

# ML-PROG

## Opis programu i języka

### 1. Wstęp

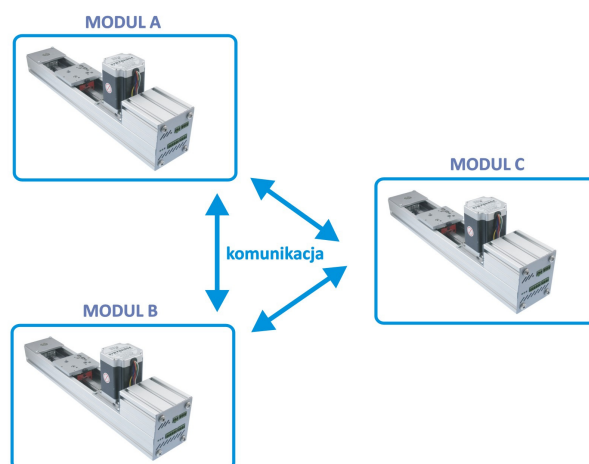
Oprogramowanie ML-PROG służy do tworzenia **programów ruchu** dla **sterowników silników krokowych** firmy **WObit** wyposażonych w programowalny generator trajektorii. Oprogramowanie współpracuje z:

- Modułami liniowymi **MLA-SI**
- Sterownikami **SIC184**

Programowanie w środowisku ML-PROG odbywa się w prostym języku tekstowym poprzez wprowadzanie odpowiednich komend. Przykładowo dla zadania ruchu modułu o 200 mm wprowadza się komendę **SetXdest(A,200)**, gdzie **A** oznacza adres modułu, którego ma dotyczyć ta komenda. **Każda** komenda w programie zawiera adres modułu. Podczas programowania modułów konkretne komendy są przesyłane do odpowiedniego sterownika (jeśli podłączonych jest kilka sterowników na jednej magistrali RS485). Dzięki temu istnieje możliwość jednoczesnego programowania do 16-stu sterowników.

Przy połączeniu kilku sterowników na jednej magistrali RS485 nie ma jednostki nadrzędnej, dlatego każdy moduł musi mieć nadany własny adres (z zakresu 0-15). Domyślnie jest to adres 0.

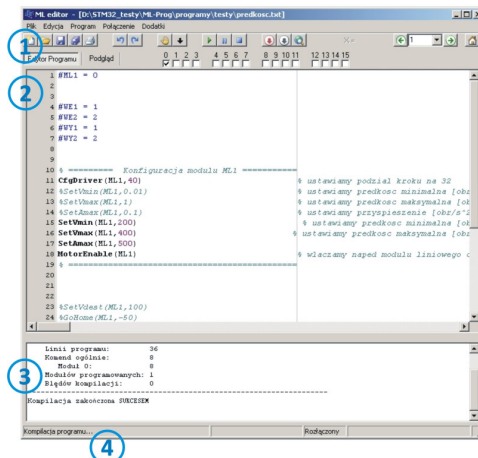
Synchronizacja między sterownikami odbywa się poprzez wysyłanie komunikatów przez dany moduł. Pozostałe moduły odbierają komunikaty i sprawdzają, czy któryś tyczy się niego. Jeśli tak, może wykonać zaprogramowaną wcześniej akcję.



Rys. 1 Komunikacja między modułami.

## 2. Opis programu ML-PROG

Program ML-PROG pozwala na tworzenie programów ruchu dla 16-stu modułów (sterowników) jednocześnie i ich równoległe programowanie (przy użyciu magistrali RS485 i konwertera USB-RS485). Program może być także przesłany do sterownika za pomocą niezależnej aplikacji (np. SIC184-KONFIGURATOR, komunikujący się ze sterownikiem SIC184 za pomocą złącza USB).












Rys. 2 Główne okno programu ML-PROG.

- 1 Pasek narzędzi i skrótów
- 2 Okno główne
- 3 Okno komunikatów kompilacji i programowania
- 4 Pasek statusu kompilacji i komunikacji

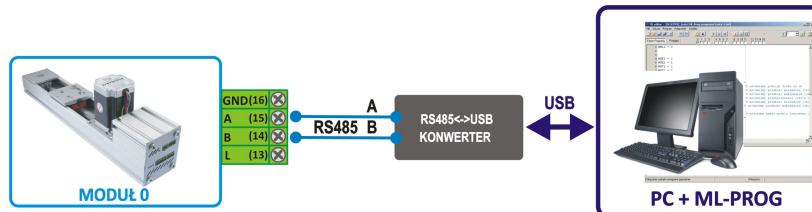
Pasek narzędzi i skrótów:



- 1 Pasek narzędzi:  
 Plik – operacje związane z plikiem programu  
 Edycja – edycja i opcje programu  
 Program – kompilacja programu, programowanie modułu, zatrzymywanie / uruchamianie programu  
 Połączenie – połączenie z konwerterem RS485<->USB, skanowanie podłączonych modułów, zmiana adresu modułu, podgląd transmisji  
 Dodatki – panel podpowiedzi komend
- 2 Skróty związane z operacją na pliku i jego edycją
- 3 Skróty związane z kompilacją programu i programowaniem modułu  
 - kompilowanie programu  
 - kompilowanie programu i przesyłanie do sterownika
- 4 Skróty związane z uruchamianiem / zatrzymywaniem programu  
 - uruchomienie programu w sterowniku  
 - wstrzymanie programu  
 - zatrzymanie programu
- 5 Skróty związane z połączeniem, oraz skanowaniem podłączonych modułów  
 - nawiązanie połączenia z konwerterem RS485<->USB  
 - zakończenie połączenia  
 - wyszukiwanie aktywnych modułów  
 - zmiana adresu modułu
- 6 Ręczne sterowanie modułem (dla wersji powyżej 1.16 dostępna tylko opcja bazowania)
- 7 Wybór okna edycji programu / podglądu programów dla poszczególnych modułów
- 8 Wybór adresu modułu (modułów) do programowania / sterowania



## 2.1 Nawiązanie połączenia z modułem i zmiana adresu

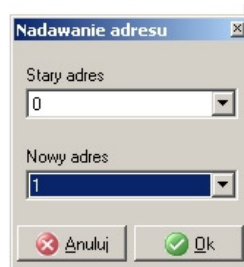
Sterownik należy połączyć z komputerem PC za pomocą konwertera RS485<->USB wg Rys. 3. Sterownik posiada domyślny adres o wartości 0. Jeśli na linii ma być połączonych kilka modułów jednocześnie, należy najpierw zmienić adresy modułów – każdy podłączając osobno do komputera PC poprzez konwerter RS485<->USB.



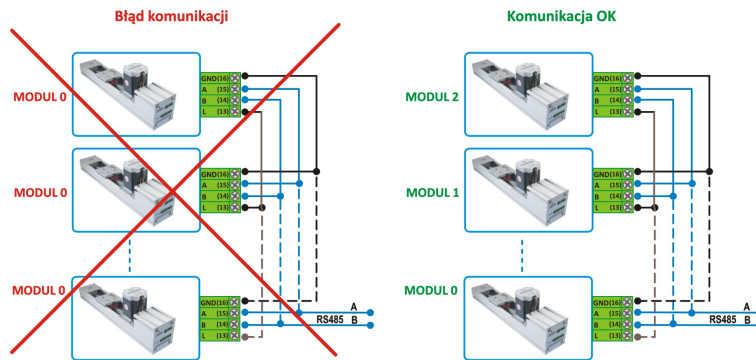
Rys. 3 Połączenie sterownika z komputerem PC

By zmienić adres modułu należy:

- Połączyć program ML-PROG z konwerterem RS485<->USB (**Połączenie** -> **Połącz** )
- Zmienić adres modułu domyślnego 0 na inny w zakresie 1..15 (**Połączenie** -> **Nadaj adres** , Rys. 4)



Rys. 4 Okno zmiany adresu modułu.



Rys. 5 Błędne i poprawne adresowanie modułów.



**UWAGA!**

Każdy moduł posiada domyślny adres o wartości 0. W przypadku programowania kilku modułów jednocześnie, przed ich pierwszym podłączeniem należy zmienić wartość adresu dla każdego modułu z osobna, tak by były one różne.

**2.2 Wyszukiwanie podłączonych (aktywnych) modułów**

Poprawnie zaadresowane moduły (każdy posiadający inny adres) mogą pracować na jednej magistrali, oraz mogą być wspólnie programowane. By sprawdzić ilość podłączonych modułów i ich adresy należy przejść do **Połączenie** -> **Skanuj moduły**. Dioda LINK powinna mrugać, sygnalizując aktywną komunikację. Skanowanie modułów trwa około 7 sekund. Po zakończonym procesie skanowania na pasku statusu pokazywana jest informacja o modułach (**Rys. 6**)



Rys. 6 Pasek statusu z widocznymi aktywnymi modułami.

**Przykłady odczytywania adresów:**

Adres 0 ..... Adres 15  
 |                    |  
 1100000000000001 - aktywne moduły o adresach 0,1 i 15)

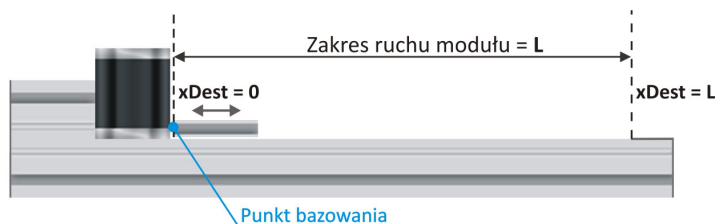
**3. Sterowanie napędem z poziomu języka ML-PROG**

Sterowanie napędem z poziomu języka programu ML-PROG polega na zadawaniu odpowiednich komend. Podstawowe komendy sterujące ruchem pozwalają zadać prędkość, pozycję względną (przesunięcie lub ilość obrotów silnika względem aktualnej pozycji), pozycje bezwzględną (zadanie położenia lub ilości obrotów silnika względem punktu bazowania) oraz przyspieszenie, z jakim realizowany jest ruch przy zadawania położenia lub prędkości.

Komendy interpretowane przez ML-PROG umożliwiają ponadto wysterowanie wyjść tranzystorowych, odczytywanie stanów wejść cyfrowych, a także realizację takich funkcji jak oczekiwanie, pętle i skoki. Możliwe jest również wysyłanie komunikatów pomiędzy sterownikami w protokole wewnętrznym lub sterowanie innymi sterownikami lub urządzeniami po magistrali MODBUS-RTU.

### 3.1 Bazowanie napędu

Po każdym uruchomieniu napędu zalecane jest wykonanie operacji bazowania, by ustalić pozycję początkową (zerową). Wykonuje się to za pomocą funkcji **GoHome(adres\_modułu, prędkość)**. Na rysunku **Rys. 7** pokazano zakres ruchu modułu, który oznaczony jest literą L. Funkcja bazowania powoduje ruch w stronę napędu, w którym to miejscu znajduje się wbudowany czujnik krańcowy. Dla sterowników SIC184 bazowanie powoduje ruch zawsze w tą samą stronę, aż do osiągnięcia sygnału z wejścia czujnika krańcowego 1 (Home 1).



Rys. 7 Punkt bazowania modułu.



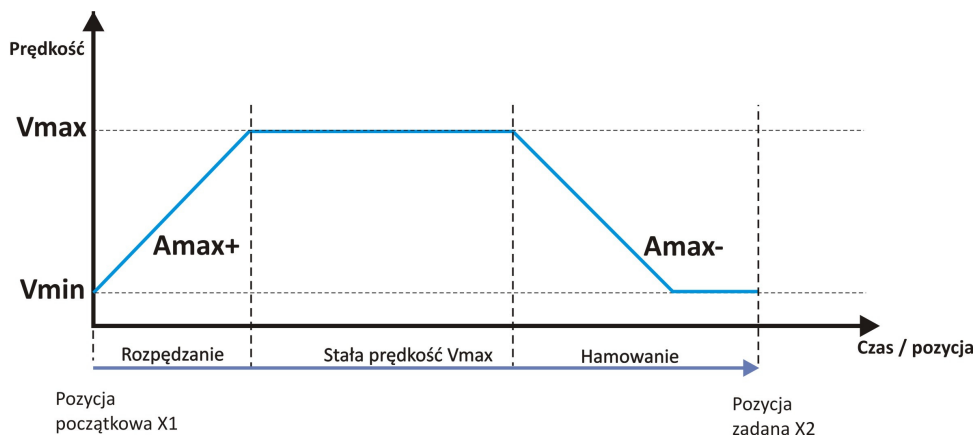
#### UWAGA!

Dla wersji modułu MLA-SI bez wbudowanego enkodera zaleca się wykonywanie bazowania także podczas pracy modułu (np. podczas dojeżdżania w pobliże pozycji bazowania), gdyż w przypadku przypadkowego zatrzymania wózka podczas jego ruchu tracona jest informacja o rzeczywistej pozycji względem punktu początkowego (bazowania).

### 3.2 Parametry ruchu napędu – prędkość i przyspieszenie, podział kroku, moc

Przy zadawaniu pozycji należy ustawić parametry ruchu (trajektorii) takie jak prędkość minimalna, maksymalna oraz przyspieszenie. W przeciwnym wypadku sterownik przyjmie parametry domyślne.

- **Prędkość minimalna (*SetVmin*)** – określa minimalną prędkość, z jaką wykonywany będzie ruch. Dotyczy trybów zadawania pozycji (*SetXmove*, *SetXdest*), a także prędkości *SetVdest*;
- **Prędkość maksymalna (*SetVmax*)** – określa maksymalną prędkość, jaką może osiągnąć napęd przy wykonywaniu ruchu. Dotyczy tylko trybów zadawania pozycji (*SetXmove*, *SetXdest*);
- **Przyspieszenie (*SetAmax*)** – określa przyspieszenie / opóźnienie, z jakim wykonywany będzie ruch. Dotyczy trybów zadawania pozycji (*SetXmove*, *SetXdest*) oraz prędkości (*SetVdest*).



Rys. 8 Trajektorja ruchu przy zadawaniu pozycji.

Ponadto możliwe jest skonfigurowanie podziału kroku sterownika oraz prąd silnika. Realizowane jest to za pomocą następujących komend:

- **CfgDriver**(adres\_modułu, podział\_kroku) – konfiguracja podziału kroku sterownika, możliwe wartości to: 2,8,16,32,64,10,20,40;
- **CfgPower**(adres\_modułu, moc\_sterownika) – konfiguracja prądu (mocy) sterownika, możliwe wartości 10...100 (%).

#### Domyślne parametry napędu :

	MLA-SI	SIC184
Prędkość minimalna (Vmin)	10 [mm/s]	0.05 [obr/s]
Prędkość maksymalna (Vmax)	50 [mm/s]	1 [obr/s]
Przyspieszenie (Amax)	50 [mm/s <sup>2</sup> ]	2 [obr/s <sup>2</sup> ]
Podział kroku (Drive)	16 [1/16 kroku]	16 [1/16 kroku]
Moc	70%	70%

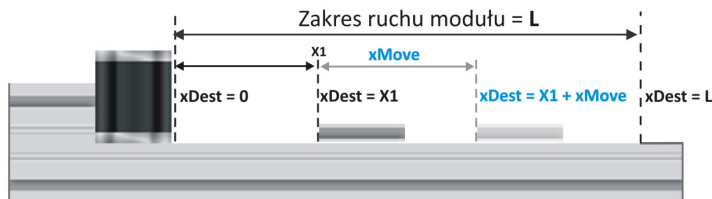
### 3.3 Zadawanie względnego / bezwzględnego położenia

Na rysunku **Rys. 9** pokazano ideę zadawania bezwzględnej oraz względnej pozycji. Zadanie pozycji bezwzględnej (**xDest**) oznacza zawsze ruch względem pozycji bazowej (zerowej). Do zadania pozycji bezwzględnej służy komenda **SetXdest**(adres\_modułu, pozycja\_bezwzględna), gdzie pozycja bezwzględna dla zbazowanego napędu może być wartością z zakresu 0...X, gdzie X – maksymalna droga jaką może osiągnąć napęd.

#### Reset pozycji bezwzględnej następuje w przypadkach:

- włączenia zasilania sterownika,
- zatrzymania wykonywania programu programu (podczas pracy bez enkodera),
- wykonania funkcji bazowania **GoHome**(),
- wykrycia sygnału z wejścia **Home1**,
- wykonania funkcji **ResetX**()

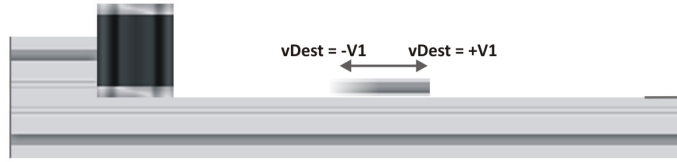
Zadanie pozycji względnej (**xMove**) oznacza wykonanie ruchu przez napęd o zadaną wartość względem pozycji aktualnej. Do zadania pozycji względnej służy komenda **SetXmove**(adres\_modułu, pozycja\_względna).



Rys. 9 Zadawanie pozycji bezwzględnej (xDest), oraz względnej (xMove) na przykładzie modułu MLA-SI.

### 3.4 Zadawanie prędkości

Prędkość zadawana jest w jednostkach [mm/s] dla modułów MLA-SI lub w [obr/s] dla sterowników SIC184. Do zadawania prędkości służy funkcja **SetVdest**(adres\_modułu, prędkość), gdzie prędkość ze znakiem „-” oznacza ruch napędu w przeciwną stronę. Ruch z zadaną prędkością trwa dopóki nie zadana zostanie inna komenda związana z ruchem (zadanie pozycji, hamowanie, bazowanie itp.).



Rys. 10 Zadanie prędkości z kierunkiem ruchu na przykładzie modułu MLA-SI.

### 3.5 Opis języka i idea programowania w ML-PROG

Programowanie w środowisku ML-PROG polega na zapisaniu komend sterujących w postaci tekstowej. Wszystkie komendy wymagają podania adresu sterownika (liczba z zakresu 0-15), dla którego ma być dana komenda przeznaczona, np.: **SetXdest** (*adres, pozycja*).

#### 3.5.1 Definicje nazw

Każdą wartość liczbową możemy zastąpić zdefiniowaną wcześniej nazwą poprzedzoną znakiem „#”. Przykładowo, by pod wartość 100 przypisać nazwę „Pozycja\_1” zapisujemy:

```
#Pozycja_1 100
```

Od tego momenty w dowolnym miejscu programu zamiast wartości 100 można użyć nazwy „Pozycja\_1” np.:

```
SetXdest (ML1, Pozycja_1)
```

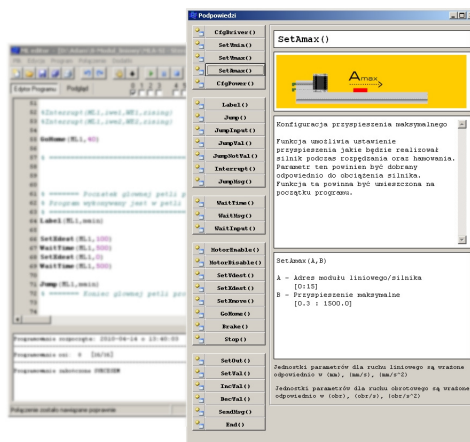
#### 3.5.2 Komentarze w programie

By dodać komentarz do programu należy poprzedzić go znakiem „%”, np.:

```
% To jest komentarz
```

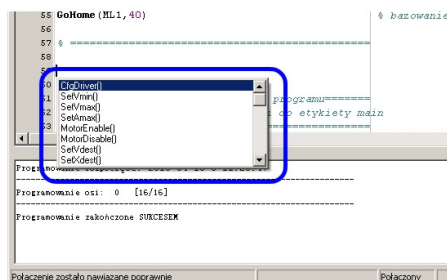
#### 3.5.3 Panel komend i podpowiedzi

Panel komend pozwala na szybkie dodanie funkcji wraz z parametrami, a także opisuje działanie funkcji. Dostęp do panelu funkcji możliwy jest poprzez przycisk z paska skrótów (**Dodatki -> Panel podpowiedzi komend**). Wciśnięcie przycisku w otwartym panelu podpowiedzi pozwala na dodanie do programu danej komendy w miejscu kursora.



Rys. 11 Panel podpowiedzi komend.

Wciśnięcie klawiszy CTRL+SPACJA na klawiaturze w oknie edycji programu powoduje pojawienie się okna z listą dostępnych funkcji i możliwość ich szybkiego dodania do programu.



Rys. 12 Lista funkcji.

### 3.5.4 Wprowadzanie parametrów ruchu – dostęp do rejestrów użytkownika

Parametry liczbowe wprowadza się jako liczby całkowite lub z przecinkiem np.:

**SetXdest** (0, 120)           % zadanie pozycji całkowitej 120 (obr lub mm)  
**SetXdest** (0, 2.105)        % zadanie pozycji niecałkowitej 2.105 (obr lub mm)

Zamiast wartości liczbowej można podać także adres rejestru sterownika, spod którego ma zostać odczytana wartość.



#### INFO

*Sterownik SIC184 posiada 50 rejestrów dowolnego przeznaczenia (adresy 50-98), dostępnych przez MODBUS-RTU. Użytkownik może przechowywać w tych rejestrach dowolne wartości, np. parametry ruchu silnika (prędkości, przyspieszenia, zadane pozycje). Zapisane przez użytkownika rejestry mogą być odczytywane przez program wykonywany w sterowniku. Dzięki temu możliwe jest wpływanie na pracę programu w sterowniku (zmiana prędkości ruchu, czasów itp.)*

*Rejestry użytkownika mogą być dodatkowo zapamiętane w pamięci nieulotnej sterownika – wówczas przesłane do nich wartości mogą być dostępne po ponownym włączeniu urządzenia.*

### Odczyt rejestrów z poziomu programu wykonywanego przez sterownik

Programista ma możliwość odwoływania się do rejestrów użytkownika z poziomu języka ML-PROG. Podczas tworzenia komend ruchu w programie ML-PROG używając komend takich jak: **SetVmin**, **SetVmax**, **SetAmax**, **SetVdest**, **SetXdest**, **SetXmove**, **WaitTime**, **GoHome** jako parametr można podać adres rejestru poprzedzając go znakiem „\$” np.:

**SetVdest** (0, \$50)   - zadanie prędkości przechowywanej w rejestrze 50  
**SetXmove** (0, \$98)   - zadanie prędkości przechowywanej w rejestrze 98

Rejestry można podawać tylko jako wartości **parzyste** (50,52,54...98), gdyż parametry ruchów przechowywane są zawsze w **dwóch sąsiednich rejestrach**. Np. podając adres rejestru 50, program odwoła się do rejestrów 50-51 przechowujących liczbę 4-bajtową.

Dane zapisywane w rejestrach 50...98 muszą zawierać liczby typu **REAL** (DWORD dla komendy **WaitTime**) i zajmować zawsze 2 kolejne rejestry, np. rejestry 50-51 → paramer A, 52-53 → paramer B itd...

## 3.6 Opis komend (funkcji) języka ML-PROG

### 3.6.1 Komendy konfiguracji sterownika i parametrów ruchu

<b>Podział kroku</b> <i>CfgDriver(A, X)</i>	Pozwala na ustawienie podziału kroku dla silnika krokowego. Dostępne wartości to: <b>2, 5, 8, 10, 16, 20, 32, 40, 64</b> . <b>Zalecane podziały to 8, 10, 16, 20 i 32. Większy podział zmniejsza moment silnika, ale zapewnia płynniejszy ruch.</b>
<b>Prędkość min/maks.</b> <i>SetVmin(A, X)</i> <i>SetVmax(A, X)</i>	Pozwala ustawić minimalną i maksymalną prędkość napędu przy osiągnięciu pozycji zadanej. Jednostki: <b>[mm/s]</b> dla MLA-SI, <b>[obr./s]</b> dla SIC184
<b>Przyspieszenie</b> <i>SetAmax(A, X)</i>	Pozwala ustawić przyspieszenie oraz zwalnianie napędu. Jednostki: <b>[mm/s<sup>2</sup>]</b> dla MLA-SI, <b>[obr./s<sup>2</sup>]</b> dla SIC184

### 3.6.2 Komendy ruchu

<b>Napęd On/Off</b> <i>MotorEnable(A)</i> <i>MotorDisable(A)</i>	Włączenie / wyłączenie końcówki mocy sterownika.
<b>Zadanie prędkości</b> <i>SetVdest(A, X)</i>	Pozwala zadać prędkość. Dokładny opis w rozdziale 3.4. Jednostki: <b>[mm/s]</b> dla MLA-SI, <b>[obr./s]</b> dla SIC184
<b>Zadanie pozycji</b> <i>SetXdest(A, X)</i> <i>SetXmove(A, X)</i>	Pozwala zadać pozycję. Kolejna komenda zostanie wykonana dopiero po osiągnięciu przez napęd zadanej pozycji. Jednostki: <b>[mm]</b> dla MLA-SI, <b>[obr]</b> dla SIC184 Dokładny opis w rozdziale 3.3.
<b>Bazowanie</b> <i>GoHome(A, X)</i>	Funkcja bazująca napędu. Funkcja ta powinna być zawsze wykonywana po włączeniu zasilania. Przed rozpoczęciem zadawania komend ruchu. Kolejna komenda zostanie wykonana dopiero po zakończeniu bazowania. Dokładny opis w rozdziale 3.1. Jednostki: <b>[mm/s]</b> dla MLA-SI, <b>[obr./s]</b> dla SIC184
<b>Zerowanie pozycji</b> <i>ResetX(A)</i>	Funkcja zerująca aktualną pozycję.
<b>Hamowanie</b> <i>Brake(A)</i>	Funkcja pozwala na płynne wyhamowanie silnika w przypadku, gdy znajdował się w ruchu np.: ... <i>SetVdest (0, 100)</i> % zadanie prędkości 100 mm/s <i>WaitTime (0, 5000)</i> % opóźnienie 5 sekund <i>Brake (A)</i> % płynne zatrzymanie napędu ...
<b>Zatrzymanie</b> <i>Stop(A)</i>	Funkcja nagłego zatrzymania modułu

### 3.6.3 Skoki warunkowe i bezwarunkowe

<p><b><u>Etykieta</u></b> <i>Label(A, ABCD)</i></p>	<p>Ustawia etykietę o dowolnej nazwie składającej się z 4 liter, do której może być wykonany skok.</p>
<p><b><u>Skok do etykiety</u></b> <i>Jump(A, ABCD)</i></p>	<p>Wykonuje skok do etykiety o podanej nazwie, np.:</p> <pre>... Label (0, <b>skok</b>) %ustawienie etykiety o nazwie „skok” ... Jump (0, <b>skok</b>) %skok do etykiety o nazwie „skok” ...</pre>
<p><b><u>Skok do etykiety, jeśli wejście</u></b> <i>JumpInput(A, ABCD, NrWe, STAN)</i></p>	<p>Wykonuje skok do etykiety, gdy na wejściu o numerze <b>NrWe</b> występuje określony stan (on - wejście aktywne, off – wejście nieaktywne) np.:</p> <pre>... Label (0, <b>skok</b>) %etykieta „skok” ... ... JumpInput (0, <b>skok</b>, 1, on ) %skok do etykiety o nazwie „skok”, gdym na wejście jeden będzie aktywny. ...</pre>
<p><b><u>Skok do etykiety, jeśli wartość</u></b> <i>JumpVal(A, ABCD, ZM, VAL)</i></p>	<p>Wykonuje skok do etykiety, gdy zmienna <b>ZM</b> będzie równa wartości VAL np.:</p> <pre>... IncVal (0, mem_1) % zwiększenie wartości zmiennej mem_1 o 1. JumpVal (0, <b>skok</b>, mem_1, 10) %skok do etykiety o nazwie „skok”, gdym wartość zmiennej mem_1 będzie równa 10. ... ... Label (0, <b>skok</b>) %etykieta „skok” ...</pre>
<p><b><u>Skok do etykiety, jeśli nie wartość</u></b> <i>JumpNoVal(A, ABCD, ZM, VAL)</i></p>	<p>Wykonuje skok do etykiety, gdy zmienna <b>ZM</b> będzie różna od wartości VAL np.:</p> <pre>... IncVal (0, mem_1) % zwiększenie wartości zmiennej mem_1 o 1. JumpVal (0, <b>skok</b>, mem_1, 10) %skok do etykiety o nazwie „skok”, gdym wartość zmiennej mem_1 będzie różna od 10. ... ... Label (0, <b>skok</b>) %etykieta „skok” ...</pre>
<p><b><u>Skok do etykiety, jeśli impuls na wejściu</u></b> <i>Interrupt(A, ABCD, NrWe, STAN)</i></p>	<p>Konfiguracją wejścia do wykonania skok do etykiety, gdy na wejściu o numerze <b>NrWe</b> pojawi się sygnał o określonym zboczu (<b>falling</b> - opadające, <b>rising</b> – rosnące, <b>falling_rising</b> – opadające lub rosnące). Funkcja Interrupt powinna znajdować się na początku (przy inicjalizacji) programu, np.:</p> <pre>Interrupt(0, int1, 1, rising) % inicjalizacja wejścia 1 - skok do etykiety „int1”, gdym na wejściu 1 pojawi się impuls rosnący ... ... Label (0, int1) %etykieta „int1”, do której wykonany zostanie skok ... ...</pre>
<p><b><u>Skok do etykiety, jeśli wiadomość</u></b> <i>JumpMsg(A, ABCD, MESG)</i></p>	<p>Wykonuje skok do etykiety, gdy pojawi się wiadomość o określonej nazwie np.:</p> <pre>... JumpMsg (0, <b>skok</b>, mes1) %skok do etykiety o nazwie „skok”, gdym pojawi się wiadomość „mes1” ... ... Label (0, <b>skok</b>) %etykieta „skok” do której wykonany zostanie skok ... ...</pre>

### 3.6.4 Opóźnienia i oczekiwania na zdarzenie

<p><b>Czekaj X ms</b> <i>WaitTime(A, X)</i></p>	<p>Funkcja wprowadzająca określona zwłokę czasową w ms między wykonywanymi komendami, np.:</p> <p>...</p> <p><i>SetOut (0, 1, on)</i>      % włączenie wyjścia 1</p> <p><i>WaitTime (0, 2000)</i>    % czekaj 2 sekundy</p> <p><i>SetOut (0, 1, off)</i>     % wyłączenie wyjścia 1</p> <p>...</p>
<p><b>Czekaj na wiadomość</b> <i>WaitMsg(A, MMSG)</i></p>	<p>Funkcja czekająca na wiadomość o określonej nazwie, np.:</p> <p>...</p> <p><i>WaitMsg (0, mes1)</i>    % oczekiwanie na wiadomość „mes1”</p> <p><i>SetOut (0, 1, on)</i>     % wyjście 1 zostanie załączone, gdy przyjdzie wiadomość „mes1”</p> <p>...</p>
<p><b>Czekaj na stan wejścia</b> <i>WaitInput(A, NrWe, STAN)</i></p>	<p>Funkcja czeka na określony stan wejścia, np.:</p> <p>...</p> <p><i>WaitInput (0, 1, on)</i>    % oczekiwanie na stan wysoki wejścia 1</p> <p><i>SetXdest (0, 100)</i>      % zadanie pozycji</p> <p>...</p>

### 3.6.5 Komendy sterujące przez MODBUS-RTU

<p><b>Ustaw wyjście</b> <b>(funkcja 5 MODBUS)</b> <i>WriteMCoil(A, B, C, STAN)</i></p>	<p>Funkcja umożliwia ustawienie (<b>STAN = set</b>) lub wyzerowanie (<b>STAN = reset</b>) bitu w rejestrze <b>C</b> w urządzeniu o adresie Modbus <b>B</b>. W momencie wysyłania komendy sterownik staje się urządzeniem nadrzędnym na magistrali.</p> <p>...</p> <p><i>WriteMCoil (0, 1, 1, on)</i> % ustawienie wyjścia w rejestrze 1 urządzenia o adresie Modbus 1</p> <p>...</p>
<p><b>Czekaj na wiadomość</b> <b>(funkcja 6 MODBUS)</b> <i>WriteM1Reg(A, B, C, VAL)</i></p>	<p>Funkcja umożliwia zapisanie wartości (<b>VAL</b>) typu INT w rejestrze <b>C</b> w urządzeniu o adresie Modbus <b>B</b>. W momencie wysyłania komendy sterownik staje się urządzeniem nadrzędnym na magistrali.</p> <p>...</p> <p><i>WriteM1Reg (0, 1, 1, 1234)</i> % zapisanie liczby 1234 do rejestru 1 urządzenia o adresie Modbus 1</p> <p>...</p>
<p><b>Czekaj na stan wejścia</b> <b>(funkcja 16 MODBUS)</b> <i>WriteM2Reg(A, B, C, VAL)</i></p>	<p>Funkcja umożliwia zapisanie wartości (<b>VAL</b>) typu DIN lub REAL do dwóch rejestrów C i C+1 w urządzeniu o adresie Modbus <b>B</b>.</p> <p>UWAGA: Liczba wysyłana jest jako REAL, gdy zawiera przecinek (np. 5.0), w przeciwnym wypadku wysyłana jest jako liczba całkowita DINT. W momencie wysyłania komendy sterownik staje się urządzeniem nadrzędnym na magistrali</p> <p>...</p> <p><i>WriteM2Reg (0, 1, 1, 12.345)</i> % zapisanie liczby typu REAL 12.345 do rejestrów 1 i 2 urządzenia o adresie Modbus 1</p> <p>...</p> <p><i>WriteM2Reg (0, 1, 1, 150000)</i> % zapisanie liczby typu DINT 150000 do rejestrów 1 i 2 urządzenia o adresie Modbus 1</p> <p>...</p>

### 3.6.6 Pozostałe funkcje

SetOut(A, NrWy, STAN)	Funkcja włącza lub wyłącza określone wyjście sterownika, np.: ... SetOut (0, 1, on)       % włączenie wyjścia 1 SetOut (0, 2, off)     % wyłączenie wyjścia 2 ...
SetVal(A, ZM, VAL)	Ustawia określoną wartość zmiennej (mem_1..Mem_9). Zakres wartości 0-999999. Np.: ... SetVal (0, mem_1, 100)     % ustawienie zmiennej mem_1 na 100 SetVal (0, mem_2, 2000)   % ustawienie zmiennej mem_2 na 2000 ...
IncVal(A, ZM) DecVal(A, ZM)	Zwiększenie lub zmniejszenie zmiennej (mem_1...mem_9) o 1, np.: ... IncVal (0, mem_1, 100)     % zwiększenie zmiennej mem_1 o 1 DecVal (0, mem_5, 2000)   % zmniejszenie zmiennej mem_5 o 1 ...
SendMsg(A, MMSG)	Funkcja pozwalająca wysłać wiadomość do innych modułów na magistrali RS485, np.: ... SendMsg (0, mes1)       % wysłanie wiadomości o nazwie „mes1” SendMsg (0, mes2)       % wysłanie wiadomości o nazwie „mes2” ...
IntEnable(A, NrWe)	Funkcja pozwala włączyć przerwanie na danym wejściu ... IntEnable (0, 1)   % włączenie przerwania na wejściu 1 ...
IntDisable(A, NrWe)	Funkcja pozwala wyłączyć przerwanie na danym wejściu ... IntDisable (0, 1)   % wyłączenie przerwania na wejściu 1 ...
Return(A)	Funkcja powrotu z przerwania. Powoduje powrót do miejsca programu, w którym nastąpiło przerwanie. ... Label (0, int1)   % etykieta przerwania 1 ... % program wykonywany w przerwaniu ... Return (0)       % powrót do programu przerwane przez przerwanie
End(A)	<b>Zakończenie działania programu.</b>

## 4. Przykładowe programy

### 4.1 Sterowanie dwoma modułami liniowymi MLA-SI

```
% Definicje nazw modułów
#ML1 = 0
#ML2 = 1
% Definicje nazw numerów wejść i wyjść sterownika
#WE1 = 1
#WE2 = 2
#WY1 = 1
#WY2 = 2
% Zdefiniowanie podziału kroku dla sterowników
#STEPS = 32

% Zdefiniowanie nazw prędkości minimalnych, maksymalnych, przyspieszeń (jednostki mm/s, mm/s^2)
#VMIN = 10
#VMAX = 200
#ACC = 100

% ===== Konfiguracja modułu ML1 =====
CfgDriver (ML1,STEPS) % ustawiamy podział kroku
SetVmin (ML1,VMIN) % ustawiamy predkosc minimalna [mm/s]
SetVmax (ML1,VMAX) % ustawiamy predkosc maksymalna [mm/s]
SetAmax (ML1,ACC) % ustawiamy przyspieszenie [mm/s^2]
MotorEnable (ML1) % włączamy napęd modułu liniowego o adresie 0

% ===== Konfiguracja modułu ML2 =====
CfgDriver (ML2,STEPS) % ustawiamy podział kroku
SetVmin (ML2,VMIN) % ustawiamy predkosc minimalna [mm/s]
SetVmax (ML2,VMAX) % ustawiamy predkosc maksymalna [mm/s]
SetAmax (ML2,ACC) % ustawiamy przyspieszenie [mm/s^2]
MotorEnable (ML2) % włączamy napęd modułu liniowego o adresie 0

% =====Bazowanie modułów =====
GoHome (ML1,1)
GoHome (ML2,1)

% ===== Początek głównej pętli programu ML1 =====
% Program wykonywany jest w pętli do etykiety main
% =====
Label (ML1,main) % Ustawienie etykiety o nazwie "main"

SetXdest (ML1,120) % Zadanie pozycji dla modułu 1 - 300 mm
SetOut (ML1,on) % Włączenie wyjścia WY1
SendMsg (ML1,m12a) % Wysłanie wiadomości o nazwie "m12a"
WaitMsg (ML1,m11a) % Czekanie na wiadomość o nazwie m11a (wysyła ją moduł ML2)

SetXdest (ML1,0) % Zadanie pozycji dla modułu 1 - 0
SetOut (ML1,off) % Wyłączenie wyjścia WY1
SendMsg (ML1,m12b) % Wysłanie wiadomości o nazwie "m12b"
WaitMsg (ML1,m11b) % Czekanie na wiadomość o nazwie m11b (wysyła ją moduł ML2)

Jump (ML1,main) % Skok do etykiety o nazwie "main"
% ===== Koniec głównej pętli programu ML1 =====

% ===== Początek głównej pętli programu ML2 =====
% Program wykonywany jest w pętli do etykiety main
% =====
Label (ML2,main) % Ustawienie etykiety o nazwie "main"

WaitMsg (ML2,m12a) % Czekanie na wiadomość o nazwie m12a (wysyła ją moduł ML1)
SetXdest (ML2,120) % Zadanie pozycji dla modułu 1 - 300 mm
SetOut (ML2,on) % Włączenie wyjścia WY1
SendMsg (ML2,m11a) % Wysłanie wiadomości o nazwie "m11a"

WaitMsg (ML2,m12b) % Czekanie na wiadomości o nazwie m12b(wysyła ją moduł ML1)
SetXdest (ML2,0) % Zadanie pozycji dla modułu 1 - 0
SetOut (ML2,off) % Wyłączenie wyjścia WY1
SendMsg (ML2,m11b) % Wysłanie wiadomości o nazwie "m11b"

Jump (ML2,main) % Skok do etykiety o nazwie "main"
% ===== Koniec głównej pętli programu ML2=====

% ===== Przerwanie od We1 modułu ML1 =====
Label (ML1,iwe1) % Ustawienie etykiety o nazwie "iwe1"
% ...
Jump (ML1,main) % Skok do początku programu
% ===== Przerwanie od We1 modułu ML2 =====
Label (ML2,iwe1) % Ustawienie etykiety o nazwie "iwe2"
% ...
Jump (ML2,main) % Skok do początku programu
% =====
```

## 4.2 Sterowanie silnikiem za pomocą sterownika SIC184

```
% Definicje nazw sterowników
#STER1 = 0

% Definicje nazw numerów wejść i wyjść sterownika
#WE1 = 1
#WE2 = 2
#WY1 = 1
#WY2 = 2

% Zdefiniowanie podziału kroku dla sterownika
#STEPS = 32

% Zdefiniowanie nazw prędkości minimalnych, maksymalnych, przyspieszeń
#VMIN = 0.05           % 0.05 obr/sek.
#VMAX = 3              % 3 obr/sek.
#ACC = 6               % 6 obr/sek^2.

% ===== Konfiguracja napędu STER1 =====
CfgDriver(STER1,STEPS) % ustawiamy podział kroku
SetVmin(STER1,VMIN)  % ustawiamy predkosc minimalna [obr/s]
SetVmax(STER1,VMAX)  % ustawiamy predkosc maksymalna [obr /s]
SetAmax(STER1,ACC)   % ustawiamy przyspieszenie [obr /s^2]
MotorEnable(STER1)  % włączenie końcówki mocy sterownika (napędu)

% =====Bazowanie napędu =====
GoHome(STER1,1)      % prędkość bazowania 1 [obr/s]

% ===== Początek głównej pętli programu=====
% Program wykonywany jest w pętli do etykiety main
% =====
Label(STER1,main)    % Ustawienie etykiety o nazwie "main"

SetXdest(STER1,50)  % Zadanie pozycji - 50 [obr]
SetOut(STER1,on)    % Włączenie wyjścia WY1
WaitTime(STER1,5000) % Czekanie 5 sek.
SetXdest(STER1,0)   % Zadanie pozycji dla modułu 1 - 0
SetOut(STER1,off)   % Wyłączenie wyjścia WY1
WaitTime(STER1,3000) % Czekanie 3 sek.

Jump(STER1,main)    % Skok do etykiety o nazwie "main"
% ===== Koniec głównej pętli programu ===

% ===== Przerwanie od We1 =====
Label(STER1,iwe1)   % Ustawienie etykiety o nazwie "iwe1"
% ...
Jump(STER1,main)   % Skok do początku programu
```

Dokładny opis stosowania poszczególnych funkcji znaleźć można w samouczku składającego się z gotowych programów (dostępny w katalogu *programu ML-PROG*).

## 5. Dodatek - spis komend interpretowanych przez ML-PROG v1.20

Komenda	Opis	Parametry
<b>Związane z konfiguracją sterownika i parametrami ruchu</b>		
<i>CfgDriver(A, X)</i>	Ustawia podział kroku sterownika	X: [2,8,16,32,64,10,20,40]
<i>SetVmin(A, X)</i>	Ustawia prędkość minimalną	Zależne od typu sterownika
<i>SetVmax(A, X)</i>	Ustawia prędkość maksymalną	Zależne od typu sterownika
<i>SetAmax(A, X)</i>	Ustawia przyspieszenie	Zależne od typu sterownika
<i>CfgPower(A, X)</i>	Ustawia moc sterownika (10..100%)	X: [10..100]
<b>Związane z zadawaniem ruchu modułu</b>		
<i>MotorEnable(A)</i>	Włączenie napędu modułu	brak
<i>MotorDisable(A)</i>	Wyłączenie napędu modułu	brak
<i>SetVdest(A, X)</i>	Zadanie prędkości docelowej	Zależne od typu sterownika
<i>SetXdest(A, X)</i>	Zadanie pozycji docelowej (bezwzględnej)	Zależne od typu sterownika
<i>SetXmove(A, X)</i>	Zadanie przesunięcia (pozycji względnej)	Zależne od typu sterownika
<i>GoHome(A, X)</i>	Funkcja bazowania modułu	Zależne od typu sterownika
<i>ResetX(A)</i>	Reset pozycji	brak
<i>Brake(A)</i>	Funkcja płynnego hamowania modułu	brak
<i>Stop(A)</i>	Funkcja nagłego zatrzymania modułu	brak
<b>Skoki warunkowe i bezwarunkowe</b>		
<i>Label(A, ABCD)</i>	Utworzenie etykiety, do której może być wykonany skok	<b>ABCD</b> - nazwa etykiety (maks. 4 znaki)
<i>Jump(A, ABCD)</i>	Skok bezwarunkowy do etykiety o nazwie <b>ABCD</b>	<b>ABCD</b> - nazwa etykiety (maks. 4 znaki)
<i>JumpInput(A, ABCD, NrWe, STAN)</i>	Skok do etykiety <b>ABCD</b> , gdy wejście o numerze <b>NrWe</b> przyjmie stan <b>STAN</b>	<b>ABCD</b> – nazwa etykiety (maks. 4 znaki) <b>NrWe</b> – 1 lub 2 <b>STAN</b> – On lub Off
<i>JumpVal(A, ABCD, ZM, VAL)</i>	Skok do etykiety <b>ABCD</b> , gdy zmienna w pamięci <b>ZM</b> będzie równa wartości zadanej <b>VAL</b> .	<b>ABCD</b> - nazwa etykiety (maks. 4 znaki) <b>ZM</b> - zmienna w pamięci [mem_1... mem_9] <b>VAL</b> - wartość do porównania [0 : 999999]
<i>JumpNoVal(A, ABCD, ZM, VAL)</i>	Skok do etykiety <b>ABCD</b> , gdy zmienna w pamięci <b>ZM</b> będzie różna od wartości zadanej <b>VAL</b> .	<b>ABCD</b> - nazwa etykiety (maks. 4 znaki) <b>ZM</b> - zmienna w pamięci [mem_1... mem_9] <b>VAL</b> - wartość do porównania [0 : 999999]
<i>Interrupt(A, ABCD, NrWe, STAN)</i>	Skok do etykiety <b>ABCD</b> w przypadku, gdy na wejściu o numerze <b>NrWe</b> pojawi się stan <b>STAN</b> . Skok ten jest wykonywany niezależnie od realizowanego programu. Funkcja ta powinna być umieszczona na początku programu.	<b>ABCD</b> – nazwa etykiety (maks. 4 znaki) <b>NrWe</b> – 1 lub 2 <b>STAN</b> – tryb zbrocza: <b>falling</b> - opadające, <b>rising</b> – rosnące, lub <b>falling_rising</b> – rosnące lub opadające
<i>JumpMsg(A, ABCD, MMSG)</i>	Skok do etykiety <b>ABCD</b> w przypadku, gdy został wcześniej wysłany komunikat <b>MMSG</b> przez inny moduł.	<b>ABCD</b> - nazwa etykiety (maks. 4 znaki) <b>MMSG</b> - nazwa wiadomości (maks. 4 znaki)
<b>Opóźnienia</b>		
<i>WaitTime(A, X)</i>	Funkcja oczekiwania w [ms]	X – wartość opóźnienia w [ms]
<i>WaitMsg(A, MMSG)</i>	Oczekiwanie na wiadomość ABCD od innego modułu	<b>MMSG</b> - nazwa wiadomości (maks. 4 znaki)
<i>WaitInput(A, NrWe, STAN)</i>	Oczekiwanie na określony stan <b>STAN</b> wejścia o numerze <b>NrWe</b>	<b>NrWe</b> – 1 lub 2 <b>STAN</b> – On lub Off
<b>Sterowanie przez MODBUS-RTU</b>		
<i>WriteMCoil(A, B, C, STAN)</i>	Ustawienie wyjścia (funkcja 5 Modbus)	B – adres modbus urządzenia slave C – numer rejestru <b>STAN</b> – stan wyjścia ( <b>set</b> lub <b>reset</b> )
<i>WriteM1Reg(A, B, C, VAL)</i>	Zapisanie 1 rejestru (liczby INT) (funkcja 6 Modbus)	B – adres modbus urządzenia slave C – numer rejestru VAL – wartość typu <b>INT</b> do zapisania
<i>WriteM2Reg(A, B, C, VAL)</i>	Zapisanie 2 rejestrów (liczby DINT lub REAL) (funkcja 16 Modbus)	B – adres modbus urządzenia slave C – numer rejestru VAL – wartość typu <b>DINT</b> lub <b>REAL</b> do zapisania
<b>Pozostałe</b>		
<i>SetOut(A, NrWy, STAN)</i>	Ustawienie wyjścia OC o numerze <b>NrWy</b> w stan <b>STAN</b>	<b>NrWy</b> – 1 lub 2 <b>STAN</b> – On lub Off
<i>SetVal(A, ZM, VAL)</i>	Przypisanie do zmiennej <b>ZM</b> wartości <b>VAL</b>	<b>ZM</b> - zmienna w pamięci [mem_1... mem_9] <b>VAL</b> – nowa wartość zmiennej [0 : 999999]
<i>IncVal(A, ZM)</i>	Zwiększenie wartości zmiennej <b>ZM</b> o 1	<b>ZM</b> - zmienna w pamięci [mem_1... mem_9]
<i>DecVal(A, ZM)</i>	Zmniejszenie wartości zmiennej <b>ZM</b> o 1	<b>ZM</b> - zmienna w pamięci [mem_1... mem_9]
<i>SendMsg(A, MMSG)</i>	Wysłanie wiadomości MMSG na magistrali RS485 do innych modułów.	<b>MMSG</b> - nazwa wiadomości (maks. 4 znaki)
<i>IntEnable(A, NrWe)</i>	Włączenie przerwania na wejściu <b>NrWe</b> .	<b>NrWy</b> – 1 lub 2
<i>IntDisable(A, NrWe)</i>	Wyłączenie przerwania na wejściu <b>NrWe</b> .	<b>NrWy</b> – 1 lub 2
<i>Return(A)</i>	Powrót z przerwania do programu	brak
<i>End(A)</i>	Funkcja umożliwiająca zakończenie programu	brak

gdzie A – adres sterownika (modułu): [0...15].



PPH WObit E.J.K. SC  
61-474 Poznań, ul. Gruszkowa 4  
tel.(061) 835 06 20, fax.(061) 835 07 04  
e-mail: wobit@wobit.com.pl [www.wobit.com.pl](http://www.wobit.com.pl)