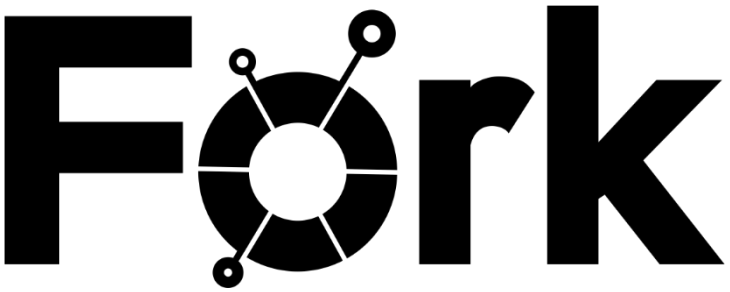

Programming Guide

Revised May 18, 2021

Table of contents

Abbreviations

DI – digital input

DO – digital output

AI – analog input

AO – analog output

I - digital interface

– channel number of input/output pin

PWM – pulse width modulation

FREQ – frequency

DUTY – duty cycle

CNT – count

Introduction

Fork API uses standard GET queries to ease human readability and support standard internet browsers like Firefox or Chrome.

Fork API also uses POST queries to implement JSON format for requests.

API commands are case insensitive.

You can access periphery one by one

"http://192.168.0.205/control?DI2"

"http://192.168.0.205/control?AI5"

"http://192.168.0.205/control?DO1=1"

or make multiple request at a time

"http://192.168.0.205/control?DI2&AI5&DO1=1"

in case of multiple request by default you`ll get plain text comma separated results, like "0,3.300,1"

Limitations

The I2C Slave function does not work well with the DO PWM COUNTER, DI COUNTER, UART RX functions.

The I2C sniffer function does not work well with the DO PWM COUNTER, DI COUNTER, UART RX functions.

When Stream works the other functions of Fork are not available.

Requests and response formats

Requests and responses can be in three formats URL_ENCODE, ASCII_HEX or JSON.

GET requests

Requests can be sent in two formats URL_ENCODE or ASCII_HEX. Format URL_ENCODE is used by default.

POST requests

Requests are sent only in JSON format.

Responses

Replies can come in three formats URL_ENCODE, ASCII_HEX or JSON. Format URL_ENCODE is used by default.

Commands G_REQ_FORM, G_RES_FORM and G_REQ_RES_FORM

To use different formats in the Fork API, there are three commands G_REQ_FORM, G_RES_FORM, G_REQ_RES_FORM and four options for their use. In all cases, they must go at the beginning of the request. If no command is used, the default format will be used.

1. Command G_REQ_FORM sets the format of the request. Used for GET requests
2. Command G_RES_FORM sets the format of the response. It is used for GET and POST requests
3. Command G_REQ_RES_FORM sets the same format for response and request. Used for GET requests

Note. Commands G_RES_FORM, G_REQ_FORM must be send only in this order

URL_ENCODE format

URL_ENCODE also known as Percent-encoding is required to send any data using ASCII characters. Binary or text data is represented either directly as an ASCII character or percent-encoded. A set of so-called reserved characters is also encoded. Symbols are divided into *reserved* and *unreserved*.

Reserved characters:

!	#	$	&	'	(	)	*	+	,	/	:	;	=	?	@	[	]

Unreserved characters:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	1	2	3	4	5	6	7	8	9	-	_	.	~												

Example URL_ENCODE

Request
G_RES_FORM=URL_ENCODE&AI_ALL&I_UART1&I_SPI_POLARITY
Response
1.140, 0.140, 1.000, 1.140, 1.540, 5.540,-5.540,-3.330,%AD%CF8%F7%D0%12,LOW

ASCII_HEX format

In ASCII_HEX format, all data is considered as binary. Each byte is encoded with two characters. If the value is an integer, it is presented as 1-4 bytes in big endian byte order. If the value is float, then it is encoded in four bytes as a single precision floating point number according to the IEEE 754 standard, the byte order is big endian. Otherwise, use the reference information.

Example ASCII_HEX

Request
G_REQ_RES_FORM=ASCII_HEX&AI_ALL&I_UART1&I_SPI_POLARITY&I_I2C_WRITE=ADCF56F7D012
Response
3F91EB85,3E0F5C29,3F800000,3F91EB85,3FC51EB8,40B147AE,C0B147AE,C0551EB8,BBBBD0,00,ADCF56F7D012

JSON format

In JSON format, binary data is encoded in base64 format. The POST method is used to send a request in JSON format.

Request
{"i_UART1" : "qq3avhgjvhgjvhgjvhgjvhrd0=""i_UART2", "DO0" : "1"}
Response
%AA%AD%DA%BE%18%23%BE%18%23%BE%18%23%BE%18%23%BE%1A%DD,%AA%AD%DA%BE%18%23%BE%18%23%BE%18%23%BE%18%23%BE%1A%DD,1

Request
{"G_RES_FORM":"JSON","i_UART1" : "qq3avhgjvhgjvhgjvhgjvhrd0=""i_UART2", "DO0" : "1"}
Response
{"G_RES_FORM":"JSON","I_UART1":"qq3avhgjvhgjvhgjvhgjvhrd","I_UART2":"qq3avhgjvhgjvhgjvhgjvhrd","DO0":1}

I2C Slave mode

In this mode, Fork can emulate an I2C slave device with up to 16 registers. Registers are available for reading and writing via the I2C interface. Registers are emulated in the memory area. Registers can be arranged in memory in any order and have any I2C address. Note that memory addresses and register addresses may not match. By register addresses, any I2C Master device can read data from them. In registers, you can also specify the size of the data that each of them stores (up to Registers memory size).
Register addresses are specified as one byte.
Memory size 200 bytes.
Also see **Limitations.**

Example
Set the address 64 of the Fork on the I2C bus
"http://192.168.0.205/control?I_I2C_SLAVE_ADDR=64" - "64"

Creating register 11 with I2C register address 4, on memory address 30 with 4 bytes size
"http://192.168.0.205/control?I_I2C_CREATE_REG=11,4,30,4" - "11,4,30,4"

Write data to register.

"http://192.168.0.205/control?I_I2C_REG=11,fgt%10" - "fgt%10"

I2C Sniffer mode

String format of sniffed data

Every byte is presented in hex format. After every byte follows “a”, “n” or “e” characters indicating ACK, NACK or ERROR state of byte, respectively. Also after an error character “e” follows one digit, which indicates how many bits received before error occurred.
“Sw” or “Sr” indicates I2C package Start, where “w” or “r” indicates write or read command.
“P” indicates Stop of the package.
Square brackets “[“ and “]” indicates complete packet from first start state in packet to stop state.
Sniffer buffer can store up to 1024 data bytes.
Also see **Limitations.**
Example
[Sw A0a 08a Sr A1a A5n P]

Byte format of sniffed data

Each received byte of data is described by two bytes. First byte stores status of received byte, second byte store data.

b15	b14	b13	b12	b11	b10	b9	b8
Bit count				WRITE/ READ	STOP	START	ACK
b7	b6	b5	b4	b3	b2	b1	b0
Data byte							

Bit count - how many bits were received. For normal states it equals 9, 8 bits of data + 1 bit ACK.
Write/Read - This bit indicates write (0) or read (1) request. This bit is set, if the Data byte is first after Start state, also meaning that 7 bits of data are an address.
Stop - is set to 1 if the byte is followed by the Stop state.
Start - is set to 1 if the byte is preceded by the Start state.
ACK - is set to 0 if the receiving device sets ACK on I2C bus and 8 data bits are received.

Example
92A090089AA19000905590559555

Digital Input (DI) commands

Command	Description	Input value	Return value
DI#	Reads DI channel # value	-	0/1
DI_ALL	Reads all DI values	-	array 0/1
DI#_PULLUP	Turns on/off DI channel # pullup	0/1	0/1
DI#_PULLUP	Reads DI channel # pullup status	-	0/1
DI_PULLUP_ALL	Reads all DI pullup status	-	array 0/1
DI#_CNT_START	Start counter on DI channel, return value of counter on DI channel	-	0..9999999
DI#_CNT	Return value of counter on DI channel	-	0..9999999
DI#_STOP	Stop counter on DI channel, return value of counter on DI channel	-	0..9999999
DI#_CNT_RESET	Reset counter on DI channel, return value of counter on DI channel	-	0

Examples:

"http://192.168.0.205/control?DI1" – "0"

"http://192.168.0.205/control?DI_ALL" – "0,1,0,0,1,1,0,0"

"http://192.168.0.205/control?DI2_PULLUP" – "1"

"http://192.168.0.205/control?DI4_PULLUP=1" – "1"

"http://192.168.0.205/control?DI1_CNT_START" – "0"

"http://192.168.0.205/control?DI1_CNT" – "345"

"http://192.168.0.205/control?DI1_STOP" – "40"

"http://192.168.0.205/control?DI1_CNT_RESET" – "0"

Digital Output (DO) commands

Command	Description	Input value	Return value
DO#	Writes DO channel # value	0/1	0/1
DO#	Reads DO channel # value	-	0/1
DO_ALL	Reads all DO values	-	array 0/1
DO#_PWM_EN	Start PWM on DO channel #	0/1	0/1
DO#_PWM DO#_PWM_DUTY	Set PWM duty cycle of DO channel #	0 .. 100	0 .. 100
DO#_PWM_FREQ	Set PWM frequency of DO channel #	0 .. 100000 1k .. 100k	0 .. 100000
DO_PWM_ALL	Reads all DO_PWM values. Response consists of three arrays. First array is flags, enabled/disabled PWM. Second array is frequencies. Third array is duty cycles	-	array 0/1, array 0 .. 100000, array 0 .. 100
DO#_PWM_CNT	Set count of impulses and start PWM	0 .. 1000	0 .. 1000
Available in Fork hardware ver. 3.0			
DO_OPENDRAIN	Set open drain for DO port	0/1	0/1
DO_OPENDRAIN	Read state open drain setting of DO port	-	0/1

Examples:

"http://192.168.0.205/control?DO1" – "0"

"http://192.168.0.205/control?DO2=1" – "1"

"http://192.168.0.205/control?DO_ALL" – "0,1,0,0,1,1,0,0"

"http://192.168.0.205/control?DO2_PULLUP" – "1"

"http://192.168.0.205/control?DO4_PULLUP=1" – "1"

"http://192.168.0.205/control?DO1_PWM=23" – "23"

"http://192.168.0.205/control?DO1_PWM_FREQ=100" – "100"

"http://192.168.0.205/control?DO1_PWM_FREQ=1k" – "1000"

"http://192.168.0.205/control?DO1_PWM_ALL" – "1,0,0,0,0,0,0,0,1000,0,0,0,0,0,0,0,23,0,0,0,0,0,0,0"

"http://192.168.0.205/control?DO1_PWM_CNT=15" – "15"

"http://192.168.0.205/control?DO_OPENDRAIN=1" – "1"

Analog Input (AI) commands

Command	Description	Input value	Return value
AI#	Reads AI channel # value	-	-10.000 .. 10.000
AI_ALL	Reads all AI values	-	array -10.000 .. 10.000
AI#_TRIG	Set the source of trigger signal. DI channels can be used or “0” to turn trigger off	0/DI0..DI7	0/DI0..DI7
AI#_TRIG	Reads the value of AI channel captured on last trigger signal. Every triggering overwrites previous value. Getting “AI# wait” means no data was still captured.	-	-10.000 .. 10.000 or AI# wait
AI#_TRIG_POLARITY	Select polarity of trigger on rising or falling edge	0_1/1_0	0_1/1_0
AI#_SINGLE_TRIG	Set the source of single trigger signal. For trigger uses DI channels	0/DI0..DI7	0/DI0..DI7
AI#_SINGLE_TRIG	Reads the value of AI channel captured on first trigger signal. Getting “AI# wait” means no data was still captured.	-	-10.000 .. 10.000 or AI# wait
AI#_SINGLE_TRIG_W	Set the source of trigger signal. For trigger uses DI channels. This type of trigger uses property of HTTP, that can send reply after some time. You don't need to poll AI value like AI#_trig or AI#_single_trig, but you cannot send another requests until trigger or timeout occurs. Timeout is 30s.	DI0..DI7	-10.000 .. 10.000
AI#_FILTER	Turn on/off filtering values of AI channel.	0/1	0/1
Available in Fork hardware ver. 3.0			
AI_DIFF	Read differential input	-	-1.500 .. 1.500

Examples:

"http://192.168.0.205/control?AI3" – "2.8"

"http://192.168.0.205/control?AI_ALL" – "2.34,1.34,6.01,2.04,2.34,1.34,6.01,2.04"

"http://192.168.0.205/control?AI3_trig=DI4" – "DI4"

"http://192.168.0.205/control?AI3_trig" – "2.34"

"http://192.168.0.205/control?AI3_trig_polarity=0_1" – "0_1"

"http://192.168.0.205/control?AI3_single_trig=DI4" – "DI4"

"http://192.168.0.205/control?AI3_single_trig" – "2.34"

"http://192.168.0.205/control?AI3_single_trig_w=DI4" – "2.34"

"http://192.168.0.205/control?AI3_filter=1" – "1"

Analog Output (AO) commands

Command	Description	Input value	Return value
AO#	Set value of AO channel	0.000 .. 10.000	0.000 .. 10.000
AO#	Reads value of AO channel	-	0.000 .. 10.000
AO_ALL	Read all values of AO channels	-	array 0.000 .. 10.000
AO#_GEN_EN	Enable/disable generator on AO channel	0/1	0/1
AO#_GEN_FREQ	Set frequency of signal on AO channel	0..100000	0..100000
AO#_GEN_DC	Set DC level of signal on AO channel	0.000 .. 10.000	0.000 .. 10.000
AO#_GEN_AMPLITUDE	Set amplitude of signal on AO channel	0.000 .. 10.000	0.000 .. 10.000
AO#_GEN_MIN	Set min value of signal, changes the parameters DC and amp, respectively	0.000 .. 10.000	0.000 .. 10.000
AO#_GEN_MAX	Set max value of signal, changes the parameters DC and amp, respectively	0.000 .. 10.000	0.000 .. 10.000
AO#_GEN_PHASE	Set phase of signal, when starting generator	0 .. 359	0 .. 359
AO#_GEN_PHASE_2	Set phase shift of AO1 relative AO0 and start generator AO0 and AO1	0 .. 359	0 .. 359
AO#_GEN_CNT	Set count of periods, which generates on AO channel	1 .. 1000	1 .. 1000
AO_GEN_ALL	Get settings about each generator in next order: enable/disable, mode, frequency, dc level, amplitude	-	
AO#_MODE_SIN	Set sine wave form on AO channel	-	SIN
AO#_MODE_TRI	Set triangle wave form on AO channel	-	TRI
AO#_MODE_SAW	Set saw wave form on AO channel	-	SAW
AO#_MODE_FFG	Set free-form generator on AO channel	-	FFG

Examples:

"http://192.168.0.205/control?AO0=2.8" – "2.8"

"http://192.168.0.205/control?AO0" – "2.8"

"http://192.168.0.205/control?AO_ALL" – "2.8,3.1"

"http://192.168.0.205/control?AO0_GEN_EN=1" – "1"

"http://192.168.0.205/control?AO0_GEN_FREQ=1000" – "1000"

"http://192.168.0.205/control?AO0_GEN_DC=3.3" – "3.3"

"http://192.168.0.205/control?AO0_GEN_AMPLITUDE=0.1" – "0.1"

"http://192.168.0.205/control?AO0_GEN_MIN=2" – "2"

"http://192.168.0.205/control?AO1_GEN_MAX=7" – "7"

"http://192.168.0.205/control?AO1_GEN_PHASE=60" – "60"

"http://192.168.0.205/control?AO0_GEN_PHASE_2=90" – "90"

"http://192.168.0.205/control?AO0_GEN_CNT=50" – "50"

"http://192.168.0.205/control?AO_GEN_ALL" – "0,SIN,1000,5.000,5.000,0,SIN,1000,5.000,5.000"

"http://192.168.0.205/control?AO1_MODE_SIN" – "SIN"

Digital Interface Lines (I) commands

On the Fork, you cannot turn on I2C and UART2 or UART3 at the same time.
On the Fork, you cannot turn on SPI and UART1 or UART4 at the same time.

Command	Description	Input value	Return value
I_MODE ***Deprecated***	Set interface mode. Be careful when setting interface mode and look at the pinout.	UART# UART#_RX UART#_TX SPI I2C	UART1 UART2 UART3 UART4 SPI I2C
I_UART#_EN	Enable or disable RX and TX lines of UART	0/1	0/1,0/1
I_UART#_EN	Read state of UART RX and TX	-	0/1,0/1
I_UART#TX_EN	Enable or disable TX lines of UART	0/1	0/1
I_UART#TX_EN	Read state of UART TX	-	0/1
I_UART#RX_EN	Enable or disable RX lines of UART	0/1	0/1
I_UART#RX_EN	Read state of UART RX	-	0/1
I_SPI_EN	Enable or disable SPI	0/1	0/1
I_SPI_EN	Read state of SPI	-	0/1
I_I2C_EN	Enable or disable I2C	0/1	0/1
I_I2C_EN	Read state of I2C	-	0/1

Examples:

"http://192.168.0.205/control?I_MODE=UART1" – "UART1"

"http://192.168.0.205/control?I_UART1_EN=1" - "1"

"http://192.168.0.205/control?I_UART1_EN" - "1"

"http://192.168.0.205/control?I_SPI_EN=1" - "1"

"http://192.168.0.205/control?I_I2C_EN" - "0"

UART commands

Command	Description	Input value	Return value
I_UART#	Send packet of bytes. For send non-printable bytes use percent-encoding format.	string and url-encoded bytes	string and url-encoded bytes
I_UART#	Read uart RX buffer. Non-prinatable bytes are shown as url-encoded bytes.	-	string and url-encoded bytes
I_UART#_ RECVSIZE	Get size of received bytes in rx buffer	-	0..512
I_UART#_BAUDRATE	Set baudrate of uart# (bits/s)	1200 .. 1000000	1200 .. 1000000
I_UART#_BAUDRATE	Read baudrate of uart# (bits/s)	-	1200 .. 1000000
I_UART#_STOPBITS	Set stopbits of uart#	1/2	1/2
I_UART#_STOPBITS	Read stopbits of uart#	-	1/2

Examples:

"http://192.168.0.205/control?I_MODE=UART1" – "UART1"

"http://192.168.0.205/control?I_UART1=rate28" – "rate28"

"http://192.168.0.205/control?I_UART1=%04%24%03%06" – "%04%24%03%06"

"http://192.168.0.205/control?I_UART1" – "Door closed."

"http://192.168.0.205/control?I_UART1" – "axf%24%03%06"

"http://192.168.0.205/control?I_UART1_BAUDRATE=115200" – "115200"

"http://192.168.0.205/control?I_UART1_STOPBITS=1" – "1"

SPI Master commands

Command	Description	Input value	Return value
I_SPI_WRITE	Send packet of bytes. For send non-printable bytes use percent-encoding format.	string and url-encoded bytes	string and url-encoded bytes
I_SPI_READ	Read packet of bytes. Non-prinatable bytes are shown as url-encoded bytes.	0..512	string and url-encoded bytes
I_SPI_WRITE_READ	Read while write. Count of reading bytes equal count of writing bytes.	string and url-encoded bytes	string and url-encoded bytes
I_SPI_BAUDRATE	Set baudrate for SPI	175781, 351563, 703125, 1406250, 2812500, 5625000, 11250000, 22500000	175781, 351563, 703125, 1406250, 2812500, 5625000, 11250000, 22500000
I_SPI_BAUDRATE	Read baudrate of SPI	-	175781, 351563, 703125, 1406250, 2812500, 5625000, 11250000, 22500000
I_SPI_DATASIZE	Set size of write/read data	8/16	8/16
I_SPI_DATASIZE	Read size of write/read data	-	8/16
I_SPI_POLARITY	Set clock polarity of SPI	low/high	low/high
I_SPI_POLARITY	Read clock polarity of SPI	-	low/high
I_SPI_PHASE	Set clock phase of SPI	1/2	1/2
I_SPI_PHASE	Read clock phase of SPI	-	1/2
I_SPI_FIRSTBIT	Set MSB or LSB first bit of SPI	msb/lsb	msb/lsb
I_SPI_FIRSTBIT	Read type of first bit of SPI	-	msb/lsb

Examples:

"http://192.168.0.205/control?I_MODE=SPI" – "SPI"

"http://192.168.0.205/control?I_SPI_WRITE_READ=%9F%00%00%00%00" – "%FF%1FG%01%00"

"http://192.168.0.205/control?I_SPI_READ=5" – "%01%02%03%04%05"

"http://192.168.0.205/control?I_SPI_BAUDRATE=22500000" – "22500000"

"http://192.168.0.205/control?I_SPI_POLARITY=low" – "LOW"

I2C commands

Command	Description	Input value	Return value
I_I2C_MODE	Set I2C mode	MASTER SLAVE SNIFFER	MASTER SLAVE SNIFFER
I_I2C_MODE	Get I2C mode	-	MASTER SLAVE SNIFFER
Master mode			
I_I2C_WRITE	Writes bytes to device Format 7bit address: [1byte address] [data bytes] Format 10bit address: [2bytes address] [data bytes]	string and url-encoded bytes	string and url-encoded bytes
I_I2C_READ	Reads bytes from device Format 7bit address: [1byte address] [2bytes size to read] Format 10bit address: [2bytes address] [2bytes size to read]	string and url-encoded bytes	string and url-encoded bytes
I_I2C_WRITE_READ	Writes then reads bytes with repeated start to/from device Format 7bit address: [1byte address] [1byte size to write] [2bytes size to read] [data bytes to write] Format 10bit address: [2bytes address] [1byte size to write] [2bytes size to read] [data bytes to write]	string and url-encoded bytes	string and url-encoded bytes
I_I2C_ADDRSIZE	Set address size for I2C	7/10	7/10
I_I2C_ADDRSIZE	Read address size for I2C	-	7/10
I_I2C_BAUDRATE	Set I2C baudrate. The baudrate is set in 10 kHz steps in the range of 10-400 kHz and in 100 kHz steps in the 400-1000 kHz range	10000..1000000	10000..1000000
I_I2C_BAUDRATE	Read I2C baudrate	-	10000..1000000
I_I2C_TIMEOUT	Set timeout in ms for I2C bus errors	disable/0..30000	disable/0..30000

I_I2C_TIMEOUT	Read timeout in ms for I2C bus errors	-	disable/0..30000
Slave mode			
I_I2C_SLAVE_ADDR	Set the address of the Fork on the I2C bus	0..1023	0..1023
I_I2C_SLAVE_ADDR	Get the address of the Fork on the I2C bus	-	0..1023
I_I2C_ADDRSIZE	Set address size of the Fork on the I2C bus	7/10	7/10
I_I2C_ADDRSIZE	Read address size of the Fork on the I2C bus	-	7/10
I_I2C_CREATE_REG	Create a register. [register id],[register address],[memory address],[register size]	[0..15],[0..255],[0..199],[1..200]	[0..15],[0..255],[0..199],[1..200]
I_I2C_REG	Write data to register	[0..15],[string and url-encoded bytes]	string and url-encoded bytes
I_I2C_REG	Read information and data from the register. Returns [register address],[memory address],[register size],[data]	0..15	[0..255],[0..199],[1..200],[string and url-encoded bytes]
I_I2C_REG_ALL	Read information and data from all registers. Returns array of [register address],[memory address],[register size],[data]	-	array [[0..255],[0..199],[1..200],[string and url-encoded bytes]]
I_I2C_MEM	Get all memory of I2C Slave	-	string and url-encoded bytes
I_I2C_MEM_CLEAR	Set all memory bytes to 0x00	-	memory cleared
Sniffer mode			
I_I2C_SNIFFER	Get sniffer data	-	sniffer format string
I_I2C_SNIFFER_CLEAR	Clear sniffer buffer	-	sniffer buffer cleared

Examples:

"http://192.168.0.205/control?I_MODE=I2C" – "I2C"

"http://192.168.0.205/control?I_I2C_BAUDRATE=400000" – "400000"

"http://192.168.0.205/control?I_I2C_ADDRSIZE=7" – "7"

"http://192.168.0.205/control?I_I2C_TIMEOUT=1000" – "1000"

"http://192.168.0.205/control?I_I2C_WRITE_READ=@%01%00%03%E3" – "%05a%01"

"http://192.168.0.205/control?I_I2C_SLAVE_ADDR=64" - "64"

"http://192.168.0.205/control?I_I2C_SLAVE_ADDR" - "64"

"http://192.168.0.205/control?I_I2C_ADDRSIZE" - "7"

"http://192.168.0.205/control?I_I2C_ADDRSIZE=7" - "7"

"http://192.168.0.205/control?I_I2C_CREATE_REG=11,4,30,4" - "11,4,30,4"

"http://192.168.0.205/control?I_I2C_REG=11,fgt%10" - "fgt%10"

"http://192.168.0.205/control?I_I2C_REG=11" - "fgt%10"

"http://192.168.0.205/control?I_I2C_REG_ALL" -

"32,32,10,ruitdfh%00%00%00,0,0,0,,0,0,0,,0,0,0,,0,0,0,,0,0,0,,0,0,0,,0,0,0,,0,0,0,,0,0,0,,0,0,0,,0,0,0,,

0,0,0,,0,0,0,,0,0,0,,0,0,0,"

"http://192.168.0.205/control?I_I2C_MEM_CLEAR" - "memory cleared"

"http://192.168.0.205/control?I_I2C_SNIFFER" - "[Sw A0a 08a Sr A1a 00a 55a 55a 55n P]"

"http://192.168.0.205/control?I_I2C_SNIFFER_CLEAR" - "sniffer buffer cleared"

Ethernet (ETH) commands

Command	Description	Input value	Return value
ETH_MODE/ ETH_DHCP	Enable or disable DHCP on device	STATIC/ DYNAMIC	STATIC/ DYNAMIC
ETH_MODE/ ETH_DHCP	Read status of DHCP	-	STATIC/ DYNAMIC
ETH_IP	Set IP address of device	[0-255].[0-255]. [0-255].[0-255]	[0-255].[0-255]. [0-255].[0-255]
ETH_IP	Read settings of device IP address	-	[0-255].[0-255]. [0-255].[0-255]
ETH_IP_MASK	Set subnet mask of device	[0-255].[0-255]. [0-255].[0-255]	[0-255].[0-255]. [0-255].[0-255]
ETH_IP_MASK	Read subnet mask of device	-	[0-255].[0-255]. [0-255].[0-255]
ETH_GATEWAY	Set gateway IP address	[0-255].[0-255]. [0-255].[0-255]	[0-255].[0-255]. [0-255].[0-255]
ETH_GATEWAY	Read gateway IP address	-	[0-255].[0-255]. [0-255].[0-255]
ETH_NAME	Set name of device. Name consists of 19 characters.	string	string
ETH_NAME	Read name of device	-	string
ETH_SAVE	Save settings in memory, settings are applied after reset by power	-	SAVE
ETH_LOAD	Load settings from memory	-	LOAD

Examples:

"http://192.168.0.205/control?ETH_MODE=DYNAMIC" – "DYNAMIC"

"http://192.168.0.205/control?ETH_IP=192.168.0.223" – "192.168.0.223"

"http://192.168.0.205/control?ETH_IP_MASK=255.255.255.0" – "255.255.255.0"

"http://192.168.0.205/control?ETH_GATEWAY=192.168.0.1" – "192.168.0.1"

"http://192.168.0.205/control?ETH_NAME=Fork_garden" – "Fork_garden"

"http://192.168.0.205/control?ETH_SAVE" – "SAVE"

"http://192.168.0.205/control?ETH_LOAD " – "LOAD"

Wi-Fi (WIFI) commands (in development)

Command	Description	Input value	Return value
WIFI_MODE	Set mode of Wi-Fi AP - Access point AP_STA - Access point and Station STA - Station	[AP, AP_STA, STA]/[0-2]	[AP, AP_STA, STA]
WIFI_MODE	Return mode of Wi-Fi	-	[AP, AP_STA, STA]
WIFI_SAVE	Save settings in memory	-	SAVE
WIFI_LOAD	Load settings from memory	-	LOAD
Station mode settings			
WIFI_DHCP	Enable or disable DHCP client on device	STATIC/ DYNAMIC 0/1	STATIC/ DYNAMIC
WIFI_DHCP	Read status of DHCP client	-	STATIC/ DYNAMIC
WIFI_IP	Set IP address of device	[0-255].[0-255].[0-255].[0-255]	[0-255].[0-255].[0-255].[0-255]
WIFI_IP	Read IP address of device	-	[0-255].[0-255].[0-255].[0-255]
WIFI_IP_MASK	Set subnet mask of device	[0-255].[0-255].[0-255].[0-255]	[0-255].[0-255].[0-255].[0-255]
WIFI_IP_MASK	Read subnet mask of device	-	[0-255].[0-255].[0-255].[0-255]
WIFI_GATEWAY	Set gateway IP address	[0-255].[0-255].[0-255].[0-255]	[0-255].[0-255].[0-255].[0-255]
WIFI_GATEWAY	Read gateway IP address	-	[0-255].[0-255].[0-255].[0-255]
WIFI_NAME	Set name of device. Name consists of 19 characters.	string	string
WIFI_NAME	Read name of device	-	string

WIFI_SSID	Set the SSID of the access point to which the device connects	string	string
WIFI_SSID	Get the SSID of the access point to which the device connects	-	string
WIFI_PASSWORD	Set the password of the access point to which the device connects.	string	string
WIFI_PASSWORD	Get the password of the access point to which the device connects	-	string
WIFI_MAC	Return MAC address of device for Station mode	-	[00-FF]:[00-FF]:[00-FF]:[00-FF]:[00-FF]:[00-FF]
WIFI_LOCAL_IP	Read IP address of device if DHCP on	-	[0-255].[0-255].[0-255].[0-255]
WIFI_LOCAL_IP_MASK	Read subnet maks of device if DHCP on	-	[0-255].[0-255].[0-255].[0-255]
WIFI_LOCAL_GATEWAY	Read gateway IP address of device if DHCP on	-	[0-255].[0-255].[0-255].[0-255]
WIFI_START_SCAN	Start scanning process	-	OK
WIFI_SCAN_RESULT	It returns the state or result of scanning. Once requesting result it is cleared. Result consists of two parts: number of networks and array of found network SSIDs with URL-encoded semicolon separator.	-	EMPTY/SCANNING/[result]
Access point mode settings			
WIFI_AP_SSID	Set SSID of device in AP mode	string	string
WIFI_AP_SSID	Get SSID of device in AP mode	-	string
WIFI_AP_PASSWORD	Set password of device in AP mode Password must be 8-19 characters.	string	string

WIFI_AP_PASSWORD	Get password of device in AP mode	-	string
WIFI_AP_ENCRYPTION	Set encryption type of AP	OPEN/WPA2	OPEN/WPA2
WIFI_AP_ENCRYPTION	Get encryption type of AP	-	OPEN/WPA2
WIFI_AP_CHANNEL	Set Wi-Fi channel. Works only in AP mode (not in AP_STA)	1..13	1..13
WIFI_AP_CHANNEL	Get Wi-Fi channel.	-	1..13
WIFI_AP_HIDDEN	Hides or shows Fork SSID for other devices(1= hidden)	0/1	0/1
WIFI_AP_HIDDEN	Return SSID hidden state	-	0/1
WIFI_AP_MAX_CONNECTION	Set the maximum number of devices that can connect	0/1	0/1
WIFI_AP_MAX_CONNECTION	Get the maximum number of devices that can connect	-	0/1
WIFI_AP_MAC	Return MAC address of Access point	-	[00-FF]:[00-FF]:[00-FF]:[00-FF]:[00-FF]:[00-FF]

Examples:

"http://192.168.0.205/control?WIFI_MODE=STA" – "STA"

"http://192.168.0.205/control?WIFI_MODE" – "STA"

"http://192.168.0.205/control?WIFI_SAVE" - "SAVE"

"http://192.168.0.205/control?WIFI_DHCP" - "STATIC"

"http://192.168.0.205/control?WIFI_DHCP=1" - "DYNAMIC"

"http://192.168.0.205/control?WIFI_IP=192.168.0.223" – "192.168.0.223"

"http://192.168.0.205/control?WIFI_IP" – "192.168.0.223"

"http://192.168.0.205/control?WIFI_IP_MASK" – "255.255.255.0"

“http://192.168.0.205/control?WIFI_NAME=Fork_lab3” – “Fork_lab3”

“http://192.168.0.205/control?WIFI_NAME” – “Fork_lab3”

“http://192.168.0.205/control?WIFI_SSID=Fabric4” – “Fabric4”

“http://192.168.0.205/control?WIFI_SSID” – “Fabric4”

“http://192.168.0.205/control?WIFI_PASSWORD=112233bb” - “112233bb”

“http://192.168.0.205/control?WIFI_PASSWORD” - “112233bb”

“http://192.168.0.205/control?WIFI_MAC” - “F0:08:D1:FF:FF:F0”

“http://192.168.0.205/control?WIFI_LOCAL_IP” - “192.168.0.197”

“http://192.168.0.205/control?WIFI_START_SCAN” – “OK”

“http://192.168.0.205/control?WIFI_SCAN_RESULT” – “EMPTY”

“http://192.168.0.205/control?WIFI_SCAN_RESULT” – “SCANNING”

“http://192.168.0.205/control?WIFI_SCAN_RESULT” – “3,Office2%3BFabric4%3BRestRoom”

“http://192.168.0.205/control?WIFI_AP_SSID=Fork_A25F” - “Fork_A25F”

“http://192.168.0.205/control?WIFI_AP_SSID” - “Fork_A25F”

“http://192.168.0.205/control?WIFI_AP_PASSWORD=1234554321” - “1234554321”

“http://192.168.0.205/control?WIFI_AP_PASSWORD” - “1234554321”

“http://192.168.0.205/control?WIFI_AP_ENCRYPTION=WPA2” – “WPA2”

“http://192.168.0.205/control?WIFI_AP_ENCRYPTION” – “WPA2”

“http://192.168.0.205/control?WIFI_AP_CHANNEL=3” - “3”

“http://192.168.0.205/control?WIFI_AP_CHANNEL” - “3”

‘http://192.168.0.205/control?WIFI_AP_HIDDEN=0” - “0”

‘http://192.168.0.205/control?WIFI_AP_HIDDEN” - “0”

“http://192.168.0.205/control?WIFI_AP_MAX_CONNECTION=3” - “3”

“http://192.168.0.205/control?WIFI_AP_MAX_CONNECTION” - “3”

“http://192.168.0.205/control?WIFI_AP_MAC” - “F0:08:D1:FF:FF:F1”

Other commands

Command	Description	Input value	Return value
G_DEVICE_INFO	Return info about device in next order: firmware version,hardware version, bootloader version, serial number, production date, loading date, wifi module version	-	

Examples:

"http://192.168.0.205/control?G_DEVICE_INFO" -
"1.0,3.0,1.0,01193901528,28.09.2019,30.09.2019,0.10"