1. Pin definition

for GPIO:
“0” - GPIO0
“1” - GPIO1, UART0TX
“2” - GPIO2, UART1TX
“3” - GPIO3, UART0RX
“4” - GPIO4
“5” - GPIO5
“12” - GPIO12
“13” - GPIO13
“14” - GPIO14
“15” - GPIO15
for ADC:
“0” - ADC0
Common:
“all” - all pins in selected group

Notes:
UART1 has only TX. Usually It uses for debug output.
GPIO6-GPIO11 usually connected to on-module EEPROM, that is why no API for this pins.

2. GPIO

1.1 gpio/write

Sets gpio pins state according to the specified parameters. Pins will be automatically initialized to output. All pins will be setted up simultaneously. Unlisted pins will not be touched.

Parameters:
Json with set of key-value, where key is pin name and value ‘0’, ‘1’ or ‘z’. ‘0’ means low level, ‘1’ means high level, ‘x’ means dummy request which added for compatibility and easy string json generation. Sample below, sets gpio10 to low level and gpio11 to high level.

```
{
        “10”:”0””,
        “11”:”1”,
        “12”:”x”
}
```

Return ‘OK’ in status on success or ‘Error’ and description in result on error.

1.2 gpio/read

Read all gpio pins state. Pins will not be initialized as input. if pins were not specified in parameters.

Parameters:
Can be empty.
Json with set of key-value, where key is pin name and value can be:
“init” - all pins are initialized as input by default, if pin was used as output or any other peripheral module before, pass this argument to reinit pin before reading. Pullup state will not be touched.
“pullup” - init pin as input and enable pullup
“nopull” - init pin as input and disable pullup
Example:

```
{
        “10”:”init”,
        “11”:”pullup”,
        “12”:”nopull”
}
```

Note: pull up and pull down are the SoC feature that allow to put input to high level or low level through resistor with very high resistance. By default each pin in float (‘Z”) condition which state not determined and reading will return random value if pin doesn’t connect to high or low source. Enabling pull up feature helps to have very weak high level on input pin by default and pull down sets very weak low level.

Return ‘OK’ in status and json like below in result on success. Or ‘Error’ and description in result on error.

```
{
        “0”:”0””,
        “1”:”1”
        ….
        “16”:”0”
}
```

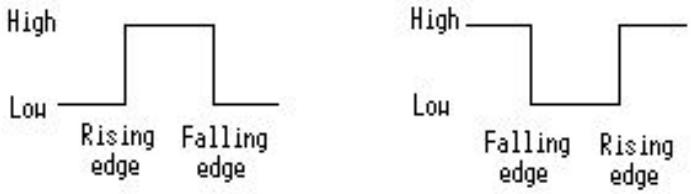
1.3 gpio/int

Enable or disable notification on pin state changes(enable interruptions).

Parameters:
Json with set of key-value, where key is pin name and value can be:
”disable” - disable interruption if it was enabled before
“rising” - send notification on rising edge
“falling” - send notification on falling edge

“both” - send notification on rising and falling edge
“low” - - send notification on low level
“high” - - send notification on high level

Note:



Example:

```
{
        “all”:”disable”,
        “11”:”rising”,
        “12”:”falling”,
        “13”:“both”,
        “14”:“low”,
        “15”:“high”,
}
```

Return ‘OK’ in status. Or ‘Error’ and description in result on error.
Notifications will be sent with name ‘gpio’. Each notification will contain list of gpio which caused interruption in ‘caused’ field and current gpio inputs(all of them) state in ‘state’ field:

```
{
        “caused”:[“0”, ”1”],
        “state”:{
                “0”:”0”,
                “1”:”1”
                ….
                “16”:”0”
        }
}
```

3. ADC

ESP8266 hasjust one ADC channel. This channel connected to dedicated pin #6 - ‘TOUT’. ADC can measure voltage in range 0..1 Volts with 10 bit resolution.

3.1 adc/read

Reads ADC channels values. ESP8266 has just one channel - ‘0’.
Parameters:
Can be empty, all channels value will be sent in this case.
Json with set of key-value, where key is ADC channel and value can be:
”read” - read channel current value

Example:

```
{
        “all”:”read”,
        “0”:”read”
}
```

Return ‘OK’ in status and json like below in result on success. Each entry contains channel number and value in volts. ‘Error’ and description will be send in result on error.

```
{
        “0”:”0.6”
}
```

3.2 adc/int

Subscribes on notifications with ADC value with some period.

Parameters:
Json with set of key-value, where key is ADC channel and value is period in milliseconds or ‘disable’ for disabling. ‘0’ value also means disable. Period can be from 50 to 8388607 ms.
Example:

```
{
        “0”:”1000”
        “0”:”disable”
}
```

In this example value of channel 0 will be sent every 1 second.

Return ‘OK’ in status and empty line in result on success. ‘Error’ and description will be returned in result on error.