Keywords, Commands on the PC Emulator

Command	Type	Description
any literal	data	Push a literal, the value of a variable or a string into the stack. If the literal is not a double precision number or a variable, it will be a bigint or a string (must be placed inside double quotes). All numbers are interpreted as decimal unless in quotes: “xabcd” is a hex number (bigint) “1234” is a decimal bigint. “b1100” is a binary number. 1234.5e-6 is a decimal double precision number “o7654” is an octal number. (not supported at present)
num‘r’, ‘l’, ‘c’ e.g., *4.7e-3l*	data	Any numeric literal followed by one of the characters‘r’, ‘l’ or ‘c’, indicating resistance, inductance or capacitance. The literal value will be stored on the stack but will be interpreted as an impedance value for a frequency in a variable called ‘f’. If variable ‘f’ doesn’t exist, a frequency value of 1Hz will be assumed. For example, 4.7e-3l is the impedance of a 4.7mH inductor, or 29.531*j* ohms at 1000Hz.
(*numR numI*)	data	Push a double precision complex number with real part *numR* and imaginary part *numI* into the stack.
(*numI*)	data	Push a double precision imaginary number *numI* into the stack.
“a string”	data	Push a string, which can be an arbitrarily long decimal integer (a *bigint*) into the stack. If the string is a bigint, {+ - * / max min > < >= <= = != } operators can be used.
@	operator	Pop value in ToS-1 is popped into the variable name at ToS.
var@	operator	Pop the ToS element in to a variable named *var*.
n@@	operator	Pop *n* (must be a number ≥0) entities from the stack into the *acc* register where *n* is the value on the top of the stack. A vector or matrix is considered a single element and will be popped entirely. If the last entity is a vector, it will be popped into a variable that can be accessed using the variable name *vacc*. If *n* is 0, the entire stack will be cleared. If *n* is less than 0, no values will be popped and an error will result. The Execution Stack, which tracks conditionals will not be cleared (vector/matrices entry is also tracked using the execStack, will be adjusted after this operation).
[*or*]	data	Start or end a vector.
{ *or* }	data	Start or end a matrix.
(*or*)	data	Encloses a complex number.
dup	operator	Duplicate the ToS.
swp	operator	Swap ToS with element prior to ToS.
if	conditional	Execute if the ToS element does not equal 0 or is a string.
el	conditional	Execute if the ToS element equals 0.
fi	conditional	End an if-el block
label:	label	A label used for jmp, jz or jnz.
jmp	unconditional jump	Jump to a label.
jz	conditional jump	Jump to a label if ToS is 0.
jnz	conditional jump	Jump to a label if ToS is not 0 or is a string.
+ - * / max min > < >= <= = !=	operator	The usual arithmetic operators. Works for regular numbers (complex) or bigint strings.
%	operator	The percentage operator. Calculates ToS (‘x’) percentage of ToS - 1 (‘y’)

__f*	Special variable	The internal write-only frequency variable, assigned as *numx* __f@. Use the keystroke alt+@ on the calculator. The ‘_’ is not available on the calculator.
//	operator	The parallel operator.
vec	operator	Accumulates all consecutive scalar elements between the ToS and the last barrier, vector or matrix or bottom of stack (BoS) into a vector. The ToS will be the last element and the element closest to the BoS will be the first. If the ToS is already a vector, then this command will split it into scalar elements. Use keystroke page 0 alt+[function for this operation on the calculator.
mat	operator	Accumulates all consecutive vector elements between the ToS and the last barrier, scalar or matrix or bottom of stack (BoS) into a matrix (row order). The vector at ToS will be the last row and the vector closest to the BoS will be the first row of the matrix. If the ToS is already a matrix, then this command will split it into vector elements. Use keystroke page 0 alt+{ function for this operation on the calculator.
cmplx	operator	Creates a complex number with real part from (ToS - 1) and imaginary part from ToS (if both of these are real numbers). If the ToS is a complex or imaginary number then the real and imaginary parts are separated. Use keystroke page 0 alt+(function for this operation on the calculator.
bar	operator	Inserts or deletes a barrier at the current ToS element. A barrier stops accumulates of elements for the creation of a vector or matrix. If the current ToS is not a barrier, this command inserts a barrier; otherwise, it deletes the barrier at the ToS.
angle	operator	Toggles between degrees and radian modes.
coord	operator	Toggles between cartesian and polar modes.
rect	operator	Converts the number at ToS to rectangular (cartesian) coordinate system irrespective current coordinate system.
polar	operator	Converts the number at ToS to polar coordinate system irrespective current coordinate system
deg	operator	Converts the ToS to degrees. 1 radian = 57.29577951308232087679 8 degrees.
rad	operator	Converts the ToS to radians. 1 degree = 0.01745329251994329576923 7 radians.
bin	operator	Converts the ToS to binary. Ignore if the ToS is a binary bigint.
hex	operator	Converts the ToS to hex. Ignore if the ToS is a hex bigint.
dec	operator	Converts the ToS to decimal. Ignore if the ToS is a decimal bigint.
oct	operator	Converts the ToS to octal. Ignore if the ToS is an octal bigint.
reim	operator	Swaps the real an imaginary parts of the number at ToS. `1 reim` gives `(1)` and `(1)` reim gives `1`
re	operator	Provides the real part of the complex number at ToS.
im	operator	Provides the imaginary part of the complex number at ToS.
neg	operator	Negates the number at ToS.

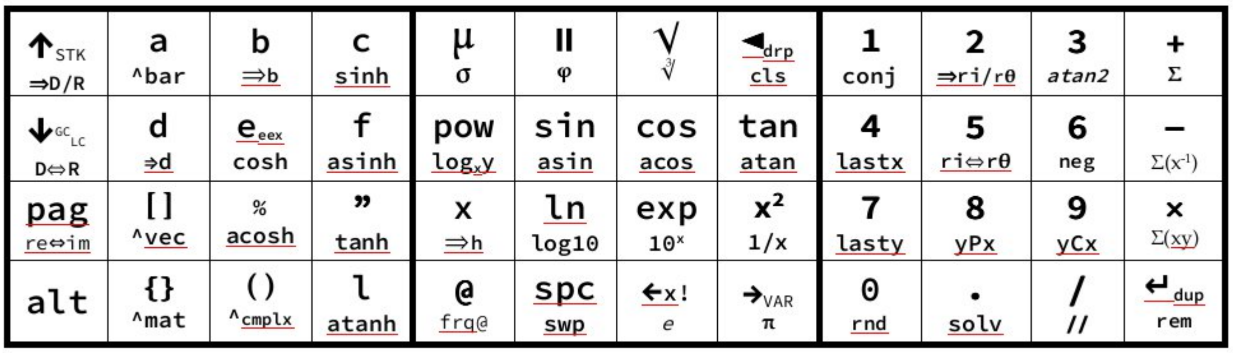
dupn*	operator	Duplicates the xth item from ToS, ToS is indexed as 0 and holds the value x. Negative or 0 values of x don't change the stack.
recip	math function	Reciprocal of the ToS (computes $1/x$).
rsum	vector operator	Sum of the reciprocal of the elements in the vector at ToS.
mean	vector operator	The mean computed over the elements in the vector at ToS.
sd	vector operator	The standard deviation computed over the elements in the vector at ToS.
var	vector operator	The variance (square of the standard deviation) computed from the values in the vector at ToS.
sqsum	vector operator	Sum of the squares of the elements in the vector at ToS.
sum	vector operator	Sum of the elements in the vector at ToS.
dot	vector operator	Sum of the product of the elements in the vector at ToS. The number of elements in the shorter vector determine how many products will be computed for the sum.
solv	matrix and vector operator	If ToS − 1 is a square matrix (not barred) and ToS is a vector, solves the simultaneous equation for the (matrix, vector). If ToS − 1 is not a matrix or is a barred matrix, and ToS is a vector, solves the polynomial equation for the vector.
conj	math function	Complex conjugate of the number at ToS.
abs	math function	Absolute value of the number x at ToS, equals *sqrt(sqr(real(x)) + sqr(imag(x)))*. This is a shorthand for: `cmplx 2 pow swp 2 pow + sqrt`
arg	math function	Argument of the number x at ToS, equals *atan(imag(x)/real(x))*
sqrt	math function	Square root of the ToS.
cbrt	math function	Cubic root of the ToS.
exp	math function	Computes e to the power of the number at ToS, where e is Euler's constant (approximately, 2.718281828..).
log	math function	Natural logarithm of the number at ToS.
pi, *e*	data	Inserts the double precision approximation of *pi* or e to ToS.
log2*	math function	Logarithm of ToS to base 2.
log10	math function	Logarithm of ToS to base 10.

logxy	math function	Logarithm of ToS -1 (‘y’) to base ToS (‘x’).
pow	math function	If ToS is a bigint and ToS - 1 is a real or bigint, computes the modular exponentiation of the bigint or integer at ToS - 1 with the bigint or integer exponent at ToS, modulo the modulus already set. If modulus has not been set or is 0, uses 2^{64} as the modulus. In this context, pow is identical to exp.
sin, cos, tan, cot*, asin, acos, atan, acot*	trig function	Trigonometric functions.
sinh, cosh, tanh, coth*, asinh, acosh, atanh, acoth*	hyberbolic trig function	Hyperbolic trigonometric functions.
atan2	trig function	Computes `y x atan` where *y* is the number at ToS - 1 and *x* is the number at ToS.
gcd, lcm	math function	Calculates the GCD and LCM of the vector at ToS. The result is a vector in the form [*gcd*, *lcm*].
fac	math function	Factorial, *x*!, where *x* is the (integer part of the real part of the) number at ToS.
inv	matrix or bigint operation	Matrix inverse if ToS is a matrix. Modular inverse if ToS is a bigint or int. If matrix inverse doesn’t exist, returns an error. For modular inverse, if the modulus is 0, uses 2^{64} as the modulus.
det	matrix operation	Computes the determinant of the matrix at ToS. If the ToS does not have a matrix, reports an error.
iden	matrix operation	
proj	matrix operation	
trace	matrix operation	
eival	matrix operation	
eivec	matrix operation	
tpose	matrix operation	
rank	matrix operation	
elem	matrix or vector operator	if ToS and ToS-1 are scalars and ToS-2 is a matrix, then returns the element matrix(ToS-2)[row=ToS-1][col=ToS]. If ToS-1 is a vector and ToS is a scalar, then returns the scalar element

		vector(ToS-1)[ToS]. If ToS-1 is a matrix and ToS is a scalar, then returns the row vector matrix(ToS-1)[ToS]
mod	bigint or integer operation	Set the modulus for bigint or integer modular operations. Used for inv, exp (pow) , +, −, ×, and / operations on bigint or integer. If modulus has not been set, uses 2^{64} as modulus. Modulus or width is required for the inv operation.
exp	complex, bigint or integer operation	If ToS is a bigint and ToS - 1 is a real or bigint, computes the modular exponentiation of the bigint or integer at ToS - 1 with the bigint or integer exponent at ToS, modulo the modulus already set. If modulus has not been set or is 0, uses 2^{64} as the modulus. In this context, exp is identical to pow. If the number x ToS is a real or complex number, computes e^x.
mont	bigint or integer operation	The montgomery representation of the bigint or integer at ToS.
wid	bigint or integer operation	Set the width in bits for bigint or integer operations. Used in or, and, binv, xor, bit, shr, shl, ror, rol, count0, count1 and rnd operations. Optionally used for bigint operations such as exp, inv etc. if modulus is 0 (not set).
and, or, and, binv, xor, shr, shl, ror, rol	bigint or integer operation	Bitwise operation if ToS and ToS - 1 are bigints or integers. Returns an error if width is not set (default is 32 bits). Shift right will extend the MS bit if the flasg signextend is set. Shift left will set zero in the LB bit position.
bit	bigint or integer operation	Test bit of number y at ToS - 1 at position x (at ToS), returns 0 or 1.
count0, count1	bigint or integer operation	Count the number of 0s or 1s in the binary representation of the bigint or integer at ToS.
1’sC, 2’sC	bigint or integer operation	One’s or two’s complement of the bigint or integer at ToS.
set, clr	bigint or integer operation	Set or clear bit at position x (at ToS) in the number y at ToS - 1, returns 0 or 1.
isp, nxp	bigint or integer operation	isp returns a 1 is ToS has a prime number. nxp returns the prime number after the number at ToS.
rnd	math operation	
gl	integer or bigint operation	Calculates the GCD and LCM of the vector at ToS. The result is a vector in the form [*gcd*, *lcm*]. Can be bigint numbers (entered and displayed as strings).
lastx	math operation	The ToS value (x) before the last operation.

lasty	math operation	The ToS - 1 value (*y*) before the last operation.
ypx	integer operation	Computes ^yP_x, where y is the integer number at ToS - 1 and x is at ToS.
ycx	integer operation	Computes ^yC_x, where y is the integer number at ToS - 1 and x is at ToS.
hms	operator	Converts the real the number (or the real part of the complex number) at ToS to three numbers that are pushed into the stack: hours, minutes and seconds, these latter being the h:m:s representation of the number. If the current angle mode is radians, then the angle is converted to degrees first.
* ***not available on calculator hardware.***		

Default mode: Page 0



↑STK ⇒D/R	a ^bar	b ⇒b	c sinh	µ σ	⏸ φ	√ ∛	◀drp cls	1 conj	2 ⇒ri/rθ	3 *atan2*	+ Σ
↓GC LC D⇔R	d ⇒d	e eex cosh	f asinh	pow $log_x y$	sin asin	cos acos	tan atan	4 lastx	5 ri⇔rθ	6 neg	− $\Sigma(x^{-1})$
pag re⇔im	[] ^vec	% acosh	” tanh	x ⇒h	ln log10	exp 10^x	x^2 1/x	7 lasty	8 yPx	9 yCx	× Σ(xy)
alt	{} ^mat	() ^cmplx	l atanh	@ frq@	spc swp	←x! *e*	→VAR π	0 rnd	• solv	/ //	↵dup rem

Keystroke	Type	Description for Page 0
↑STK ⇒D/R	primary	When pressed in compute mode, switches to stack inspection mode. In stack inspection mode, used as up cursor to point to various stack entries.
	alternate	If the current angle mode is degrees, converts the number at ToS to radians. If the current angle mode is radians, convert to degrees.
↓GC LC D⇔R	primary	When pressed in compute mode, calculates the GCD and LCM of the vector at ToS. The result is a vector in the form [*gcd*, *lcm*]. Can be bigint numbers (entered and displayed as strings).
	alternate	Toggles the current angle mode between radians and degrees. Default at star tup is radians.
pag re⇔im	primary	Cycles through the four page modes, mode 0 (default), mode 1 (additional math functions), mode **α**, the alphabet entry mode and mode **π**, the programming keyword mode and **µ**, the miscellaneous command mode.
	alternate	Swaps the real and imaginary parts of the number at ToS. ToS must contain a real or complex number.

alt	primary	Cycles between primary, alternate and alternate-locked modes.
	alternate	Not applicable.
a	primary	The character ‘a’.
^bar	alternate	Toggle the barrier attribute on the element at ToS.
d	primary	The character ‘d’.
⇒d	alternate	Convert the number at ToS to decimal representation. Only scalar elements are allowed.
[ ]	primary	The character ‘[‘ that will start the entry of a vector. If a vector is already being entered, it will close it.
^vec	alternate	Accumulates all consecutive scalar elements between the ToS and the last barrier, vector or matrix or bottom of stack (BoS) into a vector. The ToS will be the last element and the element closest to the BoS will be the first. If the ToS is already a vector, then this command will split it into scalar elements.
{}	primary	The character ‘{‘ that will start the entry of a matrix. If a matrix is already being entered, it will close it.
^mat	alternate	Accumulates all consecutive vector elements between the ToS and the last barrier, scalar or matrix or bottom of stack (BoS) into a matrix (row order). The vector at ToS will be the last row and the vector closest to the BoS will be the first row of the matrix. If the ToS is already a matrix, then this command will split it into vector elements.
b	primary	The character ‘b’. Use to begin a binary number.
⇒b	alternate	Converts the ToS to binary representation. Only scalar elements are allowed.
$\mathrm{e}_{\mathrm{eex}}$	primary	The character ‘e’. Also used to indicate than an exponent follows, e.g., 1.23e45
cosh	alternate	Hyperbolic cosine.
%	primary	Calculates x% of y, where y is value at ToS-1 and x the value at ToS.
acosh	alternate	Inverse hyperbolic cosine.
( )	primary	The character ‘(‘ that will start the entry of a complex number. If a complex number is already being entered, it will close it. The closing parenthesis is optional and the complex number will be automatically closed.
^cmplx	alternate	Creates a complex number with real part from (ToS - 1) and imaginary part from ToS (if both of these are real numbers). If the ToS is a complex or imaginary number then the real and imaginary parts are separated.
c	primary	The character ‘c’.
sinh	alternate	Hyperbolic sine.

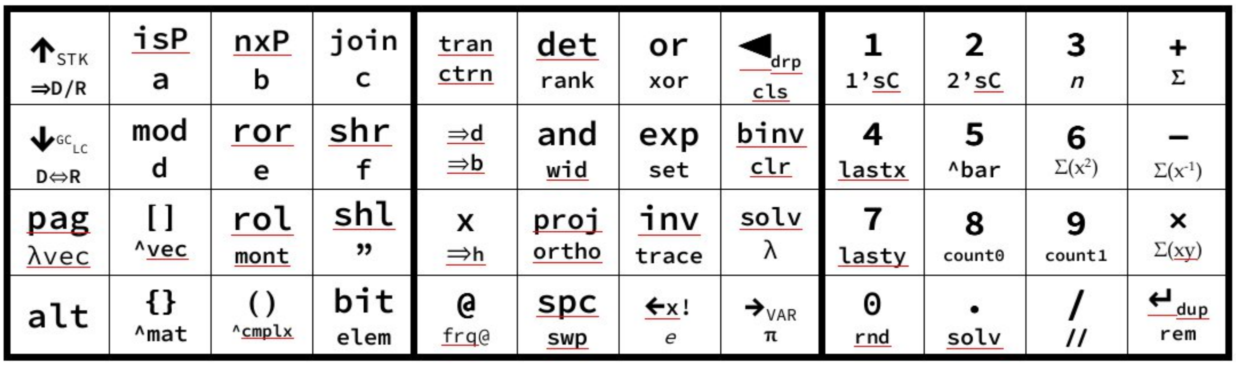
f	primary	The character ‘f’.
asinh	alternate	Inverse hyperbolic sine.
”	primary	Starts or closes a string (used for bigints). The closing quote is optional and will be automatically inserted.
tanh	alternate	Hyperbolic tangent.
l	primary	The character ‘l’, can be used as in a variable name or used to specify an inductive impedance.
atanh	alternate	
μ	primary	Inverse hyperbolic sine.
σ	alternate	The standard deviation calculated over all the elements of the vector at ToS.
pow	primary	If ToS is a bigint and ToS - 1 is a real or bigint, computes the modular exponentiation of the bigint or integerat ToS - 1 with the bigint or integerexponent at ToS, modulo the modulus already set. If modulus has not been set or is 0, uses 2^{64}as the modulus. In this context, pow is identical to exp.
log$_x$y	alternate	Logarithm of ToS -1 (‘y’) to base ToS (‘x’).
x	primary	The character ‘x’. Use to begin a hexadecimal number.
⇒h	alternate	Converts the ToS to hex. Ignore if the ToS is a hex bigint.
@	primary	Pop value in ToS-1 is popped into the variable name at ToS. Use *var*@ to store into a variable named *var*.
frq@	alternate	Pop value in ToS-1 is popped into the internal frequency variable. This is used to calculate impedances. The number at ToS must be a real; if ToS has a complex number, then the real part will be used.
||	primary	Absolute value of the number *x* at ToS, equals *sqrt(sqr(real(x)) + sqr(imag(x)))*. This is a shorthand for: cmplx 2 pow swp 2 pow + sqrt
ɸ	alternate	Argument of the number *x* at ToS, equals *atan(imag(x)/real(x))*
sin	primary	Computes the sine of the number at ToS.
asin	alternate	Computes the inverse sine of the number at ToS.
ln	primary	Computes the natural logarithm of the number at ToS.
log10	alternate	Computes the logarithm to base 10 of the number at ToS.
spc	primary	Enters the space character. Used to separate the real and imaginary portions of a complex number and the consecutive elements of a

swp		vector or matrix.
	alternate	Swap ToS with element prior to ToS (at ToS - 1).
√	primary	Computes the square root of the number at ToS.
$\sqrt[3]{}$	alternate	Computes the cube root of the number at ToS.
cos	primary	Computes the cosine of the number at ToS.
acos	alternate	Computes the inverse cosine of the number at ToS.
exp	primary	If ToS is a bigint and ToS - 1 is a real or bigint, computes the modular exponentiation of the bigint or integerat ToS - 1 with the bigint or integerexponent at ToS, modulo the modulus already set. If modulus has not been set or is 0, uses 2^{64}as the modulus. In this context, exp is identical to pow. If the number *x at* ToS is a real or complex number, computes e^x.
10^x	alternate	Computes 10^x where *x* is the number at ToS.
←x!	primary	If a user entry is being entered/edited, then this key moves the cursor to the left. If no entry is being edited, returns the factorial of the integer at ToS (or the integer portion of the complex number at ToS. The integer must be less than 210.
e	alternate	The approximate value of *e* – 2.718281828459045
◄$_{drp}$	primary	If a user entry is being entered/edited, then this key deletes the character to the left of the cursor. If the user is not editing an entry, drops the stack (removes the number or element at ToS).
cls	alternate	Clears the stack.
tan	primary	Computes the tangent of the number at ToS.
atan	alternate	Computes the inverse tangent of the number at ToS.
x^2	primary	Computes x^2 where *x* is the number at ToS.
1/x	alternate	Computes x^2 where *x* is the number at ToS.
→$_{VAR}$	primary	If a user entry is being entered/edited, then this key deletes the character to the left of the cursor. If the user is not editing an entry, this shows the detailed value of the element (number, vector or matrix) at ToS.
π	alternate	The approximate value of π – 3.141592653589793
1	primary	The character ‘1’.
conj	alternate	The complex conjugate of the element at ToS.

4	primary	The character ‘4’.
lastx	alternate	The ToS value (*x*) before the last operation.
7	primary	The character ‘7’.
lasty	alternate	The ToS - 1 value (*y*) before the last operation.
0	primary	The character ‘0’.
rnd	alternate	Uses Pico hardware ring oscillators to generate random number — a 32-bit integer.
2	primary	The character ‘2’.
⇒ri/rθ	alternate	Changes the complex number at ToS (assumed to be rectangular format (re im) in rectangular mode and polar format (r *θ*) in polar mode) to polar or rectangular format respectively.
5	primary	The character ‘5’.
ri⇔rθ	alternate	Toggles the complex number mode between rectangular and polar (default is rectangular).
8	primary	The character ‘8’.
yPx	alternate	Computes $^{y}P_{x}$, where y is the integer number at ToS - 1 and x is at ToS.
.	primary	The character ‘.’.
solv	alternate	If ToS - 1 is a square matrix (not barred) and ToS is a vector, solves the simultaneous equation for the (matrix, vector). If ToS - 1 is not a matrix or is a barred matrix, and ToS is a vector, solves the polynomial equation for the vector.
3	primary	The character ‘3’.
atan2	alternate	Computes `y x atan` where *y* is the number at ToS - 1 and *x* is the number at ToS.
6	primary	The character ‘6’.
neg	alternate	Negates the number at ToS.
9	primary	The character ‘9’.
yCx	alternate	Computes $^{y}C_{x}$, where y is the integer number at ToS - 1 and x is at ToS.
/	primary	The division operator.
//	alternate	The parallel operator. Computes *xy*/(*x* + *y*) where *x* is the number at ToS and *y* is the number at ToS - 1.
+	primary	The addition operator.
Σ	alternate	Sum of the elements in the vector at ToS. Same as the `sum` command

		on the emulator.
– Σ(x⁻¹)	primary	The subtraction operator.
	alternate	Sum of the reciprocal of the elements in the vector at ToS. Same as the rsum command on the emulator.
× Σ(xy)	primary	The multiplication operator.
	alternate	Sum of the product of the elements in the vectors at ToS and Tos – 1. Same as the dot command on the emulator. The number of elements in the shorter vector determine how many products will be computed for the sum.
↵dup rem	primary	If a user entry is being entered/edited, then pressing this key places the entry at ToS after lifting the stack. No operation takes place if the stack is full. If the user is not editing an entry, then pressing this key duplicates the entry at ToS.
	alternate	The remainder operator.

Numbers, Vectors and Matrices mode: Page 1



↑STK ⇒D/R	isP a	nxP b	join c	tran ctrn	det rank	or xor	◀drp cls	1 1’sC	2 2’sC	3 *n*	+ Σ
↓GC LC D⇔R	mod d	ror e	shr f	⇒d ⇒b	and wid	exp set	binv clr	4 lastx	5 ^bar	6 Σ(x²)	– Σ(x⁻¹)
pag λvec	[] ^vec	rol mont	shl ”	x ⇒h	proj ortho	inv trace	solv λ	7 lasty	8 count0	9 count1	× Σ(xy)
alt	{} ^mat	() ^cmplx	bit elem	@ frq@	spc swp	←x! *e*	→VAR π	0 rnd	• solv	/ //	↵dup rem

Keystroke	Type	Description
↑STK ⇒D/R	primary	When pressed in compute mode, switches to stack inspection mode. In stack inspection mode, used as up cursor to point to various stack entries.
	alternate	If the current angle mode is degrees, converts the number at ToS to radians. If the current angle mode is radians, convert to degrees.
↓GC LC D⇔R	primary	When pressed in compute mode, calculates the GCD and LCM of the vector at ToS. The result is a vector in the form [*gcd*, *lcm*]. Can be bigint numbers (entered and displayed as strings).
	alternate	Toggles the current angle mode between radians and degrees. Default at star tup is radians.
pag	primary	Cycles through the four page modes, mode 0 (default), mode 1 (additional math functions), mode **α**, the alphabet entry mode and

λvec		mode **π**, the programming keyword mode.
	alternate	Computes the eigenvector for the matrix at ToS.
alt	primary	Cycles between primary, alternate and alternate-locked modes.
	alternate	Not applicable.
isp	primary	Returns 1 if the number at ToS is a prime, 0 otherwise.
a	alternate	The character ‘a’.
mod	primary	Sets the number (integer or bigint) at ToS as the modulus for bigint inverse and exponentiation operations
d	alternate	The character ‘d’.
[ ]	primary	The character ‘[‘ that will start the entry of a vector. If a vector is already being entered, it will close it.
^vec	alternate	Accumulates all consecutive scalar elements between the ToS and the last barrier, vector or matrix or bottom of stack (BoS) into a vector. The ToS will be the last element and the element closest to the BoS will be the first. If the ToS is already a vector, then this command will split it into scalar elements.
{}	primary	The character ‘{‘ that will start the entry of a matrix. If a matrix is already being entered, it will close it.
^mat	alternate	Accumulates all consecutive vector elements between the ToS and the last barrier, scalar or matrix or bottom of stack (BoS) into a matrix (row order). The vector at ToS will be the last row and the vector closest to the BoS will be the first row of the matrix. If the ToS is already a matrix, then this command will split it into vector elements.
nxp	primary	Returns the first prime number larger than the number at ToS.
b	alternate	The character ‘b’. Use to begin a binary number.
ror	primary	Bitwise rotate right assuming the number is *width* bits wide (default 64).
e	alternate	The character ‘e’. Also used to indicate than an exponent follows, e.g., 1.23e45.
rol	primary	Bitwise rotate left assuming the number is *width* bits wide (default 64).
mont	alternate	Converts the number (integer or bigint) at ToS to its Montgomery representation.
( )	primary	The character ‘(‘ that will start the entry of a complex number. If a complex number is already being entered, it will close it. The closing parenthesis is optional and the complex number will be automatically closed.
^cmplx	alternate	Creates a complex number with real part from (ToS – 1) and imaginary part from ToS (if both of these are real numbers). If the ToS is a complex or imaginary number then the real and imaginary

		parts are separated.
join	primary	Concatenates the vectors at ToS and ToS - 1 into a single vector.
c	alternate	The character ‘c’.
shr	primary	Bitwise shift right assuming the number is *width* bits wide (default 64). The leftmost bit can be extended from the carry bit or set to 0.
f	alternate	The character ‘f’.
shl	primary	Bitwise shift left assuming the number is *width* bits wide (default 64). The carry bit can be assigned from the leftmost bit or remain unchanged.
”	alternate	Starts or closes a string (used for bigints). The closing quote is optional and will be automatically inserted.
bit	primary	Checks bit value – *y x* bit – returns value of bit at position *x* in *y*.
elem	alternate	If ToS and ToS-1 are scalars and ToS-2 is a matrix, then returns the element matrix(ToS-2)[row=ToS-1][col=ToS]. If ToS-1 is a vector and ToS is a scalar, then returns the scalar element vector(ToS-1)[ToS]. If ToS-1 is a matrix and ToS is a scalar, then returns the row vector matrix(ToS-1)[ToS]
tran	primary	Transpose of the matrix at ToS.
ctran	alternate	Conjugate transpose of the matrix at ToS.
⇛d	primary	Convert the number at ToS to decimal representation. Only scalar elements are allowed.
⇛b	alternate	Convert the number at ToS to binary representation. Only scalar elements are allowed.
x	primary	The character ‘x’. Use to begin a hexadecimal number.
⇛h	alternate	Converts the ToS to hex. Ignore if the ToS is a hex bigint.
@	primary	Pop value in ToS-1 is popped into the variable name at ToS. Use *var*@ to store into a variable named *var*.
frq@	alternate	Pop value in ToS-1 is popped into the internal frequency variable. This is used to calculate impedances. The number at ToS must be a real; if ToS has a complex number, then the real part will be used.
		inv: matrix inverse if ToS is a matrix. Modular inverse if ToS is a

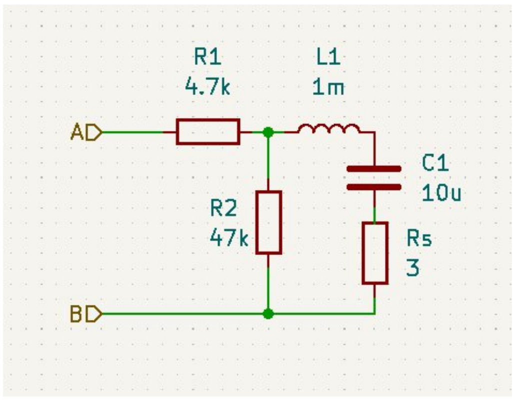
‖ ϕ	primary	bigint or int. binv: bitwise inverse if ToS is a bigint or int. 2'sC: same as neg (defined only for bigint or int), uses wid as bit-width. wid: set width. Used in or, and, binv, xor, bit, shr, shl, ror, rol, count0, count1 and rnd operations.
	alternate	Argument of the number *x* at ToS, equals *atan(imag(x)/real(x))*
sin asin	primary	Computes the sine of the number at ToS.
	alternate	Computes the inverse sine of the number at ToS.
ln log10	primary	Computes the natural logarithm of the number at ToS.
	alternate	Computes the logarithm to base 10 of the number at ToS.
spc swp	primary	Enters the space character. Used to separate the real and imaginary portions of a complex number and the consecutive elements of a vector or matrix.
	alternate	Swap ToS with element prior to ToS (at ToS – 1).
√ ∛	primary	Computes the square root of the number at ToS.
	alternate	Computes the cube root of the number at ToS.
cos acos	primary	Computes the cosine of the number at ToS.
	alternate	Computes the inverse cosine of the number at ToS.
exp 10^x	primary	If ToS is a bigint and ToS – 1 is a real or bigint, computes the modular exponentiation of the bigint or integerat ToS – 1 with the bigint or integerexponent at ToS, modulo the modulus already set. If modulus has not been set or is 0, uses 2^{64}as the modulus. In this context, exp is identical to pow. If the number *x at* ToS is a real or complex number, computes e^x.
	alternate	Computes 10^x where *x* is the number at ToS.
←x! *e*	primary	If a user entry is being entered/edited, then this key moves the cursor to the left. If no entry is being edited, returns the factorial of the integer at ToS (or the integer portion of the complex number at ToS. The integer must be less than 210.
	alternate	The approximate value of *e* – 2.718281828459045
◀$_{\text{drp}}$	primary	If a user entry is being entered/edited, then this key deletes the character to the left of the cursor.

cls		If the user is not editing an entry, drops the stack (removes the number or element at ToS).
	alternate	Clears the stack.
tan	primary	Computes the tangent of the number at ToS.
atan	alternate	Computes the inverse tangent of the number at ToS.
x^2	primary	Computes x^2 where *x* is the number at ToS.
1/x	alternate	Computes x^2 where *x* is the number at ToS.
$\rightarrow_{VAR}$	primary	If a user entry is being entered/edited, then this key deletes the character to the left of the cursor. If the user is not editing an entry, this shows the detailed value of the element (number, vector or matrix) at ToS.
π	alternate	The approximate value of π – 3.141592653589793
1	primary	The character ‘1’.
conj	alternate	The complex conjugate of the element at ToS.
4	primary	The character ‘4’.
lastx	alternate	The ToS value (*x*) before the last operation.
7	primary	The character ‘7’.
lasty	alternate	The ToS - 1 value (*y*) before the last operation.
0	primary	The character ‘0’.
rnd	alternate	Uses Pico hardware ring oscillators to generate random number – a 32-bit integer.
2	primary	The character ‘2’.
⇒ri/rθ	alternate	Changes the complex number at ToS (assumed to be rectangular format (re im) in rectangular mode and polar format (r *θ*) in polar mode) to polar or rectangular format respectively.
5	primary	The character ‘5’.
ri⇔rθ	alternate	Toggles the complex number mode between rectangular and polar (default is rectangular).
8	primary	The character ‘8’.
yPx	alternate	Computes $^{y}P_{x}$, where y is the integer number at ToS - 1 and x is at ToS.
.	primary	The character ‘.’.
solv	alternate	If ToS - 1 is a square matrix (not barred) and ToS is a vector, solves the simultaneous equation for the (matrix, vector). If ToS - 1 is not a matrix or is a barred matrix, and ToS is a

		vector, solves the polynomial equation for the vector.
3 atan2	primary	The character ‘3’.
	alternate	Computes y x atan where *y* is the number at ToS - 1 and *x* is the number at ToS.
6 neg	primary	The character ‘6’.
	alternate	Negates the number at ToS.
9 yCx	primary	The character ‘9’.
	alternate	Computes ${}^{y}C_{x}$, where y is the integer number at ToS - 1 and x is at ToS.
/ //	primary	The division operator.
	alternate	The parallel operator. Computes *xy/(x + y)* where *x* is the number at ToS and *y* is the number at ToS - 1.
+ Σ	primary	The addition operator.
	alternate	Sum of the elements in the vector at ToS. Same as the sum command on the emulator.
– $\Sigma(x^{-1})$	primary	The subtraction operator.
	alternate	Sum of the reciprocal of the elements in the vector at ToS. Same as the rsum command on the emulator.
× Σ(xy)	primary	The multiplication operator.
	alternate	Sum of the product of the elements in the vectors at ToS and Tos - 1. Same as the dot command on the emulator. The number of elements in the shorter vector determine how many products will be computed for the sum.
↵ dup rem	primary	If a user entry is being entered/edited, then pressing this key places the entry at ToS after lifting the stack. No operation takes place if the stack is full. If the user is not editing an entry, then pressing this key duplicates the entry at ToS.
	alternate	The remainder operator.

Examples

Calculate the impedance between the terminals A and B in the diagram below at a frequency of 1kHz.



The full sequence:

`1000 f@ 4.7e3r 1e-3l 10e-6c + 3 + 47e3 // +`

Explanation:

- Enter the frequency

  `1000 f@`

- Compute the impedance for the last loop as (3 -9.6323) = 3 - 9.6323*j* ohm

  `4.7e3r 1e-3l 10e-6c + 3 + 47e3 // +`

- Compute the impedance for the parallel loops as (3.0018 - 9.631) = 3.0018 - 9.631*j* ohm

  `4.7e3r (3 -9.6323) 47e3 // +`

- Compute the impedance in series with the first resistor as (4703.0018 -9.631) = 4703.0018 -9.631*j* ohm

  `4.7e3r (3.0018 -9.6311) +`

Answer is **4703.0018 - 9.631*j*** ohm

More Examples

`500000000000000000000000000000002 ↵`

`500000000000000000000000000000001 -`

This results in:

`0`

However, when the numbers are entered as bigints, enclosed in quotes:

`“500000000000000000000000000000002” ↵`

`“500000000000000000000000000000001” -`

results in:

`1`

Keyboard Layouts

↑$_{STK}$	a	b	c	µ	⏸	√	◀$_{drp}$	1	2	3	+
⇛D/R	^bar	⇛b	sinh	σ	ϕ	∛	cls	conj	⇛ri/rθ	*atan2*	Σ
↓$^{GC}{}_{LC}$	d	e$_{eex}$	f	pow	sin	cos	tan	4	5	6	−
D⇔R	⇛d	cosh	asinh	$\log_x y$	asin	acos	atan	lastx	ri⇔rθ	neg	$\Sigma(x^{-1})$
pag	[ ]	%	”	x	ln	exp	x^2	7	8	9	×
re⇔im	^vec	acosh	tanh	⇛h	log10	10^x	1/x	lasty	yPx	yCx	Σ(xy)
alt	{}	( )	l	@	spc	←x!	→$_{VAR}$	0	.	/	↵$_{dup}$
	^mat	^cmplx	atanh	frq@	swp	*e*	π	rnd	solv	//	rem

Above: Default mode: Page 0

↑$_{STK}$	isP	nxP	join	tran	det	or	◀$_{drp}$	1	2	3	+
⇛D/R	a	b	c	ctrn	rank	xor	cls	1’sC	2’sC	*n*	Σ
↓$^{GC}{}_{LC}$	mod	ror	shr	⇛d	and	exp	binv	4	5	6	−
D⇔R	d	e	f	⇛b	wid	set	clr	lastx	^bar	$\Sigma(x^2)$	$\Sigma(x^{-1})$
pag	[ ]	rol	shl	x	proj	inv	solv	7	8	9	×
λvec	^vec	mont	”	⇛h	ortho	trace	λ	lasty	count0	count1	Σ(xy)
alt	{}	( )	bit	@	spc	←x!	→$_{VAR}$	0	.	/	↵$_{dup}$
	^mat	^cmplx	elem	frq@	swp	*e*	π	rnd	solv	//	rem

Above: Numbers, Vectors and Matrices mode: Page 1

↑	a	b	c	m	n	o	◀	1	2	3	+
	%	”	@	:							
↓	d	e	f	p	q	r	s	4	5	6	−
pag	g	h	i	x	t	u	v	7	8	9	×
	[	]	(								
alt	j	k	l	w	spc	←	→	0	.	/	↵
	{	}	)	y	z						

Above: Keyboard mode: Page α

↑	a	b	c	: gt	end gte	exe	◀$_{\text{drp}}$ cls	1	2	3	+
↓	d	e	f	if	el lt	fi lte	eq neq	4	5	6	−
pag	[ ]	%	”	x	jmp	jnz	jpz	7	8	9	×
alt	{ }	( )	@	y	spc z	← *e*	→ π	0	.	/	↵

Above: Programming mode: Page π (not implemented yet)

↑$_{\text{STK}}$ ⇒D/R	a ^bar	b ⇒b	c sinh	get yyyy	get mm	day	◀$_{\text{drp}}$ cls	1 conj	2 ⇒ri/rθ	3 *atan2*	+ Σ
↓$^{\text{GC}}{}_{\text{LC}}$ D⇔R	d ⇒d	e$_{\text{eex}}$ cosh	f asinh	get latt	get longt	get timez	tan atan	4 lastx	5 ri⇔rθ	6 neg	− Σ(x^{-1})
pag re⇔im	[ ] ^vec	% acosh	” tanh	get mode	get prec	get mod	x^2 1/x	7 lasty	8 yPx	9 yCx	× Σ(xy)
alt	{} ^mat	( ) ^cmplx	l atanh	@ frq@	spc swp	←x! *e*	→$_{\text{VAR}}$ π	0 rnd	. solv	/ //	↵ dup rem

Above: Miscel mode: Page μ