicates that the input file is a text library.

/LIST[:file-spec]
/NOLIST

Specifies whether a source listing file is produced. Default is /NOLIST (interactive), /LIST (batch).

/[NO]MACHINE_CODE

Specifies whether the listing file includes a symbolic listing of the machine language code generated by the compiler. Default is /NOMACHINE_CODE.

/OBJECT[:file-spec]
/NOOBJECT

Specifies whether the compiler produces an object module. Default is /OBJECT.

/[NO]OPTIMIZE

Specifies whether the compiler optimizes the compiled program to generate more efficient code. Default is /OPTIMIZE.

/SHOW=(
[NO]INCLUDE
[NO]MAP
[NO]PROCESSOR
ALL
NONE
)

Specifies whether optionally listed source lines or the entire symbol map appear in the source listing. If /CROSS_REFERENCE is specified, the symbol map is always generated. The default is /SHOW (NOPREPROCESSOR, NOINCLUDE, MAP).

/STANDARD=(
[NO]SYNTAX
[NO]SOURCE_FORM
ALL
NONE
)

/NOSTANDARD

Specifies whether the compiler generates informational diagnostics for extensions to FORTRAN-77. Default is /NOSTANDARD.

/[NO]WARNINGS

Specifies whether the compiler generates I and W diagnostic messages in response to informational and warning-level errors. Default is /WARNINGS.

1.2 LINK Command

LINK [command-qualifiers] file-spec[,file-qualifiers],...

Command-qualifiers

/[NO]EXECUTABLE[=file-spec]

Specifies whether the linker produces an executable image. Default is /EXECUTABLE.

/[NO]SHAREABLE[=file-spec]

Specifies whether the image generated by the linker must be able to resolve references resolved and must be linked with one or more object modules to produce an executable image. Default is /NOSHAREABLE

/MAP[=file-spec]

Specifies whether a map file is generated. Default is /NOMAP (interactive), /MAP (batch)

/BRIEF

Specifies that the map file contain a summary of the image's characteristics and a list of contributing modules is to be produced.

/FULL

Specifies that the map file contain a summary of the image's characteristics, a list of contributing modules, a list of global symbols and values, and a summary of the characteristics of each section in the linked image.

/[NO]CROSS_REFERENCE

Specifies whether cross-reference information for global symbols is produced as part of the map file. Default is /NOCROSS_REFERENCE

/[NO]DEBUG

Specifies whether the VAX-11 Symbolic Debugger is included in the executable image. Default is /NODEBUG

/[NO]TRACEBACK

Specifies whether the linker includes traceback information in the image file. Default is /TRACEBACK.

File-qualifiers

/LIBRARY

Specifies that the input file is an object module or shareable image library that is searched to resolve unresolved symbols referenced in other input modules.

/INCLUDE=module-name[s]

Specifies that the input file is an object module or shareable image library and only module names specified are explicitly included as input to the linker.

1.3 RUN Command

RUN [/[NO]DEBUG] file-spec

Qualifier

/[NO]DEBUG

Specifies whether the image is run with the VAX-11 Symbolic Debugger, even if /DEBUG was not specified in the previous compile or link commands. Default is /NODEBUG

ACCEPT See READ

Arithmetic/Logical/Character Assignment
v = e

v is a variable name, an array element name, or a character substring name.

e is an expression.

ASSIGN s TO v

s is the label of a FORMAT statement or an executable statement.

v is an integer variable name.

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

u is a logical unit specifier.

ios is an I/O status specifier.

s is the label of an executable statement.

BLOCK DATA [name]

name is a symbolic name.

CALL s[([a][,a]...)]

s is a subprogram name or entry name.

a is an expression, an array name, a procedure name, or an alternate return specifier. An alternate return specifier is *s or &s, where s is the label of an executable statement.

CLOSE ([UNIT=]u[,p][,IOSTAT=ios][,ERR=s])

p is one of the following parameters:

STATUS=	SAVE
DISPOSE=	KEEP
DISP=	DELETE
	PRINT
	SUBMIT
	PRINT/DELETE
	SUBMIT/DELETE

u is a logical unit specifier.

s is the label of an executable statement.

ios is an I/O status specifier.

2.0 VAX-11 FORTRAN STATEMENT SUMMARY (Cont.)

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

cb	is a common block name.
nlist	is a list of one or more variable names, array names, or array declarators separated by commas.

CONTINUE

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

nlist	is a list of one or more variable names, array names, array element names, character substring names, or implied-DO lists separated by commas. Subscript expressions and substring expressions must be constant.
clist	is a list of one or more constants separated by commas and optionally preceded by r* where r is a constant, unsigned integer constant.

DECODE (c,f,b[,IOSTAT=ios][,ERR=s][,fmt])

c	is an integer expression.
f	is a format specifier.
b	is a variable name, array name, array element name, or character substring name.
ios	is an I/O status specifier.
s	is a label of an executable statement.
fmt	is an I/O list.

DEFINE FILE u(m,n,U,v)[,u(m,n,U,v)]...

u	is a logical unit specifier.
m	is a constant or variable.
n	is a constant or variable.
U	specifies user-open-only.
v	is an integer variable name.

DELETE ([UNIT=]u[,REC=r][,IOSTAT=ios][,ERR=s])
DELETE (u'r)[,IOSTAT=ios][,ERR=s])

u	is a logical unit specifier.
r	is a record specifier.
ios	is an I/O status specifier.
s	is the label of an executable statement.

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

a(d)	is an array declarator.

```
DO [s[,]] v = e1,e2[,e3]
```

s	is the label of an executable statement
v	is a variable name
e1,e2,e3	are numeric expressions

```
DO [s[,]] WHILE (e)
```

s	is a statement label
e	is a logical expression

```
ELSE
```

```
ELSE IF (e) THEN
```

e	is a logical expression

```
ENCODE (c,f,b) [IOSTAT ios,] [ERR=s] [,[list]]
```

c	is an integer expression
f	is a format specifier
b	is an array name, array element name, or character variable name
ios	is an I/O status specifier
s	is a label of an executable statement
list	is an I/O list

```
END
```

```
END DO
```

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

u	is an open unit number
ios	is an I/O status specifier
s	is the label of an executable statement

```
END IF
```

```
ENTRY name[([a[,a]...])]
```

name	is a user-specified name
a	is a variable name or an array name or a return specifier (*)

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

nlist	is a list of two or more variable names, array names, array element names, or character substring names separated by commas. Subscript expressions and substring expressions must be integer constant expressions.

EXTERNAL i[,i]...

i	is a subprogram name

FIND ([UNIT =]u,REC = r[,IOSTAT = ios][,ERR = s])
FIND (u[,r],IOSTAT = ios[,ERR = s])

u	is a logical unit specifier
r	is a direct access record number
ios	is an I/O status specifier
s	is the label of an executable statement

FORMAT (code,...)

Code	Form	Effect
A	A[w]	Transfers alphanumeric
BN	BN	Blanks are ignored
BZ	BZ	Blanks are zeros
D	Dw.d	Transfers real values
E	Ew.d[Ee]	Transfers real values
F	Fw.d	Transfers real values
G	Gw.d[Ee]	Transfers real values
H	nHc...c	Transmits characters
I	Iw[.m]	Transfers decimal values
L	Lw	Transfers logical data
O	Ow[.m]	Transfers octal values
Q	Q	Obtains record size
S	S	Removes scale options
SP	SP	Invokes sign options
SS	SS	Suppresses optional +
T	Tn	Specifies positional location
TL	TLn	Specifies relative location (−)
TR	TRn	Specifies relative location (+)
X	nX	n characters skipped
Z	Zw[.m]	Transfers hexadecimal values
$	$	Suppresses carriage return
:	:	Terminates format control

Default Field Descriptor Values

Field Descriptor	List Element	w	d	e
I, O, Z	BYTE	7		
I, O, Z	INTEGER-2,LOGICAL-2	7		
I, O, Z	INTEGER-4,LOGICAL-4	12		
O, Z	REAL-4	12		
O, Z	REAL-8	23		
O, Z	REAL-16	44		
L	LOGICAL	2		
F, E, G, D	REAL, COMPLEX-8	15	7	2
F, E, G, D	REAL-8, COMPLEX-16	25	16	2
F, E, G, D	REAL-16	42	33	3
A	LOGICAL-1	1		
A	LOGICAL-2/INTEGER-2	2		
A	LOGICAL-4/INTEGER-4	4		
A	REAL-4, COMPLEX-8	4		
A	REAL-8, COMPLEX-16	8		
A	REAL-16	16		
A	CHARACTER	n		

Effect of Data Magnitude on G Format Conversions

Data Magnitude	Effective Conversion Ew.dEe
0 ≤ m < 0.1	F(w-4,d, 0)' '
0.1 ≤ m < 1.0	F(w-4,d-1), 0(' ')
1.0 ≤ m < 10.0	F(w-4,[d-1), 0(' ')
	.
	.
	.
10**(d-2) ≤ m < 10**(d-1)	F(w-4,1, 0(' ')
10**(d-1) ≤ m < 10**d	F(w-4,0, 0(' ')
m ≥ 10**d	Ew.dEe

Carriage Control

Character	Meaning
+	Overprinting: starts output at the beginning of the current line and returns to the left margin after printing
' '	Single spacing: starts output at the beginning of the next line
'0'	Double spacing: skips one line before starting output
'1'	Paging: starts output at the top of a new page
'$'	Prompting: starts output at the beginning of the next line and suppresses carriage return at the end of the line
ASCII NUL	Overprinting with no advance: starts output at the beginning of the current line and does not return to the left margin after printing

[typ] FUNCTION nam[*n][(p[,p]...)]

typ	is a data type specifier
nam	is a symbolic name
*n	is a data type length specifier
p	is a formal name

GO TO s

s	is a label of an executable statement

GO TO (slist[),] e

slist	is a list of one or more statement labels separated by commas
e	is an integer expression

GO TO v[,(slist)]

v	is an integer variable name
slist	is a list of one or more statement labels separated by commas

IF (e) s1,s2,s3

e	is an expression
s1,s2,s3	are labels of executable statements

IF (e) st

e	is an expression
st	is any executable statement except a DO, END DO, block IF, ELSE IF, ELSE, or END IF

```
IF (e1) THEN
  block
ELSE IF (e2) THEN
  block
ELSE
  block
END IF
```

e1,e2	are logical expressions
block	is a series of zero or more FORTRAN statements

IMPLICIT typ (a[,a]...)[,typ(a[,a]...)]...
IMPLICIT NONE

typ	is a data type specifier
a	is either a single letter or two letters in alphabetical order separated by a hyphen (that is, A-Z)

```
INCLUDE 'file specification[(NO)LIST]'
INCLUDE 'file specification[module-name][(NO)LIST]'
```

file specification
: is a character constant that specifies the file to be included.

module-name
: is the name of a text module located in a text library.

/[NO]LIST
: indicates that the statements in the specified file are to be in the source listing.

INQUIRE ([par],par)...)

par	is a keyword specification having the form key = value
key	is a keyword as described below
value	depends on the keyword

Keyword	Values
Inputs	
FILE	fin
UNIT	e
DEFAULTFILE	fin
Outputs	
ACCESS	cv
BLANK	cv
CARRIAGECONTROL	cv
DIRECT	cv
ERR	s
EXIST	lv
FORM	cv
FORMATTED	cv
IOSTAT	v
KEYED	cv
NAME	cv
NAMED	lv
NEXTREC	v
NUMBER	v
OPENED	lv
ORGANIZATION	cv
RECL	v
RECORDTYPE	cv
SEQUENTIAL	cv
UNFORMATTED	cv

e
: is a numeric expression.

fin
: is a character expression.

i
: is an integer variable or integer array element.

lv
: is a logical variable or array element.

cv
: is a character variable, array element, or substring reference.

s
: is the label of an executable statement.

INTRINSIC v[,v]...

v — a VAX-11 FORTRAN built-in function name.

NAMELIST /group-name/ namelist[,] /group-name/ namelist]...

group-name — a symbolic name.

namelist — a list of one or more variables or array names.

OPEN (par[,par]...)

par — is a keyword specification in one of the following forms:

```
key
key = value
```

key — is a keyword, as described below.

value — depends on the keyword.

Keyword	Values
ACCESS	'SEQUENTIAL' 'DIRECT' 'KEYED' 'APPEND'
ASSOCIATEVARIABLE	v
BLOCKSIZE	n
BLANK	'NULL' 'ZERO'
BUFFERCOUNT	n
CARRIAGECONTROL	'FORTRAN' 'LIST' 'NONE'
DEFAULTFILE	c
DISP	(same as DISPOSE)
DISPOSE	'KEEP' or 'SAVE' 'PRINT' 'DELETE' 'SUBMIT' 'SUBMIT/DELETE' 'PRINT/DELETE'
ERR	s
EXTENDSIZE	n
FILE	c
FORM	'FORMATTED' 'UNFORMATTED'
INITIALSIZE	n
IOSTAT	v
KEY	keyspec
MAXREC	n
NAME	(same as FILE)
NOSPANBLOCKS	—
ORGANIZATION	'SEQUENTIAL' 'RELATIVE' 'INDEXED'
READONLY	—
RECL	n
RECORDSIZE	(same as RECL)

OPEN (par[,par[,...]) (Cont.)

RECORDTYPE	FIXED VARIABLE SEGMENTED
SHARED	—
STATUS	'OLD' 'NEW' 'SCRATCH' 'UNKNOWN'
TYPE	Same as STATUS
UNIT	c
USEROPEN	s

Keyword	Values
c	is a character expression, numeric array name, numeric variable name, numeric array element name or Hollerith constant.
e	is a numeric expression.
n	is a positive integer constant.
s	is a statement label.
v	is an integer variable name.
keyspec	s[,e1,e2[,type]]
	where:
	e1 is the beginning byte of the key field.
	e2 is the ending byte of the key field.
	type is either INTEGER or CHARACTER.

OPTIONS qualifier [,qualifier...]

qualifier	is one of the following:
	/[NO]G_FLOATING /[NO]I4 /[NO]F77 /[NO]CHECK= { ALL | [NO]OVERFLOW | NO[NO]BOUNDS | [NO]UNDERFLOW | NONE }

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

p	is a variable name.
c	is a constant or compile-time constant expression.

PAUSE [disp]

disp	is a decimal digit string (containing 1 to 5 digits) or a character constant.

PRINT

```
PROGRAM nam
```

nam	A symbolic name

```
READ ([UNIT=]u,[FMT=]f [,IOSTAT=ios],END=s],ERR=s]) [list]
READ f,[,list]
ACCEPT f,[,list]
```

u	is a logical unit specifier.
f	is a format specifier
ios	is an I/O status specifier
s	is a label of an executable statement
list	is an I/O list.

```
READ ([UNIT=]u,[FMT=]*[,IOSTAT=ios],END=s],ERR=s]) [list]
READ *,[,list]
ACCEPT *,[,list]
```

u	is a logical unit specifier.
*	denotes list-directed formatting.
ios	is an I/O status specifier
s	is a label of an executable statement.
list	is an I/O list.

```
READ ([UNIT=]u,[NML=]nm),[IOSTAT=ios],END=s],ERR=s])
READ nl
ACCEPT nl
```

u	is a logical unit specifier
nl	is a namelist group-name.
ios	is an I/O status specifier.
s	is a label of an executable statement.

```
READ ([UNIT=]u),IOSTAT=ios],END=s],ERR=s]) [list]
```

u	is a logical unit specifier.
ios	is an I/O status specifier.
s	is a label of an executable statement
list	is an I/O list.

```
READ ([UNIT=]u,[FMT=]f,REC=r[,IOSTAT=ios]),ERR=s]) [list]
READ (u'r[,FMT=]f,[IOSTAT=ios]),ERR=s]) [list]
```

u	is a logical unit specifier.
r	is a record specifier
f	is a format specifier
ios	is an I/O status specifier.
s	is a label of an executable statement.
list	is an I/O list.

```
READ ([UNIT=]u,[REC=]r,[IOSTAT=]ios,[ERR=]s)[iolst]
READ (u=r,[IOSTAT=]ios,[ERR=s])[iolst]
```

u	is a logical unit specifier.
r	is a record specifier.
ios	is an I/O status specifier.
s	is a label of an executable statement.
iolst	is an I/O list.

```
READ ([UNIT=]u,[FMT=]f,keyspec[,KEYID=kn][,IOSTAT=ios]
     [,ERR=s])[iolst]
READ ([UNIT=]u,keyspec[,KEYID=kn],IOSTAT=ios)
     [,ERR=s])[iolst]
```

u	is a logical unit specifier.
f	is a format specifier.
keyspec	is a key specifier (see Section 7.2.1.6).
kn	is a key of reference specifier.
ios	is an I/O status specifier.
s	is the label of an executable statement.
iolst	is an I/O list.

```
READ ([UNIT=]u,[FMT=]f,[IOSTAT=ios],[ERR=s],END=s)[iolst]
```

u	is an internal file specifier.
f	is a format specifier.
ios	is an I/O status specifier.
s	is the label of an executable statement.
iolst	is an I/O list.

```
RETURN [i]
```

i	is an integer value that indicates which alternate return is to be taken.

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

u	is a logical unit specifier.
ios	is an I/O status specifier.
s	is the label of an executable statement.

```
REWRITE ([UNIT=]u,[FMT=]f,[IOSTAT=ios],[ERR=s])[iolst]
REWRITE ([UNIT=]u,[IOSTAT=ios],[ERR=s])[iolst]
```

u	is a logical unit specifier.
f	is a format specifier.
ios	is an I/O status specifier.
s	is the label of an executable statement.
iolst	is an I/O list.

SAVE [a[,a]...]

a	is the name of a variable, an array, or a name of a common block enclosed in slashes.

Statement Function

f([p[,p]...])] = e

f	is a symbolic name.
p	is a symbolic name.
e	is an expression.

STOP [d[ap

d [p]	is a decimal digit string containing 1 to 5 digits or a character constant.

SUBROUTINE name[([p],p],p]...[])

name	is a symbolic name.
p	is a symbolic name or an alternate return specifier (*).

TYPE See VAX 11

Type Declaration

type v[,v[,v[,v[,v]]]]...

type	is one of the following data type specifiers: BYTE (OLOGICAL) LOGICAL*1 LOGICAL*2 LOGICAL*4 INTEGER INTEGER*2 INTEGER*4 REAL REAL*4 REAL*8 REAL*16 DOUBLE PRECISION COMPLEX COMPLEX*8 COMPLEX*16 DOUBLE COMPLEX CHARACTER[*n] CHARACTER[*(*)]
v	is a variable name, array name, or array declarator, function name, or an array declarator. The name can optionally be followed by a data type length specifier (*n). For character entities, the length specifier can be a number or (*).
c[n]	is an initial value or values to be assigned to the named data ground, variable or array element.

```
UNLOCK ([UNIT=]u[,IOSTAT=ios][,ERR=s])
UNLOCK u
```

u	is a logical unit specifier.
ios	is an I/O status specifier.
s	is the label of an executable statement.

```
VIRTUAL a(d[,d,...,d])[,a(d)]...
```

a(d)	is an array declarator.

```
WRITE ([UNIT=]u,[FMT=]f[,IOSTAT=ios][,ERR=s]) [iolist]
PRINT f[,iolist]
TYPE f[,iolist]
```

u	is a logical unit specifier.
f	is a format specifier.
ios	is an I/O status specifier.
s	is a label of an executable statement.
iol	is an I/O list.

```
WRITE ([UNIT=]u,[FMT=]f[,IOSTAT=ios][,ERR=s]) [iolist]
PRINT f[,iolist]
TYPE f[,iolist]
```

u	is a logical unit specifier.
f	specifies list directed formatting.
ios	is an I/O status specifier.
s	is a label of an executable statement.
iol	is an I/O list.

```
WRITE ([UNIT=]u,[NML=]nl[,IOSTAT=ios][,ERR=s])
PRINT nl
TYPE nl
```

u	is a logical unit specifier.
nl	is a namelist group-name.
ios	is an I/O status specifier.
s	is a label of an executable statement.

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

u	is a logical unit specifier.
s	is a label of an executable statement (obs.).
ios	is an I/O status specifier.
iol	is an I/O list.

WRITE ([UNIT=]u,[REC=r][,FMT=]f[,IOSTAT=ios],ERR=s[) [list]
WRITE (u'r[,f],ERR=s[) [list]

u	is a logical unit specifier.
r	is a record specifier.
f	is a format specifier.
ios	is an I/O status specifier.
s	is a label of an executable statement.
list	is an I/O list.

WRITE ([UNIT=]u,REC=r[,IOSTAT=ios],ERR=s)[list]
WRITE (u'r[,IOSTAT=ios],ERR=s)[list]

u	is a logical unit specifier.
r	is a record specifier.
ios	is an I/O status specifier.
s	is a label of an executable statement.
list	is an I/O list.

WRITE ([UNIT=]u,[FMT=]f[,IOSTAT=ios],ERR=s)[list]

u	is an internal file specifier.
f	is a format specifier.
s	is a label of an executable statement.
ios	is an I/O status specifier.
list	is an I/O list.

```
CA...  NAME "[qualifier(s)]

CANCEL ALL

CANCEL BREAK [/qualifier] address-expression[,...]
        /ALL

CANCEL EXCEPTION BREAK

CANCEL MODE

CANCEL MODULE [/qualifier] module [module...]
        /ALL

CANCEL SCOPE

CANCEL SOURCE

CANCEL TRACE [/qualifier] [address-expression]
        /ALL
        /BRANCH
        /CALL

CANCEL TYPE [/qualifier]

CANCEL WATCH [/qualifier] [address-expression]
        /ALL

<CTRL/C>

<CTRL/Y>

<CTRL/Z>

DEFINE symbol=expression [,symbol=expression...]

DEPOSIT [/qualifier] address-expression=data
 [,data...]
        /ASCII length
        /BYTE
        /DECIMAL
        /D_FLOAT
        /F_FLOAT
        /G_FLOAT
        /HEXADECIMAL
        /H_FLOAT
        /INSTRUCTION
        /LONG
        /OCTAL
        /OCTAWORD
        /QUADWORD
        /WORD

EVALUATE [/qualifier] expression[,...expression...]
        /ADDRESS
```

3.8 VAX-11 SYMBOLIC DEBUGGER COMMAND SUMMARY (Cont.)

EXAMINE [/qualifier]...addr-expr[:addr-expr] [addr-expr]:addr-expr[:...]
/ASCII[n]
/BYTE
/DECIMAL
/D_FLOAT
/FLOAT
/G_FLOAT
/HEXADECIMAL
/H_FLOAT
/INSTRUCTION
/LONG
/NOSYMBOLIC
/OCTAL
/OCTAWORD
/QUADWORD
/SYMBOLIC
/WORD

EXIT

[line-spec]

GO [address+address:address:n]

HELP [topic [subtopic...]]

SEARCH [/qualifier] [/qualifier] ... range [string]
/ALL
/NEXT
/IDENTIFIER
/STRING

SET keyword [/qualifier] parameter

SET BREAK[/qualifier] address-expression
[DO (cmd[;cmd...])]
/AFTER

SET EXCEPTION BREAK

SET LANGUAGE language-name

SET LOG

SET MARGINS { "n" / "m" "n" / "n" }

SET MAX_SOURCE_FILES

SET MODE mode-keyword [mode-keyword...]

SET MODULE [/qualifier] [module-name [module-name]...]
/ALL

```
SET OUTPUT parameter[,parameter,...]

SET SCOPE scope [/NOSYMBOLS]

SET SEARCH parameter [parameter]

SET SOURCE [/NOCOUNT] [directory_specification]

SET STEP parameter[,parameter,...]

SET TRACE [/qualifier] [address expression]
        /BRANCH
        /CALL

SET TYPE [/qualifier] [address expression]
        /NOCHANGE

SET WATCH [address expression]

SHOW BREAK

SHOW CALLS [image]

SHOW LANGUAGE

SHOW LOG

SHOW MARGIN

SHOW MAX_SOURCE_FILES

SHOW MODE

SHOW MODULE

SHOW OUTPUT

SHOW SCOPE

SHOW SEARCH

SHOW SOURCE

SHOW STEP

SHOW TRACE

SHOW TYPE

SHOW WATCH

STEP [/qualifier] [image]
        /INSTRUCTION
        /LINE
        /INTO
        /OVER
        /NOSOURCE
        /NOSYSTEM
        /SOURCE
        /SYSTEM

TYPE [[filename]] [line_number[:line_number]]
```

Functions	Number of Arguments	Generic Name	Specific Name	Type of Argument	Type of Result
Square Root[1] $\sqrt{x}$	1	SQRT	SQRT	REAL*4	REAL*4
			DSQRT	REAL*8	REAL*8
			CSQRT	REAL*16	REAL*16
			CSQRT	COMPLEX*8	COMPLEX*8
			CDSQRT	COMPLEX*16	COMPLEX*16
Natural Logarithm[2] $\log_e x$	1	LOG	ALOG	REAL*4	REAL*4
			DLOG	REAL*8	REAL*8
			QLOG	REAL*16	REAL*16
			CLOG	COMPLEX*8	COMPLEX*8
			CDLOG	COMPLEX*16	COMPLEX*16
Common Logarithm[2] $\log_{10} x$	1	LOG10	ALOG10	REAL*4	REAL*4
			DLOG10	REAL*8	REAL*8
			QLOG10	REAL*16	REAL*16
Exponential e^x	1	EXP	EXP	REAL*4	REAL*4
			DEXP	REAL*8	REAL*8
			QEXP	REAL*16	REAL*16
			CEXP	COMPLEX*8	COMPLEX*8
			CDEXP	COMPLEX*16	COMPLEX*16

Sine[2] Sin a	1	SIN	SIN DSIN QSIN CSIN CDSIN	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16
Sine[2] (degree) Sin a	1	SIND	SIND DSIND QSIND	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Cosine[2] Cos a	1	COS	COS DCOS QCOS CCOS CDCOS	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16	REAL*4 REAL*8 REAL*16 COMPLEX*8 COMPLEX*16
Cosine[2] (degree) Cos a	1	COSD	COSD DCOSD QCOSD	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Tangent[2] Tan a	1	TAN	TAN DTAN QTAN	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Tangent[2] (degree) Tan a	1	TAND	TAND DTAND QTAND	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16

(continued on next page)

4.0 VAX-11 FORTRAN GENERIC AND INTRINSIC FUNCTIONS (Cont.)

Functions	Number of Arguments	Generic Name	Specific Name	Type of Argument	Type of Result
Arc Sine[4,5] Arc Sin x	1	ASIN	ASIN DASIN CASIN	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arc Sine (degree) Arc Sin x	1	ASIND	ASIND DASIND QASIND	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arc Cosine[4,5] Arc Cos x	1	ACOS	ACOS DACOS QACOS	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arc Cosine (degree) Arc Cos x	1	ACOSD	ACOSD DACOSD QACOSD	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arc Tangent[5] Arc Tan x	1	ATAN	ATAN DATAN CATAN	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16
Arc Tangent[5,7] (degree) Arc Tan x	1	ATAND	ATAND DATAND QATAND	REAL*4 REAL*8 REAL*16	REAL*4 REAL*8 REAL*16

Arc Tangent[6.6] Arc Tangent	2	ATAN2	ATAN2 DATAN2 QATAN2	REAL-4 REAL-8 REAL-16	REAL-4 REAL-8 REAL-16
Arc Tangent[7] (Radians) Arc Tangent	2	ATAN2D	ATAN2D DATAN2D QATAN2D	REAL-4 REAL-8 REAL-16	REAL-4 REAL-8 REAL-16
Hyperbolic Sine Sinh	1	SINH	SINH DSINH QSINH	REAL-4 REAL-8 REAL-16	REAL-4 REAL-8 REAL-16
Hyperbolic Cosine Cosh	1	COSH	COSH DCOSH QCOSH	REAL-4 REAL-8 REAL-16	REAL-4 REAL-8 REAL-16
Hyperbolic Tangent Tanh	1	TANH	TANH DTANH QTANH	REAL-4 REAL-8 REAL-16	REAL-4 REAL-8 REAL-16
Absolute Value[8] Abs	1	ABS	IABS JIABS ABS DABS QABS CABS CDABS	INTEGER-2 INTEGER-4 REAL-4 REAL-8 REAL-16 COMPLEX-8 COMPLEX-16	INTEGER-2 INTEGER-4 REAL-4 REAL-8 REAL-16 REAL-4 REAL-8
		AIMAG	IIMAG JIMAG	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4

4.0 VAX-11 FORTRAN GENERIC AND INTRINSIC FUNCTIONS (Cont.)

Function	Number of Arguments	Generic Name	Specific Name	Type of Argument	Type of Result
Truncation[2] (cont.)	1	AINT	INT	REAL*4	INTEGER*2
			JINT	REAL*4	INTEGER*4
			IDINT	REAL*8	INTEGER*2
			JDINT	REAL*8	INTEGER*4
			IQINT	REAL*16	INTEGER*2
			JQINT	REAL*16	INTEGER*4
			—	COMPLEX*8	INTEGER*2
				COMPLEX*8	INTEGER*4
				COMPLEX*16	INTEGER*2
				COMPLEX*16	INTEGER*4
		IDINT	IIDINT	REAL*8	INTEGER*2
			JIDINT	REAL*8	INTEGER*4
		IQINT	IIQINT	REAL*16	INTEGER*2
			JIQINT	REAL*16	INTEGER*4
		ANINT	ANINT	REAL*4	REAL*4
			DNINT	REAL*8	REAL*8
			QINT	REAL*16	REAL*16

Names: I=sqr[9,12] [a+.5=sqr[a]]	1	AINT	ININT	REAL*4	INTEGER*2
			JNINT	REAL*4	INTEGER*4
			IDNNT	REAL*8	INTEGER*2
			JDNNT	REAL*8	INTEGER*4
			IQNNT	REAL*16	INTEGER*2
			JQNNT	REAL*16	INTEGER*4
		DNINT	IDNNT	REAL*8	INTEGER*2
			JDNNT	REAL*8	INTEGER*4
		QNINT	IQNNT	REAL*16	INTEGER*2
			JQNNT	REAL*16	INTEGER*4
		ANINT	ANINT	REAL*4	REAL*4
			DNINT	REAL*8	REAL*8
			QNINT	REAL*16	REAL*16
Zero-Extend Functions	1	ZEXT	IZEXT	LOGICAL*1	INTEGER*2
			—	LOGICAL*2	
			—	INTEGER*2	
			JZEXT	LOGICAL*1	INTEGER*4
			—	LOGICAL*2	
			—	LOGICAL*4	
			—	INTEGER*2	
			—	INTEGER*4	

continued on next page

4.0 VAX-11 FORTRAN GENERIC AND INTRINSIC FUNCTIONS (Cont.)

Functions	Number of Arguments	Generic Name	Specific Name	Type of Argument	Type of Result
Conversion to REAL*4	1	REAL	FLOATI	INTEGER*2	REAL*4
			FLOATJ	INTEGER*4	REAL*4
			—	REAL*4	REAL*4
			ANINT	REAL*4	REAL*4
			AINT(2)	REAL*8	REAL*4
			—	COMPLEX*8	REAL*4
			—	COMPLEX*16	REAL*4
Conversion to REAL*8	1	DBLE	—	INTEGER*2	REAL*8
			—	INTEGER*4	REAL*8
			DBLE	REAL*4	REAL*8
			DBLE(2)	REAL*8	REAL*8
			—	REAL*16	REAL*8
			—	COMPLEX*8	REAL*8
			—	COMPLEX*16	REAL*8

Conversion to REAL*16	1	QEXT	—	INTEGER*2	REAL*16
			—	INTEGER*4	REAL*16
			QEXT	REAL*4	REAL*16
			QEXTD	REAL*8	REAL*16
			—	REAL*16	REAL*16
			—	COMPLEX*8	REAL*16
			—	COMPLEX*16	REAL*16
IFIX[10,12] (REAL*4-to-integer conversion)	1	IFIX	IIFIX	REAL*4	INTEGER*2
			JFIX	REAL*4	INTEGER*4
Float[10] (integer to REAL*4 conversion)	1	FLOAT	FLOAT	INTEGER*2	REAL*4
			FLOATJ	INTEGER*4	REAL*4
REAL*8 Fcol[10] (integer to REAL*8 conversion)	1	DFLOAT	DFLOAT	INTEGER*2	REAL*8
			DFLOTJ	INTEGER*4	REAL*8
REAL*16 Float (integer-to-REAL*16 conversion)	1	DFLOAT	—	INTEGER*2	REAL*16
			—	INTEGER*4	REAL*16
Conversion to COMPLEX*8, or COMPLEX*8 from Two Arguments	1,2[12]	CMPLX	—	INTEGER*2	COMPLEX*8
	1,2		—	INTEGER*4	COMPLEX*8
	1,2		—	REAL*4	COMPLEX*8
	1,2		—	REAL*8	COMPLEX*8
	1,2		—	REAL*16	COMPLEX*8
	1		—	COMPLEX*8	COMPLEX*8
	1		—	COMPLEX*16	COMPLEX*8

more mathematical routines 23

more mathematical routines on next page

4.0 VAX-11 FORTRAN GENERIC AND INTRINSIC FUNCTIONS (Cont.)

Functions	Number of Arguments	Generic Name	Specific Name	Type of Argument	Type of Result
Conversion to COMPLEX-16 or COMPLEX-16 from Two Arguments	1,2[12] 1,2 1,2 1,2 1,2 1 1	DCMPLX	— — — — — — —	INTEGER-2 INTEGER-4 REAL-4 REAL-8 REAL-16 COMPLEX-8 COMPLEX-16	COMPLEX-16 COMPLEX-16 COMPLEX-16 COMPLEX-16 COMPLEX-16 COMPLEX-16 COMPLEX-16
Real Part of Complex	1	—	REAL DREAL	COMPLEX-8 COMPLEX-16	REAL-4 REAL-8
Imaginary Part of Complex	1		AIMAG DIMAG	COMPLEX-8 COMPLEX-16	REAL-4 REAL-8
Complex from Two Arguments	(See Conversion to COMPLEX-8 and Conversion to COMPLEX-16)				
Complex Conjugate i.e. (X,Y) CONJG (Z=X, Y)	1	CONJG	CONJG DCONJG	COMPLEX-8 COMPLEX-16	COMPLEX-8 COMPLEX-16
Double Precision of REAL-4/8 Arg	1	—	DPROD	REAL-4	REAL-8

Maximum[12]

MAX(p,p,...)

Returns the maximum value from among the arguments (at there must be at least two arguments)

Minimum[13]

MIN(p,p,...,p)

Returns the minimum value among the arguments (at there must be at least two arguments)

MAX	JMAX0	INTEGER-2	INTEGER-2
	JMAX1	INTEGER-4	INTEGER-4
	AMAX1	REAL-4	REAL-4
	DMAX1	REAL-8	REAL-8
	QMAX1	REAL-16	REAL-16
MAX0	JMAX0	INTEGER-2	INTEGER-2
	JMAX1	INTEGER-4	INTEGER-4
AMAX1	AMAX1	REAL-4	REAL-4
	JMAX1	INTEGER-4	INTEGER-4
AMAX0	AMAX0	INTEGER-2	REAL-4
	AMAX1	INTEGER-4	REAL-4
MIN	JMIN0	INTEGER-2	INTEGER-2
	JMIN1	INTEGER-4	INTEGER-4
	AMIN1	REAL-4	REAL-4
	DMIN1	REAL-8	REAL-8
	QMIN1	REAL-16	REAL-16
MIN0	JMIN0	INTEGER-2	INTEGER-2
	JMIN1	INTEGER-4	INTEGER-4
AMIN1	AMIN1	REAL-4	INTEGER-2
	JMIN1	REAL-4	INTEGER-4
AMIN0	AMIN0	INTEGER-2	REAL-4
	AMIN1	INTEGER-4	REAL-4

(CONTINUED ON next page)

Function	Number of Arguments	Generic Name	Specific Name	Type of Argument	Type of Result
Positive Difference $a_1 - \min(a_1, a_2)$ (returns the first argument minus the minimum of the two arguments)	2	DIM	IIDIM	INTEGER*2	INTEGER*2
			JIDIM	INTEGER*4	INTEGER*4
			DIM	REAL*4	REAL*4
			DDIM	REAL*8	REAL*8
			QDIM	REAL*16	REAL*16
		IIDIM	IIDIM	INTEGER*2	INTEGER*2
			JIDIM	INTEGER*4	INTEGER*4
Remainder $a_1 - (\text{int}(a_1/a_2) * a_2)$ (returns the remainder when the first argument is divided by the second)	2	MOD	IMOD	INTEGER*2	INTEGER*2
			JMOD	INTEGER*4	INTEGER*4
			AMOD	REAL*4	REAL*4
			DMOD	REAL*8	REAL*8
			QMOD	REAL*16	REAL*16
Transfer of Sign $\lvert a_1 \rvert$ Sign a_2	2	SIGN	ISIGN	INTEGER*2	INTEGER*2
			JISIGN	INTEGER*4	INTEGER*4
			SIGN	REAL*4	REAL*4
			DSIGN	REAL*8	REAL*8
			QSIGN	REAL*16	REAL*16
		IISIGN	IISIGN	INTEGER*2	INTEGER*2
			JISIGN	INTEGER*4	INTEGER*4

Bitwise AND (performs a logical AND on corresponding bits)	2	IAND	IIAND JIAND	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Bitwise OR (performs an inclusive OR on corresponding bits)	2	IOR	IIOR JIOR	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Bitwise Exclusive OR (performs an exclusive OR on corresponding bits)	2	IEOR	IIEOR JIEOR	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Bitwise Complement (complements each bit)	1	NOT	INOT JNOT	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Bitwise Shift (n1 logically shifted n2 bits)	2	ISHFT	IISHFT JISHFT	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Bit Extraction (extracts bits a2 through a2+a3-1 from a1, see also MVBITS system subroutine)	3	IBITS	IIBITS JIBITS	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Bit Set (returns the value of a1 with bit a2 set to 1)	2	IBSET	IIBSET JIBSET	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Bit Test (returns .TRUE. if bit a2 of argument a1 equals 1)	2	BTEST	BITEST BJTEST	INTEGER-2 INTEGER-4	LOGICAL-1 LOGICAL-4

(continued on next page)

Functions	Number of Arguments	Generic Name	Specific Name	Type of Argument	Type of Result
Bit Clear (returns the value of i with bit b_2 of b_1 set to 0)	2	IBCLR	BCLR IBCLR	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Shift Circular (provides a shift operation of a_2 bits of argument a_1 by a_3 places)	3	ISHFTC	ISHFTC JISHFTC	INTEGER-2 INTEGER-4	INTEGER-2 INTEGER-4
Length[1] (returns length of the character expression)	1	—	LEN	CHARACTER	INTEGER-4
Index c_1 c_2[2] (returns the position of the substring c_2 in the character expression c_1)	2	—	INDEX	CHARACTER	INTEGER-4
Character[3] (returns a character that has the ASCII value specified by the argument)	1	—	CHAR	LOGICAL-1 INTEGER-2 INTEGER-4	CHARACTER

ASCII Value[1] Returns the ASCII value of the argument. The argument must be a character expression that has a length of 1.	1	—	CHAR	CHARACTER*1	INTEGER*4
Character relationals (ASCII collating sequence)	2	—	LLT	CHARACTER*n	LOGICAL*4
	2	—	LLE	CHARACTER*n	LOGICAL*4
	2	—	LGT	CHARACTER*n	LOGICAL*4
	2	—	LGE	CHARACTER*n	LOGICAL*4

NOTES

1. The argument of SCOT, DSQRT, or CSQRT must be greater than or equal to zero. The result of CSQRT is the principal value with the real part greater than or equal to zero. When the real part is zero, the result is the principal value with the imaginary part greater than or equal to zero.
2. The argument of ALOG, DLOG, QNOT, ALOG10, DLOG10, QLOG10, ATAN2, ATAN2D, ASIND, DASIND, ACOSD, DACOSD, or DACOSD must be greater than zero. The argument of CLOG or CDLOG must not be (0,0).
3. The argument of SIN, DSIN, SIND, DSIND, COS, DCOS, COSD, DCOSD, TAN, DTAN, or DTAN must be in radians. The argument is treated modulo 2*π. The argument of SIND, COSD, or TAND must be in degrees. The argument is treated modulo 360.
4. The absolute value of the argument of ASIN, DASIN, QASIN, ACOS, DACOS, QACOS, ASIND, DASIND, QASIND, ACOSD, DACOSD, or QACOSD must be less than or equal to 1.

5. The result of ASIN, DASIN, CASIN, ACOS, DACOS, CACOS, ATAN, DATAN, CATAN, ATAN2, DATAN2, or CATAN2 is in radians. The result of ASIND, DASIND, CASIND, ACOSD, DACOSD, CACOSD, ATAND, DATAND, CATAND, ATAN2D, DATAN2D, or CATAN2D is in degrees.

6. If the value of the first argument of ATAN2, DATAN2, or CATAN2 is zero, the result is positive zero. The value of the first argument is zero, the result is zero if the second argument is positive and π if the second argument is negative. If the value of the first argument is negative, the result is negative. If the value of the second argument is zero, the absolute value of the result is π/2. Both arguments must not have the value zero. The range of the result for ATAN2, DATAN2, and CATAN2 is $-\pi < result \leq \pi$.

7. If the value of the first argument of ATAN2D, DATAN2D, or CATAN2D is positive, the result is positive. When the value of the first argument is zero, the result will be zero if the second argument is positive and 180 degrees if the second argument is negative. If the value of the first argument is negative, the result is negative. If the value of the second argument is zero, the absolute value of the result is 90 degrees. Both arguments must not have the value zero. The range of the result for ATAN2D, DATAN2D, or CATAN2D is $-180 < result \leq 180$ degrees.

8. The absolute value of a complex number (X,Y) is the real value:

$$(X^2 + Y^2)^{1/2}$$

9. The result of the argument integer whose magnitude shall not exceed the magnitude of the argument and whose sign is the same as the sign of the argument. For example, IDINT(3.7) equals 3 and IDINT(-3.7) equals -3.

10. Functions that cause conversion of one data type to another may provide the same effect as the implicit conversion in assignment statements. The function REAL with an integer argument, the function DBLE with a double precision argument, the function INT with an integer argument and the function DEXT with a REAL*8 argument (when the value of the argument without conversion).

11. See Chapter 4 of the *VAX-11 FORTRAN Language Reference Manual* for additional information on character functions.

12. The functions AINT, IDINT, ICHAR, IFIX, MAX1, MIN1 and IDIM (when INTEGER*4 arguments) that return INTEGER*2 values if the /I4 qualifier is given.

13. When DCMPLX and DCMPLX have only one argument, this argument is converted to the real part of a complex value and zero is assigned to the imaginary part; when there are two arguments (not complex), a complex value is produced by conversion of the first argument into the real part of the value, the second argument into the imaginary part.

5.0 ASCII CHARACTER SET

ASCII Decimal Number	Character	Meaning
0	NUL	Null
1	SOH	Start of heading
2	STX	Start of text
3	ETX	End of text
4	EOT	End of transmission
5	ENQ	Enquiry
6	ACK	Acknowledgement
7	BEL	Bell
8	BS	Backspace
9	HT	Horizontal tab
10	LF	Line feed
11	VT	Vertical tab
12	FF	Form feed
13	CR	Carriage return
14	SO	Shift out
15	SI	Shift in
16	DLE	Data link escape
17	DC1	Device control 1
18	DC2	Device control 2
19	DC3	Device control 3
20	DC4	Device control 4
21	NAK	Negative acknowledgement
22	SYN	Synchronous idle
23	ETB	End of transmission block
24	CAN	Cance
25	EM	End of medium
26	SUB	Substitute
27	ESC	Escape
28	FS	File separator
29	GS	Group separator
30	RS	Record separator
31	US	Unit separator
32	SP	Space or blank
33	!	Exclamation mark
34	"	Quotation mark
35	#	Number sign
36	$	Dollar sign
37	%	Percent sign
38	&	Ampersand
39	'	Apostrophe
40	(	Left Parenthesis
41	)	Right parenthesis
42	*	Asterisk
43	+	Plus sign
44	,	Comma
45	-	Minus sign or hyphen
46	.	Period or decimal point
47	/	Slash
48	0	Zero
49	1	One
50	2	Two
51	3	Three
52	4	Four
53	5	Five
54	6	Six

5.0 ASCII CHARACTER SET (Cont.)

ASCII Decimal Number	Character	Meaning
57	9	Seven
58	8	Eight
59	7	Nine
60	6	Colon
61	5	Semicolon
62	4	Less than (or open angle bracket)
63	3	Equal sign
64	2	Right angle bracket
65	1	Question mark
66	0	At sign
67	/	Uppercase A
68	.	Uppercase B
69	-	Uppercase C
70	,	Uppercase D
71	+	Uppercase E
72	*	Uppercase F
73	)	Uppercase G
74	(	Uppercase H
75	'	Uppercase I
76	&	Uppercase J
77	%	Uppercase K
78	$	Uppercase L
79	#	Uppercase M
80	"	Uppercase N
81	!	Uppercase O
82	~	Uppercase P
83	}	Uppercase Q
84	|	Uppercase R
85	{	Uppercase S
86	`	Uppercase T
87	_	Uppercase U
88	^	Uppercase V
89	]	Uppercase W
90	[	Uppercase X
91	Z	Uppercase Y
92	Y	Uppercase Z
93	X	Left square bracket
94	W	Back slash
95	V	Right square bracket
96	U	Circumflex or up arrow
97	T	Back arrow or underscore
98	S	Grave accent
99	R	Lowercase a
100	Q	Lowercase b
101	P	Lowercase c
102	O	Lowercase d
103	N	Lowercase e
104	M	Lowercase f
105	L	Lowercase g
106	K	Lowercase h
107	J	Lowercase i
108	I	Lowercase j
109	H	Lowercase k
110	G	Lowercase l
111	F	Lowercase m

ASCII Decimal Number	Character	Meaning
113	q	Lowercase q
114	r	Lowercase r
115	s	Lowercase s
116	t	Lowercase t
117	u	Lowercase u
118	v	Lowercase v
119	w	Lowercase w
120	x	Lowercase x
121	y	Lowercase y
122	z	Lowercase z
123	{	Left Brace
124	|	Vertical Line
125	}	Right Brace
126	~	Tilde
127	DEL	Delete