



**Электропривод**

<http://electroprivod.ru>

***Блок управления шаговым двигателем***

**Модели SMSD-4.2LAN и SMSD-8.0LAN**

**Протокол обмена данными**

**Ver. 04**



<b>1. Основные сведения.</b>	<b>5</b>
<b>2. Принцип передачи данных по Ethernet и USB.</b>	<b>5</b>
<b>3. Заводские настройки</b>	<b>5</b>
<b>4. Структура информационного пакета передачи данных</b>	<b>6</b>
4.1. Назначение области XOR_SUM	6
4.2. Назначение области Ver.	7
4.3. Назначение области CMD_TYPE	7
4.3.1 Начало сессии обмена данными с контроллером. Команда передачи данных CODE_CMD_REQUEST	8
4.3.2 Команда передачи данных CODE_CMD_RESPONSE	9
4.3.3 Команда передачи данных CODE_CMD_POWERSTEP01	10
4.3.4 Команды передачи данных CODE_CMD_POWERSTEP01_W_MEM0..MEM311	
4.3.5 Команды передачи данных CODE_CMD_POWERSTEP01_R_MEM0..MEM312	
4.3.6 Команда передачи данных CODE_CMD_CONFIG_SET	14
4.3.7 Команда передачи данных CODE_CMD_CONFIG_GET	14
4.3.8 Команда передачи данных CODE_CMD_PASSWORD_SET	15
4.3.9 Команда передачи данных CODE_CMD_ERROR_GET	16
4.4. Назначение области CMD_IDENTIFICATION	17
4.5. Назначение области LENGTH_DATA	17
4.6. Назначение области DATA[LENGTH_DATA]	17
<b>5. Структура COMMANDS_RETURN_DATA_Type</b>	<b>17</b>
5.1 Назначение битовых полей STATUS_POWERSTEP01	17
5.2 Список возможных значений поля ERROR_OR_COMMAND	18
<b>6. Структура исполнительных команды управления SMSD_CMD_Type</b>	<b>19</b>
6.1 Исполнительная команда CMD_PowerSTEP01_END	20
6.2 Исполнительная команда CMD_PowerSTEP01_GET_SPEED	20
6.3 Исполнительная команда CMD_PowerSTEP01_STATUS_IN_EVENT	22
6.4 Исполнительная команда CMD_PowerSTEP01_SET_MODE	22
6.5 Исполнительная команда CMD_PowerSTEP01_GET_MODE	25
6.6 Исполнительная команда CMD_PowerSTEP01_SET_MIN_SPEED	26
6.7 Исполнительная команда CMD_PowerSTEP01_SET_MAX_SPEED	26
6.8 Исполнительная команда CMD_PowerSTEP01_SET_ACC	27
6.9 Исполнительная команда CMD_PowerSTEP01_SET_DEC	27
6.10 Исполнительная команда CMD_PowerSTEP01_SET_FS_SPEED	27
6.11 Исполнительная команда CMD_PowerSTEP01_SET_MASK_EVENT	27



6.12	Исполнительная команда	CMD_PowerSTEP01_GET_ABS_POS .....	28
6.13	Исполнительная команда	CMD_PowerSTEP01_GET_EL_POS.....	28
6.14	Исполнительная команда	CMD_PowerSTEP01_GET_STATUS_AND_CLR	29
6.15	Исполнительная команда	CMD_PowerSTEP01_RUN_F .....	29
6.16	Исполнительная команда	CMD_PowerSTEP01_RUN_R.....	29
6.17	Исполнительная команда	CMD_PowerSTEP01_MOVE_F.....	30
6.18	Исполнительная команда	CMD_PowerSTEP01_MOVE_R .....	30
6.19	Исполнительная команда	CMD_PowerSTEP01_GO_TO_F .....	31
6.20	Исполнительная команда	CMD_PowerSTEP01_GO_TO_R .....	31
6.21	Исполнительная команда	CMD_PowerSTEP01_GO_UNTIL_F.....	31
6.22	Исполнительная команда	CMD_PowerSTEP01_GO_UNTIL_R .....	32
6.23	Исполнительная команда	CMD_PowerSTEP01_SCAN_ZERO_F .....	32
6.24	Исполнительная команда	CMD_PowerSTEP01_SCAN_ZERO_R .....	32
6.25	Исполнительная команда	CMD_PowerSTEP01_SCAN_LABEL_F .....	33
6.26	Исполнительная команда	CMD_PowerSTEP01_SCAN_LABEL_R.....	33
6.27	Исполнительная команда	CMD_PowerSTEP01_GO_ZERO .....	33
6.28	Исполнительная команда	CMD_PowerSTEP01_GO_LABEL .....	34
6.29	Исполнительная команда	CMD_PowerSTEP01_GO_TO .....	34
6.30	Исполнительная команда	CMD_PowerSTEP01_RESET_POS .....	34
6.31	Исполнительная команда	CMD_PowerSTEP01_RESET_POWERSTEP01 ..	35
6.32	Исполнительная команда	CMD_PowerSTEP01_SOFT_STOP .....	35
6.33	Исполнительная команда	CMD_PowerSTEP01_HARD_STOP.....	35
6.34	Исполнительная команда	CMD_PowerSTEP01_SOFT_HI_Z .....	36
6.35	Исполнительная команда	CMD_PowerSTEP01_HARD_HI_Z .....	36
6.36	Исполнительная команда	CMD_PowerSTEP01_SET_WAIT .....	36
6.37	Исполнительная команда	CMD_PowerSTEP01_SET_RELE .....	37
6.38	Исполнительная команда	CMD_PowerSTEP01_CLR_RELE .....	37
6.39	Исполнительная команда	CMD_PowerSTEP01_GET_RELE.....	37
6.40	Исполнительная команда	CMD_PowerSTEP01_WAIT_IN0.....	38
6.41	Исполнительная команда	CMD_PowerSTEP01_WAIT_IN1.....	38
6.42	Исполнительная команда	CMD_PowerSTEP01_GOTO_PROGRAM .....	38
6.43	Исполнительная команда	CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN039	
6.44	Исполнительная команда	CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN139	
6.45	Исполнительная команда	CMD_PowerSTEP01_LOOP_PROGRAM.....	40
6.46	Исполнительная команда	CMD_PowerSTEP01_CALL_PROGRAM.....	40



6.47	Исполнительная команда	CMD_PowerSTEP01_RETURN_PROGRAM.....	41
6.48	Исполнительная команда	CMD_PowerSTEP01_START_PROGRAM_MEM041	
6.49	Исполнительная команда	CMD_PowerSTEP01_STOP_PROGRAM_MEM ..	42
6.50	Исполнительная команда	CMD_PowerSTEP01_STEP_CLOCK.....	42
6.51	Исполнительная команда	CMD_PowerSTEP01_STOP_USB .....	42
6.52	Исполнительная команда	CMD_PowerSTEP01_GET_MIN_SPEED.....	42
6.53	Исполнительная команда	CMD_PowerSTEP01_GET_MAX_SPEED .....	43
6.54	Исполнительная команда	CMD_PowerSTEP01_GET_STACK .....	43
6.55	Исполнительная команда	CMD_PowerSTEP01_GOTO_PROGRAM_IF_ZERO	44
6.56	Исполнительная команда	CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN_ZERO	44
6.57	Исполнительная команда	CMD_PowerSTEP01_WAIT_CONTINUE.....	45
6.58	Исполнительная команда	CMD_PowerSTEP01_SET_WAIT_2 .....	45
6.59	Исполнительная команда	CMD_PowerSTEP01_SCAN_MARK2_F .....	45
6.60	Исполнительная команда	CMD_PowerSTEP01_SCAN_MARK2_R.....	46
<b>7. Структура SMSD_LAN_Config_Type.....</b>			<b>46</b>
<b>8. Отличия между Ethernet и USB в передаче потока данных.....</b>			<b>47</b>



## 1. Основные сведения.

Блок управления SMSD\_LAN, далее по тексту - Контроллер, предназначен для управления шаговыми двигателями. Контроллер представляет собой электронное устройство, состоящее из корпуса, электронной платы, разъёмов, передней панели на которой находятся органы управления и индикации.

В режиме работы по локальной сети Ethernet (на индикаторе «LA»), Контроллер создаёт сокет для подключения к нему управляющей пользовательской программы или устройства, далее по тексту - Пользователь. Данные передаются по физической линии Ethernet (протокол TCP).

Для управления и настройки, вместо сети Ethernet, Контроллер можно подключить через интерфейс USB (имитация COM-port). При этом система команд аналогичная, за исключением незначительных отличий в передаче потока данных на прикладном уровне, данные отличия описаны ниже.

## 2. Принцип передачи данных по Ethernet и USB.

Передачу данных необходимо осуществлять законченными информационными пакетами, содержащими только одну команду управления типа [CMD TYPE](#).

Не допускается передача одновременно нескольких команд управления подряд в одном пакете. После приёма команды, Контроллер производит необходимые действия и отправляет ответ содержащий статус результата работы или данные. Ответ производится по тому же физическому каналу, по которому поступила управляющая команда. Последовательность байт в структурах пакетов – обратный, «от младшего к старшему», (Intel).

## 3. Заводские настройки

Параметры подключения по сети Ethernet, установленные в контроллере по умолчанию:

- MAC адрес: 0x00 0xf8 0xdc 0x3f 0x00 0x00
- IP адрес: 192.168.1.2
- Порт: 5000
- Маска подсети: 255.255.0.0
- Основной шлюз: 192.168.1.1

Данные параметры в дальнейшем можно поменять как по сети Ethernet, так и по USB интерфейсу.

Параметры передачи данных RS-232 (подключение USB):

- Скорость: 115200
- Бит данных: 8
- Проверка четности: нет
- Стоп биты: 1



## 4. Структура информационного пакета передачи данных

Структура информационного пакета передачи данных:

```
typedef struct
{
    uint8_t XOR_SUM;
    uint8_t Ver;
    uint8_t CMD_TYPE;
    uint8_t CMD_IDENTIFICATION;
    uint16_t LENGTH_DATA;
    uint8_t DATA[LENGTH_DATA];
}LAN_COMMAND_Type;
```

XOR\_SUM – контрольная сумма – младший байт от суммы всех байт.

Ver – версия протокола.

CMD\_TYPE – тип команды, передаваемой по сети.

CMD\_IDENTIFICATION – уникальный идентификатор, будет передан в ответном сообщении от Контроллера на данную команду, что позволяет однозначно сопоставить переданную команду и полученный ответ.

LENGTH\_DATA – длина информационной части пакета, значения от 0 до 1024.

DATA[LENGTH\_DATA] – информационная часть пакета длиной LENGTH\_DATA байт.

### 4.1. Назначение области XOR\_SUM

Протокол TCP подразумевает под собой механизм гарантированной доставки сообщения получателю и включает в себя проверку и исправление возможных ошибок. Тем не менее, в команде управления предусмотрено поле XOR\_SUM – контрольная сумма команды запроса/ответа. Предназначена для проверки на целостность пакета в случае его передачи по USB. Алгоритм вычисления контрольной суммы XOR\_SUM:

```
COMMAND.XOR_SUM=0x00;
COMMAND.XOR_SUM=xor_sum((uint8_t*)&COMMAND.XOR_SUM,
sizeof(COMMAND));
```

```
uint8_t xor_sum(uint8_t *data,uint16_t length)
{
    uint8_t xor_temp=0xFF;
    while(length--){xor_temp+=*data;data++;}
    return (xor_temp^0xFF);
}
```

Where:

(uint8\_t\*)& COMMAND.XOR\_SUM – начало передаваемого пакета,  
sizeof(COMMAND) – длина передаваемого пакета (в байтах).



## 4.2. Назначение области Ver.

Поле данных длиной 1 байт. Текущая версия коммуникационного протокола - 0x02 (установлена 19.04.2018).

## 4.3. Назначение области CMD\_TYPE

Поле данных длиной 1 байт. Команда, передаваемая по сети. Числовые значения начинаются с 0 и последовательно инкрементируются. Список возможных вариантов значений поля CMD\_TYPE:

[CODE\\_CMD\\_REQUEST](#) – команда авторизации (поле DATA пакета содержит информацию для авторизации)

[CODE\\_CMD\\_RESPONSE](#) – команда подтверждения (поле DATA пакета зависит от отправленной контроллеру команды)

[CODE\\_CMD\\_POWERSTEP01](#) – команда управления в реальном масштабе времени (поле DATA пакета содержит команды POWERSTEP01 типа [SMSD\\_CMD\\_Type](#)).

[CODE\\_CMD\\_POWERSTEP01\\_W\\_MEM0](#) – команда записи программы управления в банк памяти 0.

[CODE\\_CMD\\_POWERSTEP01\\_W\\_MEM1](#) – команда записи программы управления в банк памяти 1

[CODE\\_CMD\\_POWERSTEP01\\_W\\_MEM2](#) – команда записи программы управления в банк памяти 2

[CODE\\_CMD\\_POWERSTEP01\\_W\\_MEM3](#) – команда записи программы управления в банк памяти 3

[CODE\\_CMD\\_POWERSTEP01\\_R\\_MEM0](#) – команда чтения программы управления из банка памяти 0

[CODE\\_CMD\\_POWERSTEP01\\_R\\_MEM1](#) – команда чтения программы управления из банка памяти 1

[CODE\\_CMD\\_POWERSTEP01\\_R\\_MEM2](#) – команда чтения программы управления из банка памяти 2

[CODE\\_CMD\\_POWERSTEP01\\_R\\_MEM3](#) – команда чтения программы управления из банка памяти 3

[CODE\\_CMD\\_CONFIG\\_SET](#) – команда записи настроек LAN

[CODE\\_CMD\\_CONFIG\\_GET](#) - команда чтения настроек LAN

[CODE\\_CMD\\_PASSWORD\\_SET](#) – изменения пароля для авторизации

[CODE\\_CMD\\_ERROR\\_GET](#) - чтения количества включений рабочего режима Контроллера и статистики по ошибкам.





#### 4.3.1 Начало сессии обмена данными с контроллером. Команда передачи данных CODE\_CMD\_REQUEST

Команда CODE\_CMD\_REQUEST используется для авторизации пользователя. Пакет данных с командой CODE\_CMD\_REQUEST отправляется контроллером пользователю как ответ на факт подключения контроллера (только в случае подключения по сети Ethernet, не используется при подключении USB).

Из контроллера:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_REQUEST= 0x00	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (младший байт)	0x00	4
LENGTH_DATA (старший байт)	0x00	5
DATA	-	-

После получения пакета с кодом команды CODE\_CMD\_REQUEST, пользователь должен отправить пакет с кодом команды CODE\_CMD\_REQUEST, поле DATA должно содержать пароль для авторизации. Поле VER (версия протокола) в этой команде контроллером не проверяется. Заводское значение пароля: x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF. Это значение можно изменить при помощи команды [CODE\\_CMD\\_PASSWORD\\_SET](#).

В контроллер:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_REQUEST = 0x00	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (младший байт)	0x08	4
LENGTH_DATA (старший байт)	0x00	5
DATA [0] (пароль – младший байт)	x	6
DATA [1]	x	7
DATA [2]	x	8
DATA [3]	x	9
DATA [4]	x	10
DATA [5]	x	11
DATA [6]	x	12
DATA [7] (пароль – старший байт)	x	13

Контроллер проверяет полученные данные и отправляет ответ, содержащий информацию о результате авторизации. Тип команды CMD\_TYPE - [CODE\\_CMD\\_RESPONSE](#), поле DATA ответа содержит





структуру [COMMANDS\\_RETURN\\_DATA](#). Описание структуры [COMMANDS\\_RETURN\\_DATA](#) приведено далее в данном руководстве.

Из контроллера:

Поле	Значение	
XOR (1 байт)	x	
VER (1 байт)	x	
CMD_TYPE (1 байт)	CODE_CMD_RESPONSE = 0x01	
CMD_IDENTIFICATION (1 байт)	x	
LENGTH_DATA (младший байт)	sizeof( <a href="#">COMMANDS_RETURN_DATA_Type</a> )	0x07
LENGTH_DATA (старший байт)		0x00
DATA [0] ( <a href="#">COMMANDS_RETURN_DATA</a> – младший байт)	x	
DATA [1]	x	
DATA [2] = ERROR_OR_COMMAND	x	
DATA [3]	0	
DATA [4]	0	
DATA [5]	0	
DATA [6] ( <a href="#">COMMANDS_RETURN_DATA</a> старший байт)	0	

В случае получения правильного пароля контроллер открывает доступ к управлению, а поле ERROR\_OR\_COMMAND структуры [COMMANDS\\_RETURN\\_DATA](#) содержит ответ OK\_ACCESS. В случае неверного пароля ERROR\_OR\_COMMAND= ERROR\_ACCESS, и контроллер закрывает соединение. Следующее соединение и попытка авторизации возможны не ранее, чем через 1 секунду. В случае, если попытка подключения и авторизации происходят ранее установленного таймаута 1с, контроллер отправляет пакет с кодом CODE\_CMD\_RESPONSE, поле ERROR\_OR\_COMMAND = ERROR\_ACCESS\_TIMEOUT независимо от правильности пароля. Таймаут 1с предотвращает подбор пароля автоматическими сервисами.

#### 4.3.2 Команда передачи данных CODE\_CMD\_RESPONSE

Пакет данных с командой CODE\_CMD\_RESPONSE отправляется контроллером в ответ на некоторые команды пользователя ([CODE\\_CMD\\_POWERSTEP01](#), [CODE\\_CMD\\_CONFIG\\_SET](#), [CODE\\_CMD\\_ID\\_SET](#), [CODE\\_CMD\\_POWERSTEP01\\_W\\_MEM](#)), а так же, в случае возникновения разного рода ошибок. Поле DATA пакета содержит структуру [COMMANDS\\_RETURN\\_DATA](#) (описание структуры приведено далее в данном руководстве).

Из контроллера:

Поле	Значение		Порядок данных в пакете
XOR (1 байт)	x		0
VER (1 байт)	x		1
CMD_TYPE (1 байт)	CODE_CMD_RESPONSE = 0x01		2
CMD_IDENTIFICATION (1 байт)	x		3
LENGTH_DATA (Младший байт)	Sizeof( <a href="#">COMMANDS_RETURN_DATA_Type</a> )	0x07	4
LENGTH_DATA (Старший байт)		0x00	5



DATA [0] ( <a href="#">COMMANDS RETURN DATA</a> Младший байт)	x	6
DATA [1]	x	7
DATA [2] = ERROR_OR_COMMAND	x	8
DATA [3]	x	9
DATA [4]	x	10
DATA [5]	x	11
DATA [6] ( <a href="#">COMMANDS RETURN DATA</a> Старший байт)	x	12

#### 4.3.3 Команда передачи данных CODE\_CMD\_POWERSTEP01

Команда передачи данных CODE\_CMD\_POWERSTEP01 используется для управления приводом в режиме реального времени (каждая отправленная команда сразу обрабатывается контроллером). Поле DATA пакета содержит структуру [SMSD\\_CMD\\_Type](#), содержащую команду управления. Описание структуры [SMSD\\_CMD\\_Type](#) и команд управления приведены далее в этом руководстве.

В контроллер:

Поле	Значение		Порядок данных в пакете
XOR (1 байт)	x		0
VER (1 байт)	x		1
CMD_TYPE (1 байт)	CODE_CMD_POWERSTEP01 = 0x02		2
CMD_IDENTIFICATION (1 байт)	x		3
LENGTH_DATA (Младший байт)	sizeof( <a href="#">SMSD_CMD_Type</a> )=0x04	0x04	4
LENGTH_DATA (Старший байт)		0x00	5
DATA [0] ( <a href="#">SMSD_CMD_Type</a> Младший байт)	x		6
DATA [1]	x		7
DATA [2]	x		8
DATA [3] ( <a href="#">SMSD_CMD_Type</a> Старший байт)	x		9

В ответ контроллер отправляет пакет с командой CMD\_TYPE = CODE\_CMD\_POWERSTEP01, поле DATA содержит структуру [COMMANDS\\_RETURN\\_DATA](#).



Из контроллера:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_POWERSTEP01 = 0x02	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	sizeof( <a href="#">COMMANDS_RETURN_DATA_Type</a> )	0x07
LENGTH_DATA (Старший байт)		0x00
DATA [0] - Младший байт ( <a href="#">COMMANDS_RETURN_DATA</a> Младший байт)	x	6
DATA [1]	x	7
DATA [2] = ERROR_OR_COMMAND	x	8
DATA [3]	x	9
DATA [4]	x	10
DATA [5]	x	11
DATA [6] - Старший байт ( <a href="#">COMMANDS_RETURN_DATA</a> Старший байт)	x	12

Содержимое структуры [COMMANDS\\_RETURN\\_DATA\\_Type](#) зависит от отправленной пользователем команды.

#### 4.3.4 Команды передачи данных CODE\_CMD\_POWERSTEP01\_W\_MEM0..MEM3

Четыре команды передачи данных - CODE\_CMD\_POWERSTEP01\_W\_MEM0, CODE\_CMD\_POWERSTEP01\_W\_MEM1, CODE\_CMD\_POWERSTEP01\_W\_MEM2, CODE\_CMD\_POWERSTEP01\_W\_MEM3 используются для записи исполнительных программ в соответствующие области памяти контроллера. Поле DATA пакета содержит последовательность исполнительных команд в формате [SMSD\\_CMD\\_Type](#). Максимальное количество команд для записи в контроллер – 255. Кодовое расстояние в адресном пространстве 4 байта.



В контроллер:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_POWERSTEP01_W_MEM0 = 0x03 or CODE_CMD_POWERSTEP01_W_MEM1 = 0x04 or CODE_CMD_POWERSTEP01_W_MEM2 = 0x05 or CODE_CMD_POWERSTEP01_W_MEM3 = 0x06	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	x	4
LENGTH_DATA (Старший байт)	x	5
(1 <sup>я</sup> исполнительная команда) DATA [0] ( <a href="#">SMSD CMD Type</a> Младший байт)	x	6
DATA [1]	x	7
DATA [2]	x	8
(1 <sup>я</sup> исполнительная команда) DATA [3] ( <a href="#">SMSD CMD Type</a> Старший байт)	x	9
.....	.....	.....
(последняя исполнительная команда – всего n команд) DATA [0] ( <a href="#">SMSD CMD Type</a> Младший байт)	x	n*4 - 3
DATA [1]	x	n*4 - 2
DATA [2]	x	n*4 - 1
(последняя исполнительная команда – всего n команд) DATA [3] ( <a href="#">SMSD CMD Type</a> Старший байт)	x	n*4

n<=255.

В ответ на запись программы контроллер отправляет пакет с кодом команды CMD\_TYPE = [CODE\\_CMD\\_RESPONSE](#).

#### 4.3.5 Команды передачи данных CODE\_CMD\_POWERSTEP01\_R\_MEM0..MEM3

Четыре команды передачи данных CODE\_CMD\_POWERSTEP01\_R\_MEM0, CODE\_CMD\_POWERSTEP01\_R\_MEM1, CODE\_CMD\_POWERSTEP01\_R\_MEM2, CODE\_CMD\_POWERSTEP01\_R\_MEM3 предназначены для чтения исполнительных программ из четырех банков памяти контроллера.



В контроллер:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_POWERSTEP01_R_MEM0 = 0x07 or CODE_CMD_POWERSTEP01_R_MEM1 = 0x08 or CODE_CMD_POWERSTEP01_R_MEM2 = 0x09 or CODE_CMD_POWERSTEP01_R_MEM3 = 0x0A	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	0	4
LENGTH_DATA (Старший байт)	0	5
DATA	-	-

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_POWERSTEP01\_R\_MEM0 (or CODE\_CMD\_POWERSTEP01\_R\_MEM1 or CODE\_CMD\_POWERSTEP01\_R\_MEM2 or CODE\_CMD\_POWERSTEP01\_R\_MEM3). Поле DATA пакета содержит последовательность исполнительных команд в формате SMSD\_CMD\_Type. Кодовое расстояние в адресном пространстве 4 байта.

Из контроллера:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	X	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_POWERSTEP01_R_MEM0 = 0x07 or CODE_CMD_POWERSTEP01_R_MEM1 = 0x08 or CODE_CMD_POWERSTEP01_R_MEM2 = 0x09 or CODE_CMD_POWERSTEP01_R_MEM3 = 0x0A	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	x	4
LENGTH_DATA (Старший байт)	x	5
(1 <sup>я</sup> исполнительная команда) DATA [0] ( <a href="#">SMSD_CMD_Type</a> Младший байт)	x	6
DATA [1]	x	7
DATA [2]	x	8
(1 <sup>я</sup> исполнительная команда)	x	9



DATA [3] ( <a href="#">SMSD_CMD_Type</a> Старший байт)		
.....	.....	.....
(последняя исполнительная команда – всего n команд) DATA [0] ( <a href="#">SMSD_CMD_Type</a> Младший байт)	x	n*4 - 3
DATA [1]	x	n*4 - 2
DATA [2]	x	n*4 - 1
(последняя исполнительная команда – всего n команд) DATA [3] ( <a href="#">SMSD_CMD_Type</a> Старший байт)	x	n*4

n<=255.

#### 4.3.6 Команда передачи данных CODE\_CMD\_CONFIG\_SET

Команда передачи данных CODE\_CMD\_CONFIG\_SET предназначена для записи в контроллер параметров подключения по сети Ethernet. Поле DATA пакета содержит структуру [SMSD\\_LAN\\_CONFIG\\_Type](#) (описание структуры далее в руководстве).

В контроллер:

Поле	Значение		Порядок данных в пакете
XOR (1 байт)	x		0
VER (1 байт)	x		1
CMD_TYPE (1 байт)	CODE_CMD_CONFIG_SET = 0x0B		2
CMD_IDENTIFICATION (1 байт)	X		3
LENGTH_DATA (Младший байт)	Sizeof(SMSD_LAN_CONFIG_Type)	0x19	4
LENGTH_DATA (Старший байт)		0x00	5
DATA [0] (SMSD_LAN_CONFIG_Type – Младший байт)	x		6
.....	.....		.....
DATA [24] (SMSD_LAN_CONFIG_Type – Старший байт)	x		30

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = [CODE\\_CMD\\_RESPONSE](#).

#### 4.3.7 Команда передачи данных CODE\_CMD\_CONFIG\_GET

Команда передачи данных CODE\_CMD\_CONFIG\_GET предназначена для чтения из контроллера параметров подключения по сети Ethernet.



В контроллер:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_CONFIG_GET = 0x0C	2
CMD_IDENTIFICATION (1 байт)	X	3
LENGTH_DATA (Младший байт)	0	4
LENGTH_DATA (Старший байт)	0	5
Data	-	-

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_CONFIG\_GET. Поле DATA пакета содержит структуру [SMSD\\_LAN\\_CONFIG\\_Type](#) (описание структуры далее в руководстве).

Из контроллера:

Поле	Значение		Порядок данных в пакете
XOR (1 байт)	x		0
VER (1 байт)	x		1
CMD_TYPE (1 байт)	CODE_CMD_CONFIG_GET = 0x0C		2
CMD_IDENTIFICATION (1 байт)	X		3
LENGTH_DATA (Младший байт)	Sizeof(SMSD_LAN_CONFIG_Type)	0x19	4
LENGTH_DATA (Старший байт)		0x00	5
DATA [0] (SMSD_LAN_CONFIG_Type – Младший байт)	x		6
.....	.....		.....
DATA [24] (SMSD_LAN_CONFIG_Type – Старший байт)	x		30

#### 4.3.8 Команда передачи данных CODE\_CMD\_PASSWORD\_SET

Команда передачи данных CODE\_CMD\_PASSWORD\_SET предназначена для задания нового пароля для авторизации.

В контроллер:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_PASSWORD_SET = 0x0D	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	0x08	4
LENGTH_DATA (Старший байт)	0x00	5
DATA [0] (Password Младший байт)	x	6



DATA [1]	x	7
DATA [2]	x	8
DATA [3]	x	9
DATA [4]	x	10
DATA [5]	x	11
DATA [6]	x	12
DATA [7] (Password Старший байт)	x	13

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = [CODE\\_CMD\\_RESPONSE](#).

#### 4.3.9 Команда передачи данных CODE\_CMD\_ERROR\_GET

Команда передачи данных CODE\_CMD\_ERROR\_GET предназначена для чтения из памяти контроллера информации о количестве включений рабочего режима контроллера и статистики по ошибкам.

В контроллер:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_ERROR_GET = 0x0E	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	0	4
LENGTH_DATA (Старший байт)	0	5
Data	-	-

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_ERROR\_GET.

Из контроллера:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_ERROR_GET = 0x0E	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	0x44 (=17*4)	4
LENGTH_DATA (Старший байт)	0	5
Data[0]	x	6
.....	.....	.....
Data[67]	x	73

Поле DATA содержит 17 последовательных значений 4-х байтовых переменных, представляющих из себя счётчики событий:

**N\_STARTS** – количество раз, когда были запитаны обмотки шагового двигателя.

**ERROR\_XT** – количество внутренних ошибок запуска тактового генератора

**ERROR\_TIME\_OUT** – количество ошибок превышения времени выполнения основного цикла программы.

**ERROR\_INIT\_POWERSTEP01** – количество ошибок инициализации чипа PowerSTEP01.

**ERROR\_INIT\_WIZNET** – количество ошибок инициализации чипа W5500.



**ERROR\_INIT\_FRAM** – количество ошибок инициализации чипа памяти FRAM.  
**ERROR\_SOCKET** – количество ошибок соединений по Ethernet  
**ERROR\_FRAM** – количество ошибок обмена с чипом памяти FRAM.  
**ERROR\_INTERRUPT** – количество ошибок обработки прерывания.  
**ERROR\_EXTERN\_5V** – количество перегрузок по току, внутреннего выходного источника питания в 5В.  
**ERROR\_EXTERN\_VDD** – количество выходов за диапазон питающего напряжения Контроллера.  
**ERROR\_THERMAL\_POWERSTEP01** – количество перегревов чипа PowerSTEP01  
**ERROR\_THERMAL\_BRAKE** – количество перегревов тормозного резистора.  
**ERROR\_COMMAND\_POWERSTEP01** – количество ошибок при передаче управляющих команд в чип PowerSTEP01  
**ERROR\_UVLO\_POWERSTEP01** – количество ошибок  
**ERROR\_STALL\_POWERSTEP01** – количество ошибок  
**ERROR\_WORK\_PROGRAM** – количество ошибок выполнения программы управления.

#### 4.4. Назначение области CMD\_IDENTIFICATION

Поле CMD\_IDENTIFICATION длиной 1 предназначено для однозначной идентификации ответа на отправленную команду. Пользователь должен обеспечить уникальность значения в рамках текущего обмена информацией, хотя бы в пределах нескольких команд.

#### 4.5. Назначение области LENGTH\_DATA

Поле LENGTH\_DATA длиной 2 байта определяет длину информационной части пакета - от 0 до 1024.

#### 4.6. Назначение области DATA[LENGTH\_DATA]

Поле DATA[LENGTH\_DATA] является информационной частью пакета. Длина поля LENGTH\_DATA байт. Структура и размер поля зависят от типа передаваемой команды CMD\_TYPE.

### 5. Структура COMMANDS\_RETURN\_DATA\_Type

В ответ на команды, в поле CMD\_TYPE которых содержится значение CODE\_CMD\_RESPONSE или CODE\_CMD\_POWERSTEP01, контроллер возвращает ответ, в области данных DATA которого содержится структура COMMANDS\_RETURN\_DATA\_Type:

```
typedef struct  
{ powerSTEP_STATUS_TypeDef      STATUS_POWERSTEP01;  
  uint8_t                      ERROR_OR_COMMAND;  
  uint32_t                     RETURN_DATA;  
}COMMANDS_RETURN_DATA_Type;
```

**STATUS\_POWERSTEP01** – битовое поле длиной 16 бит, содержащее флаги состояния, отвечающие за текущее состояние схемы управления шаговым двигателем. Важная информация, поэтому включена в каждый ответ данного формата;

**ERROR\_OR\_COMMAND** – код статуса выполнения команды или код ошибки. Длина - 1 байт.;

**RETURN\_DATA** – поле данных длиной 4 байта.

#### 5.1 Назначение битовых полей STATUS\_POWERSTEP01

Статус состояния процесса управления шаговым двигателем содержится в структуре powerSTEP\_STATUS\_TypeDef:



```
typedef struct {
uint16_t      HiZ           : 1;
uint16_t      BUSY          : 1;
uint16_t      SW_F          : 1;
uint16_t      SW_EVN        : 1;
uint16_t      DIR           : 1;
uint16_t      MOT_STATUS    : 2;
uint16_t      CMD_ERROR     : 1;
uint16_t      RESERVE       : 8;
} powerSTEP_STATUS_TypeDef;
```

**HiZ** – Z-состояние обмоток, если 1- обмотки шагового двигателя обесточены, 0 – обмотки шагового двигателя запитаны.

**BUSY** – ожидание, если 1- блок готов к выполнению следующей команды, 0 – выполняется предыдущая команда.

**SW\_F** – если 1- функция SW включена, 0 – функция SW выключена.

**SW\_EVN** – флаг события SW если 1- событие наступило, 0 – событие не наступило.

**DIR** – направление вращения, если 1- основное направление, 0 – обратное направление.

**MOT\_STATUS** – статус текущего действия, если 0 – двигатель остановлен, 1 – ускорение, 2 - торможение, 3 - равномерное вращение шагового двигателя.

**CMD\_ERROR** – ошибка выполнения команды, если 1- ошибка выполнения команды, 0 – без ошибок.

## 5.2 Список возможных значений поля ERROR\_OR\_COMMAND

Числовые значения начинаются с 0 и последовательно инкрементируются. Список возможных вариантов значений поля ERROR\_OR\_COMMAND:

<b>OK</b>	- без ошибок
<b>OK_ACCESS</b>	- признак получения доступа к управлению Контроллером
<b>ERROR_ACCESS</b>	- признак ошибки получения доступа к управлению Контроллером
<b>ERROR_ACCESS_TIMEOUT</b>	- признак того, что не истёк таймаут для повторной авторизации (1 сек)
<b>ERROR_XOR</b>	- ошибка контрольной суммы команды
<b>ERROR_NO_COMMAND</b>	- признак того, что такой команды не существует
<b>ERROR_LEN</b>	- ошибочная длина пакета
<b>ERROR_RANGE</b>	- выход за допустимый диапазон значений
<b>ERROR_WRITE</b>	- ошибка записи
<b>ERROR_READ</b>	- ошибка чтения
<b>ERROR_PROGRAMS</b>	- для внутреннего пользования
<b>ERROR_WRITE_SETUP</b>	- для внутреннего пользования
<b>NO_NEXT</b>	- для внутреннего пользования
<b>END_PROGRAMS</b>	- конец программы
<b>COMMAND_GET_STATUS_IN_EVENT</b>	- в поле данных RETURN_DATA содержится битовая карта входных сигналов
<b>COMMAND_GET_MODE</b>	- в поле данных RETURN_DATA содержится битовая карта текущих настроек Контроллера
<b>COMMAND_GET_ABS_POS</b>	- в поле данных RETURN_DATA содержится текущее положение шагового двигателя в шагах
<b>COMMAND_GET_EL_POS</b>	- в поле данных RETURN_DATA содержится текущее электрическое положение двигателя
<b>COMMAND_GET_SPEED</b>	- в поле данных RETURN_DATA содержится текущая скорость двигателя
<b>COMMAND_GET_MIN_SPEED</b>	- в поле данных RETURN_DATA содержится текущая установленная минимальная скорость двигателя
<b>COMMAND_GET_MAX_SPEED</b>	- в поле данных RETURN_DATA содержится текущая установленная максимальная скорость двигателя
<b>COMMAND_GET_STACK</b>	- в поле данных RETURN_DATA содержится информация о номере текущей выполняемой программы и номер выполняемой команды
<b>STATUS_RELE_SET</b>	– реле включено
<b>STATUS_RELE_CLR</b>	– реле выключено



## 6. Структура исполнительных команды управления SMSD\_CMD\_Type

Структура исполнительных команды управления SMSD\_CMD\_Type:

```
typedef struct
{ uint32_t      RESERVE   :3;
  uint32_t      ACTION    :1;
  uint32_t      COMMAND   :6;
  uint32_t      DATA     :22;
}SMSD_CMD_Type;
```

**RESERVE** – 3 бита, не используется;

**ACTION** - 1 бит = 0 – для внутреннего использования;

**COMMAND** - 6 бит – код исполнительной команды;

**DATA** - 22 бита – параметр команды, если исполнительная команда не требует параметра, значение данного поля = 0x00 (22 бита заполняются 0 и отправляются не изменяя длины поля).

Размер структуры – 4 байта.

Структура SMSD\_CMD\_Type используется в пакетах, содержащих команды передачи данных CMD\_TYPE = CODE\_CMD\_POWERSTEP01, CODE\_CMD\_POWERSTEP01\_W\_MEM0...MEM3, CODE\_CMD\_POWERSTEP01\_R\_MEM0...MEM3.

Числовые значения поля COMMAND начинаются с 0 и последовательно инкрементируются. Список возможных вариантов значений поля:

0x00	CMD_PowerSTEP01_END,
0x01	CMD_PowerSTEP01_GET_SPEED,
0x02	CMD_PowerSTEP01_STATUS_IN_EVENT,
0x03	CMD_PowerSTEP01_SET_MODE,
0x04	CMD_PowerSTEP01_GET_MODE,
0x05	CMD_PowerSTEP01_SET_MIN_SPEED,
0x06	CMD_PowerSTEP01_SET_MAX_SPEED,
0x07	CMD_PowerSTEP01_SET_ACC,
0x08	CMD_PowerSTEP01_SET_DEC,
0x09	CMD_PowerSTEP01_SET_FS_SPEED,
0x0A	CMD_PowerSTEP01_SET_MASK_EVENT
0x0B	CMD_PowerSTEP01_GET_ABS_POS,
0x0C	CMD_PowerSTEP01_GET_EL_POS,
0x0D	CMD_PowerSTEP01_GET_STATUS_AND_CLR,
0x0E	CMD_PowerSTEP01_RUN_F,
0x0F	CMD_PowerSTEP01_RUN_R,
0x10	CMD_PowerSTEP01_MOVE_F,
0x11	CMD_PowerSTEP01_MOVE_R,
0x12	CMD_PowerSTEP01_GO_TO_F,
0x13	CMD_PowerSTEP01_GO_TO_R,
0x14	CMD_PowerSTEP01_GO_UNTIL_F,
0x15	CMD_PowerSTEP01_GO_UNTIL_R,
0x16	CMD_PowerSTEP01_SCAN_ZERO_F,
0x17	CMD_PowerSTEP01_SCAN_ZERO_R,
0x18	CMD_PowerSTEP01_SCAN_LABEL_F,
0x19	CMD_PowerSTEP01_SCAN_LABEL_R,
0x1A	CMD_PowerSTEP01_GO_ZERO,
0x1B	CMD_PowerSTEP01_GO_LABEL,
0x1C	CMD_PowerSTEP01_GO_TO,
0x1D	CMD_PowerSTEP01_RESET_POS,
0x1E	CMD_PowerSTEP01_RESET_POWERSTEP01,
0x1F	CMD_PowerSTEP01_SOFT_STOP,
0x20	CMD_PowerSTEP01_HARD_STOP,
0x21	CMD_PowerSTEP01_SOFT_HI_Z,
0x22	CMD_PowerSTEP01_HARD_HI_Z,



```

0x23 CMD_PowerSTEP01_SET_WAIT,
0x24 CMD_PowerSTEP01_SET_RELE,
0x25 CMD_PowerSTEP01_CLR_RELE,
0x26 CMD_PowerSTEP01_GET_RELE,
0x27 CMD_PowerSTEP01_WAIT_IN0,
0x28 CMD_PowerSTEP01_WAIT_IN1,
0x29 CMD_PowerSTEP01_GOTO_PROGRAM,
0x2A CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN0,
0x2B CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN1,
0x2C CMD_PowerSTEP01_LOOP_PROGRAM,
0x2D CMD_PowerSTEP01_CALL_PROGRAM
0x2E CMD_PowerSTEP01_RETURN_PROGRAM,
0x2F CMD_PowerSTEP01_START_PROGRAM_MEM0,
0x30 CMD_PowerSTEP01_START_PROGRAM_MEM1,
0x31 CMD_PowerSTEP01_START_PROGRAM_MEM2,
0x32 CMD_PowerSTEP01_START_PROGRAM_MEM3,
0x33 CMD_PowerSTEP01_STOP_PROGRAM_MEM,
0x34 CMD_PowerSTEP01_STEP_CLOCK,
0x35 CMD_PowerSTEP01_STOP_USB,
0x36 CMD_PowerSTEP01_GET_MIN_SPEED,
0x37 CMD_PowerSTEP01_GET_MAX_SPEED,
0x38 CMD_PowerSTEP01_GET_STACK,
0x39 CMD_PowerSTEP01_GOTO_PROGRAM_IF_ZERO,
0x3A CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN_ZERO,
0x3B CMD_PowerSTEP01_WAIT_CONTINUE,
0x3C CMD_PowerSTEP01_SET_WAIT_2,
0x3D CMD_PowerSTEP01_SCAN_MARK2_F,
0x3E CMD_PowerSTEP01_SCAN_MARK2_R

```

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Byte [2]			Byte [1]			Byte [0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data (параметр команды)						Команда (код команды CMD_PowerSTEP01)				Action				Reserve		

### 6.1 Исполнительная команда CMD\_PowerSTEP01\_END

Исполнительная команда CMD\_PowerSTEP01\_END = 0x00 предназначена для обозначения конца программы. Используется в конце программы, записываемой в память контроллера.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_END = 0x00				Action				Reserve		
Значение	0						0	0	0	0	0	0	0	0	0	0	0

### 6.2 Исполнительная команда CMD\_PowerSTEP01\_GET\_SPEED

Исполнительная команда CMD\_PowerSTEP01\_GET\_SPEED = 0x01 предназначена для чтения текущего значения скорости.

**Важное замечание:** для корректного ответа контроллера о текущей скорости перед запросом CMD\_PowerSTEP01\_GET\_SPEED должна быть установлена минимальная скорость двигателя - команда CMD\_PowerSTEP01\_SET\_MIN\_SPEED = 0x00. В противном случае контроллер может выдавать неверные значение для низких скоростей и в случае остановки двигателя.

Ниже дан пример пакета передачи данных для запроса текущей скорости:



В контроллер:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_POWERSTEP01 = 0x02	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	Sizeof ( <a href="#">SMSD_CMD_Type</a> )= 0x04	4
LENGTH_DATA (Старший байт)	0x00	5
DATA [0] ( <a href="#">SMSD_CMD_Type</a> Младший байт)	0x10	6
DATA [1]	0x00	7
DATA [2]	0x00	8
DATA [3] ( <a href="#">SMSD_CMD_Type</a> Старший байт)	0x00	9

Поле DATA пакета содержит структуру SMSD\_CMD\_Type, поле command которой содержит код команды запроса скорости CMD\_PowerSTEP01\_GET\_SPEED.

Битовая раскладка структуры SMSD\_CMD\_Type:

Поле DATA	DATA[3]=0x00			DATA[2]=0x00			DATA[1]=0x00			DATA[0]=0x10							
	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = 0x00						Command CMD_PowerSTEP01_GET_SPEED = 0x01							Action	Reserve		
Значение	0						0	0	0	0	0	0	1	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = COMMAND\_GET\_SPEED, RETURN\_DATA - значение текущей скорости двигателя.



Из контроллера:

Поле	Значение	Порядок данных в пакете
XOR (1 байт)	x	0
VER (1 байт)	x	1
CMD_TYPE (1 байт)	CODE_CMD_RESPONSE = 0x01	2
CMD_IDENTIFICATION (1 байт)	x	3
LENGTH_DATA (Младший байт)	Sizeof	0x07
LENGTH_DATA (Старший байт)	(COMMANDS_RETURN_DATA_Type)	0x00
DATA [0] - Младший байт (COMMANDS_RETURN_DATA Младший байт)	x	6
DATA [1]	x	7
DATA [2] = ERROR_OR_COMMAND	= COMMAND_GET_SPEED	8
DATA [3] = RETURN_DATA[0]	(Младший байт значения текущей скорости)	9
DATA [4] = RETURN_DATA[1]	x	10
DATA [5] = RETURN_DATA[2]	x	11
DATA [6] - Старший байт (COMMANDS_RETURN_DATA Старший байт) = RETURN_DATA[3]	(Старший байт значения текущей скорости)	12

### 6.3 Исполнительная команда CMD\_PowerSTEP01\_STATUS\_IN\_EVENT

Исполнительная команда CMD\_PowerSTEP01\_STATUS\_IN\_EVENT = 0x02 предназначена для чтения текущего состояния входных сигналов.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = 0x00						Command CMD_PowerSTEP01_ STATUS_IN_EVENT = 0x02						Action	Reserve			
Значение	0			0			0	0	0	0	0	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = COMMAND\_GET\_STATUS\_IN\_EVENT, RETURN\_DATA – битовую карту состояний входных сигналов:

	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
RETURN_DATA[0]	INT_7	INT_6	INT_5	INT_4	INT_3	INT_2	INT_1	INT_0
RETURN_DATA[1]	Mask_7	Mask_6	Mask_5	Mask_4	Mask_3	Mask_2	Mask_1	Mask_0
RETURN_DATA[2]	Wait_7	Wait_6	Wait_5	Wait_4	Wait_3	Wait_2	Wait_1	Wait_0
RETURN_DATA[3]	Not use							

INT\_X – событие на входном сигнале X;  
Mask\_X – Маскирование входного сигнала X;  
Wait\_X – Ожидание входного сигнала X.

### 6.4 Исполнительная команда CMD\_PowerSTEP01\_SET\_MODE

Исполнительная команда CMD\_PowerSTEP01\_SET\_MODE = 0x03 предназначена для установки параметров управления двигателем.



Битовая раскладка структуры SMSD CMD Type:

	Байт[3]			Byte [2]			Byte [1]			Byte [0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Поле Data структуры SMSD_CMD_Type						CommandCMD_PowerSTEP01_SET_M ODE = 0x03						Action		Reserve		
Значение	Зависит от значения поля Data						0	0	0	0	1	1	0	0	0	0	

Битовая раскладка поля Data структуры SMSD\_CMD\_Type:

Байт[3] – биты 7..0								Байт[2] – биты 7..0								Байт[1] биты 7..2						
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
			STOP_CURRENT		WORK_CURRENT								MICROSTEPPING				MOTOR_TYPE					CURRENT_OR_VOLTAGE

CURRENT OR VOLTAGE - тип управления двигателем:

0 – напряжение (вольтовый режим),

1 – ток (токовый режим)

MOTOR TYPE – модель двигателя для вольтового режима управления:

Значение		Макс. ток фазы, А	Сопроти- вление фазы, Ом	Индуктив- ность фазы, мГн	Угло- вой шаг	Модель
SMSP-4.2LAN	SMSP-8.0LAN					
0	0	-	-	-	-	нет
1	1	1.33	2.1	2.5	1.8	FL42STH33-1334 1.8 deg
2	2	1.33	2.1	4.2	0.9	FL42STH33-1334 0.9 deg
3	3	1.2	3.3	3.4	0.9	FL42STH38-1206 0.9 deg
4	4	1.68	1.65	3.2	1.8	FL42STH38-1684 1.8 deg
5	5	1.68	1.64	3.2	0.9	FL42STH38-1684 0.9 deg
6	6	1.2	3.3	2.8	0.8	FL42STH47-1206 1.8 deg
7	7	1.68	1.65	2.8	1.8	FL42STH47-1684 1.8 deg
8	8	1.68	1.65	4.1	0.9	FL42STH47-1684 0.9 deg
9	9	1.2	6	7	1.8	FL42STH60-1206 1.8 deg
10	10	1.2	12.1	36.7	0.9	FL42STH60-1206 0.9 deg
11	11	1.56	1.8	3.6	1.8	FL57ST41-1564 1.8 deg
12	12	1.0	16.7	46.5	1.8	FL57ST76-1006 1.8 deg
13	13	1.5	3.6	6	1.8	FL57ST76-1506 1.8 deg
14	14	1.0	5.7	5.4	1.8	FL57STH41-1006 1.8 deg
15	15	1.0	5.7	8	0.9	FL57STH41-1006 0.9 deg
16	16	2.8	0.7	1.4	1.8	FL57STH41-2804 1.8 deg
17	17	2.8	0.7	2.2	0.9	FL57STH41-2804 0.9 deg
18	18	1.0	6.6	8.6	1.8	FL57STH51-1006 1.8 deg
19	19	2.8	0.83	2.2	1.8	FL57STH51-2804 1.8 deg
20	20	2.8	0.9	3.7	0.9	FL57STH51-2804 0.9 deg
21	21	1.0	7.4	10	1.8	FL57STH56-1006 1.8 deg
22	22	2.0	1.8	2.5	1.8	FL57STH56-2006 1.8 deg
23	23	2.8	0.9	2.5	1.8	FL57STH56-2804 1.8 deg
24	24	1.0	8.6	14	1.8	FL57STH76-1006 1.8 deg



25	25	2.8	1.13	3.6	1.8	FL57STH76-2804 1.8 deg
26	26	2.8	1.13	5.6	0.9	FL57STH76-2804 0.9 deg
27	27	2.0	1.2	4.6	1.8	FL60STH65-2008 1.8 deg параллельное соединение обмоток
28	28	2.0	4.8	18.4	1.8	FL60STH65-2008 1.8 deg последовательное соединение обмоток
29	29	2.0	1.5	6.8	1.8	FL60STH86-2008 1.8 deg параллельное соединение обмоток
30	30	2.0	6	7.2	1.8	FL60STH86-2008 1.8 deg последовательное соединение обмоток
31	31	2.8	0.7	3.9	1.8	FL86STH65-2808 1.8 deg параллельное соединение обмоток
32	32	2.8	2.8	15.6	1.8	FL86STH65-2808 1.8 deg последовательное соединение обмоток
33	33	4.2	0,375	3.4	1.8	FL86STH80-4208 1.8 deg параллельное соединение обмоток
34	34	4.2	1.5	13.6	1.8	FL86STH80-4208 1.8 deg последовательное соединение обмоток
35	35	4.2	0.45	6	1.8	FL86STH118-4208 1.8 deg параллельное соединение обмоток
36	36	4.2	1.8	24	1.8	FL86STH118-4208 1.8 deg последовательное соединение обмоток
37	37	4.2	0,625	8	1.8	FL86STH156-4208 1.8 deg параллельное соединение обмоток
38	38	4.2	2.5	32	1.8	FL86STH156-4208 1.8 deg последовательное соединение обмоток
-	39	6.0	0.6	6.5	1.8	FL86STH118-6004 1.8 deg
-	40	6.2	0.75	9	1.8	FL86STH156-6204 1.8 deg
-	41	5.5	0.9	12	1.8	FL110STH99-5504 1.8 deg
-	42	6.5	0.8	15	1.8	FL110STH150-6504 1.8 deg
-	43	8	0.67	12	1.8	FL110STH201-8004 1.8 deg
39	44	0.3	32	40	1.8	ДШ3934-0,3А
40	45	0.67	8.5	7.5	1.8	ДШ2851-0,7А
41	46	1.68	2.3	3.4	1.8	ДШ4248-1,7А
42	47	3.0	1.0	3.4	1.8	ДШ5776-3,0А
43	48	3.0	1.45	6.5	1.8	ДШ57112-3,0А
44	49	3.0	1.2	6.4	1.8	ДШ8665-3,0А
45	50	4.5	0.36	3.0	1.8	ДШ8682-4,5А
-	51	6.0	0.6	5.7	1.8	ДШ86118-6,0А
-	52	6.2	0.7	8.5	1.8	ДШ86156-6,2А
-	53	8.0	0.8	16	1.8	ДШ110201-8,0А
-	54	6.0	0.8	8.7	1.8	ДШ130280-6,0А

MICROSTEPPING – дробление шага:

- 0 - 1
- 1 - 1/2
- 2 - 1/4
- 3 - 1/8
- 4 - 1/16
- 5 - 1/32
- 6 - 1/64
- 7 - 1/128

WORK CURRENT – рабочий ток (используется в токовом режиме управления двигателем). Значение рабочего тока определяется по формуле:  $0.1A \cdot \text{Значение параметра}$ ;  $1 \leq \text{Значение} \leq 80$ . Значения тока могут быть следующие:

1 - 0.1A	15 - 1.5A	29 - 2.9A	43 - 4.3A	57 - 5.7A	71 - 7.1A
2 - 0.2A	16 - 1.6A	30 - 3.0A	44 - 4.4A	58 - 5.8A	72 - 7.2A
3 - 0.3A	17 - 1.7A	31 - 3.1A	45 - 4.5A	59 - 5.9A	73 - 7.3A
4 - 0.4A	18 - 1.8A	32 - 3.2A	46 - 4.6A	60 - 6.0A	74 - 7.4A
5 - 0.5A	19 - 1.9A	33 - 3.3A	47 - 4.7A	61 - 6.1A	75 - 7.5A
6 - 0.6A	20 - 2.0A	34 - 3.4A	48 - 4.8A	62 - 6.2A	76 - 7.6A
7 - 0.7A	21 - 2.1A	35 - 3.5A	49 - 4.9A	63 - 6.3A	77 - 7.7A
8 - 0.8A	22 - 2.2A	36 - 3.6A	50 - 5.0A	64 - 6.4A	78 - 7.8A
9 - 0.9A	23 - 2.3A	37 - 3.7A	51 - 5.1A	65 - 6.5A	79 - 7.9A
10 - 1.0A	24 - 2.4A	38 - 3.8A	52 - 5.2A	66 - 6.6A	80 - 8.0A
11 - 1.1A	25 - 2.5A	39 - 3.9A	53 - 5.3A	67 - 6.7A	
12 - 1.2A	26 - 2.6A	40 - 4.0A	54 - 5.4A	68 - 6.8A	
13 - 1.3A	27 - 2.7A	41 - 4.1A	55 - 5.5A	69 - 6.9A	
14 - 1.4A	28 - 2.8A	42 - 4.2A	56 - 5.6A	70 - 7.0A	

Допустимый диапазон значений для блоков SMSD-4.2LAN: 1 – 42; для блоков SMSD-8.0LAN: 1 – 80.

STOP CURRENT – ток удержания – устанавливается в процентах от значения рабочего тока:

- 0 - 25%
- 1 - 50%
- 2 - 75%
- 3 - 100%

В ответ контроллер отправляет пакет с кодом команды `CMD_TYPE = CODE_CMD_RESPONSE`, поле `DATA` которого содержит структуру `COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK`.

## 6.5 Исполнительная команда `CMD_PowerSTEP01_GET_MODE`

Исполнительная команда `CMD_PowerSTEP01_GET_MODE = 0x04` предназначена для чтения настроек управления двигателем.

Битовая раскладка структуры `SMSD_CMD_Type`:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = 0x00						Command CMD_PowerSTEP01_ GET_MODE = 0x04						Action	Reserve			
Значение	0			0			0	0	0	0	1	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды `CMD_TYPE = CODE_CMD_RESPONSE`, поле `DATA` которого содержит структуру `COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_MODE`, `RETURN_DATA` содержит информацию о настройках управления двигателем.



RETURN_DATA[3]	RETURN_DATA[2]						RETURN_DATA[1]			RETURN_DATA[0]		
0..7	5..7	4	3	2	1	0	2..7	1	0	7	1..6	0
Не используется	PROGRAM_N		STOP_CURRENT		WORK_CURRENT		MICRO-STEPPING		MOTOR_TYPE		CURRENT_OR_VOLTAGE	

Назначения полей STOP\_CURRENT, WORK\_CURRENT, MICROSTEPPING, MOTOR\_TYPE, CURRENT\_OR\_VOLTAGE такие же, как для команды установки настроек CMD\_PowerSTEP01\_SET\_MODE.

Поле PROGRAM\_N содержит номер программы для старта внешними сигналами.

### 6.6 Исполнительная команда CMD\_PowerSTEP01\_SET\_MIN\_SPEED

Исполнительная команда CMD\_PowerSTEP01\_SET\_MIN\_SPEED = 0x05 предназначена для установки минимальной скорости вращения двигателя. Поле DATA должно содержать значение скорости в диапазоне от 0 до 950 шагов/сек.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = Значение минимальной скорости							Command CMD_PowerSTEP01_ SET_MIN_SPEED = 0x05							Action	Reserve		
Значение	Зависит от значения поля Data							0	0	0	1	0	1	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.7 Исполнительная команда CMD\_PowerSTEP01\_SET\_MAX\_SPEED

Исполнительная команда CMD\_PowerSTEP01\_SET\_MAX\_SPEED = 0x06 предназначена для установки максимальной скорости шагового двигателя. Поле DATA должно содержать значение скорости в диапазоне от 16 до 15600 шагов/сек.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = Значение максимальной скорости							Command CMD_PowerSTEP01_ SET_MAX_SPEED = 0x06							Action	Reserve		
Значение	Зависит от значения поля Data							0	0	0	1	1	0	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.



### 6.8 Исполнительная команда CMD\_PowerSTEP01\_SET\_ACC

Исполнительная команда CMD\_PowerSTEP01\_SET\_ACC = 0x07 предназначена для установки значения ускорения двигателя. Поле DATA должно содержать значение ускорения в диапазоне от 15 до 59000 шагов/сек<sup>2</sup>.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = Ускорения							Command CMD_PowerSTEP01_SET_ACC = 0x07							Action	Reserve		
Значение	Зависит от значения поля Data							0	0	0	1	1	1	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.9 Исполнительная команда CMD\_PowerSTEP01\_SET\_DEC

Исполнительная команда CMD\_PowerSTEP01\_SET\_DEC = 0x08 предназначена для установки значения замедления шагового двигателя. Поле DATA должно содержать значение замедления в диапазоне от 15 до 59000 шагов/сек<sup>2</sup>.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = Замедление						Command CMD_PowerSTEP01_SET_DEC = 0x08							Action	Reserve			
Значение	Зависит от значения поля Data						0	0	01	0	0	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.10 Исполнительная команда CMD\_PowerSTEP01\_SET\_FS\_SPEED

Исполнительная команда CMD\_PowerSTEP01\_SET\_FS\_SPEED = 0x09 предназначена для установки скорости перехода на полношаговый режим работы. Поле DATA должно содержать скорость перехода в полношаговый режим в диапазоне от 15 до 15600 шагов/сек.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = Значение скорости перехода в полношаговый режим							Command CMD_PowerSTEP01_SET_FS_SPE ED = 0x09							Action	Reserve		
Значение	Зависит от значения поля Data							0	0	01	0	0	1	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.11 Исполнительная команда CMD\_PowerSTEP01\_SET\_MASK\_EVENT

Исполнительная команда CMD\_PowerSTEP01\_SET\_MASK\_EVENT = 0x0A предназначена для маскирования входных сигналов. В случае, если значение маска сигнала установлено 1 – контроллер

обрабатывает сигналы на соответствующем физическом входе. Если значение маски сигнала установлено 0 – контроллер не обрабатывает сигналы на соответствующем физическом входе.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Маска сигналов						Command CMD_PowerSTEP01_ SET_MASK_EVENT = 0x0A						Action		Reserve		
Значение	Зависит от значения поля Data						0	0	01	0	1	0	0	0	0	0	0

Битовая раскладка поля Data:

	Байт[3] bits 7..0								Байт[2] bits 7..0								Байт[1] bits 7..2					
Бит	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Mask_7	Mask_6	Mask_5	Mask_4	Mask_3	Mask_2	Mask_1	Mask_0

Mask\_X – Маскирование сигнала на входе X.

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.12 Исполнительная команда CMD\_PowerSTEP01\_GET\_ABS\_POS

Исполнительная команда CMD\_PowerSTEP01\_GET\_ABS\_POS = 0x0B предназначена для чтения положения двигателя.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = 0x00						Command CMD_PowerSTEP01_ GET_ABS_POS = 0x0B						Action		Reserve		
Значение	0						0	0	01	0	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = COMMAND\_GET\_ABS\_POS, RETURN\_DATA содержит значение текущего положения двигателя в диапазоне – (2<sup>21</sup>)...(2<sup>21</sup>-1).

## 6.13 Исполнительная команда CMD\_PowerSTEP01\_GET\_EL\_POS

Исполнительная команда CMD\_PowerSTEP01\_GET\_EL\_POS = 0x0C предназначена для чтения электрического положения ротора двигателя.



Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = 0x00						Command CMD_PowerSTEP01_ GET_EL_POS = 0x0C							Action	Reserve		
Значение	0						0	0	01	1	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = COMMAND\_GET\_EL\_POS, RETURN\_DATA содержит информацию о текущем электрическом положении ротора: биты 8,7 – текущий шаг, bits 6..0 – текущий микрошаг в пределах полного шага (измеряется как 1/128 от величина полного шага).

RETURN_DATA[3]								RETURN_DATA[2]								RETURN_DATA[1]								RETURN_DATA[0]							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Не используется																Текущий шаг				Текущий микрошаг											

#### 6.14 Исполнительная команда CMD\_PowerSTEP01\_GET\_STATUS\_AND\_CLR

Исполнительная команда CMD\_PowerSTEP01\_GET\_STATUS\_AND\_CLR = 0x0D предназначена для чтения текущего статуса контроллера и сброса всех флагов ошибок.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = 0x00						Command CMD_PowerSTEP01_ GET_STATUS_AND_CLR = 0x0D							Action	Reserve			
Значение	0						0	0	01	1	0	1	0	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.15 Исполнительная команда CMD\_PowerSTEP01\_RUN\_F

Исполнительная команда CMD\_PowerSTEP01\_RUN\_F = 0x0E предназначена для старта непрерывного вращения двигателя в прямом направлении на указанной скорости. Поле DATA содержит значение скорости вращения в диапазоне от 15 до 15600 шагов/сек.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Значение скорости вращения						Command CMD_PowerSTEP01_ RUN_F = 0x0E							Action	Reserve		
Значение	Зависит от значения поля Data						0	0	01	1	1	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.16 Исполнительная команда CMD\_PowerSTEP01\_RUN\_R

Исполнительная команда CMD\_PowerSTEP01\_RUN\_R = 0x0F предназначена для старта непрерывного вращения двигателя в обратном направлении на указанной скорости. Поле DATA содержит значение скорости вращения в диапазоне от 15 до 15600 шагов/сек.



Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Значение скорости вращения						Command CMD_PowerSTEP01_ RUN_R = 0x0F						Action	Reserve			
Значение	Зависит от значения поля Data						0	0	01	1	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.17 Исполнительная команда CMD\_PowerSTEP01\_MOVE\_F

Исполнительная команда CMD\_PowerSTEP01\_MOVE\_F = 0x10 предназначена для перемещения двигателя в прямом направлении на указанную величину. Поле DATA должно содержать величину перемещения в диапазоне  $-(2^{21}) \dots + (2^{21}-1)$ . Скорость перемещения определяется предварительно заданными параметрами минимальной скорости, максимальной скорости, ускорения и замедления. Перед отправкой данной команды двигатель должен быть остановлен (поле Mot\_Status структуры powerSTEP\_STATUS\_Type = 0).

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.  
В командах задания перемещения единицы измерения параметра - микрошаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Перемещение						Command CMD_PowerSTEP01_ MOVE_F = 0x10						Action	Reserve			
Значение	Зависит от значения поля Data						0	1	0	0	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.18 Исполнительная команда CMD\_PowerSTEP01\_MOVE\_R

Исполнительная команда CMD\_PowerSTEP01\_MOVE\_F = 0x10 предназначена для перемещения двигателя в обратном направлении на указанную величину. Поле DATA должно содержать величину перемещения в диапазоне  $-(2^{21}) \dots + (2^{21}-1)$ . Скорость перемещения определяется предварительно заданными параметрами минимальной скорости, максимальной скорости, ускорения и замедления. Перед отправкой данной команды двигатель должен быть остановлен (поле Mot\_Status структуры powerSTEP\_STATUS\_Type = 0).

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.  
В командах задания перемещения единицы измерения параметра - микрошаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Перемещение						Command CMD_PowerSTEP01_ MOVE_R = 0x11						Action	Reserve			
Значение	Зависит от значения поля Data						0	1	0	0	0	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.19 Исполнительная команда **CMD\_PowerSTEP01\_GO\_TO\_F**

Исполнительная команда **CMD\_PowerSTEP01\_GO\_TO\_F = 0x12** предназначена для перемещения в заданную позицию в прямом направлении. Поле DATA должно содержать требуемое положение двигателя в диапазоне  $-(2^{21}) \dots +(2^{21}-1)$ . Скорость перемещения определяется предварительно заданными параметрами минимальной скорости, максимальной скорости, ускорения и замедления.

**Внимание:** в командах установки скорости единицы измерения параметра - полные шаги в секунду. В командах задания перемещения единицы измерения параметра - микрошаги в секунду.

Битовая раскладка структуры **SMSD\_CMD\_Type**:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Требуемое положение двигателя						Command CMD_PowerSTEP01_ GO_TO_F = 0x12						Action	Reserve			
Значение	Зависит от значения поля Data						0	1	0	0	1	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды **CMD\_TYPE = CODE\_CMD\_RESPONSE**, поле DATA которого содержит структуру **COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK**.

### 6.20 Исполнительная команда **CMD\_PowerSTEP01\_GO\_TO\_R**

Исполнительная команда **CMD\_PowerSTEP01\_GO\_TO\_R = 0x13** для перемещения в заданную позицию в обратном направлении. Поле DATA должно содержать требуемое положение двигателя в диапазоне  $-(2^{21}) \dots +(2^{21}-1)$ . Скорость перемещения определяется предварительно заданными параметрами минимальной скорости, максимальной скорости, ускорения и замедления.

**Внимание:** в командах установки скорости единицы измерения параметра - полные шаги в секунду. В командах задания перемещения единицы измерения параметра - микрошаги в секунду.

Битовая раскладка структуры **SMSD\_CMD\_Type**:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Требуемое положение двигателя						Command CMD_PowerSTEP01_ GO_TO_R = 0x13						Action	Reserve			
Значение	Зависит от значения поля Data						0	1	0	0	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды **CMD\_TYPE = CODE\_CMD\_RESPONSE**, поле DATA которого содержит структуру **COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK**.

### 6.21 Исполнительная команда **CMD\_PowerSTEP01\_GO\_UNTIL\_F**

Исполнительная команда **CMD\_PowerSTEP01\_GO\_UNTIL\_F = 0x14** предназначена для старта вращения двигателя в прямом направлении на максимальной скорости до получения сигнала на вход, номер которого задан в поле DATA. После получения сигнала двигатель останавливается с заданным замедлением. При отработке команды учитывается заданная маска сигнала. Маскирование сигналов может быть изменено командой **CMD\_PowerSTEP01\_SET\_MASK\_EVENT**.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Номер сигнала						Command CMD_PowerSTEP01_ GO_UNTIL_F = 0x14						Action	Reserve			
Значение	Зависит от значения поля Data						0	1	0	1	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.22 Исполнительная команда CMD\_PowerSTEP01\_GO\_UNTIL\_R

Исполнительная команда CMD\_PowerSTEP01\_GO\_UNTIL\_F = 0x14 предназначена для старта вращения двигателя в обратном направлении на максимальной скорости до получения сигнала на вход, номер которого задан в поле DATA. После получения сигнала двигатель останавливается с заданным замедлением. При отработке команды учитывается заданная маска сигнала. Маскирование сигналов может быть изменено командой CMD\_PowerSTEP01\_SET\_MASK\_EVENT.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Номер сигнала						Command CMD_PowerSTEP01_ GO_UNTIL_R = 0x15						Action	Reserve			
Значение	Зависит от значения поля Data						0	1	0	1	0	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.23 Исполнительная команда CMD\_PowerSTEP01\_SCAN\_ZERO\_F

Исполнительная команда CMD\_PowerSTEP01\_SCAN\_ZERO\_F = 0x16 предназначена для поиска нулевого положения в прямом направлении с заданной скоростью. Движение продолжается до поступления сигнала на вход SET\_ZERO. При поступлении сигнала двигатель останавливается, текущее положение принимается за нулевое. Поле DATA должно содержать скорость движения при поиске нулевого положения.  
Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Скорость поиска нулевого положения						Command CMD_PowerSTEP01_ SCAN_ZERO_F = 0x16						Action	Reserve			
Значение	Зависит от значения поля Data						0	1	0	1	1	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.24 Исполнительная команда CMD\_PowerSTEP01\_SCAN\_ZERO\_R

Исполнительная команда CMD\_PowerSTEP01\_SCAN\_ZERO\_R = 0x17 предназначена для поиска нулевого положения в обратном направлении с заданной скоростью. Движение продолжается до поступления сигнала на вход SET\_ZERO. При поступлении сигнала двигатель останавливается, текущее положение принимается за нулевое. Поле DATA должно содержать скорость движения при поиске нулевого положения.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Скорость поиска нулевого положения						Command CMD_PowerSTEP01_ SCAN_ZERO_F = 0x17							Action	Reserve		
Значение	Зависит от значения поля Data						0	1	0	1	1	1	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.25 Исполнительная команда CMD\_PowerSTEP01\_SCAN\_LABEL\_F

Исполнительная команда CMD\_PowerSTEP01\_SCAN\_LABEL\_F = 0x18 предназначена для поиска метки положения в прямом направлении. Движение продолжается до поступления сигнала на вход IN1. При поступлении сигнала двигатель останавливается, текущее положение запоминается как метка. Поле DATA определяет скорость движения при поиске метки.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Скорость поиска метки						Command CMD_PowerSTEP01_ SCAN_LABEL_F = 0x18							Action	Reserve		
Значение	Зависит от значения поля Data						0	1	1	0	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.26 Исполнительная команда CMD\_PowerSTEP01\_SCAN\_LABEL\_R

Исполнительная команда CMD\_PowerSTEP01\_SCAN\_LABEL\_R = 0x19 предназначена для поиска метки положения в обратном направлении. Движение продолжается до поступления сигнала на вход IN1. При поступлении сигнала двигатель останавливается, текущее положение запоминается как метка. Поле DATA определяет скорость движения при поиске метки.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду.

Битовая раскладка структуры SMSD\_CMD\_Type:

Битовая раскладка структуры: CMD2_CMD1_1_0_																
---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.27 Исполнительная команда CMD\_PowerSTEP01\_GO\_ZERO

Исполнительная команда CMD\_PowerSTEP01\_GO\_ZERO = 0x1A предназначена для перемещения в нулевое положение..



Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_GO_ZERO = 0x1A							Action	Reserve		
Значение	0						0	1	1	0	1	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.28 Исполнительная команда CMD\_PowerSTEP01\_GO\_LABEL

Исполнительная команда CMD\_PowerSTEP01\_GO\_LABEL = 0x1B предназначена перемещения в положение, которое было отмечено как метка..

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_GO_LABEL = 0x1B							Action	Reserve		
Значение	0						0	1	1	0	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.29 Исполнительная команда CMD\_PowerSTEP01\_GO\_TO

Исполнительная команда CMD\_PowerSTEP01\_GO\_TO = 0x1C предназначена для перемещения в заданное положение по кратчайшему пути.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Заданное положение						Command CMD_PowerSTEP01_GO_TO = 0x1C							Action	Reserve		
Значение	Зависит от значения поля Data						0	1	1	1	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.30 Исполнительная команда CMD\_PowerSTEP01\_RESET\_POS

Исполнительная команда CMD\_PowerSTEP01\_RESET\_POS = 0x1D предназначена для обнуления счетчика текущего положения. После выполнения команды текущее положение принимается за нулевое.





Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_ RESET_POS = 0x1D						Action	Reserve			
Значение	0						0	1	1	1	0	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.31 Исполнительная команда CMD\_PowerSTEP01\_RESET\_POWERSTEP01

Исполнительная команда CMD\_PowerSTEP01\_RESET\_POWERSTEP01 = 0x1E используется для осуществления полного аппаратного и программного сброса модуля управления шаговым двигателем, но не контроллера в целом.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_ RESET_POWERSTEP01 = 0x1E						Action	Reserve			
Значение	0						0	1	1	1	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.32 Исполнительная команда CMD\_PowerSTEP01\_SOFT\_STOP

Исполнительная команда CMD\_PowerSTEP01\_SOFT\_STOP = 0x1F используется для плавной остановки двигателя с заданным ускорением. После остановки двигатель удерживает положение с заданным током удержания.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_ SOFT_STOP = 0x1F						Action	Reserve			
Значение	0						0	1	1	1	1	1	1	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.33 Исполнительная команда CMD\_PowerSTEP01\_HARD\_STOP

Исполнительная команда CMD\_PowerSTEP01\_HARD\_STOP = 0x20 используется для резкой остановки шагового двигателя. После остановки двигатель удерживает положение с заданным током удержания.



Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_ HARD_STOP = 0x20						Action		Reserve		
Значение	0						1	0	0	0	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.34 Исполнительная команда CMD\_PowerSTEP01\_SOFT\_HI\_Z

Исполнительная команда CMD\_PowerSTEP01\_SOFT\_HI\_Z = 0x21 используется для плавной остановки шагового двигателя с заданным ускорением. После остановки питания с обмоток двигателя снимается.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_SOFT_HI_Z = 0x21						Action		Reserve		
Значение	0						1	0	0	0	0	0	1	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.35 Исполнительная команда CMD\_PowerSTEP01\_HARD\_HI\_Z

Исполнительная команда CMD\_PowerSTEP01\_HARD\_HI\_Z = 0x22 используется для резкой остановки и обесточивания обмоток двигателя.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_HARD_HI_Z = 0x22						Action		Reserve		
Значение	0						1	0	0	0	0	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.36 Исполнительная команда CMD\_PowerSTEP01\_SET\_WAIT

Исполнительная команда CMD\_PowerSTEP01\_SET\_WAIT = 0x23 предназначена для задания паузы. Поле DATA должно содержать время ожидания паузы в диапазоне от 0 до 3600000мс.





Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = время ожидания						Command CMD_PowerSTEP01_ SET_WAIT = 0x23						Action	Reserve			
Значение	Зависит от значения поля Data						1	0	0	0	0	1	1	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.37 Исполнительная команда CMD\_PowerSTEP01\_SET\_RELE

Исполнительная команда CMD\_PowerSTEP01\_SET\_RELE = 0x24 предназначена для включения реле контроллера.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_SET_RELE = 0x24						Action	Reserve			
Значение	0						1	0	0	1	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = STATUS\_RELE\_SET.

### 6.38 Исполнительная команда CMD\_PowerSTEP01\_CLR\_RELE

Исполнительная команда CMD\_PowerSTEP01\_CLR\_RELE = 0x25 предназначена для выключения реле контроллера.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_CLR_RELE = 0x25						Action	Reserve			
Значение	0						1	0	0	1	0	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = STATUS\_RELE\_CLR.

### 6.39 Исполнительная команда CMD\_PowerSTEP01\_GET\_RELE

Исполнительная команда CMD\_PowerSTEP01\_GET\_RELE = 0x26 предназначена для запроса состояния реле контроллера.



Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_GET_RELE = 0x26							Action	Reserve		
Значение	0						1	0	0	1	1	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: значение ERROR\_OR\_COMMAND зависит от текущего состояния реле - STATUS\_RELE\_SET (реле включено) или STATUS\_RELE\_CLR (реле выключено).

#### 6.40 Исполнительная команда CMD\_PowerSTEP01\_WAIT\_IN0

Исполнительная команда CMD\_PowerSTEP01\_WAIT\_IN0 = 0x27 предназначена для ожидания поступления сигнала на вход IN0.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_WAIT_IN0= 0x27							Action	Reserve		
Значение	0						1	0	0	1	1	1	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.41 Исполнительная команда CMD\_PowerSTEP01\_WAIT\_IN1

Исполнительная команда CMD\_PowerSTEP01\_WAIT\_IN1 = 0x28 предназначена для ожидания поступления сигнала на вход IN1.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_WAIT_IN1= 0x28							Action	Reserve		
Значение	0						1	0	1	0	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.42 Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM

Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM = 0x29 предназначена для безусловного перехода к заданной команде заданной программы. Поле DATA должно содержать информацию о требуемом номере программы и порядковом номере команды в программе: биты 0..7 поля Data определяют номер команды, биты 8,9 поля DATA определяют номер программы.



Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Номер программы и команды						Command CMD_PowerSTEP01_ GOTO_PROGRAM = 0x29						Action	Reserve			
Значение	Зависит от значения поля Data						1	0	1	0	0	1	0	0	0	0	0

Битовая раскладка поля Data:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	0	0	0	0	Номер программы			Номер команды								

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.43 Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_IN0

Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_IN0 = 0x2A предназначена для перехода к заданной команде заданной программы, если на входе IN0 присутствует сигнал. Поле DATA должно содержать информацию о требуемом номере программы и порядковом номере команды в программе: биты 0..7 поля Data определяют номер команды, биты 8,9 поля DATA определяют номер программы.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Номер программы и команды						Command CMD_PowerSTEP01_ GOTO_PROGRAM_IF_IN0 = 0x2A						Action	Reserve			
Значение	Зависит от значения поля Data						1	0	1	0	1	0	0	0	0	0	0

Битовая раскладка поля Data:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	0	0	0	0	Номер программы			Номер команды								

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.44 Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_IN1

Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_IN1 = 0x2B предназначена для перехода к заданной команде заданной программы, если на входе IN1 присутствует сигнал. Поле DATA должно содержать информацию о требуемом номере программы и порядковом номере команды в программе: биты 0..7 поля Data определяют номер команды, биты 8,9 поля DATA определяют номер программы.

Битовая раскладка структуры SMSD\_CMD\_Type:

Сигналы раскладки структуры сигнала SMC2 = SMC2_1, SMC2_2, SMC2_3, SMC2_4, SMC2_5, SMC2_6, SMC2_7, SMC2_8, SMC2_9, SMC2_10, SMC2_11, SMC2_12, SMC2_13, SMC2_14, SMC2_15, SMC2_16, SMC2_17, SMC2_18, SMC2_19, SMC2_20, SMC2_21, SMC2_22, SMC2_23, SMC2_24, SMC2_25, SMC2_26, SMC2_27, SMC2_28, SMC2_29, SMC2_30, SMC2_31, SMC2_32, SMC2_33, SMC2_34, SMC2_35, SMC2_36, SMC2_37, SMC2_38, SMC2_39, SMC2_40, SMC2_41, SMC2_42, SMC2_43, SMC2_44, SMC2_45, SMC2_46, SMC2_47, SMC2_48, SMC2_49, SMC2_50, SMC2_51, SMC2_52, SMC2_53, SMC2_54, SMC2_55, SMC2_56, SMC2_57, SMC2_58, SMC2_59, SMC2_60, SMC2_61, SMC2_62, SMC2_63, SMC2_64, SMC2_65, SMC2_66, SMC2_67, SMC2_68, SMC2_69, SMC2_70, SMC2_71, SMC2_72, SMC2_73, SMC2_74, SMC2_75, SMC2_76, SMC2_77, SMC2_78, SMC2_79, SMC2_80, SMC2_81, SMC2_82, SMC2_83, SMC2_84, SMC2_85, SMC2_86, SMC2_87, SMC2_88, SMC2_89, SMC2_90, SMC2_91, SMC2_92, SMC2_93, SMC2_94, SMC2_95, SMC2_96, SMC2_97, SMC2_98, SMC2_99, SMC2_100, SMC2_101, SMC2_102, SMC2_103, SMC2_104, SMC2_105, SMC2_106, SMC2_107, SMC2_108, SMC2_109, SMC2_110, SMC2_111, SMC2_112, SMC2_113, SMC2_114, SMC2_115, SMC2_116, SMC2_117, SMC2_118, SMC2_119, SMC2_120, SMC2_121, SMC2_122, SMC2_123, SMC2_124, SMC2_125, SMC2_126, SMC2_127, SMC2_128, SMC2_129, SMC2_130, SMC2_131, SMC2_132, SMC2_133, SMC2_134, SMC2_135, SMC2_136, SMC2_137, SMC2_138, SMC2_139, SMC2_140, SMC2_141, SMC2_142, SMC2_143, SMC2_144, SMC2_145, SMC2_146, SMC2_147, SMC2_148, SMC2_149, SMC2_150, SMC2_151, SMC2_152, SMC2_153, SMC2_154, SMC2_155, SMC2_156, SMC2_157, SMC2_158, SMC2_159, SMC2_160, SMC2_161, SMC2_162, SMC2_163, SMC2_164, SMC2_165, SMC2_166, SMC2_167, SMC2_168, SMC2_169, SMC2_170, SMC2_171, SMC2_172, SMC2_173, SMC2_174, SMC2_175, SMC2_176, SMC2_177, SMC2_178, SMC2_179, SMC2_180, SMC2_181, SMC2_182, SMC2_183, SMC2_184, SMC2_185, SMC2_186, SMC2_187, SMC2_188, SMC2_189, SMC2_190, SMC2_191, SMC2_192, SMC2_193, SMC2_194, SMC2_195, SMC2_196, SMC2_197, SMC2_198, SMC2_199, SMC2_200, SMC2_201, SMC2_202, SMC2_203, SMC2_204, SMC2_205, SMC2_206, SMC2_207, SMC2_208, SMC2_209, SMC2_210, SMC2_211, SMC2_212, SMC2_213, SMC2_214, SMC2_215, SMC2_216, SMC2_217, SMC2_218, SMC2_219, SMC2_220, SMC2_221, SMC2_222, SMC2_223, SMC2_224, SMC2_225, SMC2_226, SMC2_227, SMC2_228, SMC2_229, SMC2_230, SMC2_231, SMC2_232, SMC2_233, SMC2_234, SMC2_235, SMC2_236, SMC2_237, SMC2_238, SMC2_239, SMC2_240, SMC2_241, SMC2_242, SMC2_243, SMC2_244, SMC2_245, SMC2_246, SMC2_247, SMC2_248, SMC2_249, SMC2_250, SMC2_251, SMC2_252, SMC2_253, SMC2_254, SMC2_255, SMC2_256, SMC2_257, SMC2_258, SMC2_259, SMC2_260, SMC2_261, SMC2_262, SMC2_263, SMC2_264, SMC2_265, SMC2_266, SMC2_267, SMC2_268, SMC2_269, SMC2_270, SMC2_271, SMC2_272, SMC2_273, SMC2_274, SMC2_275, SMC2_276, SMC2_277, SMC2_278, SMC2_279, SMC2_280, SMC2_281, SMC2_282, SMC2_283, SMC2_284, SMC2_285, SMC2_286, SMC2_287, SMC2_288, SMC2_289, SMC2_290, SMC2_291, SMC2_292, SMC2_293, SMC2_294, SMC2_295, SMC2_296, SMC2_297, SMC2_298, SMC2_299, SMC2_300, SMC2_301, SMC2_302, SMC2_303, SMC2_304, SMC2_305, SMC2_306, SMC2_307, SMC2_308, SMC2_309, SMC2_310, SMC2_311, SMC2_312, SMC2_313, SMC2_314, SMC2_315, SMC2_316, SMC2_317, SMC2_318, SMC2_319, SMC2_320, SMC2_321, SMC2_322, SMC2_323, SMC2_324, SMC2_325, SMC2_326, SMC2_327, SMC2_328, SMC2_329, SMC2_330, SMC2_331, SMC2_332, SMC2_333, SMC2_334, SMC2_335, SMC2_336, SMC2_337, SMC2_338, SMC2_339, SMC2_340, SMC2_341, SMC2_342, SMC2_343, SMC2_344, SMC2_345, SMC2_346, SMC2_347, SMC2_348, SMC2_349, SMC2_350, SMC2_351, SMC2_352, SMC2_353, SMC2_354, SMC2_355, SMC2_356, SMC2_357, SMC2_358, SMC2_359, SMC2_360, SMC2_361, SMC2_362, SMC2_363, SMC2_364, SMC2_365, SMC2_366, SMC2_367, SMC2_368, SMC2_369, SMC2_370, SMC2_371, SMC2_372, SMC2_373, SMC2_374, SMC2_375, SMC2_376, SMC2_377, SMC2_378, SMC2_379, SMC2_380, SMC2_381, SMC2_382, SMC2_383, SMC2_384, SMC2_385, SMC2_386, SMC2_387, SMC2_388, SMC2_389, SMC2_390, SMC2_391, SMC2_392, SMC2_393, SMC2_394, SMC2_395, SMC2_396, SMC2_397, SMC2_398, SMC2_399, SMC2_400, SMC2_401, SMC2_402, SMC2_403, SMC2_404, SMC2_405, SMC2_406, SMC2_407, SMC2_408, SMC2_409, SMC2_410, SMC2_411, SMC2_412, SMC2_413, SMC2_414, SMC2_415, SMC2_416, SMC2_417, SMC2_418, SMC2_419, SMC2_420, SMC2_421, SMC2_422, SMC2_423, SMC2_424, SMC2_425, SMC2_426, SMC2_427, SMC2_428, SMC2_429, SMC2_430, SMC2_431, SMC2_432, SMC2_433, SMC2_434, SMC2_435, SMC2_436, SMC2_437, SMC2_438, SMC2_439, SMC2_440, SMC2_441, SMC2_442, SMC2_443, SMC2_444, SMC2_445, SMC2_446, SMC2_447, SMC2_448, SMC2_449, SMC2_450, SMC2_451, SMC2_452, SMC2_453, SMC2_454, SMC2_455, SMC2_456, SMC2_457, SMC2_458, SMC2_459, SMC2_460, SMC2_461, SMC2_462, SMC2_463, SMC2_464, SMC2_465, SMC2_466, SMC2_467, SMC2_468, SMC2_469, SMC2_470, SMC2_471, SMC2_472, SMC2_473, SMC2_474, SMC2_475, SMC2_476, SMC2_477, SMC2_478, SMC2_479, SMC2_480, SMC2_481, SMC2_482, SMC2_483, SMC2_484, SMC2_485, SMC2_486, SMC2_487, SMC2_488, SMC2_489, SMC2_490, SMC2_491, SMC2_492, SMC2_493, SMC2_494, SMC2_495, SMC2_496, SMC2_497, SMC2_498, SMC2_499, SMC2_500, SMC2_501, SMC2_502, SMC2_503, SMC2_504, SMC2_505, SMC2_506, SMC2_507, SMC2_508, SMC2_509, SMC2_510, SMC2_511, SMC2_512, SMC2_513, SMC2_514, SMC2_515, SMC2_516, SMC2_517, SMC2_518, SMC2_519, SMC2_520, SMC2_521, SMC2_522, SMC2_523, SMC2_524, SMC2_525, SMC2_526, SMC2_527, SMC2_528, SMC2_529, SMC2_530, SMC2_531, SMC2_532, SMC2_533, SMC2_534, SMC2_535, SMC2_536, SMC2_537, SMC2_538, SMC2_539, SMC2_540, SMC2_541, SMC2_542, SMC2_543, SMC2_544, SMC2_545, SMC2_546, SMC2_547, SMC2_548, SMC2_549, SMC2_550, SMC2_551, SMC2_552, SMC2_553, SMC2_554, SMC2_555, SMC2_556, SMC2_557, SMC2_558, SMC2_559, SMC2_560, SMC2_561, SMC2_562, SMC2_563, SMC2_564, SMC2_565, SMC2_566, SMC2_567, SMC2_568, SMC2_569, SMC2_570, SMC2_571, SMC2_572, SMC2_573, SMC2_574, SMC2_575, SMC2_576, SMC2_577, SMC2_578, SMC2_579, SMC2_580, SMC2_581, SMC2_582, SMC2_583, SMC2_584, SMC2_585, SMC2_586, SMC2_587, SMC2_588, SMC2_589, SMC2_590, SMC2_591, SMC2_592, SMC2_593, SMC2_594, SMC2_595, SMC2_596, SMC2_597, SMC2_598, SMC2_599, SMC2_600, SMC2_601, SMC2_602, SMC2_603, SMC2_604, SMC2_605, SMC2_606, SMC2_607, SMC2_608, SMC2_609, SMC2_610, SMC2_611, SMC2_612, SMC2_613, SMC2_614, SMC2_615, SMC2_616, SMC2_617, SMC2_618, SMC2_619, SMC2_620, SMC2_621, SMC2_622, SMC2_623, SMC2_624, SMC2_625, SMC2_626, SMC2_627, SMC2_628, SMC2_629, SMC2_630, SMC2_631, SMC2_632, SMC2_633, SMC2_634, SMC2_635, SMC2_636, SMC2_637, SMC2_638, SMC2_639, SMC2_640, SMC2_641, SMC2_642, SMC2_643, SMC2_644, SMC2_645, SMC2_646, SMC2_647, SMC2_648, SMC2_649, SMC2_650, SMC2_651, SMC2_652, SMC2_653, SMC2_654, SMC2_655, SMC2_656, SMC2_657, SMC2_658, SMC2_659, SMC2_660, SMC2_661, SMC2_662, SMC2_663, SMC2_664, SMC2_665, SMC2_666, SMC2_667, SMC2_668, SMC2_669, SMC2_670, SMC2_671, SMC2_672, SMC2_673, SMC2_674, SMC2_675, SMC2_676, SMC2_677, SMC2_678, SMC2_679, SMC2_680, SMC2_681, SMC2_682, SMC2_683, SMC2_684, SMC2_685, SMC2_686, SMC2_687, SMC2_688, SMC2_689, SMC2_690, SMC2_691, SMC2_692, SMC2_693, SMC2_694, SMC2_695, SMC2_696, SMC2_697, SMC2_698, SMC2_699, SMC2_700, SMC2_701, SMC2_702, SMC2_703, SMC2_704, SMC2_705, SMC2_706, SMC2_707, SMC2_708, SMC2_709, SMC2_710, SMC2_711, SMC2_712, SMC2_713, SMC2_714, SMC2_715, SMC2_716, SMC2_717, SMC2_718, SMC2_719, SMC2_720, SMC2_721, SMC2_722, SMC2_723, SMC2_724, SMC2_725, SMC2_726, SMC2_727, SMC2_728, SMC2_729, SMC2_730, SMC2_731, SMC2_732, SMC2_733, SMC2_734, SMC2_735, SMC2_736, SMC2_737, SMC2_738, SMC2_739, SMC2_740, SMC2_741, SMC2_742, SMC2_743, SMC2_744, SMC2_745, SMC2_746, SMC2_747, SMC2_748, SMC2_749, SMC2_750, SMC2_751, SMC2_752, SMC2_753, SMC2_754, SMC2_755, SMC2_756, SMC2_757, SMC2_758, SMC2_759, SMC2_760, SMC2_761, SMC2_762, SMC2_763, SMC2_764, SMC2_765, SMC2_766, SMC2_767, SMC2_768, SMC2_769, SMC2_770, SMC2_771, SMC2_772, SMC2_773, SMC2_774, SMC2_775, SMC2_776, SMC2_777, SMC2_778, SMC2_779, SMC2_780, SMC2_781, SMC2_782, SMC2_783, SMC2_784, SMC2_785, SMC2_786, SMC2_787, SMC2_788, SMC2_789, SMC2_790, SMC2_791, SMC2_792, SMC2_793, SMC2_794, SMC2_795, SMC2_796, SMC2_797, SMC2_798, SMC2_799, SMC2_800, SMC2_801, SMC2_802, SMC2_803, SMC2_804, SMC2_805, SMC2_806, SMC2_807, SMC2_808, SMC2_809, SMC2_810, SMC2_811, SMC2_812, SMC2_813, SMC2_814, SMC2_815, SMC2_816, SMC2_817, SMC2_818, SMC2_819, SMC2_820, SMC2_821, SMC2_822, SMC2_823, SMC2_824, SMC2_825, SMC2_826, SMC2_827, SMC2_828, SMC2_829, SMC2_830, SMC2_831, SMC2_832, SMC2_833, SMC2_834, SMC2_835, SMC2_836, SMC2_837, SMC2_838, SMC2_839, SMC2_840, SMC2_841, SMC2_842, SMC2_843, SMC2_844, SMC2_845, SMC2_846, SMC2_847, SMC2_848, SMC2_849, SMC2_850, SMC2_851, SMC2_852, SMC2_853, SMC2_854, SMC2_855, SMC2_856, SMC2_857, SMC2_858, SMC2_859, SMC2_860, SMC2_861, SMC2_862, SMC2_863, SMC2_864, SMC2_865, SMC2_866, SMC2_867, SMC2_868, SMC2_869, SMC2_870, SMC2_871, SMC2_872, SMC2_873, SMC2_874, SMC2_875, SMC2_876, SMC2_877, SMC2_878, SMC2_879, SMC2_880, SMC2_881, SMC2_882, SMC2_883, SMC2_884, SMC2_885, SMC2_886, SMC2_887, SMC2_888, SMC2_889, SMC2_890, SMC2_891, SMC2_892, SMC2_893, SMC2_894, SMC2_895, SMC2_896, SMC2_897, SMC2_898, SMC2_899, SMC2_900, SMC2_901, SMC2_902, SMC2_903, SMC2_904, SMC2_905, SMC2_906, SMC2_907, SMC2_908, SMC2_909, SMC2_910, SMC2_911, SMC2_912, SMC2_913, SMC2_914, SMC2_915, SMC2_916, SMC2_917, SMC2_918, SMC2_919, SMC2_920, SMC2_921, SMC2_922, SMC2_923, SMC2_924, SMC2_925, SMC2_926, SMC2_927, SMC2_928, SMC2_929, SMC2_930, SMC2_931, SMC2_932, SMC2_933, SMC2_934, SMC2_935, SMC2_936, SMC2_937, SMC2_938, SMC2_939, SMC2_940, SMC2_941, SMC2_942, SMC2_943, SMC2_944, SMC2_945, SMC2_946, SMC2_947, SMC2_948, SMC2_949, SMC2_950, SMC2_951, SMC2_952, SMC2_953, SMC2_954, SMC2_955, SMC2_956, SMC2_957, SMC2_958, SMC2_959, SMC2_960, SMC2_961, SMC2_962, SMC2_963, SMC2_964, SMC2_965, SMC2_966, SMC2_967, SMC2_968, SMC2_969, SMC2_970, SMC2_971, SMC2_972, SMC2_973, SMC2_974, SMC2_975, SMC2_976, SMC2_977, SMC2_978, SMC2_979, SMC2_980, SMC2_981, SMC2_982, SMC2_983, SMC2_984, SMC2_985, SMC2_986, SMC2_987, SMC2_988, SMC2_989, SMC2_990, SMC2_991, SMC2_992, SMC2_993, SMC2_994, SMC2_995, SMC2_996, SMC2_997, SMC2_998, SMC2_999, SMC2_1000, SMC2_1001, SMC2_1002, SMC2_1003, SMC2_1004, SMC2_1005, SMC2_1006, SMC2_1007, SMC2_1008, SMC2_1009, SMC2_1010, SMC2_1011, SMC2_1012, SMC2_1013, SMC2_1014, SMC2_1015, SMC2_1016, SMC2_1017, SMC2_1018, SMC2_1019, SMC2_1020, SMC2_1021, SMC2_1022, SMC2_1023, SMC2_1024, SMC2_1025, SMC2_1026, SMC2_1027, SMC2_1028, SMC2_1029, SMC2_1030, SMC2_1031, SMC2_1032, SMC2_1033, SMC2_1034, SMC2_1035, SMC2_1036, SMC2_1037, SMC2_1038, SMC2_1039, SMC2_1040, SMC2_1041, SMC2_1042, SMC2_1043, SMC2_1044, SMC2_1045, SMC2_1046, SMC2_1047, SMC2_1048, SMC2_1049, SMC2_1050, SMC2_1051, SMC2_1052, SMC2_1053, SMC2_1054, SMC2_1055, SMC2_1056, SMC2_1057, SMC2_1058, SMC2_1059, SMC2_1060, SMC2_1061, SMC2_1062, SMC2_1063, SMC2_1064, SMC2_1065, SMC2_1066, SMC2_1067, SMC2_1068, SMC2_1069, SMC2_1070, SMC2_1071, SMC2_1072, SMC2_1073, SMC2_1074, SMC2_1075, SMC2_1076, SMC2_1077, SMC2_1078, SMC2_1079, SMC2_1080, SMC2_1081, SMC2_1082, SMC2_1083, SMC2_1084, SMC2_1085, SMC2_1086, SMC2_1087, SMC2_1088, SMC2_1089, SMC2_1090, SMC2_1091, SMC2_1092, SMC2_1093, SMC2_1094, SMC2_1095, SMC2_1096, SMC2_1097, SMC2_1098, SMC2_1099, SMC2_1100, SMC2_1101, SMC2_1102, SMC2_1103, SMC2_1104, SMC2_1105, SMC2_1106, SMC2_1107, SMC2_1108, SMC2_1109, SMC2_1110, SMC2_1111, SMC2_1112, SMC2_1113, SMC2_1114, SMC2_1115, SMC2_1116, SMC2_1117, SMC2_1118, SMC2_1119, SMC2_1120, SMC2_1121, SMC2_1122, SMC2_1123, SMC2_1124, SMC2_1125, SMC2_1126, SMC2_1127, SMC2_1128, SMC2_1129, SMC2_1130, SMC2_1131, SMC2_1132, SMC2_1133, SMC2_1134, SMC2_1135, SMC2_1136, SMC2_1137, SMC2_1138, SMC2_1139, SMC2_1140, SMC2_1141, SMC2_1142, SMC2_1143, SMC2_1144, SMC2_1145, SMC2_1146, SMC2_1147, SMC2_1148, SMC2_1149, SMC2_1150, SMC2_1151, SMC2_1152, SMC2_1153, SMC2_1154, SMC2_1155, SMC2_1156, SMC2_1157, SMC2_1158, SMC2_1159, SMC2_1160, SMC2_1161, SMC2_1162, SMC2_1163, SMC2_1164, SMC2_1165, SMC2_1166, SMC2_1167, SMC2_1168, SMC2_1169, SMC2_1170, SMC2_1171, SMC2_1172, SMC2_1173, SMC2_1174, SMC2_1175, SMC2_1176, SMC2_1177, SMC2_1178, SMC2_1179, SMC2_1180, SMC2_1181, SMC2_1182, SMC2_1183, SMC2_1184, SMC2_1185, SMC2_1186, SMC2_1187, SMC2_1188, SMC2_1189, SMC2_1190, SMC2_1191, SMC2_1192, SMC2_1193, SMC2_1194, SMC2_1195, SMC2_1196, SMC2_1197, SMC2_1198, SMC2_1199, SMC2_1200, SMC2_1201, SMC2_1202, SMC2_1203, SMC2_1204, SMC2_1205, SMC2_1206, SMC2_1207, SMC2_1208, SMC2_1209, SMC2_1210, SMC2_1211, SMC2_1212, SMC2_1213, SMC2_1214, SMC2_1215, SMC2_1216, SMC2_1217, SMC2_1218, SMC2_1219, SMC2_1220, SMC2_1221, SMC2_1222, SMC2_1223, SMC2_1224, SMC2_1225, SMC2_1226, SMC2_1227, SMC2_1228, SMC2_1229, SMC2_1230, SMC2_1231, SMC2_1232, SMC2_1233, SMC2_1234, SMC2_1235, SMC2_1236, SMC2_1237, SMC2_1238, SMC2_1239, SMC2_1240, SMC2_1241, SMC2_1242, SMC2_1243, SMC2_1244, SMC2_1245, SMC2_1246, SMC2_1247, SMC2_1248, SMC2_1249, SMC2_1250, SMC2_1251, SMC2_1252, SMC2_1253, SMC2_1254, SMC2_1255, SMC2_1256, SMC2_1257, SMC2_1258, SMC2_1259, SMC2_1260, SMC2_1261, SMC2_1262, SMC2_1263, SMC2_1264, SMC2_1265, SMC2_1266, SMC2_1267, SMC2_1268, SMC2_1269, SMC2_1270, SMC2_1271, SMC2_1272, SMC2_1273, SMC2_1274, SMC2_1275, SMC2_1276, SMC2_1277, SMC2_1278, SMC2_1279, SMC2_1280, SMC2_1281, SMC2_1282, SMC2_1283, SMC2_1284, SMC2_1285, SMC2_1286, SMC2_1287, SMC2_1288, SMC2_1289, SMC2_1290, SMC2_1291, SMC2_1292, SMC2_1293, SMC2_1294, SMC2_1295, SMC2_1296, SMC2_1297, SMC2_1298, SMC2_1299, SMC2_1300, SMC2_1301, SMC2_1302, SMC2_1303, SMC2_1304, SMC2_1305, SMC2_1306, SMC2_1307, SMC2_1308, SMC2_1309, SMC2_1310, SMC2_1311, SMC2_1312, SMC2_1313, SMC2_1314, SMC2_1315, SMC2_1316, SMC2_1317, SMC2_1318, SMC2_1319, SMC2_1320, SMC2_1321, SMC2_1322, SMC2_1323, SMC2_1324, SMC2_1325, SMC2_1326, SMC2_1327, SMC2_1328, SMC2_1329, SMC2_1330, SMC2_1331, SMC2_1332, SMC2_1333, SMC2_1334, SMC2_1335, SMC2_1336, SMC2_1337, SMC2_1338, SMC2_1339, SMC2_1340, SMC2_1341, SMC2_1342, SMC2_1343, SMC2_1344, SMC2_1345, SMC2_1346, SMC2_1347, SMC2_1348, SMC2_1349, SMC2_1350, SMC2_1351, SMC2_1352, SMC2_1353, SMC2_1354, SMC2_1355, SMC2_1356, SMC2_1357, SMC2_1358, SMC2_1359, SMC2_1360, SMC2_1361, SMC2_1362, SMC2_1363, SMC2_1364, SMC2_1365, SMC2_1366, SMC2_1367, SMC2_1368, SMC2_1369, SMC2_1370, SMC2_1371, SMC2_1372, SMC2_1373, SMC2_1374, SMC2_1375, SMC2_1376, SMC2_1377, SMC2_1378, SMC2_1379, SMC2_1380, SMC2_1381, SMC2_1382, SMC2_1383, SMC2_1384, SMC2_1385, SMC2_1386, SMC2_1387, SMC2_1388, SMC2_1389, SMC2_1390, SMC2_1391, SMC2_1392, SMC2_1393, SMC2_1394, SMC2_1395, SMC2_1396, SMC2_1397, SMC2_1398, SMC2_1399, SMC2_1400, SMC2_1401, SMC2_1402, SMC2_1403, SMC2_1404, SMC2_1405, SMC2_1406, SMC2_1407, SMC2_1408, SMC2_1409, SMC2_1410, SMC2_1411, SMC2_1412, SMC2_1413, SMC2_1414, SMC2_1415, SMC2_1416, SMC2_1417, SMC2_1418, SMC2_1419, SMC2_1420, SMC2_1421, SMC2_1422, SMC2_1423, SMC2_1424, SMC2_1425, SMC2_1426, SMC2_1427, SMC2_1428, SMC2_1429, SMC2_1430, SMC2_1431, SMC2_1432, SMC2_1433, SMC2_1434, SMC2_1435, SMC2_1436, SMC2_1437, SMC2_1438, SMC2_1439, SMC2_1440, SMC2_1441, SMC2_1442, SMC2_1443, SMC2_1444, SMC2_1445, SMC2_1446, SMC2_1447, SMC2_1448, SMC2_1449, SMC2_1450, SMC2_1451, SMC2_1452, SMC2_1453, SMC2_1454, SMC2_1455, SMC2_1456, SMC2_1457, SMC2_1458, SMC2_1459, SMC2_1460, SMC2_1461, SMC2_1462, SMC2_1463, SMC2_1464, SMC2_1465, SMC2_1466, SMC2_1467, SMC2_1468, SMC2_1469, SMC2_1470, SMC2_1471, SMC2_1472, SMC2_1473, SMC2_1474, SMC2_1475, SMC2_1476, SMC2_1477, SMC2_1478, SMC2_1479, SMC2_1480, SMC2_1481, SMC2_1482, SMC2_1483, SMC2_1484, SMC2_1485, SMC2_1486, SMC2_1487, SMC2_1488, SMC2_1489, SMC2_1490, SMC2_1491, SMC2_1492, SMC2_1493, SMC2_1494, SMC2_1495, SMC2_1496, SMC2_1																
--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Битовая раскладка поля Data:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	0	0	0	0	Номер программы		Номер команды									

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.45 Исполнительная команда CMD\_PowerSTEP01\_LOOP\_PROGRAM

Исполнительная команда CMD\_PowerSTEP01\_LOOP\_PROGRAM = 0x2C используется для создания циклов – контроллер повторяет в цикле заданное число раз заданное количество команд (начиная с команды, следующей за CMD\_PowerSTEP01\_LOOP\_PROGRAM). Поле DATA содержит количество циклов (от 1 до 1023) и количество команд в цикле (от 1 до 1023): 0..9 поля Data – количество команд, биты 10..19 поля Data – количество циклов.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Количество команд и циклов						Command CMD_PowerSTEP01_ LOOP_PROGRAM = 0x2C						Action	Reserve			
Значение	Зависит от значения поля Data						1	0	1	1	0	0	0	0	0	0	0

Битовая раскладка поля Data:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	Количество циклов										Количество команд в цикле											

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.46 Исполнительная команда CMD\_PowerSTEP01\_CALL\_PROGRAM

Исполнительная команда CMD\_PowerSTEP01\_CALL\_PROGRAM = 0x2D предназначена для вызова подпрограммы. Поле DATA содержит информацию о номере программы и порядковом номере команды в программе, с которой начинается подпрограмма: биты 0..7 поля DATA – порядковый номер команды, биты 8,9 поля DATA – номер программы. Для возврата в основную программу подпрограмма должна содержать команду возврата - CMD\_PowerSTEP01\_RETURN\_PROGRAM. Подпрограмма выполняется до тех пор, пока не дойдет до команды CMD\_PowerSTEP01\_RETURN\_PROGRAM, затем возвращает управление основной вызвавшей ее программе.



Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Номера команды и программы						Command CMD_PowerSTEP01_CALL_PROG RAM = 0x2D						Action	Reserve			
Значение	Зависит от значения поля Data						1	0	1	1	0	1	0	0	0	0	0

Битовая раскладка поля Data:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	0	0	0	0	Номер программы			Номер команды								

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.47 Исполнительная команда CMD\_PowerSTEP01\_RETURN\_PROGRAM

Исполнительная команда CMD\_PowerSTEP01\_RETURN\_PROGRAM = 0x2E используется для возврата из подпрограммы в основную программу. Вызов подпрограммы осуществляется командой CMD\_PowerSTEP01\_CALL\_PROGRAM. В случае, если перед вызовом команды возврата CMD\_PowerSTEP01\_RETURN\_PROGRAM не была вызвана подпрограмма, контроллер сгенерирует ошибку.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_RETURN_PROGRAM = 0x2E						Action	Reserve			
Значение	0						1	0	1	1	1	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.48 Исполнительная команда CMD\_PowerSTEP01\_START\_PROGRAM\_MEM0

Исполнительная команда CMD\_PowerSTEP01\_START\_PROGRAM\_MEM0 = 0x2F используется для старта программы, записанной в область памяти 0 контроллера.

Команды CMD\_PowerSTEP01\_START\_PROGRAM\_MEM1 = 0x30, CMD\_PowerSTEP01\_START\_PROGRAM\_MEM2 = 0x31, CMD\_PowerSTEP01\_START\_PROGRAM\_MEM3 = 0x32 используются для старта программ, записанных области памяти 1, 2 и 3 соответственно.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_START_PROGRAM_MEM0= 0x2F						Action	Reserve			
Значение	0						1	0	1	1	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.49 Исполнительная команда CMD\_PowerSTEP01\_STOP\_PROGRAM\_MEM

Исполнительная команда CMD\_PowerSTEP01\_STOP\_PROGRAM\_MEM = 0x33 используется для остановки выполнения программы.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_ STOP_PROGRAM_MEM = 0x33						Action	Reserve			
Значение	0						1	1	0	0	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

#### 6.50 Исполнительная команда CMD\_PowerSTEP01\_STEP\_CLOCK

Исполнительная команда CMD\_PowerSTEP01\_STEP\_CLOCK = 0x34 предназначена для изменения режима управления двигателем на импульсные сигналами EN, STEP, DIR.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_ STEP_CLOCK = 0x34						Action	Reserve			
Значение	0						1	1	0	1	0	0	0	0	0	0	0

#### 6.51 Исполнительная команда CMD\_PowerSTEP01\_STOP\_USB

Исполнительная команда CMD\_PowerSTEP01\_STOP\_USB = 0x35 предназначена для остановки работы микросхемы USB.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_ STOP_USB = 0x35						Action	Reserve			
Значение	0						1	1	0	1	0	1	0	0	0	0	0

#### 6.52 Исполнительная команда CMD\_PowerSTEP01\_GET\_MIN\_SPEED

Исполнительная команда CMD\_PowerSTEP01\_GET\_MIN\_SPEED = 0x36 предназначена для чтения текущего значения установленной минимальной скорости. Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Поле DATA пакета содержит структуру SMSD\_CMD\_Type, поле command которой содержит код команды запроса текущего значения установленной минимальной скорости CMD\_PowerSTEP01\_GET\_MIN\_SPEED.





Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_GET_MIN_SP EED = 0x36						Action	Reserve			
Значение	0						1	1	0	1	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = COMMAND\_GET\_MIN\_SPEED, RETURN\_DATA - значение текущей установленной минимальной скорости двигателя.

### 6.53 Исполнительная команда CMD\_PowerSTEP01\_GET\_MAX\_SPEED

Исполнительная команда CMD\_PowerSTEP01\_GET\_MAX\_SPEED = 0x37 предназначена для чтения текущего значения установленной максимальной скорости. Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Поле DATA пакета содержит структуру SMSD\_CMD\_Type, поле command которой содержит код команды запроса текущего значения установленной максимальной скорости CMD\_PowerSTEP01\_GET\_MAX\_SPEED.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_GET_MAX_S PEED = 0x37						Action	Reserve			
Значение	0						1	1	0	1	1	1	1	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = COMMAND\_GET\_MAX\_SPEED, RETURN\_DATA - значение текущей установленной максимальной скорости двигателя.

### 6.54 Исполнительная команда CMD\_PowerSTEP01\_GET\_STACK

Исполнительная команда CMD\_PowerSTEP01\_GET\_STACK = 0x38 предназначена для чтения информации о выполняемой в данный момент программе. Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Поле DATA пакета содержит структуру SMSD\_CMD\_Type, поле command которой содержит код команды запроса номера исполняемой программы и номер текущей команды CMD\_PowerSTEP01\_GET\_STACK.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_GET_STACK = 0x38						Action	Reserve			
Значение	0						1	1	1	0	0	0	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND =



COMMAND\_GET\_STACK, RETURN\_DATA - содержит номер текущей выполняемой команды (первые 8 бит) и номер программы (2 бита).

Битовая раскладка поля Return\_DATA:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	0	0	0	0	Номер программы			Номер команды								

### 6.55 Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_ZERO

Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_ZERO = 0x39 предназначена для перехода к заданной команде заданной программы, если значение текущей позиции равно 0. Поле DATA должно содержать информацию о требуемом номере программы и порядковом номере команды в программе: биты 0..7 поля Data определяют номер команды, биты 8,9 поля DATA определяют номер программы.

Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = Номер программы и команды							Command CMD_PowerSTEP01_GOTO_PROG RAM_IF_ZERO = 0x39							Action	Reserve		
Значение	Зависит от значения поля Data							1	1	1	0	0	1	0	0	0	0	

Битовая раскладка поля Data:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	0	0	0	0	Номер программы			Номер команды								

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.56 Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_IN\_ZERO

Исполнительная команда CMD\_PowerSTEP01\_GOTO\_PROGRAM\_IF\_IN\_ZERO = 0x3A предназначена для перехода к заданной команде заданной программы, в случае, если на входе SET\_ZERO присутствует сигнал. Поле DATA должно содержать информацию о требуемом номере программы и порядковом номере команды в программе: биты 0..7 поля Data определяют номер команды, биты 8,9 поля DATA определяют номер программы.

Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = Номер программы и команды							Command CMD_PowerSTEP01_GOTO_PROG RAM_IF_IN_ZERO = 0x3A							Action	Reserve		
Значение	Зависит от значения поля Data							1	1	1	0	1	0	0	0	0	0	



Битовая раскладка поля Data:

	Байт[3] биты 7..0								Байт[2] биты 7..0								Байт[1] биты 7..2							
Номер бита поля Data	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	0	0	0	0	Номер программы			Номер команды								

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.57 Исполнительная команда CMD\_PowerSTEP01\_WAIT\_CONTINUE

Исполнительная команда CMD\_PowerSTEP01\_WAIT\_CONTINUE = 0x3B предназначена для ожидания прихода синхросигнала на вход CONTINUE, необходимого для синхронизации программ в нескольких блоках. Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	0						Command CMD_PowerSTEP01_WAIT_CONTI NUE = 0x3B							Action	Reserve		
Значение	0						1	1	1	0	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.58 Исполнительная команда CMD\_PowerSTEP01\_SET\_WAIT\_2

Исполнительная команда CMD\_PowerSTEP01\_SET\_WAIT\_2 = 0x3C предназначена для задания паузы. Поле DATA должно содержать время ожидания паузы в диапазоне от 0 до 3600000мс. В отличие от аналогичной команды CMD\_PowerSTEP01\_SET\_WAIT, выполнение данной команды может быть прервано поступлением сигнала на вход IN0, IN1 или SET\_ZERO.

Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]								
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0	
Назначение	Data = время ожидания							Command CMD_PowerSTEP01_ SET_WAIT_2 = 0x3C							Action	Reserve		
Значение	Зависит от значения поля Data							1	0	0	0	1	1	0	0	0	0	

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

### 6.59 Исполнительная команда CMD\_PowerSTEP01\_SCAN\_MARK2\_F

Исполнительная команда CMD\_PowerSTEP01\_SCAN\_MARK2\_F = 0x3D предназначена для поиска метки положения в прямом направлении. Движение продолжается до поступления сигнала на вход IN1. При поступлении сигнала двигатель останавливается с заданным значением торможения, текущее положение запоминается как метка. Поле DATA определяет скорость движения при поиске метки.



Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду. Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Скорость поиска метки						Command CMD_PowerSTEP01_SCAN_MARK 2_F = 0x3D								Action	Reserve	
Значение	Зависит от значения поля Data						1	1	1	0	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 6.60 Исполнительная команда CMD\_PowerSTEP01\_SCAN\_MARK2\_R

Исполнительная команда CMD\_PowerSTEP01\_SCAN\_MARK2\_R = 0x3E предназначена для поиска метки положения в обратном направлении. Движение продолжается до поступления сигнала на вход IN1. При поступлении сигнала двигатель останавливается с заданным значением торможения, текущее положение запоминается как метка. Поле DATA определяет скорость движения при поиске метки.

Внимание: в командах установки скорости единицы измерения параметра - полные шаги в секунду. Данная команда добавлена в версию протокола 02. В версии 1 эта команда отсутствует.

Битовая раскладка структуры SMSD\_CMD\_Type:

	Байт[3]			Байт[2]			Байт[1]			Байт[0]							
Бит	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Назначение	Data = Скорость поиска метки						Command CMD_PowerSTEP01_SCAN_MARK 2_R = 0x3E								Action	Reserve	
Значение	Зависит от значения поля Data						1	1	1	1	1	1	0	0	0	0	0

В ответ контроллер отправляет пакет с кодом команды CMD\_TYPE = CODE\_CMD\_RESPONSE, поле DATA которого содержит структуру COMMANDS\_RETURN\_DATA\_Type: ERROR\_OR\_COMMAND = OK.

## 7. Структура SMSD\_LAN\_Config\_Type

Сетевые настройки Контроллера содержатся в структуре SMSD\_LAN\_Config\_Type:

```
typedef struct
{
    uint8_t mac[6];
    uint8_t ip[4];
    uint8_t sn[4];
    uint8_t gw[4];
    uint8_t dns[4];
    uint16_t Port;
    dhcp_mode dhcp;
} SMSD_LAN_Config_Type;
```

Значения по умолчанию:

```
{
    .mac= {0x00, 0xf8, 0xdc, 0x3f, 0x00, 0x00},
    .ip = {192, 168, 1, 2},
```



```
.sn = {255,255,0,0},  
.gw = {192, 168, 1, 1},  
.dns= {0,0,0,0},  
.Port = 5000,  
.dhcp = 1  
};
```

## 8. Отличия между Ethernet и USB в передаче потока данных

Поток данных, передаваемый по физическому каналу USB (имитация COM-port), представляет собой данные Ethernet, в начале и конце которого установлены маркеры начала и конца пакета, а уникальные символы заменены на парные по заданному алгоритму:

0xFA – маркер «начало пакета»

0xFB – маркер «конец пакета»

Если в потоке данных встречаются уникальные байты: 0xFA, 0xFB, 0xFE, они преобразуются на пары байтов 0xFE 0XX, где  $0XX = \text{байт} \wedge 0x80$

Байт 0xFA внутри пакета замещается парой байтов 0xFE 0x7A.

Байт 0xFB внутри пакета замещается парой байтов 0xFE 0x7B.

Байт 0xFE внутри пакета замещается парой байтов 0xFE 0x7E.