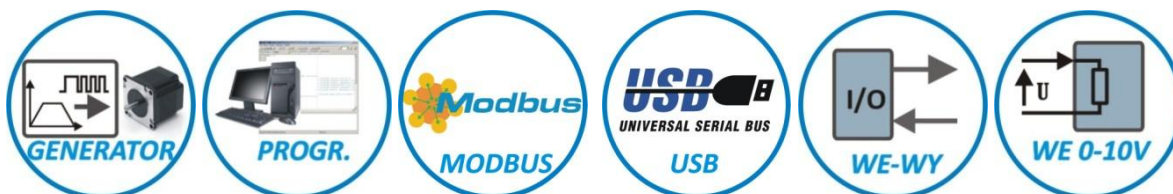


MIC488

USER MANUAL



**Programmable motion
trajectory controller for 4 axis**



P.P.H. WObit E.K.J. Ober s.c.
62-045 Pniewy, Dęboryce 16
tel. 61 22 27 422, fax. 61 22 27 439
e-mail: wobit@wobit.com.pl
www.wobit.com.pl

Thank you for selecting our product!

This instruction will help you at correct service and accurate exploitation of described device.

Information included in this instruction were prepared with high attention by our specialists and is description of the product. Based on the information should not be inferred a certain features or suitability for a particular application. This information does not release the user from the obligation of own judgment and verification. P.P.H. WObit E.K.J. Ober s.c. reserves the right to make changes without prior notice.

-
- Please read instructions below carefully and adhere to its recommendation
 - Please pay special attention to the following characters:



CAUTION!

Not adhere to instruction can cause damage or impede the use of hardware or software.



CAUTION!

The warranty does not cover mechanical or electrical damages caused by overvoltage, short circuit and fault or break down caused by defective exploitation of the user/purchaser.

Contents

1.	Safety and assembly rules.....	4
1.1	Safety rules.....	4
1.2	Assembly recommendation.....	4
2.	Device description	5
2.1	Designation	5
2.2	Functions	6
3.	Equipment description.....	7
3.1	Connectors and indicating lamps layout	7
3.2	Power supply	8
3.3	Universal inputs IN1...IN8	8
3.4	Universal inputs IN9...IN22 / for encoders	8
3.5	Universal outputs OUT1...OUT8	9
3.6	Outputs for M1...M4.....	10
3.7	Connection example.....	10
4.	Software MIC488-PC.....	11
4.1	Connection MIC488 with PC via USB.....	11
4.2	Program description.....	12
5.	Driver configuration.....	13
5.1	Initial information – motion parameters	13
5.2	Drives configuration	14
5.2.1	Homing and motion limiting mode.....	16
5.2.2.	Position control with encoder	17
5.2.3.	Configuration example	17
5.2.4.	Control of drive status	19
5.3.	Digital inputs configuration	19
5.4.	Analog inputs configuration	20
5.5.	RS232/RS485 communication configuration.....	20
6.	Manual control and diagnostics	21
6.1	Drives manual control	21
6.2.	Diagnostics	22
6.3	Errors signalization	22
7.	Driver programming	23
7.1	Introduction	23
7.2.	WBCprog program description	24
7.2.1	Main window	24
7.2.2	Saving and opening a project	25
7.2.3	Quick commands menu	25
7.2.4	Position table	26
7.2.5	Sending file to the driver	26
7.2.6	Running and testing of the program	27
7.3.	WBL language description	28
8.	Program example in WBCprog	30
8.1	Use of inputs / outputs.....	30
8.2.	Read out of analog inputs (0-10V).....	31
8.3.	Control of drives.....	31
8.3.1	Linear interpolation.....	32
8.3.2	Circular interpolation	33
8.4.	Readout / Save of encoder position	34
8.5.	Timers.....	35
8.6.	Mathematical operations and variable	35
8.7	Interrupts	35
8.8	Program example	36
9	List of commands and registers.....	37
7	MODBUS communication	39
8	Record of changes.....	39
9	Technical parameters.....	40
10	Declaration of conformity.....	42

1. Safety and assembly rules

1.1 Safety rules

- Prior to first start-up of the device please refer to this manual and keep it for further use.
- Provide appropriate working conditions in compliance with the device specification (e.g.: power supply voltage, temperature, maximum current consumption).
- Protect inside of the device from any liquids or elements – it can cause electric shock and damage of the device.
- Basic features which knowledge and use will provide safe use consonant with its designation will be demonstrate on the device or in this manual.
- The device with its parts is manufactured in way to provide its safe mounting and connection.
- The device is designed and manufactured as to conform to the principles of protection against the threats mentioned above provided that the device is used in a manner consistent with its purpose and that it is properly maintained.
- The device can cause interference of sensitive radio and television devices in nearby.

1.2 Assembly recommendation

It is recommended to follow measures described below to prevent any possible interruptions of the device operation:

- Do not power the driver on the same line as the device without a corresponding high power line filters (drivers/servo motors).
- Minimize influence of external interference.
- **To minimize noises** please use **screening** of the supply, sensor and signal cables or use **twisted pair cables** (separate kink for A and B phase). It is recommended to use a **ferrite bead** assumed on the motor wire in close to the driver.
- **Please avoid** leading signal cables (**CLK, DIR, EN**) **parallel or in close to electrical and power wire of the motor**; signal cables should be possibly short.

2. Device description

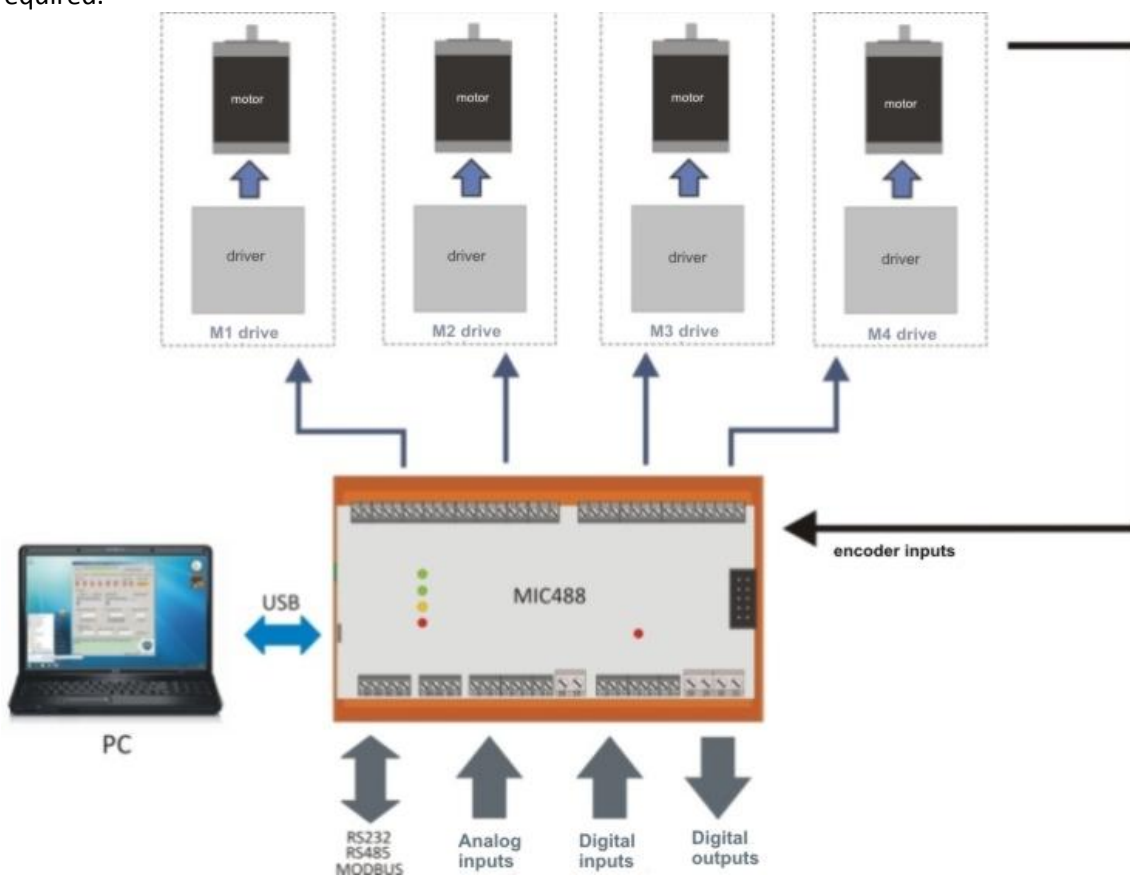
2.1 Designation

MIC488 is a programmable 4-axle driver. Its main function is to control 4 stepper or servo motors in CLK/DIR mode.

The controller allows to connect incremental encoder for master position control. Except of universal I/O and two 0-10 V inputs the driver is equipped with RS232, RS485 interfaces operating in MODBUS (slave) protocol which enables connection with e.g. HMI panels.

Dedicated software allows easy configuration of motors motion trajectory and to create programs for control of driver, outputs and respond on inputs state or values of communications variable. Creation of motion programs is made intuitively by text commands.

MIC488 can replace a traditional programmable driver (PLC) in applications where precise control of several drives is required.



Features:

- Control of up to 4 drives by CLK – DIRECTION – ENABLE outputs
- Linear and circular interpolation
- Possibility of execution of motion programs from driver memory (6 programs x 1000 commands)
- 20 digital inputs (8 opt insulated)
- Option of connection up to 4 incremental encoders for master position control
- 8 transistor outputs
- 2 analog inputs 0-10 V
- 2 COM ports (RS232, RS485)
- MODBUS-RTU communication
- USB connector for programming

- Power supply 12...26 VDC, current consumption 50 mA@24 V
- Intuitive software for driver configuration and programming

2.2 Functions

Main function of MIC488 driver is control of 4 drives in CLK/DIR mode. The controller is generating motion trajectory which includes ramp (acceleration, constant velocity, braking) to achieve smooth motion without lose of position.

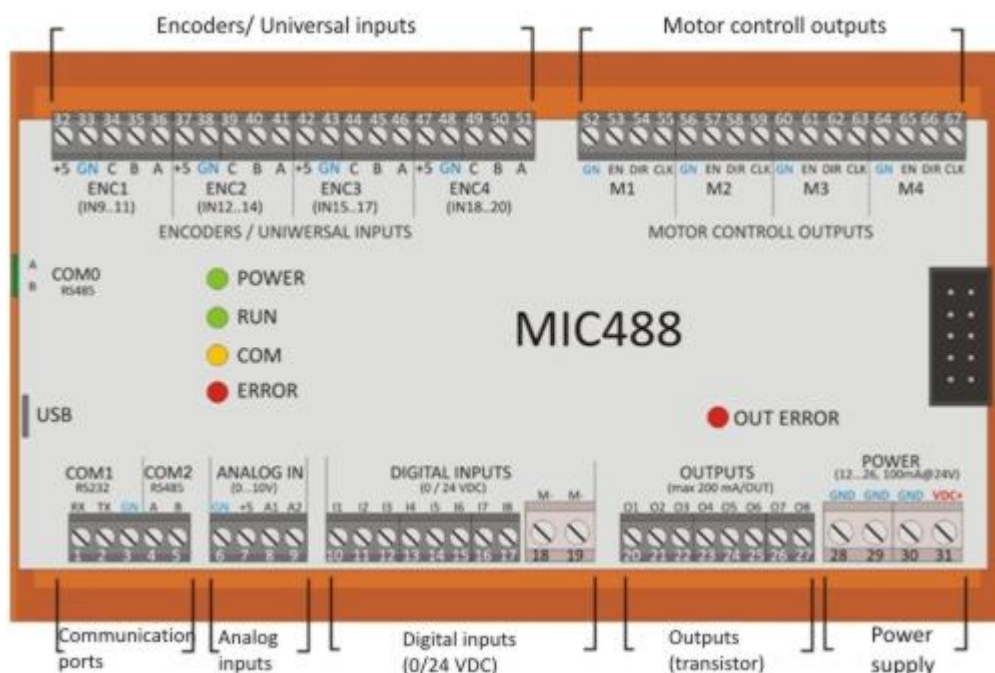
It is also possible to connect encoders for master control of motor position and for its correction by elimination errors of motor positioning. It is useful in case of control of stepper motors which sometimes can lose position. Encoder also gives information of mechanical stop of the drive.

MIC488 is equipped with universal inputs and outputs useful for motion program execution for controlling and responding for signal from external devices.

- **Drives control:**
 - Precise control of up to 4 drives
 - Automatic conversion of motion units from pulses into e.g. mm.
 - Control of motion ramp (acceleration, constant velocity, braking).
 - Set velocity mode (absolute or relative).
 - Set position mode (absolute or relative).
 - Homing mode based on proximity sensors/ encoder.
 - Master position control mode with use of encoder.
- **Control of external devices by digital outputs**
- **Respond on external signals:**
 - Digital inputs for signals from proximity sensors, encoders and controlling signals.
 - Communication by MODBUS-RTU protocol for direct control of driver and driver functions and access to memory used by programs.
 - Readout pulses from encoders.
- **Programs execution form driver memory:**
 - Option of programming up to 6 independent programs consisting of 1000 commands each.
 - Option of programming up to 8 independent „data banks” including 225 defined positions for each of 4 drives.
 - Simple commands executing free motion functions of selected drive.
 - Function of time delay, waiting for inputs, jumps and conditions.
 - Option of condition nesting.
 - Mathematic functions (addition, subtraction, multiplication, division).
 - Floating point support.
 - Access to user memory by MODBUS registers.
- **Programming of the driver is made by MIC488-PC application which allows to:**
 - Configuration of selected drives.
 - Manual motion setting for driver.
 - Fast position setting for driver (JOG mode).
 - Readout of drive operation state, current position and velocity.
 - Preview of driver inputs/outputs.
 - Preview of value in user MODBUS registers.
 - Adding of current drive position to position table.
 - Creating motion programs and its testing (preview of currently executed program line).

3. Equipment description

3.1 Connectors and indicating lamps layout



Picture. 1 Description of connectors and indicating lamps MIC488 driver.

No.	Name	Description	
1	RX	Data receiving	RS232 Interface
2	TX	Transmitting	
3	GN	Ground	
4	A	Signal +	RS485 Interface
5	B	Signal -	
6	GN	Ground	Analog inputs
7	5V	Output +5V (max. 100mA)	
8, 9	A1,A2	Analog inputs 0...10V	Inputs IN1..IN9
10...17	I1..I8	Universal inputs IN1...IN8	
18, 19	M-	Ground for inputs IN1..IN8	
20...27	O1..O8	Transistor outputs (max. 200mA/output)	
28, 29, 30	GND (GN)	Ground	Driver power supply
31	VDC+	Power supply 12...26 VDC	
32, 37, 42, 47	+5	Encoder power supply output +5V (max. 75mA/output)	Power supply and encoder inputs ENC1..ENC4
33, 38, 43, 48	GN	Ground	
34	C (IN9)	Input of encoder C channel 1 / IN9 input	
35	B (IN10)	Input of encoder B channel 1 / IN10 input	
36	A (IN11)	Input of encoder A channel 1 / IN11 input	
...			
52, 56, 60, 64	GN	Ground	Driver control outputs M1..M4
53, 57, 61, 65	EN	Output of driver operations enable signal (ENABLE)	
54, 58, 62, 66	DIR	Output of direction signal (DIR)	
55, 59, 64, 67	CLK	Output of step signal (CLK)	

	POWER	Signalization of driver power supply
	RUN	Signalization of program execution from driver memory
	COM	Signalization of MODBUS communication
	ERROR	Signalization of driver error

3.2 Power supply

Driver power supply

Driver can be supplied by 12...26 V DC voltage. For 24 V voltage supply current consumption is about 50 mA. Power supply should be connect to VDC+ clamps and GND (31, 30).

In case of use transistor outputs you should take in consideration current consumption for outputs.



CAUTION!

Driver power supply should be independent from drives power supply. Additionally at using servo drives supplied from the same grid you should equip it in proper supply filters to eliminate noises that can influence on motor operation.

Output +5V

Driver facilitate +5 V voltage which can be used for supplying encoders (TTL type) or external potentiometers connected to AIN1/AIN2 inputs. Maximal current consumption for all +5 V outputs can't exceed **400 mA**.

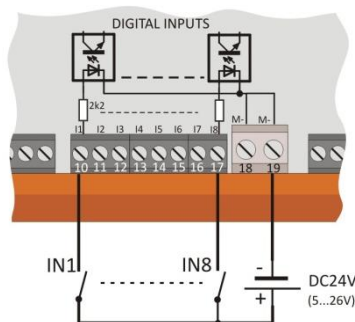


CAUTION!

Do not clench +5V outputs with ground (GND/GN) or power supply. It can cause damage of the driver. Please avoid leading cables with +5V signal near to other signals causing noises.

3.3 Universal inputs IN1...IN8

Universal opt insulated inputs IN1...IN8 allows connection of external controlling signals or signals from proximity sensors for M1...M4 drives. Input is activated by 24 V voltage (min. 5 V, max. 26 V). Common signal (negative) for IN1...IN8 inputs is **M-** inputs (18,19 clamps).



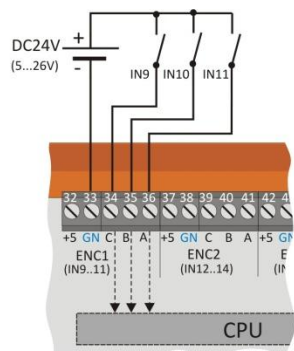
Picture. 2 Opt insulated inputs (IN1 ... IN8)

Parameters:

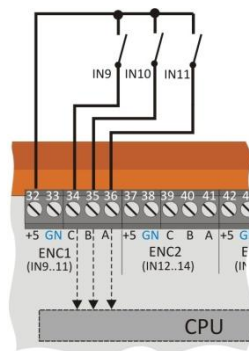
- Opt insulation
- High state: 24 VDC (min 5 V, max. 26 V)
- Low state: < 2 VDC

3.4 Universal inputs IN9...IN22 / for encoders

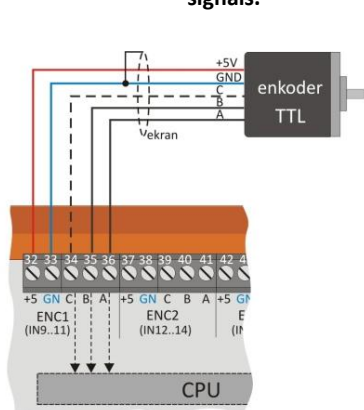
IN9...IN22 inputs can be used as universal inputs or for encoders (for connection with incremental encoders). Inputs have no insulation, common signal (negative) is ground of driver power supply GND (GN). Additionally next to inputs are located voltage outputs +5 V which can be used for direct supply of encoders TTL 5 V type.



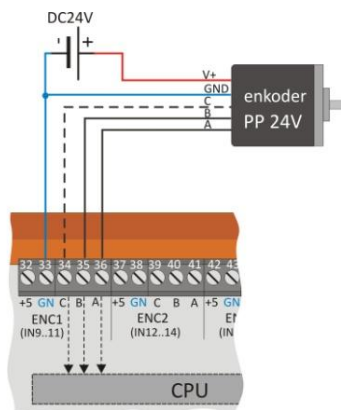
Picture. 3 Example of controlling external inputs by 24 V signals.



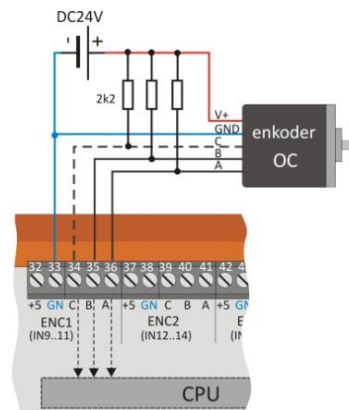
Picture. 4 Example of controlling of inputs with use of +5 V voltage.



Picture. 5 Example of connection encoder with outputs TTL type.



Picture. 6 Example of connection of encoder with outputs Push-Pull type.



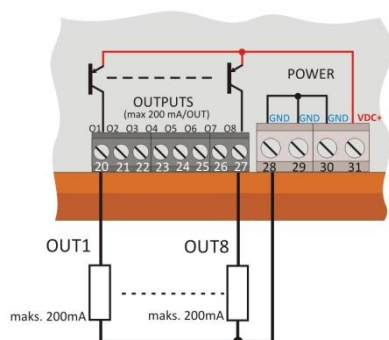
Picture. 7 Example of connection encoder with outputs OC type.



If controlling of drive with position control (from encoder) isn't operate correctly (e.g. drive pass set position) please exchange A and B signals of encoder.

3.5 Universal outputs OUT1...OUT8

Universal outputs OUT1...OUT8 allows control of external executive elements which current consumption not exceed 200 mA. Outputs in active state give driver voltage supply VDC+. Common signal for outputs is driver supply ground **GN**D (28, 29, 30, 31 clamps).



Picture. 8 Outputs OUT1..OUT8

Parameters:

- transistor outputs OC type (open collector)
- load 200 mA/output
- protection from inductive load
- protection from overload/ short circuit (>300 mA)
- high state: voltage supply VDC+

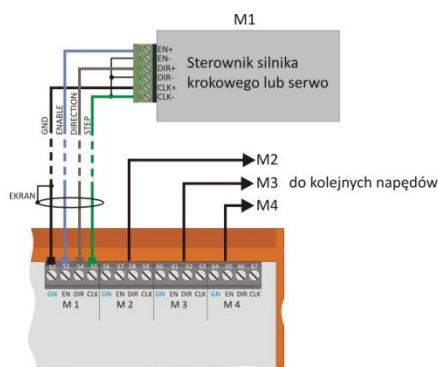


CAUTION!

Outputs are equipped with overload and short circuit protection which is activated after exceeding 300 mA current by some of outputs. Activation of protection cause turning off of all outputs and lighting red diode OUT ERROR. To restore output operations please remove reason of overload and reset the device.

3.6 Outputs for M1...M4

Outputs M1...M4 (signals: EN – enable, DIR – direction, CLK – step) are used for control of stepper motors or servo drives in CLK-DIR mode. Cables shouldn't be lead in nearby of motor cables or supply cables. It is recommended to shield cables.



Picture. 9 Outputs M1..M4 for drives control

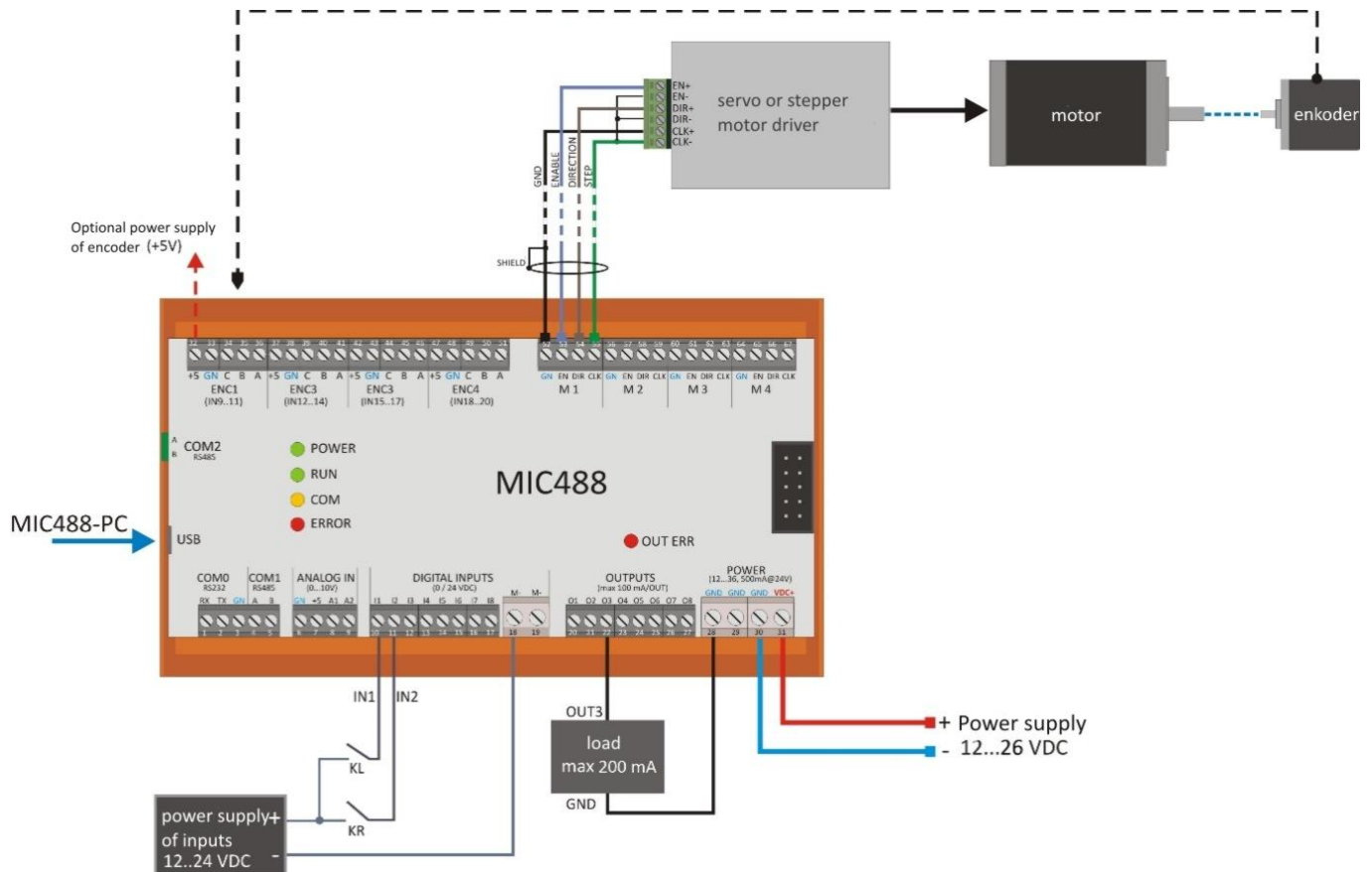
Parameters:

- High state 5 V, low state 0 V, max. 20 mA
- CLK signal (step) – max. frequency 64 kHz, pulse width 10 μ s

3.7 Connection example

On picture below is presented signals connection example to MIC488. The driver is controlling one M1 drive. Hash mark indicates optional connection of incremental encoder for additional position control (for M1 drive encoder should be connected to ENC1 encoder channel).

Furthermore was used two inputs IN1 and IN2 for driver limit signal (KL and KR) and one OUT3 output for activation of an external circuit (e.g. signaling lamp, relay etc.).



Picture. 10 Example of connection signals to controller.

4. Software MIC488-PC

4.1 Connection MIC488 with PC via USB

Configuration and programming of the driver is made by MIC488-PC application. Driver can be connected with PC using USB or RS232 (connection to COM1 port of the driver). It is recommended to use RS232 connection at applications where strong noises can occur (e.g. controlling of AC servomotors).

USB connection

The driver should be connected with PC by USB cable type A – B mini. After connecting with the computer it is permitted to turn on the supply of the driver and to run MIC488-PC software. Correct communication will be signaled by information in program upper window.



CAUTION!

- 1) **USB connection should be done always before turning on driver power supply.**
- 2) **USB connection is susceptible to noises in supply grid and to electromagnetic noises, existing in industrial conditions. In case of problems with communication it is recommended to use additional protection elements like:**
 - Use of power line filter,
 - Use a high quality USB cable with length < 1,5m equipped with ferrite beads
 - Use of opt insulated USB HUBs on PC side

At higher noises can occur that communication won't be possible.



MIC488 is communicating using USB port (1.1, 2.0).

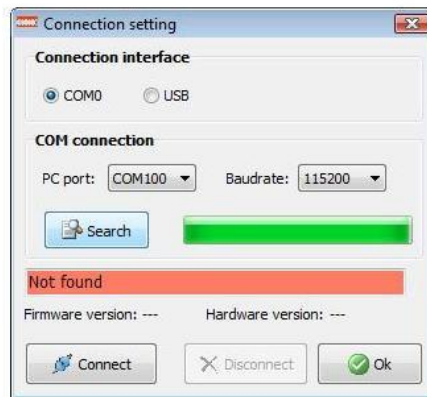
There are known compatibility issues of USB 3.0 port (blue socket) at Windows 7 system while communication with USB HID devices. In case of problems with communication please connect driver to

Serial connection RS485 (COM0)

COM0 port of MIC488 driver allows connection with PC using USB<->RS485 converter.

Signals from RS485 converter (A, B) should be connected to A and B signals of COM0 port of the driver.

To establish connection at MIC488-PC program please select on toolbar **Connection** -> **Settings**. At open window select **COM0** connection. If number COM port on PC side is known please select this port and set baudrate (57600) then press button **Connect**. If number is unknown please press **Search** button.



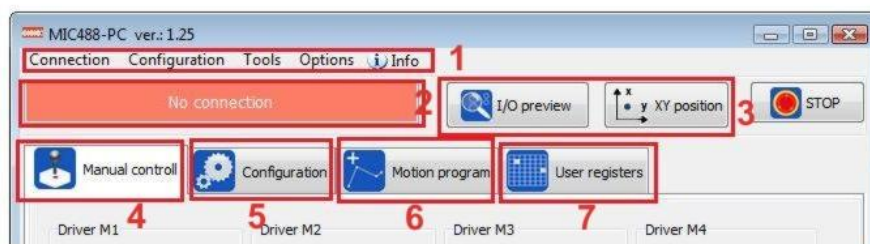
Picture. 1 Window of communication settings with PC.



MIC488-PC program should be run as admin (by right-click on program icon and selecting „Run as administrator” option). In other case program may not detect installed COM ports.

4.2 Program description

Basic program functions are available at main window tabs showed below:



Picture. 2 Toolbar.

- 1) Toolbar
- 2) Signalization of connection with the driver.
- 3) Running of diagnostic window with I/O preview and position of M1/M2 drives
- 4) Tab for manual control of drives
- 5) Tab with driver settings
- 6) Tab for creating motion programs
- 7) Tab with preview of user registers

Toolbar:



- Connection – configuration of driver connection with program
- Configuration – record and readout of driver settings from a file. Restore of factory settings of the driver.
- Add-ons – *I/O preview* [this function is also available at button (3)], *Preview of X/Y* [this function is also available at button (3)], *JOG control* (this function is also available at „Manual control”),
- Options – general settings of MIC488-PC application.
- Info – information about MIC488-PC program.

5. Driver configuration

5.1 Initial information – motion parameters

While setting velocity or position for drives the driver is generating ramp (acceleration/braking) to achieve smooth motion. There are considered parameters as below:

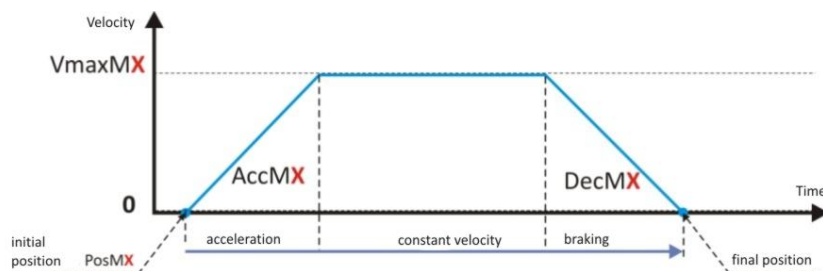
Parameter	Description	Units
VmaxMX	Max velocity at set position mode. While motion to new position the velocity will not be exceeded.	[j/s]
AccMX	Acceleration for speeding up at set position mode. Acceleration for speeding up and braking at set velocity mode.	[j/s ²]
DecMX	Acceleration for braking at set position mode.	[j/s ²]

where: X – means number of individual driver (1...4), j – motion unit dependent on drive configuration (e.g.. rpm etc.)

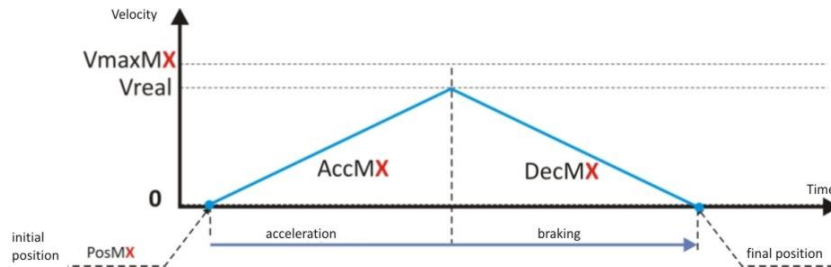
Ramp for set position

- **Speeding up:** start of motion from 0 velocity up to velocity set by VmaxMX parameter (max. velocity for position) with AccMX acceleration.
- **Constant velocity** VmaxMX (only when target distance will be longer than distance need for speed up and braking with set AccMX and DecMX parameters).
- **Braking:** decrease of velocity to 0 with DecMX delay as far as reaching target position.

While setting of position in dependence on acceleration value for speeding up and braking the driver can reach max. velocity if total distance need for speeding up and braking will be shorter than left set distance. Picture. 11 shows case of reaching of max. velocity whereas Picture. 12 shows ramp with limited velocity to Vreal velocity.



Picture. 11 Example of position ramp –Vmax velocity possible to achieve.

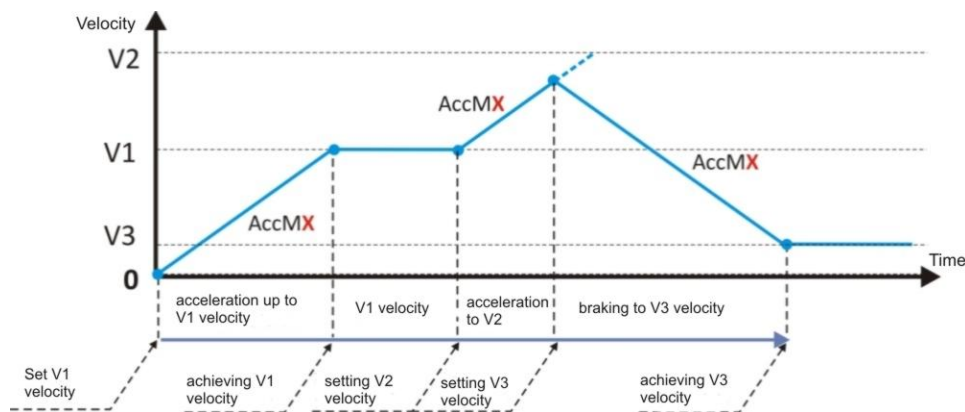


Picture. 12 Example of position ramp – Vmax velocity not possible to achieve.

Remarks:

- Setting of position cause motion always from zero velocity.
- Setting of new position during motion will cause stopping of the driver and motion from zero velocity. **This situation can cause loose of position for stepper motor while operating without encoder.**
- Setting of position during motion to a position will cause setting a new velocity with ramp AccMX (for braking or speeding up to set velocity).

Ramp for set velocity



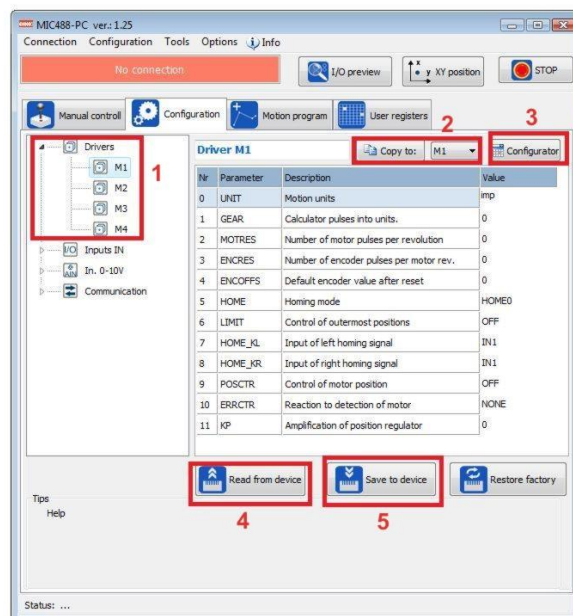
Picture. 13 Example of ramp for set velocity.

Remarks:

- Ramp for velocity(for speeding up and braking) is made always with AccMX parameter.
- **During motion it is not possible to change AccMX parameter.** New AccMX value is considered while motion from zero velocity.
- Setting of new velocity during motion will cause reaching new velocity in view of AccMX ramp.

5.2 Drives configuration

Configuration of drives is made at tab **Configuration-> Drives**.




Picture. 14 Configuration window – drives settings.

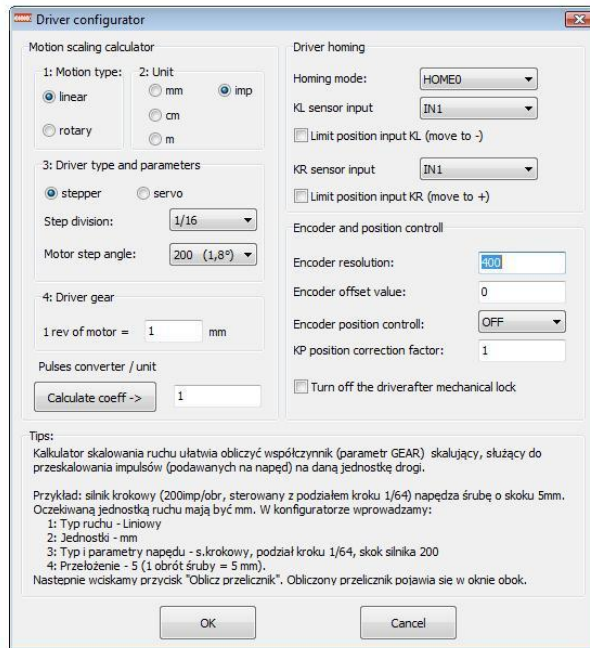
- 1) Drive selection for configuration (M1... M4).
- 2) Copying of settings between drives.
- 3) Configurator facilitating setting drives parameters.
- 4) Readout of setting from the driver (concerns all device settings).
- 5) Record of settings to the driver (concerns all device settings).

Parameters for drive configuration:

No.	Parameter	Description
0	UNIT	Setting of motion units displayed at program (has no influence on calculated values).
1	GEAR	Calculator pulses into unit. It defines quantity of pulses, which driver has to generate to drive execute one motion unit.
2	MOTRES	Quantity of motor pulses per full mechanical or electrical reverse (for linear motors). Parameter required for operations with encoder.
3	ENCRES	Encoder resolution Defines quantity of pulses from encoder per full revolution. CAUTION: Entered resolution value should be multiplied by 4 (consideration of square wave signal)
4	ENCOFFS	Default encoder value after reset.
5	HOME	Defines drive homing mode (precise description at 5.2.1 chapter)
6	LIMIT	Settings consider control of end drive positions by proximity sensors: OFF – no control of end position KL – control of left end position (for drive motion towards decreasing of position) KR – control of right end position (for drive motion towards increasing position) KL + KR – control of both end positions
7	HOME_KL	Setting left input for homing sensor (also for position control).
8	HOME_KR	Setting right input for homing sensor (also for position control).
9	POSCTR	Position control mode from encoder: OFF – position control from encoder turned off ENCODER – position control from encoder turned on ECFOLLOW – not available
10	ERRCTR	Reaction to detection stop of drive (only for operation with encoder): OFF – no reaction ERRMODE0 – stop of drive, change of drive status (PosMX = M_POS_ERR) ERRMODE1 – do. +break of executed program ERRMODE2 – do. + turning on OUT8 output
11	KP	Reinforcement of position regulator. It defines reinforcement of position regulator for position control from encoder mode. The bigger value the faster position setting after lost of step. Default 50.

- setting required in case of operation with encoder for drive position control.

For facilitating drive configuration is used window started by  button in which in next steps are given required parameters.



Picture. 15 Drive Configurator window.

After entering parameters please press „**Calculate coefficient ->**” button and then click on **OK** button. After closing window entered parameters will be add to drive settings. To enter settings in driver please press „**Save to device**” button.

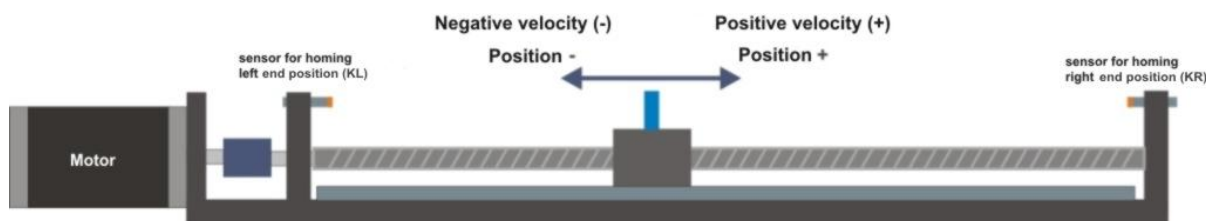


Running of configurator do not cause loading of current drive settings.

5.2.1 Homing and motion limiting mode

Driver allows setting two input signals (for each drive independently) which can be used for homing and/or for limiting end positions during motion.

Signals named as **KL** (homing or motion limiting during motion towards negative direction (decreasing of position) and **KR** (in opposite direction).



Picture. 16 End signals and motion directions of the drive.

Homing of the drive can be made in one of modes listed below, set in **HOME** setting:

- **HOME0**: Simple homing. Stop of the drive is made instantly after reaching proximity sensor.
- **HOME1**: Precise homing mode 1. Drive is reaching proximity sensor (mounted in end position), stops and then slowly move back up to signal decline from sensor (back in front of sensor).
- **HOME2**: Precise homing mode 2. Drive is reaching proximity sensor (mounted in end position), stops and then slowly forward move until signal decline from sensor (drive across sensor).

- **HOME3:** Encoder homing mode 1. Drive is reaching a mechanical blockage then stops, and then slowly move back up to detecting signal from encoder C channel (INDEX).
- **HOME4:** Encoder homing mode 2. Drive is reaching proximity sensor, stops and then slowly move back up to detecting signal from encoder C channel (INDEX).

For HOME1 up to HOME4 mode reverse motion is made with 20% of homing velocity.

Limiting of position by end signal is configured by **LIMIT** setting. The drive stops when it detects end signal. Then possible is only move in opposite direction. Do not set the same input when two end signals are active (KL+KR mode) – it will cause blocking of drive operation after sensor activation.

It is also possible to limit position using program (LimL and LimR registers are available by Modbus or WBCprog program). By default this registers store maximal range values of positions.

5.2.2. Position control with encoder

MIC488 can control motor position based on external encoder connected to ENC inputs (ENC1...4 for next drives M1...M4). Way of encoder connection is presented at **Błąd! Nie można odnaleźć źródła odwołania.** chapter.

For proper operation encoder with motor it is necessary to enter correct parameters *MOTRES* (it defines number of motor pulses per full revolution) and *ENCRES* (it defines number of encoder pulses per full revolution).

For example:

- For stepper motor with 1,8° step and driver with step resolution 1/64 *MOTRES* will be equal to $200 \cdot 64 = 12800$
- For encoder with 400 resolution *ENCRES* will be equal $4 \cdot 400 = 1600$ (multiply x 4 to respect square wave signal of encoder).

After entering parameters please test drive by setting for example absolute position = 1. Then following situations are possible:

Drive behavior	Description
Execution of full revolution.	Parameters and encoder connection are correct.
Drive executes full revolution then reverse.	Too small encoder resolution was entered (e.g. square wave wasn't taken under consideration) or too high motor resolution.
Drive will execute more than full revolution and stops.	It was entered too high resolution of encoder or too small resolution of motor.
Drive will execute full revolution and will continue operation without stopping.	Incorrectly connected encoder or motor. Please exchange one of motor phases (e.g. A with /A) or encoder channels (A with B)
Drive will execute full revolution but while losing of position its correction is made very slowly.	Please increase value of position regulator <i>KP</i> .

5.2.3. Configuration example

At configuration example as a driver was used stepper motor (with step 1,8° that means **200 step/revolution**), controlled by driver with **1/64** step resolution. Motor is driving a ball screw with lead **5 mm/revolution**. For position control were used two sensors – left as homing sensor and for limiting motion and right as sensor limiting motion on the right side. Homing can be made precisely with reaching of sensor position and leasing up to signal decline from sensor. Control unit is **mm**.



Picture. 17 Exemplary configuration of a drive (operations without encoder).

Configuration (without encoder)

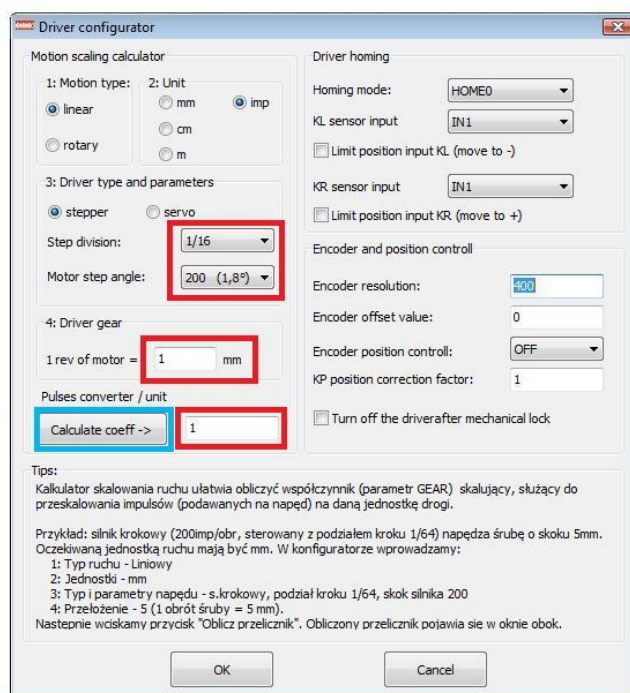
Motor driver is connected to M1 output of MIC488. Sensor KL signal is connected to IN1 input and KR signal to IN2 input. In M1 drive settings (tab **Konfiguracja** -> **Napędy** -> **M1**) (tab **Configuration** -> **Drives** -> **M1**) enter parameters as below:

- (0) UNIT: mm
- (1) GEAR: $64 * 200 / 5 \text{ mm} = 2560$
- (5) HOME: HOME1
- (6) LIMIT: KL + KR
- (7) HOME_KL: IN1
- (8) HOME_KR: IN2

Other parameters are insignificant for operation without encoder.

You can also use drive calculator (by pressing **Configurator** button). After its activation enter necessary parameters:

- 1) Type of motion: **linear**,
- 2) Units: **mm**,
- 3) Type and drive parameters: **stepper motor, step resolution 1/64, motor step 200 (1,8°)**,
- 4) Transmission ratio: **5 mm**.



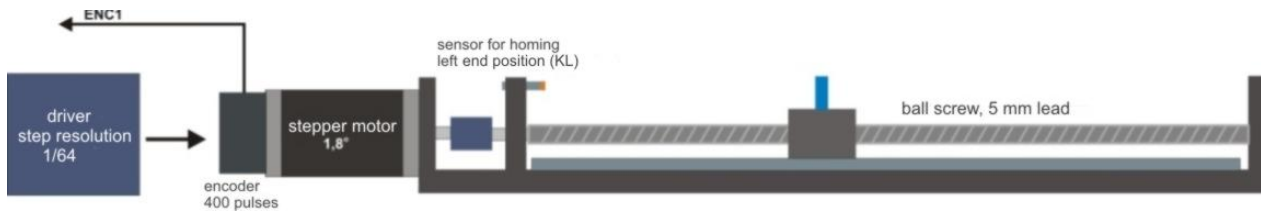
Picture. 18 Drive Configurator window with entered parameters.

Then press „**Calculate coefficient ->**” button. After closing of configurator by „**OK**” button, settings will be enter into window with M1 drive parameters.

After entering settings please press „**Save to device**” button to send settings to the driver.

Configuration (with encoder)

At this case to drive was add an incremental encoder with resolution 400 ppr. There was also deleted right end proximity sensor.



Picture. 19 Exemplary configuration of a drive (operations with encoder).

Please enter parameters as below:

- (0) UNIT: mm
- (1) GEAR: $64 * 200 / 5 \text{ mm} = 2560$
- (2) MOTRES: $64 * 200 = 12800$
- (3) ENCRS: $400 * 4 = 1600$ (x4, because it should be considered encoder square wave signal)
- (4) ENCOFFS: 0 (default position after reset)
- (5) HOME: HOME1
- (6) LIMIT: KL
- (7) HOME_KL: IN1
- (8) HOME_KR: OFF
- (9) POSCTR: ENCODER
- (10) ERRCTR: 1 (it defines driver reaction in case of drive mechanical stop).
- (11) KP: should be set experimentally. Default value is 50.



If controlling of drive with position control (from encoder) is not operate correctly (e.g. drive pass set position) please exchange A with B signals of encoder.

5.2.4. Control of drive status

Drive status can be controlled using:

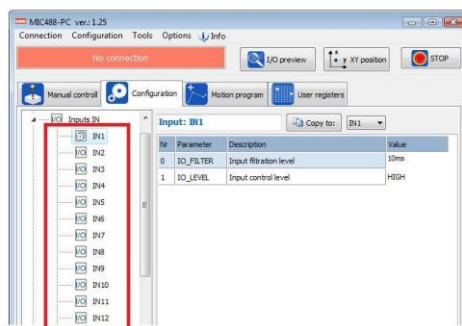
- Information at manual control window,
- MODBUS registers: *M1_STATUS (1010)...* *M4_STATUS (1013)*
- Registers at WBCprog: *StatM1...StatM4*

At table below are presented values of status register and corresponding it drive states.

Register value	Description
0	Drive turned off (signal EN = 0)
1	Drive turned on but not in motion (EN signal active)
2	Drive is in set velocity mode
3	Drive is in motion to set position mode
4	Drive achieved set position
5	Error of achieving set position (for operation with encoder)
6	Drive at homing mode
7	-
8	Drive at position correction mode (for operation with encoder)
9	Drive achieved final L position while motion towards negative direction (by program, or by activation of proximity sensor signal KL)
10	Drive achieved end R position while motion towards positive direction (by program or by activation of proximity sensor signal KR)

5.3. Digital inputs configuration

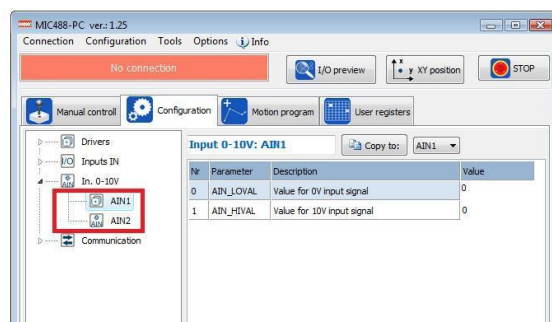
Configuration consider only IN1...IN8 inputs used in program as universal inputs. Settings has no influence on inputs operation in drive position/homing function.



Picture. 20 Inputs settings.

No.	Name	Description
0	IO_FILTER	Level of input filtration. It defines minimal time for giving signal on input to keep it active.
1	IO_LEVEL	It defines what level is a high state for input. HIGH level means that input is active after giving voltage on input. LOW level means that input is active while no signal on input.

5.4. Analog inputs configuration



Picture. 21 Settings of analog inputs.

No.	Name	Description
0	AIN_LOVAL	Value for input signal 0V
1	AIN_HIVAL	Value for input signal 10V

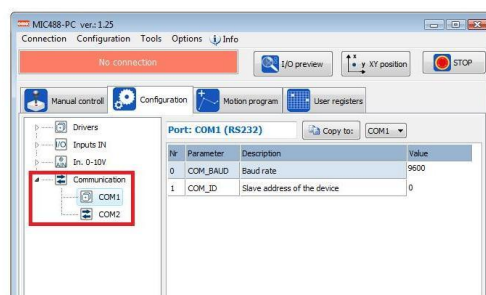
Settings allows to scale voltage input signal 0...10 V into other value which can be used in motion program for e.g. direct control of drive velocity etc.

For example for input voltage 0V AIN value is -25 and for 10V +25. Then should be entered:

AIN_LOVAL: -25

AIN_HIVAL: +25

5.5. RS232/RS485 communication configuration



Picture. 22 Setting of RS232/RS485 communication.

No.	Name	Description
0	COM_BAUD	Transmission baudrate
1	COM_ID	Address (slave) MODBUS

Other communication parameters:

- Stop bits: 1
- Parity: none

COM1 (RS232) and COM2(RS485) ports enables communication with devices using MODBUS-RTU protocol. MIC488 is a SLAVE device. In device settings is possible to change transmission baudrate for each port independently.

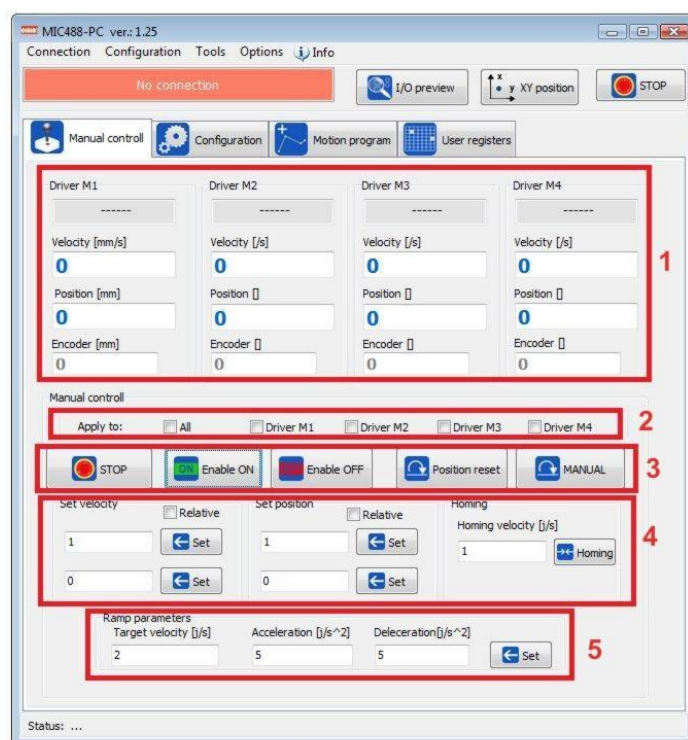


Setting of SLAVE address of the device should be made for COM1 port. It is common for COM2 port. Change of address for COM2 port doesn't matter.

6. Manual control and diagnostics

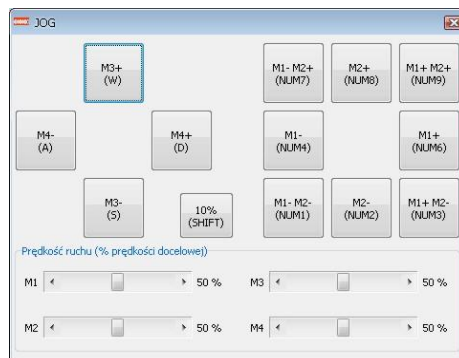
6.1 Drives manual control

Manual control allows testing of drives and setting of motion commands for individual drives.



Picture. 3 Drives manual control window

- 1) Shows information about drives (status, current velocity, current position, encoder position)
- 2) Allows to select drive (drives) to be controlled
- 3) Controlling buttons:
 - ENABLE ON – turning on of a drive (EN outputs).
 - ENABLE OFF – turning off of a drive (EN outputs).
 - Position reset – reset of current position (for 0 value).
 - STOP – stop of a drive.
 - MANUAL – opening a window for controlling in JOG mode.

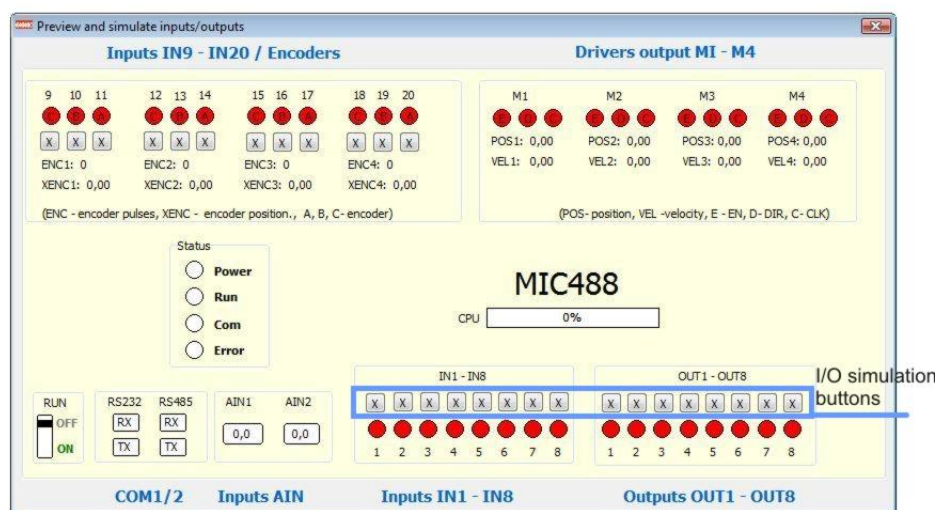


Picture. 23 Drive controlling window in JOG mode.

- 4) **Set velocity** – allows setting of velocity (drive will achieve set velocity). Marking „Relative” option cause increasing of (for negative value decreasing) current velocity by set value.
Set position – allows to set position (drive will achieve set position). Marking „Relative” option cause increasing of (for negative value decreasing) current position by set value.
Homing – cause execution of drive homing with set velocity. Negative velocity value cause homing into left end proximity sensor direction (KL) positive value into right end proximity sensor direction (KR)
- 5) **Motion parameters** – allows to define acceleration and maximal velocity for set values.

6.2. Diagnostics

I/O preview button enables fast diagnostics of the driver.



Picture. 24 Window of preview and I/O simulation.

- - input/output not active (low state)
- - input/output active (high state)

By buttons located above IN1..IN8 inputs and OUT1..OUT8 outputs is possible to simulate its operation. Inputs simulation operates also at homing and end sensors function.

6.3 Errors signalization

Depend on diodes status MIC488 can signalize following errors:

Diode status				Type of error	Deleting error
RUN	COM	ERROR	OUTERR		
-	-	-	ON	Output overload	Reset of power supply

OFF	-	ON	-	Program error	Program restart
OFF	-	Blinks	-	Error of drive position	-
Blinks	-	Blinks	-	Drive position error. Program is stopped while operation.	Program restart
Blinks	Blinks	Blinks	-	Critical error of the driver	After couple of seconds driver will automatically reset.

Program error (ERROR diode turned on) can occur when:

- Record/ readout from non existing register (e.g. *OUT10*, *AccM5*)
- Error of program proof total
- Starting interruption without interruption label

Position error (ERROR diode blinks) can occur when:

- While operation with encoder when drive can't achieve set position during 3 sec.

7. Driver programming

7.1 Introduction

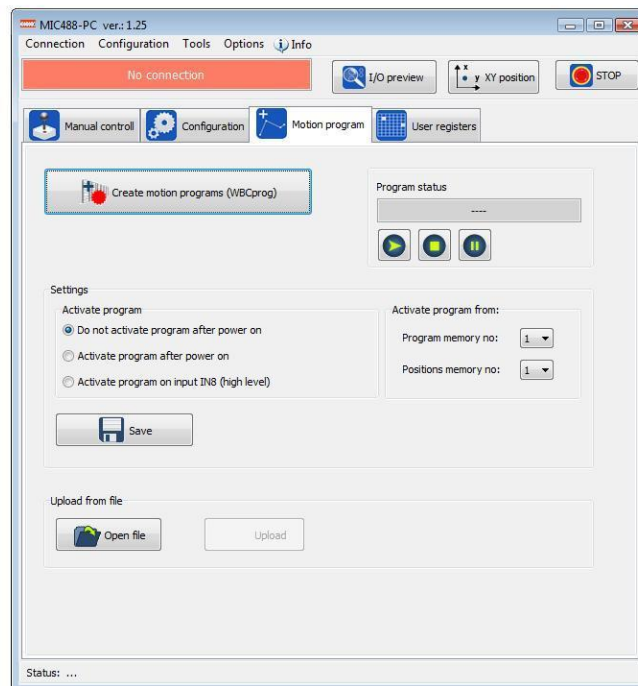
Programming of a driver is made using build-in MIC488-PC **WBCprog** application. Application is activated in „**Motion program**” tab by „**Create motion programs (WBCprog)**” button.

Programming means entering simple commands in text language, named **WBL** (WObit Basic Language), e.g.: „*PABS M1 10*” (motion into absolute position 10 of M1 drive) or „*SET OUT3 = ON*” (turning on O3 output).

This language thanks to simple text commands allows in easy and fast way create programs for MIC488 driver. From level of created program is possible freely control of drives motion, control of universal outputs, reaction on inputs, counting pulses from encoder, time delay functions, simple mathematic operations, operations on variable available through MODBUS etc.

The user can save in driver memory up to 6 individual programs consisting of up to 1000 commands each. It is also possible to save 8 individual position tables consisting of 225 position each (one position in a table contain position for 4 drives M1...M4).

Selected program from driver memory can be run automatically after turning on power supply of a driver or after activation by IN8 input. This configuration is made in window of „**Motion program**” tab.

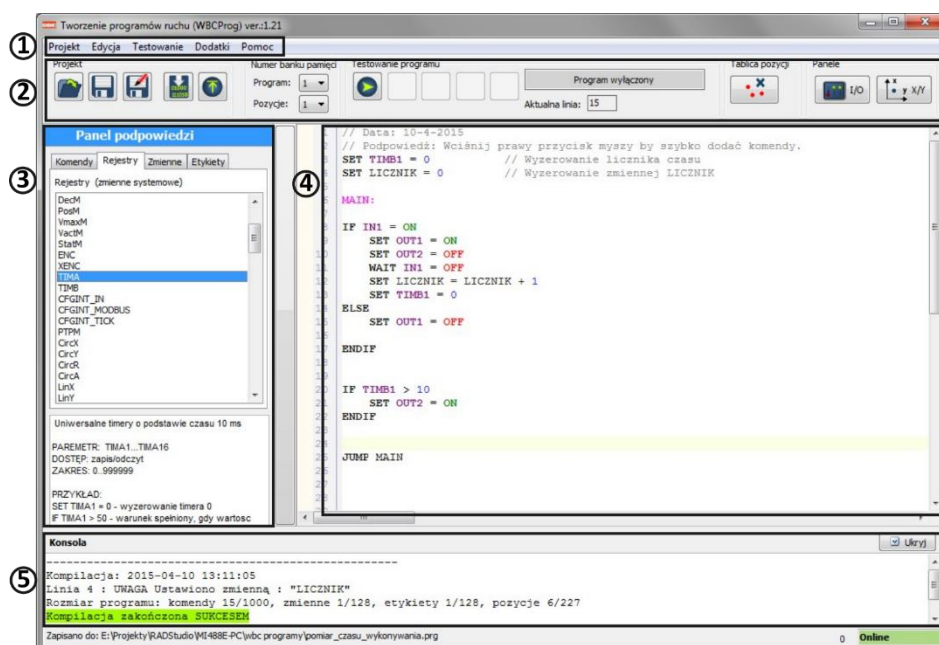


Picture. 25 Window with program run settings from driver memory.

Furthermore it is possible to run selected program and position table by proper MODBUS registers.

7.2. WBCprog program description

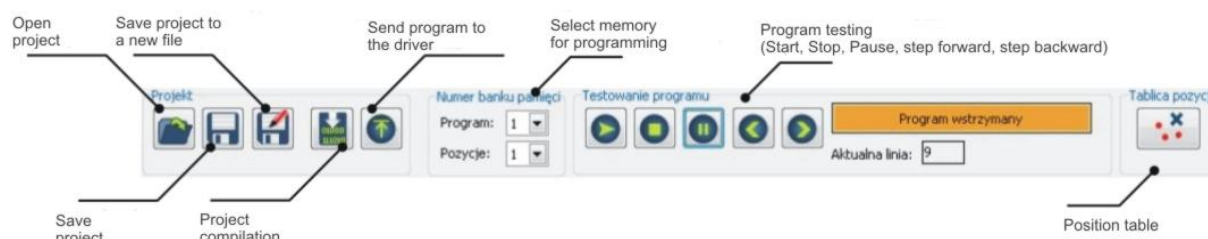
7.2.1 Main window



Picture. 26 Main window of WBCprog program.

- 1) **Toolbar:**
Contains access to all program functions.
- 2) **Shortcut bar:**

Contains shortcuts to the most important program functions. In „*testowanie programu*” part are located buttons allows program running to test its operation. Next to buttons is displayed information about current program status and errors (if occurs).



Picture. 27 Shortcut bar.

3) Help toolbar:

Contains list of all commands WBCprog compiler (**Komendy** and **Rejestry** tab). It also show used own variables used at program (**Zmienne** tab) and labels (**Etykiety** tab).

4) Code editor window:


Text editor for entering commands. It is equipped in function of highlighting of entered commands/variables which facilitate program writing.

5) Information about program compilation window:

Shows information about compiled program (program size etc.) or information about errors.

7.2.2 Saving and opening a project

Zapisz  button cause saving of changes in a project.

Zapisz jako  button cause saving a project to a new file.

While saving a project three files are created (with the same name but different extension):

- Main file with program (.prg extension).
- File with current settings of a driver (.cfg extension).
- File with positions from position table (.csv extension).

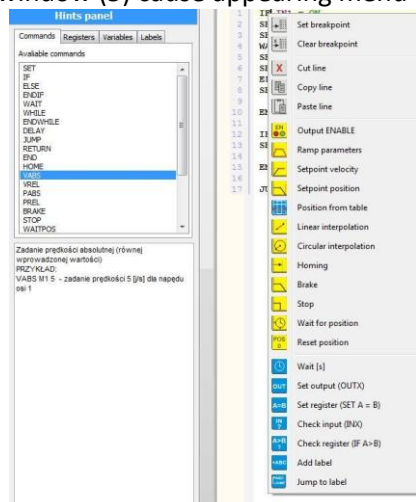
Otwórz  button cause opening a project (please open file with .prg extension).



If in catalogue with project is no configure file (***cfg**) or file with position table (***csv**) program will display prompt no file. Saving a project will cause an automatic creation of this files.

7.2.3 Quick commands menu

Right-click into selected line at editor window (5) cause appearing menu which allows adding selected commands.



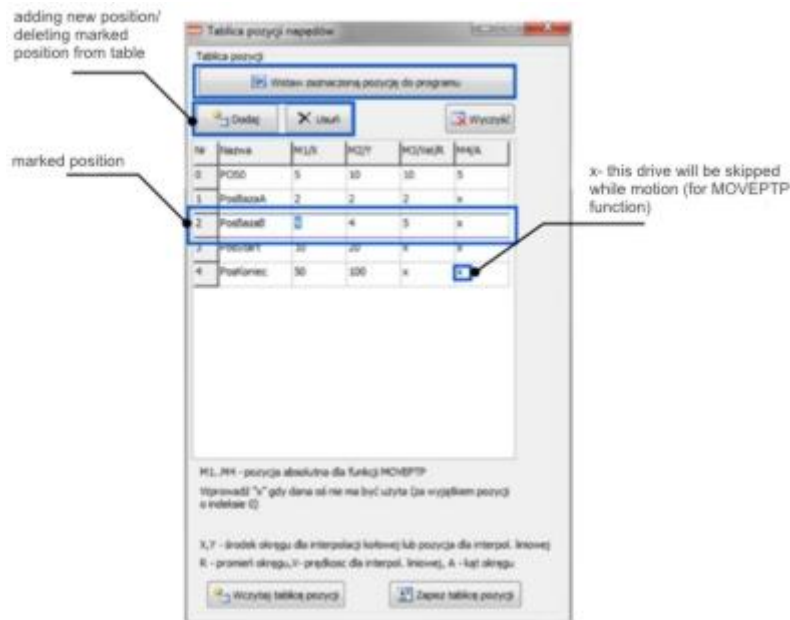
Picture. 29 Quick commands menu.

7.2.4 Position table

It allows to retain up to 227 records, which can be:

- Independent positions for 4 drives
- parameters for linear interpolation (executed by M1 and M2 drives)
- circle parameters for circular interpolation (executed by M1 and M2 drives)

Table is run by button  located on shortcut bar.



Picture. 30 Window „Position table”.


For independent control of particular drives (**MOVEPTP** function) are used **M1...M4** columns corresponding to position of particular drives.

For linear interpolation (**MOVELIN** and **MOVELIN_REL** function) are used **X** and **Y** columns (it defines final point of motion) and **Vel** column defining velocity of linear motion.

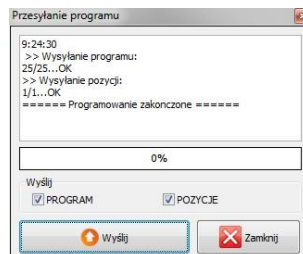
For circular interpolation (**MOVETOCIRC** and **DOCIRC** function) are used **X** and **Y** columns (it defines midpoint of circle) , **R** column (it defines radius of circle) and **A** column (it defines angle of circular motion).

Position table is save while saving a project as *.csv file which can be opened and edited using spreadsheet (e.g. Microsoft Excel program). It can be read in earlier prepared file with positions for position table by „**Read position table**” button.

7.2.5 Sending file to the driver

To program a driver please press **Send to driver**  button. It will open a window where user defines data to be send:

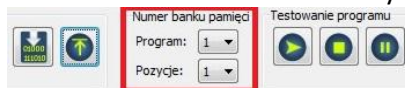
- **PROGRAM** – if we want send a program
- **POSITIONS** – if we want to send position tables




Picture. 4 Sending file to the driver window.






After pressing **Send** button program will be send to the driver.

Program / positions from position table will be saved into driver memory with number set on shortcut bar:

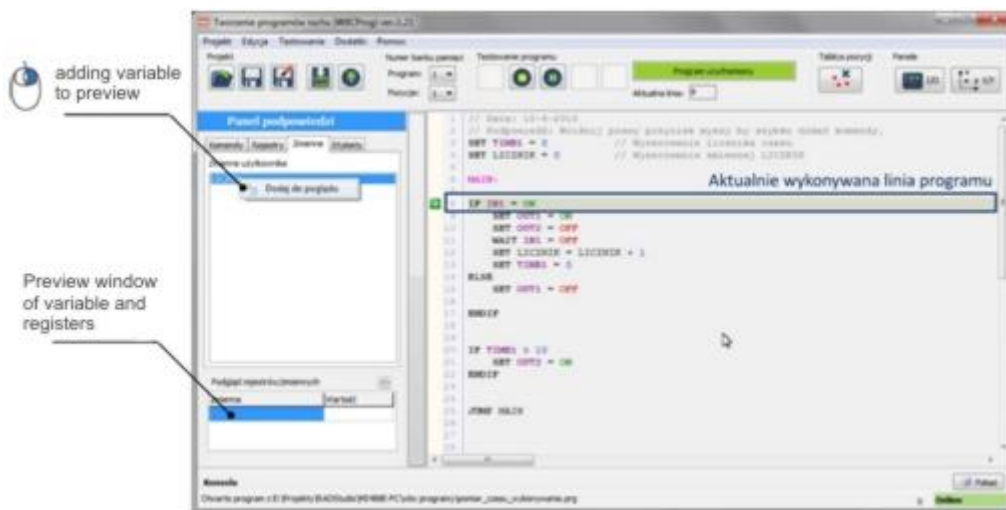


Program before sending to the driver is automatically compiled. There is no necessary to its earlier compilation by  button.

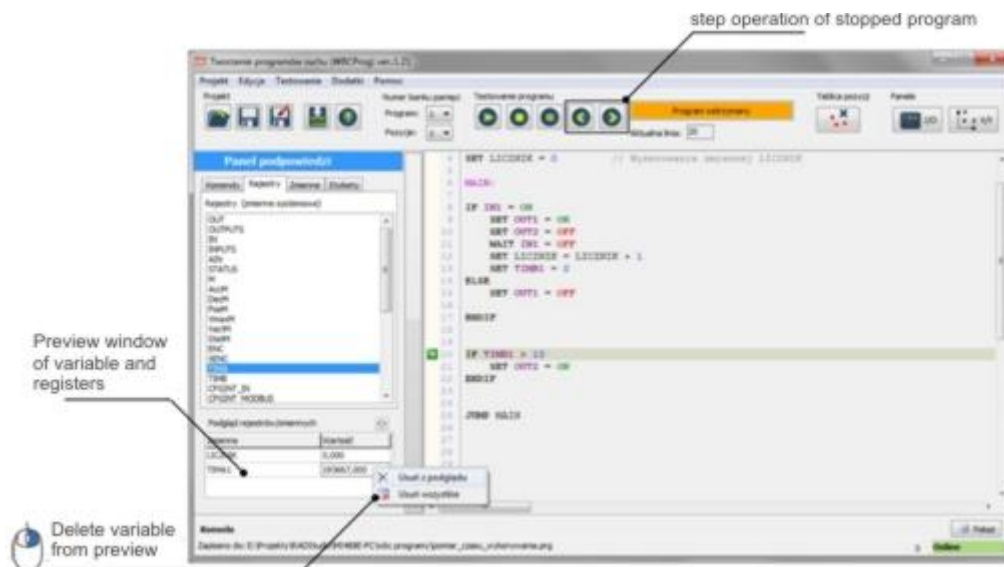
7.2.6 Running and testing of the program

Program sent to the driver can be run manually by  button. After running editor window is blocked. By  button program operations can be paused and then by   buttons it is possible to control its operation in step mode. Currently executed line is marked by  picture.

While program operation it is possible to preview current values of selected registers and variables (max 12). To add to preview on **Hints panel** please right-click on selected register (at **Registers** tab) or on variable (at **Variable** tab) and press „Add to preview”.



Picture. 5 Way of adding registers / variables to preview.



Picture. 6 Delete from preview.

7.3. WBL language description

WBL language (WObit Basic Language) is a simple text language similar to BASIC or ST (Structured Text) designed for creating motion programs for MIC488 driver. It facilitates creating programs for drives motion execution and is more flexible comparing to other languages.

Below is placed main information about programming of the driver in WBCprog.

Program compilation

Program compilation cause data generating based on entered commands which are clear for the driver. Compiled program when it has no errors can be send to the driver.

Program operations in the driver

Created program is executed by the driver command by command. Program should be „looped” by jump function. If after last command program will not jump to earlier commands it will be finished. **Finished program cause turning off all outputs and stopping and turning off of drives.**

Comments in program

Entering „//” signs before any command or description will cause that it will be skipped while program compilation. Example:

```
SET OUT1 = ON
//SET OUT2 = ON           // This program line will be skipped
SET OUT3 = ON
```

Floating points values

Numerical values with decimal should be entered using dot sign „.”. Comma sign can't be interpreted correctly and can cause an error during compilation.

```
SET VmaxM1 = 1.45          // Floating point value
```

Commands

Command means a line in program which is properly interpreter by the driver and executed. Command can contain additional parameters or not.

```
SET VmaxM1 = 2             // SET command used for record of max velocity of M1 drive register.
PABS M1 -5                 // PABS command generating absolute position
WAITPOS                   // WAITPOS command waits for achieving position (without parameters)
```

List of all commands is placed at **Błąd! Nie można odnaleźć źródła odwołania.** chapter.

Definitions

Definitions allows to create own name for constant numerical value, system register or user MODBUS register. To create definition please add „#” mark before name and after name please enter value or register to respond e.g.:

```
#REPEAT_X 10 // Constant value 10 as name "REPEAT_X"
#VELOCITY_M1 AccM1 // Register AccM1 as name "ACCELERATION_M1"
#HMI_VELOCITY $R50 // Register MODBUS 50 as name "HMI_VELOCITY"
...
IF HMI_VELOCITY < VELOCITY_MAX
SET VELOCITY_M1 = HMI_VELOCITY
ENDIF
```

Additionally program is equipped with several constant values:

Definition	Value	Description
OFF	0	Off status
ON	1	On status
M_OFF	0	Drive off
M_ON	1	Drive on
M_SPEED	2	Set velocity
M_POS_SEARCH	3	Move to set position
M_POS_OK	4	Set position achieved
M_POS_ERROR	5	Position error
M_POS_HOMING	6	Homing
M_POS_CORRECTION	8	Position correction
RIS	1	Interruption rising edge
FAL	2	Interruption falling edge
RISFALL	3	Rising and falling edge

Labels and jump functions

Jumps between commands allows execution of more complex controlling functions e.g.: execution of selected quantity of repetition, repeat selection of selected program part or its execution in dependence on fulfilling a condition. Jumps in program are made into so called **labels** – that means names addend in any program parts, ended by colon sign „:” e.g.:

```
FUNCTION_1: // Label with name "FUNCTION_1"
SET OUT1 = ON
SET OUT2 = OFF
RETURN
.. // do sth...
JUMP FUNCTION_1 // Jump to label „FUNCTION_1”
```

Names of labels and own variable

User can enter own variable, label names etc. consisting of any signs (letters and numbers). Created names can't be the same as commands and registers names. List of reserved names is placed in 9. chapter.

Registers

Registers are variable used by the driver for control of drives, inputs readout, controlling of outputs etc. There are also available universal user registers, enabling communication in MODBUS-RTU protocol with external devices. List of all registers is placed in **Błąd! Nie można odnaleźć źródła odwołania.** chapter.

User MODBUS registers

User has access to 500 registers (memory cells) to which can be saved or readied out any values. Access to this registers is also possible by RS232/RS485 interfaces through MODBUS-RTU protocol.

Values saved/readied out by MODBUS can be INT, DINT, REAL type. To correctly interpret different data types before register address please use prefix:

\$I – for values INT type

\$D – for values DINT type

```
SET $R0 = 10,2          // Record value 10,2 to 0 register (REAL type)
...
IF $I200 > 50           // Checking, if value (INT) in register 200 > 50
...
PABS M1 $R20            // Setting absolute position (REAL type) from register 20
```

Interrupts

Interrupts allows to interrupt currently executed program line (e.g. caused by signal change on selected IN input) and jump to defined label. It allows quick driver response to external signals.

Interrupt can be made for:

- Change of input status IN1...INX
- Record by Modbus of user registers (0...500).
- Cyclic interrupt made every 10ms

Interrupt cause jump to selected label which must be add to program:

Cause of interrupt	Name of label
Change of state on IN1 ... INX input	INT_IN1 ... INT_INX
Record by Modbus to 0...500 register	INT_MODBUS
Cycling interruption every 10ms	INT_TICK

Interrupt label must be ended by RETURN command and can't include jump commands (JUMP) inside. While interrupt (until RETURN command will appear) other interrupts are blocked. After return command program returns to execution program line from before interrupt.

To activate interrupt from selected signal to proper configure register please write e.g.:

```
SET CFGINT_IN1 = RIS      // Interrupt from state change from 0 to 1 on IN1 input
SET CFGINT_MODBUS = HIGH  // Interrupt from record to Modbus registers (0...500)
SET CFGINT_TICK = HIGH    // Periodic interrupt every 10 ms
```

For Interrupts configuration please use following values:

Interrupt from IN1..INX inputs:

- 0 (OFF)** – interrupt turned off
- 1 (RIS)** – interrupt turned on to rising edge (signal change from 0 to 1 on input)
- 2 (FALL)** – interrupt turned on to falling edge (signal change from 1 to 0 on input)
- 3 (RISFALL)** – interrupt turned on to rising and falling edge (signal change 0 to 1 or 1 to 0 on input)

Interrupt from user record by MODBUS, cyclic interrupts

- 0 (OFF)** – interrupt turned off
- 4 (HIGH)** – interrupt turned on

8. Program example in WBCprog

Below are placed examples using individual functions of MIC488 driver. Additionally in program catalogue named **WBC examples** are placed ready examples.

8.1 Use of inputs / outputs

Used registers:

- **OUTX** – single output
- **OUTPUTS** – all outputs
- **INX** – single input

- **INPUTS** – all inputs
- **MX_EN** – ENABLE signal for controlling of drive

Control of outputs

```
SET OUT1 = ON           // Turning on O1 output
SET OUT1 = OFF          // Turning off O1 output
SET OUTPUTS = 0         // Turning off all outputs
SET M1_EN = ON          // Activation of EN output of M1 drive
```

Checking of input state

```
IF IN2 = ON             // If I2 input is active
...                     // ...
ENDIF                   // End of condition
```

Waiting for input

```
WAIT IN2 = ON           // Wait, when I2 input will be active
```

8.2. Read out of analog inputs (0-10V)

Used registers: **AINX** – value from analog input

```
VABS M1 AIN1            // Setting velocity for M1 drive = values from AIN1 input
...                     // do sth...
IF AIN2 < 5.0           // Checking if voltage on AIN2 is smaller than 5.
...                     // do sth...
ENDIF
```



Values from AIN1/AIN2 registers can be calibrated at settings (**Configuration -> We 0-10V** tab)

8.3. Control of drives

Used registers:

- **AccMX, DecMX, VmaxMX** – parameters of ramp (acceleration, braking, max velocity)
- **PosMX** – current position
- **VactMX** – current velocity
- **MX** – control of individual drive (activation ENABLE output, STOP and BRAKE function)

Setting of motion parameters (ramp)

```
SET AccM1 = 10          // Setting of acceleration for M1 drive
SET DecM1 = 10          // Setting of braking for M1 drive
SET VMaxM1 = 0          // Setting of max velocity for M1

SET AccM2 = AccM1       // Parameters for M2 drive the same as for M1
SET DecM2 = DecM1
SET VMaxM2 = VMaxM1
```

Activation of a drive

```
SET M1_EN = ON          // Activation of M1 drive
SET M2_EN = ON          // Activation of M2 drive
```

Stopping/braking drive in motion

```
STOP M1                 // Stop of M1 drive (rapid)
BRAKE M2                 // Braking of M2 drive (braking to 0 velocity)
```

Homing of a drive

```
...
```




```
HOME M1 -2          // Start of homing of M1 and M2 drives into negative direction (toward
HOME M2 -2          // KL end proximity sensor)
WAITPOS
```

Setting velocity

```
VABS M1 5           // Setting velocity 5 for M1 drive
VABS M2 -2,5        // Setting velocity -2,5 for M2 drive(move into „negative“ direction)
...
VREL M1 -0,5        // Reducing M1 drive velocity by 0,5
VREL M2 0,5         // Increasing M2 velocity by 0,5
```

Setting position

```
PABS M1 10          // Setting position 10 for M1 drive
PABS M2 5,25        // Setting position 5,25 for M2 drive
WAITPOS             // Waiting for reaching set positions

PREL M1 1           // Increasing M1 drive position by 1
PREL M2 -1          // Reducing M2 drive position by 1
```

Setting position from position table

```
MOVEPTP @NAME_POSITION // Setting position from position table with name NAME_POSITION
WAITPOS                // Waiting for achieving position by drives
...
SET ABC = 2            // Saving to ABC variable 2 value
MOVEPTP ABC            // Setting position from position table with number of ABC
                        // variable
```

Waiting for achieving set position

1 way – command **WAITPOS** waiting for achieving position for all drives

```
PABS M1 10          // Setting 10 position for M1 drive
PABS M3 20          // Setting 20 position for M3 drive
WAITPOS             // Waiting for achieving position by all drives
```

2 way – checking **MX** register. When **MX = 0** the drive is not moving.

```
PABS M1 10          // Set 10 position for M1 drive
WAIT M1 = 0         // Waiting when M1 drive will not move
```

3 way – checking of **StatMX** state register.

```
PABS M1 10          // Setting 10 position for M1 drive
WAIT StatM1 = M_POS_OK // Waiting when M1 drive will reach set position
```



If selected drive is turned off (**EN** signal turned off) then **WAITPOS** command exclude checking its position.

Zeroing of position

```
SET PosM1 = 0       // Zeroing of M1 drive position
SET PosM2 = 10      // Overwriting current position for M2 drive by 10 value
```

8.3.1 Linear interpolation

MIC488 allows to execute linear interpolation using M1 and M2 drives which create X and Y axes. Interpolation adjust velocity of both drives to finish its motion at the same moment. Caution – please set the same parameters for ramp for both drives. Moreover ramps should be possibly short (large values of acceleration/delay).

Execution of linear motion requires following parameters:

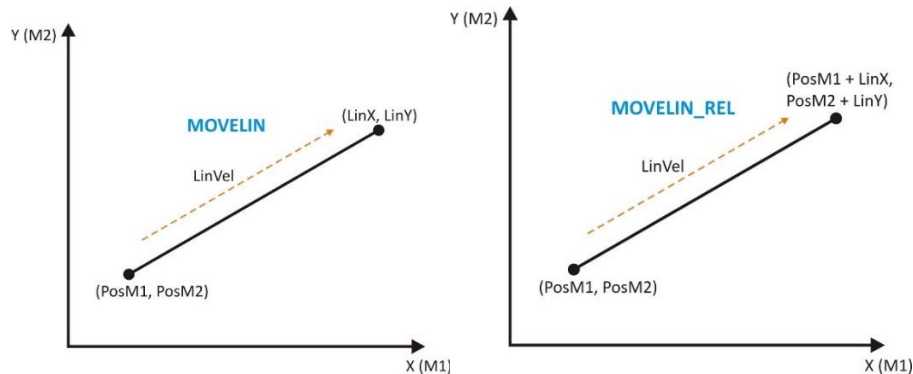
X/Y – final position

Vel – velocity of linear motion

This parameters should be entered at position table to selected position (then it defines linear motion parameters) or by **LinX**, **LinY**, **LinVel** registers which correspond directly to 0 index at position table and can be modified while program operation.

For execution a linear interpolation please use following commands:

- **MOVELIN** @ABC / index– cause drive linear motion at M1 and M2 axis to LinX, LinY point
- **MOVELIN_REL** @ABC / index – cause displacement of a drive at M1 and M2 axis by LinX, LinY value



Example of linear motion execution from position table

```
MOVELIN @POS0           // Execution of linear motion to points from table
WAITPOS                 // Waiting for end of motion
MOVELIN_REL @POS0       // Execution of relative linear motion
WAITPOS
```

Example of linear motion execution from parameters entered to a program
(**MOVELIN** / **MOVELIN_REL** commands must indicate 0 index at position table)

```
SET LinX = 50           // Target X position
SET LinY = 75           // Target Y position
SET LinVel = 20         // Linear motion velocity
...
MOVELIN 0               // Execution of linear motion to X, Y points
WAITPOS                 // Waiting for end of motion
MOVELIN_REL 0           // Execution of linear displacement by X,Y values
WAITPOS                 // Waiting for end of motion
```

8.3.2 Circular interpolation

MIC488 allows to execution of circular interpolation using M1 drives M2 which create X and Y axes.

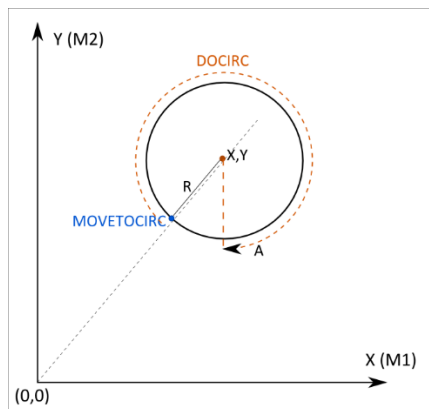
To provide correct operation of interpolation function, *GEAR* parameter at M1 and M2 drives configuration should be the same (to correctly execute a circle drives must have the same mechanical ratio). While executing an interpolation position correction from encoders is turned off.

Execution a circle using interpolation requires following parameters:

X/Y – center position of circle against zero position,

R – circle radius,

A – angle of executed circle (at degrees). If $A < 0$ motion is executed counterclockwise.



Picture. 283 Circular interpolation.

Mentioned parameters should be entered to position table at selected position (which defines then circle parameters) or by **CircX**, **CircY**, **CircR** and **CircA** registers which refers directly to 0 index at position table and can be modified while program operation.

Nr	Nazwa	M1 (X)	M2 (Y)	M3 (R)	M4 (A)
0	POS0	50	50	100	360
1	POS1	250	250	100	360
2	POS2	250	250	150	360
3	POS4	250	250	200	360
4	POS5	250	250	250	360

Picture. 329 Parameters for circle at position table.

For circle execution please use following commands:

- **MOVETOCIRC @ABC** – it cause drive motion at M1 and M2 axes to point on circle closest to zero position.
- **DOCIRC @ABC** – cause execution a circle.

Where @ABC means position name at position table which describes parameters of executed circle.

Example of circle execution on base parameters from position table

```
MOVETOCIRC @POS0      // Execution of motion to initial point on circle
WAITPOS               // Waiting for achieving a position
DOCIRC @POS0          // Execution a circle
WAITPOS               // Waiting for execution a circle
```

Example of circle execution on base parameters entered in program

(**MOVETOCIRC** and **DOCIRC** commands must indicate 0 index at position table)

```
SET CircX = 50        // X point of center of circle
SET CircY = 50        // Y point of center of circle
SET CircR = 45        // Circle radius
SET CircA = 360       // Angle of executed circle
...
MOVETOCIRC 0          // Execution of motion to initial point on a circle
WAITPOS              // Waiting for achieving a position
DOCIRC 0             // Execution a circle
WAITPOS              // Waiting for execution a circle
```

8.4. Readout / Save of encoder position

Used registers:

- **ENCX** – value of encoder counter in pulses
- **XENCX** – value of encoder counter converted into drive rotations

```

SET ENC1 = 0      // Zeroing value of ENC1 encoder counter
SET ENC2 = 100    // Entering 100 value into ENC2 encoder counter
PABS M3 XENC3     // Setting position for M1 drive equal position from ENC3 encoder

```

8.5. Timers

Used registers:

- **TIMAX** – timer with 10 ms resolution (counter value increased every 10 ms by 1)
- **TIMBX** – timer with 1 sec resolution,
where X means timer number (1...16)

```

SET TIMB1 = 0      // Zeroing of TIMB1 timer
...
IF TIMB1 > 60      // Checking if TIMB1 timer is bigger than 60 seconds
..                // If bigger...do sth
ENDIF

```

8.6. Mathematical operations and variable

```

SET ABC = 0        // Creating a new ABC variable and saving to it 0 value
SET ABC = ABC + 1   // Increasing ABC variable by 1
SET XYZ = ABC * 5    // Creating new XYZ variable equal ABC * 5
SET JKL = ENC1 + ENC2 // Creating new JKL variable equal ENC1 + ENC2 values

```

8.7 Interrupts

```

SET CFGINT_IN1 = RIS // Activation of interrupt at IN1 input on rising edge

MAIN:                                // Main loop
...                                  // we do sth there
JUMP MAIN

INT_IN1:                             // Interrupt label at IN1 input - there will occur a jump when at
...                                  // IN1 input will occur a high state
...                                  // Do sth...
REURN                                // return from interrupt

```

8.8 Program example

```
#REPEAT 10                // Defining auxiliary name for value 10
#IN_START IN3             // Defining auxiliary name for IN3 input
#IN_STOP IN4              // Defining auxiliary name for IN4 input
#IN_CLK IN5               // Defining auxiliary name for IN5 input
#OUT_STOP OUT1            // Defining auxiliary name for OUT1 output
#OUT_READY OUT2           // Defining auxiliary name for OUT2 output

SET AccM1 = 20            // Setting acceleration of motion
SET DecM1 = 20            // Setting braking of motion
SET VmaxM1 = 6            // Setting max velocity

WAIT IN_START = ON        // Waiting for active IN3 input
SET M1 = ON               // Turning on M1 drive
HOME M1 -1                // Start homing
WAITPOS                   // Waiting for end of homing

MAIN_LOOP:                // Label of main loop

    JUMP CHECK_STOP       // Jump to CHECK_STOP label
    JUMP EXECUTE           // Jump to EXECUTE label

JUMP MAIN_LOOP            // Jump to MAIN_LOOP

EXECUTE:                  // Label EXECUTE
IF StatM1 = M_POS_OK      // If drive reached set position
    SET OUT_READY = ON    // Set OUT2 output
    DELAY 5               // Wait 5 sec.
    SET OUT_READY = OFF   // Activate OUT2 output
    PREL M1 2             // Do drive motion by 2
ENDIF

RETURN                    // Return to place before jump

CHECK_STOP:               // Label SPRAWDZ_STOP
IF IN_STOP = ON           // If IN4 input
    STOP M1               // Stop the drive
    SET OUT_STOP = ON     // Activate OUT1 output
    DELAY 2               // Wait 2 sec.
    SET OUT_STOP = OFF    // Turn off OUT1 output
ENDIF
RETURN                    // Return to place before jump
```

9 List of commands and registers

Basic commands

Command	Description	Syntax
SET	It sets output, variables, register and execute mathematical operations.	SET X = Y SET X = Y operation Z SET X = function (Y) operation : available operations are described below
IF / ELSE /ENDIF	It compares status of inputs, outputs, variable, registers. When condition is fulfilled, next program lines are executed up to ENDIF command(to ELSE if exist). If not skipping commands between IF and ENDIF(to ELSE if exist).	IF X condition Y .. if fulfilled ENDIF or IF X condition Y .. if fulfilled ELSE ... if not fulfilled ENDIF condition – available conditions are described below
WHILE / ENDWHILE	It compares state of inputs, outputs, variables and registers. As long as condition is fulfilled there is a loop between WHILE and ENDWHILE. If condition is not fulfilled exit from ENDWHILE.	WHILE X condition Y .. executed as long as condition is fulfilled ENDWHILE condition – available conditions are described below
WAIT	Waits for status of inputs, outputs, variables, registers. If condition is fulfilled, transition to next line. If not, it waits until it will be fulfilled.	WAIT X condition Y condition – available conditions are described below
DELAY	Introduces time delay in seconds.	DELAY X
JUMP	Jump to existing program label.	JUMP NAME_LABEL
RETURN	Return to line after last jump command.	RETURN
END	End of program operation.	END

Parameters X, Y, Z can be user variables, device registers, MODBUS registers or constant values.

List of available mathematical operations and conditions

Mathematical operations	Mathematical functions	Conditions
<ul style="list-style-type: none"> ➤ + (addition), ➤ - (subtraction), ➤ * (multiplication), ➤ / (division), ➤ ! (negation) ➤ % (modulo), ➤ (binary or), ➤ & (binary and), ➤ (logic or), ➤ && (logic and), ➤ >> (right binary shift), ➤ << (left binary shift), 	<ul style="list-style-type: none"> ➤ sin (sinus), ➤ cos (cosines), ➤ tg (tangents), ➤ ctg (cotangents), ➤ asin (arcus sinus), ➤ acos (arcus cosines), ➤ sqrt (square root), ➤ min (min value), ➤ max (max value), <p>Trigonometric functions takes values at degrees.</p>	<ul style="list-style-type: none"> ➤ = (equal), ➤ < (smaller), ➤ > (bigger), ➤ <= (smaller or equal), ➤ >= (bigger or equal)

Motion commands

Command	Description	Syntax
HOME	It executes drive homing.	HOME MX Y Y – homing velocity
VABS	Setting absolute velocity (equal to entered value).	VABS MX Y Y – absolute velocity
VREL	Increasing (reducing) of current velocity.	VREL MX Y Y – relative velocity
PABS	Setting of absolute position (equal to entered value).	PABS MX Y Y – absolute position
PREL	Increasing (reducing) of current drive position.	PREL MX Y

		Y – relative position
BRAKE	Stop of drive (set 0 velocity).	BRAKE MX
STOP	Stop of drive (fast).	STOP MX
MOVEPTP	Set of position from position table.	MOVEPTP @NAME_POSITION
MOVELIN	Linear motion (for M1(X), M2(Y) drives)	MOVELIN @NAME_POSITION MOVELIN NUMBER_POSITION
MOVELIN_REL	Relative linear motion (for M1(X), M2(Y) drives)	
MOVETOCIRC	Motion to initial point on executing circle	MOVETOCIRC @NAME_POSITION MOVETOCIRC NUMBER_POSITION
DOCIRC	Circular motion (for M1(X), M2(Y) drives)	DOCIRC @NAME_POSITION DOCIRC NUMBER_POSITION
WAITPOS	Waiting for reaching position of all drives. CAUTION: If selected drive is not active (SET MX = OFF) it is skipped while checking position.	WAITPOS

MX: drive number (M1, M2, M3, M4)

General registers

Register	Description	Access	Range of values
OUTX	Access to selected output of universal driver. X = 1...8	R/W	0 – turned off output, > 0 – turned on output
OUTPUTS	Binary access to all universal outputs.	R/W	0 ... 65535
INX	Access to selected input of universal driver. X = 1...22	R	0 – not active output > 0 – active output
INPUTS	Binary access to all universal inputs.	R	0 ... 65535
AINX	Access to driver analog inputs X = 1...2	R	REAL
STATUS	Driver operation status.	R	Not available
TIMAX	Universal timer with basic time 10 ms X = 1...16	R/W	0 - 999999
TIMBX	Universal timer with basic time 1 sec X = 1...16	R/W	0 - 999999
CFGINT_INX	Interrupts configuration for IN1...IN8 inputs X = 1...8	R/W	0 (OFF) – interrupt off 1 (RIS) – interrupt on rising edge 2 (FALL) – interrupt on falling edge 1 (RISFALL) – interrupt on both edges
CFGINT_MODBUS	Interruption configuration for data record to registers of Modbus user.	R/W	0 (INT_OFF) - off 4 (INT_HIGH) - on
CFGINT_TICK	Configuration of cyclic interrupt executed every 10ms	R/W	0 (INT_OFF) - off 4 (INT_HIGH) - on

W – write, **R** - readout

Drives registers

Register	Description	Access	Range of values
MX	Control of ENABLE output. Turning on drive, checking if drive is in motion.	R/W	Record: 0 (OFF) – turning off of a drive 1 (ON) – turning on of a drive Readout: 0 – drive in stand-by 1 – drive in motion
StatMX	Status of drive operation.	R	0 (M_OFF) – drive turned off 1 (M_ON) – drive turned on 2 (M_SPEED) – velocity mode 3 (M_POS_SEARCH) – position mode 4 (M_POS_OK) – position achieved 5 (M_POS_ERROR) – position error 6 (M_POS_HOMING) – homing 8 (M_POS_CORRECTION) – correction of position 9 (M_POS_LIM_L) – achieved L end position 10 (M_POS_LIM_R) – achieved R end

			position
VmaxMX	Max velocity for position mode.	R/W	REAL
VactMX	Current drive velocity.	R	REAL
AccMX	Acceleration for speeding up in set position mode. Acceleration for speeding up and braking at set position mode.	R/W	REAL
DecMX	Acceleration for braking at set position mode.	R/W	REAL
PosMX	Current drive position.	R/W	REAL
LimLMX	Program position limitation towards negative direction (L)	R/W	REAL
LimRMX	Program position limitation towards positive direction (R)	R/W	REAL
ENCX	Pulses from encoder.	R/W	DINT
XENCX	Pulses from encoder converted into drive position.	R	REAL
PTPM	Positions for drivers responding to positions at position table with 0 index.	R/W	REAL
CircX, CircY, CircR, CircA	Circle parameters for circular interpolation (it corresponds parameters at position table with 0 index). X,Y centre of circle, R –radius, A – angle	R/W	REAL
LinX, LinY, LinVel	Parameters for linear motion (it corresponds to parameters at position table with 0 index). X,Y final point, Vel – velocity	R/W	REAL

X: drive number 1...4

User Modbus registers

Register	Description	Access	Range of values
\$I0 - \$I499	User registers(values INT type).	R/W	INT (-32768...32767)
\$D0 - \$D498	User registers (values DINT type). CAUTION: Available only even addresses!	R/W	DINT (-2147483648...2147483647)
\$R0 - \$R498	User registers (values REAL type). CAUTION: Available only even addresses!	R/W	REAL (floating points)

10 MODBUS communication

Driver enables communication with master device in (MASTER) MODBUS-RTU protocol. Communication can be made by RS232 (COM1) port or RS485 (COM2).

Transmission parameters

- Default address: 1 (configurable in range 1...126)
- Default transmission baudrate: **38400 b/s** (available velocities 9600, 19200, 38400, 57600, 115200)
- Stop bits: **1**, Parity: **none**
- Timeout: **750µs** (max time between next bytes in a frame)

Description of communication, user register list and way of drive controlling by MODBUS-RTU is available at „*MIC488_protokol_MODBUS.pdf*” documentation.

11 Record of changes

Version	Firmware	PC program
1.02	- first version	- first version
1.03	- Interrupts support (IN1 – IN8, Modbus) - MOVEPTP function can take as an argument also number from position table (earlier it was only name of position)	- interrupts support (IN1 – IN8, Modbus) - MOVEPTP function can take as an argument also number from position table (earlier it was only name of position)
1.04	- improved position control from encoder	



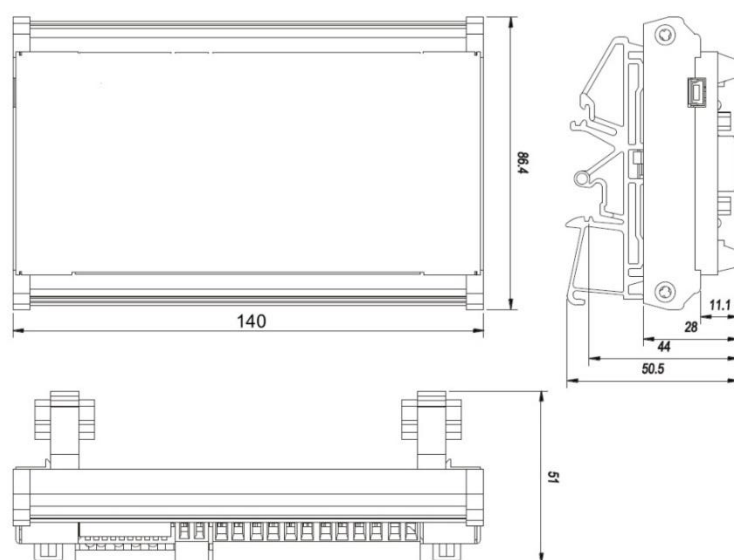
	<ul style="list-style-type: none"> - improved error modes of position control (ERRCTR) - improved errors signalization by LED diodes 	
1.05	- support of commands for circular interpolation	- support of commands for circular interpolation
1.06	- support of WHILE-ENDWHILE loop commands	- support of WHILE-ENDWHILE loop commands
1.08	<ul style="list-style-type: none"> - improved stack JUMP-RETURN - nested conditions WHILE-ENDWHILE - interrupts support for IN9...IN20 inputs (encoders) 	-
1.10	<ul style="list-style-type: none"> - improved error of IN9...IN20 inputs readout - optimized function (IF,WHILE) - new commands linear motion MOVELIN - support of "breakpoint" CAUTIONS: SETTINGS RESET, RECOMPILATION	<ul style="list-style-type: none"> - optimized function (IF,WHILE) - new commands linear motion MOVELIN - support of "breakpoint"
1.21	<ul style="list-style-type: none"> - optimized ELSE condition function for IF/ENDIF - new mathematical function (sin, cos tg,ctg, asin, acos, sqrt) - support of variable preview while program testing CAUTIONS: SETTINGS RESET	<ul style="list-style-type: none"> - support of new mathematical functions - preview of variable while program testing - option of programming driver from a file - SET command don't have to be used for settings of registers and earlier defined variables
1.22	<ul style="list-style-type: none"> - configuration of inputs level influence also to basic signal and end signals - addend program end positions - addend Modbus registers for program end positions (120...134) - added signalization of achieving end positions at drive status register CAUTIONS: SETTINGS RESET	Support for firmware v1.22: <ul style="list-style-type: none"> - correct readout and simulation of outputs and inputs - access to program end positions (LimLMX and LimRMX registers at WBL)
1.25	<ul style="list-style-type: none"> - optimize drive control function - option of programming by COM0 port (RS485) 	

SETTINGS RESET: After update factory settings will be restored.

RECOMPILATION: It requires program recompilation at current MIC488-PC software.

12 Technical parameters

Description	Parameter
Power supply	12...26V DC
Current consumption	100 mA@12 V, 50 mA@24 V
Outputs for controlling drives	4 (STEP, DIRECTION, ENABLE)
Controlling outputs of (EN, DIR, CLK) M1..M4 drives	High state 5 V, low state 0 V Max. load 20 mA/output
Inputs IN1..IN8	Opt insulated Low state: <2 V, high state: +4...+24 V DC Min. pulse length: 20ms
Inputs IN9..IN20	Low state: <2 V, high state: +3..24 V DC Possibility of connection incremental encoders
Outputs OUT1.. OUT8	Transistor OC PNP type, max. 200mA / output
Communication	COM0: RS485 (opt insulated), Communication protocol: internal for communication with PC Baudrate: 57600 COM1: RS232 COM2: RS485 Communication protocol: MODBUS-RTU SLAVE Stop bits: 1 Parity: none Baudrate: 9600, 19200, 38400, 57600, 115200 USB: 1.1, 2.0 (HID)
Device memory	Program: 6 x 1000 commands Positions: 8 x 225 positions x 4 drives User registers MODBUS: 500
Operating temperature	5..50° C
Weight	200 g
Degree of protection	IP20



Picture. 35 Technical drawing

13 Declaration of conformity

DECLARATION OF CONFORMITY EC No. 03/11/2014

P.P.H. WObit E.K.J. Ober s.c.
Dęboryce 16, 62-045 Pniewy

tel.: +48 61 22 27 410
fax: +48 61 22 27 439



We hereby declare, that manufactured product :

Name: Programmable, 4-axis trajectory controller
Type: MIC488


is in conformity with the provisions of:

- **2004/108/EC** „Electromagnetic compatibility” (EMC), taken to effect by Electromagnetic Compatibility Act of April 13, 2007 (Journal of Laws 2007, no. 82, item 556)

and is in conformity with the relevant Community harmonization legislation:

PN-EN 61000-6-2:2008 Electromagnetic compatibility (EMC) – Immunity for industrial environments

PN-EN 61000-6-4:2008 Electromagnetic compatibility (EMC) – Emission standard for industrial environments

This declaration of conformity provide the basis for labeling the product the  marking.

This declaration applied solely to the product in the state in which was introduced to the market and excludes components which are added by the end user or carried out by subsequent actions.

Declaration of conformity does not cover any modernizations made inconsistently to the instruction manual and/or without approval of the manufacturer.

This declaration of conformity is prepared under the sole responsibility of the manufacturer.

Place of issue: Dęboryce

Date of issue: 24.11.2014 r.