

Basic Reference

Binary Operators

Precedence	Operator	Notes
4	*	
	/	Forward slash is floating point divide. 22/7 is 3.142857
	\	Backward slash is integer divide, 22/7 is 3
	%	Modulus of integer division ignoring signs
	>>	Logical shifts up to 32 places, inserting zeros at the appropriate ends.
	<<	
3	+	
	-	
2	<	Compares as numbers or strings. If either is floating point it is compared as such, and the match is not exactly equal, but about 1 part in 100,000. Returns -1 for true, 0 for false.
	<=	
	>	
	>=	
	<>	
	=	
1	&	Binary operators on integers, but can be used as logical operators. Equivalent to and, or and exclusive or.
	^	

Unary Operators (General)

Operator	Notes
<code>alloc(n)</code>	Allocate n bytes of 65C02 memory, return adress
<code>asc(s\$)</code>	Return ASCII value of first character or zero for empty string
<code>atan(n)</code>	Arctangent of n in degrees
<code>chr\$(n)</code>	Covnert ASCII to string
<code>cos(n)</code>	Cosine of n, n ls in degrees.
<code>deek(a)</code>	Read word value at a
<code>event(v,r)</code>	
<code>exp(n)</code>	e to the power n
<code>inkey\$()</code>	Return the key stroke if one is in the keyboard buffer, otherwise returns a n empty string.
<code>int(n)</code>	Whole part of the float value n. Integers are unchanged.
<code>isval(s\$)</code>	Converts string to number, returns -1 if okay, 0 if fails.
<code>key(n)</code>	Return the state of the given key. The key is the USB HID key scan code.
<code>left\$(a\$,n)</code>	Left most n characters of a\$
<code>len(a\$)</code>	Return length of string in characters.
<code>log(n)</code>	Natural Logarithm (e.g. ln2) of n.
<code>max(a,b)</code>	Return the largest of a and b
<code>mid\$(a\$,f[,s])</code>	Characters from a\$ starting at f (1 indexed), s characters, s is optional and defaults to the rest of the line.
<code>min(a,b)</code>	Return the smaller of a and b
<code>peek(a)</code>	Read byte value at a
<code>rand(n)</code>	Random integer $0 < x < n$ (e.g. 0 to n-1)
<code>right\$(a\$,n)</code>	Rightmost n characters of a\$
<code>rnd(n)</code>	Random number $0 < x < 1$, ignores n.
<code>sin(n)</code>	Sine of n, n ls in degrees.

sqr(n)	Square root of n
str\$(n)	Convert n to a string
tan(n)	Tangent of n, n is in degrees.
time()	Return time since power on in 100 th of a seconds.
val(s\$)	Convert string to number. Error if bad number.

BASIC Commands (General)

Command	Notes
' <string>	Comment. This is a string for syntactic consistency. The tokeniser will process a line that doesn't have speech marks as this is not common. REM this is a comment is now ' "this is a comment"
assert <expr>	Error generated if <expr> is zero
call <name>()	Call procedure
clear	Clear out stack, strings, reset all variables.
dim <array>(n,[m]), ...	Dimension a one or two dimension string or number array, up to 255 elements in each dimension (e.g. 0-254)
do ... exit ... loop	General loop you can break out of at any point.
doke <addr>,<data>	Write word to address
end	End Program
for <var> = <start> to/downto <end> ... next	For loop. Note this is non standard, Limitations are : the index must be an integer. Step can only be 1 (to) or -1 (downto). Next does not specify an index and cannot be used to terminate loops using the 'wrong' index.
gosub <expr>	Call subroutine at line number. For porting only. See goto.
goto <expr>	Transfer execution to line number. For porting only. Use in general coding is a capital offence. If I write RENUMBER it will <u>not</u> support these.
if <expr> then	Standard BASIC if, executes command or line number. (IF .. GOTO doesn't work, use IF .. THEN nn)
if <expr>: .. else .. endif	Extended multiline if, without THEN. The else clause is optional.
let <var> = <expr>	Assignment statement. The LET is optional.
new	Erase Program
poke <addr>,<data>	Write byte to address
print <stuff>	Print strings and numbers, standard format.
proc <name>()..endproc	Delimits procedures
repeat .. until <expr>	Execute code until <expr> is true

return	Return from subroutine called with gosub.
run	Run Program
stop	Halt program with error
sys <address>	Call 65C02 machine code at given address. Passes contents of variables A,X,Y in those registers.
while <expr> .. wend	Repeat code while expression is true