Instrukcja obsługi
MIC488

Programowalny, 4-osiowy kontroler trajektorii

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

Dziękujemy za wybór naszego produktu.

Niniejsza instrukcja ułatwi Państwu prawidłową obsługę i poprawną eksploatację opisywanego urządzenia.

Informacje zawarte w niniejszej instrukcji przygotowane zostały z najwyższą uwagą przez naszych specjalistów i służą wyłącznie jako opis produktu. Na podstawie przedstawionych informacji nie należy wnioskować o określonych cechach lub przydatności produktu do konkretnego zastosowania.
Informacje te nie zwalniają użytkownika z obowiązku poddania produktu własnej ocenie i sprawdzenia jego właściwości. Zastrzegamy sobie możliwość zmiany parametrów produktu bez powiadomienia.

- Prosimy o uważne przeczytanie instrukcji i stosowanie się do zawartych w niej zaleceń.
- Prosimy o zwrócenie szczególnej uwagi na następujący znak:

UWAGA!
Niedostosowanie się do instrukcji może spowodować uszkodzenie urządzenia albo utrudnić posługiwanie się sprzętem lub oprogramowaniem.

UWAGA!
Z gwarancji wyłączone są uszkodzenia mechaniczne lub elektryczne wynikające z przepięć, zwarcia oraz usterki czy awarie, których przyczyną jest wadliwa obsługa lub eksploatacja ze strony kupującego / Użytkownika.

Spis treści

1. Zasady bezpieczeństwa i montażu

1.1 Zasady bezpieczeństwa

- Przed pierwszym uruchomieniem urządzenia należy dokładnie zapoznać się z niniejszą instrukcją obsługi i zachować ją do późniejszego wykorzystania.
- Należy zapewnić właściwe warunki pracy, zgodne ze specyfikacją urządzenia (np.: napięcie zasilania, temperatura, maksymalny pobór prądu.
- Chronić urządzenie przed przedostaniem się do jego wnętrza jakichkolwiek przedmiotów lub płynów – grozi porażeniem i uszkodzeniem urządzenia.
- Podstawowe informacje, których znajomość i stosowanie zapewnią, aby urządzenie było użytkowane bezpiecznie i zgodnie z jego przeznaczeniem, zostaną uwidocznione na urządzeniu, a w przypadku braku takiej możliwości zostaną podane w niniejszym dokumencie.
- Urządzenie, łącznie z jego częściami składowymi, jest wykonane w taki sposób, aby zapewnić jego bezpieczny i prawidłowy montaż i przyłączenie.
- Urządzenie jest zaprojektowane i wyprodukowane w sposób zapewniający jego zgodność zasadami ochrony przed zagrożeniami, wymienionymi w ww. puntach, pod warunkiem, że urządzenie to jest użytkowane w sposób zgodny z jego przeznaczeniem i że jest odpowiednio utrzymywane.
- Urządzenie może zakłócić pracę czułych urządzeń radiowo-telewizyjnych umieszczonych w pobliżu.

1.2 Zalecenia montażowe

Poniżej zawarte zostały zalecenia, do których należy się stosować, by zapewnić poprawną pracę sterownika.

- Sterownik nie powinien być zasilany z tego samego źródła co sterowniki / serwonapędy silników.
- Należy zminimalizować wpływ zakłóceń pochodzących z zewnętrznych źródeł
- W celu **minimalizacji zakłóceń** przewód łączący silnik ze sterownikiem powinien być **ekranowany** lub powinien być **skręcany parami** (osobna skrętka dla fazy A i B). Zaleca się także stosowanie **pierścienia ferrytowego** na przewodzie silnika przy sterowniku.
- Przewody sygnałowe **(CLK, DIR, EN) nie powinny biec w pobliżu przewodów silnika** i powinny być możliwie krótkie.
- Przy stosowaniu serwonapędów zasilanych z tej samej sieci należy wyposażyć je w odpowiednie filtry zasilania w celu eliminacji zakłóceń mogących wpływać na pracę sterownika. Zastosowanie filtrów może być koniecznie również w przypadku występowania innych zakłóceń z sieci.

2. Wstęp

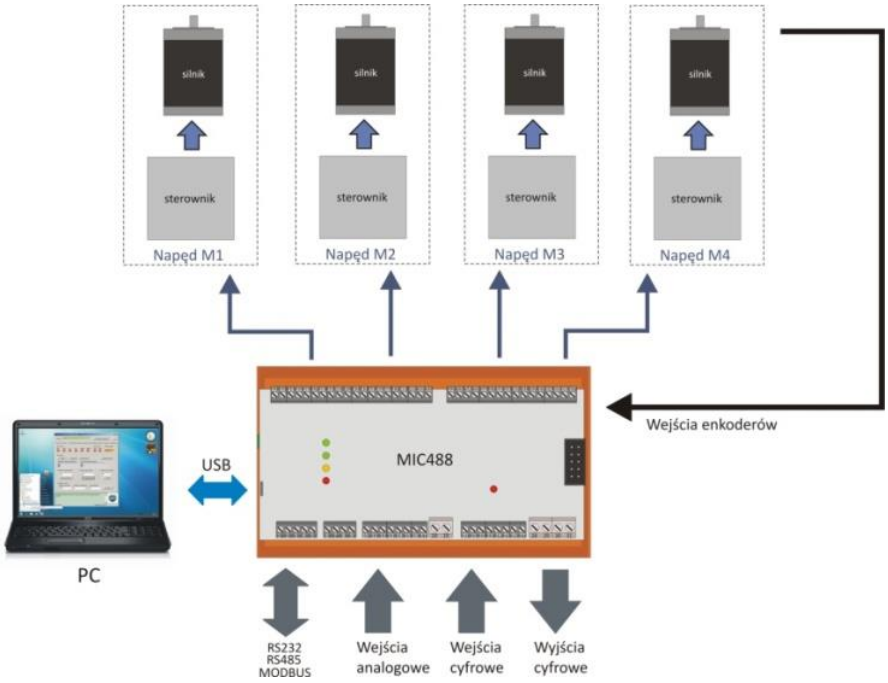
2.1 Przeznaczenie

MIC488 to programowalny 4-osiowy kontroler, którego główną funkcją jest sterowanie do 4 napędów krokowych lub serwonapędów w trybie KROK/KIERUNEK.

Kontroler umożliwia także podłączenie enkoderów inkrementalnych do nadrzędnej kontroli pozycji. Oprócz uniwersalnych wejść/wyjść oraz dwóch wejść 0-10V dostępne są także porty komunikacyjne (RS232, RS485) pracujące w protokole MODBUS (slave), pozwalające na komunikację z np. panelami HMI.

Dedykowane oprogramowanie pozwala w prosty sposób skonfigurować trajektorię ruchu napędów oraz tworzyć programy sterujące napędami, wyjściami oraz reagować na stany wejść lub wartości zmiennych komunikacyjnych. Tworzenie programów ruchu odbywa się w intuicyjnym środowisku poprzez komendy tekstowe.

MIC488 może zastąpić tradycyjny sterownik programowalny (PLC) wszędzie tam, gdzie konieczna jest precyzyjna kontrola kilku napędów.



Właściwości MIC488:

- Sterowanie do 4 napędów poprzez wyjścia KROK – KIERUNEK – ZEZWOLENIE
- Realizacja interpolacji liniowej i kołowej
- Możliwość realizacji programów ruchu z pamięci (4000 komend)
- Możliwość podłączenia do 4 enkoderów inkrementalnych do nadrzędnej kontroli pozycji
- 20 wejść cyfrowych (8 optoizolowanych)
- 8 wyjść tranzystorowych
- 2 wejścia analogowe 0-10V
- Możliwość podłączenia dodatkowego modułu rozszerzeń wejść-wyjść (8 wejść / 8 wyjść)
- 3 porty komunikacyjne (RS232, 2 x RS485)
- Komunikacja w sieci MODBUS-RTU (tryb slave i master)
- Złącze USB do programowania
- Zasilanie 12…26 VDC, pobór prądu 50mA@24V
- Intuicyjne oprogramowanie do konfigurowania i programowania sterownika

2.2 Funkcje

Główną funkcją sterownika MIC488 jest kontrola do 4 napędów w trybie KROK/KIERUNEK. Sterownik generuje trajektorię ruchu uwzględniając rampę (przyspieszenie, stała prędkość, hamowanie) tak by uzyskać płynny ruch bez utraty pozycji.

Ponadto możliwe jest podłączenie enkoderów, które mogą służyć do nadrzędnej kontroli pozycji silnika i korygować ją eliminując błędy pozycjonowania napędu. Jest to szczególnie przydatne w przypadku sterowania silnikami krokowymi, które w niektórych sytuacjach mogą utracić pozycję. Enkoder daje dodatkowo informację o mechanicznym zablokowaniu napędu.

MIC488 posiada także uniwersalne wejścia oraz wyjścia, które mogą być wykorzystane podczas realizowania programów ruchu do sterowania oraz reagowania na sygnały z urządzeń zewnętrznych.

- **Sterowanie napędami:**
  - Precyzyjne sterowanie do 4 napędów.
  - Automatyczne przeliczanie jednostek ruchu z impulsów na np. mm.
  - Kontrola rampy ruchu (przyspieszanie, stała prędkość, hamowanie).
  - Tryb zadanej prędkości (absolutnie lub relatywnie).
  - Tryb zadanej pozycji (absolutnie lub relatywnie).
  - Tryby bazowania w oparciu o czujniki krańcowe / enkoder.
  - Tryb nadrzędnej kontroli pozycji z wykorzystaniem enkodera.
  - Dwa tryby interpolacji liniowej
  - Tryb interpolacji kołowej (ruch po łuku i okręgu)

- **Sterowanie urządzeniami zewnętrznymi poprzez wyjścia cyfrowe / Modbus master**

- **Reakcja na sygnały zewnętrzne:**
  - Wejścia cyfrowe dla sygnałów z czujników krańcowych, enkoderów, sygnałów sterujących .
  - Komunikacja po protokole MODBUS-RTU do bezpośredniego sterowania napędami i funkcjami sterownika oraz dostęp do pamięci wykorzystywanej przez programy.
  - Odczytywanie impulsów z enkoderów.

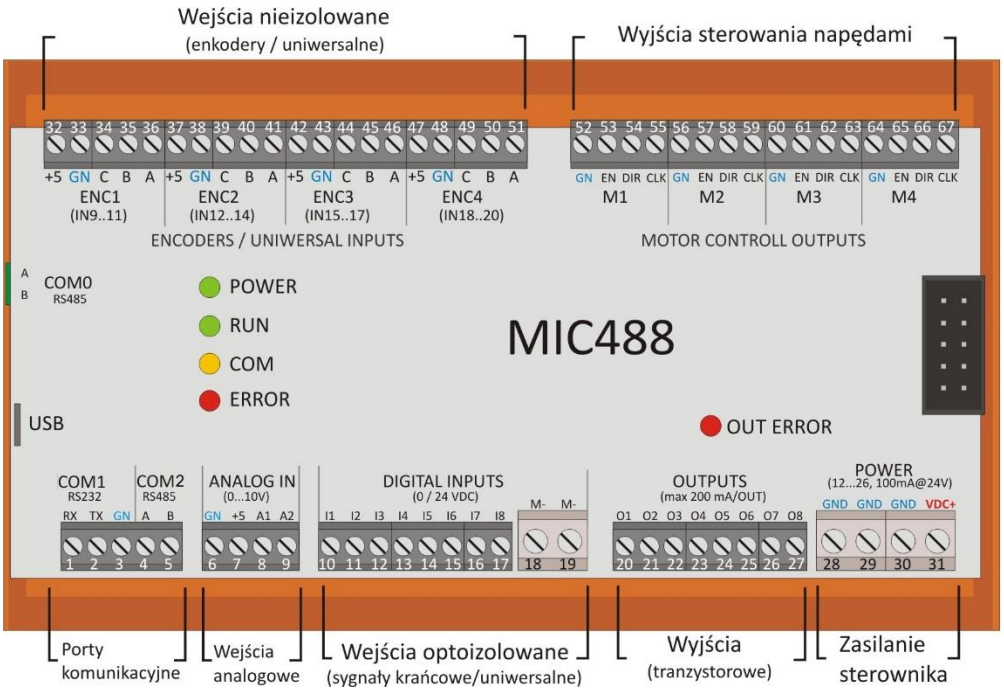
- **Realizacja programów z pamięci sterownika:**
  - Możliwość zaprogramowania sterownika programem składającym się z 4000 komend.
  - Możliwość zaprogramowania do 8 niezależnych „banków” zawierających po 200 zdefiniowanych pozycji dla 4 napędów.
  - Proste komendy realizujące dowolne funkcje ruchu wybranego napędu.
  - Funkcje opóźnień czasowych, oczekiwania na wejścia, skoków i warunków.
  - Możliwość zagnieżdżania warunków.
  - Funkcje matematyczne (dodawanie, odejmowanie, mnożenie, dzielenie).
  - Funkcje trygonometryczne (sin, cos, tg, ctg, asin, acos, sqrt)
  - Obsługa liczb zmiennoprzecinkowych.
  - Dostęp do pamięci użytkownika przez rejestry MODBUS.

Programowanie sterownika odbywa się za pomocą aplikacji MIC488-PC, pozwalającej na:

- Konfigurację poszczególnych napędów.
- Ręczne zadawanie ruchu dla napędów.
- Szybkie zadawanie pozycji dla napędów (tryb JOG).
- Odczyt stanu pracy napędu, aktualnej prędkości i pozycji.
- Podgląd wejść/wyjść sterownika.
- Podgląd wartości w rejestrach MODBUS użytkownika.
- Dodawanie aktualnych pozycji napędów do tablicy pozycji.
- Tworzenie programów ruchu oraz ich testowanie (podgląd aktualnie wykonywanej linii programu).

3. Opis sprzętu

3.1 Rozmieszczenie złącz i kontrolek



Rys. 1 Opis złącz i kontrolek sterownika MIC488.

Nr	Nazwa	Opis	
1	RX	Odbiór danych	Interfejs RS232
2	TX	Nadawanie	
3	GN	Masa	
4	A	Sygnał +	Interfejs RS485
5	B	Sygnał -	
6	GN	Masa	Wejścia analogowe
7	5V	Wyjście +5V (maks. 100mA)	
8, 9	A1,A2	Wejścia analogowe 0…10V	
10…17	I1..I8	Wejścia uniwersalne IN1…IN8	Wejścia IN1..IN9
18, 19	M-	Masa dla wejść IN1..IN8	
20…27	O1..O8	Wyjścia tranzystorowe (maks. 200mA/wyjście)	
28, 29, 30	GND (GN)	Masa	Zasilanie sterownika
31	VDC+	Zasilanie 12…26 VDC	
32, 37, 42, 47	+5	Wyjście zasilania enkodera +5V (maks. 75mA/wyjście)	Zasilanie i wejścia enkoderów ENC1..ENC4
33, 38, 43, 48	GN	Masa	
34	C (IN9)	Wejście kanału C enkodera 1 / wejście IN9)	
35	B (IN10)	Wejście kanału B enkodera 1 / wejście IN10)	
36	A (IN11)	Wejście kanału A enkodera 1 / wejście IN11)	
…			
52, 56, 60, 64	GN	Masa	Wyjścia sterowania napędów M1..M4
53, 57, 61, 65	EN	Wyjście sygnału zezwolenia pracy napędu (ENABLE)	
54, 58, 62, 66	DIR	Wyjście sygnału kierunku (DIR)	
55, 59, 64, 67	CLK	Wyjście sygnału kroku (CLK)	

	POWER	Sygnalizacja zasilania sterownika
	RUN	Sygnalizacja realizacji programu z pamięci sterownika
	COM	Sygnalizacja komunikacji MODBUS
	ERROR	Sygnalizacja błędu sterownika

3.2 Zasilanie

Zasilanie sterownika
Sterownik może być zasilany napięciem 12…26V DC. Dla napięcia zasilania 24V pobór prądu wynosi około 50mA. Zasilanie należy podłączyć pod zaciski VDC+ oraz GND (31, 30).
W przypadku wykorzystania wyjść tranzystorowych należy uwzględnić pobór prądu dla wyjść.

UWAGA!
Zasilanie sterownika powinno być niezależne od zasilania napędów. Ponadto przy stosowaniu serwonapędów zasilanych z tej samej sieci należy wyposażyć je w odpowiednie filtry zasilania w celu eliminacji zakłóceń mogących wpływać na pracę sterownika. Zastosowanie filtrów może być koniecznie również w przypadku występowania innych zakłóceń z sieci.

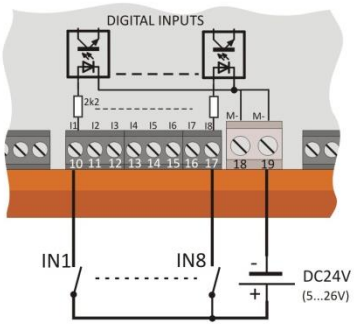
Wyjście +5V
Sterownik udostępnia napięcie +5V, które można wykorzystać do zasilania enkoderów (typu TTL) lub zewnętrznych potencjometrów podłączonych do wejść AIN1/AIN2. Maksymalny pobór prądu dla wszystkich wyjść +5V nie powinien przekraczać **400mA.**

UWAGA!
Nie należy zwierać wyjść +5V z masą (GND/GN), ani napięciem zasilania. Może to spowodować uszkodzenie sterownika. Należy także unikać prowadzenia przewodów z sygnałem +5V w pobliżu innych sygnałów mogących generować zakłócenia.

3.3 Wejścia uniwersalne IN1…IN8

Uniwersalne optoizolowane wejścia IN1…IN8 umożliwiają podłączenie zewnętrznych sygnałów sterujących lub sygnałów z czujników krańcowych dla napędów M1…M4. Wejście jest aktywowane napięciem 24V (min. 5V, maks. 26V). Sygnałem wspólnym (ujemnym) dla wejść IN1...IN8 są wejścia **M-** (zaciski 18,19).

Wejścia sią odświeżane co 10ms, przy wykorzystaniu ich do wyzwalania przerwań należy liczyć się z takimi opóźnieniami.



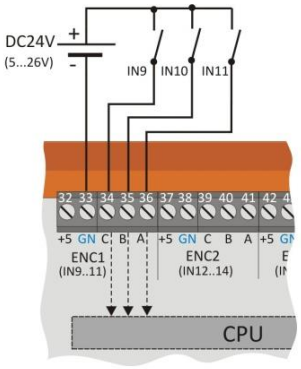
Rys. 2 Wejścia optoizolowane (IN1 … IN8)

Parametry:
- optoizolacja
- stan wysoki: 24 VDC (min 5V, maks. 26V)
- stan niski: < 2 VDC
- filtracja min. 10ms

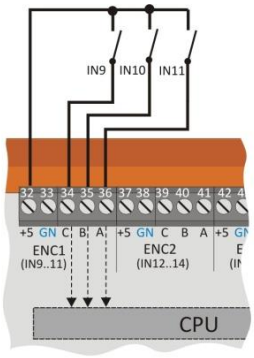
3.4 Wejścia szybkie IN9…IN22 / enkoderowe

Wejścia IN9…IN22 mogą służyć jako wejścia uniwersalne lub enkoderowe (do podłączenia enkoderów inkrementalnych). Wejścia nie posiadają izolacji, sygnałem wspólnym (ujemnym) jest masa zasilania sterownika GND (GN). Dodatkowo obok wejść znajdują się wyjścia napięcia +5V, które mogą służyć do bezpośredniego zasilania enkoderów typu TTL 5V.

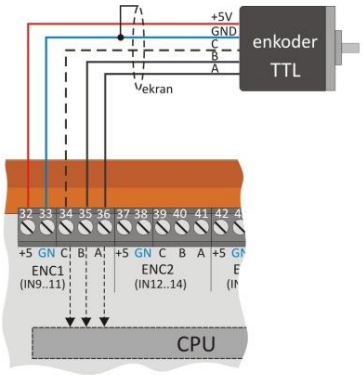
Wejścia są odczytywane w skrypcie bezpośrednio, nie wprowadzają opóźnień przy generowaniu przerwań.



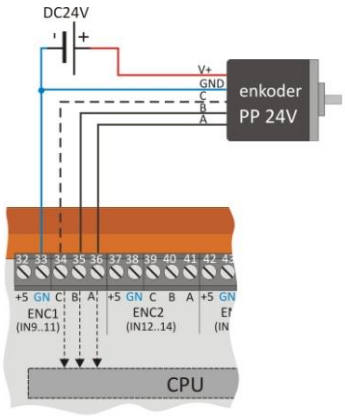
Rys. 3 Przykład sterowania wejściami zewnętrznymi sygnałami 24V.



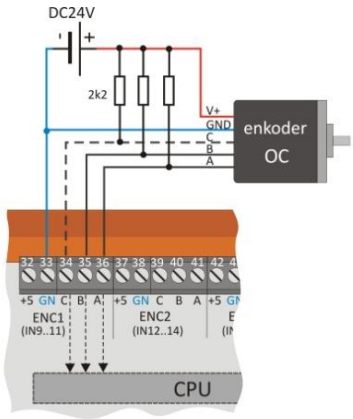
Rys. 4 Przykład sterowania wejściami z wykorzystaniem napięcia +5V.



Rys. 5 Przykład podłączenia enkodera z wyjściami typu TTL.



Rys. 6 Przykład podłączenia enkodera z wyjściami typu Push-Pull.

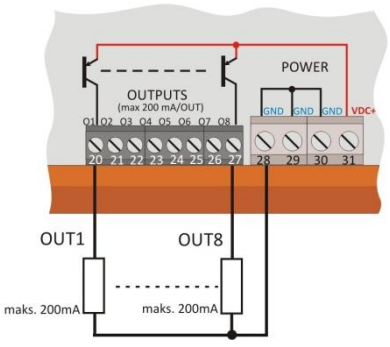


Rys. 7 Przykład podłączenia enkodera z wyjściami typu OC.

Jeśli sterowanie napędem z kontrolą pozycji z enkodera nie działa poprawnie (np. napęd przejeżdża zadaną pozycję) należy zamienić sygnały A z B enkodera.

3.5 Wyjścia uniwersalne OUT1…OUT8

Uniwersalne wyjścia OUT1…OUT1 umożliwiają wysterowanie zewnętrznych elementów wykonawczych których pobór prądu nie przekracza 200mA. Wyjścia w stanie aktywnym podają napięcie zasilania sterownika VDC+. Sygnałem wspólnym dla wyjść jest masa zasilania sterownika **GND** (zaciski 28, 29, 30, 31).



Rys. 8 Wyjścia OUT1..OUT8

Parametry:

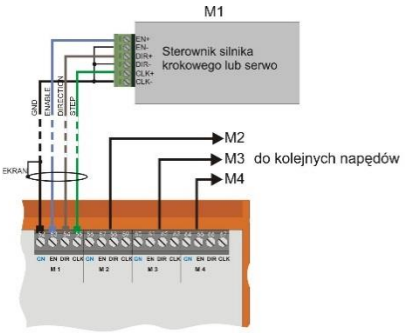
- wyjścia tranzystorowe typu OC (otwarty kolektor)
- obciążalność 200mA/wyjście
- zabezpieczenie przed obciążeniem indukcyjnym
- zabezpieczenie przed przeciążeniem/zwarciem (>300 mA)
- stan wysoki: napięcie zasilania VDC+

UWAGA!

Wyjścia posiadają zabezpieczenie przeciążeniowe i przeciwzwarciowe, które jest aktywowane po przekroczeniu prądu 300mA przez któreś z wyjść. Zadziałanie zabezpieczenia powoduje wyłączenie wszystkich wyjść i zapalenie czerwonej diody OUT ERROR. By przywrócić działanie wyjść należy usunąć przyczynę przeciążenia i zresetować zasilanie urządzenia.

3.6 Wyjścia sterowania napędami M1…M4

Wyjścia M1…M4 (sygnały: EN – zezwolenie na pracę, DIR – kierunek, CLK – krok) służą do sterowana napędami krokowymi lub serwonapędami w trybie KROK-KIERUNEK. Przewody te nie powinny biec w pobliżu przewodów silnikowych lub zasilających. Ponadto zaleca się ekranować przewody.



Parametry:

- stan wysoki 5V, stan niski 0V, maks. 20mA
- sygnał CLK (krok) – częstotliwość maks. 64kHz, szerokość impulsu 10µs

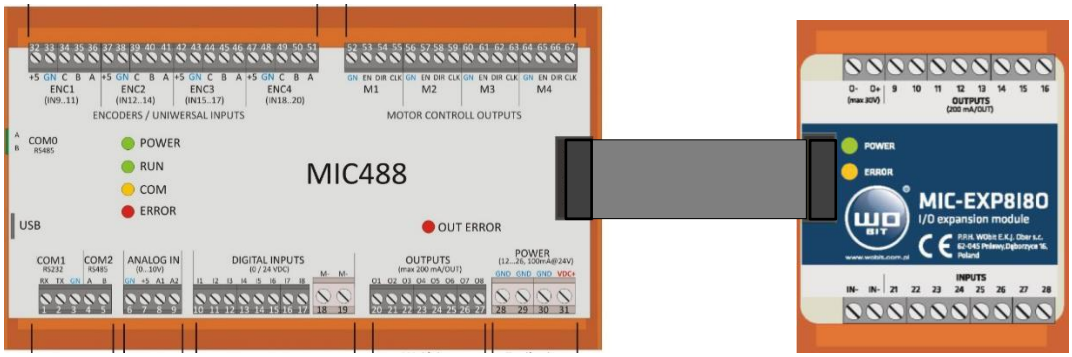
Rys. 9 Wyjścia sterowania napędami M1..M4

3.7 Moduł rozszerzeń wejść-wyjść MIC-EXP8I8O

Do sterownika MIC488 może zostać podłączony dodatkowy moduł wejść-wyjść pozwalający rozszerzyć ilosć dostępnych wejsc i wyjsc o 8. Po podłączeniu modułu i włączeniu zasilania sterownik automatycznie rozpoznaje podłączony moduł, nie ma konieczności jego konfigurowania.

Użytkownik ma do dyspozycji dodatkowe wejścia i wyjścia:

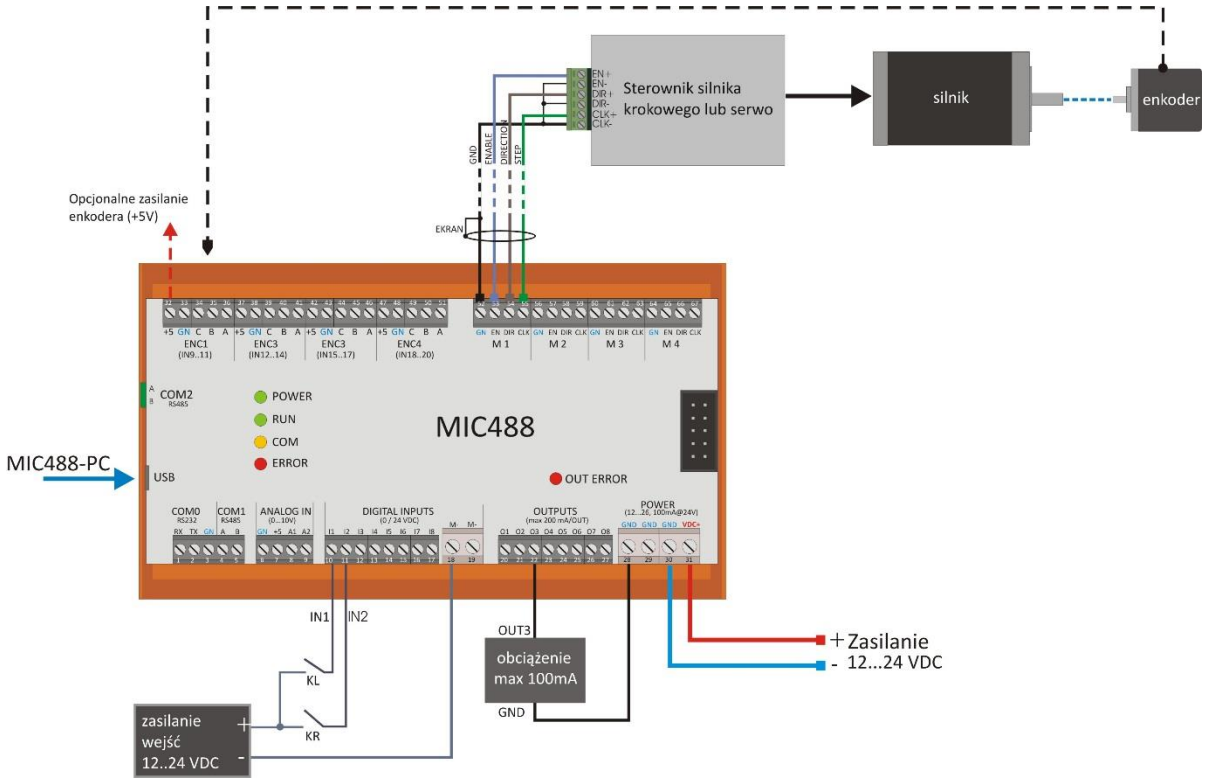
- 8 wejść IN21… IN28
- 8 wyjść IN9…IN16



MIC488 MIC-EXP8I8O

3.8 Przykładowe podłączenie

Na poniższym rysunku pokazano przykładowe podłączenie sygnałów do MIC488. Kontroler steruje jednym napędem M1. Przerywaną linią pokazano opcjonalne podłączenie enkodera inkrementalnego dla dodatkowej kontroli pozycji (dla napędu M1 enkoder należy podłączyć do kanału enkodera ENC1).
Ponadto wykorzystano dwa wejścia IN1 i IN2 dla sygnałów krańcowych napędu (KL i KR) oraz jedno wyjście OUT3 do załączania zewnętrznego obwodu (np. lampki sygnalizacyjnej, przekaźnika itp.).



Rys. 10 Przykład podłączeń sygnałów do kontrolera.

4. Oprogramowanie MIC488-PC

4.1 Połączenie MIC488 z PC

Konfiguracja i programowanie sterownika odbywa się przy pomocy aplikacji MIC488-PC. Sterownik może komunikować się z komputerem za pomocą USB lub RS485 (podłączenie do portu COM0 sterownika). Zaleca się używania połączenia RS485 w aplikacjach w których mogą pojawiać się spore zakłócenia (np. sterowanie serwonapędami AC).

Połączenie USB

Do połączenia przez USB należy użyć przewodu USB typu A – B mini. Po podłączeniu do komputera można włączyć zasilanie sterownika i uruchomić program MIC488-PC. Poprawna komunikacja będzie sygnalizowana informacją w górnym oknie programu.

UWAGA!

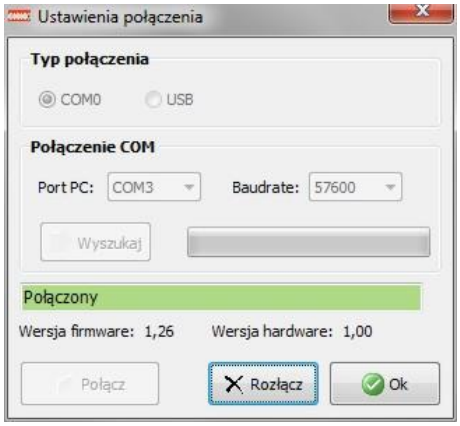
1) **Połączenie USB należy wykonać zawsze przed włączeniem zasilania sterownika.**
2) **Połączenie USB podatne jest na zakłócenia w sieci zasilającej oraz na zakłócenia elektromagnetyczne występujące w warunkach przemysłowych. W przypadku pojawiania się problemów z komunikacją należy zastosować dodatkowe elementy zabezpieczające w postaci:**
   - Stosowania filtrów sieciowych,
   - Stosowania przewodu USB dobrej jakości, o długości < 1,5m wyposażonego w koraliki ferrytowe
   - Stosowania optoizolowanych HUBów USB po stronie komputera PC

Przy większych zakłóceniach może zdarzyć się, że komunikacja nie będzie możliwa.

MIC488 komunikuje się przy wykorzystaniu portu USB (1.1, 2.0).
Znane są problemy z kompatybilnością portu USB 3.0 (niebieski kolor gniazda) w systemie Windows 7 podczas komunikacji z urządzeniami USB HID. W przypadku problemów z komunikacją należy podłączyć sterownik do portu USB 2.0.

Połączenie szeregowe RS485 (COM0)

Port COM0 sterownika MIC488 pozwala na połączenie z komputerem PC za pomocą konwertera USB<->RS485. Sygnały z konwertera RS485 (A,B) należy podłączyć do sygnałów A i B portu COM0 sterownika.
W celu nawiązania połączenia należy w programie MIC488-PC na pasku narzędzi wybrać ***Połączenie -> Ustawienia***. W otwartym oknie wybieramy połączenie ***COM0***. Jeśli znamy numer portu COM po stronie PC wybieramy ten port oraz ustawiamy prędkość transmisji (38400) i wciskamy przycisk ***Połącz***. Jeśli nie wciskamy przycisk ***Wyszukaj***.

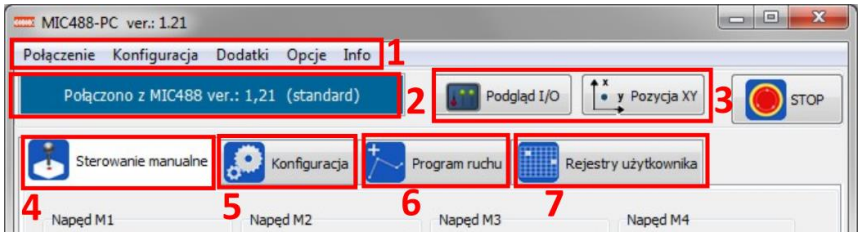


Rys. 11 Okno ustawień komunikacji z PC.

Program MIC488-PC należy uruchomić jako **administrator** (klikając prawym przyciskiem myszy na ikonę programu i wybierając opcję „***Uruchom jako administrator***”. W przeciwnym wypadku program może nie wykryć zainstalowanych portów COM.

4.2 Opis programu

Podstawowe funkcje programu dostępne są w zakładkach głównego okna pokazanego poniżej:



Rys. 12 Pasek narzędzi.

1) Pasek narzędzi
2) Sygnalizacja połączenia ze sterownikiem
3) Uruchomienie okna z podglądem stanu wejść-wyjść urządzenia oraz pozycją napędów M1/M2
4) Zakładka służąca do manualnego sterowania napędami
5) Zakładka z ustawieniami sterownika
6) Zakładka związana z tworzeniem programów ruchu.
7) Zakładka z podglądem rejestrów użytkownika

Pasek narzędzi:

- Połączenie – konfiguracja połączenia sterownika z programem
- Konfiguracja – zapis i odczyt ustawień sterownika z pliku. Przywrócenie ustawień fabrycznych sterownika.
- Dodatki - *Podgląd I/O* (funkcja dostępna także pod przyciskiem (3)), *Podgląd pozycji X/Y* (funkcja dostępna także pod przyciskiem (3)), *Sterowanie JOG* (funkcja dostępna także w zakładce „Sterowanie manualne”),
- Opcje – ustawienia programu MIC488-PC
- Info – informacje o programie MIC488-PC.

5. Konfiguracja sterownika

5.1 Informacje wstępne - parametry ruchu

Podczas zadawania prędkości lub pozycji dla napędów sterownik generuję rampę (rozpędzanie/hamowanie) w celu uzyskania płynności ruchu. Uwzględniane są następujące parametry:

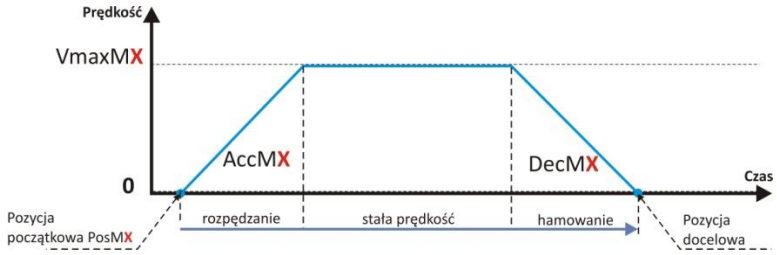
Parametr	Opis	Jednostki
VmaxM**X**	Prędkość maksymalna w trybie zadanej pozycji. Podczas ruchu na nową pozycję prędkość ta nie zostanie przekroczona.	[j/s]
AccM**X**	Przyspieszenie dla rozpędzania w trybie zadanej pozycji. Przyspieszenie dla rozpędzania i hamowania w trybie zadanej prędkości.	[j/s^2]
DecM**X**	Przyspieszenie dla hamowania w trybie zadanej pozycji.	[j/s^2]

gdzie: X – oznacza numer danego napędu (1…4), j – jednostka ruchu zależna od konfiguracji napędu (np. obr.,mm itp.)

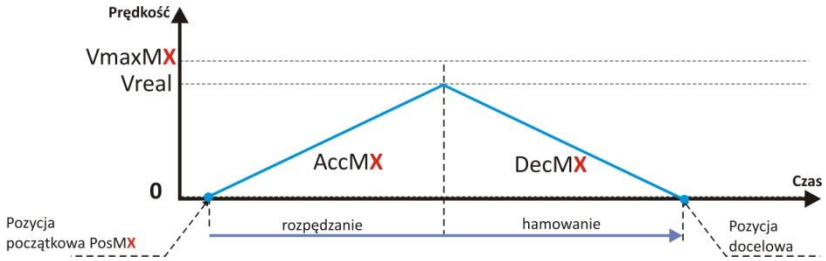
Rampa dla zadanej pozycji

- **Rozpędzanie**: rozpoczęcie ruchu następuje od prędkości 0 do prędkości określonej parametrem VmaxM**X** (prędkość maksymalna dla pozycji) z przyspieszeniem AccM**X.**
- **Stała prędkość** VmaxM**X** (tylko wówczas jeśli docelowa droga będzie dłuższa niż droga potrzebna na rozpędzenie i wyhamowanie z zadanymi parametrami AccM**X** i DecM**X**).
- **Hamowanie**: zmniejszanie prędkości do 0 z opóźnieniem DecM**X**, aż do osiągnięcia pozycji docelowej.

Podczas zadawania pozycji w zależności od wartości przyspieszeń dla rozpędzania i hamowania napęd może osiągnąć prędkość maksymalną jeśli łączna droga potrzebna na rozpędzenie i wyhamowanie będzie mniejsza niż droga zadana do przebycia. Rys. 13 pokazuje przypadek osiągnięcia prędkości maks., natomiast Rys. 14 pokazuje rampę z ograniczoną prędkością (prędkość VMax jest niemożliwa do osiągnięcia).



Rys. 13 Przykład rampy pozycji – prędkość Vmax możliwa do osiągnięcia.

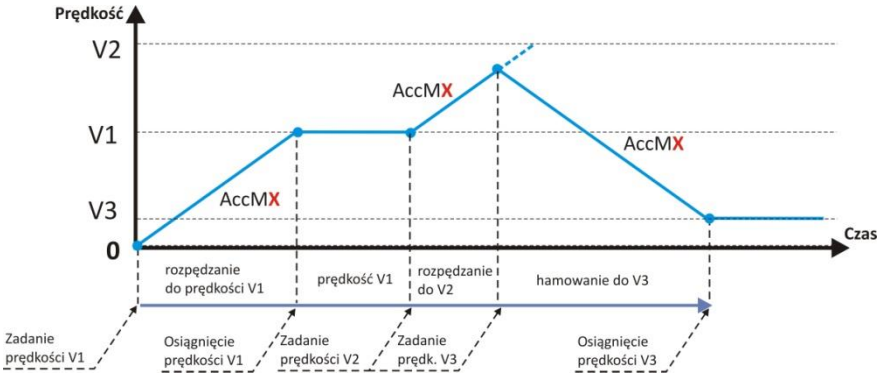


Rys. 14 Przykład rampy pozycji – prędkość Vmax niemożliwa do osiągnięcia.

Uwagi:

- Zadanie nowej pozycji może być zrealizowane podczas ruchu napędu, ale zadana pozycja nie powinna być mniejsza od aktualnej.
- Zadanie nowej pozycji, mniejszej od aktualnej, podczas trwającego ruchu spowoduje nawrót napędu bez rampy. **Taka sytuacja może spowodować utratę pozycji dla napędu krokowego podczas pracy bez enkodera.**
- Zadanie prędkości podczas trwania ruchu na pozycję spowoduje ustawienie nowej prędkości z rampą AccM**X** (dla wyhamowania lub przyspieszenia do prędkości zadanej).

Rampa dla zadanej prędkości



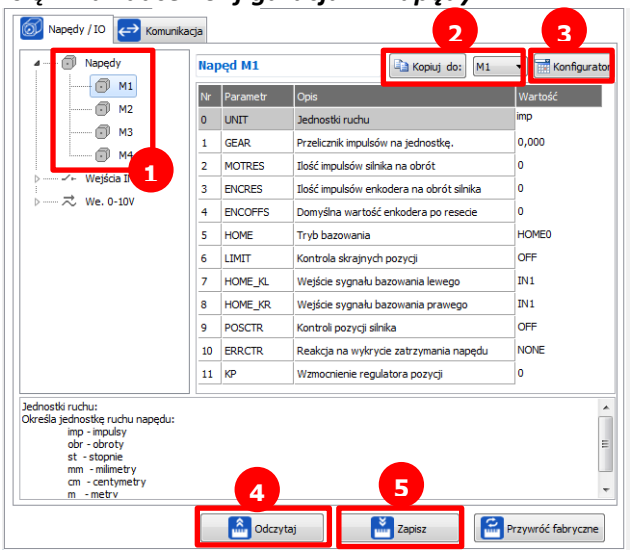
Rys. 15 Przykład rampy dla zadanej prędkości.

Uwagi:

- Rampa dla prędkości (dla rozpędzania i hamowania) realizowana jest zawsze z parametrem AccM**X.**
- **Podczas trwania ruchu nie jest możliwa zmiana parametru AccMX.** Nowa wartość AccMX uwzględniana jest podczas ruchu od prędkości zerowej.
- Zadanie nowej prędkości podczas ruchu spowoduje osiągnięcie nowej prędkości z uwzględnieniem rampy AccMX.

5.2 Konfiguracja napędów

Konfiguracja napędów odbywa się w zakładce ***Konfiguracja*** -> ***Napędy***.



Rys. 16 Okno konfiguracji – ustawienia napędów.

1) Wybór napędu do konfiguracji (M1… M4).
2) Skopiowanie ustawień między napędami.
3) Konfigurator ułatwiający wprowadzenie parametrów napędów.
4) Odczytanie ustawień ze sterownika (dotyczy wszystkich ustawień urządzenia).
5) Zapisanie ustawień do sterownika (dotyczy wszystkich ustawień urządzenia).

Parametry konfiguracyjne napędu:

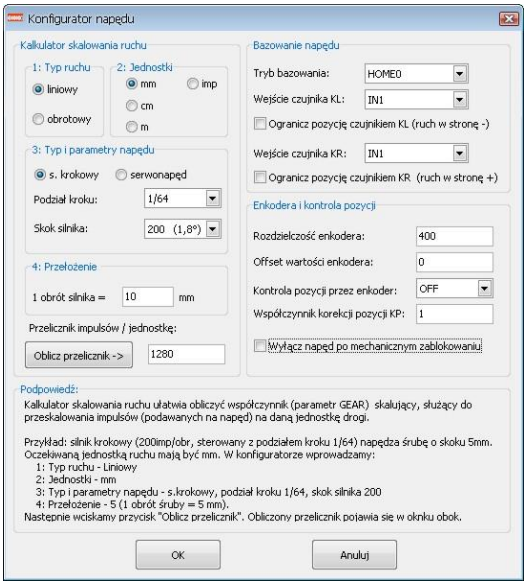
Numer	Parametr	Opis
0	UNIT	Ustawienie jednostek ruchu wyświetlanych w programie (nie mają wpływu na wartości przeliczeń).
1	GEAR	Przelicznik impulsów na jednostkę. Określa ilość impulsów, które sterownik ma wygenerować by napęd wykonał 1 jednostkę ruchu. (wartość zmiennoprzecinkowa.)
2	MOTRES	Ilość impulsów silnika na pełen obrót mechaniczny lub elektryczny (dla silników liniowych). Parametr wymagany dla pracy z enkoderem.

		(maks. wartość 32-bit.)
3	ENCRES	Rozdzielczość enkodera Określa ilość impulsów z enkodera przypadającą na pełen obrót. UWAGA: **Wprowadzaną wartość rozdzielczości należy przemnożyć x 4** (uwzględnienie kwadratury) (maks. wartość 32-bit.)
4	ENCOFFS	Domyślna wartość enkodera po resecie.
5	HOME	Określa tryb bazowania napędu (dokładny opis w rozdziale 5.2.1)
6	LIMIT	Ustawienia dotyczą kontroli skrajnych pozycji napędu za pomocą czujników krańcowych: OFF – brak kontroli skrajnych pozycji KL – kontrola skrajnej lewej pozycji (dla ruchu napędu w stronę zmniejszania pozycji) KR – kontrola skrajnej prawej pozycji (dla ruchu napędu w stronę zwiększania pozycji) KL + KR – kontrola obydwóch skrajnych pozycji
7	LIMIT_L	Programowy limit pozycji dodatniej (0 – limit wyłączony)
8	LIMIT_R	Programowy limit pozycji ujemnej (0 – limit wyłączony)
9	HOME_KL	Ustawienie wejścia dla czujnika bazowania (także kontroli pozycji) lewego.
10	HOME_KR	Ustawienie wejścia dla czujnika bazowania (także kontroli pozycji) prawego.
11	POSCTR	Tryb kontroli pozycji z enkodera: OFF – kontrola pozycji z enkodera wyłączona ENCODER – kontrola pozycji z enkodera włączona ECFOLLOW - niedostępne
12	ERRCTR	Reakcja na wykrycie zatrzymania napędu (tylko dla pracy z enkoderem): OFF –brak reakcji ERRMODE0 – zatrzymanie napędu, zmiana statusu napędu (PosMX = M_POS_ERR) ERRMODE1 – jw. + pauza wykonywanego programu ERRMODE2 – jw. + włączenie wyjścia OUT8.
13	ERR_TIME	Czas zablokowania napędu do błędu w ms. Domyślna wartość 2000 ms.
14	KP	Wzmocnienie regulatora pozycji. Określa wzmocnienie regulatora pozycji dla trybu kontroli pozycji z enkodera. Im większa wartość tym szybsze ustalenie pozycji po utracie kroku. Domyślnie 50.
15	KP_SPEED	Procent prędkości maks. korekcji pozycji. Określa procent zadanej prędkości maksymalnej (VMax) do którego ograniczona zostanie prędkość podczas korekcji pozycji. Domyślna wartość 10 (%)

- prametry wymagane w przypadku pracy z enkoderem do kontroli pozycji napędu.

Dla ułatwienia konfiguracji napędu służy okno uruchamiane przyciskiem 

, w którym w kolejnych krokach podaje się wymagane parametry.



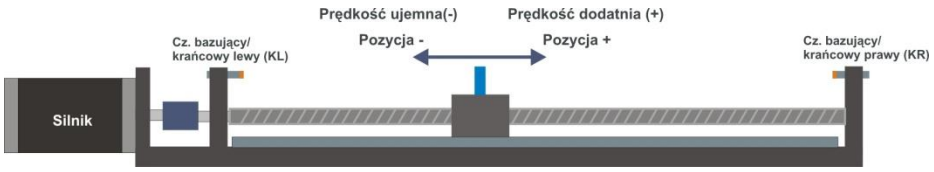
Rys. 17 Okno konfiguratora napędu.

Po wprowadzeniu parametrów należy wcisnąć przycisk „***Oblicz przelicznik ->”,*** a następnie kliknąć ***OK***. Po zamknięciu się okna wprowadzone parametry zostaną dodane do ustawień napędu. By wprowadzić ustawienia w sterowniku należy wcisnąć przycisk „***Zapisz do urządzenia***”.

Otworzenie konfiguratora nie powoduje wczytanie do niego aktualnych ustawień napędu.

5.2.1 Tryby bazowania i ograniczenia ruchu

Sterownik pozwala na ustawienie dwóch sygnałów wejściowych (dla każdego napędu niezależnie), które mogą służyć do bazowania i/lub ograniczenia skrajnych pozycji podczas ruchu.
Sygnały nazwane są jako **KL** (bazowanie lub ograniczenie ruchu podczas ruchu w kierunku ujemnym (zmniejszanie pozycji) oraz **KR** (w kierunku przeciwnym).



Rys. 18 Sygnały krańcowe i kierunki ruchu napędu.

Bazowanie napędu może odbywać się w jednym z poniższych trybów, ustawianych w nastawie ***HOME***:

- **HOME0:** Bazowanie proste. Zatrzymanie napędu następuje od razu po dojeździe do czujnika krańcowego.
- **HOME1**: Bazowanie precyzyjne tryb 1. Napęd dojeżdża do czujnika krańcowego, zatrzymuje się, a następnie powoli cofa aż do zaniku sygnału z czujnika (cofnięcie przed czujnik).
- **HOME2**: Bazowanie precyzyjne tryb 2. Napęd dojeżdża do czujnika krańcowego, zatrzymuje się, a następnie rusza powoli, aż do zaniku sygnału z czujnika (przejazd za czujnik).
- **HOME3**: Bazowanie enkoder tryb 1. Napęd dojeżdża do mechanicznej blokady, zatrzymuje się, a następnie powoli cofa, aż do wykrycia sygnału z kanału C enkodera (INDEX).
- **HOME4**: Bazowanie enkoder tryb 2. Napęd dojeżdża do czujnika krańcowego, zatrzymuje się, a następnie powoli cofa, aż do wykrycia sygnału z kanału C enkodera (INDEX).

Dla trybów HOME1 do HOME4 ruch powrotny odbywa się z prędkością 20% prędkości bazowania.

Ograniczenie pozycji przez sygnały krańcowe konfiguruje się za pomocą nastawy ***LIMIT.*** Napęd zatrzymuje się, gdy wykryty zostanie sygnał krańcowy. Możliwy jest wówczas ruch tylko w przeciwną stronę. Nie należy ustawiać tego samego wejścia gdy aktywne są dwa sygnały krańcowe (tryb KL+KR) – spowoduje to zablokowanie pracy napędu po aktywacji czujnika.

Możliwe jest także programowe ograniczenie pozycji (rejestry LimL oraz LimR dostępne przez Modbus lub program WBCprog). Domyślnie rejestry te ustawione są na 0 co oznacza wyłączenie limitów.

5.2.2 Kontrola pozycji z enkoderem

MIC488 może kontrolować pozycję silnika w oparciu o zewnętrzny enkoder podłączany do wejść ENC (ENC1…4 dla kolejnych napędów M1…M4). Sposób podłączenia enkodera pokazano w rozdziale 3.4.
Do prawidłowej współpracy enkodera z silnikiem konieczne jest wprowadzenie poprawnych parametrów *MOTRES* (określającego ilość impulsów silnika na pełen obrót) oraz *ENCRES* (określającego ilość impulsów z enkodera na pełen obrót).
Przykładowo:

- dla silnika krokowego o skoku 1,8° i sterownika z podziałem 1/64 *MOTRES* będzie równy 200*64 =**12800**
- dla enkodera o rozdzielczości podanej na obudowie 400 *ENCRES* będzie równy 4 * 400 = **1600** (mnożymy x 4 by uwzględnić kwadraturę enkodera)

Po wprowadzeniu parametrów należy przetestować napęd zadając np. pozycję absolutną równą 1. Możliwe są wówczas następujące sytuacje:

Zachowanie napędu	Opis
Wykonanie pełnego obrotu.	Parametry i podłączenie enkodera poprawne.
Napęd wykona pełen obrót, a następnie cofnie się.	Wprowadzona za mała rozdzielczość enkodera (np. nie uwzględniona została kwadratura) lub za duża rozdzielczość silnika.
Napęd wykona więcej jak pełen obrót i zatrzyma się.	Wprowadzona za duża rozdzielczość enkodera lub za mała rozdzielczość silnika.
Napęd wykona pełen obrót i będzie kontynuował pracę bez zatrzymania.	Błędnie podłączony enkoder lub silnik. Należy zamienić jedną z faz silnika (np. A z /A) lub kanały z enkodera (A z B)
Napęd wykona pełen obrót, ale podczas utraty pozycji wolno ją koryguje.	Należy zwiększyć wartość regulatora pozycji *KP.*

5.2.3 Przykładowa konfiguracja

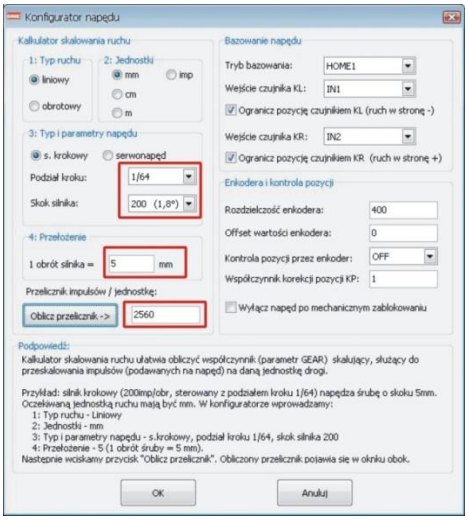
W przykładowej aplikacji do napędu użyty został silnik krokowy (o skoku **1,8°** czyli **200 kroków/obrót**), sterowany ze sterownika o podziale kroku **1/64**. Silnik napędza śrubę o skoku **5mm/obrót**. Do kontroli pozycji użyte zostały 2 czujniki – lewy jako czujnik bazujący i ograniczający ruch, oraz prawy jako czujnik ograniczający ruch z prawej strony. Bazowanie ma być wykonane precyzyjnie z dojazdem do czujnika oraz odjazdem do zaniku sygnału z czujnika. Jednostką dla sterowania mają być **mm**.

Konfiguracja (bez enkodera)
Sterownik silnika podłączony zostaje do wyjścia M1 MIC488. Sygnał czujnika KL podłączamy do wejścia IN1, a KR do wejścia IN2. W ustawieniach napędu M1 (zakładka ***Konfiguracja*** -> ***Napędy*** -> ***M1***) wprowadzamy następujące parametry:

(0) UNIT: mm
(1) GEAR: 64 * 200 / 5 mm = 2560
(5) HOME: HOME1
(6) LIMIT: KL + KR
(9) HOME_KL: IN1
(10) HOME_KR: IN2

Pozostałe parametry są nieistotne dla pracy bez enkodera.
Można także skorzystać z kalkulatora napędu (wciskając przycisk ***Konfigurator***). Po jego uruchomieniu wprowadzamy niezbędne parametry:

1) Typ ruchu: **liniowy,**
2) Jednostki: **mm,**
3) Typ i parametry napędu: **s. krokowy, podział kroku 1/64, skok silnika 200 (1,8°),**
4) Przełożenie: **5 mm.**

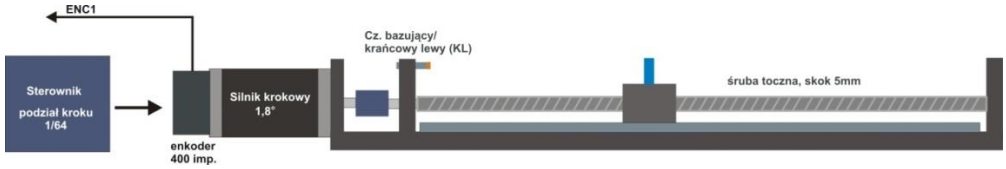


Rys. 19 Okno konfiguratora napędu z wprowadzonymi parametrami.

Następnie wciskamy przycisk „***Oblicz przelicznik ->***”. Po zamknięciu konfiguratora przyciskiem „***OK***” ustawienia zostaną wpisane w okno z parametrami napędu M1.

Po wprowadzeniu ustawień należy wcisnąć przycisk „***Zapisz do urządzenia***” by ustawienia zostały przesłane do sterownika.

Konfiguracja (z enkoderem)
W poniższym przypadku do napędu został dołożony enkoder inkrementalny o rozdzielczości 400 imp./obrót. Usunięty został także prawy czujnik krańcowy.



Rys. 20 Przykładowa konfiguracja napędu (praca z enkoderem).

Należy wprowadzić następujące parametry:

(0) UNIT: mm
(1) GEAR: 64 * 200 / 5 mm = 2560
(2) MOTRES: 64*200 = 12800
(3) ENCRES: 400 * 4 = 1600 (x4 by uwzględnić kwadraturę sygnału enkodera)
(4) ENCOFFS: 0 (domyślna pozycja po resecie)
(5) HOME: HOME1
(6) LIMIT: KL
(7) LIMIT_L: 0
(8) LIMIT_R: 0
(9) HOME_KL: IN1
(10) HOME_KR: OFF
(11)POSCTR: ENCODER
(12) ERRCTR: 50 (określa reakcję sterownika w przypadku mechanicznego zatrzymania napędu).
(13) ERR_TIME: Domyślnie 2000 ms (określa czas zablokowania / błędu pozycji napędu do zasygnalizowania błędu).
(14) KP: należy dobrać eksperymentalnie. Domyślna wartość to 10.
(15) KP_SPEED: należy dobrać eksperymentalnie. Domyślna wartość to 10.

Jeśli sterowanie napędem z kontrolą pozycji z enkodera nie działa poprawnie (np. napęd przejeżdża zadaną pozycję) należy zamienić sygnały A z B enkodera.

5.2.4 Kontrola stanu napędu

Stan napędu może być kontrolowany za pomocą:
- Informacji w oknie sterowania manualnego,
- Rejestrów MODBUS: *M1_STATUS (10010)… M4_STATUS (10014)*
- Rejestrów w programie WBCprog: *StatM1…StatM4*

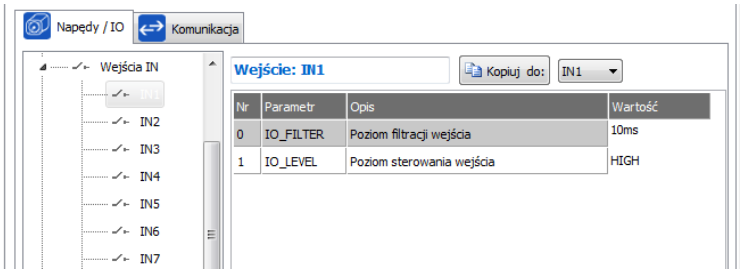
W tabeli poniżej zestawiono wartości rejestru statusowego i odpowiadające im stany napędu.

Wartość rejestru	Opis
0	Napęd wyłączony (sygnał EN = 0)
1	Napęd włączony, ale nie znajduje się w ruchu (sygnał EN aktywny)
2	Napęd znajduje się w trybie zadanej prędkości
3	Napęd znajduje się w trybie ruchu do zadanej pozycji
4	Napęd dojechał do zadanej pozycji
5	Błąd dojazdu do zadanej pozycji (dla pracy z enkoderem)
6	Napęd w trybie bazowania
7	-
8	Napęd w trybie korekcji pozycji (dla pracy z enkoderem)
9	Napęd osiągnął krańcową pozycję L podczas ruchu w stronę zmniejszającą

	pozycję (programową, lub przez aktywację sygnału krańcowego KL)
10	Napęd osiągnął krańcową pozycję R podczas ruchu w stronę zwiększającą pozycję (programową, lub przez aktywację sygnału krańcowego KL)
11	Napęd w trybie interpolacji liniowej (funkcja MOVELIN2)
12	Napęd w trybie interpolacji kołowej (funkcje DOCIRC, DOARC)

5.3 Konfiguracja wejść cyfrowych

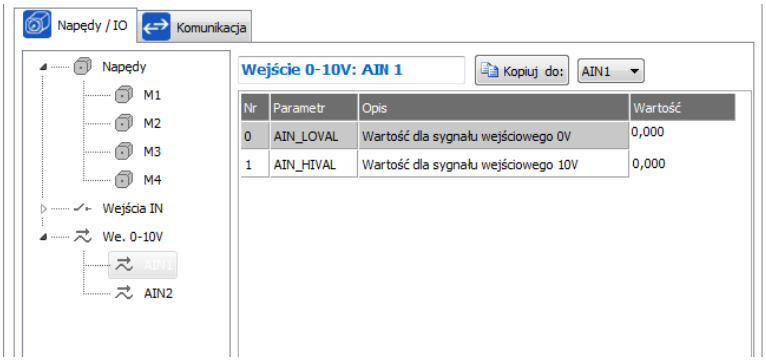
Konfiguracja ta dotyczy tylko wejść IN1…IN8 używanych w programie jako wejścia uniwersalne. Ustawienia nie mają wpływu na pracę wejść w funkcji kontroli pozycji/bazowania napędu.



Rys. 21 Ustawienia wejść.

Numer	Nazwa	Opis
0	IO_FILTER	Poziom filtracji wejścia. Określa minimalny czas przez jaki musi być podany sygnał na wejście by było ono aktywne.
1	IO_LEVEL	Określa jaki poziom jest stanem wysokim dla wejścia. Poziom HIGH oznacza, że wejście jest aktywne po podaniu napięcia na wejście. Poziom LOW oznacza, że wejście jest aktywne przy braku sygnału na wejściu.

5.4 Konfiguracja wejść analogowych



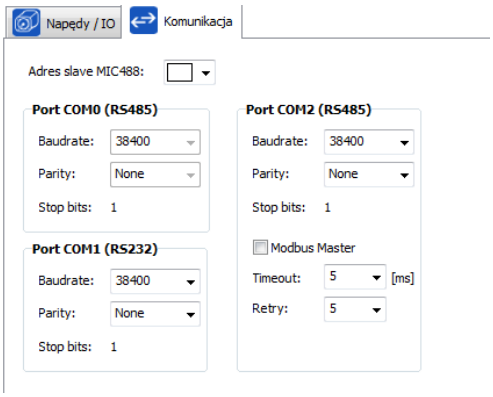
Rys. 22 Ustawienia wejść analogowych.

Numer	Nazwa	Opis
0	AIN_LOVAL	Wartość dla sygnału wejściowego 0V
1	AIN_HIVAL	Wartość dla sygnału wejściowego 10V

Ustawienia te pozwalają przeskalować wejściowy sygnał napięciowy 0…10V na inną wartość, która może być wykorzystana w programie ruchu do np. bezpośredniego sterowania prędkością napędu itp.
Przykładowo, dla napięcia wejściowego 0V wartość AIN ma wynosić -25, a dla 10V +25. Należy wówczas wprowadzić:

AIN_LOVAL: -25
AIN_HIVAL: +25

5.5 Konfiguracja komunikacji RS232/RS485



Rys. 23 Ustawienia komunikacji RS232/RS485.

Port COM0 standardowo służy do komunikacji z programem MIC488-PC.
Porty COM1 (RS232) i COM2(RS485) pozwalają na komunikację z urządzeniami w protokole MODBUS-RTU. Ponadto port COM2 może pracować w trybie Modbus Master – możliwe jest wówczas sterowanie do 16 urządzeń podłączonych do tej magistrali. **Timeout** określa czas oczekiwania na odpowiedź od urządzenia slave, **Retry** określa ilość retransmisji w przypadku nieotrzymania odpowiedzi od urządzenia slave.

6. Sterowanie manualne i diagnostyka

6.1 Sterowanie manualne napędami

Sterowanie manualne pozwala przetestować napędy oraz zadać komendy ruchu dla poszczególnych napędów.

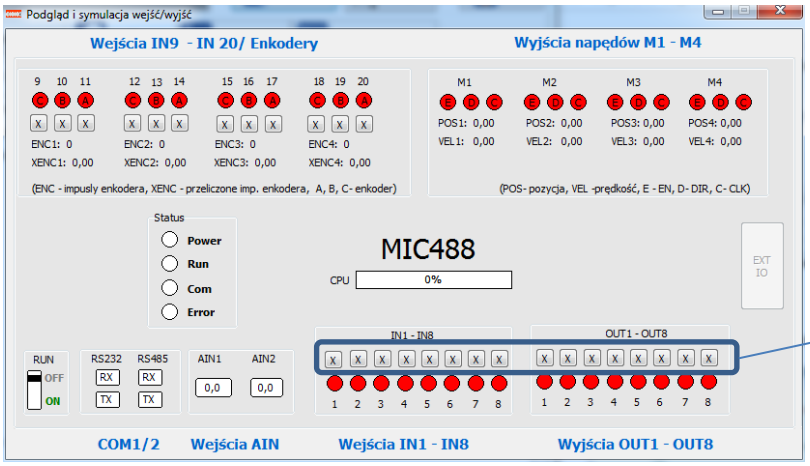
Rys. 24 Okno sterowania manualnego napędami.

1) Pokazuje informacje o napędach (status, prędkość aktualna, pozycja aktualna, pozycja z enkodera). Przycisk 0<- pozwala wyzerować pozycję.
2) Pozwala ustawić testowo parametry rampy dla każdego z napędów. Przycisk „<-„ zapisuje parametry ramp.
3) Przyciski sterujące:
   - Enable – włączenie/wyłączenie napędu (wyjścia EN).
   - Napęd STOP – zatrzymanie napędu
   - Przesuw JOG – załączenie napędu w lewą / prawą stronę tak długo jak wciśnięty jest przycisk
4) Sterowanie – pozwala wybrać napędy dla których mają zostać uruchomione podstawowe funkcje ruchu:
   - Bazowanie – powoduje wykonanie bazowania napędu z wprowadzoną prędkością. Wartość ujemna prędkości powoduje bazowanie w kierunku krańcówki lewej (KL), wartość dodatnia w kierunku krańcówki prawej (KR)
   - Prędkość ABS1/2 –pozwala zadać prędkość (napęd osiągnie zadaną prędkość).
   - Prędkość REL – pozwala zadać prędkość relatywnie. Nastąpi zwiększenie (+) lub zmniejszenie (-) aktualnej prędkości o zadaną wartość.
   - Pozycja ABS1/2 – pozwala zadać pozycję (napęd osiągnie zadaną pozycję absolutną).

- Pozycja REL – pozwala zadać pozycję relatywnie. Nastąpi zwiększenie (+) lub zmniejszenie (-) aktualnej pozycji o zadaną wartość.

6.2 Diagnostyka

Przycisk ***Podgląd wejść-wyjść*** pozwala na szybką diagnostykę sterownika.



Rys. 25 Okno podglądu i symulacji wejść-wyjść.

- wejście/wyjście nieaktywne (stan niski)
- wejście/wyjście aktywne (stan wysoki)

Za pomocą przycisków znajdujących się nad wejściami IN1..IN20 oraz wyjściami OUT1..OUT8 możliwe jest symulowanie ich działania. Symulacja wejść działa także w funkcjach czujników bazujących i krańcowych.

6.3 Sygnalizacja błędów

W zależności od stanu diod MIC488 może sygnalizować następujące błędy:

Stan diod				Typ błędu	Kasowanie błędu
RUN	COM	ERROR	OUTERR		
-	-	-	ON	Przeciążenie wyjścia	Reset zasilania
OFF	-	ON	-	Błąd programu	Ponowny start programu
OFF	-	MIGA	-	Błąd pozycji napędu	-
MIGA	-	MIGA	-	Błąd pozycji napędu. Program w trakcie pracy zostaje wstrzymany.	Ponowny start programu
MIGA	MIGA	MIGA	-	Błąd krytyczny sterownika	Po kilku sekundach sterownik automatycznie zresetuje się.

Błąd programu (dioda ERROR włączona) może pojawić się gdy:
- Zapis / odczyt z nieistniejącego rejestru (np. *OUT10*, *AccM5*)
- Błąd sumy kontrolnej programu
- Uruchomienie przerwania bez etykiety przerwania
- Błąd komunikacji Modbus master

Błąd pozycji (dioda ERROR miga) może pojawić się gdy:
- Podczas pracy z enkoderem, gdy napęd nie może osiągnąć zadanej pozycji przez 3 sek.

7. Programowanie sterownika

7.1 Wstęp

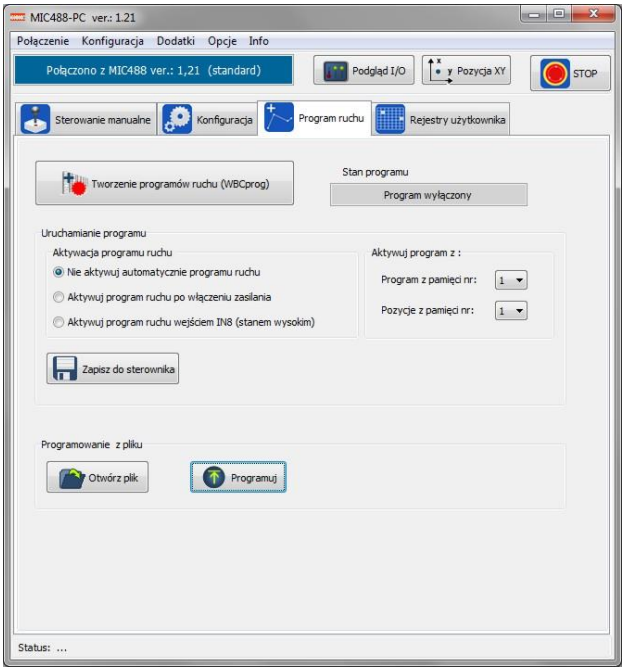
Programowanie sterownika odbywa się przy wykorzystaniu wbudowanej w MIC488-PC aplikacji **WBCprog**. Aplikację uruchamia się z zakładki „***Program ruchu”*** przyciskiem „***Tworzenie programów ruchu (WBCprog)***” lub z paska narzędzi: „***Opcje -> Programy ruchu (WBCprog)***”.

Programowanie polega na wprowadzaniu w języku tekstowym, zwanym **WBL** (Wobit Basic Language), prostych komend np.: „*PABS M1 10*” (ruch na pozycję absolutną 10 napędu M1) lub „*SET OUT3 = ON*” (włączenie wyjścia O3).

Język ten dzięki prostym komendom tekstowym pozwala w intuicyjny i szybki sposób tworzyć programy dla sterownika MIC488. Z poziomu stworzonego programu możliwe jest dowolne sterowanie ruchem napędów, sterowanie uniwersalnymi wyjściami, reakcja na wejścia, zliczanie impulsów z enkoderów, funkcje opóźnień czasowych, proste operacje matematyczne, operacje na zmiennych dostępnych przez rejestry MODBUS itp.

Użytkownik ma możliwość zapisania do pamięci sterownika programu składającego się z maks. 4000 komend. Oprócz tego możliwe jest zapisanie 8 niezależnych tablic pozycji składających się z 200 pozycji każda (jedna pozycja w tablicy zawiera pozycje dla 4 napędów M1…M4).

Wybrany program z pamięci sterownika może być uruchomiony automatycznie po włączeniu zasilania sterownika lub po aktywacji wejściem IN8. Takiej konfiguracji dokonuje się w oknie zakładki „***Program ruchu”***.

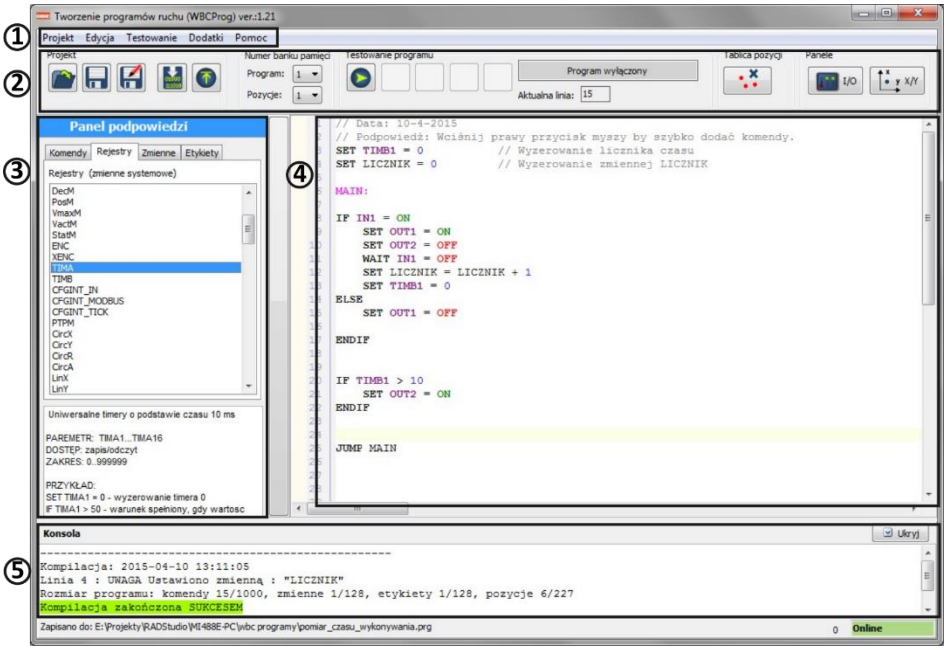


Rys. 26 Okno z ustawieniami uruchamiania programu z pamięci sterownika.

Ponadto możliwe jest uruchomienie wybranego programu oraz tablicy pozycji za pomocą odpowiednich rejestrów MODBUS.

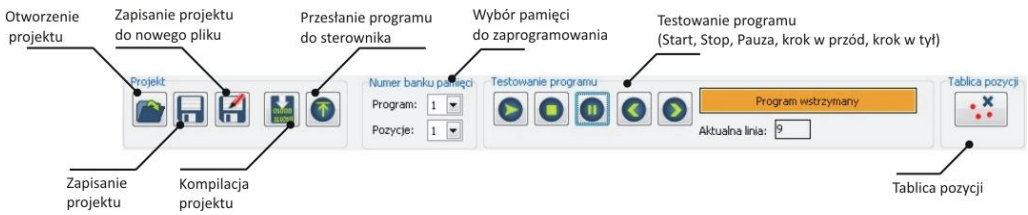
7.2 Opis programu WBCprog

7.2.1 Okno główne



Rys. 27 Główne okno programu WBCprog.

1) **Pasek narzędzi:**
   Zawiera dostęp do wszystkich funkcji programu.
2) **Pasek skrótów:**
   Zawiera skróty do najważniejszych funkcji programu. W części „***testowanie programu***" znajdują się przyciski pozwalające na uruchomienie programu w celu przetestowania jego pracy. Obok przycisków wyświetlana jest także informacja o aktualnym stanie programu oraz ewentualnych błędach.



Rys. 1 Pasek skrótów.

3) **Panel podpowiedzi:**
   Zawiera spis wszystkich komend oraz rejestrów kompilatora WBCprog (zakładki ***Komendy*** i ***Rejestry***). Pokazuje także użyte w programie własne zmienne (zakładka ***Zmienne***) oraz etykiety (zakładka ***Etykiety***).
4) **Okno edytora kodu:**
   Edytor tekstowy służący do wprowadzania komend. Edytor posiada funkcję wyróżniania wprowadzanych komend/zmiennych ułatwiając pisanie programu.
5) **Okno informacji o kompilacji programu:**
   Pokazuje informacje o skompilowanym programie (rozmiar programu itp.) lub informacje o błędach.

7.2.2 Zapisywanie i otwieranie projektu

Przycisk ***Zapisz*** powoduje zapisanie zmian w projekcie.

Przycisk ***Zapisz jako*** powoduje zapisanie projektu do nowego pliku.

Przy zapisywaniu projektu tworzone są trzy pliki (o takiej samej nazwie, ale innym rozszerzeniu):

- Plik główny z programem (rozszerzenie .**prg**).

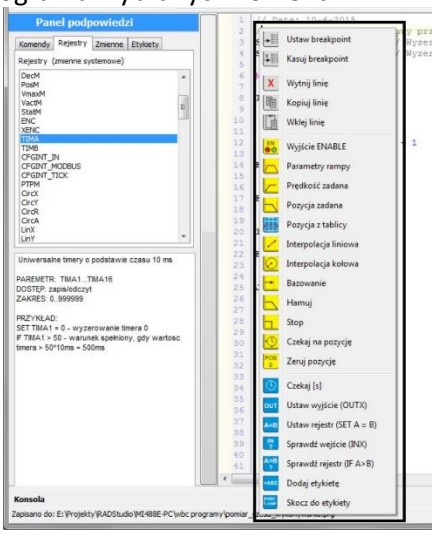
- Plik z aktualnymi ustawieniami sterownika (rozszerzenie **.cfg**).
- Plik z pozycjami z tablicy pozycji (rozszerzenie **.csv**).

Przycisk ***Otwórz*** powoduje otworzenie projektu (otwieramy plik z rozszerzeniem .prg).

Jeśli w katalogu z projektem nie ma pliku konfiguracyjnego (***cfg**) lub pliku z tablicą pozycji (***csv**) program wyświetli komunikat o braku tych plików. Zapisanie projektu spowoduje automatyczne utworzenie tych plików.

7.2.3 Menu szybkich komend

Kliknięcie prawym przyciskiem myszy w wybranej linii programu w oknie edytora (5) powoduje pojawienie się menu, które pozwala na dodanie do programu wybranych komend.



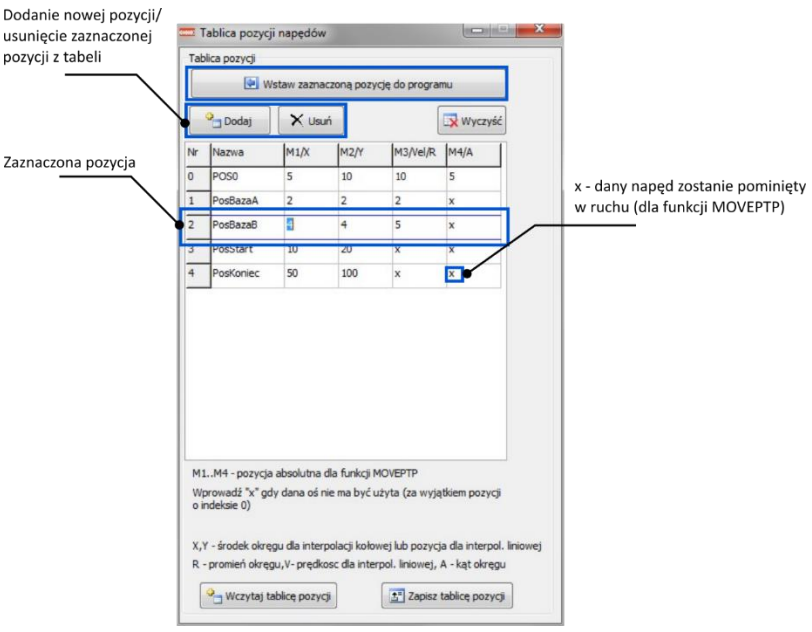
Rys. 2 Menu szybkich komend.

7.2.4 Tablica pozycji

Pozwala zapamiętać do 200 rekordów, które mogą być:

- niezależnymi pozycjami dla 4 napędów
- parametrami dla interpolacji liniowej
- parametrami okręgu dla interpolacji kołowej (realizowanej przez napędy M1 i M2)

Tablica uruchamiana jest przyciskiem znajdującym się na pasku skrótów.



Rys. 28 Okno „Tablica pozycji”.

Dla niezależnego sterowania poszczególnymi napędami (funkcja **MOVEPTP**) używane są kolumny **M1**…**M4** odpowiadające pozycjom poszczególnych napędów.

Dla interpolacji liniowej (funkcja **MOVELIN oraz MOVELIN_REL**) używane są kolumny **X** i **Y** (określające punkt końcowy ruchu) oraz kolumna **Vel** określające prędkość ruchu liniowego.

Dla drugiej interpolacji liniowej (funkcja **MOVELIN2** oraz **MOVELIN2_REL**) używane są kolumny **M1**…**M4** odpowiadające pozycjom końcowym poszczególnych napędów.

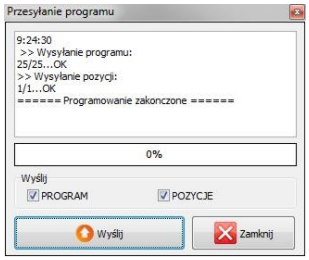
Dla interpolacji kołowej (funkcja **MOVETOCIRC** oraz **DOCIRC**) używane są kolumny **X** i **Y** (określające punkt środka okręgu) , kolumna **R** (określająca promień okręgu) oraz kolumna A (określająca kąt ruchu po okręgu).

Tablica pozycji jest zapisywana podczas zapisywania projektu w postaci pliku *.csv, który może być otwierany i edytowany za pomocą arkusza kalkulacyjnego (np. programu Microsoft Excel). Można także wczytać przygotowany wcześniej plik z pozycjami do tablicy pozycji za pomocą przycisku „***Wczytaj tablicę pozycji***”.

7.2.5 Przesyłanie programu do sterownika

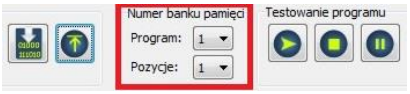
W celu zaprogramowania sterownika należy wcisnąć przycisk ***Wyślij do sterownika*** . Otworzy się okno, w którym użytkownik określa co ma zostać przesłane:

- **PROGRAM** – jeśli chcemy przesłać program
- **POZYCJE** – jeśli chcemy przesłać tablice pozycji



Rys. 29 Okno przesyłania programu do sterownika.

Po wciśnięciu przycisku ***Wyślij*** nastąpi przesłanie programu do sterownika.
Program / pozycje z tablicy pozycji zostaną zapisane w pamięci sterownika o numerze ustawionym na pasku skrótów:

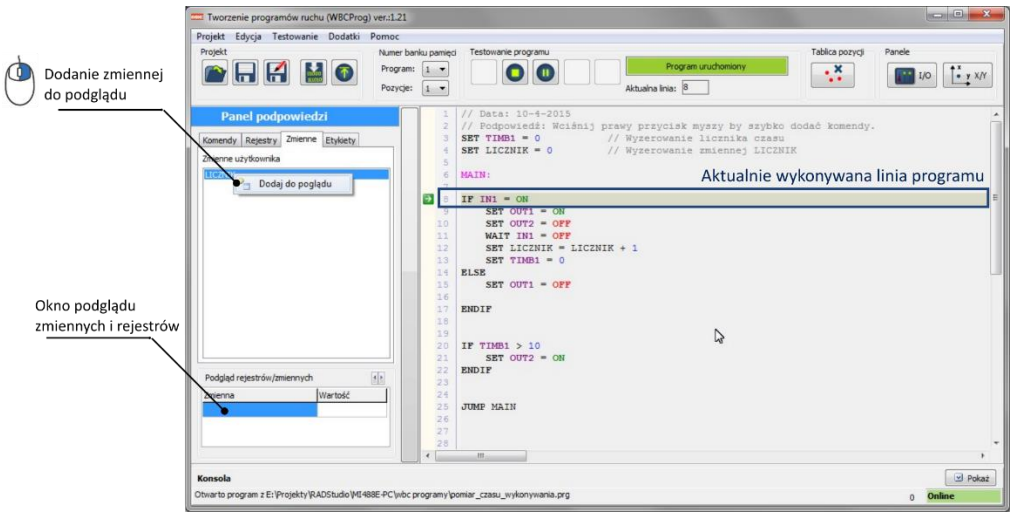


Przed wysłaniem do sterownika program jest automatycznie kompilowany. Nie ma więc konieczności jego wcześniejszej kompilacji za pomocą przycisku .

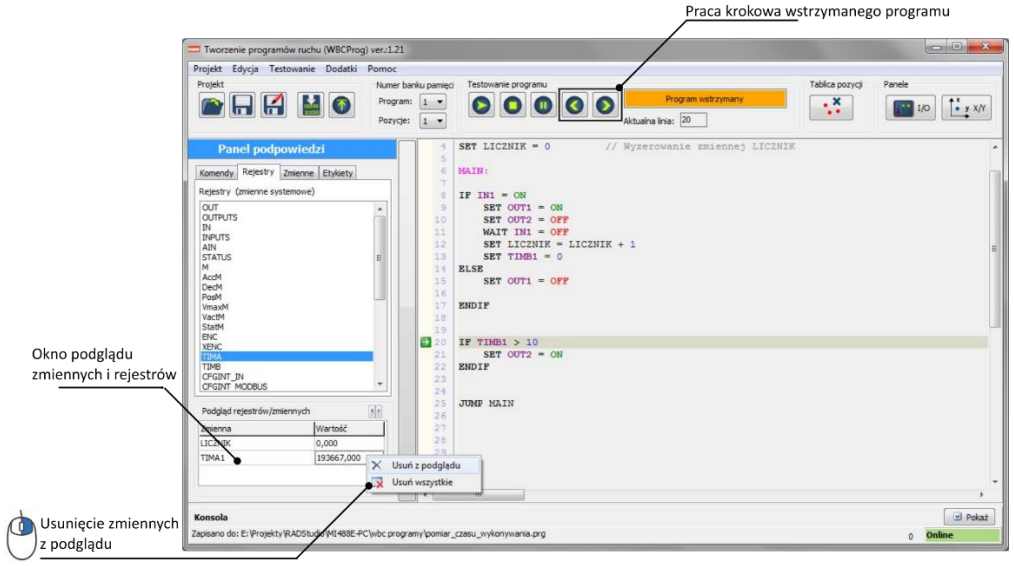
7.2.6 Uruchamianie i testowanie programu

Przesłany do sterownika program może zostać uruchomiony ręcznie za pomocą przycisku . Po uruchomieniu okno edytora zostaje zablokowane. Za pomocą przycisku można wstrzymać pracę programu, a następnie przyciskami sterować krokowo jego pracą. Aktualnie wykonywana linia oznaczana jest obrazkiem .

Podczas działania programu możliwe jest podejrzenie aktualnych wartości wybranych rejestrów i zmiennych (maks. 12). W celu dodania do podglądu na ***Panelu podpowiedzi*** należy kliknąć prawym przyciskiem myszy na interesujący nas rejestr (w zakładce ***Rejestry***) lub zmienną (w zakładce ***Zmienne***) i wcisnąć „***Dodaj do podglądu***”.



Rys. 30 Sposób dodawania rejestrów / zmiennych do podglądu.



Rys. 31 Usuwanie z podglądu.

7.3 Opis języka WBL

Język **WBL** (Wobit Basic Language) to prosty język tekstowy zbliżony do BASIC lub ST (Structured Text), ukierunkowany na tworzenie programów ruchu dla sterownika MIC488. Poniżej znajdują się najważniejsze informacje związane z programowaniem w języku **WBL** za pomocą programu **WBCprog**.

Kompilacja programu

Kompilacja programu powoduje wygenerowanie danych na podstawie wprowadzonych komend, które są zrozumiałe dla sterownika. Skompilowany program, jeśli nie zawiera błędów, może zostać przesłany do sterownika.

Działanie programu w sterowniku

Stworzony program wykonywany jest przez sterownik komenda po komendzie. Program powinien być „zapętlony" za pomocą funkcji skoków. Jeśli program po ostatniej komendzie nie wykona skoku do wcześniejszych komend zostanie zakończony. **Zakończony program powoduje wyłączenie wszystkich wyjść oraz zatrzymanie i wyłączenie napędów**.

Komentarze w programie

Wprowadzenie znaków „**//**" przed dowolną komendą lub opisem spowoduje, że będzie on pominięty podczas kompilacji programu. Przykład:

```
SET OUT1 = ON
//SET OUT2 = ON          // Ta linia programu zostanie pominięta
SET OUT3 = ON
```

Umieszczenie fragmentu programu między znakami „**/***” oraz „***/**” spowoduje, że zostanie on ominięty na etapie kompilowania programu. Przykład:

```
SET OUT1 = ON
/*
SET OUT2 = ON          // Te linie programu zostaną pominięte
SET OUT3 = ON
*/
SET OUT4 = ON
```

Wartości liczbowe

Wartości liczbowe mogę być zapisane jako stałe w postaci dziesiętnej, hex lub zmiennoprzecinkowej.
Dla wartości zmiennoprzecinkowych znakiem dziesiętnym jest znak kropki „ **.**”

```
SET VmaxM1 = 200       // Wartość dziesiętna w zapisie decymalnym
SET VmaxM1 = 0x2F      // Wartość dziesiętna w zapisie hex
SET VmaxM1 = 1.45      // Wartość zmiennoprzecinkowa
```

Komendy

Komenda to linia w programie, która zostaje odpowiednio zinterpretowana przez sterownik i wykonana. Komenda może zawierać dodatkowe parametry lub nie.

```
SET VmaxM1 = 2         // Komenda SET użyta do zapisania wartości do rejestru prędkości maks. napędu M1.
PABS M1 -5             // Komenda PABS zadająca pozycję absolutną
WAITPOS                // Komenda WAITPOS oczekiwania na dojazd do pozycji (bez parametrów)
```

Spis wszystkich komend znajduje się w rozdziale 0.

Definicje

Definicje pozwalają nadać własną nazwę dla stałej wartości liczbowej lub rejestru. By stworzyć definicje należy przed jej nazwą dodać znak „#”, a za nazwą wprowadzić wartość lub rejestr, któremu ma odpowiadać np.:

```
#POWTORZEN_X    10          // Stała wartość 10 jako nazwa "POWTORZEN_X"
#PREDKOSC_M1    AccM1       // Rejestr AccM1 jako nazwa "PRZYSPIESZENIE_M1"
#HMI_PREDKOSC   $R50        // Rejestr MODBUS 50 jako nazwa "HMI_PREDKOSC"
…
IF  HMI_PREDKOSC < PREDKOSC_MAX
SET PREDKOSC_M1 = HMI_PREDKOSC
ENDIF
```

Ponadto program posiada kilka zdefiniowanych na stałe wartości:

Definicja	Wartość	Opis
OFF	0	Stan wyłączony
ON	1	Stan włączony
M_OFF	0	Napęd wyłączony
M_ON	1	Napęd włączony
M_SPEED	2	Zadana prędkość
M_POS_SEARCH	3	Ruch do zadanej pozycji
M_POS_OK	4	Zadana pozycja osiągnięta
M_POS_ERROR	5	Błąd pozycji
M_POS_HOMING	6	Bazowanie
M_POS_CORRECTION	8	Korekcja pozycji
RIS	1	Zbocze rosnące przerwania
FAL	2	Zbocze malejące przerwania
RISFALL	3	Zbocze rosnące i malejące
MEM_READ	1234	„Klucz” do odczytu rejestrów użytkownika z pamięci nieulotnej
MEM_SAVE	2345	„Klucz” do zapisu rejestrów użytkownika do pamięci nieulotnej

MEM_RESET	4567	„Klucz” do resetu rejestrów użytkownika z pamięci nieulotnej

Etykiety i funkcje skoków

Skoki między komendami pozwalają na realizację bardziej skomplikowanych funkcji sterowania np.: wykonywania określonej ilości powtórzeń, ponowienia wybranego fragmentu programu czy jego realizacji w zależności od spełnienia warunku. Skoki w programie realizowane są do tzw. **etykiet** – czyli nazw dodanych w dowolnych fragmentach programu, zakończonych znakiem dwukropka „**:**” np.:

```
FUNKCJA_1:                  // Etykieta o nazwie “FUNKCJA_1”
SET OUT1 = ON
SET OUT2 = OFF
RETURN
..                          // zrób coś…
JUMP FUNKCJA_1              // Skok do etykiety „FUNKCJA_1”
```

Zmienne użytkownika

Użytkownik może wprowadzać własne zmienne zbudowane z dowolnych znaków będących literami i cyframi. Zmienne mogą przechowywać dowolne wartości liczbowe, mogą być wykorzystywane do wykonywania operacji matematycznych jednak nie są one adresowane przez użytkownika i nie ma do nich dostępu z zewnątrz. Stan zmiennej można podejrzeć podczas pracy programu w oknie podglądu zmiennych (po dodaniu wybranej zmiennej prawym przyciskiem myszy).

```
SET ABC = 1200              // Utworzenie nowej zmiennej o nazwie ABC i ustawienie jej wartości na 1200
..
ABC = 100                   //Zapisanie do zmiennej ABC wartości 100 (gdy zmienna już istnieje nie ma konieczności używania komendy SET)
..
SET ABC = ABC + 1           // Operacja matematyczna – zwiększenie wartości zmiennej ABC o 1 (wymagana komenda SET)
```

Rejestry systemowe (pamięć systemowa)

Rejestry systemowe to zmienne, które używane są przez sterownik do kontroli napędów, odczytu wejść, sterowania wyjść itp. Spis rejestrów systemowych i ich funkcje znajdują się w rozdziale 0**.**

Rejestry użytkownika z dostępem przez interfejs Modbus (pamięć użytkownik)

Użytkownik ma dostęp do 2000 rejestrów (komórek pamięci) do których może zapisywać lub odczytywać dowolne wartości. Dostęp do tych rejestrów jest możliwy także przez interfejsy RS232/RS485 w protokole MODBUS-RTU. Wartości zapisywane/odczytywane przez MODBUS mogą być typu INT, DINT, REAL. By poprawnie interpretować różne typy danych przed adresem rejestru należy użyć przedrostka:

$b – dla wartości typu bit
$B – dla wartości typu BYTE
$I – dla wartości typu INT
$D – dla wartości typu DINT
$R – dla wartości typu REAL

```
SET $R0 = 10.2          //  Zapisanie do rejestru 0 (typu REAL) wartości 10,2
...
IF  $I200 > 50          //  Sprawdzenie, czy wartość (INT) w rejestrze 200 > 50
...
PABS M1 $R20            //  Zadanie pozycji absolutnej (typu REAL) z rejestru 20
$b12 = 1                // Ustawienie bitu 12 na 1
```

Sposób adresowania zmiennych

BYTE	$B0	$B1	$B2	$B3	$B4	$B5	$B6	$B7
INT	$I0		$I1		$I2		$I3	
DINT/REAL	$D0 / $R0				$D2 / $R2			

Rejestry od adresu 1000 do 2000 mogą zostać zapisanie do pamięci nieulotnej przez komendy:

```
MEM_RETENT_CTRL = MEM_SAVE    //  Zapisanie rejestrów użytkownika (1000…2000) do pamięci nieulotnej
MEM_RETENT_CTRL = MEM_READ    //  Odczytanie rejestrów użytkownika (1000…2000) z pamięci nieulotnej
MEM_RETENT_CTRL = MEM_RESET   //  Reset rejestrów użytkownika (1000…2000) w pamięci nieulotnej (ustawienie 0)
```

Przerwania

Przerwania umożliwiają przerwanie aktualnie wykonywanej linii programu (np. na skutek zmiany sygnału na wybranym wejściu IN) i skok do zdefiniowanej etykiety. Pozwala to na szybką reakcję sterownika na sygnały zewnętrzne.

Przerwanie może być wykonane dla:

- zmiany stanu wejść IN1…INX
- zapisu przez Modbus któregokolwiek z rejestrów użytkownika (0…2000).
- zapisu przez Modbus wartości różnej od 0 do rejestru użytkownika o adresie 0,1,2 lub 3.
- cykliczne przerwanie realizowane z okresem 10ms, 100ms lub 1000ms
- zmiany stanu napędu M1…M4

Przerwanie powoduje skok do określonej etykiety, która musi być dodana do programu:

Źródło przerwania	Nazwa etykiety przerwania typu INT	Nazwa etykiety przerwania typu HARDINT	Rejestr konfiguracji przerwania
Zmiana stanu na wejściu IN1 … INX	INT_IN1 … INT_INX	HARDINT_IN1 …HARDINT_INX	CFGINT_INX= RIS / FALL / RIS_FALL
Zapis przez Modbus do rejestru 0…500	INT_MODBUS	HARDINT_MODBUS	CFGINT_MODBUS= HIGH
Zapisanie wartości !=0 w rejestrze użytkownika o adresie 0,1,2 lub 3 Po wykonaniu przerwania nastąpi wyzerowanie rejestru.	INT_USER_REG0… INT_USER_REG3	HARDINT_USER_REG0… HARDINT_USER_REG3	CFGINT_USER_REGX= HIGH
Przerwanie cykliczne 10ms	INT_TICK	HARDINT_TICK	CFGINT_TICK= HIGH
Przerwanie cykliczne 100ms	INT_TICK_100MS	HARDINT_TICK_100MS	CFGINT_TICK= HIGH
Przerwanie cykliczne 1000ms	INT_TICK_1000MS	HARDINT_TICK_1000MS	CFGINT_TICK= HIGH
Przerwanie od zmiany stanu napędu	INT_M1…INT_M4	HARDINT_M1… HARDINT_M4	CFGINT_TICK= RIS_FALL

Przerwania typu **INT_** wykonują skok do etykiety przerwania w trakcie trwania innej komendy. Etykieta takiego przerwania powinna być zakończona komendą **RETURN**. Jeśli etykieta zakończona jest komendą **JUMP** (następuje skoku do innej etykiety) to następuje także reset pamięci skoków i nie można użyć jako kolejnej komendy **RETURN** Podczas trwania przerwania (dopóki nie pojawi się komenda RETURN) inne przerwania są zablokowane. Po komendzie **RETURN** następuje powrót do miejsca wykonywania programu sprzed przerwania.

Przerwania typu **HARDINT_** wykonują skok do etykiety przerwania po zakończeniu aktualnej komendy. Etykieta takiego przerwania nie może być zakończona komendą **RETURN**

By aktywować przerwanie od danego sygnału należy zapisać do odpowiedniego rejestru konfiguracyjnego np.:

```
SET CFGINT_IN1 = RIS          // Przerwanie od zmiany stanu z 0 na 1 na wejściu IN1
SET CFGINT_MODBUS = HIGH      // Przerwanie od zapisu do rejestrów Modbus (0…2000)
SET CFGINT_USER_REG0 = HIGH   // Przerwanie od zapisu wartości <> 0 do rejestru użytkownika 0
SET CFGINT_TICK = HIGH        // Przerwanie czasowe wykonywane co 10 ms
SET CFGINT_M1 = RIS_FALL      // Przerwanie od zmiany stanu napędu M1
```

Do konfiguracji przerwań można użyć następujących wartości:

Rejestr konfiguracji przerwania	Dostępne wartości
CFGINT_IN1… CFGINT_INX	**0 (OFF)** – przerwania wyłączone **1 (RIS)** – przerwanie włączone na zbocze rosnące (zmiana sygnału z 0 na 1 na wejściu) **2 (FALL)** – przerwanie włączone na zbocze opadające (zmiana sygnału z 1 na 0 na wejściu) **3 (RISFALL)** – przerwanie włączone na zbocze rosnące i opadające (zmiana sygnału z 0 na 1 albo 1 na 0 na wejściu)
CFGINT_MODBUS CFGINT_USER_REG0…. CFGINT_USER_REG1	**0 (OFF)** – przerwania wyłączone **4 (HIGH)** – przerwanie włączone
CFGINT_M1… CFGINT_M4	**0 (OFF)** – przerwania wyłączone **4 (RIS_FALL)** – przerwanie włączone

8.Przykłady programów w WBCprog

Poniżej zestawiono przykłady wykorzystania poszczególnych funkcji sterownika MIC488. Dodatkowe przykłady można znaleźć w katalogu programu o nazwie ***WBC przykłady.***

8.1 Obsługa wejść / wyjść

Wykorzystywane rejestry:

- ***OUTX*** – pojedyncze wyjście
- ***OUTPUTS*** – wszystkie wyjścia
- ***INX*** – pojedyncze wejście
- ***INPUTS*** – wszystkie wyjścia
- ***MX_EN*** – sygnał ENABLE sterowania napędem

Sterowanie wyjściami

```
SET OUT1 = ON           // Włączenie wyjścia O1
SET OUT1 = OFF          // Wyłączenie wyjścia O1
SET OUTPUTS = 0         // Wyłączenie wszystkich wyjść
SET M1_EN = ON          // Załączenie wyjścia EN napędu M1
```

Sprawdzenie stanu wejścia

```
IF IN2 = ON             //  Jeśli wejście I2 aktywne
...                     //  …
ENDIF                   //  Koniec warunku
```

Oczekiwanie na wejście

```
WAIT IN2 = ON           //  Czekaj, aż wejście I2 będzie aktywne
```

8.2 Komenda Pulse

MIC488 udostępnia komendę PULSE, pozwalającą zmienić stan dowolnych wyjść cyfrowych po upływie określonego czasu. Czas zadać można z dokładnością do 10ms.

Komenda Pulse

```
SET OUT2 = ON           // Włączenie wyjścia O2
SET OUT3 = OFF          // Wyłączenie wyjścia O3
PULSE(OUT2, 2.45)       // Wyjście O2 zostanie wyłączone po upływie 2.45s (2450ms)
PULSE(OUT3, 4)          // Wyjście O3 zostanie włączone po upływie 4s (4000ms)
...                     // …
```

8.3 Odczyt wejść analogowych (0-10V)

Wykorzystywane rejestry: ***AINX*** – wartość z wejścia analogowego

```
VABS M1 AIN1            // Zadanie prędkości dla napędu M1 = wartości z wejścia AIN1
...                     // zrób coś...
IF AIN2 < 5,0           // Sprawdzenie czy napięcie na AIN2 mniejsze od 5.
...                     // zrób coś...
ENDIF
```

Wartość z rejestrów AIN1/AIN2 może być przeskalowana w ustawieniach (zakładka ***Konfiguracja -> We 0-10V***)

8.4 Komunikacja Modbus master

MIC488 może działać w trybie Modbus-Master na porcie COM2 (RS485) i sterować urządzeniami Modbus slave.

Odczyt/zapis pojedynczych rejestrów:

```
MODBUSW_OUT(1, 10, ON)          // Ustawienie wyjścia o adresie 10 w urządzenia slave 1
MODBUSW_I(1, 10, 250)           // Zapisanie do rejestru 10 urządzenia slave 1 wartości 250
MODBUSW_DI(2, 22, 12560)        // Zapisanie do rejestru 22 urządzenia slave 2 wartości 12560
MODBUSW_R(1, 24, 12.34)         // Zapisanie do rejestru 24 urządzenia slave 1 wartości 12.34
MODBUSW_R(1, 0x10, 12.34)       // Zapisanie do rejestru 0x10 urządzenia slave 1 wartości 12.34

                                // Odczytanie stanu wejścia o adresie 1 urządzenia slave 1
SET STAN = MODBUSR_IN(1, 22)    // Odczytanie wartości z rejestru 2 urządzenia slave 1
SET ABC = MODBUSR_I(2, 10)      // Odczytanie wartości z rejestru 4 urządzenia slave 2
SET ABC = MODBUSR_R(4, 22)
```

Odczyt wielu rejestrów. MIC488 pozwala odczytać wiele rejestrów przez Modbus master z urządzenia slave. Odczytane dane przekazywane są do pamięci użytkownika od wskazanego numeru pamięci.

```
MODBUSR_NREG(1, 0x200, 8, 500)  // Odczytanie 8 rejestrów (16-bitowych) od adresu 0x200
                                urządzenia slave 1 do pamięci użytkownika od numeru 500.
SET X1 = $I500                  // Przepisanie do X1 wartości odczytanej z rejestru 0x200
SET X2 = $I501                  // Przepisanie do X1 wartości odczytanej z rejestru 0x201
..
SET X7 = $I507                  // Przepisanie do X1 wartości odczytanej z rejestru 0x207
```

8.5 Sterowanie napędami

Wykorzystywane rejestry:

- ***AccMX, DecMX, VmaxMX*** – parametry rampy (przyspieszenie, hamowanie, prędkość maks.)
- ***PosMX*** – pozycja aktualna
- ***VactMX*** – prędkość aktualna
- ***MX*** – sterowanie danym napędem (załączenie wyjścia ENABLE, funkcja STOP i BRAKE)

Ustawienie parametrów ruchu (rampy)

```
SET AccM1 = 10          // Ustawienie przyspieszenia dla napędu M1
SET DecM1 = 10          // Ustawienie hamowania dla napędu M1
SET VMaxM1 = 0          // Ustawienie prędkości maks. dla M1

SET AccM2 = AccM1       // Parametry dla napędu M2 takie same jak M1
SET DecM2 = DecM1
SET VMaxM2 = VMaxM1
```

Załączenie napędu

```
SET M1 = ON             //  Włączenie napędu M1
SET M2 = ON             //  Włączenie napędu M2
```

Zatrzymanie/zahamowanie napędu w ruchu

```
STOP  M1              //  Zatrzymanie napędu M1 (nagłe)
BRAKE M2              //  Zahamowanie napędu M2 (wyhamowanie do prędkości 0)
```

Bazowanie napędu

```
...
HOME M1 -2            // Rozpoczęcie bazowania napędów M1 i M2 w kierunku ujemnym (do
HOME M2 -2            // krańcówki KL)
WAITPOS
```

Zadanie prędkości

```
VABS M1 5             // Zadanie prędkości 5  dla napędu M1
VABS M2 -2,5          // Zadanie prędkości -2,5  dla napędu M2 (ruch w stronę „ujemną")
...
VREL M1 -0,5          // Zmniejszenie prędkości napędu M1 o 0,5
VREL M2 0,5           // Zwiększenie prędkości napędu M2 o 0,5
```

Zadanie pozycji

```
PABS M1 10            // Zadanie pozycji 10  dla napędu M1
PABS M2 5,25          // Zadanie pozycji 5,25  dla napędu M2
WAITPOS               // Oczekiwanie na osiągnięcie zadanych pozycji

PREL M1 1             // Zwiększenie pozycji napędu M1 o 1
PREL M2 -1            // Zmniejszenie pozycji napędu M2 o 1
```

Zadanie pozycji z tablicy pozycji

```
MOVEPTP @NAZWA_POZYCJI     // Zadanie pozycji z tablicy pozycji o nazwie NAZWA_POZYCJI
WAITPOS                    // Oczekiwanie na osiągnięcie pozycji przez napędy
...
SET ABC = 2                // Zapisanie do zmiennej ABC wartości 2
MOVEPTP ABC                // Zadanie pozycji z tablicy pozycji o numerze ze zmiennej ABC
```

Oczekiwanie na osiągnięcie zadanej pozycji

Sposób 1 – komenda **WAITPOS** oczekująca na osiągnięcie pozycji dla wszystkich napędów

```
PABS M1 10        // Zadanie pozycji 10  dla napędu M1
PABS M3 20        // Zadanie pozycji 20  dla napędu M3
WAITPOS           // Oczekiwanie na osiągnięcie pozycji przez wszystkie napędy
```

Sposób 2 – sprawdzanie rejestru **MX**. Gdy MX = 0 to napęd nie znajduje się w ruchu.

```
PABS M1 10        // Zadanie pozycji 10  dla napędu M1
WAIT M1 = 0       // Oczekiwanie, aż napęd M1 nie będzie w ruchu
```

Sposób 3 – sprawdzenie rejestru statusowego **StatMX**.

```
PABS M1 10                 // Zadanie pozycji 10  dla napędu M1
WAIT StatM1 = M_POS_OK     // Oczekiwanie, aż napęd M1 osiągnie zadaną pozycję
```

Jeśli dany napęd jest wyłączony (sygnał EN wyłączony) to komenda **WAITPOS** pomija sprawdzanie jego pozycji.

Zerowanie pozycji

```
SET PosM1 = 0    // Wyzerowanie pozycji napędu M1
SET PosM2 = 10   // Nadpisanie aktualnej pozycji napędu M2 wartością 10
```

8.5.1 Interpolacja liniowa (w układzie X/Y)

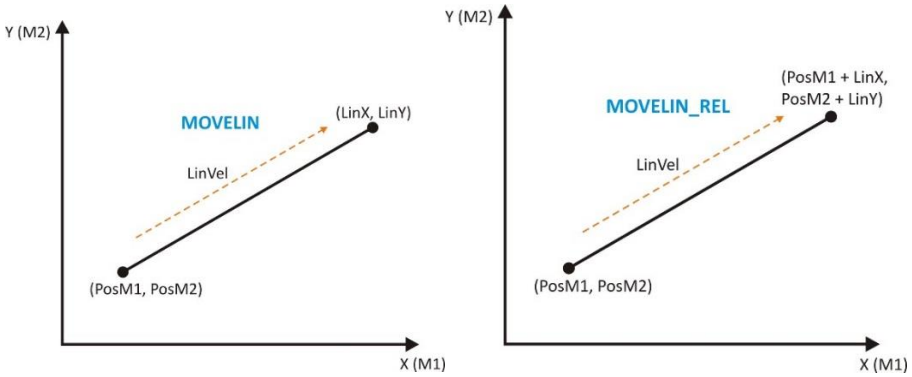
MIC488 pozwala na realizację interpolacji liniowej z wykorzystaniem napędów M1 i M2 tworzących osie X oraz Y. Interpolacja dostosowuje prędkość obu napędów tak, by ruch został zakończony w tym samym momencie. Uwaga

– należy ustawić takie same parametry rampy dla obu napędów. Ponadto rampy powinny być możliwie krótkie (duże wartości przyspieszenia/opóźnienia).
Realizacja ruchu po linii prostej wymaga następujących parametrów:
X/Y – pozycja docelowa
Vel – prędkość ruchu liniowego

Parametry te wprowadza się w tablicy pozycji pod wybraną pozycją (definiującą wówczas parametry ruchu po linii) lub za pomocą rejestrów ***LinX***, ***LinY***, ***LinVel*** które odwołują się bezpośrednio do indeksu 0 w tablicy pozycji i mogą być modyfikowane podczas działania programu.

Do wykonania ruchu po prostej służą następujące komendy:

- ***MOVELIN*** *@ABC / index*– powoduje ruch napędu w osiach M1 i M2 po linii prostej do punktu LinX, LinY
- ***MOVELIN*** **_REL** *@ABC / index* – powoduje przesunięcie napędu w osiach M1 i M2 o wartość LinX, LinY



Przykład wykonania ruchu po linii z tablicy pozycji

```
MOVELIN @POS0                  // Wykonanie ruchu po linii do punktów z tablicy
WAITPOS                        // Oczekiwanie na zakończenie ruchu
MOVELIN_REL @POS0              // Wykonanie ruchu relatywnego po linii
WAITPOS
```

Przykład wykonania ruchu po linii z parametrów wprowadzonych w programie
(komendy ***MOVELIN*** / ***MOVELIN*****_REL** muszą wskazywać na indeks 0 w tablicy pozycji)

```
SET LinX = 50        // Pozycja X docelowy
SET LinY = 75        // Pozycja Y docelowa
SET LinVel = 20      // Prędkość ruchu po linii
...
MOVELIN 0            // Wykonanie ruchu po linii do punktów X, Y
WAITPOS              // Oczekiwanie na zakończenie ruchu
MOVELIN_REL 0        // Wykonanie przesunięcia po linii o wartości X,Y
WAITPOS              // Oczekiwanie na zakończenie ruchu
```

8.5.2 Interpolacja liniowa (w dowolnym układzie)

MIC488 pozwala na dodatkową realizację interpolacji liniowej z wykorzystaniem wszystkich napędów udostępnianych przez sterownik. Od wersji oprogramowania 1.60 jest to zalecany sposób realizacji interpolacji liniowej. Interpolacja dostosowuje parametry ruchu poszczególnych napędów, by poszczególne etapy ruchu kończyły się w tym samym momencie i realizowały ruch w linii prostej (dla układów kartezjańskich). Podczas realizacji uwzględniane są parametry osi, dla której zadany dystans jest największy. Parametry ruchu pozostałych osi są kalkulowane na podstawie ruchu osi dominującej.
Realizacja ruchu po linii prostej wymaga podania pozycji końcowych dla wszystkich osi biorących udział w interpolacji.

Parametry te wprowadza się w tablicy pozycji pod wybraną pozycją (definiującą wówczas punkty końcowe ruchu po linii) lub za pomocą rejestrów ***PTPM*** które odwołują się bezpośrednio do indeksu 0 w tablicy pozycji i mogą być

modyfikowane podczas działania programu. Przy użyciu rejestrów ***PTPM*** niezbędne jest zapisanie także wartości rejestru ***LinAxis*** który zawiera bitowe flagi aktywujące poszczególne osie.

Do wykonania ruchu po prostej służą następujące komendy:

- ***MOVELIN2*** *@ABC / index*– powoduje ruch napędu we wskazanych osiach do punktu zapisanego pod pozycją ABC w tablicy pozycji
- ***MOVELIN 2_REL*** *@ABC / index* – powoduje ruch napędu we wskazanych osiach o odległość wyznaczoną przez punkt zapisany pod pozycją ABC w tablicy pozycji

Przykład wykonania ruchu po linii z tablicy pozycji

```
MOVELIN2 @POS2                  // Wykonanie ruchu po linii do punktów z tablicy
WAITPOS                         // Oczekiwanie na zakończenie ruchu
MOVELIN2_REL @POS3              // Wykonanie ruchu relatywnego po linii
WAITPOS
```

Przykład wykonania ruchu po linii z parametrów wprowadzonych w programie
(komendy ***MOVELIN2*** / ***MOVELIN*_REL2** muszą wskazywać na indeks 0 w tablicy pozycji)

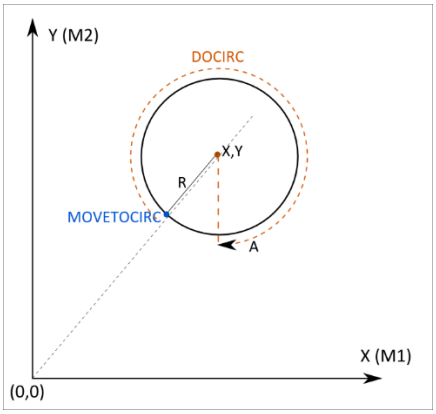
```
SET PTPM1 = 50          // Pozycja X docelowy
SET PTPM2 = 75          // Pozycja Y docelowa
SET PTPM3 = 20          // Prędkość ruchu po linii
SET LinAxis = 0x03      // Bitowa wartość b0011, wskazanie dwóch pierwszych osi
...
MOVELIN 0               // Wykonanie ruchu po linii do punktów PTPM1 i PTPM2
WAITPOS                 // Oczekiwanie na zakończenie ruchu
MOVELIN_REL @POS0       // Wykonanie przesunięcia po linii o wartości PTPM1 i PTPM2
WAITPOS                 // Oczekiwanie na zakończenie ruchu
```

8.5.3 Interpolacja kołowa

MIC488 pozwala na realizację interpolacji kołowej z wykorzystaniem napędów M1 i M2 tworzących osie X oraz Y. By funkcja interpolacji działała poprawnie parametr *GEAR* w konfiguracji napędów M1 i M2 powinien być taki sam (by poprawnie wykonać koło napędy muszą mieć mechanicznie takie same przełożenia). Podczas wykonywania interpolacji korekcja pozycji z enkoderów jest wyłączona.

Ruch po okręgu (interpolacja z podaniem współrzędnych środka okręgu)

MIC488 pozwala wykonać ruch po okręgu z podaniem parametrów określających jego środek, promień i kąt (360 stopni = pełny okrąg). Prędkość / przyspieszenie podczas ruchu po okręgu zależy od parametrów napędu M1.



X/Y – pozycja środka okręgu względem pozycji zerowej,
R – promień okręgu,
A – kąt wykonywanego okręgu (w stopniach). Jeśli A < 0 ruch wykonywany jest przeciwnie do ruchu wskazówek zegara.

Parametry te wprowadza się w tablicy pozycji pod wybraną pozycją (definiującą wówczas parametry okręgu) lub za pomocą rejestrów ***CircX***, ***CircY***, ***CircR*** oraz ***CircA***, które odwołują się bezpośrednio do indeksu 0 w tablicy pozycji i mogą być modyfikowane podczas działania programu.

Rys. 3 Parametry dla okręgu w tablicy pozycji.

Do wykonania okręgu służą następujące komendy:

- ***MOVETOCIRC*** *@ABC*– powoduje ruch napędu w osiach M1 i M2 do punktu na okręgu znajdującego się najbliżej pozycji zerowej.
- ***DOCIRC*** *@ ABC* – powoduje wykonanie okręgu.

Gdzie *@ ABC* to nazwa pozycji w tablicy pozycji opisująca parametry wykonywanego okręgu.

Przykład wykonania okręgu z parametrów z tablicy pozycji

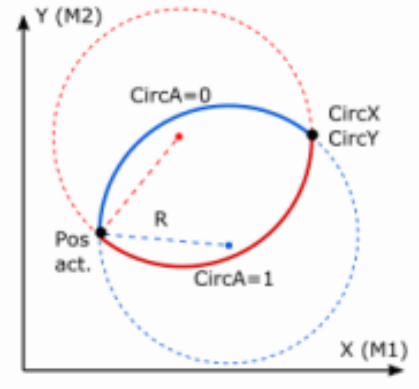
```
MOVETOCIRC @POS0                // Wykonanie ruchu do punktu startowego na okręgu
WAITPOS                         // Oczekiwanie na dojazd
DOCIRC @POS0                    // Wykonanie okręgu
WAITPOS                         // Oczekiwanie na wykonanie okręgu
```

Przykład wykonania okręgu z parametrów wprowadzonych w programie
(komendy ***MOVETOCIRC*** oraz ***DOCIRC*** muszą wskazywać na indeks 0 w tablicy pozycji)

```
SET CircX = 50          // Punkt X środka okręgu
SET CircY = 50          // Punkt Y środka okręgu
SET CircR = 45          // Promień okręgu
SET CircA = 360         // Kąt wykonywanego okręgu
...
MOVETOCIRC 0            // Wykonanie ruchu do punktu startowego na okręgu
WAITPOS                 // Oczekiwanie na dojazd
DOCIRC 0                // Wykonanie okręgu
WAITPOS                 // Oczekiwanie na wykonanie okręgu
```

Ruch po łuku (interpolacja z podaniem współrzędnych końca łuku)

MIC488 pozwala wykonać ruch po łuku od aktualnej pozycji do pozycji określonej parametrami X/Y z promieniem R oraz orientacją łuku A. Prędkość / przyspieszenie podczas ruchu zależą od parametrów napędu M1.



X/Y – pozycja końca łuku
R – promień łuku,

A – kierunek łuku

Parametry te wprowadza się w tablicy pozycji pod wybraną pozycją (definiującą wówczas parametry okręgu) lub za pomocą rejestrów ***CircX***, ***CircY***, ***CircR*** oraz ***CircA***, które odwołują się bezpośrednio do indeksu 0 w tablicy pozycji i mogą być modyfikowane podczas działania programu.

Parametry dla ruchu po łuku w tablicy pozycji.

```
DOARC @POS1          // Wykonanie ruchu po łuku z parametrów z tablicy pozycji o nazwie @POS1
WAITPOS              // Oczekiwanie na dojazd


SET CircX = 120      // Punkt X końca łuku
SET CircY = 80       // Punkt Y końca łuku
SET CircR = 45       // Promień łuku
SET CircA = 0        // Orientacja łuku
...
DOARC 0              // Wykonanie ruchu po łuku z parametrów w rejestrach CircX/Y/R/A
WAITPOS              // Oczekiwanie na dojazd
```

8.6 Odczyt / zapis pozycji enkodera

Wykorzystywane rejestry:

- ***ENCX*** – wartość licznika enkodera w impulsach
- ***XENCX*** – wartość licznika enkodera przeliczona na obroty napędu

```
SET ENC1 = 0      // Wyzerowanie wartości licznika enkodera ENC1
SET ENC2 = 100    // Wpisanie wartości 100 do licznika enkodera ENC2
PABS M3 XENC3     // Zadanie pozycji dla napędu M1 równego pozycji z enkodera ENC3
```

8.7 Liczniki czasu

Wykorzystywane rejestry:

- **TIMAX** – licznik czasu o rozdzielczości 10 ms (wartość licznika zwiększana jest o 1 co 10 ms)
- **TIMBX** – licznik czasu o rozdzielczości 1 sek.

‚gdzie X oznacza numer licznika czasu (1…16)

```
SET TIMB1 = 0     // Wyzerowanie licznika czasu TIMB1
...
IF TIMB1 > 60     // Sprawdzenie czy licznik TIMB1 większy od 60 sekund
...               // Jeśli większy...zrób coś
ENDIF
```

8.8 Operacje/funkcje matematyczne i zmienne

```
SET ABC = 0              // Utworzenie nowej zmiennej ABC i zapisanie do niej wartości 0
SET ABC = ABC + 1        // Zwiększenie zmiennej ABC o 1
SET XYZ = ABC * 5        // Utworzenie nowej zmiennej XYZ równej ABC * 5
SET JKL = ENC1 - ENC2    // Utworzenie nowej zmiennej JKL równej wartości z ENC1 - ENC2
SET OUT1 = !OUT1         // Negacja zmiennej (zmiana na stan przeciwny)
```

Spis dostępnych operacji/funkcji matematycznych znajduje się rozdziale „

Komunikacja MODBUS

Modbus RTU Slave

Sterownik pozwala na komunikację z urządzeniem nadrzędnym (MASTER) w protokole MODBUS-RTU. Komunikacja może odbywać się poprzez port RS232 (COM1) lub RS485 (COM2). MIC488 będzie pełnił funkcję urządzenia slave.

Parametry transmisji

- Domyślny adres: 1 (konfigurowane w zakresie 1…127)
- Domyślna prędkość transmisji: **38400 b/s** (dostępne prędkości 9600, 19200, 38400, 57600, 115200)
- Bity stopu: **1,** Parzystość: **brak**
- Timeout: **750µs** (maksymalny czas odstępu między kolejnymi bajtami w ramce)

Opis komunikacji, spis rejestrów użytkownika i sposób sterowania napędami przez MODBUS-RTU dostępny jest w dokumentacji „***MIC488_protokol_MODBUS.pdf***"

Dostęp do pamięcią rejestrów modbus użytkownika

Urządzenie posiada przestrzeń pamięci, do której mamy dostęp z zewnętrznych urządzeń przez port COM1/2 i protokół MODBUS RTU.

Dostęp przez bezpośrednie adresowanie ($IX, $DX, $RX)

```
SET $I20 = 250            // Zapisanie do rejestru o adresie 20 wartości 250 (typu INT)
SET $D22 = 85000          // Zapisanie do rejestru o adresie 22 wartości 85000 (typu DINT)
SET $R50 = 25.78          // Zapisanie do rejestru o adresie 50 wartości 25.78 (typu REAL)
```

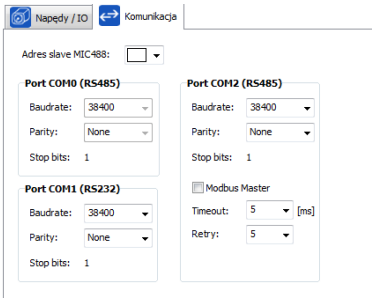
Dostęp przez rejestry indeksowane (MEM_INT, MEM_DINT, MEM_REAL)

```
SET MEM_INDEX = 0         // Brak przesunięcia, np. MEM_INT0 odpowiada $I0
SET MEM_INT0 = 250        // Zapisanie do rejestru $I0 wartości 250 (typu INT)
SET MEM_DINT2 = 85000     // Zapisanie do rejestru $D2 wartości 85000 (typu DINT)
SET MEM_REAL4 = 25.78     // Zapisanie do rejestru $R4 wartości 25.78 (typu REAL)

SET MEM_INDEX = 12        // Przesunięcie o 12 rejestrów, np.: MEM_INT0 odpowiada $I12
SET MEM_INT0  = 250       // Zapisanie do rejestru $I12 wartości 250 (typu INT)
SET MEM_DINT2 = 85000     // Zapisanie do rejestru $D14 wartości 85000 (typu DINT)
SET MEM_REAL4 = 25.78     // Zapisanie do rejestru $R16 wartości 25.78 (typu REAL)
```

8.9 Modbus RTU Master

MIC488 może działać w trybie Modbus Master na porcie COM2 (konfiguracja odbywa się w zakładce Konfiguracja->Komunikacja programu MIC488-PC).



Przykład zapisu i odczytu do urządzenia SLAVE z wykorzystaniem komunikacji Modbus Master:

```
MODBUSW_OUT(1, 10, ON)          // Ustawienie wyjścia o adresie 10 w urządzenia slave 1
MODBUSW_I(1, 10, 250)           // Zapisanie do rejestru 10 urządzenia slave 1 wartości 250
MODBUSW_DI(2, 22, 12560)        // Zapisanie do rejestru 22 urządzenia slave 2 wartości 12560
MODBUSW_R(1, 24, 12.34)         // Zapisanie do rejestru 24 urządzenia slave 1 wartości 12.34


SET STAN = MODBUSR_IN(1, 22)    // Odczytanie stanu wejścia o adresie 1 urządzenia slave 1
SET ABC = MODBUSR_I(2, 10)      // Odczytanie wartości z rejestru 2 urządzenia slave 1
SET ABC = MODBUSR_R(4, 22)      // Odczytanie wartości z rejestru 4 urządzenia slave 2
```

Odczyt wielu rejestrów. MIC488 pozwala odczytać wiele rejestrów przez Modbus master z urządzenia slave. Odczytane dane przekazywane są do pamięci użytkownika od wskazanego numeru pamięci.

```
MODBUSR_NREG(1, 0x200, 8, 500)  // Odczytanie 8 rejestrów (16-bitowych) od adresu 0x200
                                urządzenia slave 1 do pamięci użytkownika od numeru 500.
SET X1 = $I500                  // Przepisanie do X1 wartości odczytanej z rejestru 0x200
SET X2 = $I501                  // Przepisanie do X1 wartości odczytanej z rejestru 0x201
..
SET X7 = $I507                  // Przepisanie do X1 wartości odczytanej z rejestru 0x207
```

” (strona 41).

8.10 Przerwania

```
SET CFGINT_IN1 = RIS      // Włączenie przerwania od wejścia IN1 na zbocze rosnące
SET CFGINT_IN2 = RIS      // Włączenie przerwania od wejścia IN2 na zbocze rosnące

MAIN:                     // Pętla główna
...                       // Tu wykonujemy coś
JUMP MAIN

INT_IN1:                  // Etykieta przerwania typu INT od wejścia IN1 - tu nastąpi skok,
...                       gdy na  wejściu IN1 pojawi się stan wysoki
REURN                     // Powrót z przerwania

HARDINT_IN2:              // Etykieta przerwania typu HARDINT od wejścia IN2 - tu nastąpi
...                       // skok, gdy na wejściu IN2 pojawi się stan wysoki
                          // Możliwy skok w inne miejsce programu
JUMP MAIN
```

Wejścia IN1…8 odświeżane są sprzętowo co 10ms. Wejścia IN9…20 odczytują sygnał bez opóźnień.

8.11 Przykładowy program

```
#POWTORZEN 10             // Zdefiniowanie pomocniczej nazwy dla wartości 10
#WE_START  IN3            // Zdefiniowanie pomocniczej nazwy dla wejścia IN3
#WE_STOP   IN4            // Zdefiniowanie pomocniczej nazwy dla wejścia IN4
#WE_KROK   IN5            // Zdefiniowanie pomocniczej nazwy dla wejścia IN5
#WY_STOP   OUT1           // Zdefiniowanie pomocniczej nazwy dla wyjścia OUT1
#WY_GOTOWY OUT2           // Zdefiniowanie pomocniczej nazwy dla wyjścia OUT2

SET AccM1 = 20            // Ustawienie przyspieszenia ruchu
SET DecM1 = 20            // Ustawienie hamowania ruchu
SET VmaxM1 = 6            // Ustawienie prędkości maksymalnej

WAIT WE_START  = ON       // Oczekiwanie na aktywne wejście IN3
SET M1 = ON               // Włączenie napędu M1
HOME M1 -1                // Rozpoczęcie bazowania
WAITPOS                   // Oczekiwanie na zakończenie bazowania

PETLA_GLOWNA:             // Etykieta pętli głównej
  JUMP SPRAWDZ_STOP       // Skok do etykiety SPRAWDZ_STOP
  JUMP WYKONAJ            // Skok do etykiety WYKONAJ
JUMP PETLA_GLOWNA         // Skok do PETLA_GLOWNA

WYKONAJ:                  // Etykieta WYKONAJ
IF StatM1 = M_POS_OK      // Jeśli napęd osiągnął zadaną pozycję
  SET WY_GOTOWY = ON      // Ustaw wyjście OUT2
  DELAY 5                 // Czekaj 5 sek.
  SET WY_GOTOWY = OFF     // Załącz wyjście OUT2
  PREL M1 2               // Wykonaj ruch napędu o 2
ENDIF

RETURN                    // Powrót do miejsca sprzed skoku

SPRAWDZ_STOP:             // Etykieta SPRAWDZ_STOP
IF WE_STOP = ON           // Jeśli wejście IN4
  STOP M1                 // Zatrzymaj napęd
  SET WY_STOP = ON        // Załącz wyjście OUT1
  DELAY 2                 // Czekaj 2 sek.
  SET WY_STOP = OFF       // Wyłącz wyjście OUT1
ENDIF
RETURN                    // Powrót do miejsca sprzed skoku
```

9.Komunikacja MODBUS

9.1 Modbus RTU Slave

Sterownik pozwala na komunikację z urządzeniem nadrzędnym (MASTER) w protokole MODBUS-RTU. Komunikacja może odbywać się poprzez port RS232 (COM1) lub RS485 (COM2). MIC488 będzie pełnił funkcję urządzenia slave.

Parametry transmisji

- Domyślny adres: 1 (konfigurowane w zakresie 1…127)
- Domyślna prędkość transmisji: **38400 b/s** (dostępne prędkości 9600, 19200, 38400, 57600, 115200)
- Bity stopu: **1**, Parzystość: **brak**
- Timeout: **750µs** (maksymalny czas odstępu między kolejnymi bajtami w ramce)

Opis komunikacji, spis rejestrów użytkownika i sposób sterowania napędami przez MODBUS-RTU dostępny jest w dokumentacji „***MIC488_protokol_MODBUS.pdf***"

Dostęp do pamięcią rejestrów modbus użytkownika

Urządzenie posiada przestrzeń pamięci, do której mamy dostęp z zewnętrznych urządzeń przez port COM1/2 i protokół MODBUS RTU.

Dostęp przez bezpośrednie adresowanie ($IX, $DX, $RX)

```
SET $I20 = 250            // Zapisanie do rejestru o adresie 20 wartości 250 (typu INT)
SET $D22 = 85000          // Zapisanie do rejestru o adresie 22 wartości 85000 (typu DINT)
SET $R50 = 25.78          // Zapisanie do rejestru o adresie 50 wartości 25.78 (typu REAL)
```

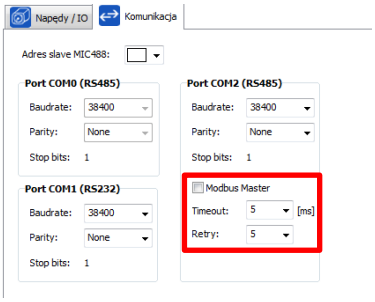
Dostęp przez rejestry indeksowane (MEM_INT, MEM_DINT, MEM_REAL)

```
SET MEM_INDEX = 0         // Brak przesunięcia, np. MEM_INT0 odpowiada $I0
SET MEM_INT0 = 250        // Zapisanie do rejestru $I0 wartości 250 (typu INT)
SET MEM_DINT2 = 85000     // Zapisanie do rejestru $D2 wartości 85000 (typu DINT)
SET MEM_REAL4 = 25.78     // Zapisanie do rejestru $R4 wartości 25.78 (typu REAL)

SET MEM_INDEX = 12        // Przesunięcie o 12 rejestrów, np.: MEM_INT0 odpowiada $I12
SET MEM_INT0  = 250       // Zapisanie do rejestru $I12 wartości 250 (typu INT)
SET MEM_DINT2 = 85000     // Zapisanie do rejestru $D14 wartości 85000 (typu DINT)
SET MEM_REAL4 = 25.78     // Zapisanie do rejestru $R16 wartości 25.78 (typu REAL)
```

9.2 Modbus RTU Master

MIC488 może działać w trybie Modbus Master na porcie COM2 (konfiguracja odbywa się w zakładce Konfiguracja->Komunikacja programu MIC488-PC).



Przykład zapisu i odczytu do urządzenia SLAVE z wykorzystaniem komunikacji Modbus Master:

```
MODBUSW_OUT(1, 10, ON)          // Ustawienie wyjścia o adresie 10 w urządzenia slave 1
MODBUSW_I(1, 10, 250)           // Zapisanie do rejestru 10 urządzenia slave 1 wartości 250
MODBUSW_DI(2, 22, 12560)        // Zapisanie do rejestru 22 urządzenia slave 2 wartości 12560
MODBUSW_R(1, 24, 12.34)         // Zapisanie do rejestru 24 urządzenia slave 1 wartości 12.34


SET STAN = MODBUSR_IN(1, 22)    // Odczytanie stanu wejścia o adresie 1 urządzenia slave 1
SET ABC = MODBUSR_I(2, 10)      // Odczytanie wartości z rejestru 2 urządzenia slave 1
SET ABC = MODBUSR_R(4, 22)      // Odczytanie wartości z rejestru 4 urządzenia slave 2
```

Odczyt wielu rejestrów. MIC488 pozwala odczytać wiele rejestrów przez Modbus master z urządzenia slave. Odczytane dane przekazywane są do pamięci użytkownika od wskazanego numeru pamięci.

```
MODBUSR_NREG(1, 0x200, 8, 500)  // Odczytanie 8 rejestrów (16-bitowych) od adresu 0x200
                                urządzenia slave 1 do pamięci użytkownika od numeru 500.
SET X1 = $I500                  // Przepisanie do X1 wartości odczytanej z rejestru 0x200
SET X2 = $I501                  // Przepisanie do X1 wartości odczytanej z rejestru 0x201
..
SET X7 = $I507                  // Przepisanie do X1 wartości odczytanej z rejestru 0x207
```

10. Spis komend i rejestrów

Komendy podstawowe

Komenda	Opis	Składnia
SET	Ustawia wyjście, zmienną, rejestr, wykonuje operacje matematyczne.	**SET** X = Y **SET** X = Y ***operacja*** Z **SET** X = funkcja(Y) ***operacja*** – dostępne operacje opisane niżej.

IF / ELSE /ENDIF	Porównuje stan wejść, wyjść, zmiennych, rejestrów. Gdy warunek spełniony wykonanie kolejnych linii programu, aż do komendy ENDIF (do ELSE jeśli istnieje). Jeśli nie przeskoczenie do ENDIF (ELSE jeśli istnieje)	**IF** X ***warunek*** Y
		.. gdy warunek spełniony
		ENDIF
		lub
		IF X ***warunek*** Y
		.. gdy warunek spełniony
		ELSE
		… gdy warunek niespełniony
		ENDIF
		warunek – dostępne warunki opisane niżej

WHILE / ENDWHILE	Porównuje stan wejść, wyjść, zmiennych, rejestrów. Dopóki warunek spełniony występuje zapętlenie między WHILE, a ENDWHILE. Jeśli warunek niespełniony następuj wyjście z ENDWHILE.	**WHILE** X ***warunek*** Y .. wykonywanie dopóki warunek spełniony **ENDWHILE** ***warunek*** – dostępne warunki opisane niżej
WAIT	Oczekuje na stan wejść, wyjść, zmiennych, rejestrów. Jeśli warunek spełniony przejście do następnej linii. Jeśli nie oczekuje, aż zostanie spełniony.	**WAIT** X ***warunek*** Y ***warunek*** – dostępne warunki opisane niżej
DELAY	Wprowadza opóźnienie czasowe w sekundach.	**DELAY** X
JUMP	Skok do istniejącej etykiety programu.	**JUMP** NAZWA_ETYKIETY
RETURN	Powrót do linii za ostatnią komendą skoku.	**RETURN**
END	Zakończenie działania programu	**END**
MODBUSW_OUT **MODBUSW_I** **MODBUSW_DI** **MODBUSW_W** **MODBUSW_DW** **MODBUSW_REAL** **MODBUSR_NREG**	Zapis rejestrów przez Modbus Master. MODBUSW_OUT – zapis wyjścia (funkcja 0x05) MODBUSW_I – zapis rejestru typu INT (funkcja 0x06) MODBUSW_DI – zapis rejestru typu DINT (funkcja 0x10) MODBUSW_W – zapis rejestru typu WORD (funkcja 0x06) MODBUSW_DW – zapis rejestru typu DWORD (funkcja 0x10) MODBUSW_REAL – zapis rejestru typu REAL (funkcja 0x10) MODBUSR_NREG – odczyt wielu rejestrów do pamięci użytkownika	**MODBUSW_OUT**(SLAVE, REG, VALUE) **MODBUSW_I**(SLAVE, REG, VALUE) **MODBUSW_DI**(SLAVE, REG, VALUE) **MODBUSW_W**(SLAVE, REG, VALUE) **MODBUSW_DW**(SLAVE, REG, VALUE) **MODBUSW_REAL**(SLAVE, REG, VALUE) SLAVE – adres urządzenia slave (1..247) REG – adres rejestru slave (0…65535) VALUE – wartość do zapisu **MODBUSR_NREG(**SLAVE, REG, NUM, DEST_ADR**)** SLAVE – adres urządzenia slave (1..247) REG – adres rejestru slave (0…65535) VALUE – wartość do zapisu NUM – ilość rejestrów DEST_ADR – początek rejestru pamięci użytkownika do którego trafią odczytane dane
MEM_COPY **MEM_SET**	Kopiowanie / ustawianie pamięci użytkownika	**MEM_COPY**(DEST_ADR, SRC_ADR, NUM) **MEM_SET**(DEST_ADR, VALUE, NUM) DEST_ADR – docelowy adres pamięci użytkownika SRC_ADR – źródłowy adres pamięci użytkownika VALUE – wartość do zapisu (16-bitowa) NUM – ilość rejestrów
PULSE	Zmiana stanu wyjścia po upływie określonego czasu	**PULSE(**REJESTR_WYJSCIA**,** CZAS**)**

Parametry X, Y, Z mogą być zmiennymi użytkownika, rejestrami urządzenia, rejestrami MODBUS lub wartościami stałymi.

Spis dostępnych operacji i warunków matematycznych

Operacje matematyczne	Funkcje matematyczne	Warunki
➢ **+** (dodawanie), ➢ **-** (odejmowanie), ➢ ***** (mnożenie), ➢ **/** (dzielenie), ➢ **!** (negacja) ➢ **%** (reszta z dzielenia), ➢ **|** (suma bitowa),	➢ **sin** (sinus), ➢ **cos** (cosinus), ➢ **tg** (tangens), ➢ **ctg** (cotangens), ➢ **asin** (arcus sinus), ➢ **acos** (arcus cosinus), ➢ **sqrt** (pierwiastek kwadratowy),	➢ **=** (równy), ➢ **<** (mniejszy), ➢ **>** (większy), ➢ **<=** (mniejszy równy), ➢ **>=** (większy równy)

➢ **&** (iloczyn bitowy), ➢ **||** (suma logiczna), ➢ **&&** (iloczyn logiczny), ➢ **>>** (przesunięcie bitowe w prawo), ➢ **<<** (przesunięcie bitowe w lewo),	➢ **min** (wartość min), ➢ **max** (wartość maks), Funkcje trygonometryczne przyjmują wartości w stopniach.	

Komendy ruchu

Komenda	Opis	Składnia
HOME	Wykonuje bazowanie napędu.	**HOME** MX Y Y – prędkość bazowania
VABS	Zadanie prędkości absolutnej (równej wprowadzonej wartości).	**VABS** MX Y Y – prędkość absolutna
VREL	Zwiększenie (zmniejszenie) aktualnej prędkości.	**VREL** MX Y Y – prędkość relatywna
PABS	Zadanie pozycji absolutnej (równej wprowadzonej wartości).	**PABS** MX Y Y – pozycja absolutna
PREL	Zwiększenie (zmniejszenie) aktualnej pozycji napędu.	**PREL** MX Y Y – pozycja relatywna
BRAKE	Zatrzymanie napędu (ustawienie prędkości 0).	**BRAKE** MX
STOP	Zatrzymanie napędu (szybkie).	**STOP** MX
MOVEPTP	Zadanie pozycji dla napędów z tablicy pozycji.	**MOVEPTP** @NAZWA_POZYCJI **MOVEPTP** NUMER_POZYCJI
MOVELIN	Ruch po linii prostej (dla napędów M1(X), M2(Y))	**MOVELIN** @NAZWA_POZYCJI **MOVELIN** NUMER_POZYCJI
MOVELIN_REL	Ruch po linii prostej relatywny (dla napędów M1(X), M2(Y))	
MOVELIN2	Ruch po linii prostej (dla wybranych napędów)	**MOVELIN2** @NAZWA_POZYCJI **MOVELIN2** NUMER_POZYCJI
MOVELIN2_REL	Ruch po linii prostej relatywny (dla wybranych napędów)	
MOVETOCIRC	Ruch do punktu startowego na wykonywanym okręgu	**MOVETOCIRC** @NAZWA_POZYCJI **MOVETOCIRC** NUMER_POZYCJI
DOCIRC	Ruch po okręgu (dla napędów M1(X), M2(Y))	**DOCIRC** @NAZWA_POZYCJI **DOCIRC** NUMER_POZYCJI
WAITPOS	Oczekiwanie na osiągnięcie pozycji we wszystkich napędach. UWAGA: Jeśli dany napęd nie jest aktywny (SET MX = OFF) to jest on pomijany przy sprawdzaniu pozycji.	**WAITPOS**

MX: numer napędu (M1, M2, M3, M4)

Rejestry ogólne

Rejestr	Opis	Dostęp	Zakres wartości
OUT**X**	Dostęp do wybranego wyjścia uniwersalnego sterownika.	R/W	0 – wyjście wyłączone, > 0 – wyjście włączone
OUTPUTS	Binarny dostęp do wszystkich wyjść uniwersalnych	R/W	0 ... 65535
IN**X**	Dostęp do wybranego wejścia uniwersalnego sterownika.	R	0 – wyjście nieaktywne > 0 – wyjście aktywne
INPUTS	Binarny dostęp do wszystkich wejść uniwersalnych.	R	0 ... 65535
AIN**X**	Dostęp do wejść analogowych sterownika	R	REAL
STATUS	Status pracy sterownika	R	Niedostępne
TIMA**X**	Uniwersalny timer o podstawie czasu 10 ms	R/W	0 – 999999
TIMB**X**	Uniwersalny timer o podstawie czasu 1 sek	R/W	0 – 999999
CFGINT_IN**X**	Konfiguracja przerwań dla wejść IN1…IN20 **X** = 1...20	R/W	0 (OFF) – przerwanie wyłączone 1 (RIS) – przerwanie na zbocze rosnące 2 (FALL) – przerwanie na zbocze opadające 1 (RISFALL) – przerwanie na oba zbocza
CFGINT_MODBUS	Konfiguracja przerwania dla zapisu danych do rejestrów użytkownika Modbus.	R/W	0 (INT_OFF) - wyłączone 4 (INT_HIGH) - włączone
CFGINT_USER_REG**X**	Konfiguracja przerwania od zmiany wartości rejestru użytkownika o adresie 0,1,2 lub 3	R/W	0 (INT_OFF) - wyłączone 4 (INT_HIGH) - włączone
CFGINT_TICK	Konfiguracja przerwania cyklicznego, wykonywanego co 10ms	R/W	0 (INT_OFF) - wyłączone 4 (INT_HIGH) - włączone
CFGINT_M**X**	Konfiguracja przerwania od zmiany stanu napędu	R/W	0 (INT_OFF) - wyłączone 3 (RIS_FALL) – włączone

W – zapis, **R** - odczyt

Rejestry napędów

Rejestr	Opis	Dostęp	Zakres wartości
M**X**	Sterowanie wyjściem ENABLE. Włączenie napędu, sprawdzenie czy napęd jest w ruchu.	R/W	Zapis: 0 (OFF) – wyłączenie napędu 1 (ON) – włączenie napędu Odczyt: 0 – napęd w spoczynku 1 – napęd w ruchu
StatM**X**	Status pracy napędu.	R	0 (M_OFF) – napęd wyłączony 1 (M_ON) – napęd włączony 2 (M_SPEED) – tryb prędkości 3 (M_POS_SEARCH) – tryb pozycji 4 (M_POS_OK) – pozycja osiągnięta 5 (M_POS_ERROR) – błąd pozycji 6 (M_POS_HOMING) – bazowanie 8 (M_POS_CORRECTION) – korekcja pozycji 9 (M_POS_LIM_L) – osiągnięta skrajna pozycja L 10 (M_POS_LIM_R) – osiągnięta skrajna pozycja R
VmaxM**X**	Prędkość maksymalna dla trybu pozycji.	R/W	REAL
VactM**X**	Prędkość aktualna napędu.	R	REAL
AccM**X**	Przyspieszenie dla rozpędzania w trybie zadanej pozycji. Przyspieszenie dla rozpędzania i hamowania w trybie zadanej prędkości.	R/W	REAL
DecM**X**	Przyspieszenie dla hamowania w trybie zadanej pozycji.	R/W	REAL
PosM**X**	Pozycja aktualna napędu.	R/W	REAL
LimLM**X**	Programowe ograniczenie pozycji w stronę ujemną (L)	R/W	REAL
LimRM**X**	Programowe ograniczenie pozycji w stronę dodatnią (R)	R/W	REAL
ENC**X**	Impulsy z enkodera.	R/W	DINT
XENC**X**	Impulsy z enkodera przeliczone na pozycję napędu.	R	REAL
PTP**M**	Pozycje dla napędów odpowiadające pozycjom w tablicy pozycji o indeksie 0.	R/W	REAL
CircX, CircY, CircR, CircA	Parametry okręgu dla interpolacji kołowej (odpowiadają parametrom w tablicy pozycji o indeksie 0). X,Y środek okręgu, R –promień, A – kąt	R/W	REAL
LinX, LinY, LinVel	Parametry dla ruchu po linii prostej (odpowiadają parametrom w tablicy pozycji o indeksie 0). X,Y punkt docelowy, Vel – prędkość	R/W	REAL
LinAxis	Wybrane osie dla realizacji interpolacji liniowej oraz ruchu PTP	R/W	INT

X: numer napędu 1…4

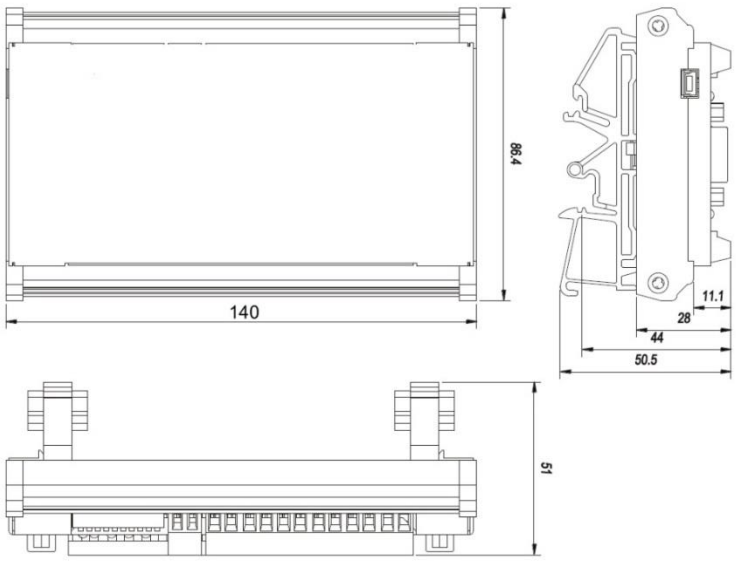
Rejestry modbus użytkownika

Rejestr	Opis	Dostęp	Zakres wartości
$I0 - $I1999	Rejestry użytkownika (wartości typu INT)	R/W	INT (-32768…32767)
$D0 -$D1998	Rejestry użytkownika (wartości typu DINT)	R/W	DINT (-2147483648…2147483647)
$R0 -$R1998	Rejestry użytkownika (wartości typu REAL)	R/W	REAL (zmiennoprzecinkowa)
UWAGA: Rejestry 1000…2000 mogą być zapisane do pamięci nieulotnej sterownika (rejestry 2100/2101/2102)			
MEM_INDEX	Przesunięcie indeksu dla dostępu indeksowanego	R/W	INT
MEM_INT**X**	Dostęp do rejestru $I(X+MEM_INDEX) pamięci modbus	R/W	INT
MEM_DINT**X**	Dostęp do rejestru $D(X+MEM_INDEX) pamięci modbus	R/W	DINT
MEM_REAL**X**	Dostęp do rejestru $R(X+MEM_INDEX) pamięci modbus	R/W	REAL

UWAGA: rejestry $I, $D i $R zajmują tą samą przestrzeń pamięci.

11. Parametry techniczne

Opis	Parametr
Zasilanie	12…26V DC
Pobór prądu	100mA@12V, 50mA@24V
Wyjścia sterowania napędami	4 (KROK, KIERUNEK, ZEZWOLENIE)
Wyjścia sterujące (EN, DIR, CLK) napędami M1..M4	Stan wysoki 5V, stan niski 0V Maks. obciążenie 20mA/wyjście
Wejścia IN1..IN8	Optoizolowane Stan niski: <**2V**, stan wysoki: **+4**…+**24V DC** Min. długość impulsu: 10ms.
Wejścia IN9..IN20	Stan niski: <**2V**, stan wysoki: **+3..24V DC** Możliwość podłączenia enkoderów inkrementalnych
Wyjścia OUT1.. OUT8	Tranzystorowe typu OC PNP, maks. 200mA / wyjście
Komunikacja	**COM0**: RS485 (optoizolowany), Protokół komunikacyjny: wewnętrzny do komunikacji z PC Baudrate: 38400 **COM1**: RS232 **COM2**: RS485 Protokół komunikacyjny: MODBUS-RTU SLAVE (COM1/2) MODBUS-RTU MASTER (COM2) Bity stopu: 1 Parzystość: brak Baudrate: 9600, 19200, 38400, 57600, 115200 **USB: 1.1, 2.0 (HID)**
Pamięć urządzenia	Program: 1 x 4000 komend Pozycje: 8 x 200 pozycji Pamięć użytkownika z dostępem przez Modbus: 2000 słów Zapis pamięci użytkownika do pamięci nieulotnej: 1000 słów
Zakres temperatur pracy	5..50° C
Masa	200g
Stopień szczelności	IP20



Rys. 4 Rysunek wymiarowy.

12. Historia zmian

Wersja	Firmware	Program PC
1.02	- wersja pierwsza	- wersja pierwsza
1.03	- obsługa przerwań (IN1 – IN8, Modbus) - funkcja MOVEPTP może przyjmować jako argument także numer z tablicy pozycji (wcześniej tylko nazwa pozycji)	- obsługa przerwań (IN1 – IN8, Modbus) - funkcja MOVEPTP może przyjmować jako argument także numer z tablicy pozycji (wcześniej tylko nazwa pozycji)
1.04	- usprawniona kontrola pozycji z enkodera - poprawione tryby błędów kontroli pozycji (ERRCTR) - poprawiona sygnalizacja błędów diodami LED	
1.05	- obsługa komend realizujących interpolację kołową	- obsługa komend realizujących interpolację kołową
1.06	- obsługa komend pętli WHILE-ENDWHILE	- obsługa komend pętli WHILE-ENDWHILE
1.08	- poprawiony stos JUMP-RETURN - zagnieżdżanie warunków WHILE-ENDWHILE - obsługa przerwań dla wejść IN9...IN20 (enkodery)	-
1.10	- poprawiony błąd odczytu wejść IN9…IN20 - optymalizacja funkcji (IF,WHILE) - nowe komendy ruchu liniowego MOVELIN - obsługa “breakpoint” **UWAGI**: RESET USTAWIEŃ, RE-KOMPILACJA	- optymalizacja funkcji (IF,WHILE) - nowe komendy ruchu liniowego MOVELIN - obsługa “breakpoint”
1.21	- optymalizacja funkcji warunek ELSE dla IF/ENDIF - nowe funkcje matematyczne (sin, cos tg,ctg, asin, acos, sqrt) - obsługa podglądu zmiennych podczas testowania programu **UWAGI**: RESET USTAWIEŃ	- obsługa nowych funkcji matematycznych - podgląd zmiennych podczas testowania programu - możliwość zaprogramowania sterownika z pliku - komenda SET nie musi być używana do ustawiania rejestrów i zdefiniowanych wcześniej zmiennych
1.22	- konfiguracja poziomu wejść wpływa także na sygnały bazowe i krańcowe - dodane programowe pozycje krańcowe - dodane rejestry Modbus dla programowych pozycji krańcowych - dodana sygnalizacja osiągnięcia pozycji krańcowych w rejestrze statusowym napędu **UWAGI**: RESET USTAWIEŃ	Wsparcie dla firmware v1.22: - poprawny odczyt i symulacja wejść/wyjść - dostęp do programowych pozycji krańcowych (rejestry LimLMX oraz LimRMX w WBL)
1.25	- optymalizacje w funkcjach kontroli napędu - możliwość programowania przez port COM0 (RS485)	
1.26	- poprawki w funkcji zadawania pozycji - predkosc pobierana ze zmiennej Vdest, a nie VXmax, by nie nadpisywac VXmax - poprawki w funkcjach interpolacji kolowej (ruch w napedach M3/M4 nie powoduje zatrzymywania interpolacji) - funkcja DOARC do wykonywania luku (ruchu po luku od punktu do punktu)	- wsparcie dla komendy DOARC
1.27	poprawiony błąd reakcji na przerwania wejść IN9..IN20	
1.33	obsluga warunku SWITCH/CASE	- wsparcie dla warunków SWITCH/CASE
1.40	- WSPÓŁPRACA TYLKO z MIC488-PC v140 lub wyższy - modbus MASTER - obsluga zmiennych indeksowych MEM_INT, MEM_DINT, MEM_REAL i MEM_INDEX - licznik TIM o podstawie 1MS	- wsparcie dla nowych funkcji firmware v140 - osobna zakładka konfiguracji parametrów komunikacyjnych
1.41	- zwiekszenie pamieci Modbus do 2000 - poprawka w funkcjach odczytu I/O Modbus Master	
1.51	- Dodany dostep bitowy (do rozmiaru 1998 bitow) i bajtowy do pamieci modbus uzytkownika - Zwiekszony rozmiar pamieci programu do 1400 linii - Ilosc bankow pamieci programu zmniejszona do 4 - Ilosc pamieci pozycji zmniejszona z 227 na 200 - Obsluga warunkow logicznych typu IF ARG1 && ARG2, IF ARG1 || ARG2	- wsparcie dla nowych funkcji firmware v150 - nowy interfejs dla sterowania manualnego
1.60	- dodana funkcja interpolacji liniowej dla 4 osi - dodana funkcja PULSE - mozliwosc zadawania nowej pozycji w trakcie ruchu **UWAGI**: RESET USTAWIEŃ	- wsparcie dla nowych funkcji firmware v160
1.61	- wsparcie dla modułu rozszerzeń IO MIC-EXP8I8O	- wsparcie dla modułu rozszerzeń IO MIC-EXP8I8O
1.62	- poprawka w odczycie bitów z rejestrów modbus - możliwość wyskoczenia z funkcji przerwania JUMP - możliwość wstrzymania TIMERow podczas PAUZY	- wsparcie dla nowych funkcji firmware v162
1.64	- poprawki w Modbus master – może poprawić komunikacje z niektórymi urządzeniami slave przy zakłóceniach na magistrali - dodane rejestry odczytu zadanej pozycji	
1.65	- dodane rejestry odczytu parametru ACC/DEC/VMAX przez Modbus dla trybu REAL - poprawka w sygnalizacji błędów za pomocą diod dla błędów Modbus master - poprawki w interpreterze w operacjach matematycznych - operacje błąd (np. dzielenie przez 0) przełącza sterownik w tryb błędu - rejestr globalnego statusu sterownika (M_STATUS modbus 10009)	
1.66	- poprawki w funkcji korekcji pozycji dla dużych wartości pozycji.	
1.70	- WSPÓŁPRACA TYLKO z MIC488-PC v170 lub wyższy	- wsparcie dla nowych funkcji firmware v170

	- zapis programu o rozmiarze do 4000 linii **- dostępny tylko jeden bank pamięci programu** - dodane parametry konfiguracyjne PosLimL/PosLimR/KpSpeed/ErrTime - zapis / odczyt/ reset rejestrów użytkownika z pamięci nieulotnej - dodane rejestry CFGINT_M, MEM_RETENT_CTRL oraz stałe MEM_READ, MEM_SAVE, MEM_RESET - dodane funkcje MEM_COPY, MEM_SET, MODBUSR_NREG **UWAGI**: RESET USTAWIEŃ	- drobne poprawki i dodatki
1.72	- poprawka w ladowaniu programu przez port COM (wczesniej pojawial sie timeout w programie na PC)	
1.73	- dostep przez modbus do wejsc z modulu rozszerzen (ten sam adres co wejsc wbudowanych)	
1.74	- poprawki związane z błędnym sygnalizowaniem zatrzymania mechanicznego - wyłączenie filtracji od wejść używanych jako przerwania	

RESET USTAWIEŃ: Po aktualizacji nastąpi przywrócenie ustawień fabrycznych.
RE-KOMPILACJA: Wymaga ponownego skompilowania programu w aktualnym oprogramowaniu MIC488-PC.

DEKLARACJA ZGODNOŚCI WE
Nr 03/11/2014

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

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

Oświadczamy, że wyprodukowany przez nas wyrób :

Nazwa: Programowalny, 4-osiowy kontroler trajektorii
Typ: MIC488

spełnia wymogi zasadnicze następujących dyrektyw:

- **2004/108/WE** „Kompatybilność elektromagnetyczna" (EMC), wdrożonej Ustawą z dnia 13 kwietnia 2007 r. o kompatybilności elektromagnetycznej (Dz.U.2007r. nr 82, poz. 556)

oraz spełnia wymogi norm i norm zharmonizowanych:
PN-EN 61000-6-1:2008 Kompatybilność elektromagnetyczna (EMC) -Odporność w środowisku lekko uprzemysłowionym
PN-EN 61000-6-3:2008 Kompatybilność elektromagnetyczna (EMC) -Norma emisji w środowisku lekko uprzemysłowionym

Niniejsza deklaracja zgodności jest podstawą do oznakowania wyrobu znakiem CE

Ta deklaracja odnosi się wyłącznie do wyrobu w stanie, w jakim został wprowadzony do obrotu i nie obejmuje części składowych dodanych przez użytkownika końcowego lub przeprowadzonych przez niego późniejszych działań.

Deklaracja zgodności nie obejmuje wszelkich modernizacji dokonanych niezgodnie z instrukcją obsługi i/lub zgody producenta.

Niniejsza deklaracja zgodności została sporządzona na wyłączną odpowiedzialność producenta.

Miejscowość: Dęborzyce

Data sporządzenia: 12.11.2014 r.