ECKERT

Version 2019 User's Manual

Yuishin Kikuchi

June 10, 2020

Contents

Chapter 0

Introduction

0.1 What is ECKERT

ECKERT is a calculator software with command line interface, whose name is short for *Engineering Calculator with KEyboard and Refined Tools*.

```
 Engineering Calculator with KEyboard and Refined Tools
 (C) 2014-2019  Yuishin Kikuchi
------------------------------------------------------------
 HOMURA: (FD) (Rad) (Hex) (Dword)
 Std:  9/15,  Stack: 6,  History: 0/10
============================================================
  #  TYPE       :                                      VALUE
____________________________________________________________
  6: Integer    :                                         12
  5: Floating   :                                        1.5
  4: Complex    :                              3/25 - i4/25
  Z: Matrix     :                          [[2, 3], [3, 4]]
  Y: Rational   :                                    2.1/4
  X: Tuple(Col) :                 (1 + i2, 2 + i3, 3 + i4)
------------------------------------------------------------
 MAKE COLUMN TUPLE
 Ready to operate
------------------------------------------------------------
>
```

Watching the display, type keywords and values to calculate. This software adopts RPN (Reverse Polish Notation), so you do not have to use parenthnesses to determine calculation priorities.

0.2 Audiences

ECKERT is recommended for following users:

Physical or Chemical scientists, electrical scientists, machine engineer, architect, civil engineer, medical scientists, pharmacists, sologists and so on.

0.3 Supporting functions

- SI prefix, binary prefix[*1]
- Rational calculations
- Complex calculations
- Exponent and logarithm
- Trigonometric func
- Hyperbolic func
- Percent calculation
- Include/exclude tax
- Multiply/divide by prefix
- Multiply/divide by 2π
- Decibel conversion
- Base conversion
- Logical calculations
- Vector calculations
- Matrix calculations
- Register functions[*2]
- Unit conversions[*3]
- Math/sci constants[*3]

0.4 Operating Environments

Windows 7, Windows 8, Windows 8.1 and the latter versions.

0.5 Disclaimer

If you find bugs or unnatural specifications, please send messages to me.

Contact: mailto:only.my.truth@gmail.com

I NEED YOUR HELP!

This user's manual was translated from Japanese version. If you find the English in the document something wrong, please send reports to me, thanks.
これは日本語からの翻訳です。不自然な英語表現にお気づきの際はご連絡ください。

[*1] Numeric formats such as '12k' (12 kilos) or '32u' (32 micros) and so on.
[*2] You can store data from stack to register, also can load/delete from register.
[*3] 2014 CODATA

Chapter 1

Preparation

1.1 Installation and Uninstallation

You can find `eckert86.exe` and `eckert64.exe` in the package. The both are executable files. The file `eckert86.exe` is for 32-bit Windows system and the file `eckert64.exe` is for 64-bit Windows system. Please check your system.

Each exe file is independent so you can delete unused one. This software does not change registories in your system. Thus, this is portable.

The installation of thie software is just copying.

The uninstallation is just delete. You can also delete the config file.

1.2 How to read this manual

This manual explains whole functions of ECKERT and it is just user's manual so the fundmental mathematical definitions are omitted.

If you do not know RPN calculator, please read chapter 2 and 4 first. If you get used to the operations, read chapter 5, 6, 7 and 11.

If you know about RPN calculator, you can read chapter 4 diagonally to make comprehension of the operations of this software.

To configurate display digits or value format, please read chapter 3.

1.3 Format of this document

This manual uses following format:

Important thngs

Things to notice

Command input

Command is shown like Cmd `INPUT`. Just type and press enter to operate.

This software adopts stack concept, which is one of data storage structures. Please read chapter 3 to get more information about stack. This manual uses tables like following to describe a state of a stack.

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	12
X:	Floating	:	1.5

The column `TYPE` means data type and the column `VALUE` means data value.
This document uses list in following format to show functions.

Function	Keyword	R	D	Math
Add	`ADD, +`	2	2	$Y + X$
Subtract	`SUB, -`	2	2	$Y - X$

The column Function means function name and the column Keyword means command to call corresponding function.

Please refer chapter 4 to get more information about reading list like above one.

Chapter 2

Display and Operations

2.1 Launch and Terminate

Just double click the executable file to launch.

Type Cmd. EXIT, Cmd. QUIT, or Cmd. Q and press enter to terminate the program. Inputs are non-capital-sensitive except for numerical value input.

Function	Keyword
Terminate	EXIT, QUIT, Q

Special start up is available. Please refer chapter 15.

2.2 Display of calculation mode

The following chart is the display of calculation mode:

```
 Engineering Calculator with KEyboard and Refined Tools
 (C) 2014-2019  Yuishin Kikuchi
------------------------------------------------------------
 HOMURA: (FD) (Rad) (Hex) (Dword)
 Std:  9/15,  Stack: 6,  History: 0/10
============================================================
  #  TYPE       :                                      VALUE
____________________________________________________________
  6: Integer    :                                         12
  5: Floating   :                                        1.5
  4: Complex    :                              3/25 - i4/25
  Z: Matrix     :                         [[2, 3], [3, 4]]
  Y: Rational   :                                    2.1/4
  X: Tuple(Col) :                 (1 + i2, 2 + i3, 3 + i4)
------------------------------------------------------------
 MAKE COLUMN TUPLE
 Ready to operate
------------------------------------------------------------
>
```

The first two lines mean name of this software and the copyright.

Following a split line, calculation config and states display.

```
HOMURA: (FD) (Rad) (Hex) (Dword) [i.a/b]
Std:  9/15,  Stack: 6,  History: 0/10
```

Below a doubly split line, you can find stack display there.

```
 #  TYPE       :                                        VALUE
═══════════════════════════════════════════════════════════════
 6: Integer    :                                           12
 5: Floating   :                                          1.5
 4: Complex    :                                  3/25 - i4/25
 Z: Matrix     :                               [[2, 3], [3, 4]]
 Y: Rational   :                                       2.1/4
 X: Tuple(Col) :                     (1 + i2, 2 + i3, 3 + i4)
```

The right column is data number, the center is data type, and the left is value.

Below the stack display is 2-line message are.

```
------------------------------------------------------------
 MAKE COLUMN TUPLE
 Ready to operate
------------------------------------------------------------
```

The bottom of the display is input field.

```
>
```

Go on to the next section to make comprehension of reading the display.

2.3 Calculation mode and states display

You can find 2-line calculation mode and states display.

```
HOMURA: (FD) (Rad) (Hex) (Dword) [i.a/b]
Std:  9/15,  Stack: 6,  History: 0/10
```

In the first line, you can notice symbols in the following table:

2.3.1 Decimal display

Symbol	State
(AD)	Auto Decimal display
(FD)	Force Decimal display
(FF)	Force Fractional display

2.3.2 Angle mode

Symbol	State
(Deg)	Degree mode
(Rad)	Radian mode
(Gra)	Grade mode

2.3.3 Unsigned integer display

Symbol	State
(Bin)	Binary display
(Oct)	Octal display
(Sdec)	Signed decimal display
(Udec)	Unsigned decimal display
(Hex)	Hexadecimal display

2.3.4 Logical calculation

Symbol	State
(Byte)	8-bit mode
(Word)	16-bit mode
(Dword)	32-bit mode
(Qword)	64-bit mode

2.3.5 General

Symbol	State
`[Reg]`	Register display
`[Eul]`	Euler display
`[Eul(Pi)]`	Euler display (π radians)
`[i.a/b]`	Mixed fractional display

`(Symbol)` selected in each class is always displayed.

`[Symbol]` is displayed if the mode is enabled.

```
HOMURA: (FD) (Rad) (Hex) (Dword) [i.a/b]
```

You can see display above and you get force fractional display, radian, hexadecimal display, 32-bit and mixed fractional display mode.

There are three sections in the second line. The first consists of decimal display mode and display digits.

Symbol	State
`Std`	Standard display
`Fix`	Fixed display
`Sci`	Scientific display
`Eng`	Engineering display

The fraction `Int/Int` in the first section means this: the first means the current display digits and the second is the number of max digits you can set in the selected display mode. To change the number of digits, please read chapter 3.

The second is the number of elements in the stack. If the number is zero, Empty is displayed.

The third is history display.

Display	State
OFF	History is disabled
Init	Initial state
Int/Int	(Described later)

The fraction `Int/int` in the second section means this: the first integer is the times that you have called undo and the second integer is the items in the history.

```
Std:  6/15,  Stack: 6,  History: 0/10
```

You get that the decimal display mode is standard display mode and the current number of selected (standard) display digits is 6 and the maximum number of digits you can set is

15. There are 11 elements in the stack. The number of history items is 10 and the pointer for older history is 4.

2.4 Stack display

Learn the concept of stack.

```
#  TYPE       :                                    VALUE
-------------------------------------------------------
6: Integer    :                                       12
5: Floating   :                                      1.5
4: Complex    :                              3/25 - i4/25
Z: Matrix     :                          [[2, 3], [3, 4]]
Y: Rational   :                                   2.1/4
X: Tuple(Col) :                 (1 + i2, 2 + i3, 3 + i4)
```

Stack is one of the data containers. This software has one stack.

In each line in the stack display contains item number, data type and value. A data type means a kind of a number. If a data type is integer, `Integer` is displayed in the `TYPE` column and if the type is rational number, `Rational` is displayed.

This manual shows the stack like below:

```
#   TYPE       :                                 VALUE
-------------------------------------------------------
4:             :
Z:             :
Y:  Integer    :                                    12
X:  Floating   :                                   1.5
-------------------------------------------------------
```

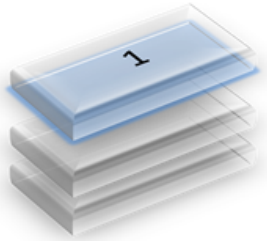
> The stack size is unlimited.

X is the bottom of the stack. Y is the second bottom and Z is the third bottom. After that, the data numbers are displayed as integers such as 4, 5... The data in X is called just X, the data in Y is just Y, and so on.

Go on to the next page and make comprehension of stack graphically.

You can see a stack like a pile of cards. You draw one by one from the top of the pile and you put into the pile one by one.

```
  #    TYPE        :                          VALUE
  Z:               :
  Y:   Integer     :                              1
  X:   Integer     :                              2
```



Please look at the chart above. There are some cards. You put a card '1' and card '2' in turn.

This situation is expressed like below:

```
  #    TYPE        :                          VALUE
  Z:               :
  Y:               :
  X:   Integer     :                              1
```

See addition with stack.

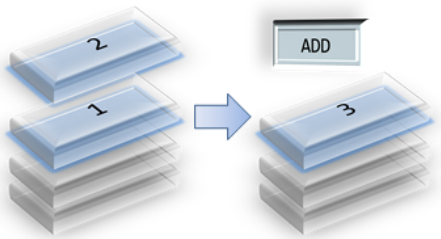
You draw 2 cards from the top and you put the value of $1 + 2$ on the top.

This is the fundamental flow of calculation with stack.

```
  #    TYPE        :                          VALUE
  Z:               :
  Y:   Integer     :                              1
  X:   Integer     :                              2
> +
```

Operate addition

```
  #    TYPE        :                          VALUE
  Z:               :
  Y:               :
  X:   Integer     :                              3
```



There are 3 fundamental operations: add (push), remove (drop) and execution.

ECKERT handles these data types:

Symbol	State
`Error`	String value means error
`String`	String value
`Integer`	Integer
`Floating`	Floating point number
`Rational`	Rational number
`Infinity`	Infinity
`Complex`	Complex number
`Boolean`	Boolean (true of false)
`Byte`	8-bit unsigned integer
`Word`	16-bit unsigned integer
`Dword`	32-bit unsigned integer
`Qword`	64-bit unsigned integer
`Tuple[Row]`	Row vector
`Tuple(Col)`	Column vector
`Matrix`	Matrix

2.5 Message display

In the message display, the last called function and error / notice / confirm message are displayed.

```
------------------------------------------------------------
 MAKE COLUMN TUPLE
 Ready to operate
------------------------------------------------------------
```

In the first line is called function and the second line is the other messages.

If unoperatable commands such as division by zero is input, the operation is stopped and an error message is displayed in the second line.

```
 [!] ADD Y+X
 Error: Too few arguments
```

If there is error or notice message, the message is displayed second line with a symbol in the first line.

Symbol	State
[!]	Operation is terminated by error
[i]	Unordinal operation
[C]	Waiting input or confirm

For more information, please read chapter 16.

If [?] is displayed, it means that there are software bugs. Please send me a bug report.

2.6 Configuration mode display

Type Cmd CONFIG to go to configuration display.

```
 Engineering Calculator with KEyboard and Refined Tools
 (C) 2014-2019  Yuishin Kikuchi
------------------------------------------------------------
 CONFIGURATION MODE
============================================================
 Interface
   History size   (hist): 10
   Display width (width): 79
   Display lines (lines): 11
 Management
   Load config   (load)
   Save config   (save)
   Reset config  (reset)
------------------------------------------------------------
 ECKERT Config
 Type "calc" or "homura" to quit config
------------------------------------------------------------
>
```

```
 Interface
   History size   (hist): 10
   Display width (width): 79
   Display lines (lines): 11
```

Maximum history size, display width and the number of stack display lines are shown. Please read chapter 3 to configurate these.

```
 Management
   Load config   (load)
   Save config   (save)
   Reset config  (reset)
```

Those are command for config management.

2.7 Fundamental operation

Input keywords or values to operate. Only half-width (one byte) characters are supported.

Type one or several space-splitted keywords or values and hit enter to calculate or configurate. If the number of tokens, which are keywords or numerical values, is not single, each token is processed in turn.

This way, “type and enter” is the flow of the operations. Please notice that the display changes only pressing enter. Then, only SI or binary prefixes are case-sensitive, the others are not.

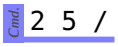
This software supports only printable characters input.

For instance, type like below to operate add and multiply in turn.

Some keywords are aliases, in other words, some ones are connected with the same function. And more, there are some keywords depend on calculation modes.

Type numerical values to input. You can put space-splitted values in order.

You can even mix values and keywords.



Go on to the next section to get how to input numeric values.

2.8 Input numeric values

This section shows how to input numeric values in this software.

2.8.1 Integer and decimal fraction

Just type an integer value to input an integer.

Just type a value with decimal point to decimal fraction. You can omit integer part (like Cmd. `.2`) or decimal part (like Cmd. `1.`).

2.8.2 Exponential

Type a decimal value and append `E` or `e` and a decimal exponent.

For instance, 6.02×10^{-23} is expressed like `6.02E-23` and 1.01325×10^{5} is expressed like `1.01325E5`.

2.8.3 Complex number

Positive imaginary unit is Cmd. `i` or Cmd. `+i` and negative imaginary unit is Cmd. `-i`.

You can pure imaginary number with prefixing `i` to integer, decimal fraction, or exponential notation.

Non-case-sensitive.

2.8.4 Infinity

Positive infinity is Cmd. `INF`, Cmd. `+INF` or Cmd. `+INFINITY`,

Negative infinity is Cmd. `-INF` or Cmd. `-INFINITY`.

2.8.5 Boolean

True value is Cmd. `TRUE` or Cmd. `T` and false value is Cmd. `FALSE` or Cmd. `F`.

2.8.6 Unsigned decimal value

Prefix `u` and type integer without sign and spaces.

2.8.7 N-base value

Binary: Prefix `0b`.

Octal: Prefix `0o`.

Hex: Prefix `0x`.

2.9 Value with SI or binary prefix

You can append SI or binary prefix to integer, decimal, exponential and imaginary value. SI and binary prefixes are case-sensitive.

Symbol	Name	Value	Symbol	Name	Value
da	DECA	$\times 10^{+01}$	d	DECI	$\times 10^{-01}$
h	HECTO	$\times 10^{+02}$	c	CENTI	$\times 10^{-02}$
K, k	KILO	$\times 10^{+03}$	m	MILLI	$\times 10^{-03}$
M	MEGA	$\times 10^{+06}$	u	MICRO	$\times 10^{-06}$
G	GIGA	$\times 10^{+09}$	n	NANO	$\times 10^{-09}$
T	TERA	$\times 10^{+12}$	p	PICO	$\times 10^{-12}$
P	PETA	$\times 10^{+15}$	f	FEMTO	$\times 10^{-15}$
E	EXA	$\times 10^{+18}$	a	ATTO	$\times 10^{-18}$
Z	ZETTA	$\times 10^{+21}$	z	ZEPTO	$\times 10^{-21}$
Y	YOTTA	$\times 10^{+24}$	y	YOCTO	$\times 10^{-24}$
Ki, ki	KIBI	$\times 1024^{1}$			
Mi, mi	MEBI	$\times 1024^{2}$			
Gi, gi	GIBI	$\times 1024^{3}$			
Ti, ti	TEBI	$\times 1024^{4}$			
Pi, pi	PEBI	$\times 1024^{5}$			
Ei, ei	EXBI	$\times 1024^{6}$			
Zi, zi	ZEBI	$\times 1024^{7}$			
Yi, yi	YOBI	$\times 1024^{8}$			

You can use binary prefixes alias.

2.10 Examples of value input

Integer

Cmd. `-3`

Prefixed

Cmd. `3k`

Scientific

Cmd. `2.998e8`

Imaginary unit

Cmd. `-i`

Imaginary num

Cmd. `i12`

Imag with sign

Cmd. `-i5`

Infinity

Cmd. `-inf`

Boolean

Cmd. `t`

Unsigned

Cmd. `u65536`

Binary

Cmd. `0b1010`

Octal

Cmd. `0o100`

Hexadecimal

Cmd. `0xFFFE`

You can also input math or scientific constants with keywords. Please read chapter 14 to get more information.

Name	Keyword	Value
PI	`PI`	`3.141 592 653 589 79`
Napier's constant	`E`	`2.718 281 828 459 05`
Euler-Mascheroni constant	`EG`	`0.577 215 664 901 533`

In addition to these, you can input string value. Use double quotation to input string value.

String Cmd. `"This is test"`

You can use string to put memos in the register or use macro function.

2.11 When error messages displayed

When the error occurs while operating some functions, the operating and the left unoperated functions are cancelled. This means, the state is the before one cancelled operation. And then, the error messages are shown.

If you see error messages, you can operate as usual. Input commands and if the operations are successful, error messages are disappeared.

Even if operating space-splitted tokens, the functions called one by one, so this software do not stop the operations if no errors.

```
Cmd: 5 0 /
```

(You can make sense of the notation if you read chapter 4.)

For instance, if you input like above, the error "division by zero" occurs. But the push operations are done, so the value 5 and the value 0 is added into the stack and the division cancelled with the stack keeped.

If the error messages are shown, undo and redo are recommended. Please read chapter 15 to get more information.

If you look at the list of error messages, please read chapter 16.

Function	Keyword
Undo	`UNDO, U`
Redo	`REDO, R`

Please read Chapter 15 to get more information.

The error message list is in Chapter 16.

When unsupported inputs are detected, the error message below is displayed:

```
[!] OPERATIONAL ERROR
Error: Unsupported operation or notation
```

If you see this, please check the spelling.

Even if the keyword is supported, you can see this when the calculation mode or state is not inappropriate, or greater than one settings-changing keywords.

Chapter 3

Configuration Mode

3.1 Settings in configuration mode

You can set max history size, display width and the number of lines of stack display in configuration mode.

Please type the keyword Cmd. `CONFIG` to go to config mode. Input keyword Cmd. `HOMURA` or Cmd. `CALC` to return to calculation mode.

3.1.1 Max history size

Type Cmd. `HIST` and an integer. You can input splited-tokens like Cmd. `hist 10`. You can set the size to 0 to disable history function. The default max history size is 10.

Type below to set the max history size to 20.

Cmd. `hist 20`

3.1.2 Display width

Type Cmd. `WIDTH` and an integer. You can input splited-tokens like Cmd. `width 79`. If the value is less than the least width, the least width is set.

The default display width is 79 and the least size is 60.

Type below to set the display width to 69.

Cmd. `width 69`

3.1.3 Number of stack lines

Type Cmd. `LINES` and an integer without +/−. You can input splited-tokens like Cmd. `lines 11`. If the value is less than the least number, the least number is set.

The default number of stack lines is 11 and the least is 4.

Type below to set the number of stack lines to 20.

Cmd. `lines 20`

3.1.4 Management of config

You can save configurations as a `config` file. You can use the following functions to manage `config` file.

Function	Keyword
Load config file	`LOAD`
Save config file	`SAVE`
Reset config	`RESET`, `RST`

If the config file exists, this software loads it on startup. So the max history size and display width are restored automatically.

You can load config file explicitly with Cmd. `LOAD` function.

The function Cmd. `RESET` sets all settings in config mode to default. However, this function does not save or change a config file.

3.1.5 Functions in configuration mode

Here is the list of keywords for configuration mode:

Function	Keyword
Config mode	`CONFIG`
Calculation mode	`CALC`, `HOMURA`
History size	`HIST`
Display width	`WIDTH`
Number of stack lines	`LINES`
Load config file	`LOAD`
Save config file	`SAVE`
Reset config	`RESET`, `RST`

3.2 Settings in calculation mode

Angle mode, type display and number of display digits can be changed in calculation mode.

3.2.1 Rational display mode

When the decimal display is set to standard, you can choose rational number display following:

Auto decimal display
: If a rational number can be displayed as finite decimal display, show a decimal fraction. In other cases, show a fraction.

Force decimal display
: All rational numbers are displayed as decimal fraction.

Force fractional display
: All rational numbers are displayed as standard fraction.

To choose mode, use the followingkeywords:

Mode	Keyword	Display
Auto decimal display	`AD`	`(AD)`
Force decimal display	`FD`	`(FD)`
Force fractional display	`FF`	`(FF)`

The default rational display mode is Force decimal.

3.2.2 Angle mode

You can choose angle unit with setting angle mode. Angle mode affects trigonometric functions.

To choose mode, use the keywords below:

Mode	Keyword	Display
Degree	`DEG`	`(Deg)`
Radian	`RAD`	`(Rad)`
Grade	`GRAD, GRA`	`(Gra)`

This mode is connected with Cmd. `SIN`, Cmd. `ARG` and so on. The default angle mode is Radian.

3.2.3 N-ary number display mode

You can select the display of 8-bit ,16-bit, 32-bit and 64-bit data.

To choose mode, use the following keywords:

Mode	Keyword	Display
Binary display	BIN	(Bin)
Octal display	OCT	(Oct)
Signed decimal display	SDEC	(Sdec)
Unsigned decimal display	UDEC	(Udec)
Hexadecimal display	HEX	(Hex)

The default N-ary number display mode is Hexadecimal.

3.2.4 N-bit input mode

You can choose the binary size to input from 8, 16, 32 or 64 bits. If unsigned decimal with 'u' is detected, the value is generated as selected bit mode.

To choose mode, use the keywords below:

Mode	Keyword	Display
8-bit mode	BYTE	(Byte)
16-bit mode	WORD	(Word)
32-bit mode	DWORD	(Dword)
64-bit mode	QWORD	(Qword)

The default size is 32-bit.

3.2.5 Type display

You can switch the type display in the stack display on/off. Input "TYPE" without any other keywords to switch.

The default type display is enabled.

3.2.6 Register display

You can enable or disable the register display. Use the keywords Cmd. REG or Cmd. REGISTER to switch the display. Register is displayed above the stack. If the register display is enabled, the stack display gets smaller.

When register display is enabled, the symbol [Reg] is displayed.

The default setting is disabled.

3.2.7 Euler display

You can switch the complex number display: $a + ib$ (rectangular) or $r\exp(i\theta)$ (polar) style. Use the keyword Cmd. EULER or Cmd. EUL to switch.

If the Euler display is enabled, the symbol `[Eul]` is displayed.

The argument of Euler display is depends on angle mode.

Mode	Math	Display
Rectangular	$5+12i$	`5 + i12`
Polar (deg)	$13\angle 67$[deg]	`13 exp(+i67.d)`
Polar (rad)	$13\angle 1.3$[rad]	`13 exp(+i1.3)`
		`13 exp(+i0.41 Pi)`
Polar (gra)	$13\angle 75$[gra]	`13 exp(+i75.g)`

If you select radian, you can convert the argument to π radians. To switch the display, type Cmd `PIRAD` or Cmd `PRAD`. When π radian mode is enabled and Euler display is also enabled, then the symbol `[Eul(Pi)]` is displayed.

The default setting is disabled.

3.2.8 Mixed fractional display

You can get mixed fractional display. Use Cmd `FRACTION` or Cmd `FRAC` to enable/disable mixed fractional display.

The display of rational number is below:

Value	Provisional	Mixed	Decimal
+3/2	`3/2`	`1.1/2`	`1.5`
−6/5	`-6/5`	`-1.1/5`	`-1.2`

If the mode is enabled, the symbol `[i.a/b]` is displayed.

The default setting is disabled.

3.2.9 Decimal display

You can choose decimal display mode. There are four modes: standard, fixed, exponential and engineering.

Standard display
: Value display changes flexibly. Rational number display depends on the rational display mode.

Fixed display
: Fix the digits of decimal part. Integers and rational numbers are displayed as decimal.

Scientific display
: All scalars are displayed as scientific notation such as `1.2E+10`. The range of mantissa m is $0 \leq m < 1000$. Integers and rational numbers are displayed as decimal.

Engineering display

All scalars are displayed as scientific notation such as `12E+10`. The range of mantissa m is $0 \leq m < 1000$. Integer and rational number is displayed as decimal.

To choose display mode, use the following keywords:

Mode	Keyword	Display
Standard display	`STD`	`Std`
Fixed display	`FIX`	`Fix`
Scientific display	`SCI`	`Sci`
Engineering display	`ENG`	`Eng`

Rational number is displayed as decimal without in standard mode.
The default display mode is standard.

3.2.10 Decimal digits

You can change the digits of decimal. Here is the list of "digit" meaning:

Mode	Meaning of "digits"
Standard	Significant digits
Fixed	Digits in decimal part
Scientific	Significant digits
Engineering	Significant digits

Use the keyword Cmd. `DISP` or Cmd. `DIGIT` and input an integer to set the number of digits.
If you would set to 3 digits, type below:

Cmd. `digit 3`

You can set digits in each mode.

The maximum number of digits exists in each mode. Too large number is read as max and too small number does as minimum.

Mode	Minimum	Maximum
Standard	1	15
Fixed	0	15
Scientific	1	15
Engineering	1	15

Example: 10 times of π(= 31.4159265358979)

```
Std:  5/15  31.416
Fix:  5/15  31.41593
Sci:  5/15  3.1416E+01
Eng:  5/15  31.416E+00
```

The default numbers of digits are all 9.

If you put any other tokens after digit settings like `disp 10 36`, these are ignored.

3.3 Keywords of settings in calculation mode

Here is the list of keywords of settings in calculation mode:

Mode	Keyword	Display
Auto decimal display	AD	(AD)
Force decimal display	FD	(FD)
Force fractional display	FF	(FF)
Degree mode	DEG	(Deg)
Radian mode	RAD	(Rad)
Grade mode	GRA,GRAD	(Gra)
Binary display	BIN	(Bin)
Octal display	OCT	(Oct)
Signed decimal display	SDEC	(Sdec)
Unsigned decimal display	UDEC	(Udec)
Hexadecimal display	HEX	(Hex)
8-bit mode	BYTE	(Byte)
16-bit mode	WORD	(Word)
32-bit mode	DWORD	(Dword)
64-bit mode	QWORD	(Qword)
Type display	TYPE	
Register display	REG	[Reg]
Euler display	EULER, EUL	[Eul]
π radian argument display	PIRAD, PRAD	[Eul(Pi)]
Mixed fraction display	FRACTION, FRAC	[i.a/b]
Standard decimal display	STD	Std
Fixed decimal display	FIX	Fix
Scientific decimal display	SCI	Sci
Engineering decimal display	ENG	Eng
Set number of digits	DISP, DIGIT	

3.4 Next/previous pages in stack

If there are many elements in the stack, you cannot see the all data.

```
 Std:  9/15,  Stack: 11,  History: 0/10
============================================================
  #  TYPE       :                                    VALUE
  ^  ......                                      ......  ^
  6: Integer    :                                       6
  5: Integer    :                                       7
  4: Integer    :                                       8
  Z: Integer    :                                       9
  Y: Integer    :                                      10
  X: Integer    :                                      11
------------------------------------------------------------
 PUSH Integer
 Ready to operate
------------------------------------------------------------
```

If you need to see unshown data, use stack page function. There are 11 data in stack but only 6 is shown in the chart above.

Use the keyword Cmd. NEXT or Cmd. N to turn to the next page.

```
 Std:  9/15,  Stack: 11,  History: 0/10
============================================================
  #  TYPE       :                                    VALUE
------------------------------------------------------------
  -:            :
 11: Integer    :                                       1
 10: Integer    :                                       2
  9: Integer    :                                       3
  8: Integer    :                                       4
  7: Integer    :                                       5
  v  ......                                      ......  v
 NEXT PAGE of STACK
 Ready to operate
------------------------------------------------------------
```

Use the keyword Cmd. PREV or Cmd. P to turn to the previous page. If you would like to return to first page, use the keyword Cmd. FIRST or Cmd. FST.

If a stack-changing function is called, the page is set to first.

Function	Keyword
Next page of stack	NEXT, N
Previous page of stack	PREV, P
First page of stack	FIRST, FST

3.5 Next/previous pages in register

This software has registers which is used for saving location of data. There are 26 registers in this software: RA to RZ. You can not view all registers at once without changing the number of stack lines.

Look at the following chart. RA to RC are displayed but the others are not.

```
 Std:  9/15,  Stack: 3,  History: 0/10
============================================================
  #  TYPE       :                                    VALUE

------------------------------------------------------------
 RA: Floating   :                              3.14159265
 RB:            :
 RC:            :

------------------------------------------------------------
  Z: Integer    :                                       4
  Y: Integer    :                                       5
  X: Integer    :                                       6
------------------------------------------------------------
 NEXT PAGE of STACK
 Ready to operate
------------------------------------------------------------
```

You can change the register page. Type Cmd. REGNEXT or Cmd. RN to change to next page of registers.

```
 Std:  9/15,  Stack: 3,  History: 0/10
============================================================
  #  TYPE       :                                    VALUE

------------------------------------------------------------
 RD:            :
 RE:            :
 RF:            :

------------------------------------------------------------
  Z: Integer    :                                       4
  Y: Integer    :                                       5
  X: Integer    :                                       6
------------------------------------------------------------
 NEXT PAGE of REGISTERS
 Ready to operate
------------------------------------------------------------
```

On the other hand, type Cmd. REGPREV or Cmd. RP to change to previous page of registers. The keyword Cmd. REGFIRST or Cmd. RF is for returning to first page of the registers. Here is the list of register page functions:

Function	Keyword
Next page of registers	REGNEXT, RN
Previous page of registers	REGPREV, RP
	to next page

from prev page

Function	Keyword
First page of registers	REGFIRST, RF

3.6 View full string of data

In case of the value display is too long, only the left part is displayed. The following chart is the stack which has a complex number consists of 2 rationals but the right part is omited.

```
  #  TYPE       :                                          VALUE
 ____________________________________________________________
  -:            :
  -:            :
  -:            :
  Z:            :
  Y:            :
  X: Complex    : 2432902008176640000/243290200817664000...
```

To view full data, use the keywords Cmd VIEW or Cmd V.

Function	Keyword
View full data	VIEW, V

```
------------------------------------------------------------
------------
 STACK VIEW
============
X: Complex:
2432902008176640000/2432902008176640001 + i24329020081766400
00/2432902008176640001

------------------------------------------------------------
(Press Return or Enter)
>
```

View mode shows data, which are displayed in calculation mode. Press Enter to return to calculation mode.

3.7 Version display

Type the keyword Cmd VER or Cmd VERSION to display current version.

Function	Keyword
Version display	`VERSION`, `VER`

If you find bugs in this app, please send reports to me with the version.

Chapter 4

Fundamental operations – four arithmetics

This chapter includes the most important things about operating this software, like RPN. So please read carefully.

4.1 Elementary stack operations

Let's input an integer.

Cmd: `12`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	12

12 is added into X in the stack display area.

Next, type one more integer.

Cmd: `9`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	12
X:	Integer	:	9

The data 9 is added into X.

This way, addition is executed into X.

The next, input decimals. You can put tokens with spaces.

Cmd: `1.6 6.0e-23`

#	TYPE	:	VALUE
4:	Integer	:	12
Z:	Integer	:	9
Y:	Floating	:	1.6
X:	Floating	:	6E-23

This way, just write numbers to add into the stack. The addition into the bottom of the stack is called push.

Type Cmd: `DROP` or Cmd: `\` to remove the data at the bottom of the stack. The removal of the bottom of the bottom of the stack is called drop.

Cmd: `\`

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	12
Y:	Integer	:	9
X:	Floating	:	1.6

Just hit enter without any input to duplicate the bottom of the stack (X) and push.

The keyword Cmd: `DUP` call the same function.

Cmd: `(Just hit enter)`

#	TYPE	:	VALUE
4:	Integer	:	12
Z:	Integer	:	9
Y:	Floating	:	1.6
X:	Floating	:	1.6

Type Cmd: `CLEAR` or Cmd: `CLR` to empty the stack.

Cmd: `clear`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:		:	

Here is the list of keywords described in this section:

Function	Keyword	R	D
Push		0	0
Drop	DROP, \	1	1
Duplicate [1]	DUP	1	1

to next page

from prev page

Function	Keyword	R	D
Clear stack	`CLEAR`, `CLR`	N>0	N

[1] You can call the function just hitting enter without any input.

4.2 Four arithmetic operations

The four arithmetics are the basics of calculating with this software.

Use following keywords to calculate the four arithmetics:

Function	Keyword	R	D	Math
Add	`ADD`, `+`	2	2	$Y + X$
Subtract	`SUB`, `-`	2	2	$Y - X$
Multiply	`MUL`, `*`	2	2	$Y \times X$
Divide	`DIV`, `/`	2	2	$Y \div X$
Modulo	`MOD`, `%`	2	2	$Y \mod X$
Quotient and remainder	`QM`	2	2	$Y \leftarrow Y \div X$ $X \leftarrow Y \mod X$

Please try following the tutlrial.

The first step is a simple addition. Challenge $2 + 3$. Push two numbers as following:

Cmd: `2 3`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	2
X:	Integer	:	3

Cmd: `+`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	5

You can see X is 5, which is the the result of $Y + X (= 2 + 3)$. The previous Y and X are removed. Your inputs mean the pushing 2 and 3 before adding.

Following this, try this:

Cmd: `9 -`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	-4

You get X is −4. You have pushed 9 and called subtraction. You can see this software calculates with using the bottom of the stack.

Function	Keyword	R	D	Math
Add	ADD, +	2	2	$Y + X$

This manual uses tables like above one. The column R is the number of required data. If you call the function without the stack containing enough data, error messages are displayed. The column D is the number of dropped data.

Addition requires two data. Once the function is called, two data are dropped and the result of Y+X is pushed. The other arithmetics are similar with addition.

In the case of not-enough data, you see error messages like following:

```
 Engineering Calculator with KEyboard and Refined Tools
 (C) 2014-2019  Yuishin Kikuchi
------------------------------------------------------------
 HOMURA: (FD) (Rad) (Hex) (Dword)
 Std:  9/15,  Stack: 1,  History: 0/8
============================================================
  #  TYPE       :                                    VALUE
____________________________________________________________
  -:            :
  -:            :
  -:            :
  Z:            :
  Y:            :
  X: Integer    :                                        5
------------------------------------------------------------
 [!] ADD Y+X
 Error: Too few arguments
------------------------------------------------------------
>
```

4.3 Multiple arithmetics

Try higher level operations.

Calculate the area of the trapezoid: the upper base is 2, the lower is 1, the hight is 5. The formula of of calculating this is:

$$5 \times (2 + 1) \div 2$$

You can read this like the multiplication of 5 and $(2 + 1)$. First, push 5 and the result of

$2 + 1$, and call multiply. The final step is halfing.

Type as following to calculate at one time.

```
Cmd: 5 2 1 + * 2 /
```

However, this expression is difficult for the beginners. I divided this into the steps: (1) – (5). Read carefully and operate to understand easily.

(1) Push 5, 2 and 1

```
Cmd: 5 2 1
```

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	5
Y:	Integer	:	2
X:	Integer	:	1

(2) Add

```
Cmd: +
```

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	5
X:	Integer	:	3

(3) Multiply

```
Cmd: *
```

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	15

(4) Push 2

```
Cmd: 2
```

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	15
X:	Integer	:	2

(5) Divide

```
Cmd: /
```

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Rational	:	7.5

You can calculate with pushing and calling functions in appropriate order without parentheses.

4.4 Elementary functions

Here is the list of elementary functions without the four arithmetics:

Function	Keyword	R	D	Math
Increment	`INC, ++`	1	1	$X+1$
Decrement	`DEC, --`	1	1	$X-1$
Absolute value	`ABS`	1	1	$\|X\|$
Negate	`PM, NEG`	1	1	$-X$
Invert (incl. matrix)	`INV`	1	1	X^{-1}

You can increment or decrement only integers. Increment is adding 1 and decrement is adding -1.

For example, type this to find the inverse of 5:

```
Cmd: 5 inv
```

These functions require one argument.

4.5 Concept of RPN

You have to put opeartion 'latter' with calculating with stack. The notation of expression is called Reverse Polish Notation (RPN).

```
Cmd: 2 1 -
```

RPN don't require any brackets. If you get used to this idea, you can put sequently calculations like following.

```
Cmd: 5 6 * pm
```

Chapter 5

Mathematical Functions

5.1 How to use math functions

This software supports many math functions. Please notice that the usages of these functions are similar with the usage of the ones of four arithmetics. Push first and call functions.

Some functions have restricted domains.

5.2 Exponent and logarithm

Use the following keywords with operating exponents and logarithms.

Function	Keyword	R	D	Math
Square	`SQ`	1	1	X^2
Square root	`SQRT`	1	1	$\sqrt{X}$
Cube root	`CBRT`	1	1	$\sqrt[3]{X}$
Hypotenuse	`HYPOT`	2	2	$\sqrt{\lvert Y\rvert^2 + \lvert X\rvert^2}$
Power	`POW, ^, **`	2	2	Y^X
N-th root	`NRT`	2	2	$\sqrt[X]{Y}$
Natural exponent	`EXP`	1	1	$\exp(X)$
Power of 10	`TPOW`	1	1	10^X
Power of 2	`BPOW`	1	1	2^X
Logarithm of X to base Y	`LOGB`	2	2	$\log_Y(X)$
Natural logarithm	`LN`	1	1	$\log_e(X)$
Common logarithm	`LOG`	1	1	$\log_1 0(X)$
Binary logarithm	`LB`	1	1	$\log_2(X)$

EX 1 $\log_{10} 3000$

Cmd. `3000 log`

EX 2 $\sqrt{5^2 + 12^2}$

Cmd. `5 sq 12 sq + sqrt`

EX 3 $\log_3 22$

Cmd. `3 22 logb`

EX 4 $\exp(-3/2)$

Cmd. `3 sq 2 / pm exp`

5.3 Trigonometric functions

Here is the list of trigonometric and inverse trigonometric functions:

Function	Keyword	R	D	Math
Sine	SIN	1	1	$\sin(X)$
Cosine	COS	1	1	$\cos(X)$
Tangent	TAN	1	1	$\tan(X)$
Arcsine	ASIN	1	1	$\arcsin(X)$
Arccosine	ACOS	1	1	$\arccos(X)$
Arctangent	ATAN	1	1	$\arctan(X)$

These keywords depend on the angle mode. If you input Cmd sin in degree mode, this software calls "sin (degree)".

The radian trigonometric functions are here:

Function	Keyword	R	D	Math
Sine (rad)	SINR	1	1	$\sin(X)$ [rad]
Coine (rad)	COSR	1	1	$\cos(X)$ [rad]
Tangent (rad)	TANR	1	1	$\tan(X)$ [rad]
Arcsine (rad)	ASINR	1	1	$\arcsin(X)$ [rad]
Arccosine (rad)	ACOSR	1	1	$\arccos(X)$ [rad]
Arctangent (rad)	ATANR	1	1	$\arctan(X)$ [rad]

The degree trigonometric functions are here:

Function	Keyword	R	D	Math
Sine (deg)	SIND	1	1	$\sin(X)$ [deg]
Coine (deg)	COSD	1	1	$\cos(X)$ [deg]
Tangent (deg)	TAND	1	1	$\tan(X)$ [deg]
Arcsine (deg)	ASIND	1	1	$\arcsin(X)$ [deg]
Arccosine (deg)	ACOSD	1	1	$\arccos(X)$ [deg]
Arctangent (deg)	ATAND	1	1	$\arctan(X)$ [deg]

The grade trigonometric functions are here:

Function	Keyword	R	D	Math
Sine (gra)	SING	1	1	$\sin(X)$ [gra]
Cosine (gra)	COSG	1	1	$\cos(X)$ [gra]
Tangent (gra)	TANG	1	1	$\tan(X)$ [gra]

to next page

from prev page

Function	Keyword	R	D	Math
Arcsine (gra)	ASING	1	1	$\arcsin(X)$ [gra]
Arccosine (gra)	ACOSG	1	1	$\arccos(X)$ [gra]
Arctangent (gra)	ATANG	1	1	$\arctan(X)$ [gra]

EX 1 sin(30) (mode dependent)

Cmd. `30 sin`

EX 2 cos(52°)

Cmd. `52 tand`

5.4 Hyperbolic functions

Use following keywords to calculate hyperbolic functions:

Function	Keyword	R	D	Math
Hyperbolic sine	SINH	1	1	$\sinh(X)$
Hyperbolic cosine	COSH	1	1	$\cosh(X)$
Hyperbolic tangent	TANH	1	1	$\tanh(X)$
Inverse hyperbolic sine	ASINH	1	1	$\operatorname{asinh}(X)$
Inverse hyperbolic cosine	ACOSH	1	1	$\operatorname{acosh}(X)$
Inverse hyperbolic tangent	ATANH	1	1	$\operatorname{atanh}(X)$

EX 1 cosh(1.2)

Cmd. `1.2 cosh`

5.5 Stats functions

Stats functions are here:

Function	Keyword	R	D	Math
Beta function	BETA	2	2	$B(Y, X)$
Gamma function	GAMMA	1	1	$\Gamma(X)$
Logarithm of gamma function	LNGAMMA	1	1	$\log_e \lvert\Gamma(X)\rvert$
Error function	ERF	1	1	$\operatorname{erf}(X)$
Complementary error function	ERFC	1	1	$1 - \operatorname{erf}(X)$

EX 1 $B(0.5, 1.6)$

Cmd. `0.5 1.6 beta`

EX 2 $\Gamma(2)$

Cmd. `2 gamma`

5.6 Integer roundings

Integer roundings are here:

Function	Keyword	R	D	Math
Floor function	`FLOOR, FLR`	1	1	$\lfloor X \rfloor$
Ceiling function	`CEIL`	1	1	$\lceil X \rceil$
Round	`ROUND, RND`	1	1	$\lfloor X + 0.5 \rfloor$

EX 1 $\lfloor -2.2 \rfloor$

Cmd. `-2.2 flr`

EX 2 $\lceil \pi \rceil$

Cmd. `pi ceil`

5.7 Integer functions

Functions for integers such as GCD and LCM are here:

Function	Keyword	R	D	Math
Factorial	`FACT, !`	1	1	$X!$
Greatest common divisor	`GCD`	1	1	$\mathrm{GCD}(Y, X)$
Least common multiple	`LCM`	1	1	$\mathrm{LCM}(Y, X)$
Permutations	`PERM`	1	1	$P(Y, X) = \frac{Y!}{(Y-X)!}$
Combinations (binomial coefficient)	`COMB`	1	1	$C(Y, X) = \frac{Y!}{X!(Y-X)!}$

EX 1 $P(5, 2)$

Cmd. `5 2 perm`

EX 2 $\mathrm{LCM}(12, 50)$

Cmd. `12 50 lcm`

Chapter 6

Useful Functions

6.1 Percent calculations

Percent calculations such as including tax are here:

Function	Keyword	R	D	Math
X percent of Y	`PERC, PC`	2	1	$Y \times X/100$
Delta percent between Y and X	`DPERC, DP`	2	2	$\frac{X-Y}{Y} \times 100$
Include tax	`INTAX`	2	2	$\frac{Y \times (100+X)}{100}$
Exclude tax	`EXTAX`	2	2	$\frac{Y \times 100}{100+X}$

These functions support only scalars.

EX 1 3% of 5.15

Cmd. `5.15 3 pc`

EX 2 Delta-% between 1.2 and 1.3

Cmd. `1.2 1.3 dp`

EX 3 Include 8% tax to 1250

Cmd. `1250 8 intax`

EX 4 Exclude 8% tax from 120

Cmd. `120 8 extax`

6.2 Time conversion

Conversions between sec, min, hour, day and week are here.

Function	Keyword	R	D	Math
Seconds to minutes	`STOM`	1	1	$X/60$
Seconds to hours	`STOH`	1	1	$X/3600$
Seconds to days	`STOD`	1	1	$X/86400$
Seconds to weeks	`STOW`	1	1	$X/604800$
Minutes to seconds	`MTOS`	1	1	$X \times 60$
Minutes to hours	`MTOH`	1	1	$X/60$
Minutes to days	`MTOD`	1	1	$X/1440$
Minutes to weeks	`MTOW`	1	1	$X/10080$
Hours to seconds	`HTOS`	1	1	$X \times 3600$
Hours to minutes	`HTOM`	1	1	$X \times 60$
Hours to days	`HTOD`	1	1	$X/24$
Hours to weeks	`HTOW`	1	1	$X/168$
Days to seconds	`DTOS`	1	1	$X \times 86400$
Days to minutes	`DTOM`	1	1	$X \times 1440$
Days to hours	`DTOH`	1	1	$X \times 24$
Days to weeks	`DTOW`	1	1	$X/7$
Weeks to seconds	`WTOS`	1	1	$X \times 604800$
Weeks to minutes	`WTOM`	1	1	$X \times 10080$
Weeks to hours	`WTOH`	1	1	$X \times 168$
Weeks to days	`WTOD`	1	1	$X \times 7$

These functions support only scalars.

EX 1 45 mins to hours

```
Cmd: 45 mtoh
```

EX 2 65536 secs to days

```
Cmd: 65536 stod
```

6.3 DMS conversion

DMS conversion divides a scalar value into degrees / minutes / seconds.
Inverse DMS conversion combines degrees / minutes / seconds into a value.

Function	Keyword	R	D	Math
Decimal deg to deg/min/sec	`TODMS`	1	1	$Z \leftarrow D$ $Y \leftarrow M$ $X \leftarrow S$
Deg/min/sec to decimal deg	`DMSTO`	3	3	$Z + \frac{Y}{60} + \frac{X}{3600}$

These functions support only scalars.

EX 1 4096 sec to h:m:s

Cmd. `4096 stoh todms`

EX 2 30°20'10" to degrees

Cmd. `30 20 10 dmsto`

6.4 Whole stack calculations

You can find sum or infinite product in the stack.

Function	Keyword	R	D	Math
Sum of sequence	`SUM`	N>1	N	$\sum_{i=1}^{n} x_i$
Product of sequence	`PROD`	N>1	N	$\prod_{i=1}^{n} x_i$
Arithmetic average	`AVR`	N>1	N	$\frac{1}{n}\sum_{i=1}^{n} x_i$
Geometric average	`GAVR`	N>1	N	$\sqrt[n]{\prod_{i=1}^{n} x_i}$
Harmonic average	`HAVR`	N>1	N	$\frac{n}{\prod_{i=1}^{n} x_i^{-1}}$

If there are errors in the process of the functions, the calculation is cancelled and the stack keeps on.

Other versions available:

Function	Keyword	R	D	Math
Partial sum of sequence	`PSUM`	N>2	M+1	
Partial product of sequence	`PPROD`	N>2	M+1	
Partial arithmetic average	`PAVR`	N>2	M+1	
Partial geometric average	`PGAVR`	N>2	M+1	
Partial harmonic average	`PHAVR`	N>2	M+1	
Sum of sequence without drop	`SUMW`	N>1	0	

to next page

from prev page

Function	Keyword	R	D	Math
Product of sequence without drop	PRODW	N>1	0	
Arithmetic average without drop	AVRW	N>1	0	
Geometric average without drop	GAVRW	N>1	0	
Harmonic average without drop	HAVRW	N>1	0	
Partial sum of sequence without drop	PSUMW	N>2	1	
Partial product of sequence without drop	PPRODW	N>2	1	
Partial arithmetic average without drop	PAVRW	N>2	1	
Partial geometric average without drop	PGAVRW	N>2	1	
Partial harmonic average without drop	PHAVRW	N>2	1	

6.5 Multiply by prefix

Multiplication by prefix means the removal of prefix. For instance, if you have to get meter from kilometer, multiply by 1000, which means kilo.

Here is the list of multiplications by prefix:

Function	Keyword	R	D	Math
Multiply by yocto	YOCTO	1	1	$X \times 10^{-24}$
Multiply by zepto	ZEPTO	1	1	$X \times 10^{-21}$
Multiply by atto	ATTO	1	1	$X \times 10^{-18}$
Multiply by femto	FEMTO	1	1	$X \times 10^{-15}$
Multiply by pico	PICO	1	1	$X \times 10^{-12}$
Multiply by nano	NANO	1	1	$X \times 10^{-09}$
Multiply by micro	MICRO	1	1	$X \times 10^{-06}$
Multiply by milli	MILLI	1	1	$X \times 10^{-03}$
Multiply by centi	CENTI	1	1	$X \times 10^{-02}$
Multiply by deci	DECI	1	1	$X \times 10^{-01}$
Multiply by deca	DECA	1	1	$X \times 10^{+01}$
Multiply by hecto	HECTO	1	1	$X \times 10^{+02}$

to next page

from prev page

Function	Keyword	R	D	Math
Multiply by kilo	KILO	1	1	$X \times 10^{+03}$
Multiply by mega	MEGA	1	1	$X \times 10^{+06}$
Multiply by giga	GIGA	1	1	$X \times 10^{+09}$
Multiply by tera	TERA	1	1	$X \times 10^{+12}$
Multiply by peta	PETA	1	1	$X \times 10^{+15}$
Multiply by exa	EXA	1	1	$X \times 10^{+18}$
Multiply by zetta	ZETTA	1	1	$X \times 10^{+21}$
Multiply by yotta	YOTTA	1	1	$X \times 10^{+24}$
Multiply by kibi	KIBI	1	1	$X \times 2^{10}$
Multiply by mebi	MEBI	1	1	$X \times 2^{20}$
Multiply by gibi	GIBI	1	1	$X \times 2^{30}$
Multiply by tebi	TEBI	1	1	$X \times 2^{40}$
Multiply by pebi	PEBI	1	1	$X \times 2^{50}$
Multiply by exbi	EXBI	1	1	$X \times 2^{60}$
Multiply by zebi	ZEBI	1	1	$X \times 2^{70}$
Multiply by yobi	YOBI	1	1	$X \times 2^{80}$

6.6 Divide by prefix

Division by prefix means the addition of prefix. For instance, if you have to get millimeter from meter, divide by 0.001, which means milli.

Here is the list of divisions by prefix:

Function	Keyword	R	D	Math
Divide by yocto	TOYOCTO	1	1	$X/10^{-24}$
Divide by zepto	TOZEPTO	1	1	$X/10^{-21}$
Divide by atto	TOATTO	1	1	$X/10^{-18}$
Divide by femto	TOFEMTO	1	1	$X/10^{-15}$
Divide by pico	TOPICO	1	1	$X/10^{-12}$
Divide by nano	TONANO	1	1	$X/10^{-09}$
Divide by micro	TOMICRO	1	1	$X/10^{-06}$
Divide by milli	TOMILLI	1	1	$X/10^{-03}$
Divide by centi	TOCENTI	1	1	$X/10^{-02}$
Divide by deci	TODECI	1	1	$X/10^{-01}$
Divide by deca	TODECA	1	1	$X/10^{+01}$
Divide by hecto	TOHECTO	1	1	$X/10^{+02}$
Divide by kilo	TOKILO	1	1	$X/10^{+03}$
Divide by mega	TOMEGA	1	1	$X/10^{+06}$

to next page

from prev page

Function	Keyword	R	D	Math
Divide by giga	TOGIGA	1	1	$X/10^{+09}$
Divide by tera	TOTERA	1	1	$X/10^{+12}$
Divide by peta	TOPETA	1	1	$X/10^{+15}$
Divide by exa	TOEXA	1	1	$X/10^{+18}$
Divide by zetta	TOZETTA	1	1	$X/10^{+21}$
Divide by yotta	TOYOTTA	1	1	$X/10^{+24}$
Divide by kibi	TOKIBI	1	1	$X/2^{10}$
Divide by mebi	TOMEBI	1	1	$X/2^{20}$
Divide by gibi	TOGIBI	1	1	$X/2^{30}$
Divide by tebi	TOTEBI	1	1	$X/2^{40}$
Divide by pebi	TOPEBI	1	1	$X/2^{50}$
Divide by exbi	TOEXBI	1	1	$X/2^{60}$
Divide by zebi	TOZEBI	1	1	$X/2^{70}$
Divide by yobi	TOYOBI	1	1	$X/2^{80}$

6.7 Angle conversion

Angle conversions here:

Function	Keyword	R	D	Math
Radian to degree	RTOD	1	1	$180X/\pi$
Radian to grade	RTOG	1	1	$200X/\pi$
Degree to radian	DTOR	1	1	$\pi X/180$
Degree to grade	DTOG	1	1	$10X/9$
Grade to radian	GTOR	1	1	$\pi X/200$
Grade to degree	GTOD	1	1	$9X/10$

6.8 Angle calculation

Complementary / supprementaly angle:

Function	Keyword	R	D	Math
Complementary angle [1]	CANG	1	1	
Complementary angle (Radian)	CANGR	1	1	$\pi/2 - X$
Complementary angle (Degree)	CANGD	1	1	$90 - X$
Complementary angle (Grade)	CANGG	1	1	$100 - X$

to next page

from prev page

Function	Keyword	R	D	Math
Supplementary angle [1]	SANG	1	1	
Supplementary angle (Radian)	SANGR	1	1	$\pi - X$
Supplementary angle (Degree)	SANGD	1	1	$180 - X$
Supplementary angle (Grade)	SANGG	1	1	$200 - X$

[1] Depends on angle mode

6.9 Ratio

Convert a rational number into two integers.

Function	Keyword	R	D	Math
Ratio	RATIO	1	1	$Y \leftarrow$ Numerator $X \leftarrow$ Denominator

6.10 Random numbers

You can generate random numbers:

Function	Keyword	R	D	Math
Random integer	RAND	0	0	Push Int
Random floating	FRAND	0	0	Push Flt

A random integer has 63 bits and random floating is generated from a random integer.
The algorithm of random generator is mersenne twister.

6.11 Cast

You can cast data types:

Function	Keyword	R	D	Math
Cast into integer	TOINT	1	1	
Cast into floating	TOFLT	1	1	
Cast into rational	TORAT	1	1	
Cast into bool	TOBOOL	1	1	
Cast into byte	TOBYTE	1	1	
Cast into word	TOWORD	1	1	
Cast into dword	TODWORD	1	1	

to next page

from prev page

Function	Keyword	R	D	Math
Cast into qword	`TOQWORD`	1	1	
Cast into word (Sign extend)	`TOSWORD`	1	1	
Cast into dword (Sign extend)	`TOSDWORD`	1	1	
Cast into qword (Sign extend)	`TOSQWORD`	1	1	

You can approximate floating to rational with "cast into rational". The approximation is using continued fraction.

6.12 Calculations for engineers

These are useful calculations for engineers:

Function	Keyword	R	D	Math
Multiply by 2π	`TPIX`	1	1	$2\pi X$
Divide by 2π	`DTPI`	1	1	$X/2\pi$
Parallel	`PARA`	1	1	$(Y^{-1}+X^{-1})^{-1}$
To decibel	`TODB`	1	1	$10\log_{10}\lvert X\rvert$
Decibel to	`DBTO`	1	1	$10^{X/10}$

6.13 Calculation for earthquakes

You can calculate magnitude of earthquakes

Function	Keyword	R	D	Math
Joule to magnitude of EQ	`TOEQM`	1	1	$(\log_{10}(X-4.8))/1.5$
Magnitude of EQ to joule	`EQMTO`	1	1	$10^{4.8+1.5X}$

6.14 Health calculations

These calculations are extras:

Function	Keyword	R	D	Math
Discomfort index	`DISCOM`	2	2	$0.81Y - 0.01X \times (0.99Y - 14.3) + 46.3$
Body mass index	`BMI`	2	2	$\frac{X}{(Y/100)^2}$

Chapter 7

Complex Calculations

7.1 Display of complex numbers

This software displays complex numbers as following:

Mode	Math	Display
Rectangular	$5+12i$	`5 + i12`
Polar (deg)	$13\angle 67$[deg]	`13 exp(+i67.d)`
Polar (rad)	$13\angle 1.3$[rad]	`13 exp(+i1.3)`
		`13 exp(+i0.41 Pi)`
Euler (gra)	$13\angle 75$[gra]	`13 exp(+i75.g)`

Type Cmd. `EULER` or Cmd. `EUL` to toggle Euler mode. If Euler mode is enabled, complex numbers are shown as polar display.

The argument display depends on angle mode. Use the keywords Cmd. `DEG`, Cmd. `RAD` or Cmd. `GRA` to change angle mode.

If you select radian, you can convert the argument to π radians. To switch the display, type Cmd. `PIRAD` or Cmd. `PRAD`.

Mode	Keyword	Display
Euler display	`EULER, EUL`	`[Eul]`
Degree mode	`DEG`	`(Deg)`
Radian mode	`RAD`	`(Rad)`
Grade mode	`GRAD, GRA`	`(Gra)`
π radian mode	`PIRAD, PRAD`	`[Eul(Pi)]`

When Euler display is on, even scalars are treated as complex numbers so its argument is displayed if its value is not 0.

7.2 How to make complex numbers

There are three ways to make complex numbers.

> Real and imaginary part accept only scalars.

7.2.1 Input imaginary and add or subtract

Try to make $2 + i3$.

Cmd. `2 i3 +`

7.2.2 Make complex from real and imaginary part

Push real and imaginary part in turn and make complex. Use the keyword Cmd. `MKCMP` or Cmd. `MKC` to make complex from rectangular.

Cmd. `2 3`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	2
X:	Integer	:	3

Cmd. `mkc`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Complex	:	2 + i3

7.2.3 Make complex from absolute value and argument

Push absolute value and argument in turn and make complex. Use the keyword Cmd. `MKE` to make complex from polar.

This keyword depends on angle mode. For example, make $1.5\angle 30°$ in degree mode.

Cmd. `1.5 30`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Floating	:	1.5
X:	Integer	:	30

Cmd. `mke`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Complex	:	1.29903811 + i0.75

Radian version is Cmd. `MKER`.
Degree version is Cmd. `MKED`.
Grade version is Cmd. `MKEG`.
You can generate complex with following keywords:

Function	Keyword	R	D	Math
Make complex from rectangular	`MKCMP`, `MKC`	2	2	$Y + iX$
Make complex from polar	`MKE`	2	2	$Y\angle X$
Make complex from polar (radian)	`MKER`	2	2	$Y\angle X$ [rad]
Make complex from polar (degree)	`MKED`	2	2	$Y\angle X$ [deg]
Make complex from polar (grade)	`MKEG`	2	2	$Y\angle X$ [gra]

7.3 Complex calculations

You can operate complex calculations:

Function	Keyword	R	D	Math
Real part	`RE`	1	1	$\mathrm{Re}(X)$
Imaginary part	`IM`	1	1	$\mathrm{Im}(X)$
Complex argument	`ARG`	1	1	$\arg X$
Complex argument (rad)	`ARGR`	1	1	$\arg X$ [rad]
Complex argument (deg)	`ARGD`	1	1	$\arg X$ [rad]
Complex argument (gra)	`ARGG`	1	1	$\arg X$ [rad]
Complex conjugation	`CONJ`	1	1	X^*

Complex magnitude is Cmd. `ABS`.

EX 1 $\arg(1 + i2)$
Cmd. `1 2 mkc arg`

EX 2 $\mathrm{Re}(15\angle 32°)$
Cmd. `15 32 mked re`

EX 3 $(6 + i3)^*$

Cmd. `6 3 mkc conj`

7.4 Disassemble complex

You can disassemble complex numbers:

Function	Keyword	R	D	Math
Real and imaginary	REIM	1	1	$Y \leftarrow \mathrm{Re}(X)$ $X \leftarrow \mathrm{Im}(X)$
Magnitude and argument	MAGA	1	1	$Y \leftarrow \lvert X \rvert$ $X \leftarrow \arg X$
Magnitude and argument (radian)	MAGAR	1	1	$Y \leftarrow \lvert X \rvert$ $X \leftarrow \arg X\,[\mathrm{rad}]$
Magnitude and argument (degree)	MAGAD	1	1	$Y \leftarrow \lvert X \rvert$ $X \leftarrow \arg X\,[\mathrm{deg}]$
Magnitude and argument (grade)	MAGAG	1	1	$Y \leftarrow \lvert X \rvert$ $X \leftarrow \arg X[\mathrm{gra}]$

EX 1 $15\angle 32°$ to Re/Im part (deg)

Cmd. `15 32 mked reim`

Cmd. `5 3 mkc magad`

EX 2 $5 + i3$ to magnitude and arg

7.5 Complex functions

This software supports complex functions:

- Square root, cube root
- Power, logarithm
- Trigonometric functions
- Hyperbolic functions

Complex trigonometric functions are available only in radian.

Chapter 8

Logical Calculations

8.1 Unsigned decimal and Boolean

This software displays unsigned decimal and Boolean as following:

Mode	Math	Display
Boolean	TRUE	`T`
	FALSE	`F`
Binary mode	255	`0b11111111`
Octal mode	255	`0377`
Signed decimal mode	255	`-1`
Unsigned decimal mode	255	`255`
Hexadecimal mode	255	`0xFF`

8.2 Bit length

You can operate logical calculations in calculation mode.

This software supports 8, 16, 32, 64 bits. The bit length setting is shown in the display. Switch the mode to change the bit length.

Mode	Keyword	Display
8-bit mode	`BYTE`	`(Byte)`
16-bit mode	`WORD`	`(Word)`
32-bit mode	`DWORD`	`(Dword)`
64-bit mode	`QWORD`	`(Qword)`

Set bit length and the bit length symbol changes.

Please notice that if you input too large value for selected bit length, the software masks its lower N-bit (N is selected length) and push the result.

8.3 N-ary number switching

You can find N-ary number mode in the display.

Use the keywords to switch N-ary number display mode:

Mode	Keyword	Display
Binary display	`BIN`	`(Bin)`
Octal display	`OCT`	`(Oct)`
Signed decimal display	`SDEC`	`(Sdec)`
Unsigned decimal display	`UDEC`	`(Udec)`
Hexadecimal display	`HEX`	`(Hex)`

Set N-ary and the N-ary symbol changes.

8.4 Input binary and Boolean

Input value as binary (unsigned integer) to operate logical calculations.

Boolean
: True value is `TRUE` or `T` and false value is `FALSE` or `F`.

Unsigned
: Type `u` and postfix non-signed integer.

Bin value
: Type `0b` and postfix binary expression using 0 and 1.

Oct value
: Type `0o` and postfix octal expression using 0 to 7.

Hex value
: Type `0x` and postfix hexadecimal expression using 0 to 9 and A to F.

The input data is shown as selected N-ary display mode. For example, input binary `0b1010` and the display is `0x0000000A` in hexadecimal mode.

```
Cmd. 0b1010
```

```
#    TYPE       :                    VALUE
4:              :
Z:              :
Y:              :
X:   Dword      :                0x0000000A
```

You can push binaries and Booleans at one time.

```
Cmd. t f
```

#	TYPE	:	VALUE
4:		:	
Z:	Dword	:	0x0000000A
Y:	Boolean	:	T
X:	Boolean	:	F

8.5 Fundamental logical calculations

Here is the list of fundamental logical calculations:

Function	Keyword	R	D	Math
Bitwise NOT	`NOT, ~`	1	1	$\overline{X}$
Bitwise AND	`AND, &`	2	2	$Y \wedge X$
Bitwise OR	`OR, |`	2	2	$Y \vee X$
Bitwise XOR	`XOR`	2	2	$Y \oplus X$
Bitwise NAND	`NAND`	2	2	$\overline{(Y \wedge X)}$
Bitwise NOR	`NOR`	2	2	$\overline{(Y \vee X)}$

EX 1 `0x1234 & 0b0111`

Cmd. `0x1234 0b0111 and`

EX 2 `not(65535)`

Cmd. `u65535 not`

8.6 Bit shift

Bit shift functions are here:

Function	Keyword	R	D	Math
Shift left	`SHL, <<`	1	1	`X << 1`
Shift logical right	`SHR, >>`	1	1	`X >> 1`
Shift arithmetic right	`SAR, >»`	1	1	`X >>> 1`
Shift Left (N times)	`SHLC`	2	2	`X << N`
Shift Right (N times)	`SHRC`	2	2	`X >> N`
Shift Arithmetic Right (N times)	`SARC`	2	2	`X >>> N`
Shift byte left	`SBL`	1	1	`X << 8`
Shift byte right	`SBR`	1	1	`X >> 8`
Shift nibble left	`SNL`	1	1	`X << 4`
Shift nibble right	`SNR`	1	1	`X >> 4`

EX 1 `0x1234 & 0b0111`

Cmd. `0x1234 0b0111 and`

EX 2 `not(65535)`

Cmd. `u65535 not`

8.7 Rotate

Bit rotates are here:

Function	Keyword	R	D	Math
Rotate left	`ROL`	1	1	Rotate X Left
Rotate right	`ROR`	1	1	Rotate X Right

EX 1 `rol(31)`

Cmd `u31 rol`

8.8 Other functions that support unsigned integer

Function	Keyword	R	D	Math
Increment	`INC, ++`	1	1	$X + 1$
Decrement	`DEC, --`	1	1	$X - 1$
Add	`ADD, +`	2	2	$Y + X$
Subtract	`SUB`	2, -	2	$Y - X$
Multiply	`MUL`	2, *	2	$Y \times X$
Divide	`DIV`	2, /	2	Y/X
Negate	`NEG, PM`	1	1	$-X$

The addition of 2 Booleans is XOR, and the multiplication of 2 Booleans is AND. If you increment Boolean, the result is always true.

8.9 Whole-stack logical calcultions

You can operate logical calculations for whole-stack.

Function	Keyword	R	D	Math
All AND	`ALLAND`	N>1	N	$x_1 \wedge x_2 \ldots$
All OR	`ALLOR`	N>1	N	$x_1 \vee x_2 \ldots$
All XOR	`ALLXOR`	N>1	N	$x_1 \oplus x_2 \ldots$

Chapter 9

Vector Calculations

9.1 Display of vectors

This software displays vectors as following:

Mode	Math	Display
Horizontal [Row]	[1 2 3]	`[1, 2, 3]`
Vertical (Col)	(3 2 1)	`(3, 2, 1)`

9.2 Making of vector

You can include scalars, complex numbers or even binaries in vectors.

The input of vectors is complicated. I recommend using register function. Please read chapter 11 to get more information.

You can make vector with the following keywords:

Function	Keyword	R	D	Math
Make row tuple	`MRTUP`	N	N	Push Tup.R
Make column tuple	`MCTUP`	N	N	Push Tup.C

There are three steps for making a vector:

Push elements of a vector
: Push data in turn.

Push the number of elements the vector contains
: Set the dimension of the vector.

Call making function
: The vector is pushed.

You can include integers, floatings, rationals, complexes, Booleans and unsigned integers in a vector.

Let's make row tuple $[1+i2 \quad 6]$.

(1) Push elements

Cmd: `1 2 mkc 6`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Complex	:	1 + i2
X:	Integer	:	6

(2) Push number of elements

Cmd: `2`

#	TYPE	:	VALUE
4:		:	
Z:	Complex	:	1 + i2
Y:	Integer	:	6
X:	Integer	:	2

(3) Make row tuple

Cmd: `mrtup`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Tuple[Row]	:	[1 + i2, 6]

Making column tuple is similar with this case.

You can make unit vectors easily.

Function	Keyword	R	D	Math
Make row unit tuple	MRUTUP	2	2	Push Tup.R
Make column unit tuple	MCUTUP	2	2	Push Tup.C

These functions requires 2 arguments: a dimension and a position.

- Push a integer as a dimension
- Push a integer as a position (starting with 1)
- Call making function

Let's make (0 1 0).

(1) Push the dimension

Cmd. `3`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	3

(2) Push the position

Cmd. `2`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	3
X:	Integer	:	2

(3) Make column unit tuple

Cmd. `mcutup`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Tuple(Col)	:	(0, 1, 0)

9.3 Extract element from tuple

Use the keyword Cmd. `TGET` to extract one element from a tuple.

Please make sure *Y* is a tuple and *X* is an integer as a position (starting with 1) to extract.

Function	Keyword	R	D	Math
Get element from tuple	TGET	2	2	Extract

This function drops 2 data, so the vector from that you extract is dropped. I recommend storing the tuple to a register and call to extract.

Please read chapter 11 to make comprehension of using register function.

You can crave a tuple into elements.

Function	Keyword	R	D	Math
Crave up	CUT, CRAVE	1	1	

The used tuple is dropped and the extracted elements are pushed in turn.

Let us extract the second element from (6 9 12).

(0) Make sure that the tuple exists

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Tuple[Col]	:	(6, 9, 12)

(1) Set a position

Cmd. 2

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Tuple[Col]	:	(6, 9, 12)
X:	Integer	:	2

(2) Extract

Cmd. tget

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	9

9.4 Four arithmetics of vectors

The four aritmetics keywords of vectors are same as those of scalars.

EX 1 (3 2 1) + (5 6 9)

Cmd. 3 2 1 3 mctup

Cmd. 5 6 9 3 mctup

Cmd. +

EX 2 (3 2 1) × 9

Cmd. 3 2 1 3 mctup 9 *

Please check that the calculations are defined.

9.5 Dot / cross product

Use the keywords to calculate dot / cross product:

Function	Keyword	R	D	Math
Dot product	`INNER`, `DOT`	2	2	$\vec{Y} \cdot \vec{X}$
Cross product	`OUTER`, `CROSS`	2	2	$\vec{Y} \times \vec{X}$

Outer product suppots only 3-dimensional tuples.

EX 1 $(3 \quad 2 \quad 1) \cdot (7 \quad 8 \quad 9)$

Cmd. `3 2 1 3 mctup`

Cmd. `7 8 9 3 mctup`

Cmd. `dot`

EX 2 $(1 \quad 2 \quad 3) \times (4 \quad 5 \quad 6)$

Cmd. `1 2 3 3 mctup`

Cmd. `4 5 6 3 mctup`

Cmd. `cross`

9.6 Norms of vectors

Here is the keywords of norms of vectors:

Function	Keyword	R	D	Math
Euclidian norm	`NORM`	1	1	$\sqrt{\sum_{i=1}^{\infty} \lvert x_i \rvert^2}$
Euclidian norm squared	`NSQ`	1	1	$\sum_{i=1}^{\infty} \lvert x_i \rvert^2$
Lp-norm	`PNORM`	2	2	$\left(\sum_{i=1}^{\infty} \lvert y_i \rvert^x\right)^{1/X}$
Max norm	`MAXNORM`	1	1	$\max(\lvert x_1 \rvert, \ldots, \lvert x_n \rvert)$

Euclidian norm, Euclidian norm squared and maxnorm require one vector.

For example, type following to find the Euclidian norm of $[3 \quad 5 \quad 7]$.

Cmd. `3 5 7 3 mrtup norm`

Lp-norm requires one vector and an integer as a dimension.

Type following to find the third norm of $[7 \quad 8 \quad 9]$.

Cmd. `7 8 9 3 mrtup 3 pnorm`

9.7 Transpose vectors

Use the keyword Cmd. `TRANS` to transpose vectors.

Function	Keyword	R	D	Math
Transpose	TRANS	1	1	X^T

This function supports matrices.

Chapter 10

Matrix Calculations

10.1 Display of matrices

This software handles matrices as tuples of row tuples.

Mode	Math	Display
-	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	`[[1, 2, 3], [4, 5, 6], [7, 8, 9]]`

10.2 Making matrices

This software supports matrix calculations. Matrices of this software can include scalars, complexes, Booleans and unsigned integers.

The input of matrices is complicated. I recommend using register function. Please read chapter 11 to get more information.

Use the keyword Cmd `MKMAT` to make a matrix.

Function	Keyword	R	D	Math
Make matrix	`MKMAT`	N	N	Push Mat

There are three steps for making a matrix:

Prepare same dimensional and directional vectors
: Make sure vectors are all row or all column.

Push an integer as a number of vectors
: The integer must be greater than zero.

Call making function
: Make a matrix from vectors.

Matrices include row tuples. The data types that tuples cannot include are not supported in matrices.

Let's input matrix A:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

(1-1) Make two row vectors

Cmd: `1 2 2 mrtup 3 4 2 mrtup`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Tuple[Row]	:	[1, 2]
X:	Tuple[Row]	:	[3, 4]

(1-2) Set a number of vectors

Cmd: `2`

#	TYPE	:	VALUE
4:		:	
Z:	Tuple[Row]	:	[1, 2]
Y:	Tuple[Row]	:	[3, 4]
X:	Integer	:	2

(1-3) Make matrix

Cmd: `mkmat`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Matrix	:	[[1, 2], [3, 4]]

(2-1) Push two column tuple

Cmd: `1 3 2 mctup 2 4 2 mctup`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Tuple(Col)	:	(1, 3)
X:	Tuple(Col)	:	(2, 4)

(2-2) Set a number of vectors

Cmd: `2`

#	TYPE	:	VALUE
4:		:	
Z:	Tuple(Col)	:	(1, 3)
Y:	Tuple(Col)	:	(2, 4)
X:	Integer	:	2

(2-3) Make matrix

Cmd `mkmat`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Matrix	:	[[1, 2], [3, 4]]

Make sure the sizes and directions of all vectors to make a matrix are same.

You can make a unit matrix easily. Use the keyword Cmd `MKUMAT`.

Function	Keyword	R	D	Math
Make unit matrix	MKUMAT	1	1	Push Mat

Set an integer as a dimension and call the function. For instance, input this to make 3-dim unit matrix:

Cmd `3 mkumat`

10.3 Get element or tuple from matrix

Get a tuple or a element from matrix to use following keywords:

Function	Keyword	R	D	Math
Get element from matrix	MGET	3	3	Extract
Get row tuple from matrix	MGETR	2	2	Extract
Get column tuple from matrix	MGETC	2	2	Extract

These functions drop a matrix. I recommend using register function. Please read chapter 11 to get more information about registers.

You can crave up matrices.

Function	Keyword	R	D	Math
Crave up	CUT, CRAVE	1	1	

A matrix is craved up into row tuples and they are pushed in turn. Go on to the next

sections to get usages of Cmd. MGET, Cmd. MGETR and Cmd. MGETC.

10.3.1 Get an element from matrix

You can get a tuple from a matrix. Cmd. MGETR is the row tuple version and Cmd. MGETC is the column tuple version.

Use the keyword Cmd. MGET to get an element from a matrix.

Make sure Z is matrix, Y is position i, X is position j. The position counting starts with 1.

Try to extract element $(1, 2)$ from matrix `[[1, 2], [3, 4]]`.

(0) Matrix is pushed

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Matrix	:	[[1, 2], [3, 4]]

(1) Select a position of an element

Cmd. `1 2`

#	TYPE	:	VALUE
4:		:	
Z:	Matrix	:	[[1, 2], [3, 4]]
Y:	Integer	:	1
X:	Integer	:	2

(2) Get an element from matrix

Cmd. `mget`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	2

10.3.2 Get tuple from matrix

You can get a tuple from a matrix. Cmd. MGETR is the row tuple version and Cmd. MGETC is the column tuple version.

Make sure Y is matrix and X is position. The position counting starts with 1.

Try to extract second column tuple from `[[1, 2], [3, 4]]`.

(0) Matrix is pushed

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Matrix	:	[[1, 2], [3, 4]]

(1) Select a position

Cmd: `2`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Matrix	:	[[1, 2], [3, 4]]
X:	Integer	:	2

(2) Get a column tuple from matrix

Cmd: `mgetc`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Tuple(Col)	:	(2, 4)

10.4 Four arithmetics of matrices

The four arithmetics keywords are similar with those of scalars.

EX 1 $\begin{bmatrix} 3 & 7 \\ 9 & 5 \end{bmatrix} - \begin{bmatrix} 2 & 6 \\ 2 & 4 \end{bmatrix}$

Cmd: `3 7 2 mrtup 9 5 2 mrtup 2 mkmat`

Cmd: `2 6 2 mrtup 2 4 2 mrtup 2 mkmat`

Cmd: `-`

EX 2 $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 6 \end{bmatrix}$

Cmd: `1 2 2 mrtup 3 4 2 mrtup 2 mkmat`

Cmd: `5 6 2 mctup`

Cmd: `*`

10.5 Determinant and inverse matrix

Here is determinant and finding inverse matrix:

Function	Keyword	R	D	Math
Determinant	`DET`	1	1	$\det X$
Invert	`INV`	1	1	X^{-1}

These functions support only square matrices. You cannot find inverse matrix of A if the

determinant of A is zero.

EX 1 $\begin{bmatrix} \sqrt{2} & 1 \\ 1 & \sqrt{2} \end{bmatrix}^{-1}$

```
Cmd: 2 sqrt 1 2 mrtup 1 2 sqrt 2 mrtup 2 mkmat inv
```

10.6 Transpose matrix

Here is the list of transpose functions:

Function	Keyword	R	D	Math
Transpose	`TRANS`	1	1	X^{T}
Hermitian transpose	`HTRANS`, `HCONJ`	1	1	$(X^{\mathrm{T}})^*$

Hermitian transpose function transposes matrix or vector and conjugate each element in it.

10.7 Other matrix functions

Here is the list of other matrix functions:

Function	Keyword	R	D	Math
Trace	`TRACE`	1	1	$\mathrm{tr}(X)$

Trace function supports only square matrices.

Chapter 11

Register Operations

11.1 What is register

A register is kind of a memory. Each register in this software is independent of the stack. You can calculate more quickly with register function. There are 26 registers: RA – RZ.

```
HOMURA: (FD) (Rad) (Hex) (Dword) [Reg]
Std:  9/15,  Stack: 3,  History: 0/2
============================================================
 #  TYPE       :                                      VALUE
RA: Floating   :                                 3.14159265
RB:            :
RC:            :

 Z: Integer    :                                          4
 Y: Integer    :                                          3
 X: Integer    :                                          2
------------------------------------------------------------
STORE to selected register
Ready to operate
```

You can store one data to each register and can load or delete any time. Even if the stack is changed or cleared, the registers keep on.

Registers can hold any data: scalars, vectors, errors even strings.

This manual uses following tables:

#	TYPE	:	VALUE
RA:		:	
RB:		:	
Z:		:	
Y:		:	
X:		:	

This table shows X, Y, Z in a stack and RA and RB in registers.

11.2 Register display

Type `REGISTER` or `REG` to switch register display. If register display is enabled, the symbol `[Reg]` is displayed.

You can change display page of registers:

Function	Keyword
Switch register display	`REGISTER`, `REG`
Next page of registers	`REGNEXT`, `RN`
Previous page of registers	`REGPREV`, `RP`
First page of registers	`REGFIRST`, `RF`

See also chapter 3 and chapter 4.

> Switching register display function and register page changing functions do not affect registers. So you can use registers without displaying registers.

11.3 Store to selected register

You can store X to selected register. Then X is dropped.

Use the following keywords to storing functions:

Function	Keyword	R	D
Store to RA	`STRA`	1	1
Store to RB	`STRB`	1	1
⋮	⋮		
Store to RZ	`STRZ`	1	1

Use the format Cmd `STR?` and replace ? by one alphabet.

Let's store the integer 5 to RA.

(1) Push

Cmd `5`

#	TYPE	:	VALUE
RA:		:	
RB:		:	
Z:		:	
Y:		:	
X:	Integer	:	5

(2) Store to RA

Cmd: stra

#	TYPE	:	VALUE
RA:	Integer	:	5
RB:		:	
Z:		:	
Y:		:	
X:		:	

If the selected register has data, the data is overwritten.

(1) Initial state

#	TYPE	:	VALUE
RA:	Integer	:	5
RB:		:	
Z:		:	
Y:		:	
X:	Integer	:	7

(2) Push

Cmd: 9

#	TYPE	:	VALUE
RA:	Integer	:	5
RB:		:	
Z:		:	
Y:	Integer	:	7
X:	Integer	:	9

(3) Overwrite RA

Cmd: stra

#	TYPE	:	VALUE
RA:	Integer	:	9
RB:		:	
Z:		:	
Y:		:	
X:	Integer	:	7

Cases of RB – RZ are similar with this.

11.4 Load from selected register

You can load from selected register to *X*. The selected register keeps its data. If it has no data, error message is displayed.

Use the following keywors to load:

Function	Keyword	R	D
Load RA	LDRA	0	0
Load RB	LDRB	0	0
⋮	⋮		
Load RZ	LDRZ	0	0

Use the format Cmd. `LDR?` and replace ? by one alphabet.

Try to add RA and RB.

(1) Initial state

#	TYPE	:	VALUE
RA:	Integer	:	9
RB:	Integer	:	4
Z:		:	
Y:		:	
X:		:	

(2) Load RA

Cmd. `ldra`

#	TYPE	:	VALUE
RA:	Integer	:	9
RB:	Integer	:	4
Z:		:	
Y:		:	
X:	Integer	:	9

(3) Load RB

Cmd. `ldrb`

#	TYPE	:	VALUE
RA:	Integer	:	9
RB:	Integer	:	4
Z:		:	
Y:	Integer	:	9
X:	Integer	:	4

(4) Add

Cmd. `+`

#	TYPE	:	VALUE
RA:	Integer	:	9
RB:	Integer	:	4
Z:		:	
Y:		:	
X:	Integer	:	13

Cases of RC – RZ are similar with this.

11.5 Delete selected register

You can remove data in selected register.

Use the following keywords to delete selected register:

Function	Keyword	R	D
Delete RA	DELRA	0	0
Delete RB	DELRB	0	0
⋮	⋮		
Delete RZ	DELRZ	0	0

Use the format Cmd. `DELR?` and replace ? by one alphabet.

(1) Initial state

#	TYPE	:	VALUE
RA:	Integer	:	9
RB:	Integer	:	4
Z:		:	
Y:		:	
X:		:	

(2) Delete RA

Cmd. `delra`

#	TYPE	:	VALUE
RA:		:	
RB:	Integer	:	4
Z:		:	
Y:		:	
X:		:	

11.6 Register calculation

You can calculate with selected register and store the result to it.

Here is the list of register calculations:

Function	Keyword	R	D	Math
Register increment	IR?, ++R?	0	0	$R \leftarrow R + 1$
Register decrement	DR?, --R?	0	0	$R \leftarrow R - 1$
Register addition	ADDR?, +R?	1	1	$R \leftarrow R + X$
Register subtraction	SUBR?, -R?	1	1	$R \leftarrow R - X$
Register multiplication	MULR?, *R?	1	1	$R \leftarrow R \times X$
Register division	DIVR?, /R?	1	1	$R \leftarrow R$ 竅ㄐ

Operate register calculations to drop one data and overwrite selected register with the result.

Example: register addition and register increment

(1) Initial state

#	TYPE	:	VALUE
RA:	Integer	:	9
RB:	Integer	:	4
Z:		:	
Y:		:	
X:		:	

(2) Increment RA

Cmd ira

#	TYPE	:	VALUE
RA:	Integer	:	10
RB:	Integer	:	4
Z:		:	
Y:		:	
X:		:	

(3) Push

Cmd `1.2`

#	TYPE	:	VALUE
RA:	Integer	:	10
RB:	Integer	:	4
Z:		:	
Y:		:	
X:	Floating	:	1.2

(4) Add to RB

Cmd `+rb`

#	TYPE	:	VALUE
RA:	Integer	:	10
RB:	Floating	:	5.2
Z:		:	
Y:		:	
X:		:	

11.7 Register clear

You can clear all registers:

Function	Keyword	R	D	Math
Register clear	`REGCLEAR`, `RCLR`	0	0	

If you would like to delte one or some registers, use delete functions.

> You can clear registers and stack with the keyword Cmd `AC`. It is all clear function.

11.8 Strings and registers

The registers accept strings. You can put a landmark to registers with strings.

The macro and registers combo is very affective. See also chapter 15.

Chapter 12

Stack Operations

12.1 Special stack operations

You can remove, insert or duplicate data or change the order of elements in the stack. There are many special stack operations.

12.2 Fundamental stack operations

Here is the list of fundamental stack operations:

Function	Keyword	R	D	Math
Drop	`DROP, \`	1	1	
Duplicate [1]	`DUP`	1	1	
Stack clear	`CLEAR,` `CLR`	N>0	N	

[1] Press enter without any input to call the function

12.3 Order changing functions

Here is the list of order changing functions:

Function	Keyword	R	D	Math
Swap	`SWAP, $`	2	0	
Rotate	`ROT`	3	0	
Unrotate	`UNROT`	3	0	
Roll	`ROLL`	N	1	
Roll D	`ROLLD`	N	1	

12.3.1 Swap

Swap function swaps 2 data at bottom of the stack. This function requires 2 arguments. The keywords are Cmd. `SWAP` and Cmd. `$`.

(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Integer	:	3
X:	Rational	:	9/4

(2) Swap

Cmd. `swap`

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Rational	:	9/4
X:	Integer	:	3

12.3.2 Rotate

Rotate function rotates Z, Y, X. This function requires 3 arguments.

$$\begin{pmatrix} Z \\ Y \\ X \end{pmatrix} \rightarrow \begin{pmatrix} Y \\ X \\ Z \end{pmatrix}$$

The keyword is Cmd. `ROT`.

(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Integer	:	3
X:	Rational	:	9/4

(2) Rotate

Cmd. `rot`

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Rational	:	9/4
X:	Integer	:	3

12.3.3 Unrotate

Rotate function rotates Z, Y, X reversely. This function requires 3 arguments.

$$\begin{pmatrix} Z \\ Y \\ X \end{pmatrix} \rightarrow \begin{pmatrix} X \\ Z \\ Y \end{pmatrix}$$

The keyword is Cmd. UNROT.

12.3.4 Roll

Roll function rotates data from selected position through X. The selected data is moved to X. The keyword is Cmd. ROLL.

The keyword is Cmd. ROLL.

(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Integer	:	3
X:	Rational	:	9/4

(2) Set a position

Cmd. 3

#	TYPE	:	VALUE
4:	Integer	:	256
Z:	Rational	:	9/4
Y:	Integer	:	3
X:	Integer	:	3

(3) Roll

Cmd. roll

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	3
Y:	Rational	:	9/4
X:	Integer	:	256

12.3.5 Roll D

Roll function rotates data from selected position through X reversely. X is moved to selected position.

The keyword is Cmd. ROLLD.

12.4 Duplicate and overwrite functions

Here is the list of duplicate and overwrite functions:

Function	Keyword	R	D
Over	OVER, O	2	0
Pick	PICK	N	0
Unpick	UNPICK	N	1
Duplicate last 2 items	XY, YX, DUP2	2	0
Duplicate twice	DUPDUP	1	0
Duplicate 2nd last item N times and push N	NDUPN	1	1

12.4.1 Over

Over function duplicates *Y* and push it. The keywords are Cmd. OVER and Cmd. O.
The keyword is Cmd. OVER or Cmd. O.
(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	16
X:	Integer	:	32

(2) Over
Cmd. o

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	16
Y:	Integer	:	32
X:	Integer	:	16

12.4.2 Pick

Pick function duplicates data in selected position.
The keyword is Cmd. PICK.
(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Integer	:	3
X:	Rational	:	9/4

(2) Set a position

Cmd. `3`

#	TYPE	:	VALUE
4:	Integer	:	256
Z:	Integer	:	3
Y:	Rational	:	9/4
X:	Integer	:	3

(3) Pick

Cmd. `pick`

#	TYPE	:	VALUE
4:	Integer	:	256
Z:	Integer	:	3
Y:	Rational	:	9/4
X:	Integer	:	256

12.4.3 Unpick

Unpick function replaces data in selected position X by Y.

The keyword is Cmd. `UNPICK`.

(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	256
X:	Integer	:	3

(2) Push

Cmd. `64`

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Integer	:	3
X:	Integer	:	64

(3) Push a position

Cmd. 2

#	TYPE	:	VALUE
4:	Integer	:	256
Z:	Integer	:	3
Y:	Integer	:	64
X:	Integer	:	2

(4) Unpick

Cmd. unpick

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	64
X:	Integer	:	3

12.4.4 Duplicate last 2 items

Duplicate last 2 items function duplicates Y and X and push them in turn.
The keywords are Cmd. XY, Cmd. YX and Cmd. DUP2.

(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	16
X:	Integer	:	32

(2) Duplicate last 2 items

Cmd. xy

#	TYPE	:	VALUE
4:	Integer	:	16
Z:	Integer	:	32
Y:	Integer	:	16
X:	Integer	:	32

12.4.5 Duplicate twice

Duplicate twice function operate duplicate function twice.
The keywords are Cmd. dupdup and Cmd. dd.

12.4.6 Duplicate last item N times and push N

Duplicate last N-1 items and push N function drops X and duplicate Y X times and then push X.

The keyword is Cmd. `NDUPN`.

(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	16
X:	Integer	:	32

(2) Set a number of items

Cmd. `2`

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	16
Y:	Integer	:	32
X:	Integer	:	2

(3) Duplicate 2nd last item N times and push N

Cmd. `ndupn`

#	TYPE	:	VALUE
4:	Integer	:	16
Z:	Integer	:	32
Y:	Integer	:	32
X:	Integer	:	2

12.5 Removal functions

Here is the list of removal functions:

Function	Keyword	R	D
Drop 2 items	`DROP2, \\`	2	2
Drop 3 items	`DROP3, \\\`	3	3
Drop N items	`DROPN`	N+1	N+1
Nip	`NIP`	2	2
Nip N-th item	`NIPN`	N	2

The details are here:

12.5.1 Drop 2 items

Drop 2 items function drops 2 items.
The keywords are Cmd. DROP2 and Cmd. \\.

12.5.2 Drop 3 items

Drop 3 items function drops 3 items.
The keywords are Cmd. DROP3 and Cmd. \\\.

12.5.3 Drop N items

Drop N items function drops X + 1 items. The keyword is Cmd. DROPN.
(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	256
Y:	Integer	:	3
X:	Rational	:	9/4

(2) Push a number of items to drop
Cmd. 2

#	TYPE	:	VALUE
4:	Integer	:	256
Z:	Integer	:	3
Y:	Rational	:	9/4
X:	Integer	:	2

(3) Drop N items
Cmd. dropn

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	256

12.5.4 Nip

Nip function removes *Y*.
The keyword is Cmd. NIP.
(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	16
X:	Integer	:	32

(2) Nip

Cmd: `nip`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	32

12.5.5 Nip N-th item

Nip N function removes data in the position $X - 1$.

The keyword is Cmd: `NIPN`.

(1) Initial state

#	TYPE	:	VALUE
4:		:	
Z:	Integer	:	64
Y:	Integer	:	16
X:	Integer	:	32

(2) Set a position

Cmd: `3`

#	TYPE	:	VALUE
4:	Integer	:	64
Z:	Integer	:	16
Y:	Integer	:	32
X:	Integer	:	3

(3) Nip N

Cmd: `nipn`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:	Integer	:	16
X:	Integer	:	32

12.6 Other stack operations

Here is the list of ther stack operations:

Function	Keyword	R	D
Number of stack items	DEPTH	0	0

Number of stack items function pushes the number of data in stack.

Chapter 13

Unit Conversions

> See also chapter 6 to get more information about conversions.

13.1 Supporting units

This software supports the units:

- length
- inv of length
- area
- inv of area
- volume
- inv of volume
- time
- inv of time
- mass
- velocity
- acceleration
- force
- pressure
- energy
- temperature

> These conversion functions support only scalars.

13.2 How to use unit conversion function

Type Cmd. `CONV` or Cmd. `CV` to call unit conversion. Then type 2 units to convert. The keyword Cmd. `REC` or Cmd. `Z` calls previous conversion.

Function	Keyword	R	D	Math
Unit conversion	`CONV, CV`	1	1	Unit conversion
Unit conversion (redo)	`REC, Z`	1	1	Unit conversion (Redo)

You can convert X with calling the function and type “from unit” and “to unit”.
For example, type this to convert inches into centimeter.
Cmd. `conv in cm`

If the each unit has different dimension, the combination is error.

13.3 Units of length

Here is the list of units of length:

Unit		Keyword	Ratio
Meter	[m]	M	1
Kilometer	[km]	KM	1 E+03
Centimeter	[cm]	CM	1 E-02
Millimeter	[mm]	MM	1 E-03
Nautical mile*1	[nmi]	NMI	1 852
Yard*1	[yd]	YD	0.914 4
Foot*1	[ft]	FT	0.304 8
Inch*1	[in]	IN	0.025 4
Mile*1	[mi]	MI	1 609.344
Fathom*2	[fath]	FATH	1.828 8
Pica*2	[pc]	PC	127/30000
Point*2	[pt]	PT	127/360000
*Shaku*4		SHAKU	10/33
*Sun*4		SUN	1/33
*Ken*4		KEN	20/11
*Jou*4		JOU	100/33
*Chou*4		CHOU	1200/11
*Ri*4		RI	43200/11

13.4 Units of length inverse

Here is the list of units of length inverse:

Unit		Keyword	Ratio
Per meter	[1/m]	/M	1
Per kilometer	[1/km]	/KM	1 E-03
Per centimeter	[1/cm]	/CM	1 E+02
Per millimeter	[1/mm]	/MM	1 E+03

to next page

*1 International unit
*2 British fathom
*3 PostScript unit
*4 Japanese traditional unit

from prev page

Unit		Keyword	Ratio
Per nautical mile	[1/nmi]	/NMI	1/1852
Per yard	[1/yd]	/YD	1250/1143
Per foot	[1/ft]	/FT	1250/381
Per inch	[1/in]	/IN	5000/127
Per mile	[1/mi]	/MI	125/201168
Per fathom	[1/fath]	/FATH	625/1143
Per pica	[1/pc]	/PC	30000/127
Per point	[1/pt]	/PT	360000/127
Per *Shaku*		/SHAKU	3.3
Per *Sun*		/SUN	33
Per *Ken*		/KEN	0.55
Per *Jou*		/JOU	0.33
Per *Chou*		/CHOU	11/1200
Per *Ri*		/RI	11/43200

13.5 Units of area

Here is the list of units of area:

Unit		Keyword	Ratio
Square meter	[m^2]	M2	1
Square kilometer	[km^2]	KM2	1 E+06
Square centimeter	[cm^2]	CM2	1 E-04
Square millimeter	[mm^2]	MM2	1 E-06
Are	[a]	ARE	1 E+02
Hectare	[ha]	HA	1 E+04
Acre	[ac]	ACRE	4 046.856 422 4
Square yard	[yd^2]	YD2	0.836 127 36
Square foot	[ft^2]	FT2	9.290 304 E-02
Square inch	[in^2]	IN2	6.451 6 E-04
Square mile	[mi^2]	MI2	2 589 988.110 336
Tsubo		TSUBO	400/121
Isse		ISSE	12000/121
Ittan		ITTAN	120000/121
Choubu		CHOUBU	1200000/121

13.6 Units of area inverse

Here is the list of units of area inverse:

Unit		Keyword	Ratio
Per square meter	[1/m^2]	/M2	1
Per square kilometer	[1/km^2]	/KM2	1 E-06
Per square centimeter	[1/cm^2]	/CM2	1 E+04
Per square millimeter	[1/mm^2]	/MM2	1 E+06
Per are	[1/a]	/ARE	1 E-02
Per hectare	[1/ha]	/HA	1 E-04
Per acre	[1/ac]	/ACRE	78125/316160658
Per square yard	[1/yd^2]	/YD2	1562500/1306449
Per square foot	[1/ft^2]	/FT2	1562500/145161
Per square inch	[1/in^2]	/IN2	25000000/16129
Per square mile	[1/mi^2]	/MI2	15625/40468564224
Per *tsubo*		/TSUBO	121/400
Per *isse*		/ISSE	121/12000
Per *ittan*		/ITTAN	121/120000
Per *choubu*		/CHOUBU	121/1200000

13.7 Units of volume

Here is the list of units of volume:

Unit		Keyword	Ratio
Cubic meter	[m^3]	M3	1
Cubic kilometer	[km^3]	KM3	1 E+09
Cubic centimeter	[cm^3]	CM3	1 E-06
Cubic millimeter	[mm^3]	MM3	1 E-09
Liter	[L]	L	1 E-03
Deciliter	[dL]	DL	1 E-04
Kiloliter	[kL]	KL	1
Milliliter	[mL]	ML	1 E-06
Cubic yard	[yd^3]	YD3	0.764 554 857 984
Cubic foot	[ft^3]	FT3	0.028 316 846 592
Cubic inch	[in^3]	IN3	1.638 706 4 E-05
Cubic mile	[mi^3]	MI3	4 168 181 825.440 579 584
UK gallon	[gal_{imp}]	IMG	4.546 09 E-03

to next page

from prev page

Unit		Keyword	Ratio
US gallon	[gal_{us}]	USG	3.785 411 784 E-03
Gou		GOU	2401/13310000
Shou		SHOU	2401/1331000
Itto		ITTO	2401/133100
Koku		KOKU	2401/13310

13.8 Units of volume inverse

Here is the list of units of volume inverse:

Unit		Keyword	Ratio
Per cubic meter	[$1/m^3$]	/M3	1
Per cubic kilometer	[$1/km^3$]	/KM3	1 E-09
Per cubic centimeter	[$1/cm^3$]	/CM3	1 E+06
Per cubic millimeter	[$1/mm^3$]	/MM3	1 E+09
Per liter	[1/L]	/L	1 E+03
Per deciliter	[1/dL]	/DL	1 E+04
Per kiloliter	[1/kL]	/KL	1
Per milliliter	[1/mL]	/ML	1 E+06
Per cubic yard	[$1/yd^3$]	/YD3	1953125000/1493271207
Per cubic foot	[$1/ft^3$]	/FT3	1953125000/55306341
Per cubic inch	[$1/in^3$]	/IN3	125000000000/2048383
Per cubic mile	[$1/mi^3$]	/MI3	1953125/8140980127813632
Per UK gallon	[$1/gal_{imp}$]	/IMG	100000000/454609
Per US gallon	[$1/gal_{us}$]	/USG	125000000000/473176473
Per *gou*		/GOU	13310000/2401
Per *shou*		/SHOU	1331000/2401
Per *itto*		/ITTO	133100/2401
Per *koku*		/KOKU	13310/2401

13.9 Units of time

Here is the list of units of time:

Unit		Keyword	Ratio
Second	[s]	SEC, S	1
Minute	[min]	MIN	60

to next page

from prev page

Unit		Keyword	Ratio
Hour	[h]	HOUR, H	3 600
Day	[d]	DAY, C	86 400
Week	[wk]	WEEK, WK	604 800
Normal year	[yr]	YEAR, YR	31 536 000
Gregolian year		GYEAR	31 556 952
Julian year		JYEAR	31 557 600

13.10 Units of time inverse

Here is the list of units of time inverse:

Unit		Keyword	Ratio
Per second	[1/s]	/SEC, /S	1
Per minute	[1/min]	/MIN	1/60
Per hour	[1/h]	/HOUR, /H	1/3600
Per day	[1/d]	/DAY, /D	1/86400
Per week	[1/wk]	/WEEK, /WK	1/604800
Per normal year	[1/yr]	/YEAR, /YR	1/31536000
Per Gregolian year		/GYEAR	1/31556952
Per Julian year		/JYEAR	1/31557600

13.11 Units of mass

Here is the list of units of mass:

Unit		Keyword	Ratio
Kilogram	[kg]	KG	1
Gram	[g]	G	1 E-03
Milligram	[mg]	MG	1 E-06
Metric ton	[t]	TON	1 E+03
Long ton	[lt]	LTON	1 016.046 908 8
Short ton	[st]	STON	907.184 74
Once	[ozav]	OZ	0.028 349 523 125
Pound	[lbav]	LB	0.453 592 37
Kan		KAN	3.75

to next page

from prev page

Unit	Keyword	Ratio
Ryou	RYOU	3.75 E-02
Momme	MOMME	3.75 E-03
Kin	KIN	0.6

13.12 Units of velocity

Here is the list of units of velocity:

Unit		Keyword	Ratio
Meter per second	[m/s]	M/S	1
Meter per minute	[m/min]	M/MIN	1/60
Kilometer per second	[km/s]	KM/S	1 E+03
Kilometer per hour	[km/h]	KM/H, KPH	5/18
Inch per second	[ips]	IPS	0.025 4
Foot per second	[fps]	FPS	0.304 8
Mile per hour	[mph]	MPH	0.447 04
Knot	[kn]	KN, KT	463/900

13.13 Units of acceleration

Here is the list of units of acceleration:

Unit		Keyword	Ratio
Meter per square second	[m/s^2]	M/S2	1
Kilometer per hour per second	[km/h/s]	KM/H/S, KPH/S	5/18
Gal	[Gal]	GAL	1 E-02
Inch per ssquare econd	[ips^2]	IPS2	0.025 4
Foot per square second	[fps^2]	FPS2	0.304 8
Mile per hour per second	[mph/s]	MPH/S	0.447 04
Knot per second	[kn/s]	KN/S	463/900

13.14 Units of force

Here is the list of units of force:

Unit		Keyword	Ratio
Newton	[N]	NEWTON	1
Dynne	[dyn]	DYN	1 E-05
Kilogram weight	[kgf]	KGF	9.806 65
Gram weight	[gf]	GF	9.806 65 E-03

13.15 Units of pressure

Here is the list of units of pressure:

Unit		Keyword	Ratio
Pascal	[Pa]	PA	1
Hectopascal	[hPa]	HPA	1 E+02
Kilopascal	[kPa]	KPA	1 E+03
Megapascal	[MPa]	MPA	1 E+06
Bar	[bar]	BAR	1 E+05
Millimeter of mercury	[mmHg]	MMHG	101325/760
Inch of mercury	[inHg]	INHG	3 386.388 64

13.16 Units of energy

Here is the list of units of energy:

Unit		Keyword	Ratio
Joule	[J]	J	1
Kilojoule	[kJ]	KJ	1 E+03
Megajoule	[MJ]	MJ	1 E+06
Electronvolt	[eV]	EV	1.602 176 620 8 E-19
Kilo-electronvolt	[keV]	KEV	1.602 176 620 8 E-16
Mega-electronvolt	[MeV]	MEV	1.602 176 620 8 E-13
Giga-electronvolt	[GeV]	GEV	1.602 176 620 8 E-10
Thermochemical calorie	[cal_{th}]	CAL	4.184
Kilocalorie	[$kcal_{th}$]	KCAL	4 184
Ton of TNT	[t_{TNT}]	TTNT	4.184 E+09

to next page

from prev page

Unit		Keyword	Ratio
Kilowatt hour	[kWh]	KWH	3.6 E+06
British thermal unit	[Btu]	BTU	1055.06

13.17 Units of temperature

Here is the list of units of temperature:

Unit		Keyword	D-ratio	Zero
Kelvin	[K]	KEL	1	0
Celsius	[°C]	DEGC	1	-273.15
Rankine	[si°R]	DEGR	5/9	0
Fahrenheit	[°F]	DEGF	5/9	-459.67

The values of absolute temperature of Celsius and Fahrenheit are not same.

For instance, conversion from Celcius to Fahrenheit is following:

$$\theta[°\mathrm{C}] = (\theta + 273.15) \times \frac{9}{5} - 459.67[°\mathrm{F}]$$

Chapter 14

Math / Scientific / Engineering constants

14.1 Input constants

This software supports many math / scientific / engineering constants. Type a keyword to push a constant.

> The source of the values of scientific constants is 2014 CODATA.

14.2 Math constants

Here is the list of math constants:

Name	Keyword	Value
PI	`PI`	3.141 592 653 589 79
Napier's constant	`E`	2.718 281 828 459 05
Euler-Mascheroni constant	`EG`	0.577 215 664 901 533

14.3 Fundamental physical constants

Here is the list of fundamental constants in physics:

Name	Symbol		Keyword	Value
Speed of light in vacum	c_0	[m/s]	`LIGHT, C0`	299 792 458
Magnetic constant	μ_0	[H/m]	`MAGNETIC, MU0`	1.256 637 061 436 E-06
Electric constant	ε_0	[F/m]	`ELECTRIC, EPS0`	8.854 187 817 620 E-12
Characteristic impedance of vacum	Z_0	[Ω]	`IMPEDANCE, Z0`	376.730 313 461

to next page

from prev page

Name	Symbol	Keyword	Value
Gravitation constant	G_0 [$m^3/kg/s^2$]	GRAVITATION, NEWTONIAN	6.674 08 E-11
Planck constant	h [J s]	PLANCK, H	6.626 070 040 E-34
Reduced Planck constant	$\hbar$ [J s]	RPLANCK, HBAR	1.054 571 800 E-34

14.4 Electromagnetics

Here is the list of constants in electromagnetics:

Name	Symbol	Keyword	Value
Elementary charge	e [C]	ECHARGE, EVOLT	1.602 176 620 8 E-19
Magnetic flux quantum	Φ_0 [Wb]	Q.FLUX	2.067 833 831 E-15
Conductance quantum	G_0 [S]	Q.CONDUCT	7.748 091 731 0 E-05
Resistance quantum	R_0 [Ω]	Q.RESIST	12 906.403 727 8
Josephson constant	K_J [Hz/V]	JOSEPHSON	483 597.852 5 E-09
von Klitzing constant	R_K [Ω]	KLITZING	25 812.807 455 5
Bohr magneton	μ_B [J/T]	B.MAGNETON	927.400 999 4 E-26
Nuclear magneton	μ_N [J/T]	N.MAGNETON	5.050 783 699 E-27

14.5 Nuclear physics

Here is the list of constants in nuclear physics:

Name	Symbol	Keyword	Value
Fine-structure constant	α [1]	FSTRUCT	7.297 352 566 4 E-03
Rydberg constant	R_∞ [m^{-1}]	RYDBERG	10 973 731.568 508
Bohr radius	a_0 [m]	B.RADIUS	0.529 177 210 67 E-10
Hartree energy	E_h [J]	HARTREE	4.359 744 650 E-18

Constants connected with electron:

Name	Symbol	Keyword	Value
Mass of electron	m_e [kg]	E.MASS	9.109 383 56 E-31

to next page

from prev page

Name	Symbol		Keyword	Value
Compton wavelength of electron	λ_e	[m]	E.COMPTON	2.426 310 236 7 E-12
Classical electron radius	r_e	[m]	E.RADIUS	2.817 940 322 7 E-15
Magnetic moment of electron	μ_e	[J/T]	E.MAGNETIC	-928.476 462 0 E-26
Gyromagnetic ratio of electron	γ_e	[$s^{-1}T^{-1}$]	E.GYRO	1.760 859 644 E+11

Constants connected with proton:

Name	Symbol		Keyword	Value
Mass of proton	m_p	[kg]	P.MASS	1.672 621 898 E-27
Compton wavelength of proton	λ_p	[m]	P.COMPTON	1.321 409 853 96 E-15
Magnetic moment of proton	μ_p	[J/T]	P.MAGNETIC	1.410 606 787 3 E-26
Gyromagnetic ratio of proton	γ_p	[$s^{-1}T^{-1}$]	P.GYRO	2.675 221 900 E+08

Constants connected with neutron:

Name	Symbol		Keyword	Value
Mass of neutron	m_n	[kg]	N.MASS	1.674 927 471 E-27
Compton wavelength of neutron	λ_n	[m]	N.COMPTON	1.319 590 904 81 E-15
Magnetic moment of neutron	μ_n	[J/T]	N.MAGNETIC	-0.966 236 50 E-26
Gyromagnetic ratio of neutron	γ_n	[$s^{-1}T^{-1}$]	N.GYRO	1.832 471 72 E+08

Other constants in nuclear physics:

Name	Symbol		Keyword	Value
Mass of muon	m_μ	[kg]	MU.MASS	1.883 531 594 E-28
Magnetic moment of muon	μ_μ	[J/T]	MU.MAGNETIC	-4.490 448 26 E-26
Mass of tauon	m_τ	[kg]	TAU.MASS	3.167 47 E-27

14.6 Physicochemistry

Here is the list of constants in physicochemistry:

Name	Symbol		Keyword	Value
Boltzmann constant	k	[J/K]	BOLTZMANN	1.380 648 52 E-23
Avogadro constant	N_A	[mol^{-1}]	AVOGADRO	6.022 140 857 E+23
Atomic mass constant	m_u	[kg]	DALTON	1.660 539 040 E-27
Faraday constant	F	[C/mol]	FARADAY	96 485.332 89
Molar gas constant	R	[$J K^{-1} mol^{-1}$]	GAS	8.314 4598
Molar volume[*1] (m^3)	V_m	[m^3/mol]	MOLV	22.413 962 E-03
Molar volume[*1] (L)	V_m	[L/mol]	MOLVL	22.413 962
Loschmidt' s constant[*1]	n_0	[m^{-3}]	LOSCHMIDT	2.686 7811 E+25

Here is the list of constants in thermal radiation:

Name	Symbol		Keyword	Value
Stefan-Boltzmann constant	σ	[$W m^{-2} K^{-4}$]	STEFAN	5.670 367 E-08
First radiation constant	c_1	[$W m^2$]	F.RAD	3.741 771 790 E-16
Second radiation constant	c_2	[m K]	S.RAD	1.438 777 36 E-02

14.7 Agreement value

Here is the list of agreement values:

Name	Symbol		Keyword	Value
Standard gravity	g_n	[m/s^2]	GRAVITY	9.806 65
Standard atmosphere	1atm	[Pa]	ATM	10 1325
Zero degrees Celsius in Kelvin	0°C	[K]	ZEROD	273.15

[*1] In 0 degrees centigrade and standard atomosphere pressure (273.15[K], 1[atm]).

14.8 Planck unit

Here is the list of Planck unit:

Name	Symbol	Keyword	Value
Planck mass	m_P [kg]	PL.MASS	2.176 470 E-08
Planck energy	E_P [GeV]	PL.ENERGY	1.220 910 E+19
Planck temperature	T_P [K]	PL.TEMP	1.416 808 E+32
Planck length	l_P [m]	PL.LENGTH	1.616 229 E-35
Planck time	t_P [s]	PL.TIME	5.391 16 E-44

14.9 Astronomy

Here is the list of constants of astronomy:

Name	Symbol	Keyword	Value
Astronomical unit	AU [m]	ASTRO	149 597 870 700
Parsec	pc [m]	PARSEC	3.085 677 581 E+16
Light year	ly [m]	LYEAR	9 460 730 472 580 800

14.10 Paper size

Here is the list of constants of paper size:

Name	Keyword	Value	Name	Keyword	Value
Short side of A0(mm)	A0.S	841	Long side of A0(mm)	A0.L	1189
Short side of A1(mm)	A1.S	594	Long side of A1(mm)	A1.L	841
Short side of A2(mm)	A2.S	420	Long side of A2(mm)	A2.L	594
Short side of A3(mm)	A3.S	297	Long side of A3(mm)	A3.L	420
Short side of A4(mm)	A4.S	210	Long side of A4(mm)	A4.L	297
Short side of A5(mm)	A5.S	148	Long side of A5(mm)	A5.L	210
Short side of A6(mm)	A6.S	105	Long side of A6(mm)	A6.L	148
Short side of A7(mm)	A7.S	74	Long side of A7(mm)	A7.L	105
Short side of A8(mm)	A8.S	52	Long side of A8(mm)	A8.L	74

to next page

from prev page

Name	Keyword	Value	Name	Keyword	Value
Short side of A9(mm)	A9.S	37	Long side of A9(mm)	A9.L	52
Short side of B0(mm)	B0.S	1030	Long side of B0(mm)	B0.L	1456
Short side of B1(mm)	B1.S	728	Long side of B1(mm)	B1.L	1030
Short side of B2(mm)	B2.S	515	Long side of B2(mm)	B2.L	728
Short side of B3(mm)	B3.S	364	Long side of B3(mm)	B3.L	515
Short side of B4(mm)	B4.S	257	Long side of B4(mm)	B4.L	364
Short side of B5(mm)	B5.S	182	Long side of B5(mm)	B5.L	257
Short side of B6(mm)	B6.S	128	Long side of B6(mm)	B6.L	182
Short side of B7(mm)	B7.S	91	Long side of B7(mm)	B7.L	128
Short side of B8(mm)	B8.S	64	Long side of B8(mm)	B8.L	91
Short side of B9(mm)	B9.S	45	Long side of B9(mm)	B9.L	64
letter short (mm)	LET.S	215.9	letter long (mm)	LET.L	279.4
legal short (mm)	LEG.S	215.9	legal long (mm)	LEG.L	355.6
letter short (inch)	LET.SI	8.5	letter long (inch)	LET.LI	11
legal short (inch)	LEG.SI	8.5	legal long (inch)	LEG.LI	14

Chapter 15

Other functions

15.1 All clear

You can clear stack and registers with all clear function.

Function	Keyword
All clear	AC
Stack clear	CLEAR, CLR
Register clear	REGCLEAR, RCLR

You can use undo after you call clear functions.

15.2 All reset

Type Cmd. RESET or Cmd. RST to reset all settings without those in config mode. Call the function and type Cmd. YES or Cmd. NO to confirm.

15.3 Undo / redo

Undo and redo function is available:

Function	Keyword
Undo	UNDO, U
Redo	REDO, R

See also chapter 2 and chapter 3.

15.4 JSON output

Type Cmd. JSON or Cmd. OUT to output JSON formatted text file.

Function	Keyword
JSON output	`JSON`

This software output files to the directory it exists. The format of file name is following: `eckert_YYYY_MMDD_HHMMSS.json`

Output JSON file and its file name is displayed in message area. You can save stack and registers states.

15.5 Macro function

This software supports macro with strings.

Function	Keyword	R	D
Run macro	`RUN`	1	1

Macro function reads X as a string and operate.

Here is an example of using macro function:

(1) Push string `"2 3 +"`

Cmd. `"2 3 +"`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	String	:	2 3 +

(2) Run macro

Cmd. `run`

#	TYPE	:	VALUE
4:		:	
Z:		:	
Y:		:	
X:	Integer	:	5

You can make easy user defined function with macro function.

For example, the macro string of $RA + \sqrt{RB \times RC}$ is `"ldra ldrb ldrc * sqrt +"`. Store it to RE.

Set RA, RB and RC. Then load RE and run macro to calculate $RA + \sqrt{RB \times RC}$.

> You cannot include keyword Cmd. `RUN`, which is macro, in a string for macro function. This specification is for avoiding infinite loop.

> Similarly, you cannot include mode-changing, display-changing keywords.

15.6 Test precisions

You can test precisions of this software.

Function	Keyword	R	D	Math
Radix of floating	`RADIX`	0	0	Push Int
Machine epsilon	`EPSILON`	0	0	Push Flt

These functions are for debugging.

15.7 Special startup

This software supports command line arguments.

Argument	Setting
`-d`	Do not clear display
`-j`	JSON file output
`-jd`	JSON display (console)
`--`	Split for JSON expression

If you would like to keep display buffers, use Cmd. `-d` option.

Cmd. `eckert64.exe -d`

JSON file output and JSON display uses Cmd. `--`. Write expressions after Cmd. `--`.

Example:

Cmd. `eckert64.exe -j -- 1 2 3 sum stra pi exp strz sum dup i mul 2`

Replace Cmd. `-j` into Cmd. `-jd` to display upon console.

Chapter 16

Messages

16.1 Error messages

The list of error messages in this software is following:

- `Bad argument count`
- `Bad argument type`
- `Bad element`
- `Bad matrix size`
- `Bad tuple size`
- `Determinant is zero`
- `Division by zero`
- `Empty input`
- `Failed to output file`
- `Final page of register`
- `Final page of stack`
- `First page of register`
- `First page of stack`
- `From ______ to ______ : INVALID` (Unit conversion)
- `Invalid conversion`
- `Invalid input`
- `Invalid range`
- `Invalid value`
- `Latest history`
- `Logarithm of zero`
- `Maximum integer`
- `Minimum integer`
- `Negative-th power of zero`
- `No history`
- `No older history`
- `Not a positive integer`

- `Registers are empty`
- `Selected register is empty`
- `Stack and registers are empty`
- `Stack is empty`
- `Too few arguments`
- `Too large or small input`
- `Too large to operate`
- `Unsupported in current version`
- `Unsupported operation or notation`
- `Zero-th power of zero`

16.2 Notice messages

The list of notice messages is following:

- `Error calculation`
- `Floating overflow`
- `Integer overflow`
- `Rational overflow`

16.3 Confirm messages

The list of confirm messages is here:

- `Cancelled`
- `Done`
- `From _____ to _____` (Unit conversion)
- `Input integer`
- `Maximum value set`
- `Minimum value set`
- `OK? Y/N`
- `Setting completed`

Chapter 17

Technical Information

17.1 Supporting types

The list of data types this software supports is following:

Type	Value range	Class
`Integer`	64-bit integer	Scalar
`Floating`	long double	Scalar
`Rational`	Pair of 64-bit integers	Scalar
`Complex`	Pair of scalars	Number
`Boolean`	True, False	Unsigned
`Byte`	Unsigned 8-bit	Unsigned
`Word`	Unsigned 16-bit	Unsigned
`Dword`	Unsigned 32-bit	Unsigned
`Qword`	Unsigned 64-bit	Unsigned
`Tuple`	Tuple of scalars ortuple of binaries	Tuple
`Matrix`	Tuple of tuples	Tuple
`Infinity`	Positive, negative, complex	Not a number
`String`	String	Not a number
`Error`	String	Not a number

If integer overflow occurs, the calculation is retried as floating point number. If floating-point overflow occurs, the result of calculation is handled as Infinity.

17.2 Calculation precision

The concept of this software is useful for engineers, but no accuracy assurances. So this software is not suitable for high precision calculations.

The internal precision of this software is displayed with calculation settings. The data are using binaries, so floating-point calculations cause calculation errors. Then, this software does not correct calculation errors.

17.3 Mathematical definitions

Mathematical definitions this software adopts is following:

17.3.1 Remainde of integers (Modulo)

Remainde of integers is defined as:

A	B	Quotient	Remainder
Neg	Neg	$(-A) \div (-B)$	$-((-A) \mod (-B))$
Neg	Pos	$(-A) \div (-B)$	$-((-A) \mod (-B))$
Zero	Nonzero	0	0
Pos	Neg	$-(A \div (-B)$	$(-A) \mod B$
Pos	Pos	$A \div B$	$A \mod B$

17.3.2 Odd number-th root of negative value

The odd number-th root, such as cubic root or 5th root of negative value is not defined in range of real number. For instance, the cubic root of -1 is not -1.

The odd number-th root is defined in complex number:

$$\begin{aligned}\sqrt[n]{a+ib} &= \sqrt[n]{r}\exp(i\theta/n) \\ &= \sqrt[n]{r}(\cos(\theta/n)) + i\sin(\theta/n)\end{aligned}$$

17.3.3 Definition of complex numbers

Complex absolution and argument are defined as:

$$\mathrm{abs}(a+ib) = r = \sqrt{a^2+b^2}$$

$$\arg(a+ib) = \theta = \begin{cases} \mathrm{atan}(b/a) & (a>0) \\ \pi/2 & (a=0, b>0) \\ -\pi/2 & (a=0, b<0) \\ \pi - \mathrm{atan}(b/a) & (a>0, b>0) \\ \mathrm{atan}(b/a) - \pi & (a<0, b<0) \\ \mathrm{Error} & (a=b=0) \end{cases}$$

This is the basics of complex functions.

17.3.4 Complex functions

Here is the table of definitions of complex functions.

Function	Definition
Sqiare root	$\sqrt{a+ib} = \sqrt{r}\,(\cos(\theta/2) + i\sin(\theta/2))$
Cube root	$\sqrt[3]{a+ib} = \sqrt[3]{r}\,(\cos(\theta/3) + i\sin(\theta/3))$
Exponent	$\exp(a+ib) = e^a(\cos b + i\sin b)$
Natural log	$\ln(a+ib) = \ln r + i\theta$
Power	$(a+ib)^{c+id} = r^c e^{-d\theta}[\cos(c\theta + d\ln r) + i\sin(c\theta + d\ln r)]$
sin	$\sin(a+ib) = \sin a\cosh b + i\cos a\sinh b$
cos	$\cos(a+ib) = \cos a\cosh b - i\sin a\sinh b$
tan	$\tan(a+ib) = \frac{1}{2}\cdot\frac{\sin 2a}{\cos^2 a + \sinh^2 b} + i\frac{1}{2}\cdot\frac{\sinh 2b}{\cos^2 a + \sinh^2 b}$
asin	$\operatorname{asin}(z) = -i\ln\left(\sqrt{1-z^2} + zi\right)$
acos	$\operatorname{acos}(z) = -i\ln\left(z + i\sqrt{1-z^2}\right)$
atan	$\operatorname{atan}(z) = \frac{i}{2}\ln\left(\frac{i+z}{i-z}\right) \quad (z \neq \pm i)$
sinh	$\sinh(a+ib) = \sinh a\cos b + i\cosh a\sin b$
cosh	$\cosh(a+ib) = \cosh a\cos b + i\sinh a\sin b$
tanh	$\tanh(a+ib) = \frac{\sinh 2a}{\cosh 2a + \cos 2b} + i\frac{\sin 2b}{\cosh 2a + \cos 2b}$
asinh	$\operatorname{asinh}(z) = \ln\left(z + \sqrt{z^2+1}\right)$
acosh	$\operatorname{acosh}(z) = \ln\left(z + \sqrt{z+1}\sqrt{z-1}\right)$
atanh	$\operatorname{atanh}(z) = \frac{1}{2}\ln\left(\frac{1+z}{1-z}\right) \quad (z \neq \pm 1)$

Chapter 18

Troubleshootings

18.1 I have no idea to operate this software

Please restart this software and read chapter 4.

This software adopts RPN-style (stack). You can make comprehension of it with reading chapter 4 so please read it carefully.

18.2 I'd like to view full data

If you find `...` in stack or register display, type Cmd `V` to show full data (VIEW mode). Press enter to return to calculation mode from view mode.

18.3 I'd like to change rational or floating display

Use the following keywords to change rational or floating display:

Function	Keyword
Auto decimal display	`AD`
Force decimal display	`FD`
Force floating display	`FF`
Standard decimal display	`STD`
Fixed decimal display	`FIX`
Scientific decimal display	`SCI`
Engineering decimal display	`ENG`

Please read chapter 3 to get more information.

18.4 I'd like to change complex display

Type Cmd `EUL` to switch complex number display. The argument of complex display depends on angle mode.

Please read chapter 3 to get more information.

18.5 I'd like to view all values in the stack and the registers

JSON output function is recommended. Please read chapter 15.

If you would like to look at some data, try page-flipping function. Please read chapter 3 to get more information.

18.6 I saw doubtful calculation result

Restart the software and retry.

Supported numbers in this software are expressed in binary so the calculations may have small errors. The expected answer is `0.1` but this shows `0.0999...`? That is within the spec.

18.6.1 Check keywords

Did not you type wrong spelling? Check the keywords.

18.6.2 Check display mode

Were not you confused by display mode? Try another display mode and check the value. Please read chapter 3 to change modes.

18.6.3 Check angle mode

Did you noticed the unit of angle in your calculation? Trigonometric functions depend on angle mode. So a called trigonometric function is determined by a keyword and angle mode.

Please read chapter 3 to change modes.

18.6.4 Check range of value

Some functions may cause large errors depending on range of value. For instance, input a large value to trigonometric functions to make unreliable results.

See also chapter 17.

18.6.5 Check the order of calculations

If the expression is changeable in math, calculators may make small errors. Please calculate by changing orders with consideration of less error.

18.7 Stopped by errors

Check types or values of data. For instance, the factorial of floating-point number is not defined.

18.7.1 Check types

You can check the type of data in the second left column in the stack display. If types are not shown, type Cmd TYPE to display. Check types of arguments of functions.

18.7.2 Check values

Did you input error value? Some functions have undefined input. For example, logarithm of 0 is undefined.

18.7.3 Check sizes of vectors and matrices

Please notice that the calculations of vectors or matrices are defined.

18.7.4 Read error messages

The messages may help you to detect operational errors.

18.8 I found doubtful behaviors

If you find bugs or unnatural specifications, please send messages to me.

Contact: mailto:only.my.truth@gmail.com