SMART MOTOR DEVICES

https://smd.ee

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

Data communications protocol

Modbus TCP/IP version

2024

1. Brief introduction

The controllers SMSD-4.2LAN and SMSD-8.0LAN are designed to control stepper motors and provide programming and control via USB or Ethernet. When operating over an Ethernet local network ("LA" indicator at the front panel), the controller creates a socket for connecting a control user program or device. Data is transmitted over a physical Ethernet line (TCP protocol).

This manual applies to the controllers with special firmware that provides control via the Modbus TCP/IP protocol. The communication protocol using USB remains unchanged with the exception of the structure of the LAN transmission parameters (see the description of the standard firmware protocol SMSD-4.2LAN and SMSD-8.0LAN).

2. Data transmission

Default Ethernet connection parameters:

- Node ID: 1
- MAC address: 0x01 0xF8 0xDC 0x3F 0x00 0x00
- IP address: 192.168.1.2
- Port: 502
- 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.

Data transmission via LAN is carried out in accordance with the standard Modbus TCP/IP protocol (https://modbus.org/). Commands for writing and reading registers are transmitted in accordance with the register table - Appendix A.

Transmission frame:

MBAP Header				PDU	
Transaction ID	Protocol ID (0 for Modbus)	Data length	Node ID Device address	Function code	Data
2 bytes	2 bytes	2 bytes	1 byte	1 byte	

Register types and function codes

Register type	Size	Function code		
		Read multiple registers	Write single register	Write multiple registers
Discretes Input	1 bit	0x02	-	-
Coils	1 bit	0x01	0x05	0x0F
Input Registers	16 bits (word)	0x04	-	-
Holding Registers	16 bits (word)	0x03	0x06	0x10

3. Authorization

Access to the controller data is protected by a 64-bit password with an authorization timeout of 1 second.

Before start working with the controller, it is needed to confirm a password. Authorization will be valid for the current connection session.

Authorization procedure:

1. Write the password value to the Holding Registers 0x2100 and 0x2102.
2. Write the value 0 to the Holding Register register 0x2104
3. Read the authorization result from the Discrete Inputs register 0x2200.

After successful authorization (register value 0x2200 = TRUE), the controller provides access to all control registers.

Address HEX	Register type	Size	Data type	Register name	Description
Authorization					
2100	HR	32	UINT_HEX	Password_LOW32	Low 32 bits of the password (default value 0x89ABCDEF)
2102	HR	32	UINT_HEX	Password_HIGH32	High 32 bits of the password (default value 0x01234567)
2104	HR	16	UINT_DEC	Password_CMD	Write value: = 0 - authorization attempt = 1 - change password
2200	DI	1	BOOL	Access	Displaying authorization status FALSE - no access TRUE - access is allowed

4. Control settings

4.1. Motor configuration

Motor configuration settings include operating and holding current, microstepping mode and control type (current or voltage). Configuration parameters can only be changed when the motor phases are de-energized. Before changing configuration parameters, ensure that the motor is in the de-energized (Hiz) state (Discrete inputs 0x1200).

Address HEX	Register type	Size	Data type	Register name	Description
Motor configuration					
110A	HR	16	UINT_DEC	CURRENT_OR_VOLTAGE	Control type
110B	HR	16	UINT_DEC	MOTOR_TYPE	Motor model for the voltage control mode
110C	HR	16	UINT_HEX	MICROSTEPPING	Microstepping mode
110D	HR	16	UINT_DEC	WORK_CURRENT	Operating current for the current control mode
110E	HR	16	UINT_DEC	STOP_CURRENT	Holding current

CURRENT_OR_VOLTAGE - Control type:

0 – voltage mode,
1 – current mode

MOTOR_TYPE – Motor model 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 – microstepping mode:

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 - 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%

4.2. Drive parameter settings

The drive operating parameters can be read or changed using the corresponding Modbus registers, or during a user program executing.

Address HEX	Register type	Size	Data type	Register name	Description
Control					
1100	HR	16	UINT_DEC	MIN_SPEED	Minimum motor speed
1101	HR	16	UINT_DEC	MAX_SPEED	Maximum motor speed
1102	HR	16	UINT_DEC	ACC	Acceleration
1103	HR	16	UINT_DEC	DEC	Deceleration
1104	HR	16	UINT_DEC	FS_SPEED	Full step speed
1105	HR	16	UINT_DEC	TARGET_SPEED	Target speed
1106	HR	32	INT_DEC	TARGET_POS	Target position
1108	HR	16	UINT_DEC	TARGET_INPUT	Input number
1109	HR	16	UINT_HEX	CMD	Command code
1400	DO	1	BOOL	CLR	Resetting error flags
1401	DO	1	BOOL	Reset Pos	Resetting the current position counter
1402	DO	1	BOOL	Reset powerSTEP01	Resetting the stepper motor control module
1403	DO	1	BOOL	Soft STOP	Smooth stop with specified deceleration and transition to holding mode
1404	DO	1	BOOL	Hard STOP	Abrupt stop and transition to holding mode
1405	DO	1	BOOL	Soft HiZ	Smooth stop with specified deceleration and de-energizing the motor
1406	DO	1	BOOL	Hard HiZ	Abrupt stop and de-energizing the motor

MIN_SPEED – setting the motor minimum speed. Allowable setting range from 0 to 950 steps/sec. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

MAX_SPEED – setting the motor maximum speed. Allowable setting range from 16 to 15600 steps/sec. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

ACC – stepper motor acceleration value. Allowable setting range from 15 to 59000 steps/sec^2.

DEC – stepper motor deceleration value. Allowable setting range from 15 to 59000 steps/sec^2.

FS_SPEED – setting the speed of transition to the full-step operating mode. Allowable setting range from 15 to 15600 steps/sec. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

TARGET_SPEED – set value of the stepper motor shaft rotation speed. Allowable setting range from 16 to 15600 steps/sec. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

TARGET_POS – setting the value of a displacement or target coordinate for positioning. Allowable setting range from -2 097 152 to + 2 097 151 steps (microsteps).

TARGET_INPUT – setting the input number for movement commands with input signal conditions.

CMD – command code. Writing a value to the register (see below) will lead to the start of execution of the command parameterized in registers 0x1100...0x1110.

Valid values for writing to the CMD register:

0x0e - RUN_F – continuous movement in the forward direction at a constant set speed (TARGET_SPEED).

0x0f - RUN_R – continuous movement in the backward direction at a constant set speed (TARGET_SPEED.

0x10 - MOVE_F – offset by a given value (TARGET_POS) in the forward direction. The motion speed is determined by preset minimum and maximum speed, acceleration and deceleration parameters. The motor must be stopped before sending this command.

0x11 - MOVE_R – offset by a given value (TARGET_POS) in the backward direction. The motion speed is determined by preset minimum and maximum speed, acceleration and deceleration parameters. The motor must be stopped before sending this command.

0x12 - GO_TO_F – moving to a target coordinate (TARGET_POS) in the forward direction. The motion speed is determined by preset minimum and maximum speed, acceleration and deceleration parameters.

0x13 - GO_TO_R – moving to a target coordinate (TARGET_POS) in the backward direction. The motion speed is determined by preset minimum and maximum speed, acceleration and deceleration parameters.

0x14 - GO_UNTIL_F – continuous movement in the forward direction at maximum speed until a signal arrives at a given input (TARGET_INPUT). After receiving the signal, the motor stops with the specified deceleration. When processing the command, the specified signal mask is taken into account.

0x15 - GO_UNTIL_R – continuous movement in the backward direction at maximum speed until a signal arrives at a given input (TARGET_INPUT). After receiving the signal, the motor stops with the specified deceleration. When processing the command, the specified signal mask is taken into account.

0x16 - SCAN_ZERO_F – search for the zero position in the forward direction with a given speed (TARGET_SPEED). The movement continues until a signal is received at the SET_ZERO input. When a signal is received, the motor stops and the current position is taken as zero.

0x17 - SCAN_ZERO_R – search for the zero position in the backward direction with a given speed (TARGET_SPEED). The movement continues until a signal is received at the SET_ZERO input. When a signal is received, the motor stops and the current position is taken as zero.

0x18 - SCAN_MARK_F – search for the mark position in the forward direction at a given speed (TARGET_SPEED). The movement continues until a signal arrives at the input IN1. When a signal is received, the motor stops and the current position is remembered as the mark position.

0x19 - SCAN_MARK_R – search for the mark position in the backward direction at a given speed (TARGET_SPEED). The movement continues until a signal arrives at the input IN1. When a signal is received, the motor stops and the current position is remembered as the mark position.

0x1a - GO_ZERO – movement to the zero position.

0x1b - GO_MARK – movement to the mark position.

0x1c - GO_TO – moving to a target coordinate (TARGET_POS) along the shortest path.

0x1f - SOFT_STOP – Smooth stop with specified deceleration (DEC) and transition to holding mode After stopping, the motor holds the position with the specified holding current (STOP_CURRENT).

0x20 - HARD_STOP – Abrupt stop of the stepper motor. After stopping, the motor holds the position with the specified holding current (STOP_CURRENT).

0x21 - SOFT_HI_Z – Smooth stop of the stepper motor with specified deceleration (DEC), then de-energizing the motor phases.

0x22 - HARD_HI_Z – Abrupt stop and de-energizing the motor phases.

0x2f - START_PROGRAM_MEM0 – start execution of the user program from the memory 0 of the controller.

0x30 - START_PROGRAM_MEM1 – start execution of the user program from the memory 1 of the controller.

0x31 - START_PROGRAM_MEM2 – start execution of the user program from the memory 2 of the controller.

0x32 - START_PROGRAM_MEM3 – start execution of the user program from the memory 3 of the controller.

0x33 - STOP_PROGRAM_MEM – stopping the execution of a user program.

0x3D - SCAN_MARK2_F – search for the mark position in the forward direction at a given speed (TARGET_SPEED). The movement continues until a signal arrives at the input IN1. When a signal is received, the motor stops with specified deceleration (DEC) and the current position is remembered as the mark position.

0x3E - SCAN_MARK2_R - – search for the mark position in the backward direction at a given speed (TARGET_SPEED). The movement continues until a signal arrives at the input IN1. When a signal is received, the motor stops with specified deceleration (DEC) and the current position is remembered as the mark position.

CLR – Coil – write TRUE to the register to clear all error flags

Reset Pos – Coil – write TRUE to the register to reset the current position counter. After the command is executed, the current position is taken as zero.

Reset powerSTEP01 – Coil – write TRUE to the register to perform a full hardware and software reset of the stepper motor control module, but not the controller as a whole.

Soft STOP – Coil – write TRUE to the register to perform a smooth stop of the stepper motor with a specified deceleration (DEC). After stopping, the motor holds the position with the specified holding current (STOP_CURRENT).

Hard STOP – Coil – write TRUE to the register to perform abrupt stop of the stepper motor. After stopping, the motor holds the position with the specified holding current (STOP_CURRENT).

Soft HiZ – Coil – write TRUE to the register to perform a smooth stop of the stepper motor with a specified deceleration (DEC). After the motor stops, the motor phases are de-energized.

Hard HiZ – Coil – write TRUE to the register to perform abrupt stop and are de-energizing of the stepper motor.

4.3. Monitoring of operation parameters

Operating parameters can be read from the corresponding registers using commands via the Modbus protocol.

Address HEX	Register type	Size	Data type	Register name	Description
Information about the motor current state					
1000	IR	16	UINT_DEC	SPEED	Current motor speed
1001	IR	32	INT_DEC	ABS_POS	Current motor position
1003	IR	16	UINT_HEX	EL_POS	Electrical rotor position
1004	IR	16	UINT_HEX	STATUS	Current state of the controller
1200	DI	1	BOOL	HiZ	Motor phases state
1201	DI	1	BOOL	STOP	Motor stop
1202	DI	1	BOOL	CONST_SPEED	Motor rotates with constant speed
1203	DI	1	BOOL	ACC	Motor acceleration
1204	DI	1	BOOL	DEC	Motor deceleration
1205	DI	1	BOOL	READY	Ready for the next task
1206	DI	1	BOOL	SW_F	Function SW
1207	DI	1	BOOL	SW_EVN	SW function event
1208	DI	1	BOOL	DIR	Direction of rotation
1209	DI	1	BOOL	CMD_ERROR	Command execution error
Information about user program executing					
3000	IR	16	UINT_DEC	MODE_N_PROGRAMS	Set number of the user program (for autonomous operation mode)
3200	DI	1	BOOL	PROGRAM_RUN	Program execution flag
3001	IR	16	UINT_DEC	N_PROGRAM	Currently running user program
3002	IR	16	UINT_DEC	N_COMMAND	Line number of the program that is currently running

SPEED – Current speed of the stepper motor

ABS_POS – Current position of the stepper motor

EL_POS – Information about the current electrical position of the rotor: bits 8.7 – current step, bits 6..0 – current microstep within the full step (measured as 1/128 of the full step value).

HiZ – Information about motor phases state (powered or de-energized).

STOP – Stepper motor stop flag

CONST_SPEED – Stepper motor constant speed motion flag

ACC – Stepper motor accelerating motion flag

DEC – Stepper motor decelerating motion flag

READY – Flag of readiness to perform the next task.

SW_F – SW function flag: 1- SW function ON, 0 – SW function OFF

SW_EVN – SW function event: 1- if the event has come, 0 – if the event hasn't come

DIR – Information about stepper motor rotation direction

CMD_ERROR – Command execution error flag: 1- command error, 0 – no errors

MODE_N_PROGRAMS – The number of the program that the user has set to run in autonomous operation mode (bF) via the front panel of the controller.

PROGRAM_RUN – Program execution flag

N_PROGRAM – Number of the currently running user program

N_COMMAND – Line number of the program that is currently running

4.4. Input/output signals

Address HEX	Register type	Size	Data type	Register name	Description
Inputs/outputs					
1005	IR	16	UINT_HEX	Inputs	State of inputs, state and setting of mask and waiting for input signals
110F	HR	16	UINT_HEX	Mask	
1110	HR	16	UINT_HEX	Wait	
1407	DO	1	BOOL	Relay	Relay output control

Inputs – Inputs state

Mask – Input signals mask. When executing commands with input conditions, the signal mask is taken into account. If the mask of the corresponding signal is 1, the command is executed.

Wait - Waiting for input signals.

Relay – coil – Relay output control

5. Reading and writing user program

The controller allows to create user programs and write them to non-volatile memory. The controller has 4 memory areas for storing programs, every area can hold up to 255 instructions. Reading and writing a program is possible in blocks - up to 32 instruction lines of program at a time. Each instruction takes 2 Modbus registers (4 bytes of memory) and has the structure corresponding to the main (standard) transfer protocol for SMSD-LAN controllers:

```
typedef struct
{
uint32_t RESERVE :4;
uint32_t COMMAND :6;
uint32_t DATA :22;
} SMSD_CMD_Type;
```

The first 4 bits are reserved and contain 0 when reading/writing program lines. The next 6 bits contain the command code in accordance with Appendix B. The last 22 bits contain the command data.

Example of reading/writing a command “maximum motor speed” with data value = 100 steps/sec:

The command for setting the maximum speed SET_MAX_SPEED in accordance with Appendix B has code 0x06, in binary form b1100. Command data 100 in binary form b1100100. When writing to the SMSD_CMD_Type structure, the value is b11001000001100000 = d102496 = 0x00019060. Thus, the command to set the maximum speed to 100 steps/sec when reading/writing a program looks like 0x00019060.

Bit layout of the structure SMSD_CMD_Type:

Byte 3	Byte 2	Byte 1		Byte 0	
Bits 0..7	Bits 0..7	Bits 2..7	Bits 0..1	Bits 4..7	Bits 0..3
Command data = 100 = b1100100			Command code = 0x06 = b110		0 (rezerved)
0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	1 1 0 0 1 0 0	0	0 1 1 0	0 0 0 0

Special Modbus registers are intended for reading/writing programs:

Address HEX	Register type	Size	Data type	Register name	Description
Programming					
3100	HR	32	UINT_HEX	CMD 00	Command buffer for writing/reading a program
3102	HR	32	UINT_HEX	CMD 01	
3104	HR	32	UINT_HEX	CMD 02	
…	HR	32	UINT_HEX	…	
313E	HR	32	UINT_HEX	CMD 31	
3180	HR	16	UINT_DEC	N_PROG	Program area number (from 0 to 3).
3181	HR	16	UINT_DEC	N_STR	Starting address of the instruction to read/write program
3182	HR	16	UINT_DEC	SECTOR_SIZE	Number of instructions that will be written to memory from the buffer or read from memory to the buffer
3183	HR	16	UINT_DEC	PROG_SIZE	Total program size
3184	HR	16	UINT_DEC	MEM_CMD	Writing a value to a register will run the procedure: 0 – read instructions to the buffer 1 – write instructions from the buffer 2 – erase program (including PROG_SIZE) 3 – reading program size to the register PROG_SIZE 4 – recording program size from the register PROG_SIZE
3400	DO	1	BOOL	B_COMPLETE	Flag of completing the procedure (MEM_CMD). Resets manually.

Registers 0x3100...0x3182 are a buffer for storing a packet of instructions for reading or writing. Commands for working with the controller memory are written to the MEM_CMD register. The B_COMPLETE register is the completion flag for the command passed to MEM_CMD. When the MEM_CMD command completes, the B_COMPLETE flag arises. This flag must be cleared manually.

Operations procedure for writing a program of N instructions to the controller memory:

1. Write the number of the memory area for recording the program - N_PROG register (values 0 to 3).
2. Reset the flag B_COMPLETE
3. Execute the memory erase command (write MEM_CMD = 2)
4. Wait until the B_COMPLETE command completion flag is set
5. Reset the flag B_COMPLETE
6. Divide N instructions of the program into n groups, each group with X instructions (x <= 32, since write buffer size = 32 lines). For each group of X instructions, do the following:

6.1. Set the program line number to start writing (N_STR register). The first group of instructions is written to the line 0 (if the program is supposed to be written to the beginning of the memory). For each subsequent group this value should be increases by X.

6.2. Set the sector size for writing (SECTOR_SIZE register) – the number of instructions for writing at a time (SECTOR_SIZE = X).

6.3. Write X instructions to the transfer buffer (registers CMD_00...CMD_31) - one program line (one instruction) takes 2 data words (two Modbus registers).

6.4. Execute the command to write the instructions from the buffer to the controller memory (MEM_CMD = 1).

6.5. Wait until the B_COMPLETE command completion flag is set

6.6. Reset the flag B_COMPLETE

7. Write the value of the full program size (instructions number) to the register PROG_SIZE = N
8. Execute the write program size command (MEM_CMD = 4)
9. Wait until the B_COMPLETE command completion flag is set
10. Reset the flag B_COMPLETE

Operations procedure for reading a program from the controller memory:

1. Write the number of the memory area for reading the program - N_PROG register
2. Reset the flag B_COMPLETE
3. Execute the command to read the full size of the program (MEM_CMD = 3)
4. Wait until the B_COMPLETE command completion flag is set
5. Reset the flag B_COMPLETE
6. Read the value N - the full size of the program - from the PROG_SIZE register
7. Divide N instructions of the program into n groups, each group with X instructions (x <= 32, since read buffer size = 32 lines). For each group of X instructions, do the following:

   7.1. Set the program line number to start reading (N_STR register). The first group of instructions is read from the line 0 (if the program is supposed to be read from the beginning of the memory). For each subsequent group this value should be increases by X.

   7.2. Set the sector size for reading (SECTOR_SIZE register) – the number of instructions for reading at a time (SECTOR_SIZE = X).

   7.3. Execute the command to read instructions from the controller memory to the buffer (MEM_CMD = 0)

   7.4. Wait until the B_COMPLETE command completion flag is set

   7.5. Reset the flag B_COMPLETE

   7.6. Read X program lines from the buffer (registers CMD_00...CMD_31) - one program line (one instruction) takes 2 data words (two Modbus registers).

6. Changes in the transfer protocol when connecting via USB

When connected via USB, the standard version of the communication protocol is used, with the exception of reading and setting LAN parameters.

The field unitID is added to 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;
   uint8_t unitID;
} SMSD_LAN_Config_Type;
```

Default settings:

```
{
    .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 =  502,
    .dhcp =  1
    .unitID = 1
};
```

Appendix A. Register table

Register table for the stepper motor controllers SMSD-4.2LAN and SMSD-8.0LAN – version Modbus TCP/IP:

Address HEX	Register type	Size	Data type	Register name	Description
Authorization					
2100	HR	32	UINT_HEX	Password_LOW32	Low 32 bits of the password (default value 0x89ABCDEF)
2102	HR	32	UINT_HEX	Password_HIGH32	High 32 bits of the password (default value 0x01234567)
2104	HR	16	UINT_DEC	Password_CMD	Write value: = 0 - authorization attempt = 1 - change password
2200	DI	1	BOOL	Access	Displaying authorization status FALSE - no access TRUE - access is allowed
Motor configuration					
110A	HR	16	UINT_DEC	CURRENT_OR_VOLTAGE	Control type
110B	HR	16	UINT_DEC	MOTOR_TYPE	Motor model for the voltage control mode
110C	HR	16	UINT_HEX	MICROSTEPPING	Microstepping mode
110D	HR	16	UINT_DEC	WORK_CURRENT	Operating current for the current control mode
110E	HR	16	UINT_DEC	STOP_CURRENT	Holding current
Motion control					
1000	IR	16	UINT_DEC	SPEED	Minimum motor speed
1001	IR	32	INT_DEC	ABS_POS	Maximum motor speed
1003	IR	16	UINT_HEX	EL_POS	Acceleration
1100	HR	16	UINT_DEC	MIN_SPEED	Deceleration
1101	HR	16	UINT_DEC	MAX_SPEED	Full step speed
1102	HR	16	UINT_DEC	ACC	Target speed
1103	HR	16	UINT_DEC	DEC	Target position
1104	HR	16	UINT_DEC	FS_SPEED	Input number
1105	HR	16	UINT_DEC	TARGET_SPEED	Command code
1106	HR	32	INT_DEC	TARGET_POS	Minimum motor speed
1108	HR	16	UINT_DEC	TARGET_INPUT	Maximum motor speed
1109	HR	16	UINT_HEX	CMD	Acceleration
1004	IR	16	UINT_HEX	STATUS	Current state of the controller
1200	DI	1	BOOL	HiZ	Motor phases state
1201	DI	1	BOOL	STOP	Motor stop
1202	DI	1	BOOL	CONST_SPEED	Motor rotates with constant speed
1203	DI	1	BOOL	ACC	Motor acceleration
1204	DI	1	BOOL	DEC	Motor deceleration
1205	DI	1	BOOL	READY	Ready for the next task
1206	DI	1	BOOL	SW_F	Function SW
1207	DI	1	BOOL	SW_EVN	SW function event
1208	DI	1	BOOL	DIR	Direction of rotation
1209	DI	1	BOOL	CMD_ERROR	Command execution error
1400	DO	1	BOOL	CLR	Resetting error flags
1401	DO	1	BOOL	Reset Pos	Resetting the current position counter
1402	DO	1	BOOL	Reset powerSTEP01	Resetting the stepper motor control module

Address HEX	Register type	Size	Data type	Register name	Description
1403	DO	1	BOOL	Soft STOP	Smooth stop with specified deceleration and transition to holding mode
1404	DO	1	BOOL	Hard STOP	Abrupt stop and transition to holding mode
1405	DO	1	BOOL	Soft HiZ	Smooth stop with specified deceleration and de-energizing the motor
1406	DO	1	BOOL	Hard HiZ	Abrupt stop and de-energizing the motor
Inputs/outputs					
1005	IR	16	UINT_HEX	Inputs	State of inputs, state and setting of mask and waiting for input signals
110F	HR	16	UINT_HEX	Mask	
1110	HR	16	UINT_HEX	Wait	
1407	DO	1	BOOL	Relay	Relay output control
Information about user program executing					
3000	IR	16	UINT_DEC	MODE_N_PROGRAMS	Set number of the user program (for autonomous operation mode)
3200	DI	1	BOOL	PROGRAM_RUN	Program execution flag
3001	IR	16	UINT_DEC	N_PROGRAM	Currently running user program
3002	IR	16	UINT_DEC	N_COMMAND	Line number of the program that is currently running
Programming					
3100	HR	32	UINT_HEX	CMD 00	Command buffer for writing/reading a program
3102	HR	32	UINT_HEX	CMD 01	
3104	HR	32	UINT_HEX	CMD 02	
3106	HR	32	UINT_HEX	CMD 03	
3108	HR	32	UINT_HEX	CMD 04	
310A	HR	32	UINT_HEX	CMD 05	
310C	HR	32	UINT_HEX	CMD 06	
310E	HR	32	UINT_HEX	CMD 07	
3110	HR	32	UINT_HEX	CMD 08	
3112	HR	32	UINT_HEX	CMD 09	
3114	HR	32	UINT_HEX	CMD 10	
3116	HR	32	UINT_HEX	CMD 11	
3118	HR	32	UINT_HEX	CMD 12	
311A	HR	32	UINT_HEX	CMD 13	
311C	HR	32	UINT_HEX	CMD 14	
311E	HR	32	UINT_HEX	CMD 15	
3120	HR	32	UINT_HEX	CMD 16	
3122	HR	32	UINT_HEX	CMD 17	
3124	HR	32	UINT_HEX	CMD 18	
3126	HR	32	UINT_HEX	CMD 19	
3128	HR	32	UINT_HEX	CMD 20	
312A	HR	32	UINT_HEX	CMD 21	
312C	HR	32	UINT_HEX	CMD 22	
312E	HR	32	UINT_HEX	CMD 23	
3130	HR	32	UINT_HEX	CMD 24	
3132	HR	32	UINT_HEX	CMD 25	
3134	HR	32	UINT_HEX	CMD 26	
3136	HR	32	UINT_HEX	CMD 27	
3138	HR	32	UINT_HEX	CMD 28	
313A	HR	32	UINT_HEX	CMD 29	
313C	HR	32	UINT_HEX	CMD 30	
313E	HR	32	UINT_HEX	CMD 31	

Address HEX	Register type	Size	Data type	Register name	Description
3180	HR	16	UINT_DEC	N_PROG	Program area number (from 0 to 3).
3181	HR	16	UINT_DEC	N_STR	Starting address of the instruction to read/write program
3182	HR	16	UINT_DEC	SECTOR_SIZE	Number of instructions that will be written to memory from the buffer or read from memory to the buffer
3183	HR	16	UINT_DEC	PROG_SIZE	Total program size
3184	HR	16	UINT_DEC	MEM_CMD	Writing a value to a register will run the procedure: 0 – read instructions to the buffer 1 – write instructions from the buffer 2 – erase program (including PROG_SIZE) 3 – reading program size to the register PROG_SIZE 4 – recording program size from the register PROG_SIZE
3400	DO	1	BOOL	B_COMPLETE	Flag of completing the procedure (MEM_CMD). Resets manually.
LAN parameters					
7000	HR	16	UINT_DEC	ID	Device ID - identifier in the Modbus TCP network
7001	HR	16	UINT_HEX	MAC 0	MAC address Default value: 0x00 0xF8 0xDC 0x3F 0x00 0x00
7002	HR	16	UINT_HEX	MAC 1	
7003	HR	16	UINT_HEX	MAC 2	
7004	HR	16	UINT_HEX	MAC 3	
7005	HR	16	UINT_HEX	MAC 4	
7006	HR	16	UINT_HEX	MAC 5	
7007	HR	16	UINT_DEC	IP 0	IP address Default value: 192.168.1.2
7008	HR	16	UINT_DEC	IP 1	
7009	HR	16	UINT_DEC	IP 2	
700A	HR	16	UINT_DEC	IP 3	
700B	HR	16	UINT_DEC	Subnet Mask 0	IP sub-network mask Default value: 255.255.0.0
700C	HR	16	UINT_DEC	Subnet Mask 1	
700D	HR	16	UINT_DEC	Subnet Mask 2	
700E	HR	16	UINT_DEC	Subnet Mask 3	
700F	HR	16	UINT_DEC	Gateway IP 0	Gateway Default value: 192.168.1.1
7010	HR	16	UINT_DEC	Gateway IP 1	
7011	HR	16	UINT_DEC	Gateway IP 2	
7012	HR	16	UINT_DEC	Gateway IP 3	
7013	HR	16	UINT_DEC	DNS server IP Address 0	DNS Default value: 0.0.0.0
7014	HR	16	UINT_DEC	DNS server IP Address 1	
7015	HR	16	UINT_DEC	DNS server IP Address 2	
7016	HR	16	UINT_DEC	DNS server IP Address 3	
7017	HR	16	UINT_DEC	Port	Port Default value: 502
7018	HR	16	UINT_DEC	DHCP	Dynamic address setting 1 – Static (default) 2 – DHCP
7400	DO	1	BOOL	Apply Network Configuration	Apply and save settings
Version and identification registers					
8001	IR	16	UINT_DEC	HW_MAJOR	Device ID: 6 - SMSD-4.2ModbusTCP

Address HEX	Register type	Size	Data type	Register name	Description
					7 - SMSD-8.0ModbusTCP
8002	IR	16	UINT_DEC	HW_VER	Hardware version: 1
8003	IR	16	UINT_DEC	FW_MAJOR	Firmware version: 1.3
8004	IR	16	UINT_DEC	FW_MINOR	
8007	IR	16	UINT_DEC	RM_MAJOR	Register map version: 4.3
8008	IR	16	UINT_DEC	RM_MINOR	

Appendix B. Controller executing instructions

Executing instruction CMD_PowerSTEP01_SET_MODE

Executing instruction CMD_PowerSTEP01_SET_MODE = 0x03 is intended for setting motor and control parameters. The motor windings must be de-energized at the moment the command is executed.

Bit mapping of the Data field of the SMSD_CMD_Type structure:

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

The values at the fields CURRENT_OR_VOLTAGE, MOTOR_TYPE, MICROSTEPPING, WORK_CURRENT, STOP_CURRENT correspond to the similar registers in the section «4.1. Motor configuration».

Executing instruction CMD_PowerSTEP01_SET_MIN_SPEED

Executing instruction 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. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_SET_MAX_SPEED

Executing instruction 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. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_SET_ACC

Executing instruction 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.

Executing instruction CMD_PowerSTEP01_SET_DEC

Executing instruction CMD_PowerSTEP01_SET_DEC = 0x08 is intended for setting the motor deceleration to getting the motor maximum speed. The DATA field should contain the acceleration value in range 15 – 59000 steps/sec^2

Executing instruction CMD_PowerSTEP01_SET_FS_SPEED

Executing instruction 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. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_SET_MASK_EVENT

Executing instruction 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 Data field of the SMSD_CMD_Type structure:

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.

Executing instruction CMD_PowerSTEP01_RUN_F

Executing instruction 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. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_RUN_R

Executing instruction CMD_PowerSTEP01_RUN_F = 0x0E 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. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_MOVE_F

Executing instruction 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). Important: the speed commands are always set as full steps per second. The motion commands are always set as microstepping measured displacements.

Executing instruction CMD_PowerSTEP01_MOVE_R

Executing instruction CMD_PowerSTEP01_MOVE_F = 0x10 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). Important: the speed commands are always set as full steps per second. The motion commands are always set as microstepping measured displacements.

Executing instruction CMD_PowerSTEP01_GO_TO_F

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

Executing instruction CMD_PowerSTEP01_GO_TO_R

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

Executing instruction CMD_PowerSTEP01_GO_UNTIL_F

Executing instruction 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 instruction CMD_PowerSTEP01_SET_MASK_EVENT.

Executing instruction CMD_PowerSTEP01_GO_UNTIL_R

Executing instruction CMD_PowerSTEP01_GO_UNTIL_R = 0x15 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 instruction CMD_PowerSTEP01_SET_MASK_EVENT.

Executing instruction CMD_PowerSTEP01_SCAN_ZERO_F

Executing instruction 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. Important: the speed commands are always set as full steps per second.

Executing instruction CMD_PowerSTEP01_SCAN_ZERO_R

Executing instruction CMD_PowerSTEP01_SCAN_ZERO_R = 0x17 is intended for searching zero position in backward direction. The movement continues until signal to SET_ZERO input received. The DATA field determines the motion speed during searching the zero position. Important: the speed commands are always set as full steps per second.

Executing instruction CMD_PowerSTEP01_SCAN_MARK_F

Executing instruction CMD_PowerSTEP01_SCAN_MARK_F = 0x18 is intended for searching MARK position in a forward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the MARK position. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_SCAN_MARK_R

Executing instruction CMD_PowerSTEP01_SCAN_MARK_R = 0x19 is intended for searching MARK position in backward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the MARK position. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_GO_ZERO

Executing instruction CMD_PowerSTEP01_GO_ZERO = 0x1A is intended for movement to the ZERO position. Data field is ignored.

Executing instruction CMD_PowerSTEP01_GO_LABEL

Executing instruction CMD_PowerSTEP01_GO_LABEL = 0x1B is intended for movement to the MARK position. Data field is ignored.

Executing instruction CMD_PowerSTEP01_GO_TO

Executing instruction CMD_PowerSTEP01_GO_TO = 0x1C 0x1C is intended for the shortest movement to the specified position. Important: the speed commands are always set as full steps per second. The motion commands are always set as microstepping measured displacements.

Executing instruction CMD_PowerSTEP01_RESET_POS

Executing instruction 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). Data field is ignored.

Executing instruction CMD_PowerSTEP01_RESET_POWERSTEP01

Executing instruction CMD_PowerSTEP01_RESET_POWERSTEP01 = 0x1E is intended for hardware and software reset of the stepper motor control module, but not of the whole Controller.. Data field is ignored.

Executing instruction CMD_PowerSTEP01_SOFT_STOP

Executing instruction CMD_PowerSTEP01_SOFT_STOP = 0x1F is intended for smooth decelerating of the stepper motor and stop. After that the motor holds the current position (with preset holding current).. Data field is ignored.

Executing instruction CMD_PowerSTEP01_HARD_STOP

Executing instruction CMD_PowerSTEP01_HARD_STOP = 0x20 is intended for sudden stop of the stepper motor and holding the current position (with preset holding current). Data field is ignored.

Executing instruction CMD_PowerSTEP01_SOFT_HI_Z

Executing instruction CMD_PowerSTEP01_SOFT_HI_Z = 0x21 is intended for smooth decelerating of the stepper motor and stop. After that the motor phases are deenergized. Data field is ignored.

Executing instruction CMD_PowerSTEP01_HARD_HI_Z

Executing instruction CMD_PowerSTEP01_HARD_HI_Z = 0x22 is intended for sudden stop and deenergizing the stepper motor. Data field is ignored.

Executing instruction CMD_PowerSTEP01_SET_WAIT

Executing instruction 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.

Executing instruction CMD_PowerSTEP01_SET_RELE

Executing instruction CMD_PowerSTEP01_SET_RELE = 0x24 is intended to turn on the controller relay. Data field is ignored.

Executing instruction CMD_PowerSTEP01_CLR_RELE

Executing instruction CMD_PowerSTEP01_CLR_RELE = 0x25 is intended to turn off the controller relay. Data field is ignored.

Executing instruction CMD_PowerSTEP01_WAIT_IN0

Executing instruction CMD_PowerSTEP01_WAIT_IN0 = 0x27 is used to wait until receiving a signal to the input IN0. Data field is ignored.

Executing instruction CMD_PowerSTEP01_WAIT_IN1

Executing instruction CMD_PowerSTEP01_WAIT_IN1 = 0x28 is used to wait until receiving a signal to the input IN1. Data field is ignored.

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM

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

Bit mapping of the Data field of the SMSD_CMD_Type structure:

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		Instruction number							

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN0

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN0 = 0x2A is intended for conditional branching – to jump to a specified instruction 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 instruction sequence number: bits 0..7 of the DATA field contain the instruction number, bits 8,9 of the DATA field contain the program number.

Bit mapping of the Data field of the SMSD_CMD_Type structure:

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		Instruction number							

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN1

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 instruction sequence number: bits 0..7 of the DATA field contain the instruction number, bits 8,9 of the DATA field contain the program number.

Bit mapping of the Data field of the SMSD_CMD_Type structure:

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		Instruction number							

Executing instruction CMD_PowerSTEP01_LOOP_PROGRAM

Executing instruction CMD_PowerSTEP01_LOOP_PROGRAM = 0x2C is used to create a loop – the controller repeats specified times specified number of instructions (start from the first instruction after this instruction. The DATA field contains the information about instruction s number and cycles number: bits 0..9 of the DATA field contain the instructions number, bits 10..19 of the DATA field contain the cycles number.

Bit mapping of the Data field of the SMSD_CMD_Type structure:

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										Number of instructions in the loop									

Executing instruction CMD_PowerSTEP01_CALL_PROGRAM

Executing instruction CMD_PowerSTEP01_CALL_PROGRAM = 0x2D is intended for calling a subprogram. The DATA field contains the information about a program memory number and a instruction sequence number, which starts a subprogram: bits 0..7 of the DATA field contain the instruction 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 instruction - 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 Data field of the SMSD_CMD_Type structure:

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		Instruction number							

Executing instruction CMD_PowerSTEP01_RETURN_PROGRAM

Executing instruction 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 instruction CMD_PowerSTEP01_CALL_PROGRAM was not called, the executing of CMD_PowerSTEP01_RETURN_PROGRAM will call an error.

Executing instruction CMD_PowerSTEP01_START_PROGRAM_MEM0

Executing instruction CMD_PowerSTEP01_START_PROGRAM_MEM0 = 0x2F is used to start program executing from the controller memory area Mem0. Data field is ignored.

instructions CMD_PowerSTEP01_START_PROGRAM_MEM1 = 0x30, CMD_PowerSTEP01_START_PROGRAM_MEM2 = 0x31, CMD_PowerSTEP01_START_PROGRAM_MEM3 = 0x32 are used to start program executing from the controller memory areas Mem1, Mem2, Mem3 accordingly.

Executing instruction CMD_PowerSTEP01_STOP_PROGRAM_MEM

Executing instruction CMD_PowerSTEP01_STOP_PROGRAM_MEM = 0x33 is used to stop executing a program. Data field is ignored.

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM_IF_ZERO

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM_IF_ZERO = 0x39 is intended for conditional branching – to jump to a specified instruction 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 instruction sequence number: bits 0..7 of the DATA field contain the instruction number, bits 8,9 of the DATA field contain the program number.

Bit mapping of the Data field of the SMSD_CMD_Type structure:

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		Instruction number							

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN_ZERO

Executing instruction CMD_PowerSTEP01_GOTO_PROGRAM_IF_IN_ZERO = 0x3A is intended for conditional branching – to jump to a specified instruction 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 instruction sequence number: bits 0..7 of the DATA field contain the instruction number, bits 8,9 of the DATA field contain the program number.

Bit mapping of the Data field of the SMSD_CMD_Type structure:

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		Instruction number							

Executing instruction CMD_PowerSTEP01_WAIT_CONTINUE

Executing instruction CMD_PowerSTEP01_WAIT_CONTINUE = 0x3B is intended for waiting of synchronization signal at the input CONTINUE, which is used for synchronization of executing programs in different controllers. Data field is ignored.

Executing instruction CMD_PowerSTEP01_SET_WAIT_2

Executing instruction 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 instruction CMD_PowerSTEP01_SET_WAIT, executing of this instruction can be interrupted by input signals IN0, IN1 or SET_ZERO..

Executing instruction CMD_PowerSTEP01_SCAN_MARK2_F

Executing instruction CMD_PowerSTEP01_SCAN_MARK2_F = 0x3D is intended for searching MARK position in a forward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the MARK position. The motor stops according to the deceleration value, current position is set as MARK position. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Executing instruction CMD_PowerSTEP01_SCAN_MARK2_R

Executing instruction CMD_PowerSTEP01_SCAN_MARK2_R = 0x3E is intended for searching MARK position in a backward direction. The movement continues until signal to IN1 input received. The DATA field determines the motion speed during searching the MARK position. The motor stops according to the deceleration value, current position is set as MARK position. Important: all speed values are specified as full steps per second, regardless of the set microstepping mode.

Last modified 31 May 2024