Code	Additional	Description	All Codes are in HEX

***** Operators *****

4A	=	Equal
4C	<>	Not equal
50	<	Less than
4E	>	Greater than
54	<=	Less than or equal
52	>=	Greater than or equal
0A	+	Add
0C	-	Subtract
0E	*	Multiply
10	/	Divide
42	&	Join strings
3A	&&	Bitwise AND
3C	||	Bitwise OR
3E	^^	Bitwise XOR
40	~~	Bitwise NOT
34	OR	As in IF (a OR b)
32	AND	As in IF (a AND b)
36	XOR	
46	MOD	
48	DIV	Divide (integer)
38	NOT	(float)
44	INSTR	
30	^	Raise to a power
56	==	Almost equals

***** Actual values *****

8A	[2 bytes]	An actual integer to put on stack
88	[6 bytes]	An actual (6 byte) floating point to put on stack
92	[4 bytes]	An actual (4 byte) floating point to put on the stack
8C	[undefined]	An actual string to put on stack

***** Normal variables *****

D2, 80, D4 [2 bytes] Get a variable (I don't know what the difference is)

78 [3 bytes] Get a Name list entry with a separator.
1st bye is separator
2nd word is variable reference

7E [3 bytes] Get a string slice (from an array)
1st byte is the number of indexes
2nd word is variable reference

D6, D8, 84 [2 bytes] Assign a variable

***** Arrays *****

6C [4 bytes] DIMention a integer array 1st byte is number of elements, 2nd word var ref
6A [3 bytes] DIMention a float array 1st byte is number of elements, 2nd word var ref
6E [2 bytes] DIMention a string array 1st byte is number of elements, 2nd word var ref

86 [4 bytes] Get an array element (string) ? slice?
1st byte is a separator to go afterwards e.g. for a comma - x(3),
2nd byte is the number of indexes
3rd word is the variable reference

7A [3 bytes] Get an array element
1st byte is the number of the indexes (on stack)
2nd word is the variable reference

7C [3 bytes] Assign an array element
1st byte is index number
2nd word is the variable reference

68 [3 bytes] Assign an array element e.g. x$(y,3) or x$(y, 1 TO 10)
1st byte is number of indexes
2nd word is the variable reference

***** Stack manipulation *****

14		Convert to a negative (float)
16		Duplicate the item that is on the top of the stack
66	[1 byte]	Add a parameter separator to an item on the top of the stack Like second byte of type word in QDOS 00=none 10=, 20=; 30=\ 40=! 50=TO 80=precede with #
BC	[1 byte]	Place a parameter separator on the stack. Codes as above

***** Keyword table commands *****

02		Precedes actual parameters of a command

***** Procedures and Functions *****

58		Precedes a namelist keyword function, or a Proc/Fun call
CE	[4 bytes]	Call a Proc/Fun
96	[2bytes]	Call a keyword table entry, procedure or function
76	[undefined]	Define a Proc/Fun. 1st byte is number of parameters, Repeating words are the variables
9A	[2 bytes]	LOCal variable(integer)
98	[2 bytes]	LOCal variable (float)
9C	[2 bytes]	LOCal variable (string)
72	[3 bytes]	LOCal integer array 1st byte is number of indexes
70	[3 bytes]	LOCal float array 1st byte is number of indexes
74	[3 bytes]	LOCal string array 1st byte is number of indexes
5E		RETurn/END DEF
5C	[byte 02]	End Define Function - (Don't know what the 02 is for, also seen 0)
60		RETurn a value on the stack

***** FOR loops *****

9E	[2bytes]	Get FOR control variable
A0	[6 bytes]	Used in mixed selections 1st word is the variable reference 2nd long is a pointer to just past the END FOR
A4	[undefined]	Start the FOR 1st word is the variable reference 2nd long is a pointer to the statements There are then 0 to the number of (selections-1) long words that point at the selection number+1 There is then either - For a simple FOR x= 1 TO 10 There is a long word that points at just past the END FOR For a mixed FOR x= 1,3,5,7 TO 10 There is a long word with the value 4, pointing to the statements
A2	[undefined]	END FOR 1st word is the variable reference 2nd long is a pointer to the statements There are then 0 to the number of (selections-1) long words that point at the selection number+1 There is then a long word with the value 4

***** IF..THEN *****

CC	[4 bytes]	offset pointer to ELSE or END IF

***** SELECT ON *****

D0	[4 bytes]	pointer to ON..=..true long offset to true section

***** Various QDOS functions *****

04	INT()
12	ABS()
18	COS()
1A	SIN()
1C	TAN()
1E	COT()
20	ASIN()
22	ACOS()
24	ATAN()
26	ACOT()
28	SQRT()
2A	LN()
2C	LOG10()
2E	EXP()
AE	CODE()
B2	CHR$()
AC	LEN()
B6	RESPR()
B8	FILL$()
B0	EOF for embedded DATA statements
62	ERNUM If 62 is followed by [8A] [2 bytes] [4A] then it's ERR_xx
64	ERLIN

***** Various QDOS commands *****

CA	[4 bytes]	GOTO	watch out for Def Proc/Fun & REPeat & IF/THEN/ELSE long word is offset to destination
A6	[4 bytes]	GO SUB	long word is offset to destination
5A		STOP	
BE	[3 bytes]	READ	1st byte is D6 = integer, D8 = float, 84 = string 80 = array, data is READ as a variable and placed on the stack. To be followed by an array assignment 2nd word variable reference
C0		DATA	Get value off the stack
A8	[4 bytes]	RESTORE	Long word pointer to DATA line + 6
94	[4 bytes]	WHEN ERRor	Long word is an offset to middle of END WHEN which is a B4, CONTINUE
C8		RETRY	Expects a word on the stack as a line number
B4		CONTINUE	

AA [6 + (4 * number of options) + (6 * number of options)] ON..GOTO, ON..GOSUB

1st word is number of values
2nd long is pointer to next line
For the number of options there is a table of long words (options)
each is a pointer from the current position to another table of GO TO/GO SUB's
Then for each option
word A6 for a GO SUB, or CA for a GO TO. Then for each option a long offset
ON GO SUB ends in CA, long offset

96 [2 bytes] Name list command, Word is name list reference (increments in 8's)