SMART MOTOR DEVICES

https://smd.ee

PROGRAMMABLE STEP MOTOR CONTROLLER SMSD-4.2LAN and SMSD-8.0LAN

Data communications protocol
Ver. 04

2019

1. Brief introduction.

The controller SMSD-4.2LAN (further in the text - Controller) is intended for control of stepper motors and provides programming and control via USB or Ethernet.

SMSD-4.2LAN is designed as a circuit plate with electronic components, indicators, control elements, terminal blocks and connectors installed on a heat sink plate. Control and indication elements are located at the front side of the controller.

When use local network Ethernet operation mode (LA indication on the display), the controller creates a socket for connection of a user software or electronic device (further in the text - User). The data transfer is provided through a physical line Ethernet, protocol TCP. In case of USB connection virtual COM port RS-232 is used.

Command codes and data transfer structure are the same for Ethernet and USB connections with exception of little differences in application layer of data stream transmission. So, the following manual is given for Ethernet connection. Data transmission difference for USB connection is given in a separate chapter of this manual.

This manual applies to the standard firmware version of the controllers. Please refer to the manual “Data Communication Protocol - Modbus TCP/IP” for the controllers with Modbus firmware.

2. USB and Ethernet data communication basis

It is required to transfer data as whole information packets, every packet conforms the structure, described in this manual. Every packet contains only one data transmission command. It is not possible to transfer more than one data transmission command inside one information packet. Every information packet should be continuously transferred, without interruptions.

After receiving an information packet, the controller handles it and sends a response, the response is sent the same physical line as the command was received.

A sequence of bytes in the information packets is inverted – “little-endian”, (Intel).

3.Default settings

Ethernet connection settings:

- MAC address: 0x00 0xf8 0xdc 0x3f 0x00 0x00
- IP address: 192.168.1.2
- Port: 5000
- IP sub-network mask: 255.255.0.0
- Gateway: 192.168.1.1

These parameters can be changed afterwards by commands sent through a USB or Ethernet connection.

RS-232 parameters (USB connection):

- Baud rate - 115200
- Data bits - 8
- Parity – none
- Stop bits – 1

4. The structure of data transfer packets

The data transfer packet structure is the next:

```
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 – checksum – low-order byte off the amount of all bytes in the packet.

VER – communication protocol version.

CMD_TYPE – type of the data transmission command

CMD_IDENTIFICATION – unique identifier of the data transfer packet. The same identifier is sent inside the response information packet from the controller. The identifier uniquely associates a transferred command and received response.

LENGTH_DATA – length of the data portion of the packet, values from 0 to 1024

DATA[LENGTH_DATA] – the data portion of the packet, length of the data portion is LENGTH_DATA bytes.

4.1. The purpose of the XOR_SUM field

1 byte field. TCP protocol means assured data transfer from a sender to a receiver and includes control and error-check of the data. However, the data transfer packet includes the XOR_SUM field – the checksum of the packet. This field is intended for control of the data transmission continuity in case of using USB connection. The XOR_SUM algorithm for computing is the next:

```
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– start of the data transfer packet,
sizeof(COMMAND) – length of the data transfer packet (bytes).

4.2. The purpose of the VER field

1 byte field VER - the current version of the data communication protocol.

4.3. The purpose of the CMD_IDENTIFICATION field

1 byte field. The field CMD_IDENTIFICATION is intended for unique identification of a response to a sent command. The User should provide unique values during data transfer process.

4.4. The purpose of the LENGTH_DATA field

2 bytes field. The field LENGTH_DATA determines the length of the DATA field - information part of the packet, possible values from 0 to 1024.

4.5. The purpose of the DATA[LENGTH_DATA] field

DATA[LENGTH_DATA] field - array DATA (LENGTH_DATA bytes size) - is the information part of the packet. The structure of the field depends on the CMD_TYPE field.

4.6. The purpose of the CMD_TYPE field

1 byte field. The data transmission command of the packet. Values start from 0 and gradually-increase. The list of data transmission commands (CMD_TYPE field) is the next:

CODE_CMD_REQUEST – authentication (the DATA field of the packet contains authentification information)

CODE_CMD_RESPONSE – confirmation (the entry of the DATA field depends on a sent data transmission command)

CODE_CMD_POWERSTEP01 – motor control (the DATA field of the packet contains POWERSTEP01 commands - SMSD_CMD_Type type)

CODE_CMD_POWERSTEP01_W_MEM0 – writing of an executing program into the controller memory 0.

CODE_CMD_POWERSTEP01_W_MEM1 – writing of an executing program into the controller memory 1

CODE_CMD_POWERSTEP01_W_MEM2 – writing of an executing program into the controller memory 2

CODE_CMD_POWERSTEP01_W_MEM3 – writing of an executing program into the controller memory 3

CODE_CMD_POWERSTEP01_R_MEM0 – reading o fan executing program from the controller memory 0

CODE_CMD_POWERSTEP01_R_MEM1 – reading of an executing program from the controller memory 1

CODE_CMD_POWERSTEP01_R_MEM2 – reading of an executing program from the controller memory 2

CODE_CMD_POWERSTEP01_R_MEM3 – reading of an executing program from the controller memory 3

CODE_CMD_CONFIG_SET - writing of LAN parameters

CODE_CMD_CONFIG_GET - reading of LAN parameters

CODE_CMD_PASSWORD_SET - changing of authentication password

CODE_CMD_ERROR_GET - reading of information about number of operation mode starts and error statistics.

4.6.1 Beginning of data transmission and data transmission command CODE_CMD_REQUEST

The data transmission command CODE_CMD_REQUEST is used for authorizing purpose. The data transfer packet with CODE_CMD_REQUEST code is sent from the controller to the user as a response to a LAN connection event (only for LAN connection, not used for USB connection).

The order of packet exchange during the authorization process:

1. From the controller (only in case of LAN connection):

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	CODE_CMD_REQUEST= 0x00	2
CMD_IDENTIFICATION (1 byte)	x	3
LENGTH_DATA (Low byte)	0x00	4
LENGTH_DATA (High byte)	0x00	5
DATA	-	-

After receiving of the packet with CODE_CMD_REQUEST command, the User should send a data transfer packet, which contains authentication password (8 bytes). The default password is 0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF. The controller doesn't check version of the communication protocol (field VER) in this data packet. This password can be changed using data transmission command CODE_CMD_PASSWORD_SET.

2. From the User:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	CODE_CMD_REQUEST = 0x00	2
CMD_IDENTIFICATION (1 byte)	x	3
LENGTH_DATA (Low byte)	0x08	4
LENGTH_DATA (High byte)	0x00	5
DATA [0] (Password Low byte)	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 High byte)	x	13

The controller checks received password and sends a response, which contains a result. CMD_TYPE of the response is CODE_CMD_RESPONSE, the data field of the response contains COMMANDS_RETURN_DATA structure. Please, learn the COMMANDS_RETURN_DATA structure below in this manual.

3. From the controller:

Field	Value		Packet data order
XOR (1 byte)	x		0
VER (1 byte)	x		1
CMD_TYPE (1 byte)	CODE_CMD_RESPONSE = 0x01		2
CMD_IDENTIFICATION (1 byte)	x		3
LENGTH_DATA (Low byte)	sizeof(COMMANDS_RETURN_DATA_Type	0x07	4

LENGTH_DATA (High byte)		0x00	5
DATA [0] - Low byte (COMMANDS_RETURN_DATA Low byte)	x		6
DATA [1]	x		7
DATA [2] = ERROR_OR_COMMAND	x		8
DATA [3]	0		9
DATA [4]	0		10
DATA [5]	0		11
DATA [6] - High byte (COMMANDS_RETURN_DATA High byte)	0		12

In case of the correct password, the Controller allows the further access to the controller and ERROR_OR_COMMAND field = OK_ACCESS. In case of the incorrect password the code ERROR_OR_COMMAND= ERROR_ACCESS and the Controller closes the connection. The next connection and authentication attempt is possible no earlier than in 1 second. In case of authentication attempt is done before this timeout, the controller send CODE_CMD_RESPONSE with code ERROR_OR_COMMAND = ERROR_ACCESS_TIMEOUT. Such timeout prevents guessing the password.

4.6.2 Data transmission command CODE_CMD_RESPONSE

The data transfer packet with CODE_CMD_RESPONSE code is sent from the controller to the user as a response to some data transmission commands - CODE_CMD_POWERSTEP01, CODE_CMD_CONFIG_SET, CODE_CMD_ID_SET, CODE_CMD_POWERSTEP01_W_MEM, and in case of errors occur. The data field of the packet contains COMMANDS_RETURN_DATA structure. Please, learn the COMMANDS_RETURN_DATA structure below in this manual.

From the controller:

Field	Value		Packet data order
XOR (1 byte)	x		0
VER (1 byte)	x		1
CMD_TYPE (1 byte)	CODE_CMD_RESPONSE = 0x01		2
CMD_IDENTIFICATION (1 byte)	x		3
LENGTH_DATA (Low byte)	sizeof(COMMANDS_RETURN_DATA_Type	0x07	4
LENGTH_DATA (High byte)		0x00	5
DATA [0] - Low byte (COMMANDS_RETURN_DATA)	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] - High byte (COMMANDS_RETURN_DATA)	x		12

4.6.3 Data transmission command CODE_CMD_POWERSTEP01

The data transmission command CODE_CMD_POWERSTEP01 is used to control the stepper motor. Data field of the packet contains SMSD_CMD_Type structure.

Please, learn detailed information about the stepper motor control commands and SMSD_CMD_Type structure below in this manual.

From the User:

Field	Value		Packet data order
XOR (1 byte)	x		0
VER (1 byte)	x		1
CMD_TYPE (1 byte)	CODE_CMD_POWERSTEP01 = 0x02		2
CMD_IDENTIFICATION (1 byte)	x		3
LENGTH_DATA (Low byte)	sizeof(SMSD_CMD_Type)=0x04	0x04	4
LENGTH_DATA (High byte)		0x00	5
DATA [0] (SMSD_CMD_Type Low byte)	x		6
DATA [1]	x		7
DATA [2]	x		8
DATA [3] (SMSD_CMD_Type High byte)	x		9

As a response the Controller sends a result in the packet with CMD_TYPE field = CODE_CMD_POWERSTEP01, the data field contains COMMANDS_RETURN_DATA structure.

From the controller:

Field	Value		Packet data order
XOR (1 byte)	x		0
VER (1 byte)	x		1
CMD_TYPE (1 byte)	CODE_CMD_POWERSTEP01 = 0x02		2
CMD_IDENTIFICATION (1 byte)	x		3
LENGTH_DATA (Low byte)	sizeof(COMMANDS_RETURN_DATA_Type	0x07	4
LENGTH_DATA (High byte)		0x00	5
DATA [0] - Low byte (COMMANDS_RETURN_DATA)	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] - High byte (COMMANDS_RETURN_DATA)	x		12

The content of COMMANDS_RETURN_DATA_Type depends on a command sent from the User.

4.6.4 Data transmission command CODE_CMD_POWERSTEP01_W_MEM0..MEM3

Four data transmission commands - CODE_CMD_POWERSTEP01_W_MEM0, CODE_CMD_POWERSTEP01_W_MEM1, CODE_CMD_POWERSTEP01_W_MEM2, CODE_CMD_POWERSTEP01_W_MEM3 are used to write an executing program into the four memory banks of the Controller accordingly. DATA field of the packet contains the sequence of executing commands. The maximum

quantity of the commands in a sequence – 255. The code distance in the address space is 4 bytes. Every command corresponds to SMSD_CMD_Type structure.

From the User:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	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 byte)	x	3
LENGTH_DATA (Low byte)	x	4
LENGTH_DATA (High byte)	x	5
(1st executing command) DATA [0] (SMSD_CMD_Type Low byte)	x	6
DATA [1]	x	7
DATA [2]	x	8
(1st executing command) DATA [3] (SMSD_CMD_Type High byte)	x	9
……..	……	…….
(last executing command – total n commands) DATA [0] (SMSD_CMD_Type Low byte)	x	n*4 - 3
DATA [1]	x	n*4 – 2
DATA [2]	x	n*4 – 1
(last executing command – total n commands) DATA [3] (SMSD_CMD_Type High byte)	x	n*4

n<=255.

As a response the controller sends a packet with CMD_TYPE = CODE_CMD_RESPONSE.

4.6.5 Data transmission command CODE_CMD_POWERSTEP01_R_MEM0..MEM3

Four data transmission commands - CODE_CMD_POWERSTEP01_R_MEM0, CODE_CMD_POWERSTEP01_R_MEM1, CODE_CMD_POWERSTEP01_R_MEM2, CODE_CMD_POWERSTEP01_R_MEM3 are used to read an executing program from the four memory banks of the Controller accordingly.

From the User:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	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 byte)	x	3
LENGTH_DATA (Low byte)	0	4
LENGTH_DATA (High byte)	0	5
DATA	-	-

As a response the controller sends a packet with the same 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). The DATA field contains the executing commands sequence. The code distance in the address space is 4 bytes. Every command corresponds to SMSD_CMD_Type structure.

From the Controller:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	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 byte)	x	3
LENGTH_DATA (Low byte)	x	4
LENGTH_DATA (High byte)	x	5
(1st executing command) DATA [0] (SMSD_CMD_Type Low byte)	x	6
DATA [1]	x	7
DATA [2]	x	8
(1st executing command) DATA [3] (SMSD_CMD_Type High byte)	x	9
……..	……	……..
(last executing command – total n commands) DATA [0] (SMSD_CMD_Type Low byte)	x	n*4 - 3
DATA [1]	x	n*4 – 2
DATA [2]	x	n*4 – 1
(last executing command – total n commands) DATA [3] (SMSD_CMD_Type High byte)	x	n*4

n<=255.

4.6.6 Data transmission command CODE_CMD_CONFIG_SET

The data transmission commands CODE_CMD_CONFIG_SET is intended to change LAN connection parameters of the controller. The DATA field of the packet contains LAN parameters as a SMSD_LAN_CONFIG_Type structure.

From the User:

Field	Value		Packet data order
XOR (1 byte)	x		0
VER (1 byte)	x		1
CMD_TYPE (1 byte)	CODE_CMD_CONFIG_SET = 0x0B		2
CMD_IDENTIFICATION (1 byte)	X		3
LENGTH_DATA (Low byte)	Sizeof(SMSD_LAN_CONFIG_Type)	0x19	4
LENGTH_DATA (High byte)		0	5
DATA [0] (SMSD_LAN_CONFIG_Type – Low byte)	x		6
……	……		…….
DATA [24] (SMSD_LAN_CONFIG_Type – High byte)	x		30

As a response the controller sends a packet with CMD_TYPE = CODE_CMD_RESPONSE.

4.6.7 6 Data transmission command CODE_CMD_CONFIG_GET

The data transmission commands CODE_CMD_CONFIG_GET is intended to read LAN connection parameters from the controller.

From the User:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	CODE_CMD_CONFIG_GET = 0x0C	2
CMD_IDENTIFICATION (1 byte)	X	3
LENGTH_DATA (Low byte)	0	4
LENGTH_DATA (High byte)	0	5
Data	-	-

As a response the controller sends a packet with CMD_TYPE = CODE_CMD_CONFIG_GET. The DATA field of the packet contains LAN parameters as a SMSD_LAN_CONFIG_Type structure.

From the Controller:

Field	*Value*		*Packet data order*
XOR (1 byte)	**x**		**0**
VER (1 byte)	**x**		**1**
CMD_TYPE (1 byte)	**CODE_CMD_CONFIG_GET = 0x0C**		**2**
CMD_IDENTIFICATION (1 byte)	**X**		**3**
LENGTH_DATA (Low byte)	**Sizeof(SMSD_LAN_CONFIG_Type)**	**0x19**	**4**

LENGTH_DATA (High byte)		**0**	**5**
DATA [0] (SMSD_LAN_CONFIG_Type – Low byte)	**x**		**6**
……	**……..**		**…….**
DATA [24] (SMSD_LAN_CONFIG_Type – High byte)	**x**		**30**

4.6.8 Data transmission command CODE_CMD_ PASSWORD_SET

The data transmission commands CODE_CMD_PASSWORD_SET is intended to change the authentication password.

From the User:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	CODE_CMD_PASSWORD_SET = 0x0D	2
CMD_IDENTIFICATION (1 byte)	x	3
LENGTH_DATA (Low byte)	0x08	4
LENGTH_DATA (High byte)	0x00	5
DATA [0] (Password Low byte)	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 High byte)	x	13

As a response the controller sends a packet with CMD_TYPE = CODE_CMD_RESPONSE.

4.6.9 Data transmission command CODE_CMD_ERROR_GET

The data transmission commands CODE_CMD_ERROR_GET is intended to read from the controller information about number of operation mode starts and error statistics.

From the User:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	CODE_CMD_ERROR_GET = 0x0E	2
CMD_IDENTIFICATION (1 byte)	x	3
LENGTH_DATA (Low byte)	0	4
LENGTH_DATA (High byte)	0	5
Data	-	-

As a response the controller sends a packet with CMD_TYPE = CODE_CMD_ERROR_GET.

From the Controller:

Field	Value	Packet data order
XOR (1 byte)	x	0
VER (1 byte)	x	1
CMD_TYPE (1 byte)	CODE_CMD_ERROR_GET = 0x0E	2
CMD_IDENTIFICATION (1 byte)	x	3
LENGTH_DATA (Low byte)	0x44 (=17*4)	4
LENGTH_DATA (High byte)	0	5
Data[0]	x	6
……..	…….	…….
Data[67]	x	73

The DATA field of the Controller response contains 17 successive values of 4-bytes variables, which represent event counters:

N_STARTS – counter of stepper motor phases energizing
ERROR_XT – quantity of internal errors of clock enables
ERROR_TIME_OUT – quantity of timeout errors of the main process executing
ERROR_INIT_POWERSTEP01 – quantity of chip PowerSTEP01 initialization failures
ERROR_INIT_WIZNET – quantity of chip W5500 initialization failures
ERROR_INIT_FRAM - quantity of memory chip FRAM initialization failures
ERROR_SOCKET – quantity of LAN connection errors
ERROR_FRAM – quantity of errors of data exchange with the memory chip FRAM.
ERROR_INTERRUPT – quantity of interrupt handling errors
ERROR_EXTERN_5V – quantity of current overloads of the internal 5VDC power source
ERROR_EXTERN_VDD – quantity of exceeding the limits of power supply voltage
ERROR_THERMAL_POWERSTEP01 – quantity of chip PowerSTEP01 overheatings
ERROR_THERMAL_BRAKE – quantity of the brake resistor overheatings
ERROR_COMMAND_POWERSTEP01 – quantity of errors during commands transfer to the chip PowerSTEP01
ERROR_UVLO_POWERSTEP01 – for internal use
ERROR_STALL_POWERSTEP01 – for internal use
ERROR_WORK_PROGRAM – quantity of program executing errors

5. The structure of COMMANDS_RETURN_DATA_Type

In the Controller responses with CMD_TYPE = CODE_CMD_RESPONSE, CODE_CMD_POWERSTEP01 the field DATA contains COMMANDS_RETURN_DATA_Type structure:

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

STATUS_POWERSTEP01 – 16-bits length field, contains state flags of the current stepmotor control system status. As the information is important, this field is included to all response packets with COMMANDS_RETURN_DATA_Type structure;
ERROR_OR_COMMAND – 1 byte field – the command result;
RETURN_DATA – 4 byte field – the information data of the response.

5.1 Bits assignments of the STATUS_POWERSTEP01 field

The current stepper motor control system status is described in the structure 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 – phases Z-state: HiZ = 1 – phases deenergized, Hiz = 0 – phases energized;
BUSY – standby: BUSY = 1 – the Controller is ready for the next command, BUSY = 0 – the Controller is executing a previous instruction;
SW_F – SW_F = 1 – the function SW is turned ON, SW_F = 0 – the function SW is turned OFF;
SW_EVN – the flag of SW event: SW = 1 – the event happened, SW = 0 – not happened;
DIR – rotation direction: DIR = 1 – forward rotation, DIR = 0 – backward rotation;
MOT_STATUS – motor running state: MOT_STATUS = 0 – motor stop, MOT_STATUS = 1 – motor accelerates, MOT_STATUS = 2 – motor decelerates, MOT_STATUS = 3 – steady rotation of the motor;
CMD_ERROR – command executing error: CMD_ERROR = 1 – error; : CMD_ERROR = 0 – no error.

5.2 Possible meanings of the field ERROR_OR_COMMAND

Numerical values of the field ERROR_OR_COMMAND start from 0 and gradually-increase. The list of possible values is the next:
OK - command accepted without errors;
OK_ACCESS - successful authentication (the User has got access to the Controller control);
ERROR_ACCESS - authentication error (the User has not got access to the Controller control);
ERROR_ACCESS_TIMEOUT – authentication timeout is not elapsed (authentication timeout is 1 sec);
ERROR_XOR - checksum error;
ERROR_NO_COMMAND - the command does not exist;
ERROR_LEN - the packet length error;
ERROR_RANGE - exceeding values limits;
ERROR_WRITE - writing error;
ERROR_READ - reading error;
ERROR_PROGRAMS - program error;
ERROR_WRITE_SETUP
NO_NEXT - no next command;
END_PROGRAMS - end of program;
COMMAND_GET_STATUS_IN_EVENT – the field RETURN_DATA contains the bit map of input signals;
COMMAND_GET_MODE - the field RETURN_DATA contains the bit map of the Controller parameters;
COMMAND_GET_ABS_POS - the field RETURN_DATA contains the current position of the stepper motor (measured as steps);
COMMAND_GET_EL_POS - the field RETURN_DATA contains the current electrical position of the rotor;
COMMAND_GET_SPEED - the field RETURN_DATA contains the current motor speed;
COMMAND_GET_MIN_SPEED - the field RETURN_DATA contains the current set minimum motor speed;
COMMAND_GET_MAX_SPEED - the field RETURN_DATA contains the current set maximum motor speed;
COMMAND_GET_STACK - the field RETURN_DATA contains information about executing program number and command number;
STATUS_RELE_SET – relay is turned ON;
STATUS_RELE_CLR – relay is turned OFF;

6. The executing commands SMSD_CMD_Type

The executing commands structure SMSD_CMD_TYPE is the next:

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

RESERVE – 3 bit field, not used;
ACTION - 1 bit field – for internal use, send as 0;
COMMAND - 6 bits field - the executing command code;
DATA - 22 bits field - the command parameter; if the command doesn't need a parameter, this field value = 0x00 (22 bits are filled in with 0).
The whole structure size is always 4 bytes.
The structure SMSD_CMD_Type is used in data transmission packets, which include executing commands: CMD_TYPE = CODE_CMD_POWERSTEP01, CODE_CMD_POWERSTEP01_W_MEM0…MEM3, CODE_CMD_POWERSTEP01_R_MEM0…MEM3.

Numerical values of the field COMMAND start from 0 and gradually-increase. List of executing commands codes is below:

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

The bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte [2]			Byte [1]			Byte [0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data (Command parameter)							Command (CMD_PowerSTEP01 command code)						Action	Reserve		

6. 1 Executing command CMD_PowerSTEP01_END

The executing CMD_PowerSTEP01_END = 0x00 is intended to mark the end of executing program.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0						Command code CMD_PowerSTEP01_END = 0x00							Action	Reserve		
Bit value	0						0	0		0	0	0	0	0	0	0	0

6.2 Executing command CMD_PowerSTEP01_GET_SPEED

The executing command CMD_PowerSTEP01_GET_SPEED = 0x01 is intended for reading of the current motor speed.

The important notice: for the correct response to the CMD_PowerSTEP01_GET_SPEED command the minimum speed should be set = 0x00 by command CMD_PowerSTEP01_SET_MIN_SPEED before sending the command CMD_PowerSTEP01_GET_SPEED. Otherwise the result could be wrong for low speed movement and stops.

Below is an example of data transmission packet for reading the current speed in a real-time mode:

From the User:

Field	Value		Packet data order
XOR (1 byte)	x		0
VER (1 byte)	x		1
CMD_TYPE (1 byte)	CODE_CMD_POWERSTEP01 = 0x02		2
CMD_IDENTIFICATION (1 byte)	x		3
LENGTH_DATA (Low byte)	sizeof(SMSD_CMD_Type)=0x04	0x04	4
LENGTH_DATA (High byte)		0x00	5
DATA [0] (SMSD_CMD_Type Low byte)	0x10		6
DATA [1]	0x00		7
DATA [2]	0x00		8
DATA [3] (SMSD_CMD_Type High byte)	0x00		9

DATA field of the packet = SMSD_CMD_Type structure, which contains the command CMD_PowerSTEP01_GET_SPEED.

Bit mapping of the SMSD_CMD_Type structure:

Data field byte	DATA[3]=0x00			DATA[2]=0x00			DATA[1]=0x00			DATA[0]=0x10							
	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_GET_SPEED = 0x01						Action	Reserve		
Bit value	0			0			0	0	0	0	0	0	1	0	0	0	0

The bit mapping of the SMSD_CMD_Type structure is the same for all the executing commands.

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_SPEED, RETURN_DATA – the value of the current motor speed.

From the controller:

Field	Value		Packet data order
XOR (1 byte)	x		0
VER (1 byte)	x		1
CMD_TYPE (1 byte)	CODE_CMD_RESPONSE = 0x01		2
CMD_IDENTIFICATION (1 byte)	x		3
LENGTH_DATA (Low byte)	sizeof(COMMANDS_RETURN_DATA_Type	0x07	4
LENGTH_DATA (High byte)		0x00	5
DATA [0] - Low byte (COMMANDS_RETURN_DATA Low byte)	x		6
DATA [1]	x		7
DATA [2] = ERROR_OR_COMMAND	= COMMAND_GET_SPEED		8
DATA [3] = RETURN_DATA[0]	x (Low byte of the current motor speed)		9
DATA [4] = RETURN_DATA[1]	x		10
DATA [5] = RETURN_DATA[2]	x		11
DATA [6] - High byte (COMMANDS_RETURN_DATA High byte) = RETURN_DATA[3]	x (High byte of the current motor speed)		12

6.3 Executing command CMD_PowerSTEP01_STATUS_IN_EVENT

The executing command CMD_PowerSTEP01_STATUS_IN_EVENT = 0x02 is intended for reading information about current signals inputs state.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_ STATUS_IN_EVENT = 0x02						Action	Reserve		
Bit value	0			0			0	0	0	0	0	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_STATUS_IN_EVENT, RETURN_DATA – the bit mapping of inputs state:

RETURN_DATA field byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 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 – Event at the input X: 1 – happened, 0 – not happened;
Mask_X – Masking of the input X;
Wait_X – Waiting of the input X.

6.4 Executing command CMD_PowerSTEP01_SET_MODE

The executing command CMD_PowerSTEP01_SET_MODE = 0x03 is intended for setting motor and control parameters.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte [2]			Byte [1]			Byte [0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data field of SMSD_CMD_Type							Command code CMD_PowerSTEP01_SET_MODE = 0x03						Action	Reserve		
Bit value	Depend on Data field value							0	0	0	0	1	1	0	0	0	0

Bit mapping of the Data field of the SMSD_CMD_Type structure:

Byte[3] – bits 7..0								Byte[2] – bits 7..0								Byte[1] bits 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 - motor control mode:
0 – voltage mode,
1 – current mode

MOTOR_TYPE – motor type for the voltage control mode:

Value		Max. current per phase, Amp	Resistance per phase, Ohm	Inductance per phase, mH	Step angle	Motor model
SMSD-4.2LAN	SMSD-8.0LAN					
0	0	-	-	-	-	No motor
1	1	1.33	2.1	2.5	1.8	
2	2	1.33	2.1	4.2	0.9	
3	3	1.2	3.3	3.4	0.9	
4	4	1.68	1.65	3.2	1.8	
5	5	1.68	1.64	3.2	0.9	
6	6	1.2	3.3	2.8	0.8	
7	7	1.68	1.65	2.8	1.8	SM4247
8	8	1.68	1.65	4.1	0.9	
9	9	1.2	6	7	1.8	
10	10	1.2	12.1	36.7	0.9	
11	11	1.56	1.8	3.6	1.8	
12	12	1.0	16.7	46.5	1.8	
13	13	1.5	3.6	6	1.8	
14	14	1.0	5.7	5.4	1.8	
15	15	1.0	5.7	8	0.9	
16	16	2.8	0.7	1.4	1.8	
17	17	2.8	0.7	2.2	0.9	
18	18	1.0	6.6	8.6	1.8	
19	19	2.8	0.83	2.2	1.8	
20	20	2.8	0.9	3.7	0.9	
21	21	1.0	7.4	10	1.8	
22	22	2.0	1.8	2.5	1.8	
23	23	2.8	0.9	2.5	1.8	
24	24	1.0	8.6	14	1.8	
25	25	2.8	1.13	3.6	1.8	SM5776
26	26	2.8	1.13	5.6	0.9	
27	27	2.0	1.2	4.6	1.8	
28	28	2.0	4.8	18.4	1.8	
29	29	2.0	1.5	6.8	1.8	
30	30	2.0	6	7.2	1.8	
31	31	2.8	0.7	3.9	1.8	
32	32	2.8	2.8	15.6	1.8	
33	33	4.2	0,375	3.4	1.8	SM8680 Parallel connection
34	34	4.2	1.5	13.6	1.8	SM8680 Serial connection
35	35	4.2	0.45	6	1.8	-

36	36	4.2	1.8	24	1.8	-
37	37	4.2	0,625	8	1.8	-
38	38	4.2	2.5	32	1.8	-
-	39	6.0	0.6	6.5	1.8	-
-	40	6.2	0.75	9	1.8	-
-	41	5.5	0.9	12	1.8	-
-	42	6.5	0.8	15	1.8	-
-	43	8	0.67	12	1.8	SM110201
39	44	0.3	32	40	1.8	-
40	45	0.67	8.5	7.5	1.8	-
41	46	1.68	2.3	3.4	1.8	-
42	47	3.0	1.0	3.4	1.8	-
43	48	3.0	1.45	6.5	1.8	-
44	49	3.0	1.2	6.4	1.8	-
45	50	4.5	0.36	3.0	1.8	-
-	51	6.0	0.6	5.7	1.8	-
-	52	6.2	0.7	8.5	1.8	-
-	53	8.0	0.8	16	1.8	-
-	54	6.0	0.8	8.7	1.8	-

MICROSTEPPING – the motor main step dividing:

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

WORK_CURRENT – operating current for the current control mode. The motor operation current is calculated as 0.1Amp*Value; 1<=Value<=80. Available range for controllers SMSD-4.2LAN: 1 – 42; for controllers SMSD-8.0LAN: 1 – 80. The values are the next:

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	

STOP_CURRENT – holding current – as a percentage of an operating current:

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

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.5 Executing command CMD_PowerSTEP01_GET_MODE

The executing command CMD_PowerSTEP01_GET_MODE = 0x04 is intended for reading motor control parameters from the controller.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_ GET_MODE = 0x04						Action	Reserve		
Bit value	0			0			0	0	0	0	1	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_MODE, RETURN_DATA contains the information about motor and control parameters:

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
Not use		PROGRAM_N		STOP_ CURRENT			WORK_ CURRENT		MICRO- STEPPING		MOTOR_ TYPE	CURRENT_ OR_ VOLTAGE	

Fields STOP_CURRENT, WORK_CURRENT, MICROSTEPPING, MOTOR_TYPE, CURRENT_OR_VOLTAGE are the same as in the executing command CMD_PowerSTEP01_SET_MODE.

Field PROGRAM_N contains a number of program, which is available to be started by external signals.

6.6 Executing command CMD_PowerSTEP01_SET_MIN_SPEED

The executing command CMD_PowerSTEP01_SET_MIN_SPEED = 0x05 is intended for setting the motor minimum speed. The DATA field should contain the speed value in range 0 – 950 steps/sec.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Min Speed Value							Command code CMD_PowerSTEP01_ SET_MIN_SPEED = 0x05						Action	Reserve		
Bit value	Depend on Data value							0	0	0	1	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.7 Executing command CMD_PowerSTEP01_SET_MAX_SPEED

The executing command CMD_PowerSTEP01_SET_MAX_SPEED = 0x06 is intended for setting the motor maximum speed. The DATA field should contain the speed value in range 16 – 15600 steps/sec.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Max Speed Value							Command code CMD_PowerSTEP01_ SET_MAX_SPEED = 0x06						Action	Reserve		
Bit value	Depend on Data value							0	0	0	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.8 Executing command CMD_PowerSTEP01_SET_ACC

The executing command CMD_PowerSTEP01_SET_ACC = 0x07 is intended for setting the motor acceleration to getting the motor maximum speed. The DATA field should contain the acceleration value in range 15 – 59000 steps/sec^2.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Acceleration Value							Command code CMD_PowerSTEP01_SET_ACC = 0x07						Action	Reserve		
Bit value	Depend on Data value							0	0	0	1	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.9 Executing command CMD_PowerSTEP01_SET_DEC

The executing command CMD_PowerSTEP01_SET_DEC = 0x08 is intended for setting the motor deceleration. The DATA field should contain the DECELERATION value in range 15 – 59000 steps/sec^2.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Deceleration Value							Command code CMD_PowerSTEP01_SET_DEC = 0x08						Action	Reserve		
Bit value	Depend on Data value							0	0	1	0	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.10 Executing command CMD_PowerSTEP01_SET_FS_SPEED

The executing command CMD_PowerSTEP01_SET_FS_SPEED = 0x09 is intended for setting the running speed, when the motor switches to a full step mode. The DATA field should contain the speed value in range 15 – 15600 steps/sec.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Full Step Speed Value						Command code CMD_PowerSTEP01_ SET_FS_SPEED = 0x09							Action	Reserve		
Bit value	Depend on Data value						0	0		1	0	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.11 Executing command CMD_PowerSTEP01_SET_MASK_EVENT

The executing command CMD_PowerSTEP01_SET_MASK_EVENT = 0x0A is intended for masking input signals. If the input signal MASK value = 1 – the Controller handles the signal state at the physical input. If the signal MASK is 0 – the controller doesn't take a care the physical input state.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Signals Mask state						Command code CMD_PowerSTEP01_ SET_MASK_EVENT = 0x0A							Action	Reserve		
Bit value	Depend on Data value						0	0		1	0	1	0	0	0	0	0

The Data bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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 – Masking of the input X.

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.12 Executing command CMD_PowerSTEP01_GET_ABS_POS

The executing command CMD_PowerSTEP01_GET_ABS_POS = 0x0B is intended for reading the current motor position.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_ GET_ABS_POS = 0x0B						Action	Reserve		
Bit value	0			0			0	0	0	1	0	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_ABS_POS, RETURN_DATA contains the value of the motor current position in a range –(2^21)…+(2^21-1).

6.13 Executing command CMD_PowerSTEP01_GET_EL_POS

The executing command CMD_PowerSTEP01_GET_EL_POS = 0x0C is intended for reading the current motor electrical microstepping position.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_ GET_EL_POS = 0x0C						Action	Reserve		
Bit value	0			0			0	0	0	1	1	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_EL_POS, RETURN_DATA contains the value of the motor current electrical microstepping position: bits 8,7 – current step, bits 6..0 – current microstep inside the current full step (measured as 1/128 of the full step):

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
Not used																								Current step							Current microstep							

6.14 Executing command CMD_PowerSTEP01_GET_STATUS_AND_CLR

The executing command CMD_PowerSTEP01_GET_STATUS_AND_CLR = 0x0D is intended for reading the current state of the controller, and the Controller clears all error flags.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_ GET_STATUS_AND_CLR = 0x0D						Action	Reserve		
Bit value	0			0			0	0	0	1	1	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.15 Executing command CMD_PowerSTEP01_RUN_F

The executing command CMD_PowerSTEP01_RUN_F = 0x0E is intended to start motor rotation in forward direction at designated speed. The DATA field should contain the final rotation speed value in range 15 – 15600 steps/sec.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Final rotation Speed							Command code CMD_PowerSTEP01_ RUN_F = 0x0E						Action	Reserve		
Bit value	Depend on Data value							0	0	1	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.16 Executing command CMD_PowerSTEP01_RUN_R

The executing command CMD_PowerSTEP01_RUN_R = 0x0F is intended to start motor rotation in backward direction at designated speed. The DATA field should contain the final rotation speed value in range 15 – 15600 steps/sec.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Final rotation Speed							Command code CMD_PowerSTEP01_ RUN_R = 0x0F						Action	Reserve		
Bit value	Depend on Data value							0	0	1	1	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.17 Executing command CMD_PowerSTEP01_MOVE_F

The executing command CMD_PowerSTEP01_MOVE_F = 0x10 is intended for motor displacement in forward direction. The DATA field should contain the displacement value in range –(2^21)…+(2^21-1). The motion speed is determined by specified minimum and maximum speed and acceleration value. The motor should be stopped before executing this command (field Mot_Status of the powerSTEP_STATUS_Type structure = 0).

Attention: the speed commands are always set as full steps per second. The motion commands are always set as microstepping measured displacements.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Displacement							Command code CMD_PowerSTEP01_ MOVE_F = 0x10						Action	Reserve		
Bit value	Depend on Data value							0	1	0	0	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.18 Executing command CMD_PowerSTEP01_MOVE_R

The executing command CMD_PowerSTEP01_MOVE_R = 0x11 is intended for motor displacement in backward direction. The DATA field should contain the displacement value in range –(2^21)…+(2^21-1). The motion speed is determined by specified minimum and maximum speed and acceleration value. The motor should be stopped before executing this command (field Mot_Status of the powerSTEP_STATUS_Type structure = 0).

Attention: the speed commands are always set as full steps per second. The motion commands are always set as microstepping measured displacements.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Displacement							Command code CMD_PowerSTEP01_ MOVE_R = 0x11						Action	Reserve		
Bit value	Depend on Data value							0	1	0	0	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.19 Executing command CMD_PowerSTEP01_GO_TO_F

The executing command CMD_PowerSTEP01_GO_TO_F = 0x12 is intended for motor displacement to the specified position in forward direction. The DATA field should contain the position value in range –(2^21)…+(2^21-1). The motion speed is determined by specified minimum and maximum speed and acceleration value.

Attention: the speed commands are always set as full steps per second. The motion commands are always set as microstepping measured displacements.

Bit mapping of the SMSD_CMD_Type structure:

	Byte [3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Desired position							Command code CMD_PowerSTEP01_ GO_TO_F = 0x12						Action	Reserve		
Bit value	Depend on Data value							0	1	0	0	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.20 Executing command CMD_PowerSTEP01_GO_TO_R

The executing command CMD_PowerSTEP01_GO_TO_R = 0x13 is intended for motor displacement to the specified position in backward direction. The DATA field should contain the position value in range: –(2^21)…+(2^21-1). The motion speed is determined by specified minimum and maximum speed and acceleration value.

Attention: the speed commands are always set as full steps per second. The motion commands are always set as microstepping measured displacements.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Desired position							Command code CMD_PowerSTEP01_ GO_TO_R = 0x13						Action	Reserve		
Bit value	Depend on Data value							0	1	0	0	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.21 Executing command CMD_PowerSTEP01_GO_UNTIL_F

The executing command CMD_PowerSTEP01_GO_UNTIL_F = 0x14 is intended for the motor forward motion at the maximum speed until receiving a signal at the input SW (taking into account the signal masking). After that the motor decelerates and stops. The MASK state of the signal can be changed by the executing command CMD_PowerSTEP01_SET_MASK_EVENT

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Signal number							Command code command CMD_PowerSTEP01_ GO_UNTIL_F = 0x14						Action	Reserve		
Bit value	Depend on Data value							0	1	0	1	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.22 Executing command CMD_PowerSTEP01_GO_UNTIL_R

The executing command CMD_PowerSTEP01_GO_UNTIL_F = 0x14 is intended for the motor backward motion at the maximum speed until receiving a signal at the input SW (taking into account the signal masking). After that the motor decelerates and stops. The MASK state of the signal can be changed by the executing command CMD_PowerSTEP01_SET_MASK_EVENT

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Signal number							Command code command CMD_PowerSTEP01_ GO_UNTIL_R = 0x15						Action	Reserve		
Bit value	Depend on Data value							0	1	0	1	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.23 Executing command CMD_PowerSTEP01_SCAN_ZERO_F

The executing command CMD_PowerSTEP01_SCAN_ZERO_F = 0x16 is intended for searching zero position in a forward direction. The movement continues until signal to SET_ZERO input received. The DATA field determines the motion speed during searching the zero position.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Motion Speed							Command code CMD_PowerSTEP01_ SCAN_ZERO_F = 0x16						Action	Reserve		
Bit value	Depend on Data value							0	1	0	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.24 Executing command CMD_PowerSTEP01_SCAN_ZERO_R

The executing command CMD_PowerSTEP01_SCAN_ZERO_R = 0x17 is intended for searching zero position in a backward direction. The movement continues until signal to SET_ZERO input received. The DATA field determines the motion speed during searching the zero position.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Motion Speed							Command code CMD_PowerSTEP01_ SCAN_ZERO_R = 0x17						Action	Reserve		
Bit value	Depend on Data value							0	1	0	1	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.25 Executing command CMD_PowerSTEP01_SCAN_LABEL_F

The executing command CMD_PowerSTEP01_SCAN_LABEL_F = 0x18 is intended for searching LABEL position in a forward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the LABEL position.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Motion Speed							Command code CMD_PowerSTEP01_ SCAN_LABEL_F = 0x18						Action	Reserve		
Bit value	Depend on Data value							0	1	1	0	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.26 Executing command CMD_PowerSTEP01_SCAN_LABEL_R

The executing command CMD_PowerSTEP01_SCAN_LABEL_R = 0x19 is intended for searching LABEL position in a backward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the LABEL position.

Attention: the speed commands are always set as full steps per second.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Motion Speed							Command code CMD_PowerSTEP01_ SCAN_LABEL_R = 0x19						Action	Reserve		
Bit value	Depend on Data value							0	1	1	0	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.27 Executing command CMD_PowerSTEP01_GO_ZERO

The executing command CMD_PowerSTEP01_GO_ZERO = 0x1A is intended for movement to the ZERO position.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_GO_ZERO = 0x1A						Action	Reserve		
Bit value	0							0	1	1	0	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.28 Executing command CMD_PowerSTEP01_GO_LABEL

The executing command CMD_PowerSTEP01_GO_LABEL = 0x1B is intended for movement to the LABEL position.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_GO_LABEL = 0x1B						Action	Reserve		
Bit value	0							0	1	1	0	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.29 Executing command CMD_PowerSTEP01_GO_TO

The executing command CMD_PowerSTEP01_GO_TO = 0x1C is intended for the shortest movement to the specified position.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = specified position							Command code CMD_PowerSTEP01_GO_TO = 0x1C						Action	Reserve		
Bit value	Depend on Data value							0	1	1	1	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.30 Executing command CMD_PowerSTEP01_RESET_POS

The executing command CMD_PowerSTEP01_RESET_POS = 0x1D is intended to set ZERO position (to clear internal steps counter and specify a current position as a ZERO position).

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_ RESET_POS = 0x1D						Action	Reserve		
Bit value	0							0	1	1	1	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.31 Executing command CMD_PowerSTEP01_RESET_POWERSTEP01

The executing command CMD_PowerSTEP01_RESET_POWERSTEP01 = 0x1E is used for hardware and software reset of the stepper motor control module, but not of the whole Controller.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_ RESET_POWERSTEP01 = 0x1E						Action	Reserve		
Bit value	0							0	1	1	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.32 Executing command CMD_PowerSTEP01_SOFT_STOP

The executing command CMD_PowerSTEP01_SOFT_STOP = 0x1F is used for smooth decelerating of the stepper motor and stop. After that the motor holds the current position (with preset holding current).

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_ SOFT_STOP = 0x1F						Action	Reserve		
Bit value	0							0	1	1	1	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.33 Executing command CMD_PowerSTEP01_HARD_STOP

The executing command CMD_PowerSTEP01_HARD_STOP = 0x20 is used for sudden stop of the stepper motor and holding the current position (with preset holding current).

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_ HARD_STOP = 0x20						Action	Reserve		
Bit value	0							1	0	0	0	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.34 Executing command CMD_PowerSTEP01_SOFT_HI_Z

The executing command CMD_PowerSTEP01_SOFT_HI_Z = 0x21 is used for smooth decelerating of the stepper motor and stop. After that the motor phases are deenergized.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_SOFT_HI_Z = 0x21						Action	Reserve		
Bit value	0							1	0	0	0	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.35 Executing command CMD_PowerSTEP01_HARD_HI_Z

The executing command CMD_PowerSTEP01_HARD_HI_Z = 0x22 is used for sudden stop of the stepper motor and deenergizing the stepper motor.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_HARD_HI_Z = 0x22						Action	Reserve		
Bit value	0							1	0	0	0	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.36 Executing command CMD_PowerSTEP01_SET_WAIT

The executing command CMD_PowerSTEP01_SET_WAIT = 0x23 is intended for setting pause. The DATA field contains the waiting time measured as ms. Allowed value range 0 – 3600000 ms.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Waiting time							Command code CMD_PowerSTEP01_ SET_WAIT = 0x23						Action	Reserve		
Bit value	Depend on Data value							1	0	0	0	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.37 Executing command CMD_PowerSTEP01_SET_RELE

The executing command CMD_PowerSTEP01_SET_RELE = 0x24 is intended to turn on the controller relay.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_SET_RELE = 0x24						Action	Reserve		
Bit value	0							1	0	0	1	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = STATUS_RELE_SET.

6.38 Executing command CMD_PowerSTEP01_CLR_RELE

The executing command CMD_PowerSTEP01_CLR_RELE = 0x25 is intended to turn off the controller relay.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_CLR_RELE = 0x25						Action	Reserve		
Bit value	0							1	0	0	1	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = STATUS_RELE_CLR.

6.39 Executing command CMD_PowerSTEP01_GET_RELE

The executing command CMD_PowerSTEP01_GET_RELE = 0x26 is intended to read a current state of the controller relay.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_GET_RELE = 0x26						Action	Reserve		
Bit value	0							1	0	0	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND depend on a real current relay state - STATUS_RELE_SET or STATUS_RELE_CLR.

6.40 Executing command CMD_PowerSTEP01_WAIT_IN0

The executing command CMD_PowerSTEP01_WAIT_IN0 = 0x27 is used to wait until receiving a signal to the input IN0.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_WAIT_IN0= 0x27						Action	Reserve		
Bit value	0							1	0	0	1	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.41 Executing command CMD_PowerSTEP01_WAIT_IN1

The executing command CMD_PowerSTEP01_WAIT_IN1 = 0x28 is used to wait until receiving a signal to the input IN1.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_WAIT_IN1= 0x28						Action	Reserve		
Bit value	0							1	0	1	0	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.42 Executing command CMD_PowerSTEP01_GOTO_PROGRAM

The executing command CMD_PowerSTEP01_GOTO_PROGRAM = 0x29 is intended for unconditional branching – to jump to a specified command number in a specified program number. The DATA field contains the information about a program memory number and a command sequence number: bits 0..7 of the DATA field contain the command number, bits 8,9 of the DATA field contain the program number.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Command and Program numbers							Command code CMD_PowerSTEP01_ GOTO_PROGRAM = 0x29						Action	Reserve		
Bit value	Depend on Data value							1	0	1	0	0	1	0	0	0	0

The Data field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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	Program Number		Command number							

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.43 Executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN0

The executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN0 = 0x2A is intended for conditional branching – to jump to a specified command number in a specified program number if there is a signal at the input IN0. The DATA field contains the information about a program memory number and a command sequence number: bits 0..7 of the DATA field contain the command number, bits 8,9 of the DATA field contain the program number.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Command and Program numbers							Command code CMD_PowerSTEP01_ GOTO_PROGRAM_IF_IN0 = 0x2A						Action	Reserve		
Bit value	Depend on Data value							1	0	1	0	1	0	0	0	0	0

The Data field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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	Program Number		Command number							

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.44 Executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN1

The executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN1 = 0x2B is intended for conditional branching – to jump to a specified command number in a specified program number if there is a signal at the input IN1. The DATA field contains the information about a program memory number and a command sequence number: bits 0..7 of the DATA field contain the command number, bits 8,9 of the DATA field contain the program number.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Command and Program numbers							Command code CMD_PowerSTEP01_ GOTO_PROGRAM_IF_IN1 = 0x2B						Action	Reserve		
Bit value	Depend on Data value							1	0	1	0	1	1	0	0	0	0

The Data field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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	Program Number		Command number							

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.45 Executing command CMD_PowerSTEP01_LOOP_PROGRAM

The executing command CMD_PowerSTEP01_LOOP_PROGRAM = 0x2C is used loop organization – the Controller repeats specified times specified number of commands (start from the first command after this command.

The DATA field contains the information about commands number and cycles number: bits 0..9 of the DATA field contain the commands number, bits 10..19 of the DATA field contain the cycles number.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Commands and Cycles							Command code CMD_PowerSTEP01_ LOOP_PROGRAM = 0x2C						Action	Reserve		
Bit value	Depend on Data value							1	0	1	1	0	0	0	0	0	0

The Data field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	Cycles number										Commands number									

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.46 Executing command CMD_PowerSTEP01_CALL_PROGRAM

The executing command CMD_PowerSTEP01_CALL_PROGRAM = 0x2D is intended for calling a subprogram. The DATA field contains the information about a program memory number and a command sequence number, which starts a subprogram: bits 0..7 of the DATA field contain the command number, bits 8,9 of the DATA field contain the program number. For returning back to the main program, the subprogram should contain a RETURN command - CMD_PowerSTEP01_RETURN_PROGRAM. The subprogram is executed until the CMD_PowerSTEP01_RETURN_PROGRAM and after that returns to the next command of the main program after CMD_PowerSTEP01_CALL_PROGRAM.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Command and Program numbers							Command code CMD_PowerSTEP01_CALL_PRO GRAM = 0x2D						Action	Reserve		
Bit value	Depend on Data value							1	0	1	1	0	1	0	0	0	0

The Data field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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	Program Number		Command number							

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.47 Executing command CMD_PowerSTEP01_RETURN_PROGRAM

The executing command CMD_PowerSTEP01_RETURN_PROGRAM = 0x2E is used to specify the end of a subprogram and to return back to the main program. If previously the command CMD_PowerSTEP01_CALL_PROGRAM was not called, the executing of CMD_PowerSTEP01_RETURN_PROGRAM will call an error.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_ RETURN_PROGRAM = 0x2E						Action	Reserve		
Bit value	0							1	0	1	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.48 Executing command CMD_PowerSTEP01_START_PROGRAM_MEM0

The executing command CMD_PowerSTEP01_START_PROGRAM_MEM0 = 0x2F is used to start program executing from the Controller memory area Mem0.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0							Command code CMD_PowerSTEP01_ START_PROGRAM_MEM0= 0x2F						Action	Reserve		
Bit value	0							1	0	1	1	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

The same commands CMD_PowerSTEP01_START_PROGRAM_MEM1 = 0x30, CMD_PowerSTEP01_START_PROGRAM_MEM2 = 0x31, CMD_PowerSTEP01_START_PROGRAM_MEM3 = 0x32 are used to start an executing program from the Controller memory Mem1, Mem2 and Mem3 accordingly.

6.49 Executing command CMD_PowerSTEP01_STOP_PROGRAM_MEM

The executing command CMD_PowerSTEP01_STOP_PROGRAM_MEM = 0x33 is used to stop executing a program.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0						Command code CMD_PowerSTEP01_ STOP_PROGRAM_MEM = 0x33							Action	Reserve		
Bit value	0						1	1	0	0	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.50 Executing command CMD_PowerSTEP01_STEP_CLOCK

The executing command CMD_PowerSTEP01_STEP_CLOCK = 0x34 is intended to change the control mode to pulse control using external input signals EN, STEP, DIR.

Deprecated command, not used in recent controller versions.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0						Command code CMD_PowerSTEP01_ STEP_CLOCK = 0x34							Action	Reserve		
Bit value	0						1	1	0	0	1	0	0	0	0	0	0

6.51 Executing command CMD_PowerSTEP01_STOP_USB

The executing CMD_PowerSTEP01_STOP_USB = 0x35 is intended to stop data transfer via USB interface.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0						Command code CMD_PowerSTEP01_ STOP_USB = 0x35							Action	Reserve		
Bit value	0						1	1	0	0	1	0	0	0	0	0	0

6.52 Executing command CMD_PowerSTEP01_GET_MIN_SPEED

The executing command CMD_PowerSTEP01_GET_MIN_SPEED = 0x36 is intended for reading of the current set minimum motor speed.

DATA field of the packet = SMSD_CMD_Type structure, which contains the command CMD_PowerSTEP01_GET_MIN_SPEED.

Bit mapping of the SMSD_CMD_Type structure:

Data field byte	DATA[3]=0x00 Byte[3]			DATA[2]=0x00 Byte[2]			DATA[1]=0x00 Byte[1]			DATA[0]=0x10 Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00						Command code CMD_PowerSTEP01_GET_MIN_SPEED = 0x36							Action	Reserve		
Bit value	0			0			0	1	1	0	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_MIN_SPEED, RETURN_DATA – the value of the current set minimum motor speed.

6.53 Executing command CMD_PowerSTEP01_GET_MAX_SPEED

The executing command CMD_PowerSTEP01_GET_MAX_SPEED = 0x37 is intended for reading of the current set maximum motor speed.

DATA field of the packet = SMSD_CMD_Type structure, which contains the command CMD_PowerSTEP01_GET_MAX_SPEED.

Bit mapping of the SMSD_CMD_Type structure:

Data field byte	DATA[3]=0x00			DATA[2]=0x00			DATA[1]=0x00			DATA[0]=0x10							
	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_GET_MAX_SPEED = 0x37						Action	Reserve		
Bit value	0			0			0	1	1	0	1	1	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_MAX_SPEED, RETURN_DATA – the value of the current set maximum motor speed.

6.54 Executing command CMD_PowerSTEP01_GET_STACK

The executing command CMD_PowerSTEP01_GET_STACK = 0x38 is intended for reading from the controller information about current executing command number and program number.

DATA field of the packet = SMSD_CMD_Type structure, which contains the command CMD_PowerSTEP01_GET_STACK.

Bit mapping of the SMSD_CMD_Type structure:

Data field byte	DATA[3]=0x00			DATA[2]=0x00			DATA[1]=0x00			DATA[0]=0x10							
	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = 0x00							Command code CMD_PowerSTEP01_GET_STACK = 0x38						Action	Reserve		
Bit value	0			0			0	1	1	1	0	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = COMMAND_GET_STACK, RETURN_DATA – information about current executing command number (bits 0..7) and program number (bits 8,9).

The RETURN_DATA field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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	Program Number		Command number							

6.55 Executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_ZERO

The executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_ZERO = 0x39 is intended for conditional branching – to jump to a specified command number in a specified program number if the current position value is 0. The DATA field contains the information about a program memory number and a command sequence number: bits 0..7 of the DATA field contain the command number, bits 8,9 of the DATA field contain the program number.

This command is valid from 2d version of communication protocol.

Bit mapping of the SMSD_CMD_Type structure:

Data field byte	DATA[3]=0x00 Byte[3]			DATA[2]=0x00 Byte[2]			DATA[1]=0x00 Byte[1]			DATA[0]=0x10 Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Command and Program numbers							Command code CMD_PowerSTEP01_GOTO_ PROGRAM_IF_ZERO = 0x39						Action	Reserve		
Bit value	Depend on Data value							1	1	1	0	0	1	0	0	0	0

The Data field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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	Program Number		Command number							

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.56 Executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN_ZERO

The executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN_ZERO = 0x3A is intended for conditional branching – to jump to a specified command number in a specified program number if there is a signal at the input SET_ZERO. The DATA field contains the information about a program memory number and a command sequence number: bits 0..7 of the DATA field contain the command number, bits 8,9 of the DATA field contain the program number.

This command is valid from 2d version of communication protocol.

Bit mapping of the SMSD_CMD_Type structure:

Data field byte	DATA[3]=0x00			DATA[2]=0x00			DATA[1]=0x00			DATA[0]=0x10							
	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Command and Program numbers						Command code CMD_PowerSTEP01_GOTO _PROGRAM _IF_IN_ZERO = 0x3A							Action	Reserve		
Bit value	Depend on Data value						1	1		1	0	1	0	0	0	0	0

The Data field bit mapping:

	Byte[3] bits 7..0								Byte[2] bits 7..0								Byte[1] bits 7..2					
Data bit	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	Program Number		Command number							

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.57 Executing command CMD_PowerSTEP01_WAIT_CONTINUE

The executing command CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN_ZERO = 0x3B is intended for waiting of synchronization signal at the input CONTINUE, which is used for synchronization of executing programs in different controllers.

This command is valid from 2d version of communication protocol.

Bit mapping of the SMSD_CMD_Type structure:

Data field byte	DATA[3]=0x00			DATA[2]=0x00			DATA[1]=0x00			DATA[0]=0x10							
	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	0						Command code CMD_PowerSTEP01_GOTO_ PROGRAM_IF_IN_ZERO = 0x3B							Action	Reserve		
Bit value	0						1	1		1	0	1	1	0	0	0	0

6.58 Executing command CMD_PowerSTEP01_SET_WAIT_2

The executing command CMD_PowerSTEP01_SET_WAIT_2 = 0x3C is intended for setting a pause. The DATA field contains the waiting time measured as ms. Allowed value range 0 – 3600000 ms. Unlike with the similar command CMD_PowerSTEP01_SET_WAIT, executing of this command can be interrupted by input signals IN0, IN1 or SET_ZERO.

This command is valid from 2d version of communication protocol.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Waiting time							Command code CMD_PowerSTEP01_SET_ WAIT_2 = 0x3C						Action	Reserve		
Bit value	Depend on Data value							1	1	1	1	0	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.59 Executing command CMD_PowerSTEP01_SCAN_MARK2_F

The executing command CMD_PowerSTEP01_SCAN_MARK2_F = 0x3D is intended for searching LABEL position in a forward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the LABEL position. The motor stops according the deceleration value, current position is set as “Mark” position.

Attention: the speed commands are always set as full steps per second.

This command is valid from 2d version of communication protocol.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Motion Speed							Command code CMD_PowerSTEP01_ SCAN_MARK2_F = 0x3D						Action	Reserve		
Bit value	Depend on Data value							1	1	1	1	0	1	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

6.60 Executing command CMD_PowerSTEP01_SCAN_MARK2_R

The executing command CMD_PowerSTEP01_SCAN_MARK2_R = 0x3E is intended for searching LABEL position in backward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the LABEL position. The motor stops according the deceleration value, current position is set as “Mark” position.

Attention: the speed commands are always set as full steps per second.

This command is valid from 2d version of communication protocol.

Bit mapping of the SMSD_CMD_Type structure:

	Byte[3]			Byte[2]			Byte[1]			Byte[0]							
Bit №	7	...	0	7	...	0	7..2	1	0	7	6	5	4	3	2	1	0
Designation	Data = Motion Speed							Command code CMD_PowerSTEP01_ SCAN_MARK2_F = 0x3E						Action	Reserve		
Bit value	Depend on Data value							1	1	1	1	1	0	0	0	0	0

As a response the Controller sends a data transmission packet with CMD_TYPE = CODE_CMD_RESPONSE, the DATA field of the packet contains COMMANDS_RETURN_DATA_Type: ERROR_OR_COMMAND = OK.

7. Structure SMSD_LAN_Config_Type

LAN parameters of the controller are kept in the structure 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;
```

Default LAN parameters:

```
{
   .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. Differences in Ethernet and USB data transmission

Data transmission packets, which are transferred via physical connection USB (virtual COM port), are the same packets as transferred via Ethernet connection, but in the beginning and end of the packet special markers are added and unique symbols are masked by pairs of symbols:

0xFA – marker of the beginning of the packet
0xFB – marker of the end of the packet

If the unique symbols 0xFA, 0xFB or 0xFE are present inside the packet, they should be replaced by the pair of symbols: 0xFE 0xXX. 0xXX is the unique symbol ^0x80.

The byte 0xFA inside the packet should be replaced by the pair 0xFE 0x7A.
The byte 0xFB inside the packet should be replaced by the pair 0xFE 0x7B.
The byte 0xFE inside the packet should be replaced by the pair 0xFE 0x7E.

9. Manufacturer information

Smart Motor Devices adheres to the line of continuous development and reserves the right to make changes and improvements in the design and software of the product without prior notice.

The information contained in this manual is subject to change at any time and without prior notice.

Last modified 04 June 2024