SMART MOTOR DEVICES

https://www.smd.ee

DC BRUSH MOTOR CONTROLLER
BMSD-20Modbus / BMSD-40Modbus

User Manual
BMSD.Modbus.001
2025

1. Product designation

Brush motor controllers BMSD-20Modbus and BMSD-40Modbus are electronic devices designed to operate and control DC brush motors.

2. Functions and possibilities

The controllers are designed to control speed, acceleration, deceleration, and direction of rotation of the motor. The units also provide positioning based on signals from an encoder in case of using a motor with Hall sensors.

The motor is controlled either by external signals or by commands transmitted via RS-485 by the Modbus protocol. The controller can also operate according to an algorithm previously recorded into its memory by a user.

Control via RS-485 Modbus

The controller can be remotely controlled through the physical RS-485 communication line using the industrial Modbus protocol:

- The setting is done by writing or reading the corresponding parameters to/from the controller registers.
- Supported protocols are RTU and ASCII, rates are 600,1200, 2400, 4800, 9600, 14400, 19200,38400, 57600, 115200,128000 baud.
- Movements to a target preset position or one of the four predefined reference points are implemented.
- The controller can operate autonomously under the control of a user program (up to 1024 commands), which is pre-recorded in non-volatile memory. Conditional and unconditional jumps (relative and absolute), subroutine calls, loops, and timers are supported.
- Programmable inputs IN1 and IN2 can be used as signals START/STOP, REVERSE, or for other purposes at the user's discretion.
- The controller can perform positioning in the range from - 2147483647 to + 2147483648 pulses if it uses a motor with Hall sensors.
- The controller has RT contacts on the front panel for a terminal resistor connection.

Control using external signals

To control the motor by external signals, the following are provided:

- Contacts for connecting the external signals IN1 and IN2, the purpose and processing of which are determined by the user. The inputs can also be used as START/STOP (motion start/stop) and DIR (direction) signals.
- HARD STOP contact for connecting an alarm signal-controlled motor braking in case of the emergency circuit breaking.

The controller provides a motor overcurrent protection function. The maximum allowed current in the motor phase is set by the user.

3. Technical characteristics

Model	BMSD-20Modbus	BMSD-40Modbus
Power supply voltage, VDC	12 - 24	
Power supply protection, VDC	8 - 30	
Rated current in motor phase, A	<20	<40
Hardware short-circuit protection (operation time 15 µs), A	30	100
Phase current limitation, A	1…20	2…40
Maximum voltage in the motor phase, V	0.98 x U_{sup}	
Minimum non-zero in the motor phase, V	0.01 x U_{sup}	
Parameters of external signals IN1 and IN2:		
Maximum resistance of closed contacts, kOhm	4.7	
Maximum input current, mA	0.5	
Dimensions (no more), mm	116x100x23	
Communication interface	RS-485, Modbus – ASCII or RTU	

The overall and connecting dimensions of the controller are shown in Fig. 1.

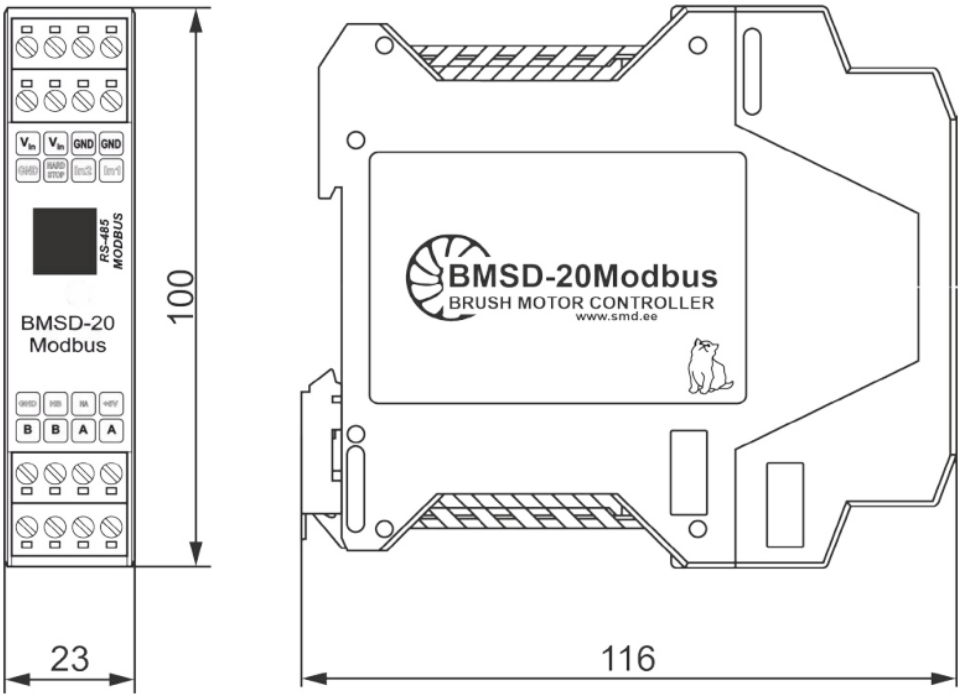

Fig.1. The overall and connecting dimensions of the controller BLSD-20Modbus

The connection scheme is shown in Fig. 2

Environmental Conditions:

Ambient Temperature: 0…+50°C

Humidity: 90% RH or less upon condition +25°C

Condensation and freezing: none

4. Construction and control elements

The controller is designed as a circuit plate with electronic elements, covered with a case with a DIN rail mount. On top of the case, there are graphical symbols for the controls and pin assignments. Besides electronic components, there are indicating and control elements, connection terminals, and connectors on the board:

- screw terminals for connecting power supply, brushless motor windings, encoder lines, and control circuit;
- terminals IN1 and IN2 for connecting control input signals;
- terminal for connecting emergency stop signal contacts;
- LED indicator of device operation;
- RJ11 (6P6C) connector for connecting RS-485 lines;
- contacts for connecting the internal terminal resistor RT;
- built-in braking circuit for absorbing the energy generated by the motor (during coasting, forced rotation).

An external signal input "HARD STOP" is provided for emergency motor stop. External inputs IN1 and IN2 can be used to start and stop the motor using external signals, as well as to control the direction of rotation of the motor.

All parameters of the motor operation and motion control can be carried out by software and by commands transmitted via RS-485 via the Modbus protocol.

The layout and purpose of the terminals are shown in Fig. 2

1. Power supply GND
2. Power supply GND
3. Power supply 12 – 24 VDC
4. Power supply 12 – 24 VDC
5. Signal “IN1” (clean contact)
6. Signal “IN2” (clean contact)
7. Emergency stop signal “HARD STOP”
8. Signal GND
9. +5V output for the supply of Hall sensors
10. Hall sensor data line A
11. Hall sensor data line B
12. GND of Hall sensors
13. Motor wire A
14. Motor wire A
15. Motor wire B
16. Motor wire B

RS-485 Modbus - RJ11 (6P6C) connector for connecting RS-485 data lines

RT - contacts for connecting the terminal resistance

Fig. 2. Layout and purpose of terminals and control elements

5. Assembly and connection

Please, learn this manual carefully before connection and assembly.

Please, wire just when the power is off. Do not attempt to change wiring while the power is ON.

Please provide a reliable contact at the connection terminals. During wiring, please observe the polarity and wire management. Reverse polarity and overvoltage will damage the controller.

IMPORTANT: Due to the high currents, it is recommended to locate the power supply in close proximity to the unit and use wires with a cross-section of 3 mm2 (AWG-12). The power supply must provide 20% more current than the maximum possible consumed during operation.

Recommended length of supply wires:

- No more than 100 cm at currents up to 10 A.
- No more than 50 cm at currents from 10 to 20A.
- No more than 25 cm at currents from 20 to 40A.

At a maximum current of up to 20 A, it is allowed to use both supply and phase terminals on one line. For a maximum current of more than 20 A, both lines of both supply and phase terminals must be used.

Follow the next instruction during connection:

1. Connect a motor to the controller according to Fig. 2. The Motor must be connected to the terminals A and B (13 – 16) of the controller. Hall sensor signals must be connected to the terminals HA and HB (10 – 11). GND of HALL sensors must be connected to the terminal GND (12), the supply of HALL sensor signals must be connected to the terminal +5V (9).
2. Connect external control elements according to the connection diagram schemes in Fig. 3:

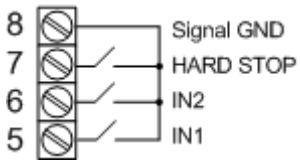

Fig. 3. External control elements connection

Type of external signals «IN1», «IN2», «HARD STOP» - clean contact.

3. Connect the lines of the RS-485 interface to the RJ11 connector according to Fig. 4.

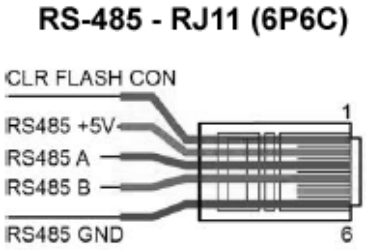

Fig.4. Pin assignment of the RJ11 connector - RS-485 Modbus

4. If necessary, place a jumper on the RT pins to connect the internal terminal resistance.
5. Connect the device to the power supply unit, observing the polarity. The power source unit must be selected with a margin (to prevent voltage drops). The thickness of the connecting wires must correspond to the current consumption of the motor. Connect the "+" of the power supply to the terminals 3 and 4, connect the "-" of the power supply to the terminals 1 and 2 of the controller. The disassembly procedure is in reverse order.

6. Operation

1. Make sure the power supply is turned off. Please, wire just when the power is off.
2. Connect the motor and power supply to the controller according to section 5.
3. Select the control method: control by commands via Modbus or external signals (register MODE_DEVICE - see section 6.5, Fig. 6 and Fig. 7).
4. Connect the control signals "IN1", "IN2", and the emergency stop signal "HARD STOP" according to section 5. The signal "HARD STOP" is used for an emergency stop of the motor. Operation is permitted with closed contact.
5. Connect the lines of the RS-485 interface according to section 5.

6. Turn on the power supply. The device is ready for configuration via Modbus protocol.
7. Set the necessary operation parameters by commands via Modbus protocol - motor current limitation, rotation direction, setting of operation of external inputs IN1 and IN2, and control method.
8. To control the drive using Modbus protocol, send commands through the RS-485 interface. The registers table of the controller, their purpose, and possible motor control commands are given below.
9. To control the drive with external signals to start and stop the motor and to control the direction of rotation, use signals IN1 and IN2. Speed regulation is performed by commands via the Modbus protocol.

MODBUS control

For data transmission via the RS-485 interface, the standard Modbus communication protocol (ASCII or RTU) is used.

The control unit has the following factory settings:

- ID = 1
- Rate: 115200 baud
- parity check: even
- Data bits: 8
- Stop bit: 1
- MODBUS RTU

6.1. Input control registers

Address	Type	Name	Size	Description
1000h	Discrete Input	IN1_bit	1-bit	State of the input signal IN1
1001h	Discrete Input	IN2_bit	1-bit	State of the input signal IN2
1002h	Discrete Input	IN_HARD_STOP_bit	1-bit	State of the input signal HARD_STOP
5007h	Holding Register	MODE_EXT_IN	16-bit	Configuration of the external inputs IN, IN2.
5013h	Holding Register	PRESSED_INPUTS_EXTERN	16-bit	Minimum positive pulse time at digital inputs IN1, IN2 (ms)
5014h	Holding Register	WAITED_INPUTS_EXTERN	16-bit	Minimum time for the negative part of the pulse at digital inputs IN1, IN2 (ms)

Input state registers 1000h..1002h are read-only.

1000h – **IN1_bit** – represents the state of the physical input IN1.

1001h – **IN2_bit** – represents the state of the physical input IN2.

1002h – **IN_HARD_STOP_bit** – represents the state of the physical input HARD_STOP.

Possible register values 1000h..1002h:

- 1 – input shorted to output GND
- 0 – input open with output GND

5007h - **MODE_EXT_IN** – the register value determines the purpose and method of signals IN1 and IN2 processing (see description in section 6.5).

5013h – **PRESSED_INPUTS_EXTERN** and 5014h – **WAITED_INPUTS_EXTERN** – the minimum time (set in ms) of the positive and negative parts of the pulse at the digital inputs IN1, IN2 - registers are used to suppress contact bounce (see Fig. 5).

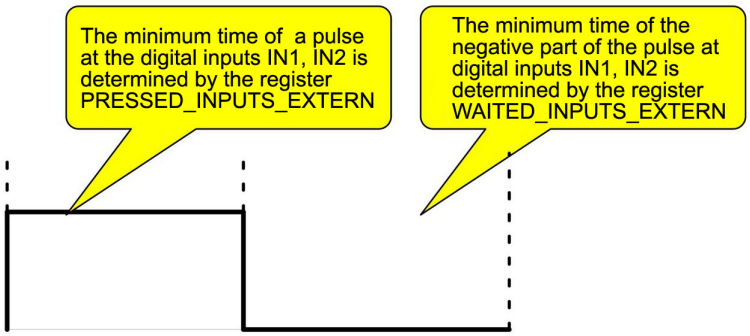

Fig. 5. Contact bounce suppression

6.2. Motor control registers

Address	Type	Name	Size	Description
2000h	Coils	START_bit	1-bit	Start motor rotation with a set acceleration
2001h	Coils	STOP_bit	1-bit	Stop motor rotation with a set deceleration
2002h	Coils	HARD_STOP_bit	1-bit	Abrupt emergency stop of motor rotation
2003h	Coils	CLR_POSITION_bit	1-bit	Resetting to zero the current position register POSITION_VALUE.

These control registers are available for both reading and writing. When writing to the corresponding register, the value 1 activates one or another function, after which the register is reset to 0.

6.3. State registers

Address	Type	Name	Size	Description
3000h	Input Register	STATUS	16-bit	The current state of the motor control.
3001h	Input Register	CURRENT_VALID	16-bit	The current value of the current consumed by the motor.
3002h	Input Register	SPEED_VALID	16-bit	The instantaneous value of the rotation speed.
3003h	Input Register	CURRENT_POSITION	32-bit	Current position. Values range from -2147483647 to + 2147483648
3005h	Input Register	TEMPERATURE_MCU	16-bit	CPU temperature
3006h	Input Register	TEMPERATURE_MOSFET	16-bit	Power circuit temperature
3007h	Input Register	TEMPERATURE_BRAKE	16-bit	Brake circuit temperature
3008h	Input Register	TASK_COUNTER	16-bit	Returns a random value (to check the transmission channel operation)
3009h	Input Register	STATUS_USER_PROGRAM	16-bit	The current state of the user program execution

This group of registers is read-only and represents the current state of the controller.

3000h – **STATUS** - current state of motor control, possible values:

0 motor stop
1 forward rotation
2 backward rotation

3001h - **CURRENT_VALID** - the current value of the current consumed by the motor. Represents the value of the current consumed by the motor from the power supply, the value in mA.

3002 - **SPEED_VALID** - the instantaneous value of the rotation speed. Units of measurement are revolutions per minute.

3003h - **CURRENT_POSITION** – the current position. Positioning is performed in the range of values from -2147483647 to + 2147483648 of the total number of Hall sensor switchings, both leading and trailing edges are taken into account.

3005h - **TEMPERATURE_MCU** — temperature of the central processor. The temperature is calculated as TEMPERATURE_MCU/10 deg/C.

3006h - **TEMPERATURE_MOSFET** - power circuit temperature - the temperature in the area of installation of MOSFET switches. The temperature is calculated as TEMPERATURE_MOSFET/10 deg/C.

3007h - **TEMPERATURE_BRAKE** - brake circuit temperature – the temperature in the area of the braking resistor. The temperature is calculated as TEMPERATURE_MOSFET/10 deg/C.

3008h - **TASK_COUNTER** - returns a random number in the range 0x0000 to 0xFFFF.

3009h - **STATUS_USER_PROGRAM** - the current state of the user program execution, possible values:

- 1 – user program is stopped by a command via Modbus
- 2 – user program is started by a command via Modbus (this state is a short period only, before state 4 is set)
- 3 – user program is started after supply voltage is on (this state is a short period only, before state 4 is set)
- 4 – user program is being executed
- 5 – user program is finished with the END command
- 6 – user program is finished with the ENDF command
- 7 – user program is stopped due to an error

6.4. RS-485 Modbus data transmission settings registers

Address	Type	Name	Size	Description
5000h	Holding Register	SLAVE_ADDRESS_MODBUS	16-bit	ID of the controller (device address). Valid values: 0...247. (0 - broadcast address, no reply message)
5001h	Holding Register	TYPE_MODBUS	16-bit	Data transmission settings
5002h	Holding Register	BITRATE_MODBUS	16-bit	Setting the baud rate
5003h	Holding Register	TIMEOUT_BROADCAST_MODBUS	16-bit	An additional delay between the received packet and the reply message.

5001h - **TYPE_MODBUS** – data transmission settings, valid values:

- 1 – ASCII, 7 data bit, even, 1 stop bit,
- 2 – ASCII, 7 data bit, odd, 1 stop bit
- 3 – RTU, 8 data bit, even, 1 stop bit
- 4 – RTU, 8 data bit, odd, 1 stop bit
- 5 – RTU, 8 data bit, none, 2 stop bit

5002h - **BITRATE_MODBUS** – Modbus baud rate, possible values:

- 0 – 600
- 1 – 1200,
- 2 – 2400,
- 3 – 4800,
- 4 – 9600,
- 5 – 14400,
- 6 – 19200,
- 7 – 38400,
- 8 – 57600,
- 9 – 115200,
- 10 – 128000

5003h - **TIMEOUT_BROADCAST_MODBUS** - is used if the control device takes a long time to switch from transmit mode to receive mode.

After changing the communication settings, the new values must be saved using the FLAG_SAVE_INI register (see section 6.5), and then the controller must be rebooted. After the reboot, the RS-485 connection will be performed using the new settings.

6.5. Registers for the setting of drive operation

Address	Type	Name	Size	Description
5004h	Holding Register	MODE_DEVICE	16-bit	Controller operation mode.
5005h	Holding Register	MODE_USER_PROGRAM	16-bit	Control command for starting a user program.
5006h	Holding Register	MODE_ROTATION	16-bit	Rotation mode.
5007h	Holding Register	MODE_EXT_IN	16-bit	Configuration of the external inputs IN, IN2.
5008h	Holding Register	POSITION_N	16-bit	The position number to move. Values range from 1 to 4
5009h	Holding Register	REF_CURRENT	16-bit	Current consumption limiting. Range: 1000 mA to 20000 mA for BMSD-20Modbus 2000 mA to 40000 mA for BMSD-40Modbus
500Ah	Holding Register	RATED_SPEED	16-bit	Rated speed of the motor: Range from 1000 to 15000 rpm
500Bh	Holding Register	SPEED	16-bit	Set rotation speed. Range from 30 to 15000 rpm.
500Ch	Holding Register	ACC	16-bit	Set acceleration. Range from 10 to 1000.
500Dh	Holding Register	DEC	16-bit	Set deceleration. Range from 10 to 1000.
500Eh	Holding Register	DIRECTION	16-bit	Rotation direction. 1 – forward 2 – backward
500Fh	Holding Register	PULSES_PER_REVOLUTION	16-bit	Number of pulses per revolution from one Hall sensor: Range from 1 to 12

5010h	Holding Register	USE_HALL	16-bit	Use of Hall sensors: 0 – without Hall sensors 1 – one Hall sensor 2 – two Hall sensors
5011h	Holding Register	MODE_COIL	16-bit	State of motor terminals when stopped: 0 - open 1 - closed
5012h	Holding Register	OFFSET_COMPENSATION	16-bit	Motor braking correction
5013h	Holding Register	PRESSED_INPUTS_EXTERN	16-bit	Minimum positive pulse time at digital inputs IN1, IN2 (ms)
5014h	Holding Register	WAITED_INPUTS_EXTERN	16-bit	Minimum time for the negative part of the pulse at digital inputs IN1, IN2 (ms)
5015h	Holding Register	OFFSET	32-bit	The offset to move, the value is changed by the controller during operation. Range of values from -2147483647 to + 2147483648
5017h	Holding Register	OFFSET_CONST	32-bit	Increment - the offset to which it is necessary to move.
5019h	Holding Register	TARGET_POSITION	32-bit	The position to move. Range of values from -2147483647 to + 2147483648
501Bh	Holding Register	TARGET_POSITION1	32-bit	User preset position №1 Range of values from -2147483647 to + 2147483648
501Dh	Holding Register	TARGET_POSITION2	32-bit	User preset position №2 Range of values from -2147483647 to + 2147483648
501Fh	Holding Register	TARGET_POSITION3	32-bit	User preset position №3 Range of values from -2147483647 to + 2147483648
5021h	Holding Register	TARGET_POSITION4	32-bit	User preset position №4 Range of values from -2147483647 to + 2147483648
5023h	Holding Register	ERROR	16-bit	Errors register
5024h	Holding Register	FLAG_SAVE_INI	16-bit	Register to save user settings. (Registers 5000h..501Fh). Value: 0x37FA
5025h	Holding Register	FLAG_SAVE_USER_PROGRAM	16-bit	Register for writing or reading a user program to or from non-volatile memory. Value to write: 0x8426 Value to read: 0x9346

5026h	Holding Register	FLAG_RESTART	16-bit	Reboot register. Value: 0x95AF

5004h - **MODE_DEVICE** - controller operation mode, possible values:

- 1 – control via Modbus
- 2 – control by external signals

5005h – **MODE_USER_PROGRAM** – control command for starting a user program, possible values:

- 1 – stop user program
- 2 – start user program
- 3 – start user program as soon as the power is on (pre-saving of settings is required - see register FLAG_SAVE_INI)

The user program can be stored in the internal flash memory of the controller. To launch executing a user program, it is necessary to write 2 into the register MODE_USER_PROGRAM. To stop the program executing, it is necessary to write 1 to the register MODE_USER_PROGRAM.

5006h – **MODE_ROTATION** – rotation mode, possible values:

- 1 – continuous rotation
- 2 – offset by the amount specified by the OFFSET register
- 3 – moving to a given position. The position number is specified by the POSITION_N register (from 1 to 4), coordinates of the positions are specified by the registers POSITION1, POSITION2, POSITION3, and POSITION4 accordingly.

5007h – **MODE_EXT_IN** – configuration of the operation mode of the external inputs IN1, IN2 in the control mode of external signals, possible values:

- 1 – Input IN1 is used as a start/stop signal of the drive, it is processed on the falling edge of the pulse; input IN2 is used as a reverse signal, it is processed on the falling edge of the pulse.
- 2 - Input IN1 is used as a start/stop signal of the drive, it is processed on the falling edge of the pulse; input IN2 is used as a direction reference signal, processed according to the signal level.
- 3 - Input IN1 is used as a start/stop signal of the drive, it is processed according to the signal level: , presence of a signal - enable the motor rotation, no signal - stop; input IN2 is used as a reverse signal, it is processed on the falling edge of the pulse.
- 4 - Input IN1 is used as a start/stop signal of the drive, it is processed according to the signal level: , presence of a signal - enable the motor rotation, no signal - stop; input IN2 is used as a direction reference signal, processed according to the signal level.
- 5 – input IN1 is used as a signal to start and stop the drive in the forward direction, IN2 - as a signal to start and stop the drive in the opposite direction; both signals are processed by level.

5008h – **POSITION_N** – selection of the position number for movement (from 1 to 4) - used in the mode of movement to a given position (MODE_ROTATION = 3) in conjunction with the registers POSITION1, POSITION2, POSITION3, POSITION4.

5009h – **REF_CURRENT** – setting the current consumption limit. The range of permissible values for BMSD-20Modbus: from 1000 mA to 20000 mA; for BMSD-40Modbus: from 2000 mA to 40000 mA.

500Ah – **RATED_SPEED** – rated speed of the motor: from 1000 to 15000 rpm. It is impossible to know the actual motor speed if the motor is used without Hall sensors. For this reason, the maximum motor speed (which corresponds to the maximum supply on the motor) is used to calculate the set values of the speed. If the RATED_SPEED is not set correctly, the actual motor speed substantially differs from the target value, which is set in the SPEED register.

500Bh – **SPEED** – target rotation speed (from 30 to 15000 rpm). The value of the target speed must be below the MAX_SPEED value.

500Ch – **ACC** – given acceleration (from 10 to 1000)

500Dh – **DEC** – given deceleration (from 10 to 1000)

Acceleration and deceleration values in registers ACC and DEC are conventional linear values that determine the rate of acceleration/deceleration. A value of 10 corresponds to 100 rps^2, and a value of 1000 corresponds to 5000 rps^2.

500Eh – **DIRECTION** – direction of rotation, possible values:

- 1 – forward rotation
- 2 – backward rotation.

500Fh – **PULSES-PER_REVOLUTION** – number of pulses per revolution from one Hall sensor (from 1 to 12).

5010h – **USE_HALL** – using of Hall sensors:

- 0 – a motor without Hall sensors
- 1 – one Hall sensor
- 2 – two Hall sensors.

5011h – **MODE_COIL** – State of motor terminals when stopped:

- 0 - open
- 1 - closed

5012h - **OFFSET_COMPENSATION** – Experimentally calculated motor braking correction for current settings ACC, DEC, SPEED. If the stop occurs beyond the calculated point, then the compensation value is equal to the positioning error with a negative sign, if the motor stops before the stop point, then the compensation value is positive. For example, the error of reaching the specified position is 15 increments, which means OFFSET_COMPENSATION = -15, and this correction is valid only for the current settings of acceleration, deceleration, and speed.

5013h and 5014h – **PRESSED_INPUTS_EXTERN** and **WAITED_INPUTS_EXTERN** – minimum time of a positive and a negative part of a pulse at the digital inputs IN1, IN2 – used to suppress contact bounce (see fig. 5).

5015h – **OFFSET** – the offset to be moved is used when MODE_ROTATION = 2, valid values from -2147483647 to + 2147483648. Before starting the movement, it is necessary to set the required offset value in the OFFSET or OFFSET_CONST register, which is processed by the controller as a counter of the remaining movement. During the execution of the specified movement, the OFFSET value decreases.

5017h – **OFFSET_CONST** - Increment - the offset to which it is necessary to move. If the value of the register OFFSET_CONST ≠ 0, it is copied to the register OFFSET at every motor start.

5019h – **TARGET_POSITION** – position to move to, allowed values from -2147483647 to + 2147483648. In the mode of moving to a given position, the coordinate from POSITION1-POSITION4 is copied to this register before moving, while the POSITION1-POSITION4 registers do not change their values.

501Bh - 5021h – **TARGET_POSITION1..4** – User preset position №1..4, used when MODE_ROTATION = 3, position number is determined by register POSITION_N, allowed values from -2147483647 to + 2147483648.

5023h – **ERROR** – errors that occur during controller operation - each bit of the register signals a specific error:

- bit 0 - out of the supply voltage range;
- bit 1 - short circuit of the motor windings;
- bit 2 - overheating of the brake circuit;
- bit 3 - overheating of the power circuit;
- bit 4 - Hall sensors connecting error;
- bit 5 - emergency stop;
- bit 6 - MCU overheating;
- bit 7 - test control program;
- bit 8 - user program execution error;
- bit 9 - error reading or writing settings;

- bit 10 - error in the operation of the output transistor switches;
- bit 11 - consumption current limit is exceeded;
- bit 12 - warning about the impossibility of calculating the breakpoint;
- bit 13 - warning about an attempt to write to the register a value that is out of range;
- bit 14 – RS-485 transmitting parity error;
- bit 15 - using positioning mode with less than two Hall sensors.

5024h – **FLAG_SAVE_INI** – register for saving user settings - when writing the value 0x37FA to this register, the settings defined by registers 5000h..501Fh will be saved to the non-volatile memory.

5025h – **FLAG_SAVE_USER_PROGRAM** – writing the value 0x8426 to this register starts the procedure for saving the user program from the temporary buffer to the nonvolatile memory of the controller. Writing the value 0x9346 starts the procedure for reading the user program from the nonvolatile memory of the controller into a temporary buffer (see section 6.6).

5026h – **FLAG_RESTART** – writing the value 0x95AF to this register causes the controller reboot.

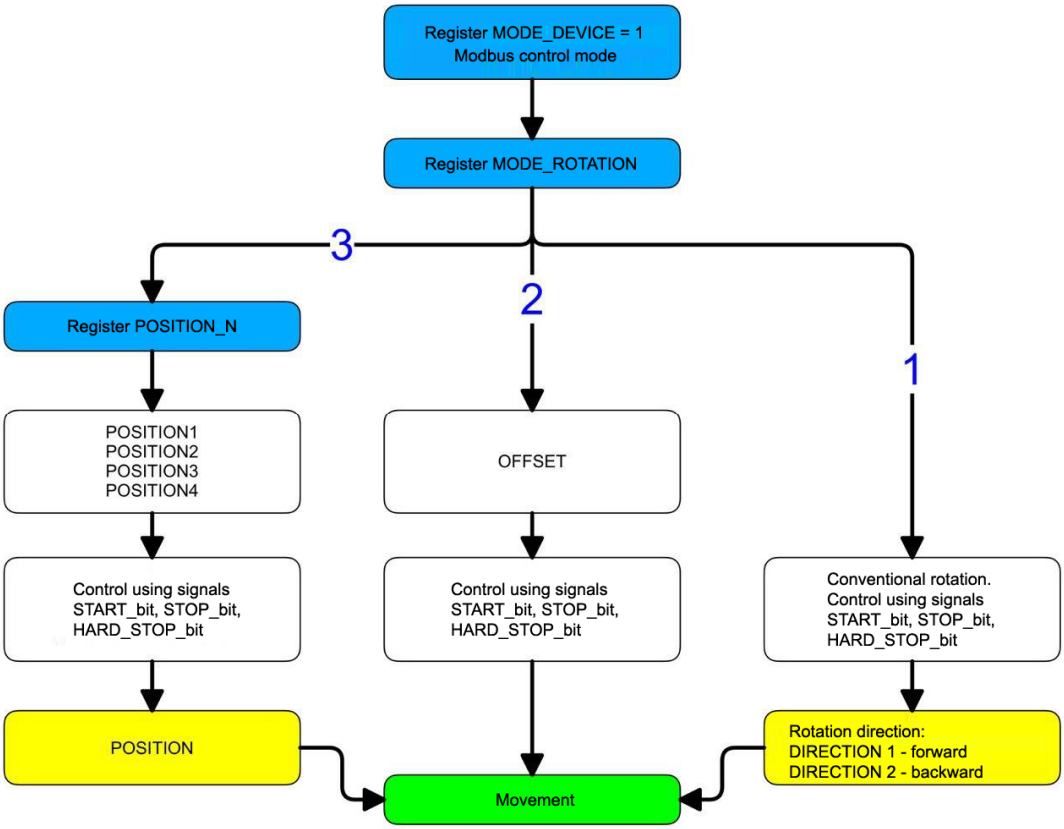

Fig.6. Flow diagram for selecting the operating mode when controlling the drive via Modbus

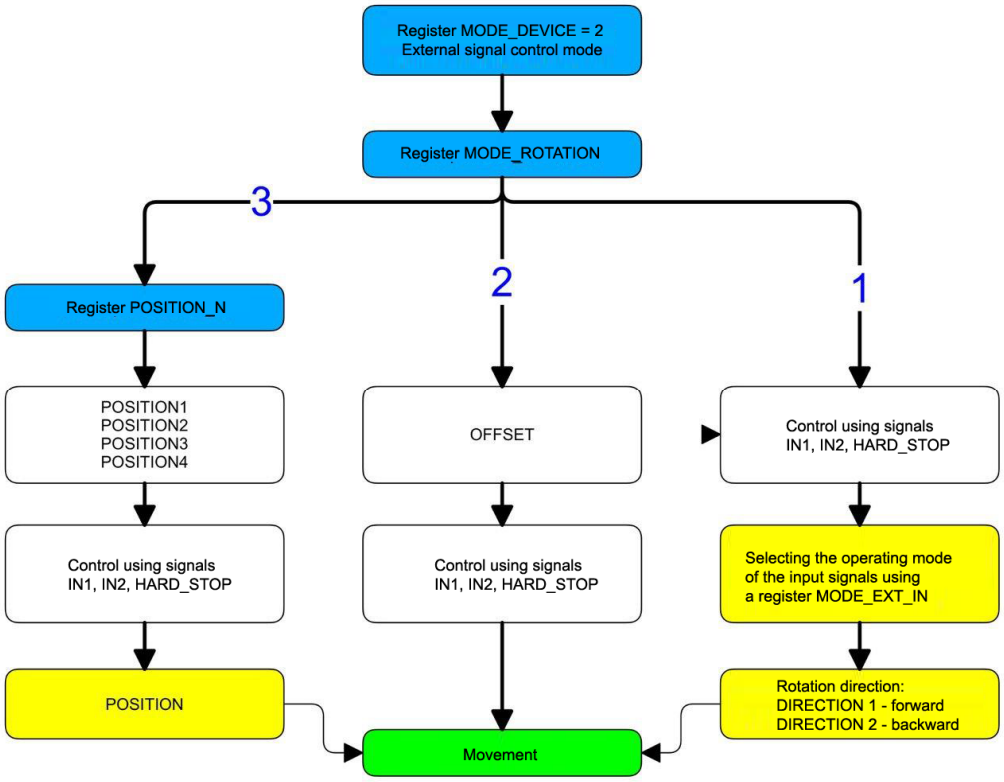

Fig.7. Flow diagram for selecting the operating mode when controlling the drive using external signals

6.6. Reading and writing a user program

A temporary buffer of 1024 commands is used to read and write a user program. The special register FLAG_SAVE_INI is used to save the program from the temporary buffer to the non-volatile memory and to read the program from the controller memory to the temporary buffer.

Address	Type	Name	Size	Description
5025h	Holding Register	FLAG_SAVE_USER_PROGRAM	16-bit	Register for writing or reading a user program to or from the non-volatile memory of the controller. Value for writing: 0x8426 Value for reading: 0x9346
6000h	Holding Register	WRITE_CMD	16-bit	Address register. When the address value is written to this register, the command is transferred from the CMD_W field to the specified address of the temporary buffer of the user program.
6001h	Holding Register	CMD_W	32-bit	User program's instructions
6003h	Holding Register	READ_CMD	16-bit	Address register. When the address value is written to this register, the command is transferred from the specified address of the temporary buffer of the user program to the CMD_R field.

6004h	Holding Register	CMD_R	32-bit	User program's instructions

Assembling and writing a user program to the controller memory

Every user program instruction consists of two memory words (32 bits) - command (16 bits) and command data (16 bits). When assembling a user program, instructions are first written into a temporary buffer in the controller. To write to the temporary buffer, it is necessary to write an instruction to the CMD_W register, and then write the address of this instruction in the internal buffer to the WRITE_CMD register. When the address is written to the WRITE_CMD register, the instruction is transferred from the CMD_W register to the internal buffer. After composing a user program in a temporary buffer, it is necessary to write the value 0x8426 to the FLAG_SAVE_USER_PROGRAM register - the program will be transferred from the temporary buffer to the controller's FLASH memory.

Reading a user program from the controller memory

It is necessary to transfer a user program to the temporary buffer of the controller to read it from the FLASH memory. To do this, write the value 0x9346 to the FLAG_SAVE_USER_PROGRAM register - the program will be transferred from the controller's FLASH memory to the temporary buffer. Then, to read an instruction from the temporary buffer, it is necessary to write the instruction address to the READ_CMD register - when the address is written to this register, the instruction will be copied from the temporary buffer to the CMD_R register.

A diagram of the procedures for reading and writing a user program is shown in Fig. 8.

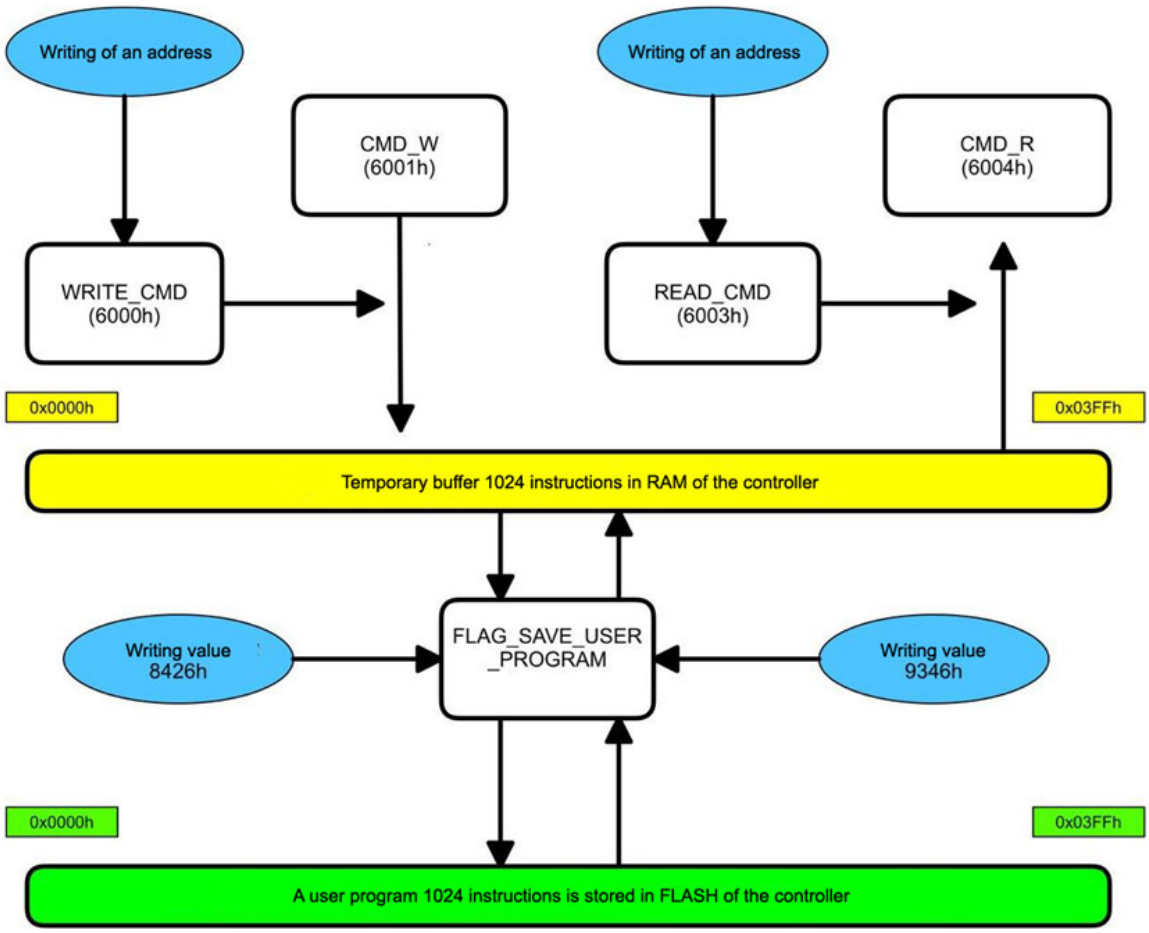

Fig.8. Flow diagram of the procedures for reading and writing a user program

6.7. System registers

Address	Type	Name	Size	Description
7000h	Holding Register	AX_REG	16-bit	Data storage register for user program
7001h	Holding Register	BX_REG	16-bit	Data storage register for user program
7002h	Holding Register	CX_REG	16-bit	Data storage register for user program
7003h	Holding Register	DX_REG	16-bit	Data storage register for user program
7004h	Holding Register	EX_REG	16-bit	Data storage register for user program
7005h	Holding Register	FX_REG	16-bit	Data storage register for user program
7006h	Holding Register	PC_REG	16-bit	Register-pointer to the current executing user command
7007h	Holding Register	GX_REG	16-bit	Data storage register for user program
7008h	Holding Register	HX_REG	16-bit	Data storage register for user program
7009h	Holding Register	IX_REG	16-bit	Data storage register for user program
700Ah	Holding Register	JX_REG	16-bit	Data storage register for user program

System registers AX_REG..FX_REG (value range 0..65535) are intended for temporary storage of data during the execution of a user program. PC_REG is a pointer to the currently executing user program instruction. More detailed description in section 6.9.

6.8. Identification registers

Address	Type	Name	Size	Description
8001h	Input Register	HW_MAJOR	16-bit	Driver type
8002h	Input Register	HW_MINOR	16-bit	Hardware version
8003h	Input Register	FW_MAJOR	16-bit	Software identifier
8004h	Input Register	FW_MINOR	16-bit	Software version

The registers are necessary to determine the functional purpose of the control unit, its characteristics, and software version via the network.

For BMSD-20Modbus, the values are:

- HW_MAJOR 1001
- HW_MINOR x
- FW_MAJOR x
- FW_MINOR x

For BMSD-40Modbus, the values are:

- HW_MAJOR 1002
- HW_MINOR x
- FW_MAJOR x
- FW_MINOR x

6.9. User program instructions

The structure of user program instructions is shown in the diagram in Fig. 9.

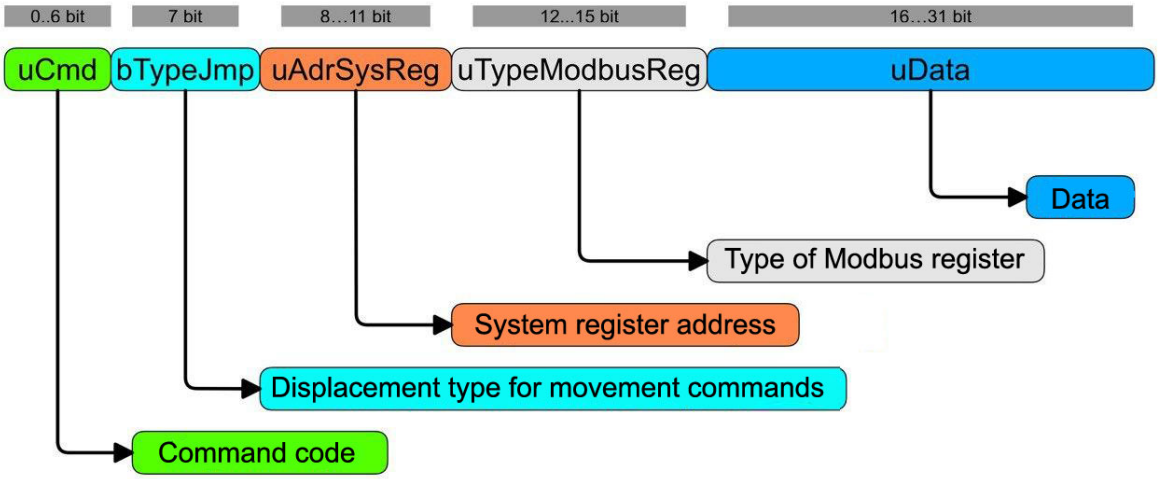

Fig. 9. Structure of a user program instruction

The user program instruction is 32 bits in size and contains the following fields:

uCmd – 7 bits – command code

bTypeJmp – 1 bit – displacement type for movement commands:

- 0 – absolute value
- 1 – relative value

uAdrSysReg – 4 bits - address of the system register AX_REG ... FX_REG with numbers 0 ... 5

uTypeModbusReg – 4 bits – type of a Modbus register:

- 0 – Discrete Inputs
- 1 – Coils
- 2 – Inputs
- 3 – Holding registers

uData – 16 bits - data.

64-bit instructions consist of two 32-bit command lines. The first command line of this instruction is a command similar to a 32-bit instruction. The second command line contains 16 or 32-bit **uCMD_DATA data**.

Instructions with a D prefix, such as DMOV, operate on 32-bit data located in two consecutive 16-bit registers. In the address field of such instructions, the lower register is indicated. When reading a value, the command automatically reads two consecutive 16-bit registers starting from the specified address and forms a single 32-bit value. When writing a value, the instruction splits a 32-bit number into two 16-bit parts and writes them into two consecutive registers starting at the specified address.

The following table lists commands with options for filling fields. If the field is not specified, then it is not used in this command.

uCmd		-	-	-
0x00	CMD_STOP_PROGRAM	-	-	-
0x0D	CMD_FULL_STOP_PROGRAM			
uCmd		**bTypeJmp**	**uData**	-
0x05	CMD_JMP	0 – absolute value	Move address 0..1024	-
0x06	CMD_JMP_AX_PARI_BX			

0x07	CMD_JMP_AX_NOPARI_BX	1 – relative value	or offset +-1024	
0x08	CMD_JMP_AX_MORE_BX			
0x09	CMD_JMP_AX_LESS_BX			
0x63	CMD_DJMP_GX_PARI_IX			
0x64	CMD_DJMP_GX_NOPARI_IX			
0x65	CMD_DJMP_GX_MORE_IX			
0x66	CMD_DJMP_GX_LESS_IX			
uCmd		**uData**	-	-
0x04	CMD_DELAY	Delay time 0..65535 ms	-	-
0x0A	CMD_CALL	Subroutine address 0..1024		
0x0B	CMD_RETURN	-		
0x0C	CMD_FOR	Cycle length and number of cycles		
uCmd		**uAdrSysReg**	-	-
0x15	CMD_NOT_SYSREG	System register address (0..9)	-	-
0x17	CMD_DNOT_SYSREG			
uCmd		**uTypeModbusReg**	**uData**	-
0x16	CMD_NOT_MODBUS	1 – Coils 3 – Holding registers	Modbus register address 0..65535	-
0x18	CMD_DNOT_MODBUS			
uCmd		**uAdrSysReg**	**uData**	-
0x01	CMD_MOV_SYSREG_CONST	System register address (0..9)	Constant 0..65535	-
0x19	CMD_ADD_SYSREG_CONST			
0x1A	CMD_SUB_SYSREG_CONST			
0x1B	CMD_DIV_SYSREG_CONST			
0x1C	CMD_MUL_SYSREG_CONST			
0x1D	CMD_AND_SYSREG_CONST			
0x1E	CMD_OR_SYSREG_CONST			
0x1F	CMD_XOR_SYSREG_CONST			
uCmd		**uAdrSysReg**	**uTypeModbusReg**	**uData**
0x03	CMD_MOV_SYSREG_MODBUS	System register address (0..9)	0 – Discrete Inputs 1 – Coils 2 – Inputs 3 – Holding registers	Modbus register address 0..65535
0x21	CMD_ADD_SYSREG_MODBUS			
0x22	CMD_SUB_SYSREG_MODBUS			
0x23	CMD_DIV_SYSREG_MODBUS			
0x24	CMD_MUL_SYSREG_MODBUS			
0x25	CMD_AND_SYSREG_MODBUS			
0x26	CMD_OR_SYSREG_MODBUS			
0x27	CMD_XOR_SYSREG_MODBUS			
uCmd		**uTypeModbusReg**	**uData**	**uCMD_DATA**
0x0E	CMD_MOV_MODBUS_CONST	1 – Coils 3 – Holding registers	Modbus register address 0..65535	Constant 0..65535 (separate command line)
0x28	CMD_ADD_MODBUS_CONST			
0x29	CMD_SUB_MODBUS_CONST			
0x2A	CMD_DIV_MODBUS_CONST			
0x2B	CMD_MUL_MODBUS_CONST			
0x2C	CMD_AND_MODBUS_CONST			
0x2D	CMD_OR_MODBUS_CONST			

0x2E	CMD_XOR_MODBUS_CONST			
uCmd		**uAdrSysReg**	**uTypeModbusReg**	**uData**
0x02	CMD_MOV_MODBUS_SYSREG	System register address (0..9)	0 – Discrete Inputs 1 – Coils 2 – Inputs 3 – Holding registers	Modbus register address 0..65535
0x30	CMD_ADD_MODBUS_SYSREG			
0x31	CMD_SUB_MODBUS_SYSREG			
0x32	CMD_DIV_MODBUS_SYSREG			
0x33	CMD_MUL_MODBUS_SYSREG			
0x34	CMD_AND_MODBUS_SYSREG			
0x35	CMD_OR_MODBUS_SYSREG			
0x36	CMD_XOR_MODBUS_SYSREG			
uCmd		**uAdrSysReg**	**uData**	-
0x0F	CMD_MOV_SYSREG_SYSREG	System register address (0..9)	System register address (0..9)	-
0x37	CMD_ADD_SYSREG_SYSREG			
0x38	CMD_SUB_SYSREG_SYSREG			
0x39	CMD_DIV_SYSREG_SYSREG			
0x3A	CMD_MUL_SYSREG_SYSREG			
0x3B	CMD_AND_SYSREG_SYSREG			
0x3C	CMD_OR_SYSREG_SYSREG			
0x3D	CMD_XOR_SYSREG_SYSREG			
uCmd		**uAdrSysReg**	**uCMD_DATA**	-
0x10	CMD_DMOV_SYSREG_CONST	System register address (0..9)	Constant 0…4294967295 (separate command line)	-
0x3E	CMD_DADD_SYSREG_CONST			
0x3F	CMD_DSUB_SYSREG_CONST			
0x40	CMD_DDIV_SYSREG_CONST			
0x41	CMD_DMUL_SYSREG_CONST			
0x42	CMD_DAND_SYSREG_CONST			
0x43	CMD_DOR_SYSREG_CONST			
0x44	CMD_DXOR_SYSREG_CONST			
uCmd		**uAdrSysReg**	**uTypeModbusReg**	**uData**
0x11	CMD_DMOV_SYSREG_MODBUS	System register address (0..9)	0 – Discrete Inputs 1 – Coils 2 – Inputs 3 – Holding registers	Modbus register address 0..65535
0x46	CMD_DADD_SYSREG_MODBUS			
0x47	CMD_DSUB_SYSREG_MODBUS			
0x48	CMD_DDIV_SYSREG_MODBUS			
0x49	CMD_DMUL_SYSREG_MODBUS			
0x4A	CMD_DAND_SYSREG_MODBUS			
0x4B	CMD_DOR_SYSREG_MODBUS			
0x4C	CMD_DXOR_SYSREG_MODBUS			
uCmd		**uTypeModbusReg**	**uData**	**uCMD_DATA**
0x12	CMD_DMOV_MODBUS_CONST	1 – Coils 3 – Holding registers	Modbus register address 0..65535	Constant 0…4294967295 (separate command line)
0x4D	CMD_DADD_MODBUS_CONST			
0x4E	CMD_DSUB_MODBUS_CONST			
0x4F	CMD_DDIV_MODBUS_CONST			
0x50	CMD_DMUL_MODBUS_CONST			
0x51	CMD_DAND_MODBUS_CONST			
0x52	CMD_DOR_MODBUS_CONST			
0x53	CMD_DXOR_MODBUS_CONST			

uCmd		uAdrSysReg	uTypeModbusReg	uData
0x13	CMD_DMOV_MODBUS_SYSREG	System register address (0..9)	0 – Discrete Inputs 1 – Coils 2 – Inputs 3 – Holding registers	Modbus register address 0..65535
0x55	CMD_DADD_MODBUS_SYSREG			
0x56	CMD_DSUB_MODBUS_SYSREG			
0x57	CMD_DDIV_MODBUS_SYSREG			
0x58	CMD_DMUL_MODBUS_SYSREG			
0x59	CMD_DAND_MODBUS_SYSREG			
0x5A	CMD_DOR_MODBUS_SYSREG			
0x5B	CMD_DXOR_MODBUS_SYSREG			
uCmd		**uAdrSysReg**	**uData**	-
0x14	CMD_DMOV_SYSREG_SYSREG	System register address (0..9)	System register address (0..9)	-
0x5C	CMD_DADD_SYSREG_SYSREG			
0x5D	CMD_DSUB_SYSREG_SYSREG			
0x5E	CMD_DDIV_SYSREG_SYSREG			
0x5F	CMD_DMUL_SYSREG_SYSREG			
0x60	CMD_DAND_SYSREG_SYSREG			
0x61	CMD_DOR_SYSREG_SYSREG			
0x62	CMD_DXOR_SYSREG_SYSREG			
uCmd		**uAdrSysReg**	**bTypeJmp**	**uData**
0x20	CMD_SH_SYSREG_CONST	System register address (0..9)	0 – left shift 1 – right shift	Shift value
0x45	CMD_DSH_SYSREG_CONST			
uCmd		**uTypeModbusReg**	**uData**	**bTypeJmp**
0x2F	CMD_SH_MODBUS_CONST	1 – Coils 3 – Holding registers	Modbus register address 0..65535	0 – left shift 1 – right shift
0x54	CMD_DSH_MODBUS_CONST			

CMD_STOP_PROGRAM – (command code 0x00) – stop executing a user program, without exiting the user program execution mode. At the end of the program execution, all registers and states of the motor remain as they were before the command was executed (the motor continues to rotate if it was rotating before the command was executed). Before the next start of the program, you must send the CMD_FULL_STOP_PROGRAM command.

CMD_FULL_STOP_PROGRAM – (command code 0x0D) - stops the execution of the user program and exits the program operation mode. At the end of the program execution, all registers and the state of the motor return to their original values, and the motor stops.

CMD_MOV_SYSREG_CONST – (command code 0x01) – writing to the system register with the uAdrSysReg address, values from the uData data field

CMD_MOV_MODBUS_SYSREG – (command code 0x02) – writing the contents from the uAdrSysReg system register to the ModBUS register space defined by the TypeModbusReg field and its address in the uData field.

CMD_MOV_MODBUS_SYSREG – (command code 0x03) – reading the content from the ModBUS register space determined by the TypeModbusReg field and its address in the uData field into one of the uAdrSysReg system registers

CMD_MOV_MODBUS_CONST – (command code 0x0E) – writing the constant contained in the following command line uCMD_DATA to the ModBUS register space defined by the TypeModbusReg field and its address in the uData field.

CMD_MOV_SYSREG_SYSREG – (command code 0x0F) – writing the contents from the system register uAdrSysReg to another system register with the address uData.

CMD_DELAY – (command code 0x04) – pause, ms.

CMD_JMP – (command code 0x05) – jump to the address specified in the uData field.

CMD_JMP_AX_PARI_BX – (command code 0x06) – jump to the address specified in the uData field, if the value in the system register AX_REG is equal to the value in BX_REG

CMD_JMP_AX_NOPARI_BX – (command code 0x07) – jump to the address specified in the uData field if the value in the system register AX_REG is not equal to the value in BX_REG

CMD_JMP_AX_MORE_BX – (command code 0x08) – jump to the address specified in the uData field, if the value in the system register AX_REG is greater than the value in BX_REG

CMD_JMP_AX_LESS_BX – (command code 0x09) – jump to the address specified in the uData field, if the value in the system register AX_REG is less than the value in BX_REG

CMD_CALL – (command code 0x0A) – call a subroutine starting at the address specified in the uData field.

CMD_RETURN – (command code 0x0B) – return from the subroutine.

CMD_FOR – (command code 0x0C) – cyclic execution of a sequence of commands. The high byte of the uData field contains the number of commands located after the CMD_FOR command that will be repeated in a cycle. The least significant byte of the uData field contains the number of repetitions. For example: uData = 0x1705 - 0x17 = 23 commands executed in a loop, 0x05 = 5 - the number of repetitions.

CMD_DJMP_GX_PARI_IX – (command code 0x63) – jump to the address specified in the uData field, if the contents of a pair of system registers GX_REG and HX_REG are equal to the contents of IX_REG and JX_REG.

CMD_DJMP_GX_NOPARI_IX – (command code 0x64) – jump to the address specified in the uData field, if the contents of a pair of system registers GX_REG and HX_REG are not equal to the contents of IX_REG and JX_REG.

CMD_DJMP_GX_MORE_IX – (command code 0x65) – jump to the address specified in the uData field, if the contents of a pair of system registers GX_REG and HX_REG are greater than the contents of IX_REG and JX_REG.

CMD_DJMP_GX_LESS_IX – (command code 0x66) – jump to the address specified in the uData field, if the contents of a pair of system registers GX_REG and HX_REG are less than the contents of IX_REG and JX_REG.

Mathematical operations are possible between system registers, Modbus registers, and constants. The operands of instructions that operate on 32-bit data are located in two consecutive 16-bit registers. When accessing a register, the instruction specifies the address of the lower register. The high register address is obtained by incrementing the low register address by one.

The table below shows math instructions, instruction codes for operation with 16-bit and 32-bit data, and operand locations:

Command			**Operand 1**	**Operand 2**	**Result**
Name	**Code**				
	16-bit data	**32-bit data**	**S1**	**S2**	**D**
Addition (S1 + S2 = D)					
CMD_ADD_SYSREG_CONST	0x19	0x3E	SYS_REG_1	CONST_1	SYS_REG_1
CMD_ADD_SYSREG_MODBUS	0x21	0x46	SYS_REG_1	Modbus_REG	SYS_REG_1
CMD_ADD_MODBUS_CONST	0x28	0x4D	Modbus_REG	CONST_2	Modbus_REG
CMD_ADD_MODBUS_SYSREG	0x30	0x55	Modbus_REG	SYS_REG_1	Modbus_REG
CMD_ADD_SYSREG_SYSREG	0x37	0x5C	SYS_REG_1	SYS_REG_2	SYS_REG_1
Subtraction (S1 - S2 = D)					
CMD_SUB_SYSREG_CONST	0x1A	0x3F	SYS_REG_1	CONST_1	SYS_REG_1
CMD_SUB_SYSREG_MODBUS	0x22	0x47	SYS_REG_1	Modbus_REG	SYS_REG_1
CMD_SUB_MODBUS_CONST	0x29	0x4E	Modbus_REG	CONST_2	Modbus_REG

CMD_SUB_MODBUS_SYSREG	0x31	0x56	Modbus_REG	SYS_REG_1	Modbus_REG
CMD_SUB_SYSREG_SYSREG	0x38	0x5D	SYS_REG_1	SYS_REG_2	SYS_REG_1
Division, remainder discarded (S1 / S2 = D)					
CMD_DIV_SYSREG_CONST	0x1B	0x40	SYS_REG_1	CONST_1	SYS_REG_1
CMD_DIV_SYSREG_MODBUS	0x23	0x48	SYS_REG_1	Modbus_REG	SYS_REG_1
CMD_DIV_MODBUS_CONST	0x2A	0x4F	Modbus_REG	CONST_2	Modbus_REG
CMD_DIV_MODBUS_SYSREG	0x32	0x57	Modbus_REG	SYS_REG_1	Modbus_REG
CMD_DIV_SYSREG_SYSREG	0x39	0x5E	SYS_REG_1	SYS_REG_2	SYS_REG_1
Multiplication (S1 * S2 = D)					
CMD_MUL_SYSREG_CONST	0x1C	0x41	SYS_REG_1	CONST_1	SYS_REG_1
CMD_MUL_SYSREG_MODBUS	0x24	0x49	SYS_REG_1	Modbus_REG	SYS_REG_1
CMD_MUL_MODBUS_CONST	0x2B	0x50	Modbus_REG	CONST_2	Modbus_REG
CMD_MUL_MODBUS_SYSREG	0x33	0x58	Modbus_REG	SYS_REG_1	Modbus_REG
CMD_MUL_SYSREG_SYSREG	0x3A	0x5F	SYS_REG_1	SYS_REG_2	SYS_REG_1
Bitwise logical AND (S1 & S2 = D)					
CMD_AND_SYSREG_CONST	0x1D	0x42	SYS_REG_1	CONST_1	SYS_REG_1
CMD_AND_SYSREG_MODBUS	0x25	0x4A	SYS_REG_1	Modbus_REG	SYS_REG_1
CMD_AND_MODBUS_CONST	0x2C	0x51	Modbus_REG	CONST_2	Modbus_REG
CMD_AND_MODBUS_SYSREG	0x34	0x59	Modbus_REG	SYS_REG_1	Modbus_REG
CMD_AND_SYSREG_SYSREG	0x3B	0x60	SYS_REG_1	SYS_REG_2	SYS_REG_1
Bitwise logical OR (S1 | S2 = D)					
CMD_OR_SYSREG_CONST	0x1E	0x43	SYS_REG_1	CONST_1	SYS_REG_1
CMD_OR_SYSREG_MODBUS	0x26	0x4B	SYS_REG_1	Modbus_REG	SYS_REG_1
CMD_OR_MODBUS_CONST	0x2D	0x52	Modbus_REG	CONST_2	Modbus_REG
CMD_OR_MODBUS_SYSREG	0x35	0x5A	Modbus_REG	SYS_REG_1	Modbus_REG
CMD_OR_SYSREG_SYSREG	0x3C	0x61	SYS_REG_1	SYS_REG_2	SYS_REG_1
Bitwise logical XOR (S1 ^ S2 = D)					
CMD_XOR_SYSREG_CONST	0x1F	0x44	SYS_REG_1	CONST_1	SYS_REG_1
CMD_XOR_SYSREG_MODBUS	0x27	0x4C	SYS_REG_1	Modbus_REG	SYS_REG_1
CMD_XOR_MODBUS_CONST	0x2E	0x53	Modbus_REG	CONST_2	Modbus_REG
CMD_XOR_MODBUS_SYSREG	0x36	0x5B	Modbus_REG	SYS_REG_1	Modbus_REG
CMD_XOR_SYSREG_SYSREG	0x3D	0x62	SYS_REG_1	SYS_REG_2	SYS_REG_1
Data shift (S1 >> S2 = D or S1 << S2 = D – shift direction is indicated in the field bTypeJmp: 0 – left, 1 - right)					
CMD_SH_SYSREG_CONST	0x20	0x45	SYS_REG_1	CONST_1	SYS_REG_1

CMD_SH_MODBUS_CONST	0x2F	0x54	Modbus_REG	CONST_2	Modbus_REG

CONST_1 - Constant in uData data field

CONST_2 - A constant on the following uCMD_DATA command line

SYS_REG_1 is the value contained in the system register (or two consecutive system registers for 64-bit instructions) at uAdrSysReg. Addresses of system registers 0..9

SYS_REG_2 the value is contained in the system register (or two consecutive system registers for 32-bit data) whose address is specified in the following uCMD_DATA command line

Modbus_REG – the value contained in the Modbus register: register type in the TypeModbusReg field, register address in the uData field

Important: when the execution of a user program is stopped by the command CMD_STOP_PROGRAM, the state of the motor and all registers remain the same as they were set during the program. For this reason, if the motor was rotating at the moment the CMD_STOP_PROGRAM command was executed, it would continue executing the last task, i.e., movement would continue when the program had finished. After the program has finished, the drive rotation can be stopped by writing a register via Modbus (Coils 2001h or Coils 2002h). To prevent uncontrolled rotation, a motor stop command can be set before the program end command, or use the command CMD_END_STOP_PROGRAM.

7. Reset to factory defaults

If necessary, the controller parameters can be reset to the factory values. To do this, before turning on the power supply to the unit, close the first contact of the RJ11 connector - CLR_FLASH_CON (RS-485 connector, see Fig. 4) with the ground GND of the power supply. The red and green LEDs on the unit will light up alternately. Keep CLR_FLASH_CON and GND closed for 5s. After that, the controller parameters will be reset to the factory values.

8. Delivery in complete sets

DC brush motor controller BMSD-20Modbus or BMSD-40Modbus 1 pcs

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.

10. Warranty

Any repairs or modifications are performed by the manufacturer or an authorized company.

The manufacturer guarantees the failure-free operation of the controller for 12 months from the date of sale when the operation conditions are satisfied.

The manufacturer's sales department address:
Smart Motor Devices OÜ
Akadeemia tee 21/6, 12618, Tallinn, Estonia
Phone: + 372 6559914
Email: mail@smd.ee
URL: https://smd.ee

Last modified: 07.2025