# EVO-NG controller. Base unit.

## Embedded HTTP server

Встроенный в контроллер HTTP сервер обслуживает GET запросы для передачи статических файлов расположенных на карте памяти в папке 'html' и POST запросы для передачи данных в сервер и формирования динамических ответов, в зависимости от текущего состояния контроллера. В большинстве случаев данные в теле POST запроса и в ответе на него имеют JSON формат. Исключение составляют только запросы на обновление прошивки контроллера или запросы на запись файлов на карту памяти.

## Описание POST запросов и ответов на них

Далее приведены URI на которые необходимо отправить запрос, описание ожидаемого тела запроса и ответ сервера в случае успешной обработки запроса. В случае возникновения какой-либо ошибки или неверного запроса сервер отвечает пакетом следующего формата:

{"status": "error", "message": "Error description"}

### Login: /login.html

Request - user name and encrypted password:

{"login":"User Name", "password":"Password's MD5 hash"

Response - redirect to 'home.html'

### Get description list of connected devices: /api/devlist

Request - empty.

Response - array with descriptions of connected modules:

{"devices":[{

"addr":0, // MODBUS address of device ('0' for base module)

"type":"NGC", // "NGC" - base module,

// "NGBO" - relay module,

// "NGI" - inputs extension module

// "NGO" - analog outputs module

"inp\_bin":12, // number of binary inputs present in module

"inp\_adc":4, // number of ADCs present in module

"inp\_btn":32, // maximum supported number of 1-wire buttons

// (DS2401 or DS2413)

"inp\_key":32, // maximum supported number of 1-wire keys (DS1990A)

"inp\_tmp":16, // maximum supported number of 1-wire thermometers

// (DS18B20)

"out\_bin":12, // number of binary outputs (open collector) present

// in module

"out\_rly":0, // number of relays present in module

"out\_pwm":4, // number of PWM outputs present in module

"out\_dac":0 // number of analog outputs present in module

},

{...}

]}

### Get current IO state: /api/state

Request:

{

"device":0 // N - extension module address or '0' for base module

}

Response:

{

"inp\_bin":[0, 1, ...], // binary input states: 0 - open, 1 - closed

"inp\_adc":[0, 1000, ...], // analog input states in percent multiplied by 10:

// 0 -> 0%, 456 -> 45.6%, 1000 -> 100%

"inp\_btn":[ // 1-wire input buttons state. for DS2413:

{"000000000000":[0, 0]}, // "1-wire address":[in\_state, out\_state]

{"FFFFFFFFFFFF":[1, 0]}, // for DS2401:

... // "1-wire address":[in\_state, 0]

],

"inp\_key":[ // array with addresses of present on 1-wire bus

"000000000000", // keys (DS1990A)

"FFFFFFFFFFFF",

...

],

"inp\_tmp":[

{"000000000000":366}, // array with data from 1-wire thermometers in

{"FFFFFFFFFFFF":-103}, // degrees multiplied by 10:

... // 366 -> 36.6°C, -103 -> -10.3°C

],

"out\_bin":[0, 1, ...], // binary outputs (open collector) state:

// 0 - open, 1 - closed

"out\_rly":[0, 1, ...], // relay outputs state (only for 'NGBO'):

// 0 - open, 1 - closed

"out\_pwm":[0, 1000, ...], // PWM outputs state in percent multiplied by 10

// 0 -> 0%, 234 -> 23.4%, 1000 -> 100%

"out\_dac":[0, 1000, ...] // analog outputs state (only for 'NGAO')

// in percent multiplied by 10

}

### Set binary output state: /api/out\_bin

Request:

{

"device":0, // extension module address or '0' for base module

"output":1, // binary (open collector) output number

"value":1 // 0 - set to open state, 1 - set to closed state

}

Response:

{

"out\_bin":[0, 1, ...] // binary outputs state

}

### Set relay state: /api/out\_rly

Request:

{

"device":0, // extension module address or '0' for base module

"output":1, // relay output number

"value":1 // 0 - set to open state, 1 - set to closed state

}

Response:

{

"out\_rly":[0, 1, ...] // relay outputs state

}

### Set PWM output state: /api/out\_pwm

Request:

{

"device":0, // extension module address or '0' for base module

"output":1, // PWM output number

"value":500 // 0 -> 0%, 526 -> 52.6%, 1000 -> 100%

}

Response:

{

"out\_pwm":[0, 1000, ...] // PWM outputs state

}

### Set analog output state: /api/out\_dac

Request:

{

"device":0, // extension module address or '0' for base module

"output":1, // analog output number

"value":500 // 0 -> 0%, 526 -> 52.6%, 1000 -> 100%

}

Response:

{

"out\_dac":[0, 1000, ...] // analog outputs state

}

### Get the last modification time: /api/last\_mod

Request:

{

"device":0 // extension module address or '0' for base module

}

Response - the last modification time of each input/output:

{

"inp\_bin":[

1595308925, // 32-bit UNIX timestamp (the number of seconds

1595308929, // since 00:00 hours, Jan 1, 1970 UTC) represents the time

... // when the state of binary input changed the last time

],

"inp\_adc":[

1595308925,

1595308925,

...

],

"inp\_btn":[ // "id":"time", where 'time' is timestamp

{"000000000000":1595308925}, // represents the time when DS2401 was on bus

{"FFFFFFFFFFFF":1595308925}, // or DS2413 input or output was changed

...

],

"inp\_key":[

{"000000000000":1595308925}, // "id":"time", where 'time' is the timestamp

{"FFFFFFFFFFFF":1595308925}, // represents the time when 1-wire key with

... // 'id' was last present on the bus

],

"inp\_tmp":[

{"000000000000":1595308925}, // "id":"time", where 'time' is the timestamp

{"000000000000":1595308925}, // represents the time when value measured by

... // thermometer with 'id' changed the last time

],

"out\_bin":[

1595308925,

1595308925,

...

],

"out\_rly":[

1595308925,

1595308925,

...

],

"out\_pwm":[

1595308925,

1595308925,

...

],

"out\_dac":[

1595308925,

1595308925,

...

]

}

### Set user name and password: /api/set\_user

Request:

{

"username":"User Name",

"password":"md5"

}

Response:

{

"status":"OK"

}

### Get user name: /api/get\_user

Request - empty

Response:

{

"username":"User Name"

}

### Get list of functions: /api/get\_flist

Request - empty

Response:

{

"func\_names":[

"User defined name of function number 1",

"User defined name of function number 2"

...

]

}

### Object descriptor

To describe input, output, time or internal variable the next declarations are used in function definition:

// for binary and analog inputs or outputs

{

"type":"ibin", // 'ibin' - binary input

// 'iadc' - analog output

// 'obin' - binary output

// 'orly' - relay output

// 'opwm' - PWM output

"device":0, // address of extension module or '0' for base module

"addr":1 // number of input or output

}

// for 1-wire buttons, keys or thermometers

{

"type":"ibtn", // 'ibtn' - 1-wire buttons (DS2401 or DS2413)

// 'ikey' - 1-wire keys (DS1990A)

// 'itmp' - 1-wire thermometers (DS18B20)

"device":0, // address of extension module or '0' for base module

"addr":"0123456789AB" // 1-wire address

}

// for internal variables

{

"type":"var", // internal variable

"addr":1 // number (name) of internal variable

}

// for time

{

"type":"time" // time

}

### Get function description: /api/get\_func

Request:

{

"function":1 // index of function to get

}

Response:

{

/\* TRIGGER \*/

"trigger":

// for binary IO, relays and 1-wire buttons and keys

{

"object":{...}, // see 'Object descriptor'

"event":">" // '>' - trigger on the front edge

// '<' - trigger on the fall edge

}

// for analog IO, PWM outputs, 1-wire thermometers and internal variables

{

"object":{...}, // see 'Object descriptor'

"event":">", // '>' - triggered when the value at input rises above

// the threshold for time longer then defined one

// '<' - trig-gered when the value at input falls below

// the threshold for time longer then time

"thre":1234, // threshold in %\*10

"time":1000 // time in milliseconds

}

// or

{

"object":{...}, // see 'Object descriptor'

"event":"><", // '><' - triggered when the level at input enters

// the interval for time longer than defined one

// '<>' - triggered when the level at input leaves

// the interval for time longer than defined one

"max":1234, // upper limit

"min":65, // lower limit

"time":10000 // time in milliseconds

}

// for time

{

"object":{...}, // see 'Object descriptor'

"event":"at", // 'at' - at exact time

// 'period' - periodically (**not supported yet**)

"time":1595308925, // timestamp

},

/\* LOGIC \*/

"logic":"and", // 'and' - triggered if all the conditions is true

// 'or' - triggered if at least one condition is true

/\* CONDITIONS \*/

"conditions":[

// for binary IO, relays and 1-wire buttons and keys

{

"object":{...}, // see 'Object descriptor'

"cond":1 // 1 - condition is true if input is closed

// 0 - condition is true if input is open

}

// for analog IO, 1-wire thermometers and internal variables

{

"object":{...}, // see 'Object descriptor'

"cond":"<", // condition is true if the level at input is

// '<' - less than 'value'

// '>' - more than 'value'

// '>=' - more than 'value' or equal to 'value'

// '<=' - less than 'value' or equal to 'value'

// '==' - equal to 'value'

// '!=' - not equal to 'value'

"value":1234 // comparison value

}

// for time intervals

{

... // not supported yet

}

...

],

/\* ACTIONS \*/

"actions":[

// for binary and relay outputs and DS2413 outputs

{

"object":{...}, // see 'Object descriptor'

"act":"set", // 'set' - set output state to 1

// 'reset' - set output state to 0

// 'inv' - invert current output state

"delay":5000, // delay between the triggering and starting the action

"time":15000 // action time

}

// or

{

"object":{...}, // see 'Object descriptor'

"act":"copy", // repeat for the other object

"source":{...}, // object to repeat (see 'Object descriptor')

"delay":5000, // delay between the triggering and starting the action

"time":15000 // action time

}

// for analog/PWM outputs and internal variables

{

"object":{...}, // see 'Object descriptor'

"act":"set", // 'set' - set output value to the certain value

// 'mod' - modify output value by the certain value

"value":1234, // value to set to/modify by

"delay":5000, // delay between the triggering and starting the action

"time":15000 // action time

}

// or

{

"object":{...}, // see 'Object descriptor'

"act":"change", // smoothly change the current value to the certain value

// in certain interval

"value":1234, // value to set

"intr":1000, // interval to change

"delay":5000, // delay between the triggering and starting the action

"time":15000 // action time

}

// or

{

"object":{...}, // see 'Object descriptor'

"act":"copy", // repeat for the other object

"source":{...}, // object to repeat (see 'Object descriptor')

"delay":5000, // delay between the triggering and starting the action

"time":15000 // action time

},

]

}

### Add new function: /api/add\_func

Request :

{

"fname":"User defined function name",

"trigger":{...}, // as defined in 'get\_func'

"logic":"and", // as defined in 'get\_func'

"conditions":[{...}, {...}, ...], // as defined in 'get\_func'

"actions":[{...}, {...}, ...] // as defined in 'get\_func'

}

Response:

{

"func":N // index of the newly added function

}

### Delete function: /api/del\_func

Request:

{

"function":1 // index of function to delete

}

Response - array of reminded functions names (will be empty if the last function is deleted):

{

"func\_names":[

"User defined name of function number 1",

"User defined name of function number 2"

...

]

}

### Get parameters: /api/get\_param

Request:

{

"device":0 // extension module address or '0' for base module

}

Response:

{

// embedded HTTP server settings (only for base module)

"ip":{

"address":192.168.0.26,

"mask":255.255.255.0,

"gate":192.168.0.1

},

// MODBUS settings

"mbus":{

"rate":38400,

"address":0

},

// analog inputs mode: 'v' - voltage, 'i' - current

"iadc":["v", "i", "v", "i"],

// GSM modem and HTTP client settings (only for base module)

"gsm":{

"server":"http://controller.evo.by/api.php",

"port":80,

"interval":10000,

"apn":"",

"usr":"",

"pwd":"",

"mode":"auto" or "Cat-M" or "NB-IoT"

},

// embedded real time clock correction source (only for base module)

"rtc":"none" or "gsm" or "eth" or "srv"

}

### Set parameters: /api/set\_param

Request:

{

// extension module address or '0' for base module

"device":0,

// embedded HTTP server settings (only for base module)

"ip":{

"address":192.168.0.26,

"mask":255.255.255.0,

"gate":192.168.0.1

},

// MODBUS settings (for base module address must be 0)

"mbus":{

"rate":38400,

"address":0

},

// analog inputs mode: 'v' - voltage, 'i' - current

"iadc":["v", "i", "v", "i"],

// GSM modem and HTTP client settings (only for base module)

"gsm":{

"server":"http://controller.evo.by/api.php",

"port":80,

"interval":10000,

"apn":"",

"usr":"",

"pwd":"",

"mode":"auto" or "Cat-M" or "NB-IoT"

},

// embedded real time clock correction source (only for base module)

"rtc":"none" or "gsm" or "eth" or "srv"

}

to be continued...

Прошу подвтердить, что данный заказ ограничен только функционалом формирования и обработки получения json на странице https://alpachini.com/function.html в соответствии с api (прилагается)