

The EGA Reference Manual

Written by Katayama Hirofumi MZ.

Copyright (C) 2020-2026 Katayama Hirofumi MZ.

What is EGA?

EGA is a small programming language of a simple grammar, written in C++11. It is very tiny (< 500 KiB). You can extend its functions easily.

The source code of EGA will be found at <https://github.com/katahiromz/EGA> .

How to use

Please start up EGA. The following text will be displayed:

```

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ EGA Version 15 by katahiromz                                     @
@ Type 'exit' to exit. Type 'help' to see help. @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

```

Enter an EGA expression (for example, `print(+(1, 2))`) and press `Enter` key. `3` will be shown.

```

EGA> print(+(1, 2));
3

```

To quit EGA, please enter `exit` .

```

EGA> exit;

```

The syntax of the EGA is similar to one or more function call(s) of C language. But, every operator in EGA is a function.

Enter `help` to see all the EGA functions:

```

EGA> help;
EGA has the following functions:
!
!=
%
&

```

```
&&  
...  
and  
array  
...
```

To see brief usage of `print` function, enter `help print`.

```
EGA> help print;  
EGA function 'print':  
  arity: 0..32767  
  usage: print(value, ...)
```

The detailed descriptions of the EGA functions will be described later.

What's New

Change in Version 15:

- Added `RES_extract` function.

Change in Version 14:

- Changed arity of function `bitor ( | )`, `bitand ( & )`, and `xor ( ^ )`.
- Added `localtime` and `gmtime` functions.
- Added `load` and `save` functions.

Change in Version 13:

- Arity changed of function `add ( + )`, `mul ( * )`, `or ( || )` and `and ( && )`.

Change in Version 11:

- Removed EGA `input` function.
- Modified design of `input` function.

Change in Version 10:

- Added `RES_str_get` and `RES_str_set` functions.
- Added `RES_get_text` and `RES_set_text` functions.

Change in Version 9:

- Improved `binary` function.
- Added `RES_set_binary` function.
- Added `RES_const` function.

Change in Version 8:

- Improved `or` function in evaluation way.
- Improved `and` function in evaluation way.

Comments

A comment begins with the at symbol (`@`) at the beginning of a line and ends at the newline.

The comments are ignored in execution of EGA program.

For example:

```
@ This is a comment.
```

Values

The EGA values are integers, strings, and/or arrays.

An EGA integer literal is a sequence of digit(s) (`0` , ..., `9`).

An EGA string literal is a string wrapped by double quotations (`" "`). If the string value contains a double quotation, it will be doubled in the string literal.

The EGA array literal is a list of the EGA values it contains, separated by commas (`,`), and wrapped by braces (`{` and `}`).

Variables

You can set a value into a variable by using the `set` special function. For example, `set(A, 123);` will create a variable named `A` whose value is `123` .

The `define` special function can store the unevaluated expression into a variable.

Integers

Expression `+(1, 2)` is the sum of two integers `1` and `2` . Expression `*(3, 4)` is the multiplication of two integers `3` and `4` .

You can compare two integers by `==` , `!=` , `<` , `<=` , `>` , `>=` functions.

To specify a negative value, use `minus` (`-`) function: `-(123)` .

Strings

Expression `"This is a string."` is a string literal of length 17.

Expression `"This is a \"string\"."` is a string literal of length 19.

You can compare two strings by `==`, `!=`, `<`, `<=`, `>`, `>=` functions.

See also `left`, `len`, `mid`, `right`, `replace`, `remove` and `str` functions.

Binaries

In EGA, the binary data is a string. See `binary` function.

Arrays

Expression `{1, 2, "string"}` is an array literal of length 3.

Expression `set(ary, {1, 2, "string"})`; can store the array to the `ary` variable.

To get the 2nd element of `ary`, use `at(ary, 1)`. To set 999 to the 2nd element of `ary`, use `at(ary, 1, 999)`.

See also `left`, `len`, `mid`, `right`, `replace`, `remove` and `array` functions.

Booleans

In EGA, the boolean value is an integer value. Zero means false. Non- 0 means true.

See `not`, `and`, `or` functions.

Conditional execution

The special function `if` can switch the execution by condition.

For example:

```
if(==(+(1, 1), 2), println("1 + 1 == 2"), println("1 + 1 != 2"))
```

Loops

The special functions `for`, `foreach` and `while` can make an execution loop. The `break` special function can break the loop.

For example:

```
for(i, 1, 10, println(i));
```

Normal Functions vs. Special Functions

In a call of the normal function, the parameters will be evaluated in the order of parameters. The special functions can change the order of expression evaluations and can ignore some parameters.

Output

The `print`, `println`, `dump` and `dumpln` functions shows text of the specified values to the user.

Unlike `print` and `println`, the `dump` and `dumpln` functions add quotes and commas to the string values. `println` and `dumpln` add a newline at the end of the output text.

Samples

Sample `break.ega`

```
for(i, 1, 10000, (println(i), if(>=(i, 10), break())));
```

This EGA program will output `1 -to- 10` . Output:

```
1
2
3
4
5
6
7
8
9
10
```

The `break` special function breaks the `for` loop.

Sample `fact.ega`

```
define(fact, do(set(prod, 1), for(i, 2, n, set(prod, *(prod, i)))));

for(k, 1, 12, (
    set(n, k),
    fact,
```

```

        println("n = ", n, ": fact == ", prod)
    )
)

```

This EGA program prints the factorial values for numbers from 1 to 12 . Output:

```

n = 1: fact == 1
n = 2: fact == 2
n = 3: fact == 6
n = 4: fact == 24
n = 5: fact == 120
n = 6: fact == 720
n = 7: fact == 5040
n = 8: fact == 40320
n = 9: fact == 362880
n = 10: fact == 3628800
n = 11: fact == 39916800
n = 12: fact == 479001600

```

The `define` special function defines a macro variable. This is the same as:

```

for(k, 1, 12, (
    set(n, k),
    do(set(prod, 1), for(i, 2, n, set(prod, *(prod, i)))),
    println("n = ", n, ": fact == ", prod)
)
)

```

The EGA Functions

The following sections are a list of the EGA functions.

EGA and Function

```

EGA function 'and':
arity: 2..32767
usage: and(value1, value2)

```

Calculates logical AND of two integers or more. Returns an integer.

Same as `&&` .

EGA array Function

```
EGA function 'array':  
  arity: 0..32767  
  usage: array(value1[, ...])
```

Creates an array from specified parameters. Returns an array.

EGA at Function

```
EGA function 'at':  
  arity: 2..3  
  usage: at(ary_or_str, index[, value])
```

Gets or sets the item at the specified index.

`ary_or_str` must be an array or a string.

If the value is not specified, the function gets the value at the position of the specified index.

If the value is specified, the function sets the value at the position of the specified index.

Returns the value.

Same as `[]` .

EGA binary Function

```
EGA function 'binary':  
  arity: 0..32767  
  usage: binary(string_or_byte[, ...])
```

Creates a binary string of the specified parameters. Each parameter is an integer or a string.

Returns a binary string.

EGA bitand Function

```
EGA function 'bitand':  
  arity: 2..32767  
  usage: bitand(value1, value2[, ...])
```

Calculates bitwise AND of two integers, or more. Returns an integer.

Same as `&` .

EGA bitor Function

```
EGA function 'bitor':  
  arity: 2..32767  
  usage: bitor(value1, value2[, ...])
```

Calculates bitwise OR of two integers, or more. Returns an integer.

Same as `|`.

EGA break Function

```
EGA function 'break':  
  arity: 0  
  usage: break()
```

Goes out of an EGA loop.

EGA cat Function

```
EGA function 'cat':  
  arity: 1..32767  
  usage: cat(ary_or_str_1, ary_or_str_2, ...)
```

Concatenates the specified arrays and/or strings. Returns an array or a string.

EGA compare Function

```
EGA function 'compare':  
  arity: 2  
  usage: compare(value1, value2)
```

Compares two values. Returns 0 if `value1` and `value2` are equal, `-(1)` if `value1` was less, or `1` if `value1` was greater.

EGA compl Function

```
EGA function 'compl':  
  arity: 1  
  usage: compl(value)
```

Calculates bitwise NOT. Returns an integer.

Same as `~` .

EGA `define` Function

```
EGA function 'define':  
  arity: 1..2  
  usage: define(var[, expr])
```

Defines an EGA macro variable. `var` is a variable. Unlike the `set` function, the `expr` argument will be not evaluated. If `expr` is omitted, `var` will be reset.

Returns `expr` .

Same as `:=` .

EGA `div` Function

```
EGA function 'div':  
  arity: 2  
  usage: div(int1, int2)
```

Divides an integer value `int1` by another integer value `int2` .

Returns an integer.

Same as `/` .

EGA `do` Function

```
EGA function 'do':  
  arity: 0..32767  
  usage: do(expr, ...)
```

Does loop while `expr` is non- 0 . The arguments will be evaluated in order.

Returns the last argument.

You can break the execution by `break` function.

EGA `dump` Function

```
EGA function 'dump':  
  arity: 0..32767  
  usage: dump(value, ...)
```

Outputs the values with quotations and commas if necessary. No return value.

EGA `dumpln` Function

```
EGA function 'dumpln':  
  arity: 0..32767  
  usage: dumpln(value, ...)
```

Same as `dump` except `dumpln` adds a newline.

Same as `?` .

EGA `equal` Function

```
EGA function 'equal':  
  arity: 2  
  usage: equal(value1, value2)
```

Compares two values. Returns `1` if two values are equal. `0` if not equal.

Same as `==` .

EGA `exit` Function

```
EGA function 'exit':  
  arity: 0..1  
  usage: exit([value])
```

Ends the program with a value.

EGA `find` Function

```
EGA function 'find':  
  arity: 2  
  usage: find(ary_or_str, target)
```

Finds a target value from an array or a string.

Returns the `0` -based offset of the found target. Returns `-(1)` if not found.

EGA `for` Function

```
EGA function 'for':  
  arity: 4  
  usage: for(var, min, max, expr)
```

Does loop from `min` to `max` (inclusive).

The `expr` argument will be evaluated repeatedly. The `min` and `max` values must be integers. The `var` is the name of a loop variable.

1. First, `min` is stored into the `var` variable.
2. `expr` is evaluated.
3. `var` is incremented by 1.
4. If `var` is less than or equal to `max`, go back to step 2.

You can break the loop by `break` function.

EGA foreach Function

```
EGA function 'foreach':  
  arity: 3  
  usage: foreach(var, ary, expr)
```

Does loop using an array. `ary` is an array. The item in the `ary` array will be evaluated and stored into variable `var` repeatedly. You can break the loop by `break` function.

EGA gmtime Function

```
EGA function 'gmtime':  
  arity: 0  
  usage: gmtime()
```

Returns the UTC date/time string like `YYYY-MM-DD hh:mm:ss`.

EGA greater Function

```
EGA function 'greater':  
  arity: 2  
  usage: greater(value1, value2)
```

Compares two values. Returns `1` if `value1` was greater than `value2`. `0` if not greater.

Same as `>`.

EGA `greater_equal` Function

```
EGA function 'greater_equal':  
  arity: 2  
  usage: greater_equal(value1, value2)
```

Compares two values. Returns `1` if `value1` was greater than `value2` or equal. Otherwise returns `0`.

Same as `>=`.

EGA `hex` Function

```
EGA function 'hex':  
  arity: 1  
  usage: hex(value)
```

Converts an integer value to a hexadecimal string.

Returns a string.

EGA `if` Function

```
EGA function 'if':  
  arity: 2..3  
  usage: if(cond, true_case[, false_case])
```

Chooses the execution by the condition.

If the integer value `cond` was non-`0`, then `true_case` will be evaluated. If `cond` was `0`, then `false_case` will be evaluated if any.

Returns the evaluated value of `true_case` or `false_case`.

EGA `int` Function

```
EGA function 'int':  
  arity: 1  
  usage: int(value)
```

Converts a value to an integer value.

Returns an integer.

EGA `left` Function

```
EGA function 'left':  
  arity: 2  
  usage: left(ary_or_str, count)
```

Returns an array or a string of `count` items at the left side of an array or a string.

EGA `len` Function

```
EGA function 'len':  
  arity: 1  
  usage: len(ary_or_str)
```

Returns the length of an array or a string.

EGA `less` Function

```
EGA function 'less':  
  arity: 2  
  usage: less(value1, value2)
```

Compares two values. Returns 1 if `value1` was less than `value2` . 0 if not less.

Same as `<` .

EGA `less_equal` Function

```
EGA function 'less_equal':  
  arity: 2  
  usage: less_equal(value1, value2)
```

Compares two values. Returns 1 if `value1` was less than `value2` or equal. Otherwise returns 0 .

Same as `<=` .

EGA `load` Function

```
EGA function 'load':  
  arity: 1  
  usage: load(filename)
```

Loads the file contents.

Returns the binary string or 0 .

EGA localtime Function

```
EGA function 'localtime':  
  arity: 0  
  usage: localtime()
```

Returns the local date/time string like `YYYY-MM-DD hh:mm:ss` .

EGA mid Function

```
EGA function 'mid':  
  arity: 3..4  
  usage: mid(ary_or_str, index, count[, value])
```

Returns the sequence of the specified range of an array or a string. `ary_or_str` must be an array or a string.

The range starts from offset `index` . The length of the range is `count` . If `value` is specified, the range will be replaced with a value of `value` .

EGA minus Function

```
EGA function 'minus':  
  arity: 1..2  
  usage: minus(int1[, int2])
```

Negates or subtract. `int1` and `int2` must be integers.

Returns an integer.

Same as `-` .

EGA mod Function

```
EGA function 'mod':  
  arity: 2  
  usage: mod(int1, int2)
```

Calculates the remainder of division of two integers. `int2` must be non- 0 .

Returns an integer.

Same as `%` .

EGA mul Function

```
EGA function 'mul':  
  arity: 2..32767  
  usage: mul(int1, int2, ...)
```

Calculates multiplication of two integers or more.

Returns an integer.

Same as `*` .

EGA not Function

```
EGA function 'not':  
  arity: 1  
  usage: not(value)
```

Calculates logical NOT of the value.

Returns an integer.

Same as `!` .

EGA not_equal Function

```
EGA function 'not_equal':  
  arity: 2  
  usage: not_equal(value1, value2)
```

Compares two values. Returns 1 if `value1` was different from `value2` . Otherwise returns `0` . Same as `!=` .

EGA or Function

```
EGA function 'or':  
  arity: 2..32767  
  usage: or(value1, value2, ...)
```

Calculates logical OR of two or more values.

Returns an integer.

Same as `||` .

EGA `plus` Function

```
EGA function 'plus':  
  arity: 1..32767  
  usage: plus(int1, ...)
```

Calculates sum of two integer values or more.

Returns an integer.

Same as `+`.

EGA `print` Function

```
EGA function 'print':  
  arity: 0..32767  
  usage: print(value, ...)
```

Outputs the values without quotation. No return value.

EGA `println` Function

```
EGA function 'println':  
  arity: 0..32767  
  usage: println(value, ...)
```

Outputs the values without quotation with a newline. No return value.

EGA `remove` Function

```
EGA function 'remove':  
  arity: 2  
  usage: remove(ary_or_str, target)
```

Returns an array or a string, whose parts are removed.

If `ary_or_str` is an array, the items with the same value as `target` are removed. If `ary_or_str` is a string, the substrings `target` are removed.

Returns the array or the string of the results.

This function doesn't change `ary_or_str`.

EGA `replace` Function

```
EGA function 'replace':  
  arity: 3  
  usage: replace(ary_or_str, from, to)
```

If `ary_or_str` is an array, every item with the same value as the `from` value are replaced with the `to` value. If `ary_or_str` is a string, the substrings `from` are replaced with the `to` string.

Returns the array or the string of the results.

This function doesn't change `ary_or_str`.

EGA right Function

```
EGA function 'right':  
  arity: 2  
  usage: right(ary_or_str, count)
```

Returns an array or a string of `count` items at the right side of an array or a string.

EGA save Function

```
EGA function 'save':  
  arity: 2  
  usage: save(filename, contents)
```

Writes a file with contents.

Returns `1` if file writing is successful. Returns `0` otherwise.

NOTE: For security reasons, RisohEditor EGA cannot output files with the extensions "exe", "dll", or "sys".

NOTE: For security reasons, RisohEditor EGA cannot write to system folders.

EGA set Function

```
EGA function 'set':  
  arity: 1..2  
  usage: set(var[, value])
```

Creates a variable whose value is `value`. If `value` is not specified, the variable is cleared.

Returns the value.

Same as `=` .

EGA `str` Function

```
EGA function 'str':  
  arity: 1  
  usage: str(value)
```

Converts the value to a string.

Returns a string.

EGA `typeid` Function

```
EGA function 'typeid':  
  arity: 1  
  usage: typeid(value)
```

Returns the type ID of the value.

If the value is NULL, then returns `-(1)` . If the value is an integer, then returns `0` . If the value is a string, then returns `1` . If the value is an array, then returns `2` .

EGA `u8fromu16` Function

```
EGA function 'u8fromu16':  
  arity: 1  
  usage: u8fromu16(utf16str)
```

Converts a UTF-16 string to a UTF-8 string.

NOTE: The EGA standard string is UTF-8. You can convert a UTF-8 string into a UTF-16 binary string by this function.

EGA `u16fromu8` Function

```
EGA function 'u16fromu8':  
  arity: 1  
  usage: u16fromu8(utf8str)
```

Converts a UTF-8 string to a UTF-16 string.

EGA `while` Function

```
EGA function 'while':
  arity: 2
  usage: while(cond, expr)
```

Does loop while the specified condition is non- 0 . The `expr` argument will be evaluated repeatedly. The `cond` is the condition.

1. At first `cond` will be evaluated. If it was 0 , then loop will be ended.
2. `expr` will be evaluated. Back to 1.

You can break the loop by `break` function.

EGA xor Function

```
EGA function 'xor':
  arity: 2..32767
  usage: xor(value1, value2, ...)
```

Calculates bitwise XOR of two integers, or more. Returns an integer.

Same as `^` .

RisohEditor EGA extension

RisohEditor EGA has the following functions as EGA extension:

- RES_clone_by_lang
- RES_clone_by_name
- RES_const
- RES_delete
- RES_extract
- RES_get_binary
- RES_get_text
- RES_load
- RES_save
- RES_search
- RES_select
- RES_set_binary
- RES_set_text
- RES_str_get
- RES_str_set

- RES_unload_res

EGA RES_clone_by_lang Function

```
EGA function 'RES_clone_by_lang':  
  arity: 4  
  usage: RES_clone_by_lang(type, name, src_lang, dest_lang)
```

Clones a resource item from one language to another language. `type` must be an integer or a string of a resource type. If `type` is `"*"`, then all resource types are searched. `name` must be an integer or a string of a resource name. If `name` is `"*"`, then all resource names are searched. `src_lang` must be an integer that specifies the source language ID. If `src_lang` is `-(1)`, then all resource languages are searched. `dest_lang` must be an integer that specifies the destination language ID.

Returns `1` if successfully cloned. Otherwise returns `0`.

EGA RES_clone_by_name Function

```
EGA function 'RES_clone_by_name':  
  arity: 3  
  usage: RES_clone_by_name(type, src_name, dest_name)
```

Clones a resource item and gives it a new name.

`type` must be an integer or a string of a resource type. If `type` is `"*"`, then all resource types are searched. `src_name` must be an integer or a string of a resource name. If `src_name` is `"*"`, then all resource names are searched. `dest_name` must be an integer or a string of a new resource name.

Returns `1` if successfully cloned. Otherwise returns `0`.

EGA RES_const Function

```
EGA function 'RES_const':  
  arity: 1  
  usage: RES_const(name)
```

The `RES_const` function queries the database for the value of a constant.

`name` must be a constant name. Returns the value if successful, otherwise `0`.

EGA RES_delete Function

```
EGA function 'RES_delete':  
  arity: 0..3
```

```
usage: RES_delete([type[, name[, lang]])
```

Deletes the resource items.

`type` must be an integer or a string of a resource type. If `type` is "*" or omitted, then search all resource types. `name` must be an integer or a string of a resource name. If `name` is "*" or omitted, then search all resource names. `lang` must be an integer that specifies the language ID. If `lang` is - (1) or omitted, then search all resource languages.

Returns 1 if deleted. Otherwise returns 0 .

EGA RES_extract Function

```
EGA function 'RES_extract':  
arity: 3..4  
usage: RES_extract(type, name, lang[, filename])
```

Extracts resource data.

`type` must be an integer or a string of a resource type. `name` must be an integer or a string of a resource name. `lang` must be an integer that specifies the language ID. `filename` must be a string to extract. If `filename` was omitted, a preferred filename will be used.

Returns the pathname of the extracted file if successful. Returns zero if failed.

EGA RES_get_binary Function

```
EGA function 'RES_get_binary':  
arity: 0..3  
usage: RES_get_binary([type[, name[, lang]])
```

`RES_get_binary` gets the binary data of the specified resource data.

Returns the binary string.

EGA RES_get_text Function

```
EGA function 'RES_get_text':  
arity: 3..3  
usage: RES_get_text(type, name, lang)
```

`RES_get_text` gets the text of the resource. If failed, returns an empty string.

EGA RES_load Function

```
EGA function 'RES_load':  
  arity: 1..2  
  usage: RES_load(filename[, options])
```

RES_load loads the resource file.

options is an empty string or "(no-load-res-h)" ;

EGA RES_save Function

```
EGA function 'RES_save':  
  arity: 1..2  
  usage: RES_save(filename[, options])
```

RES_save saves the resource file.

options is an empty string or the combinations of the following strings.

- "(idc-static)"
- "(compress)"
- "(sep-lang)"
- "(no-res-folder)"
- "(lang-macro)"
- "(less-comments)"
- "(wrap-manifest)"
- "(begin-end)"
- "(utf-16)"
- "(bom)"
- "(backup)"
- "(ms-msgtbl)"

For example: RES_save("C:\Users\katahiromz\Desktop\a.res", "(sep-lang)(compress)");

NOTE: For security reasons, RisohEditor EGA cannot output files with the extensions "exe", "dll", or "sys".

NOTE: For security reasons, RisohEditor EGA cannot write to system folders.

EGA RES_search Function

```
EGA function 'RES_search':  
  arity: 0..3
```

```
usage: RES_search([type[, name[, lang]])
```

`RES_search` returns an array of the resource items.

`type` must be an integer or a string of a resource type. If `type` is "*" or omitted, then search all resource types. `name` must be an integer or a string of a resource name. If `name` is "*" or omitted, then search all resource names. `lang` must be an integer that specifies the language ID. If `lang` is - (1) or omitted, then search all resource languages.

EGA RES_select Function

```
EGA function 'RES_select':  
arity: 0..3  
usage: RES_select([type[, name[, lang]])
```

`RES_select` selects an item on the RisoHEditor treeview.

Returns 1 if successful, 0 if failed.

EGA RES_set_binary Function

```
EGA function 'RES_set_binary':  
arity: 4  
usage: RES_set_binary(type, name, lang, binary)
```

`RES_set_binary` sets the binary data as the specified resource type, resource name, and language.

Returns 1 if successful, 0 if failed.

EGA RES_set_text Function

```
EGA function 'RES_set_text':  
arity: 4..4  
usage: RES_set_text(type, name, lang, text)
```

`RES_set_text` sets the text of the resource item and compiles the text.

Returns 1 if successful. Returns 0 if failed.

EGA RES_str_get Function

```
EGA function 'RES_str_get':  
arity: 1..2
```

```
usage: RES_str_get(lang[, str_id])
```

`RES_str_get` reads the resource string table. If `str_id` specified, then returns a UTF-8 string. If `str_id` not specified, then returns an array of pairs of string ID and text. If failed, returns an empty string or an empty array.

EGA `RES_str_set` Function

```
EGA function 'RES_str_set':  
arity: 2..3  
usage: RES_str_set(lang, str_id, str) or RES_str_set(lang, ary)
```

`RES_str_set` writes the resource string table. If `str_id` specified, then write a UTF-8 string to the string table. If the string was empty, then the resource string will be cleared. If `str_id` not specified, then set an array of pairs of string ID and text to the string table.

Returns `1` if successful. Returns `0` if failed.

EGA `RES_unload_res` Function

```
EGA function 'RES_unload_res':  
arity: 0  
usage: RES_unload_res()
```

`RES_unload_res` unloads the "resource.h" file.

Always returns `1`.

How can I extend EGA?

1. Import `libega`.
2. Include `ega.hpp`.
3. Call the following EGA C++ functions: `EGA_init`, `EGA_set_input_fn` and `EGA_set_print_fn`.
4. Add your EGA functions by `EGA_add_fn` C++ function.