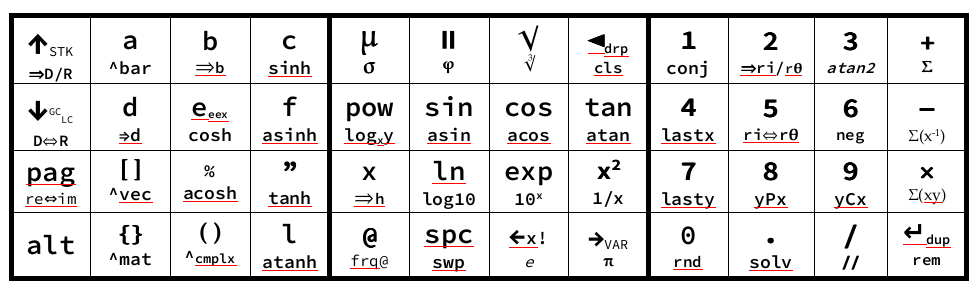
# Keywords, Commands on the PC Emulator

|  |  |  |
| --- | --- | --- |
| **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.” |
| *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 = 57.295779513082320876798 degrees. |
| rad | operator | Converts the ToS to radians. 1 degree = 0.017453292519943295769237 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 *x*th 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 from the values in the vector at ToS. |
| sd | vector operator | The standard deviation computed from the values 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. |
| 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 264 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 | Equals 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 2width - 1 as the modulus. If width is not set, returns an error. |
| 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 264 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 264 as the modulus. In this context, exp is identical to pow.  If the number *x* ToS is a real or complex number, computes *ex.* |
| 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* atToS - 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 |  |
|  |  |  |
| 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 |  |
| yCx | integer operation |  |
| *\* not available on calculator hardware.* | | |

Default mode: Page 0



|  |  |  |
| --- | --- | --- |
| **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. |
| **↓**GCLC  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 startup 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. |
| 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  ^bar | primary | The character ‘a’. |
| alternate | Toggle the barrier attribute on the element at ToS. |
| d  ⇒d | primary | The character ‘d’. |
| alternate |  |
| **[ ]**  ^vec | primary |  |
| alternate |  |
| **{}**  ^mat | primary |  |
| alternate |  |
| b  ⇒b | primary |  |
| alternate |  |
| eeex  cosh | primary |  |
| alternate |  |
| %  acosh | primary |  |
| alternate |  |
| **( )**  ^cmplx | primary |  |
| alternate |  |
| c  sinh | primary |  |
| alternate |  |
| f  asinh |  |  |
|  |  |
| **”**  tanh |  |  |
|  |  |
| l  atanh |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Primary functions:

**↓**GCLC Down: 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).

In stack inspection mode, use as down cursor to point to various stack entries.

Pag: Changes the page mode (0-4, alpha)

Alt: Selects the alternate function

a-f: Characters a-f, used to enter hex numbers or named variables.

[]: Starts or closes a vector.

{}: Starts or closes a matrix.

%: Calculates x% of y, where y is value at ToS-1 and x the value at ToS.

(): Starts or closes a complex number.

”: Starts or closes a string (used for bigints)

l: Character l, can be used as a variable name or used to specify an inductive impedance.

μ:

Alternate functions:

⇒D/R: changes the number at ToS (assumed to be an angle in radians in radian mode and degrees in degree mode) to degrees or radians respectively.

D⇔R: toggles the angle mode between radians and degrees (default is radians).

re⇔im: swaps the real and imaginary part of the number at ToS

⇒HMS: 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.

^bar: 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.

⇒d: converts the ToS to decimal representation. Only scalar elements are allowed.

^vec: 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.

^mat: 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: converts the ToS to binary representation. Only scalar elements are allowed.

cosh: hyperbolic cosine

acosh: inverse hyperbolic cosine

^cmplx: create 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.

sinh: hyperbolic sine

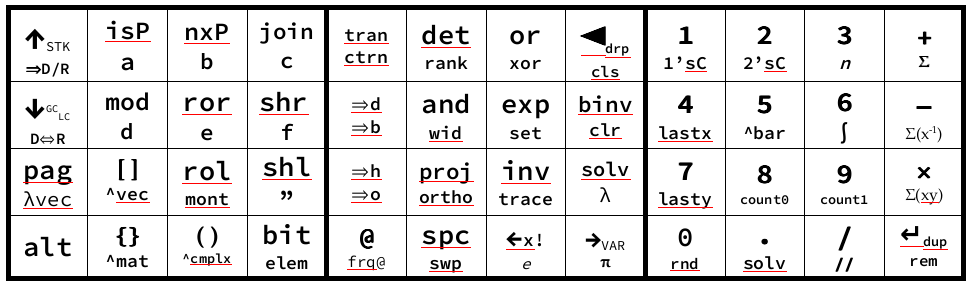
asinh: inverse hyperbolic sine

FIXME

⇒ri/rθ: 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.

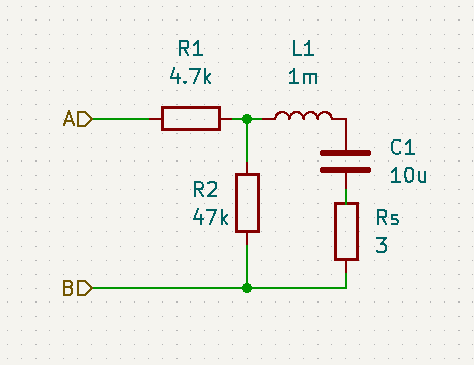
changes the complex number (assumed to be in the rectangular format (re im)) at the ToS to the polar format (r *θ*).

ri⇔rθ: toggles the complex number mode between rectangular and polar (default is rectangular).



# 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

# Keypad Layout in Default Page 0

|  |  |  |
| --- | --- | --- |
| **Command** | **Type** | **Description** |
| ***Up*** | keystroke | When pressed in compute mode, switch to stack inspection mode and then use as up cursor to point to various stack entries. |
| ***Down*** | keystroke | When pressed in compute mode, calculates the GCD of the scalar (real numbers, use integer portions) at ToS and ToS-1. Can be bigint numbers (entered and displayed as strings). |
|  | operator |  |
|  | operator |  |
|  | data modifier |  |
|  | operator |  |
|  | data modifier |  |
|  | operator |  |

# Keypad Layout in Numbers, Vectors and Matrices Page 1

|  |  |  |
| --- | --- | --- |
| **Command** | **Type** | **Description** |
| ***numy numx* bit** | operator | Tests whether bit at position *numx* of the number *numy* is set. Returns 1 if true on the stack, else 0. Consumes *numx* and *numy*. |
| ***numx* wid** | operator | Sets the width of the result in bits. All results show only the lowest *numx* bits and upper bits are truncated. Setting *numx* = 0 disables this. |
| **shl** | operator | Shift left. Bit position [0] gets a ‘0’ after this operation. |
| **shr** | operator | Shift right. Bit position [/wid/ - 1] gets ‘0’ after this operation in case width is set to a non-zero value. |
| **“b** | data modifier | Starts a binary number. The double-quotation mark is automatically closed. |
| **=>b** | operator | Convert to binary bigint. |
| **“x** | data modifier | Starts a hex number. The double-quotation mark is automatically closed. |
| **=>h** | operator | Convert to octal bigint. |

# More Examples

500000000000000000000000000000002 **↵**

500000000000000000000000000000001 **-**

This results in:

0

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

“500000000000000000000000000000002” **↵**

“500000000000000000000000000000001” **-**

results in:

1