DC BRUSHLESS MOTOR CONTROLLER
B105-20Modbus

2020

1. Product designation

Brushless motor controllers B105-20Modbus are electronic devices designed to operate and control brushless synchronous 3-phase dc motors with Hall encoder.

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 built-in Hall sensors. Position holding is also possible.

The brushless 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 in the block memory by a user.

Control via RS-485 Modbus

- The controller can be remotely controlled 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 to one of 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, timers are supported.
- Programmable inputs IN1 and IN2 can be used as signals START/STOP, REVERS or for other purposes at the user's discretion.
- The controller can perform positioning in the range from - 2147483647 to + 2147483648 Hall sensor switching.
- 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:

- Terminals for connecting a potentiometer or an external signal 0-5VDC for analog speed control.
- Contacts for connecting the external signals In1 and In2, the purpose and processing of which is 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 that ensures the Safe Stop 1 (SS1) function - 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 characteristic

Model	B105-20Modbus
Power supply voltage, VDC	24 - 48
Power supply protection, VDC	20 - 51
Rated current in motor phase, A	<20
Max current in motor phase, A	<80
Regulation of peak power, W	300..960
Input resistance of SPEED input, kOhm	20
Input voltage range of SPEED input, VDC	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

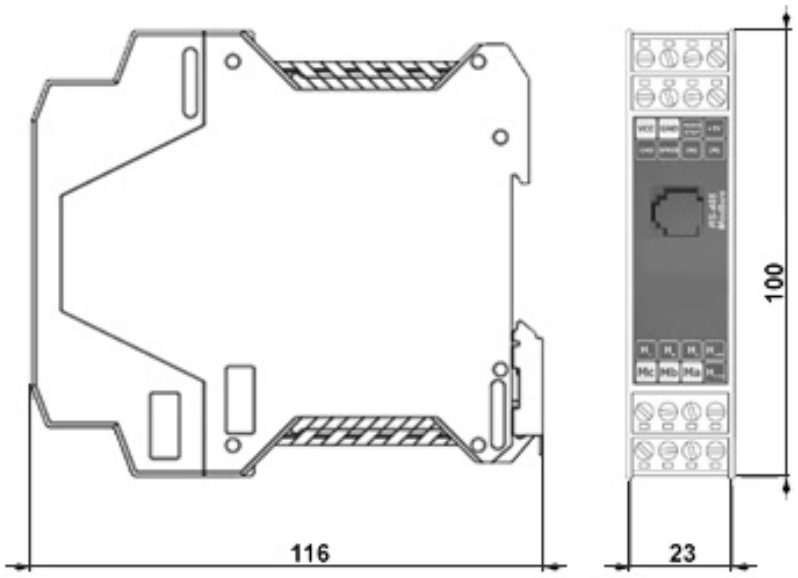

Fig.1. The overall and connecting dimensions of the controller BLSD-20Modbus

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

B105-20MODBUS is designed as circuit plate with electronics elements, covered with a case with DIN rail mount. On the 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;
- - terminals for connecting an external potentiometer to regulate the motor rotation speed;
- - 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).

The “SPEED” input is intended for speed control by an external analog signal. 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. Output +5 VDC for external potentiometer
2. Emergency stop signal “HARD STOP”
3. Power supply GND
4. Power supply 24 – 48 VDC
5. Signal “IN1” (clean contact)
6. Signal “IN2” (clean contact)
7. Analog signal input – for connection of an external speed regulation potentiometer
8. Signal GND
9. Output for supply of Hall sensors
10. Hall sensor – phase A
11. Hall sensor – phase B
12. Hall sensor – phase C
13. GND of Hall sensors
14. Motor phase A
15. Motor phase B
16. Motor phase C

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

RT - contacts for connecting 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 power is off. Do not attempt to change wiring while the power is ON.

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

Follow the next instruction during connection:

1. Connect a motor to the controller according to the fig. 2. Motor phases must be connected to the terminals 14 – 16. Hall sensors signals must be connected to the terminals 10 – 12. GND of HALL sensors must be connected to the terminal 13, supply of HALL sensors signals must be connected to the terminal 9.
2. Connect external control elements according to the connection diagram schemes in fig. 3:

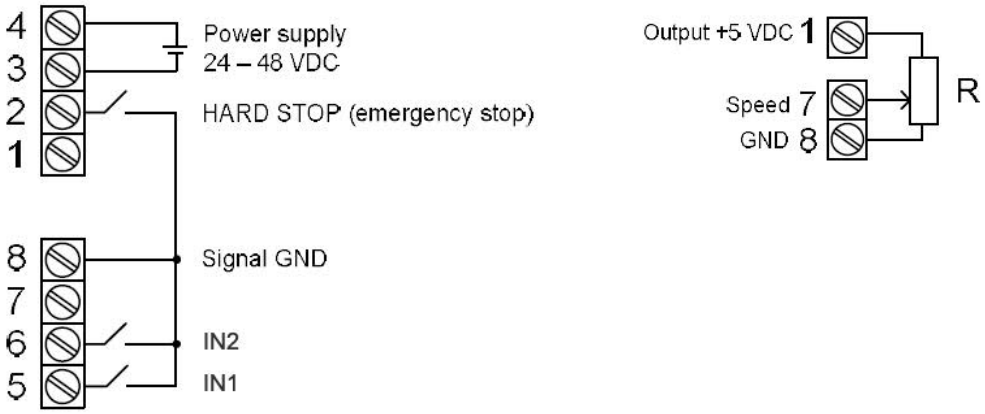

Fig. 3. Connection of power supply and external control elements

- type of external signals «IN1», «IN2», «HARD STOP» - clean contact;
- full resistance of external potentiometer for speed control - approximately 4..5 KOhm.

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

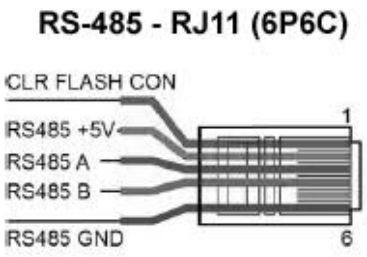

Fig.4 - pin assignment of the RJ11 connector - RS-485 Modbus.

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 "+" of the power supply - to input 4, connect "-" of the power supply - to input 3. Disassembly procedure is in the reverse order.

6. Operation

1. Make sure the power supply is turned off. Please, wire just when power is off.
2. Connect the motor and power supply to the controller according to the 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. To adjust the speed using an external signal source, connect an external resistor to terminals 1 "+5 V", 7 "SPEED" and 8 "GND". The minimum resistance corresponds to the maximum speed; as the resistance increases, the speed decreases. To be able to control the speed by an external analog signal, set the register USE_EXTERN_SPEED = 1 (see description of registers below).
6. Connect the control signals "IN1", "IN2" and the emergency stop signal "HARD STOP" according to the section 5. The signal "HARD STOP" is used for emergency stop of the motor. Operation is permitted with closed contact.
7. Connect the lines of the RS-485 interface according to the section 5.
8. Turn on the power supply. The device is ready for configuration via Modbus protocol.
9. Set the necessary operation parameters by commands via Modbus protocol - motor current limitation, rotation direction, setting of operation of external inputs IN1 and IN2, control method.
10. 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.
11. 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 an external analog signal applied to the SPEED input, or by commands via 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).

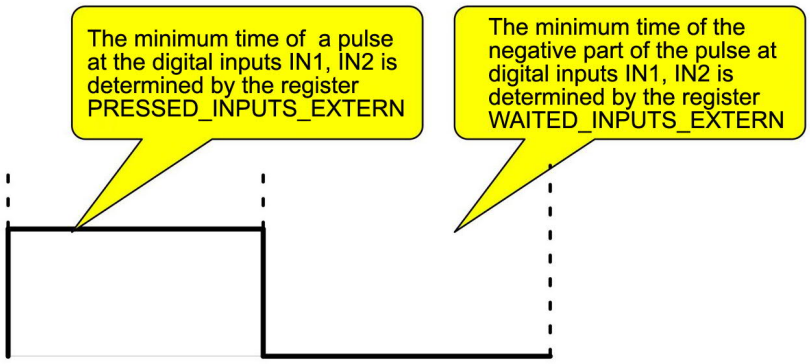

Fig. 5 - Contact bounce suppression

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_	16-bit	The current state of the user program

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, 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	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 slow 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 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	Limiting the maximum operating current in the motor winding. Range from 1000 mA to 20000 mA
500Ah	Holding Register	HOLD_CURRENT	16-bit	Holding current in stop mode. Range from 0 to 1000 mA
500Bh	Holding Register	SPEED	16-bit	Set rotation speed. Range from 35 to 4105 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	N_POLE	16-bit	Number of motor poles. Range from 1 to 12
5010h	Holding Register	USE_EXTERN_SPEED	16-bit	Speed control method: 1 – commands via Modbus 2 – external signal at the SPEED input
5011h	Holding Register	RESERVED	16-bit	reserved
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. Range of values from -2147483647 to + 2147483648
5017h	Holding Register	OFFSET_CONST	32-bit	Increment - the offset to which it is necessary to move, the value is changed by the controller during operation.
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 for saving 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).

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, 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 maximum operating current in the motor winding – from 1000 mA to 20000 mA

500Ah – **HOLD_CURRENT** – holding current in stop mode (from 0 to 1000 mA)

500Bh – **SPEED** – target rotation speed for Modbus control (from 35 to 4105 rpm).

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 rps2, a value of 1000 corresponds to 5000 rps2.

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

- 1 – forward rotation
- 2 – backward rotation.

500Fh – **N_POLE** – number of the motor poles (from 1 to 12).

5010h – **USE_EXTERN_SPEED** – setting the rotation speed setting mode, possible values:

- 1 – commands via Modbus
- 2 – external analog signal at the input SPEED

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, speed.

5013h and 5014h – **PRESSED_INPUTS_EXTERN** and **WAITED_INPUTS_EXTERN** – minimum time of a positive and a negative parts 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 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, the value is changed by the controller during operation.

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 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 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

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.

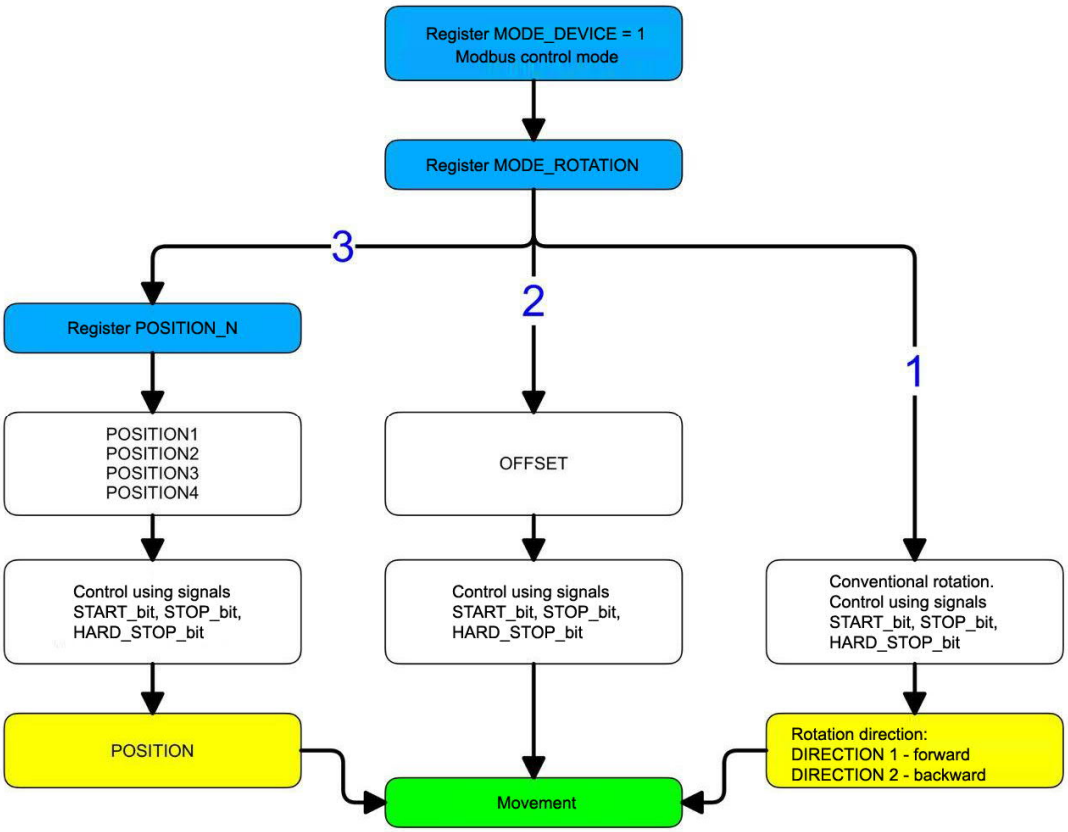

Fig.6. Flow diagram for selecting the operating mode when controlling the drive via Modbus

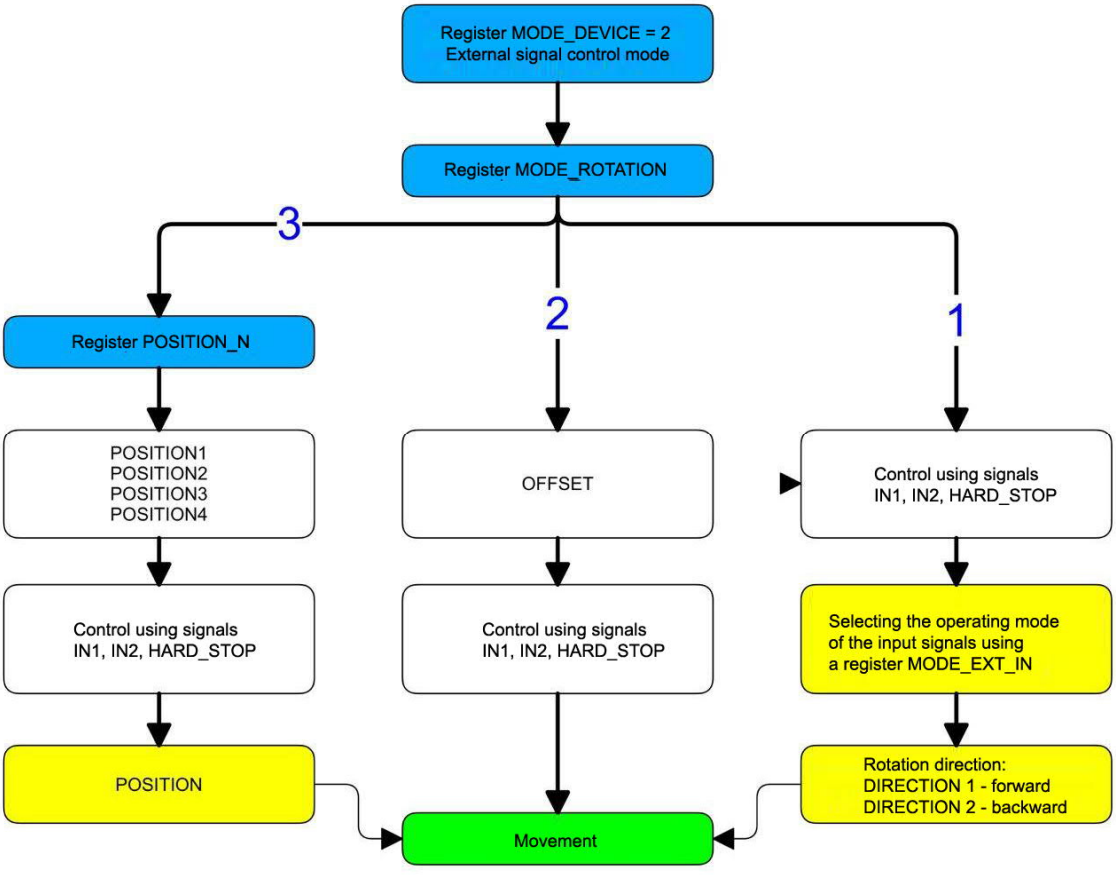

Fig.7. Flow diagram for selecting the operating mode when control the drive using external signals

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 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 instruction
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 instruction

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 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.

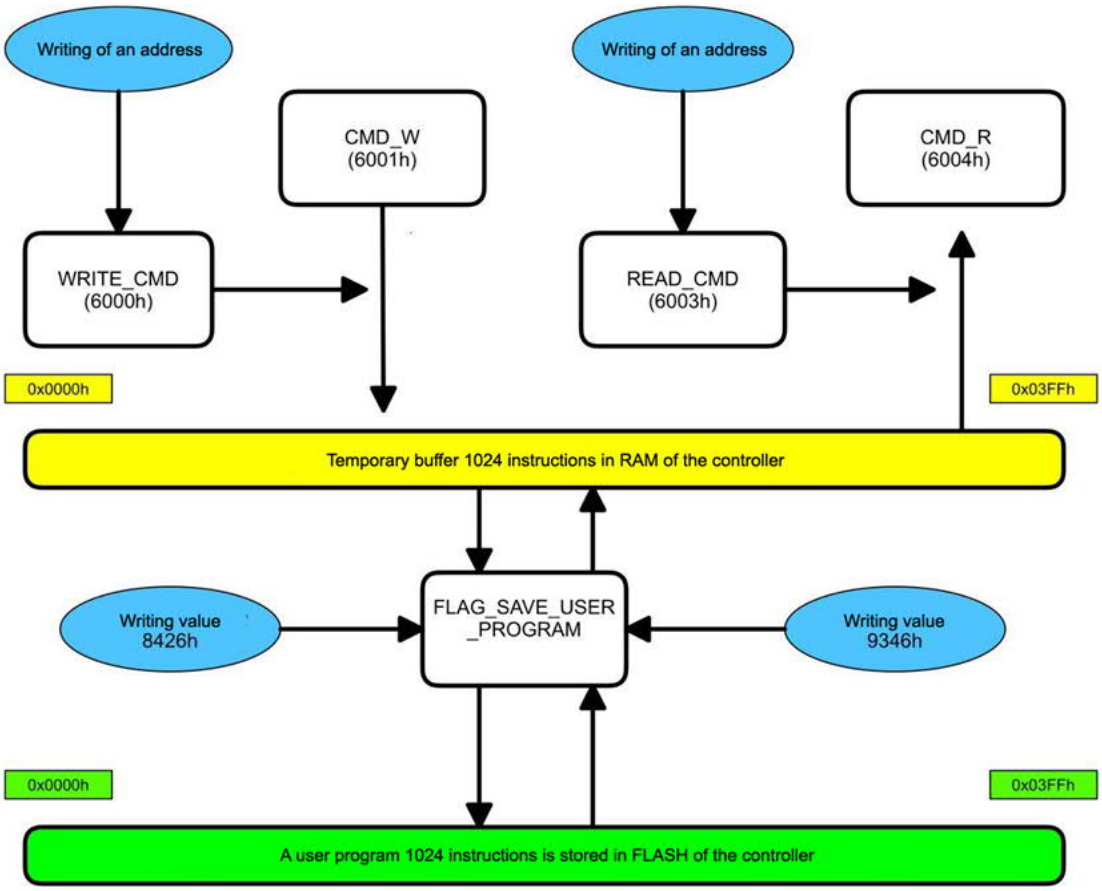

Fig.8. Flow diagram of the procedures for reading and writing a user program.

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

System registers AX_REG..FX_REG (value range 0..65535) are intended for temporary storage of data during executing of a user program. PC_REG is a pointer to the currently executing user program instruction. More detailed description in the 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 indentifier
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 B105-20ModBus the values are:

- HW_MAJOR 1000
- 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.

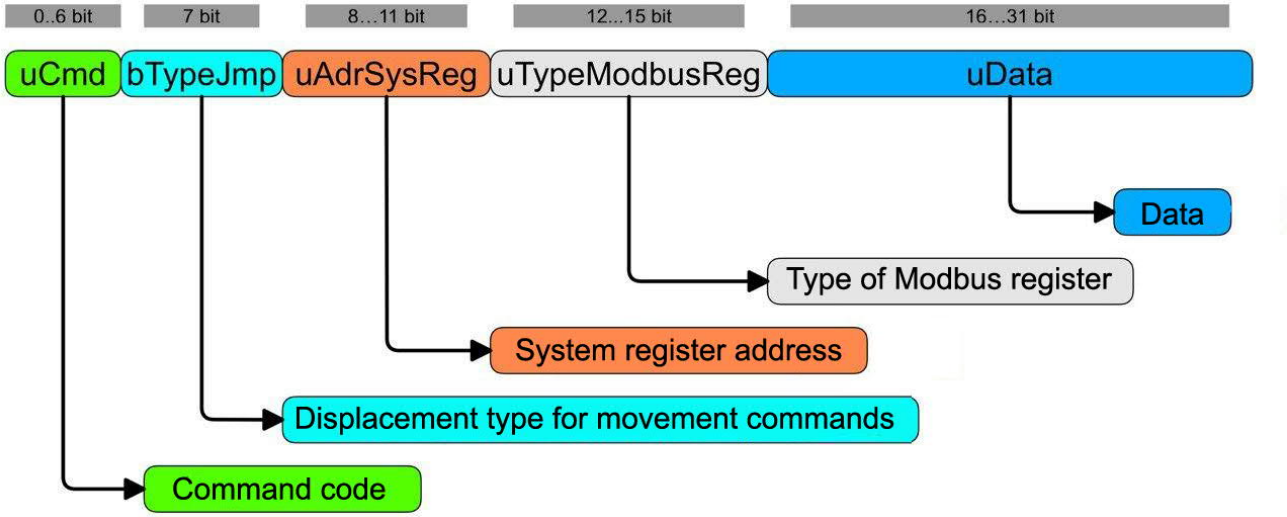

Fig. 9. Structure of a user program instruction

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.

uCmd		bTypeJmp	uAdrSysReg	uTypeModbusReg	uData
0x00	CMD_STOP_PROGRAM	-	-	-	-
0x01	CMD_SET_SYSTEM_REG	-	System register address (0..5)	-	Constant 0..65535
0x02	CMD_WRITE_REG_MODBUS	-	System register address (0..5)	0 – Discrete Inputs 1 – Coils 2 – Inputs 3 – Holding registers	ModBUS register address 0..65535
0x03	CMD_READ_REG_MODBUS	-			
0x04	CMD_DELAY	-	-	-	Pause duration, ms 0..65535
0x05	CMD_JMP	0 – absolute value 1 – relative value	-	-	Move address 0..1024 or offset +-1024
0x06	CMD_JMP_AX_PARI_BX		-	-	
0x07	CMD_JMP_AX_NOPARI_BX		-	-	
0x08	CMD_JMP_AX_MORE_BX		-	-	
0x09	CMD_JMP_AX_LESS_BX		-	-	
0x0A	CMD_CALL	-	-	-	Subroutine address 0..1024
0x0B	CMD_RETURN	-	-	-	-
0x0C	CMD_FOR	-	-	-	Cycle length and number of cycles
0x0D	CMD_FULL_STOP_PROGRAM	-	-	-	-

CMD_STOP_PROGRAM – (command code 0x00) – stop executing of a user program, without exiting the user program execution mode. At the end of the program execution, all registers and state 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) - stopping the execution of the user program and exiting the program operation mode. At the end of the program execution, all registers and the state of the motor return to their original values, the motor stops.

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

CMD_WRITE_REG_MODBUS – (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_READ_REG_MODBUS – (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_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.

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 will continue executing the last task, i.e. movement will continue when the program has 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.

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 5 s. After that, the controller parameters will be reset to the factory values.