Команды:

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Описание | Запрос | Ответ |
| NOP (0x00) |  |  |  |
| ERR (0x01) |  |  |  |
| ECHO (0x02) |  |  |  |
| [INFO](#_Команда_INFO) (0x03) | Запрос информации об устройстве | 0 байт | До 32 байт данных |
| [BOOT](#_Команда_BOOT) (0x04) | Переключение в Bootloader | 12 байт | ERR ил BOOT |
|  |  |  |  |
| [PORTS\_IDR](#_Команда_PORTS_IDR) (0x06) | Считать состояние входных портов | 0 байт | 2 байта |
| [PORTS\_ODRR](#_Команда_PORTS_ODRR_(0x07)) (0x07) | Считать состояние выходных портов | 0 байт | 2 байта |
| [PORTS\_ODRW](#_Команда_PORTS_ODRW_(0x08)) (0x08) | Записать данные в выходные порты | 2 байта | 0 байт |
| [PORTS\_SET](#_Команда_PORTS_SET_(0x09)) (0x09) | Установить соответствующие выходы в 1 | 2 байта | 0 байт |
| [PORTS\_RESET](#_Команда_PORTS_RESET_(0xA)) (0x0A) | Установить соответствующие выходы в 0 | 2 байта | 0 байт |
|  |  |  |  |
| [RELAYS\_IDR](#_Команда_RELAYS_IDR_(0x0C)) (0x0C) | Считать состояние дополнительных входов | 0 байт | 2 байта |
| [RELAYS\_ODRR](#_Команда_RELAYS_ODRR_(0x0D)) (0x0D) | Считать состояние дополнительных выходов | 0 байт | 2 байта |
| [RELAYS\_ODRW](#_Команда_RELAYS_ODRW_(0x0E)) (0x0E) | Записать данные в дополнительные выходы | 2 байта | 0 байт |
| [RELAYS\_SET](#_Команда_RELAYS_SET_(0x0F)) (0x0F) | Установить дополнительные выходы в 1 | 2 байта | 0 байт |
| [RELAYS\_RESET](#_Команда_RELAYS_RESET_(0x10)) (0x10) | Установить дополнительные выходы в 0 | 2 байта | 0 байт |
|  |  |  |  |
| [POWERS\_IDR](#_Команда_POWERS_IDR_(0x11)) (0x11) | Считать регистр аварий управляемых силовых выходов | 0 байт |  |
| [POWERS\_ODRR](#_Команда_POWERS_ODRR_(0x12)) (0x12) | Считать состояние управляемых силовых выходов |  |  |
| [POWERS\_ODRW](#_Команда_POWERS_ORRW_(0x13)) (0x13) | Записать данные в управляемые силовые выходы |  |  |
| [POWERS\_SET](#_Команда_POWERS_SET_(0x14)) (0x14) | Установить управляемые силовые выходы в 1 |  |  |
| [POWERS\_RESET](#_Команда_POWERS_RESET_(0x15)) (0x15) | Установить управляемые силовые выходы в 0 |  |  |
|  |  |  |  |
| [WIEGAND](#_Команда_WIEGAND_(0x0B)) (0x0B) | Считать данные из Wiegand |  |  |
| [CLIMATE\_GET](#_Команда_CLIMATE_GET_(0x16)) (0x16) | Считать информацию по климатике | 0 байт | 17 байт |
| [CLIMATE\_SET](#_Команда_CLIMATE_SET_(0x17)) (0x17) | Установить режим работы климатики | 1 байт | 0 байт |

Протокол обмена данными со слейвом – немного модифицированный SLIP, который называется Wake. Все описано в файле wake.pdf, но есть одно !!!изменение!!! Алгоритм CRC-8 основан на полиноме 0x131, начальное значение 0x00, обращенный, выход не XOR. Для проверки CRC8(b’123456789’) == 0xA1.

Программы управления в папке Tools. Всем программам нужно передать название COM порта и адрес устройства (в данном случае – 2):

1. Info.py – Получение информации о названии, версии, уникальном ID
2. ChangeToBoot.py – Переключение слейва в загрузчик
3. Ports.py – работа с дискретами PORTA, PORTB, группа команд PORTS\_\*
4. Relays.py – работа с дополнительными дискретами, группа команд RELAYS\_\*
5. Power.py – работа с управляемыми портами питания, группа команд POWERS\_\*
6. Climate.py – работа с модулем климатики, группа команд CLIMATE\_\*
7. Wiegand.py – чтение модуля Виганда.

# Команда INFO

## Описание:

Возвращает информацию об устройстве:

## Запрос:

C0 82 03 00 23

## Ответ:

80 00 34 FF 67 06 4D 50 32 36 28 22 11 43 01 00 00 00 53 6C 61 76 65 2D 61 72 6D

1. 2 байта – размер памяти (little-endian) (80 00)
2. 12 байт – уникальный ID (34FF67064D50323628221143)
3. 4 байта – версия или порядковый номер устройства (01 00 00 00)
4. Строка до 14 байт – текстовое название устройства (Slave-arm)

# Команда [BOOT](#_Команда_BOOT) (0x04)

## Описание

Переключает устройство в режим Bootloader для загрузки ПО, обновления…

Работа загрузчика описана в документе AN3155 и AN2606 для процессора STM32F1 series.

## Запрос

12 байт, сформированных из уникального ID по XOR с ключом '64 CA 56 BA 15 57 63 39 DA 57 40 21'

## Ответ

BOOT без данных – в случае успешности. Устройство переключится в загрузчик через 1 секунду.

ERR – во всех остальных случаях.

# Команда [PORTS\_IDR](#_Команда_PORTS_IDR) (0x06)

## Описание

Считывание состояние входов PORTA, B

## Запрос

C0 82 06 00 DC

## Ответ

C0 81 06 02 00 00 14

2 байта данных. Младший байт – PORTA, старший – PORTB. Значащие в этих байтах – младшие 4 бита, отвечающие за Px0 – Px3 (x = A or B).

# Команда PORTS\_ODRR (0x07)

## Описание

Считывание состояние выходов PORTA и PORTB

## Запрос

C0 82 07 00 18

## Ответ

C0 81 07 02 08 0E F2

2 байта данных (08 0E). Младший байт – PORTA (08), старший байт – PORTB (0E). Значащие в этих байтах – младшие 4 бита, отвечающие за Px0 – Px3 (x = A or B).

# Команда PORTS\_ODRW (0x08)

## Описание

Запись данных в выходные порты (дискретные выходы) PORTA, B

## Запрос

C0 82 08 02 07 03 C3

2 байта данных. Младший байт – состояние PORTA (07), старший – PORTB (03). Значащие в этих байтах – младшие 4 бита, отвечающие за Px0 – Px3 (x = A or B).

## Ответ

PORTS\_ODRW в случае успешности,

ERR – во всех остальных

# Команда PORTS\_SET (0x09)

## Описание

Установить дискретные выходы PORTA, B в единицу.

## Запрос

C0 82 09 02 07 03 4C

2 байта данных. Младший байт – состояние PORTA (07), старший – PORTB (03). Значащие в этих байтах – младшие 4 бита, отвечающие за Px0 – Px3 (x = A or B).

## Ответ

PORTS\_SET в случае успешности

ERR – во всех остальных

# Команда PORTS\_RESET (0x0A)

## Описание

Установить дискретные выходы PORTA, B в нули.

## Запрос

C0 82 0A 02 07 03 C4

2 байта данных. Младший байт – состояние PORTA (07), старший – PORTB (03). Значащие в этих байтах – младшие 4 бита, отвечающие за Px0 – Px3 (x = A or B).

## Ответ

PORTS\_RESET в случае успешности

ERR – во всех остальных

# Команда RELAYS\_IDR (0x0C)

## Описание

Чтение дополнительных дискретых входов

## Запрос

C0 82 0C 00 3B

## Ответ

C0 81 0C 02 02 09 02

2 байта данных 02 09. Значение на дискретных входах: 0 – разомкнут, 1 – замкнут. Значащими битами являются 10 бит (10\_0000\_1001).

# Команда RELAYS\_ODRR (0x0D)

## Описание

Чтение дополнительных дискретных выходов.

## Запрос

C0 82 0D 00 FF

## Ответ

C0 81 0D 02 02 C2 67

2 байта данных 02 C2. Состояние дополнительных дискретных выходов: 0 – разомкнут, 1 – замкнут. Значащими битами являются 10 бит (10\_1100\_0010)

# Команда RELAYS\_ODRW (0x0E)

## Описание

Установить дискретные выходы в соответствующее состояние.

## Запрос

C0 82 0E 02 03 2A 4E

2 байта данных (03 2А). Перевести дискретные выходы в указанное состояние: 0 – разомкнуты, 1 – замкнуты. Значащими битами являются 10 бит (11\_0010\_1010)

## Ответ

RELAYS\_ODRW в случае успешности

ERR – во всех остальных

# Команда RELAYS\_SET (0x0F)

## Описание

Установить дискретные выходы в ЕДИНИЦУ.

## Запрос

C0 82 0F 02 03 2A C1

2 байта данных (03 2А). Перевести указанные дискретные выходы в ЕДИНИЦУ. Значащими битами являются 10 бит (11\_0010\_1010)

## Ответ

RELAYS\_SET в случае успешности

ERR – во всех остальных

# Команда RELAYS\_RESET (0x10)

## Описание

Установить дискретные выходы в НУЛИ.

## Запрос

C0 82 10 02 03 2A 63

2 байта данных (03 2А). Перевести указанные дискретные выходы в НОЛЬ. Значащими битами являются 10 бит (11\_0010\_1010)

## Ответ

RELAYS\_RESET в случае успешности

ERR – во всех остальных

# Команда POWERS\_IDR (0x11)

## Описание

Считать значение ошибок силовых выходов.

## Запрос

C0 82 11 00 5E

## Ответ

C0 81 11 02 00 00 AA

2 байта, всегда нули, не реализовано.

# Команда POWERS\_ODRR (0x12)

## Описание

Считать состояние силовых выходов.

## Запрос

C0 82 12 00 0B

## Ответ

C0 81 12 02 02 A7 9F

2 байта 02 A7. Состояние силовых выходов: 0 – отключен, 1 – включен. Значащими битами являются 10 бит (10\_1010\_0111)

# Команда POWERS\_ODRW (0x13)

## Описание

Установить Силовые выходы в соответствующее состояние.

## Запрос

C0 82 13 02 02 A7 5E

2 байта данных (02 А7). Перевести силовые выходы в указанное состояние: 0 – отключен, 1 – включен. Значащими битами являются 10 бит (10\_1010\_0111).

## Ответ

POWERS\_ODRW в случае успешности

ERR – во всех остальных

# Команда POWERS\_SET (0x14)

## Описание

Установить Силовые выходы в включенное состояние.

## Запрос

C0 82 14 02 02 A7 D8

2 байта данных (02 А7). Перевести силовые выходы в включенное состояние. Значащими битами являются 10 бит (10\_1010\_0111).

## Ответ

POWERS\_SET в случае успешности

ERR – во всех остальных

# Команда POWERS\_RESET (0x15)

## Описание

Установить Силовые выходы в отключенное состояние.

## Запрос

C0 82 15 02 02 A7 57

2 байта данных (02 А7). Перевести силовые выходы в отключенное состояние. Значащими битами являются 10 бит (10\_1010\_0111).

## Ответ

POWERS\_RESET в случае успешности

ERR – во всех остальных

# Команда WIEGAND (0x0B)

## Описание

Чтение данных из Виганда.

## Запрос

C0 82 0B 00 55

## Ответ

C0 81 0B 06 01 1A 8F CD CF 80 68

# Команда CLIMATE\_GET (0x16)

## Описание

Запрос состояние контроллера климата. Считывание температур, влажности, состояние печки, вентилятора…

## Запрос

C0 82 16 00 30

## Ответ

C0 81 16 11 33 B3 96 43 33 A7 96 43 52 68 96 43 3B 0C 19 42 00 83

В полезной нагрузке 4 флоата по 32 бита (little-endian) и 1 байт статус. Температура в Кельвинах

Первый флоат – Локальная температура (33 B3 96 43), 301.400,

Второй флоат – Температура внешнего датчика (33 A7 96 43), 301.30,

Третий флоат – Температура гигрометра (52 68 96 43), 300.815

Четверный флоат – Влажность относительная в процентах (3B 0C 19 42), 38.3%

Статусный байт ST:

* ST[0] – состояние нагревателя (0 – выключен, 1 – включен)
* ST[1] - состояние вентилятора (0 – выключен, 1 – включен)
* ST[2] – режим работы (0 – ручное управление нагревателем и вентилятором, 1 – автоматическое)

# Команда CLIMATE\_SET (0x17)

## Описание

Установить режим работы климатики: ручное, автоматическое.

## Запрос

C0 82 17 01 03 33

1 байт полезной информации ST:

* ST[0] – состояние нагревателя (0 – выключен, 1 – включен)
* ST[1] – состояние вентилятора (0 – выключен, 1 – включен)
* ST[2] – режим работы (0 – ручное управление нагревателем и вентилятором, 1 – автоматическое)

ST[0], ST[1] работают только при ST[2] == 0. В автоматическом режиме изменение ST[0:1] ни на что не влияет.

## Ответ

CLIMATE\_SET без данных в случае успешности,

ERR – во всех остальных

# Команда CMD\_READ\_ALL (0x50)

## Описание

Запросить состояние всех входов, выходов и Вигандов Слейва.

## Запрос

Один байт аргумента (маска, комбинируем через ИЛИ):

|  |  |  |
| --- | --- | --- |
| Подкоманда | Значение | Описание |
| SCMD\_PORTS\_IDR | (1 << 0) (0x01) | Считать значение входов портов A и B |
| SCMD\_PORTS\_ODR | (1 << 1) (0x02) | Считать значение выходов портов А и B |
| SCMD\_RELAYS\_IDR | (1 << 2) (0x04) | Считать значение входов реле |
| SCMD\_RELAYS\_ODR | (1 << 3) (0x08) | Считать значение выходов реле |
| SCMD\_WIEGAND\_1 | (1 << 4) (0x10) | Считать Виганд со входа 1 |
| SCMD\_WIEGAND\_2 | (1 << 5) (0x20) | Считать Виганд со входа 2 |

## Ответ

Возвращает Protobuf (ResponseAll) с установленными полями в соответствии с запросом:

message Wieg {

required uint32 size = 1;

required bytes data = 2;

}

message ResponseAll {

optional uint32 PORTA\_IDR = 1;

optional uint32 PORTB\_IDR = 2;

optional uint32 PORTA\_ODR = 3;

optional uint32 PORTB\_ODR = 4;

optional uint32 RELAYS\_IDR = 5;

optional uint32 RELAYS\_ODR = 6;

optional Wieg WiegandCh1 = 7;

optional Wieg WiegandCh2 = 8;

}

# Команда CMD\_PULSE (0x51)

## Описание

«Дерныть» выход слейва с заданной длительностью и задержкой. Под словом дернуть подразумевается инвертирование выхода относительно текущего состояния с последующим возвратом в исходное.

## Запрос

Запрос идет в виде Protobuf (Pulse).

enum PulseError {

PE\_DECODE\_ERROR = 1;

PE\_NO\_RESOURCES = 2;

PE\_PIN\_BUSY = 3;

}

message Pulse {

required uint32 pin = 1; // Номер вывода, 0-3 - PORTA, 4-7 - PORTB, 8-17 - Relays

required uint32 width = 2; // Длительность импульса

optional uint32 delay = 3; // Задержка переключения вывода

}

Где pin – номер вывода (0-3 – PORTA, 4-7 – PORTB, 8-17 – Релейные выходы), обязательный параметр; width – длительность импульса в мс, обязательный параметр; delay – задержка переключение после получения команды, мс, необязательный.

После получения команды при установленном delay выход не меняет своего состояния delay мс. По прошествии delay мс выход переключается. Обратно вывод переключается по прошествии width мс. Минимально достигнутое время переключений 1 мс. Если выход включен в режим Импульса, то его не переключить обратно, пока не отработает импульс. Состояние выхода можно переключать командами PORTS\* и RELAY\*.

Максимальное одновременное количество выходов в режиме Импульс 5.

## Ответ

Ответ успешного добавления выхода в Импульс – пакет данных с номером команды (0x51) и нулевой длинной данных.

Не успешное завершение команды – пакет CMD\_ERR (0x01) и 1 байт статуса (PulseError):

PE\_DECODE\_ERROR – Ошибка декодирования Protobuf

PE\_NO\_RESOURCES – Закончились свободные ресурсы для импульсных выходов (максимум 5)

PE\_PIN\_BUSY – Заданный вывод уже в режиме Импульс, подождите.

# Команда OUTPUT\_ALL\_ODRW (0x54), OUTPUT\_ALL\_SET (0x55), OUTPUT\_ALL\_RESET (0x56), OUTPUT\_ALL\_TOGGLE(0x57)

## Описание

Записать выходное состояние PORTA, PORTB и Relays. По сути, соединение команд PORTS\* и RELAYS\* в одну.

\*\_ODRW – установка значения; \_SET – установка в 1; \_RESET – установка в 0; \_TOGGLE – переключение выхода.

## Запрос

C0 82 54 04 01 02 00 84 61

4 значащих байта (01 02 00 84). 01 – PORTA, 02 – PORTB, 0084 – RELAYS. Позиции байт такие же как в материнских командах.