PikoCNC PLC manual

wersja 3.0

2024.03.21

PLC (*Programmable Logic Controller*)

„Podstawową zasadą pracy sterowników jest praca cykliczna, w której sterownik wykonuje kolejno po sobie pojedyncze rozkazy programu w takiej kolejności, w jakiej są one zapisane w programie. Na początku każdego cyklu program odczytuje "obraz" stanu wejść sterownika i zapisuje ich stany (obraz wejść procesu). Po wykonaniu wszystkich rozkazów i określeniu (wyliczeniu) aktualnego dla danej sytuacji stanu wyjść, sterownik wpisuje stany wyjść do pamięci będącej obrazem wyjść procesu a system operacyjny wysterowuje odpowiednie wyjścia sterujące elementami wykonawczymi. Tak więc wszystkie połączenia sygnałowe spotykają się w układach (modułach) wejściowych sterownika, a program śledzi ich obraz i reaguje zmianą stanów wyjść w zależności od algorytmu.” - Wikipedia

Wstęp

Procesor PLC w kontrolerze umożliwia dowolne łączenie sygnałów, operacje logiczne i czasowe między nimi – czyli o wiele bardziej elastyczne podejście niż „na stałe” ustalone możliwości i przypisane zasoby.

Procesor PLC wykonuje program cyklicznie 1000 razy/sek. Praca każdego cyklu przebiega według następującego algorytmu:

1. Odczyt wejść, stanów kontrolera typu R (read)
2. Wykonanie programu
3. Zapis wyjść oraz stanów kontrolera typu W (write)

PLC rozpoczyna pracę od razu po nawiązaniu połączenia z kontrolerem. Przetwarzaniem programu zajmuje się wyłącznie kontroler i nie podejmie pracy jeśli nie dostanie programu dla PLC lub wystąpił jakiś błąd z nim związany – w takiej sytuacji wystąpi E_STOP z odpowiednim komunikatem.

Rozmiar maksymalny programu PLC zależy od modelu płyty. Dla płyt typu A,B jest to 120 instrukcji, dla płyt typu E 150 instrukcji.

Programowanie

Program jest wykonywany linia po linii, jedna linia tekstu może zawierać tylko jedną instrukcję.

Podstawowym, bardzo często używanym w dalszej części pojęciem jest „bufor linii” (w skrócie BL). Jest to zmienna, która jest swoistym pośrednikiem we wszystkich działaniach. Zmienna jest typu bool czyli może przyjmować dwie wartości „1” lub „0”.

Generalnie instrukcje dzielą się na następujące grupy:

- Zapis bitu z rejestru do BL (tabela 1).
- Zapis BL do bitu w rejestrze (tabela 2).
- Operacje logiczne na BL (tabela 3).
- Instrukcje sterujące przebiegiem programu w zależności od stanu BL (IF, ELSE, ENDIF) (tabela 4).
- Instrukcje dla trybu drabinkowego (tabela 5).

Przykład: przepisanie stanu wejścia numer 0 na wyjście numer 5.

```
<< IN 0   // instrukcja przepisuje stan wejścia numer 0 do BL
>> OUT 5  // instrukcja przepisuje stan BL na wyjście nr 5
```

BL utrzymuje bieżący stan do momentu, aż go nie zmienimy.

Przykład: przepisanie stanu wejścia numer 0 na wyjścia numer 5,6. Na wyjście 7 podawana jest zanegowany stan wejścia 0.

```
<< IN 0    // instrukcja przepisuje stan wejścia numer 0 do BL
>> OUT 5   // przepisanie stanu BL na wyjście nr 5
>> OUT 6   // przepisanie stanu BL na wyjście nr 6
!> OUT 7   // przepisanie zanegowanego stanu BL na wyjście nr 7
```

Tabela 1. Operacje zapisu do bufora linii

Skróty: SRC – adres źródłowy. BL – bufor linii.

Składnia	Opis
<< SRC	Przepisanie SRC do BL
!< SRC	Przepisanie zanegowanej wartości SRC do BL
+< SRC	Wpisanie „1” do BL jeśli wystąpiło narastające zbocze w SRC, inaczej wpisanie „0”. Detekcja zbocza to porównanie bieżącego stanu bitu oraz stanu tego bitu na końcu poprzedniego cyklu.
-< SRC	Wpisanie „1” do BL jeśli wystąpiło opadające zbocze w SRC, inaczej wpisanie „0”
^< SRC	Wpisanie „1” do BL jeśli wystąpiło opadające lub narastające zbocze w SRC, inaczej wpisanie „0”
=< SRC	Wpisanie „1” do BL jeśli nie wystąpiło opadające lub narastające zbocze w SRC, inaczej wpisanie „0”
C< SRC [v2]	Przepisanie SRC do BL, następnie wyzerowanie SRC
<<_FIRST_RUN [v2]	Ustawia BL przy pierwszym obiegu programu PLC. Może być wykorzystane do inicjowania rejestrów.

Tabela 2. Operacje zapisu do rejestrów

Skróty: DST – adres przeznaczenia. BL – bufor linii.

Składnia	Opis
>> DST	Bezpośrednie przepisanie BL do DST
!> DST	Przypisanie zanegowanej wartości BL do DST
S> DST	Wpisanie „1” do DST jeśli BL=1, inaczej DST nie zostanie zmienione.
R> DST	Wpisanie „0” do DST jeśli BL=1, inaczej DST nie zostanie zmienione.
T> DST	Zmiana stanu DST na przeciwny jeśli BL=1, inaczej DST nie zostanie zmieniony.
H> DST	Wpisanie „1” do DST.
L> DST	Wpisanie „0” do DST

Tabela 3. Operacje logiczne na buforze linii

Skróty: SRC – adres źródłowy. BL – bufor linii.

Składnia	Opis
NOT	Negacja BL<= NOT BL.
AND SRC	Wpisanie do BL wyniku operacji BL<=BL AND SRC
AND! SRC	Wpisanie do BL wyniku operacji BL<=BL AND (NOT SRC)
OR SRC	Wpisanie do BL wyniku operacji BL<=BL OR SRC
OR! SRC	Wpisanie do BL wyniku operacji BL<=BL OR (NOT SRC)

Składnia	Opis
XOR SRC	Wpisanie do BL wyniku operacji BL<=BL XOR SRC
XOR! SRC	Wpisanie do BL wyniku operacji BL<=BL XOR (NOT SRC)

Tabela 4. Składnia instrukcji skoków warunkowych

Skróty: BL – bufor linii.

Składnia	Opis
IF_TRUE_BEGIN	Kolejne instrukcje będą wykonywane jeśli BL=1, w innym przypadku nastąpi skok do ELSE_BEGIN lub do ENDIF jeśli brak ELSE_BEGIN. Instrukcje IF..ENDIF mogą być dowolnie zagnieżdżone - czyli jeden warunek IF ENDIF może zawierać w sobie kolejny warunek IF..ENDIF.
IF_FALSE_BEGIN	Kolejne instrukcje będą wykonywane jeśli BL=0, w innym przypadku nastąpi skok do ELSE_BEGIN lub ENDIF jeśli brak ELSE_BEGIN.
ENDIF	Kończy sekwencję IF.. ELSE..
ELSE_BEGIN	Jw.
IF.TRUE [v2]	Jeśli BL=1 wykonanie kolejnej (jednej) instrukcji.
IF..TRUE [v2]	Jeśli BL=1 wykonanie kolejnych (dwóch) instrukcji.
IF...TRUE [v2]	Jeśli BL=1 wykonanie kolejnych (trzech) instrukcji.

IF....TRUE [v2]	Jeśli BL=1 wykonanie kolejnych (czterech) instrukcji. Liczba kropek określa ile instrukcji będzie wykonanych (1-4), jeśli BL=0, to licznik programu zwiększany jest automatycznie o liczbę kropek. Instrukcje IF..TRUE są szybsze od IF_TRUE_BEGIN i zajmują mniej miejsca w pamięci, jednak wymagają więcej uwagi przy programowaniu. Nie można ich łączyć z innymi IF, ELSE, ENDIF.
	
	Przykład:
	`IF....TRUE`
	`  T> M 10`
	`  MOV. R6 0`
	`  ADD. R0 1`
	`  AND. R1 R0`

IF.FALSE [v2] IF..FALSE IF...FALSE IF....FALSE	Jw. z tym, że wykonanie instrukcji jeśli BL=0.

Tabela 5. Instrukcje dla trybu drabinkowego

Skróty: BL – bufor linii. BLD – indeks głębokości bufora linii

Składnia	Opis
BEG	(BEGIN) Rozpoczęcie trybu drabinkowego BL<=1, BLD <=0
END	Zakończenie gałęzi.
PAR	(PARALLEL) Otwarcie gałęzi równolegle połączonych elementów.
SER	(SERIAL) Otwarcie gałęzi szeregowo połączonych elementów.

Operacje na całych rejestrach [v2]

W wersji 2 PLC wprowadzono obsługę dodatkowych, 32-bitowych rejestrów. Mamy do dyspozycji 22 rejestry oznaczone R0 – R21.

Wprowadzono także zestaw instrukcji (komend), które umożliwiają operacje na całych rejestrach, a nie jak wcześniej tylko na poszczególnych bitach.

Składnia: INST. <D> <S>

Gdzie:

INST. - kod instrukcji zakończony kropką

<D> Nazwa rejestru na którym zostanie wykonana operacja: R0-R21, IN, OUT, MEMO …

<S> Operand. Nazwa rejestru jw. lub wartość bezpośrednia. Wartości bezpośrednie mogą być podane jako liczby:

- Dziesiętne (ze znakiem): 1, 12, -67
- Szesnastkowe: $FE, $FAB4A0001
- Binarne: 0b10001, 0b10111110 lub: 0b1000,1011,0000,1100 (można używać przecinka dla poprawienia czytelności)

Przykłady:

```
MOV. R2 R10 //przepisanie do rejestru R2 zawartości rejestru R10
ADD. R5 R6  //dodanie do R5 zawartości R6
MOV. R7 IN  //przepisanie rejestru IN do R7
OR. OUT R4  //operacja OR na rejestrze OUT

MOV. R2  1  //wpisanie do rejestru R2 wartości 1
ADD. R5  2  //dodanie liczby 2 do R5

AND. R10 0b10,1010
MOV. R5 $FE
MOV. R5 0b1111
```

Tabela podstawowych operacji

Składnia	Opis

MOV. <D> <S>	Przepisanie <S> do <D>.
	
	Przykład:
	`MOV. R2 R10 //przepisanie do rejestru R2 zawartości rejestru R10`
	`MOV. R5 1 //wpisanie do R5 liczby 1`

ADD. <D> <S>	Dodanie <S> do <D>. (Arytmetyka ze znakiem)
	
	Przykład:
	`ADD. R2 R3 //dodanie do rejestru R2 rejestru R3`
	`ADD. R5 2 //dodanie liczby 2 do R5`

SUB. <D> <S>	Odjęcie <S> do <D>. (Arytmetyka ze znakiem)
MUL. <D> <S>	Mnożenie <D> razy <S>.
DIV. <D> <S>	Dzielenie <D> przez <S>. UWAGA! Dzielenie zostanie wykonane jeżeli wartości <S> będą różne od zera.
MOD. <D> <S>	Wartość modullo (reszta z dzielenia <D> przez <S>)
SHR. <D> <S>	Przesuniecie bitowe <D> w prawo. <S> zawiera liczbę bitów przesunięcia. Przykład: `SHR. R5 1 //Przesuniecie R5 o jeden bit w prawo`
SHL. <D> <S>	Przesuniecie bitowe <D> w lewo. <S> zawiera liczbę bitów przesunięcia.
AND. <D> <S>	Iloczyn logiczny. <D> = <D> AND <S>.
ANDN. <D> <S>	Iloczyn logiczny z zanegowaną wartością. <D> = <D> AND (NOT <S>).
OR. <D> <S>	Suma logiczna. <D> = <D> OR <S>.
NOT. <D>	Negacja <D>.
XOR. <D> <S>	Operacja XOR (alternatywa rozłączna) <D> = <D> XOR <S>
TCM. <D> <S>	Operacja ustawia BL jeśli <D> zmienił wartość w porównaniu do poprzedniego obiegu pętli. <S> zawiera maskę bitową. Pod uwagę brane są tylko bity ustawione w masce. Przykład: `TCM. IN 0b1111 //Ustawi BL jeśli na którymś z pierwszych czterech wejść nastąpi zmiana stanu`
TEQ. <D> <S>	Operacja ustawia BL jeśli <D> oraz <S> są równe. Przykład: `TEQ. R2 10 //Ustawi BL jeśli R2 zawiera wartość 10`
TNE. <D> <S>	Operacja ustawia BL jeśli <D> oraz <S> są różne. Przykład: `TNE. R2 10 //Ustawi BL jeśli R2 jest różne od 10`
TGT. <D> <S>	Operacja ustawia BL jeśli <D> jest większe niż <S>.
TLT. <D> <S>	Operacja ustawia BL jeśli <D> jest mniejsze niż <S>.
TGE. <D> <S>	Operacja ustawia BL jeśli <D> jest większe lub równe <S>.

Składnia	Opis
TLE. <D> <S>	Operacja ustawia BL jeśli <D> jest mniejsze lub równe <S>.
TCH. <D>	Operacja ustawia BL jeśli <D> jest różny od swojej wartości w poprzednim obiegu pętli.
TSM. <D>	Operacja ustawia BL jeśli <D> nie zmienił swojej wartości względem poprzedniego obiegu pętli.
TUP. <D>	Operacja ustawia BL jeśli <D> zwiększył wartość względem poprzedniego obiegu pętli.
TDW. <D>	Operacja ustawia BL jeśli <D> zmniejszył wartość względem poprzedniego obiegu pętli.
SNB. <D> <S>	Ustawia w <D> określoną w <S> liczbę pierwszych bitów. Reszta bitów jest zerowana. Przykład: `SNB. R10 5 //Po operacji R10 zawiera wartość 0b11111`
FSB. <D> <S>	Wpisuje do <D> numer pierwszego zapalonego bitu w <S>. Przykład: `FSB. R10 $F00 //po operacji R10 zawiera wartość 8`
CSB. <D> <S>	Wpisuje do <D> liczbę zapalonych bitów w <S>. Przykład: `CSB. R10 0b1111010 //po operacji R10 zawiera wartość 5`
MWR. <D> <S>	Zapisywanie bitów w <D> zależnie od aktualnego stanu BL oraz maski bitowej w <S>. Przykład: `MWR. R10 0b1111 //Pierwsze cztery bity R10 przyjmą taki sam stan jaki ma aktualnie BL. Pozostałe bity nie będą zmienione`
MSR. <D> <S>	Ustawianie bitów zależnie od aktualnego stanu BL oraz maski bitowej w <S>. Przykład: `MSR. R10 0b1111 //Jeśli BL jest w stanie wysokim, to ustawione bity w <S> zostaną ustawione w <D>`
MCR. <D> <S>	Kasowanie bitów zależnie od aktualnego stanu BL oraz maski bitowej w <S>. Przykład: `MCR. R10 0b1111 //Jeśli BL jest w stanie wysokim, to ustawione bity w <S> zostaną wyzerowane w <D>`
MTR. <D> <S>	Negacja bitów zależnie od aktualnego stanu BL oraz maski bitowej w <S>. Przykład: `MTR. R10 0b1111 //Jeśli BL jest w stanie wysokim, to ustawione bity w <S> zostaną zanegowane w <D>`
BHR. <D> <S>	Ustawia w <D> bit o numerze podanym w <S>. Pozostałe bity nie są zmieniane. Wykonanie niezależne od stanu BL. Przykład: `BHR. R10 4 //Po operacji, bit numer 4 w R10 jest ustawiony`
BLR. <D> <S>	Kasuje w <D> bit o numerze podanym w <S>. Pozostałe bity nie są zmieniane. Wykonanie niezależne od stanu BL.

Składnia	Opis
	Przykład: `BLR. R10 4 //Po operacji, bit numer 4 w R10 jest wygaszony`
BWR. <D> <S>	W <D> bit o numerze podanym w <S> przyjmuje wartość BL. Przykład: `BWR. R10 4 //Po operacji, bit numer 4 w R10 przyjmuje wartość BL`
BSR. <D> <S>	Ustawianie bitu <D> zależnie od aktualnego stanu BL. <S> zawiera numer bitu. Jeśli BL jest w stanie niskim <D> pozostaje bez zmian.
BCR. <D> <S>	Kasowanie bitu <D> zależnie od aktualnego stanu BL. <S> zawiera numer bitu. Jeśli BL jest w stanie niskim <D> pozostaje bez zmian.
BTR. <D> <S>	Negowanie bitu <D> zależnie od aktualnego stanu BL. <S> zawiera numer bitu. Jeśli BL jest w stanie niskim <D> pozostaje bez zmian.
BRR. <D> <S>	Ustawia BL wartością bitu z <D>, którego numer podano w <S>.
MAX. <D> <S>	Jeżeli <S> jest większa od <D>, to jest przepisywana do <D>.
MIN. <D> <S>	Jeżeli <S> jest mniejsza od <D>, to jest przepisywana do <D>.
PUSH. <D>	Odłożenie <D> na stos. Jeżeli brakuje nam rejestrów, możemy tymczasowo odłożyć rejestr na stos, a później przywrócić jego wartość instrukcją POP.

POP. <D>	Pobranie <D> ze stosu. Instrukcji PUSH oraz POP musi być tyle samo, kolejność pobierania musi być odwrotna do odkładania!
	
	Przykład:
	`PUSH R5`
	`PUSH R6`
	…
	…
	`POP R6`
	`POP R5`

STAT. <D> <S>	Pobranie do <D> stanu maszyny. <S> zawiera numer parametru. Patrz rozdział „Odczyt stanu kontrolera”

BTD. <D> <S>	Konwersja liczby binarnej do dziesiętnej. <S> zawiera liczbę do konwersji. W <D> otrzymujemy wynik. Uzyskana liczba dziesiętna może mieć maksymalnie 8 cyfr. Liczba binarna może być ujemna, ale wynik jest zawsze dodatni.
	
	Przykład:
	`MOV. R2 $4D3 // czyli 1235 dziesiętnie`
	`BTD. R5 R2 // Po wykonaniu operacji w R5 mamy wartość $1235`

Tablice LUT (Lookup table)

Tablice LUT są to obszary pamięci, z której możemy odczytywać wartość podając adres (indeks). Adres bazowy ustala komenda LUTBAS. Domyślnie wskazuje na rejestr R0. Cyferka przy komendzie mówi o rozmiarze danych wyrażonych w bitach. Jeden rejestr ma 32 bity, zatem pomieści 32 pozycje o rozmiarze 1bit, 16 pozycji o rozmiarze 2 bity itd. Podany indeks odczytu nie musi mieścić się w jednym rejestrze np. LUT1 R5, 33 odczyta wartość z drugiego bit rejestru R1 itd. Rozmiar tablicy LUT ogranicza tylko ilość rejestrów.

Przykład:

```
MOV. R0 $FEDC6734 // Inicjujemy dane tablicy
MOV. R1 $AB123456 // dane zajmują dwa rejestry...

LUT4. R5 0  // wpisanie liczby $4 do R5
LUT4. R5 7  // wpisanie liczby $F do R5
LUT4. R5 15 // wpisanie liczby $A do R5
LUT8. R5 1  // wpisanie liczby $67 do R5
LUT16. R5 0 // wpisanie liczby $6734 do R5
LUT16. R5 2 // wpisanie liczby $3456 do R5
```

Składnia	Opis
LUT1. <D> <S>	Przepisanie do <D> wartości z 1 bitowej tablicy LUT. <S> zawiera adres w tablicy.
LUT2. <D> <S>	Przepisanie do <D> wartości z 2 bitowej tablicy LUT. <S> zawiera adres w tablicy.
LUT4. <D> <S>	Przepisanie do <D> wartości z 4 bitowej tablicy LUT. <S> zawiera adres w tablicy.
LUT8. <D> <S>	Przepisanie do <D> wartości z 8 bitowej tablicy LUT. <S> zawiera adres w tablicy.
LUT16. <D> <S>	Przepisanie do <D> wartości z 16 bitowej tablicy LUT. <S> zawiera adres w tablicy.
LUT32. <D> <S>	Przepisanie do <D> wartości z 32 bitowej tablicy LUT. <S> zawiera adres w tablicy.
LUTBAS. <D>	Ustawienie <D> jako adresu bazowego tablicy LUT. Możemy podać tylko rejestry R0 – R21. Domyślnie rejestrem bazowym jest R0.

Odczyt stanu kontrolera

Komendą STAT. Możemy odczytać niektóre stany kontrolera np.:

```
STAT. R5 2 // Odczyt pozycji osi nr 2(Z) (G53.1) w impulsach.
```

Tabela indeksów.

Indeks	Opis
0	Pozycja osi nr 0 (G53.1) wyrażona w impulsach (pozycja G53.1 zawiera offset narzędzia)
1	Pozycja osi nr 1 (G53.1) wyrażona w impulsach
2	Pozycja osi nr 2 (G53.1) wyrażona w impulsach
3	Pozycja osi nr 3 (G53.1) wyrażona w impulsach
4	Pozycja osi nr 4 (G53.1) wyrażona w impulsach
10	Pozycja osi nr 0 (G53) wyrażona w impulsach
11	Pozycja osi nr 1 (G53) wyrażona w impulsach
12	Pozycja osi nr 2 (G53) wyrażona w impulsach
13	Pozycja osi nr 3 (G53) wyrażona w impulsach
14	Pozycja osi nr 4 (G53) wyrażona w impulsach
20	Pozycja licznika enkodera
25	Pozycja osi technicznej
30	ADC kanał 0
31	ADC kanał 1
40	Aktualna prędkość zadana (F)
41	Procentowe przyśpieszenie F (ruchu robocze)
42	Procentowe przyśpieszenie F (ruchu ustawcze G0)
50	Aktualnie zadane obroty (S)
51	Procentowe przyśpieszenie S
52	Tryb CSS (zwraca 1 jeśli aktywny, inaczej zwraca zero)
53	Aktualne S
55	Załączony tryb S_GEN
60	Odebrana ramka modułu zdalnych wejść (REMOTE)
61	Odczyt wejść „klejących”. Normalnie wejścia skanowane są 1000 razy/s i jeśli nastąpi jakaś zmiana stanu pomiędzy tymi okresami, to PLC je przeoczy. Wejścia „klejące” zapamiętują (zależnie od typu kontrolera) zbocza opadające lub narastające jakie się wydarzyły w międzyczasie na wybranych wejściach. Dla kontrolera „M” są to wejścia 2,3,4,5 oraz 18,19,20,21 i źródło sygnału musi być NC. Dla kontrolera „P” są to wejścia 2,3,4,5 i źródło sygnału musi być NO. Jeżeli na danym wejściu zarejestrowano zmianę zbocza, to bit ten jest ustawiony.
70	Typ ruchu referencyjnego.
200	Numer aktualnego narzędzia
201	Typ aktualnego narzędzia
202	Tag aktualnego narzędzia

Dyrektywy kompilatora

Dyrektywy same w sobie nie są programem natomiast spełniają rolę pomocniczą. Zawsze rozpoczynają się znakiem „#”.

Składnia	Opis
#NAME_I Index = Name	Nadaje nazwę wejściu o podanym indeksie (Index 0 - 63). Nadawanie nazw ma dwojaką funkcję: • Nazwa jest wyświetlana w oknie kontrolek • Możemy odwoływać się w adresie do nazwy a nie numeru bitu, co jest zdecydowanie lepszym rozwiązaniem. Nadanie nazwy dwukrotnie jest traktowane jako błąd.
#NAME_I_N Index = Name	Jw. z tą różnicą, że wejście ma zanegowany stan aktywny – co widoczne jest także w kontrolkach.

#DELAY_I Index = Delay	Deklaracja opóźnienia dla wejścia o numerze Index (0-31) wyrażona w ms. Maksymalnie można podać 250ms. Aby wejście zmieniło stan logiczny, stan na wejściu będzie musiał trwać dłużej niż podane opóźnienie. Możemy podawać opóźnienie tylko dla pierwszych 32 wejść. Zamiast numeru wejścia możemy też podać jego nazwę (zalecane rozwiązanie).
	
	przykłady:
	`#DELAY_I 15 = 100 // Opóźnienie 100ms dla wejścia nr 15.`
	
	`#DELAY_I SENSOR_P = 200 // Opóźnienie 200ms dla wejścia o nazwie SENSOR_P.`

#NAME_O Index = Name	Nadaje nazwę wyjściu o podanym indeksie. (Index 0-31)

#NAME_M Index = Name	Nadaje nazwę bitowi w rejestrze M. (Index 0-31). Dyrektywa ma dwa dodatkowe warianty:
	`#NAME_M 10 = Name1,Name2,Name3 // Przypisze bitowi nr 10 nazwę Name1, 11-Name2, 12-Name3`
	
	`#NAME_M Name1,Name2,Name3 // Jeśli nie podamy indeksu, to nazwy zostaną automatycznie przypisane do pierwszych wolnych bitów w zakresie 0-31. Priorytet mają bity przypisywane statycznie`

#NAME_ML Name1, Name2, ...	J.w. lecz automatycznie przypisze nazwę do pierwszego wolnego bitu z zakresu 0-15. Działa tylko w trybie automatycznej alokacji wejść. Brak wolnych bitów w zakresie spowoduje błąd.
#NAME_MH Name1, Name2, ...	J.w. lecz automatycznie przypisze nazwę do pierwszego wolnego bitu z zakresu 16-31. Działa tylko w trybie automatycznej alokacji wejść. Brak wolnych bitów w zakresie spowoduje błąd.
#PLC_V2 [v2]	Przełącza tryb pracy kompilatora. W tym trybie dostępne są wszystkie komendy i polecenia oznaczone [v2]. Należy użyć tej dyrektywy przed pierwszym użyciem funkcji z tej wersji. Konieczne jest też zainstalowanie w kontrolerze FirmWare z wersją obsługującą PLC_V2 inaczej, przy próbie połączenia otrzymamy komunikat „PLC nie pracuje”!
#NAME_REG Index = Name [v2]	Nadaje nazwę rejestrowi „R” (Index 0-21). Dyrektywa ma dwa dodatkowe warianty:

Składnia	Opis

	`#NAME_REG 10 = Licznik1,Licznik2,Licznik3 // Przypisze rejestrowi nr 10 nazwę Licznik1, 11- Licznik1, 12- Licznik1`
	
	`#NAME_REG Licznik1,Licznik2,Licznik3 // Jeśli nie podamy indeksu, to nazwy zostaną automatycznie przypisane do pierwszych wolnych rejestrów. Priorytet mają rejestry przypisywane statycznie.`

#DEF Name = value [v2]	Definiuje ciąg tekstowy Name, który w czasie kompilacje będzie podmieniony na „value”.
	
	przykład:
	`#DEF Mask = 0b110100`
	`mov. R5 Mask`

#SET_OPTION Index = Value	Nadanie wartości rejestrowi opcji kompilacji o danym indeksie.
#IF_OPTION Index = Value	Sprawdza czy rejestr opcji zawiera daną wartość. Jeśli nie zawiera podanej wartości kolejne instrukcje aż do #END_IF nie będą kompilowane – czyli nie znajdą się w programie wysłanym do kontrolera. Także inne dyrektywy znajdujące się w obrębie #IF.. #END nie będą uwzględniane.
#END_OPTION	Kończy #IF_OPTION
#TXT_MSG Index = Text	Ustawienie tekstu powiadomienia. Patrz rozdział: Komunikaty ekranowe.
#TXT_ESTOP Index = Text	Ustawienie tekstu alarmu jaki pojawi się na ekranie po wystąpieniu E_STOP o numerze równym Index.
#SET_M Index = Value	Nadanie wartości początkową bitu o danym indeksie w rejestrze M. Jest to wartość wyjściowa po uruchomieniu PLC.
#SET_TIMER Index = Time	Ustawienie w sekundach czasu timera o danym indeksie. (Index 0-7), (Time 0.001 – 65 sek)

#MACRO_NAME Index = Macro	Ustawienie makra jakie zostanie wykonane po aktywowaniu w rejestrze MEMO lub IN bitu o nazwie C_MACRO_% - gdzie % to numer indeksu. Indeks może przyjmować wartości w zakresie 0-4.
	
	np.
	
	Nadajemy nazwę makra wraz z argumentami
	
	`#MACRO_NAME 0 = G0 X0 Y0`
	`#MACRO_NAME 1 = G1 X100 F1000`
	
	Następnie bitowi w rejestrze IN lub MEMO nadajemy nazwę:
	
	`#NAME_I 8 = C_MACRO_0`
	`#NAME_M 14 = C_MACRO_1`

#DIRECT_OUT Index	Deklarujemy wyjście, które będzie pracować w sposób bezpośredni tzn. jeżeli w rejestrze MEMO ustawimy bit o podanym indeksie, to wyjście o tym indeksie automatycznie przyjmie stan bitu z MEMO.
	
	Przykładowo, deklaracja jak niżej staje się zbędna, choć obsługa OUT 12 będzie dokładnie tak wyglądała - tylko w sposób sprzętowy.
	
	`<< M 12`
	`>> OUT 12`

Komentarze

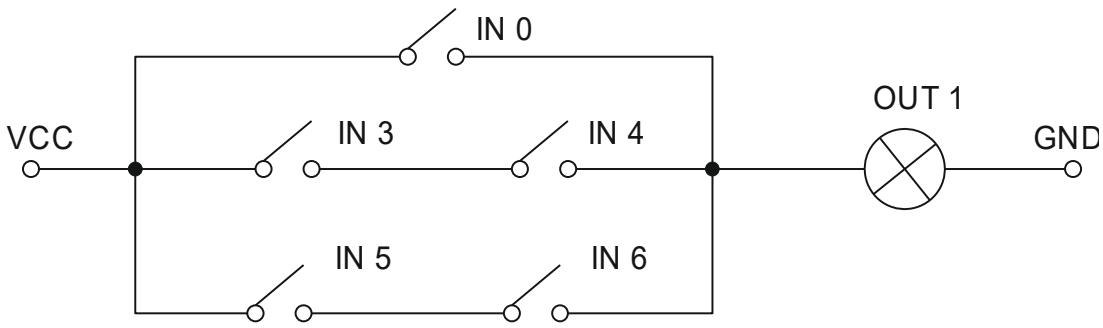
Możemy stosować dwa rodzaje komentarzy dla pojedynczej linii „//” lub dla większej ilości linii nawiasy klamrowe: „{„ oraz „}”. Znaki klamry muszą być jako pierwsze znaki w linii nie licząc spacji i tabulacji. Natomiast komentarz typu „//” może być w dowolnym miejscu linii tekstu.

Tryb drabinkowy

W trybie drabinkowym (tzn. między słowami kluczowymi PAR..END lub SER..END) operacje zapisu do bufora linii funkcjonują nieco inaczej – stan aktywny „1” bitu źródłowego oznacza klucz załączony a „0” wyłączony. Słowo kluczowe „BEG” Ustawia BL na „1” oraz zeruje indeks zagnieżdżenia drabinki. W trybie drabinkowym nie można stosować warunków IF.. ELSE.. ENDIF.

```
BEG
PAR
<< IN 0
   SER
      << IN 3
      << IN 4
   END
   SER
      << IN 5
      << IN 6
   END
END
>> OUT 1
```

Powyższy zapis odpowiada schematowi jak na rysunku.



Rejestry

Skróty: R - read, W - write.

Rodzaj	Typ	Rejestry
Fizyczne wejścia	R	IN
Fizyczne wyjścia	W	OUT
Logiczne wejścia	R	CONTROL1, CONTROL2, 1/2 TIMER
Logiczne wyjścia	W	STATUS, 1/2 TIMER
Pamięć	R/W	MEMO, R0 - R21

Jednym z zadań programu w sterowniku PLC jest połączyć te elementy które chcemy wykorzystać np. wejścia fizyczne IN z wejściami logicznymi HOME, LIMIT etc. Kontroler „sam z siebie” nie obsługuje żadnych I/O nawet wyjście „enable” czy „e-stop” musimy oprogramować w PLC.

Rejestry zależnie od rodzaju są typu R lub W. Jednak nie oznacza to że nie można zapisać rejestru typu R lub odczytać rejestru typu W – należy tylko mieć świadomość, że przy rozpoczęciu kolejnej pętli programu rejestry R są inicjowane danymi wartościami np. stanem wejść fizycznych, natomiast rejestry typu W są zerowane. Jedynie rejestry MEMO oraz R0-R21 pamiętają swoją zawartość.

Rejestr MEMO

Rejestr ten stanowi podręczną pamięć PLC. Może go zapisywać zarówno program sterujący jak i PLC w kontrolerze. Do bitów rejestru program sterujący może się odwoływać przez ich numer lub nazwę nadaną dyrektywą #NAME_M. Pewne nazwy są zarezerwowane do z góry ustalonych celów:

Komendy zdefiniowane na stałe:

Nazwa	Opis
C_M3	Bit ustawiany gdy w programie wystąpi komenda M3, a gaszony kiedy wystąpi M5
C_M4	Bit ustawiany gdy w programie wystąpi komenda M4, a gaszony kiedy wystąpi M5
C_M7	Bit ustawiany gdy w programie wystąpi komenda M7, a gaszony kiedy wystąpi M9
C_M8	Bit ustawiany gdy w programie wystąpi komenda M8, a gaszony kiedy wystąpi M9
C_M10	Bit ustawiany gdy w programie wystąpi komenda M10, a gaszony kiedy wystąpi M11

Komendy definiowane przez użytkownika:

Nazwa	Opis
C_KEY_%	Gdzie % to liczba w zakresie 0-9. Bit ustawiany kiedy zostanie naciśnięty odpowiedni klawisz z zakresu 0 do 9 na klawiaturze numerycznej. Bit jest gaszony kiedy puścimy. klawisz.

Nazwa	Opis

C_KEY_S%
BC_KEY_S%
	Bit ustawiany kiedy naciśnięty jest klawisz o kodzie %
	
	np.
	
	`#NAME_M 12 = C_KEY_S72  // klawisz "H"`
	`#NAME_M 13 = C_KEY_SS50  // klawisz "2" + SHIFT`
	
	Aby uzyskać kod klawisza w menu podręcznym okna edytora mamy pozycję „Pobierz do schowka kod klawisza”. Kiedy okienko się otworzy naciskamy dany klawisz, zamykamy okno a następnie przez „CTRL+V” wklejamy kod klawisza.
	
	Poprzedzenie nazwy literką „B” sprawia, że oryginalna funkcja przypisana klawiszowi w programie nie zostanie wykonana.
	
	np.
	
	`#NAME_M 12 = BC_KEY_S32  // klawisz spacja, funkcja pauzy na klawiszu spacji przestanie działać`

C_KEY_SS% BC_KEY_SS%	Jw. ale razem z klawiszem SHIFT.
C_KEY_SC% BC_KEY_SC%	Jw. ale razem z klawiszem CTRL.
C_KEY_SHIFT	Bit ustawiany kiedy zostanie naciśnięty klawisz SHIFT
C_KEY_CTRL	Bit ustawiany kiedy zostanie naciśnięty klawisz CTRL
MSG_TXT_%	Kiedy bit o tej nazwie jest aktywny na ekranie wyświetlany jest komunikat informacyjny o numerze % zdefiniowany dyrektywą #TXT_MSG.
C_MACRO_%	Kiedy bit o tej nazwie zostanie aktywowany wykonane zostanie makro o numerze % zdefiniowane dyrektywą #MACRO_NAME.

Jak widać z pierwszej tabeli program sterujący po wystąpieniu w g-kodzie kodu M3 nie wysteruje bezpośrednio jakiegoś wyjścia, ale zapala flagę C_M3 w rejestrze MEMO. To, co z tym zostanie zrobione dalej zależy wyłącznie od programu w PLC. W najprostszym wariancie może być np. :

```
<< M C_M3
>> OUT 1
```

Czyli bezpośrednie przekierowanie na OUT 1.

Stan początkowy rejestru możemy zmieniać dyrektywą #SET_M ustawiającą dane bity w momencie nawiązania połączenia. W monecie wystąpienia E_STOP bity 0-15 są automatycznie zerowane natomiast pozostałe zachowują wartość. Jeśli chcemy wyzerować górne 16 bitów (16-31) musimy ustawić bit CLR_MEM_H (rejestr CONTROL 2).

Rejestr TIMER

Kontroler posiada 8 timerów każdy może odmierzać czas od 0.001 do 65 sekund. Każdy timer ma wyjście stanu T# oraz dwa wejścia sterujące: ustawiające T#_S oraz kasujące T#_R. Ustawienie „1” na wejściu T#_S powoduje, że do licznika timera jest przepisywana wartość czasu która jest przypisana timerowi. Załóżmy, że przypisano timerowi czas 1s zatem do licznika jest wpisywana wartość 1000, która to, co 1/1000 sek. jest zmniejszana o 1. Wyjście T# stanu timera przyjmuje wartość „1” tak długo jak długo stan licznika timera jest większy od zera. Ustawienie „1” na wejściu T#_R zeruje licznik timera – czyli natychmiast przerywa jego pracę.
Wejścia T#_S oraz T#_R są automatycznie zerowane przed wykonaniem cyklu programu.

Przykład:

```
#SET_TIMER 0 = 1.0  // Ustawiamy czas timera numer zero  na 1 sek.
+< IN 0             // BL=1 jeśli narastające zbocze na IN 0
>> T0_S             // Jeśli BL=1 to startujemy timer 0
                    // Podpięcie wyjścia stanu timera do OUT 1
<< T0               // BL=T0
>> OUT 1            // OUT1 = BL
```

Zatem jeśli zamkniemy obwód na IN 0 na wyjściu OUT 1 pojawi się 1 sek. impuls. Jeśli zmodyfikujemy powyższy przykład i zamiast „+< IN 0” zrobimy „<< IN 0” uzyskamy timer z podtrzymaniem.

Rejestry R0-R21 [v2]

Pełnią one podobną rolę jak MEMO, z tym, że pamiętają nawet po rozłączeniu komunikacji z kontrolerem - tak długo jak jest on zasilany. Z rejestrów można korzystać zarówno z poziomu PLC jak i programu sterującego (PC). Do rejestrów można się odwoływać przez ich numer lub nazwę nadaną dyrektywą #NAME_REG.

Tablice rejestrów

Rejestr CONTROL 1

Bit	Nazwa	Typ	Opis
0	E_STOP_0	W	Wejścia zgłaszania E_STOP
1	E_STOP_1	W	
2	E_STOP_2	W	
3	E_STOP_3	W	
4	E_STOP_4	W	
5	RESET	W	Wejście przycisku RESET.
6	START	W	Wejście przycisku START.
7	STOP	W	Wejście przycisku STOP.
8	PAUSE	W	Wejście przycisku PAUZA.
9	REF	W	Wejście przycisku cyklu REF (jazda referencyjna)
10	MAT_REF	W	Wejście przycisku cyklu pomiaru wysokości materiału.
11	WH_PULSE	W	Wejście sygnału PULSE impulsatora (enkodera).
12	WH_DIR	W	Wejście sygnału DIR impulsatora.
13	WH_MD0	W	Wejścia sterujące trybem pracy impulsatora: 0 – nieaktywny (000) 1 - Zmiana F (001) 2 - Zmiana S (010) 3 - Zmiana położenia osi 0 (011) 4 - Zmiana położenia osi 1 (100) 5 - Zmiana położenia osi 2 (101) 6 - Zmiana położenia osi 3 (110) 7 - Zmiana położenia osi 4 (111)
14	WH_MD1	W	
15	WH_MD2	W	
16	JOG_L0	W	Wejście Jog w lewo
17	JOG_L1	W	
18	JOG_L2	W	
19	JOG_L3	W	
20	JOG_L4	W	
21	JOG_L5	W	
22	JOG_R0	W	Wejście Jog w prawo
23	JOG_R1	W	
24	JOG_R2	W	
25	JOG_R3	W	
26	JOG_R4	W	
27	JOG_R5	W	
28	JOG_FAST	W	Tryby szybki dla JOG i impulsatora
29	WH_JOG_L	W	Wejście Jog w lewo oraz w prawo. Numer osi wyznaczają bity trybu pracy impulsatora (WH_MD0 - WH_MD2)
30	WH_JOG_R	W	
31	WH_ZERO	W	Zerowanie parametru wyznaczonego przez bity trybu pracy

Bit	Nazwa	Typ	Opis
			impulsatora. Osie są zerowane, natomiast parametry F, S ustawiane na 100%.

Rejestr CONTROL 2

Bit	Nazwa	Typ	Opis
0	LIMIT_L0	W	Wejścia LIMT_L (lewy) dla osi.
1	LIMIT_L1	W	
2	LIMIT_L2	W	
3	LIMIT_L3	W	
4	LIMIT_L4	W	
5	LIMIT_L5	W	
6	LIMIT_R0	W	Wejścia LIMT_R (prawy) dla osi.
7	LIMIT_R1	W	
8	LIMIT_R2	W	
9	LIMIT_R3	W	
10	LIMIT_R4	W	
11	LIMIT_R5	W	
12	HOME_0	W	Wejścia HOME osi.
13	HOME_1	W	
14	HOME_2	W	
15	HOME_3	W	
16	HOME_4	W	
17	HOME_5	W	
18	INDEX_0	W	Wejścia indeks osi.
19	INDEX_1	W	
20	INDEX_2	W	
21	INDEX_3	W	
22	INDEX_4	W	
23	INDEX_5	W	
24	FIFO_WAIT	W	Wejście zatrzymujące wykonywanie komend z kolejki FIFO tak długo jak długo wejście jest w stanie aktywnym.
25	N_WAIT_G0	W	Jeżeli wejście jest w stanie aktywnym to FIFO_WAIT nie dotyczy komendy G0.
26	PROBE	W	Wejście wykorzystywane do pomiaru długości narzędzia, wysokości materiału.
27	CLR_MEM_H	W	Ustawienie bitu czyści (zeruje) górne 16 bitów rejestru MEM
28	TA_PROBE	W	Wejście HOME osi technicznej
29	TA_JOG_R	W	Jog osi technicznej (prawo)
30	TA_JOG_L	W	Jog osi technicznej (lewo)
31	PARK_0	W	Wejście przycisku jazdy do pozycji PARK_0

Rejestr STATE

Bit	Nazwa	Typ	Opis
0	IS_MOVE_0	R	Bity informują, że dana oś jest aktualnie w ruchu.
1	IS_MOVE_1	R	
2	IS_MOVE_2	R	
3	IS_MOVE_3	R	
4	IS_MOVE_4	R	
5	S_IS_STOP	R	Rampa „S” osiągnęła dolny poziom (PWM, 0-10V)
6	DIR_0	R	Bity informują o aktualnym kierunku jazdy osi.
7	DIR_1	R	
8	DIR_2	R	
9	DIR_3	R	
10	DIR_4	R	
11	S_AT_SPEED	R	Rampa „S” osiągnęła górny poziom (PWM, 0-10V)
12	CORNER	R	Sygnał informuje, że ruch wyhamowuje lub nie osiągnął jeszcze prędkości zadanej.
13	IS_JOG	R	Informuje, że jest wykonywany jog.
14	TA_DIR	R	Informuje o kierunku jazdy osi technicznej
15	TA_REF	R	Informuje, że oś techniczna wykonuje jazdę referencyjną.
16	TA_MOVE	R	Informuje, że oś techniczna jest w ruchu.
17	F_IS_100	R	Informuje, że procentowe wzmocnienie F jest 100%
18	IS_G1	R	Wykonywany jest ruch G1
19	IS_G0	R	Wykonywany jest ruch G0
20	REF_0	R	Jazda referencyjna do momenty wystąpienia stanu aktywnego na wejściu HOME osi
21	REF_1	R	Jazda referencyjna do momenty zaniku stanu aktywnego na wejściu HOME osi.
22	REF_2	R	Jazda referencyjna do momenty wystąpienia stanu aktywnego na wejściu INDX osi.
23	REF_3	R	Jazda referencyjna do momenty wystąpienia stanu aktywnego na wejściu PROBE
24	REF_4	R	Jazda referencyjna do momenty zaniku stanu aktywnego na wejściu PROBE.
25	IS_EXE	R	Informuje, że wykonywany jest program
26	IS_PAUSE	R	Program w stanie pauzy.
27	IS_AUTO	R	Ruch w trybie automatycznym.
28	RUN	R	Kontroler w stanie „RUN”
29	S_IS_100	R	Informuje, że procentowe wzmocnienie S jest 100%
30	E_STOP	R	Wystąpiło zatrzymanie przez E_STOP
31	MOVE_INIT	R	Bit ustawiany każdorazowo po zainicjowaniu ruchu G1. Kasowany automatycznie po wykonaniu pętli programu przez PLC

Rejestr TIMER

Bit	Nazwa	Typ	Opis
0	T0	R	Zawierają aktualny stan timera.
1	T1	R	
2	T2	R	
3	T3	R	
4	T4	R	
5	T5	R	
6	T6	R	
7	T7	R	
8	T0_S	W	Wejście ustawiające timer.
9	T1_S	W	
10	T2_S	W	
11	T3_S	W	
12	T4_S	W	
13	T5_S	W	
14	T6_S	W	
15	T7_S	W	
16	T0_R	W	Wejście kasujące timer.
17	T1_R	W	
18	T2_R	W	
19	T3_R	W	
20	T4_R	W	
21	T5_R	W	
22	T6_R	W	
23	T7_R	W	
24	PULSE_0	R	Przebieg o częstotliwości 50 Hz
25	PULSE_1	R	Przebieg o częstotliwości 25 Hz
26	PULSE_2	R	Przebieg o częstotliwości 12,5Hz
27	PULSE_3	R	Przebieg o częstotliwości 6,25 Hz
28	PULSE_4	R	Przebieg o częstotliwości 3,12 Hz
29	PULSE_5	R	Przebieg o częstotliwości 1,56 Hz
30	PULSE_6	R	Przebieg o częstotliwości 0,78 Hz
31	REMOTE_IN	R	Wejście modułu zdalnych wejść.

Rejestr MEMO

Bit	Nazwa	Typ	Opis
0-31	M	R/W	Rejestr pomocniczy – pamięć podręczna.

Rejestr IN

Bit	Nazwa	Typ	Opis
0-63	IN	R	Rejestr wejść fizycznych. Liczba rzeczywistych dostępnych bitów zależy typu kontrolera.

0-31	IN2	R	Rejestr jest górną częścią rejestru IN (górne 32 bity) np. bit 32 IN, to to samo co bit 0 IN2. Czasami trzeba użyć takiej formy ponieważ rozmiar operacji jest 32-bitowy, przykład:
			
			`MOV. R10 IN  // Załadowanie do R10 bitów 0-31 IN`
			`MOV. R11 IN2 // Załadowanie do R11 bitów 32-63 IN`

Rejestr OUT

Bit	Nazwa	Typ	Opis
0-31	OUT	W	Rejestr wyjść fizycznych. Liczba rzeczywistych dostępnych bitów zależy typu kontrolera.

Funkcje zapisu/odczytu rejestrów PLC z poziomu makr.

num – numer bitu (0-31), dla rejestru IN 0-63.
devname – nazwa bitu zdefiniowana w PLC, lub nazwa rejestru „R”.
st – docelowy stan bitu.
val – wartość wpisywana do rejestru

Rejestr	R/W	Funkcja / procedura
MEMO	W	`procedure SetMemo(num:cardinal; st:boolean)`
	W	`procedure SetMemoN(devname:string; st:boolean)`
	R	`function Memo(num:cardinal):boolean`
	R	`function MemoN(devname:string):boolean`
IN	R	`function Input(num:cardinal):boolean`
	R	`function InputN(devname:string):boolean`
OUT	R	`function Output(num:cardinal):boolean`
	R	`function OutputN(devname:string):boolean`
TIMER	W	`procedure SetTimer(num:cardinal; time:extended)` Procedura ustala czas timera gdzie: num – numer timera (0-7), time – czas timera w sekundach (0.001 – 65 sek.)
R0 – R21 [v2]	W	`procedure SetRegN(devname:string; val:integer)`
	W	`procedure SetReg(num:cardinal; val:integer)`
	R	`function GetRegN(devname:string):integer`
	R	`function GetReg(num:cardinal):integer`
Kontrolki wirtualne [v2]	R	`function GetVirtual(devname:string):boolean` Funkcja odczytuje stan kontrolki wirtualnej o podanej nazwie. Jeżeli kontrolka o takiej nazwie nie jest zadeklarowana w PLC, to funkcja zwróci wartość FALSE;

Kontrolki wirtualne [v2]

Kontrolki wirtualne, to specjalne nazwy bitów zdefiniowane w rejestrach: IN,OUT,MEMO. Kontrolki takie obsługiwane są z poziomu oprogramowania na komputerze. Aby używać takiej kontrolki wystarczy zadeklarować bit o danej nazwie w którymś z wymienionych rejestrów i w jakiś sposób go kontrolować. Nazwy bitów wirtualnych zarówno w tekście programu jaki i na tablicy kontrolek wyświetlane są w kolorze.

Przykład:

```
#NAME_I   9 = AUTO_DISABLE // aktywacja wejścia 9 będzie blokować wszystkie przyciski funkcji „auto” interfejsu START,REF etc.
```

Tabela kontrolek wirtualnych sterujących pracą maszyny.

Nazwa	Opis
JOG_F_LIMIT	Ograniczenie prędkości JOG do trybu wolnego (tak jak z klawiszem Ctrl)
TOOL_REF	Uruchomienie cyklu pomiaru narzędzia
MOV_REV	Jazda po konturze (ścieżce) do tyłu.
MOV_FF	Jazda po konturze (ścieżce) do przodu.
MOV_ZERO	W trybie pracy „Względem narzędzia” powrót do zera programu.
PROG_RST	W profilu plazmy odpowiada naciśnięciu przycisku „UPDATE”
WH_STEP_1	Wybór kroku nr 1 dla impulsatora (ustawiany w ustawieniach kontrolera / Impulsator zadajnika)
WH_STEP_2	Wybór kroku nr 2 dla impulsatora ...
WH_STEP_3	Wybór kroku nr 3 dla impulsatora ...
WH_SPEED	Wybór trybu „prędkość” dla impulsatora
WH_F_RAPID	Tryb zmiany / zerowania (F) dotyczy ruchów ustawczych
WH_F_ALL	Tryb zmiany / zerowania (F) dotyczy ruchów wszystkich typów (roboczych i ustawczych.

Tabela kontrolek wirtualnych sterujących dostępnością funkcji.

Nr bitu	Nazwa	Opis
0	AUTO_DISABLE	Blokownie wszystkich przycisków mogących uruchomić ruch automatyczny (START, REF, Pomiary etc.)
1	M3_DISABLE	Blokada przycisków M3/M4 (wrzeciono)
2		Rezerwacja. Zostawić wyzerowane.
3	M7_DISABLE	Blokada przycisków M7 (woda)
4	M8_DISABLE	Blokada przycisków M8 (mgła)
5	F_PRC_DISABLE	Blokownie zmiany procentowej dla przejazdów roboczych
6	FR_PRC_DISABLE	Blokownie zmiany procentowej dla przejazdów ustawczych
7	S_PRC_DISABLE	Blokownie zmiany procentowej dla „S”
8	ATC_DISABLE	Blokownie przycisku ATC powodującego automatyczną zmianę narzędzia, oraz przycisku odłożenia narzędzia do magazynka. Przyciski ręcznej wymiany zostają aktywne. Należy także oprogramować tą kontrolkę w makrze M6, aby blokada działa również z poziomu MDI.
9	AUX0_DISABLE	Blokownie przycisku AUX0
10	AUX1_DISABLE	Blokownie przycisku AUX1
11	AUX2_DISABLE	Blokownie przycisku AUX2
12	AUX3_DISABLE	Blokownie przycisku AUX3
13	AUX4_DISABLE	Blokownie przycisku AUX4
14	START_DISABLE	Blokownie przycisku START (również z klawiatury czy PLC)
15	STOP_DISABLE	Blokownie przycisku STOP
16	PAUSE_DISABLE	Blokownie przycisku PAUZA
17	ZERO_DISABLE	Blokownie przycisków ZERO przy licznikach osi.
18	REF_DISABLE	Blokownie przycisku REF
19	TOOL_M_DISABLE	Blokowanie przycisku pomiaru narzędzia
20	MAT_M_DISABLE	Blokowanie przycisku pomiaru materiału
21	JOG_DISABLE	Blokowanie całkowite JOG
22-31		Rezerwacja. Zostawić wyzerowane.

Sterować dostępnością funkcji możemy na dwa sposoby: albo deklarując nazwy jak wyżej opisano, albo deklarując rejestr o nazwie DISABLE_REG i sterując jego bitami. Kolumna nr. bitu w tabeli informuje, który bit należy ustawić aby zablokować dane funkcje. Bitów opisanych jako „rezerwacja” nie należy ustawiać, ponieważ nie wiadomo jaką funkcję będą pełnić w przyszłości. Metoda z rejestrem jest bardziej bardziej ekonomiczna – pozwala jedną instrukcją ustawić wiele kontrolek.

Przykład:

```
#NAME_REG 1 = DISABLE_REG
mov. DISABLE_REG %11000 // ustawienie M7_DISABLE oraz M8_DISABLE
```

Komunikaty ekranowe.

Za pomocą dyrektywy **#TXT_MSG** możemy zdefiniować do 32 komunikatów ekranowych. Są dwa sposoby aby kontrolować widoczność komunikatu na ekranie. Pierwszy, to za pomocą bitu w MEMO o nazwie MSG_TXT_% - gdzie % to numer komunikatu w zakresie 0-4.

Przykład:

```
#NAME_M 17 = MSG_TXT_0
#TXT_MSG 0 = Lewy silnik trochę się grzeje! // Komunikat

// Wyświetlenie komunikatu gdy aktywne wejście nr.0

<< IN 0
>> M MSG_TXT_0
```

Drugi sposób, to za pomocą rejestru o nazwie **TXT_MSG_REG** gdzie zapalenie bitu o danym numerze powoduje wyświetlanie komunikatu o tym numerze. Zatem, jednocześnie (teoretycznie) można wyświetlać do 32 komunikatów, a nie tyko pięć jak w poprzedniej metodzie.

```
#NAME_REG 5 = TXT_MSG_REG
#TXT_MSG 0 = Lewy silnik trochę się grzeje! // Komunikat

// Wyświetlenie komunikatu gdy aktywne wejście nr.0

<< IN 0
>> TXT_MSG_REG 0
```

Formatowanie komunikatów

Począwszy od wersji 5.8 programu, komunikaty ekranowe mogą zawierać kody formatujące zaczynające się znakiem %.

Kody zawiązane z rejestrami.

Kod	Opis	Przykład
%R#	Liczbowa wartość rejestru gdzie # to numer rejestru (0-21)	%R15
%R#.$	Stan bitu nr. $ (0-31) w rejestrze # wyświetlany jako ON/OFF	%R15.7
%R#.$+100	stan bitu nr. $ w rejestrze # wyświetlany jako 1/0	%R15.107
%R#.$+200	stan bitu nr. $ w rejestrze # wyświetlany jako 1/-1	%R15.207

Kody zawiązane z wejściami.

Kod	Opis	Przykład
%I$	stan wejścia nr $ (0-63) wyświetlany jako ON/OFF	%I15
%I$+100	stan wejścia nr $ wyświetlany jako 1/0	%I115
%I$+200	stan wejścia nr $ wyświetlany jako 1/-1	%I215

Kody zawiązane z rejestrem MEMO.

Kod	Opis	Przykład
%M$	stan bitu MEMO nr $ wyświetlany jako ON/OFF (0-31)	%M15
%M$+100	stan bitu MEMO nr 1 wyświetlany jako 1/0	%M115
%M$+200	stan bitu MEMO nr 1 wyświetlany jako 1/-1	%M215

Kody związane ze zmiennymi makr.

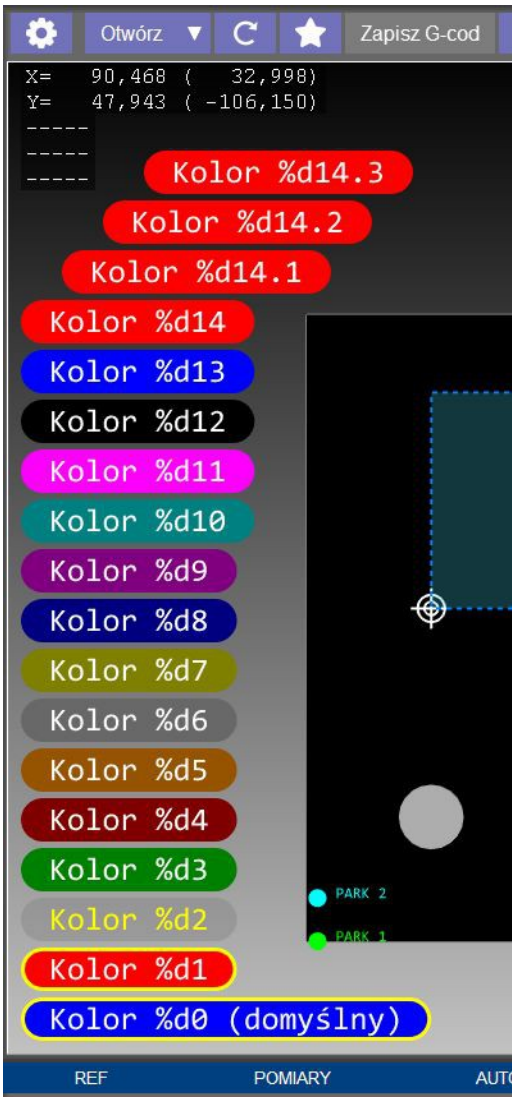
Kod	Opis	Przykład
%V#	Wartość zmiennej nr # rejestrów makr. (SetVar/GetVar) Wartość wyświetlana zgodnie z typem danej	%V10

Bloki obliczeń.

Kod	Opis	Przykłady
%[...]	Blok obliczeń np: %[2*10] wyświetli liczbę 20. W bloku obliczeń możemy stosować nawiasy, rejestry czy wartości zwracające liczbę. Jeżeli w formule obliczeniowej jest błąd, to zamiast wyniku wyświetlany będzie napis #CALC_ERR.	%[2.1 * %R10 + (R11 / 2.5)] %[%R10 * %R11.201] %[%V100 * %R10 + 1.2]

Kody związane z wyglądem komunikatów.

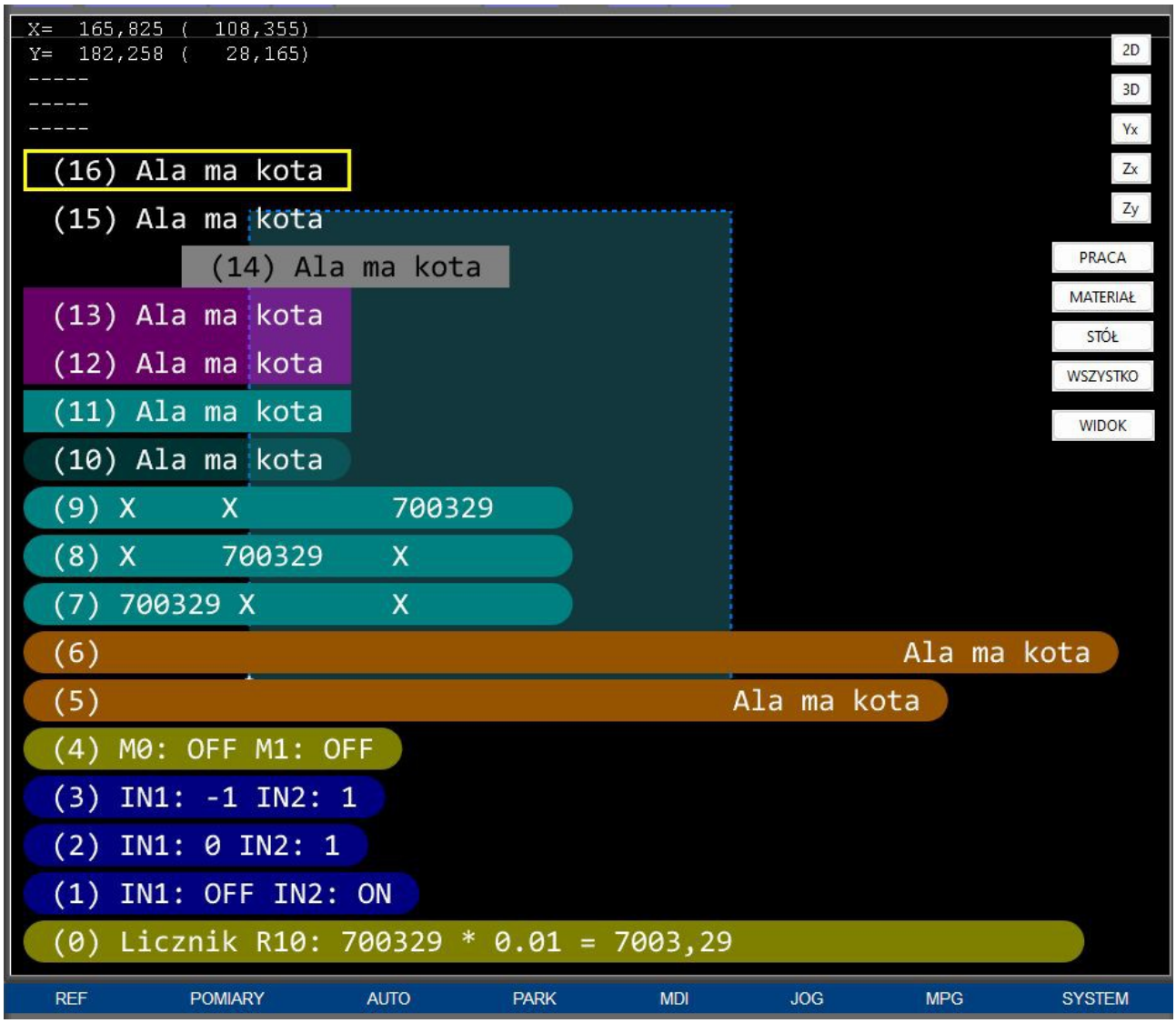
Kod	Opis	Przykłady
%T#	Przeniesie karetkę do pozycji o numerze # w linii. Wolne miejsca będą wypełnione spacją. Kod umożliwia justowanie danych i organizowanie ich w kolumny.	%T40%[R10 * 1.23]
%F#.#	Flagi decydujące o sposobie wyświetlania komunikatu. Jednorazowo możemy podać dwie flagi. 0 - komunikat z ramką. 1 - komunikat pulsujący. 2 - brak tła, sam napis. 3 - przeźroczyste tło. 4 - kwadratowe narożniki. 5 - powiększone obszary tła. Komunikaty wyświetlane obok siebie tworzą jeden obszar. Uwaga! Dla pierwszych trzech kodów (%D0-%D2) flagi nie działają!	%F0.1 – komunikat będzie miał ramkę i będzie pulsował. %F0.1%F4 – jw. plus kwadratowe narożniki

Kod	Opis	Przykłady
%D#.$	# - Numer koloru komunikatu (0 - 14). $ - Wcięcie komunikatu. Pierwsze trzy numery koloru definiują trzy systemowe komunikaty: 0 - niebieski informacyjny (pulsujący) 1 - czerwony pulsujący (jak E-STOP) 2 - przeźroczysty jak informacja o aktualnym narzędziu. Kod %D musi być jako pierwszy w linii ! Kod %D1 wyświetlany jest tylko wtedy, gdy wyświetlany jest komunikat E_STOP i wyświetlany jest pod nim. 	%D7%F1.2

Przykład 1:

```
#PLC_V2

#NAME_REG 0 = TXT_MSG_REG
#TXT_MSG 0=%d7 (0) Licznik R10: %R10 * 0.01 = %[%R10 * 0.01] %T60
#TXT_MSG 1 = %d8 (1) IN1: %I1 IN2: %I2
#TXT_MSG 2 = %d8 (2) IN1: %I101 IN2: %I102
#TXT_MSG 3 = %d8 (3) IN1: %I201 IN2: %I202
#TXT_MSG 4 = %d7 (4) M0: %M0 M1: %M1
#TXT_MSG 5 = %d5 (5) %T40 Ala ma kota
#TXT_MSG 6 = %d5 (6) %T50 Ala ma kota
#TXT_MSG 7 = %d10 (7) %R10%T10 X%T20 X%T30
#TXT_MSG 8 = %d10 (8) X%T10 %R10%T20 X%T30
#TXT_MSG 9 = %d10 (9) X%T10 X%T20 %R10%T30
#TXT_MSG 10 = %d10 %F3 (10) Ala ma kota
#TXT_MSG 11 = %d10 %F4 (11) Ala ma kota
#TXT_MSG 12 = %d11 %F5.4%F3 (12) Ala ma kota
#TXT_MSG 13 = %d11 %F5.4%F3 (13) Ala ma kota
#TXT_MSG 14 = %d12.4 %F1.4 (14) Ala ma kota
#TXT_MSG 15 = %d12 %F2.4 (15) Ala ma kota
#TXT_MSG 16 = %d12 %F0.4 (16) Ala ma kota

+< RUN
IF.TRUE
mov. R10 0 // Zerowanie R10 przy RUN


add. R10 1 // Zwiększamy R10 w każdym obiegu pętli
mov. TXT_MSG_REG 0xFFFFF // Załączenie wszystkich komunikatów
```

Rezultat działania powyższego programu. Paski dynamicznie wyświetlają zmieniającą się zawartość rejestru R10. Paski 1,2,3 pokazują aktualny stan wejść. Komunikaty wyświetlane są w kolejności rosnącego indeksu - licząc od dołu.



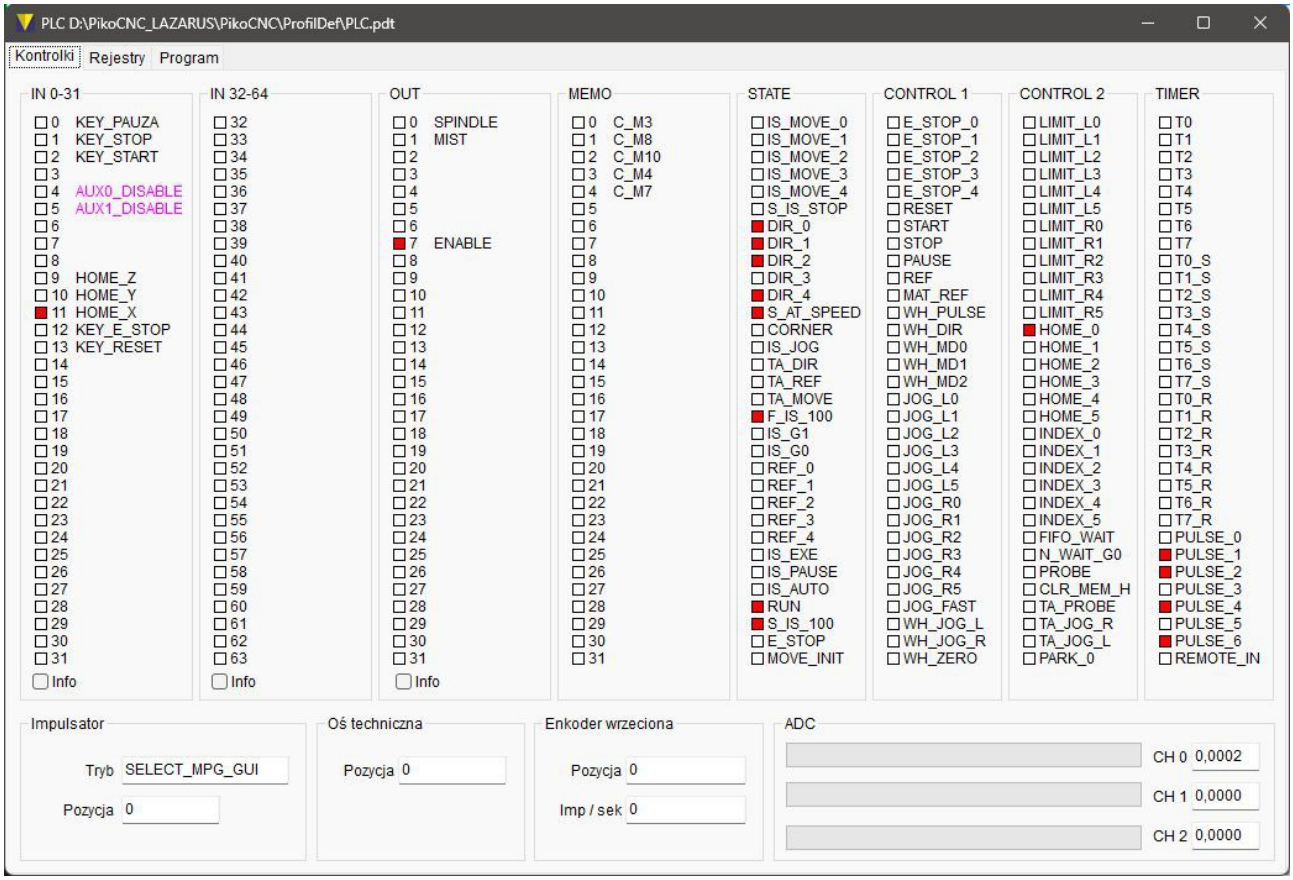
Przykład 2:

Wykorzystanie jednego kanału E_STOP do sygnalizacji błędu 5 napędów osi.

```
#PLC_V2
#NAME_I 6 = SER_X // Deklaracja wejść błędów
#NAME_I 7 = SER_Y
#NAME_I 8 = SER_Z
#NAME_I 16 = SER_A
#NAME_I 3 = SER_B

#NAME_REG TXT_MSG_REG
#TXT_ESTOP 1 = Napęd >>
#TXT_MSG 0 = %d1.2 >> Błąd serwonapędu osi X
#TXT_MSG 1 = %d1.2 >> Błąd serwonapędu osi Y
#TXT_MSG 2 = %d1.2 >> Błąd serwonapędu osi Z
#TXT_MSG 3 = %d1.2 >> Błąd serwonapędu osi A
#TXT_MSG 4 = %d1.2 >> Błąd serwonapędu osi B
#DEF maska = 0b11111 // Używamy pierwsze pięć bitów

<<_FIRST_RUN
MCR. TXT_MSG_REG maska // Czyszczenie bitów przy pierwszym
uruchomieniu

+< RUN
MCR. TXT_MSG_REG maska // Czyszczenie bitów przy załączeniu RUN

<< RUN
IF_TRUE_BEGIN
  << IN SER_X
  S> TXT_MSG_REG 0
  S> E_STOP_1
  << IN SER_Y
  S> TXT_MSG_REG 1
  S> E_STOP_1
  << IN SER_Z
  S> TXT_MSG_REG 2
  S> E_STOP_1
  << IN SER_A
  S> TXT_MSG_REG 3
  S> E_STOP_1
  << IN SER_B
  S> TXT_MSG_REG 4
  S> E_STOP_1
ENDIF
```

Okno PLC.

Zakładka „Kontrolki”



W zakładce możemy obserwować stany wszystkich wejść, wyjść oraz pozostałych rejestrów. Nazwy bitów wirtualnych rysowane są kolorem. Po kliknięciu w ptaszek „Info” widzimy dodatkowe informacje np. czy wyjście jest NO/NC, zastosowane opóźnienia dla wejścia. Dla wyjść widzimy czy wyjście jest w trybie „DIRECT”.

Na dolnych panelach:

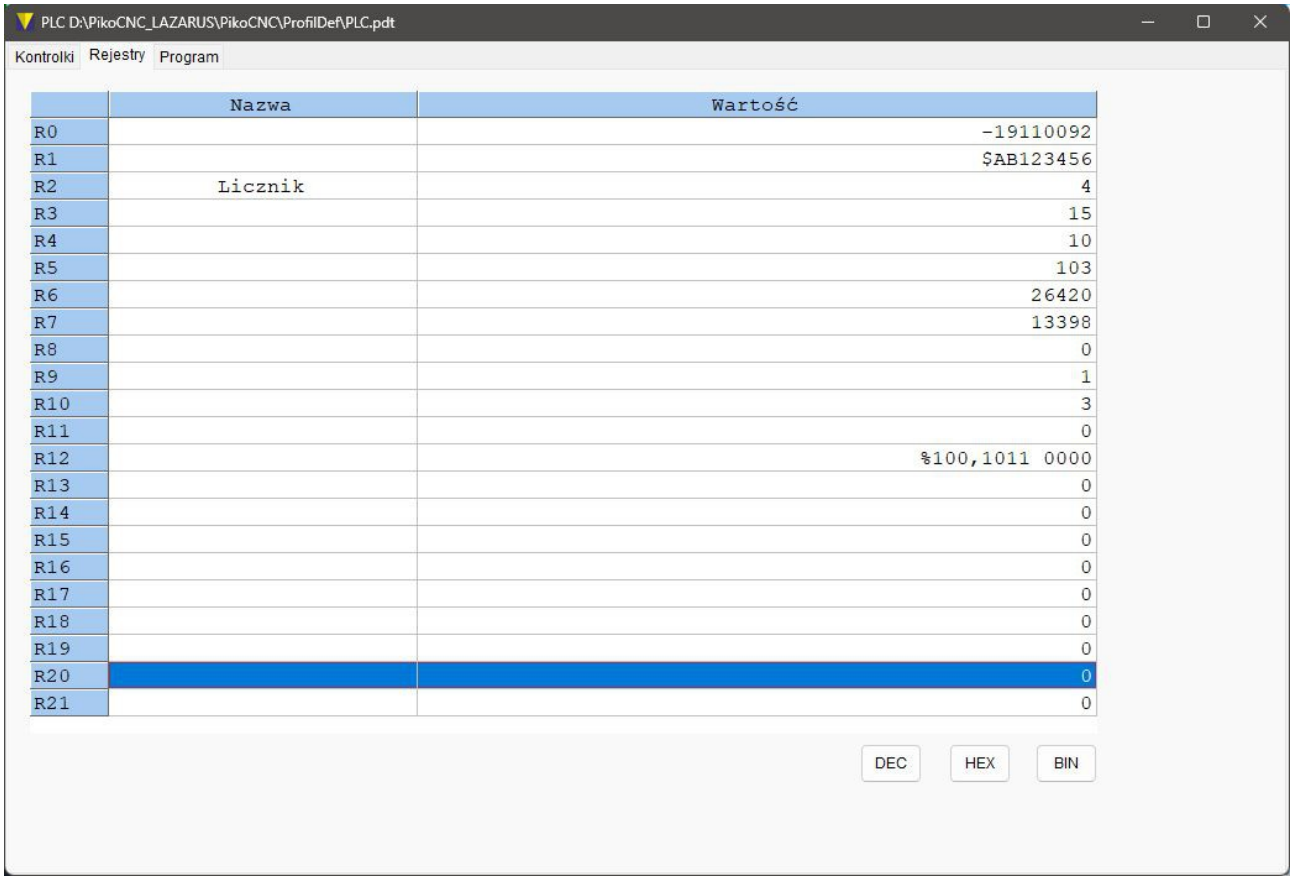
Impulsator – informacje o stanie impulsatora MPG opartego o wejścia i PLC.

Oś techniczna – pozycja osi technicznej.

Enkoder wrzeciona – pozycja i prędkość obrotowa enkodera wrzeciona.

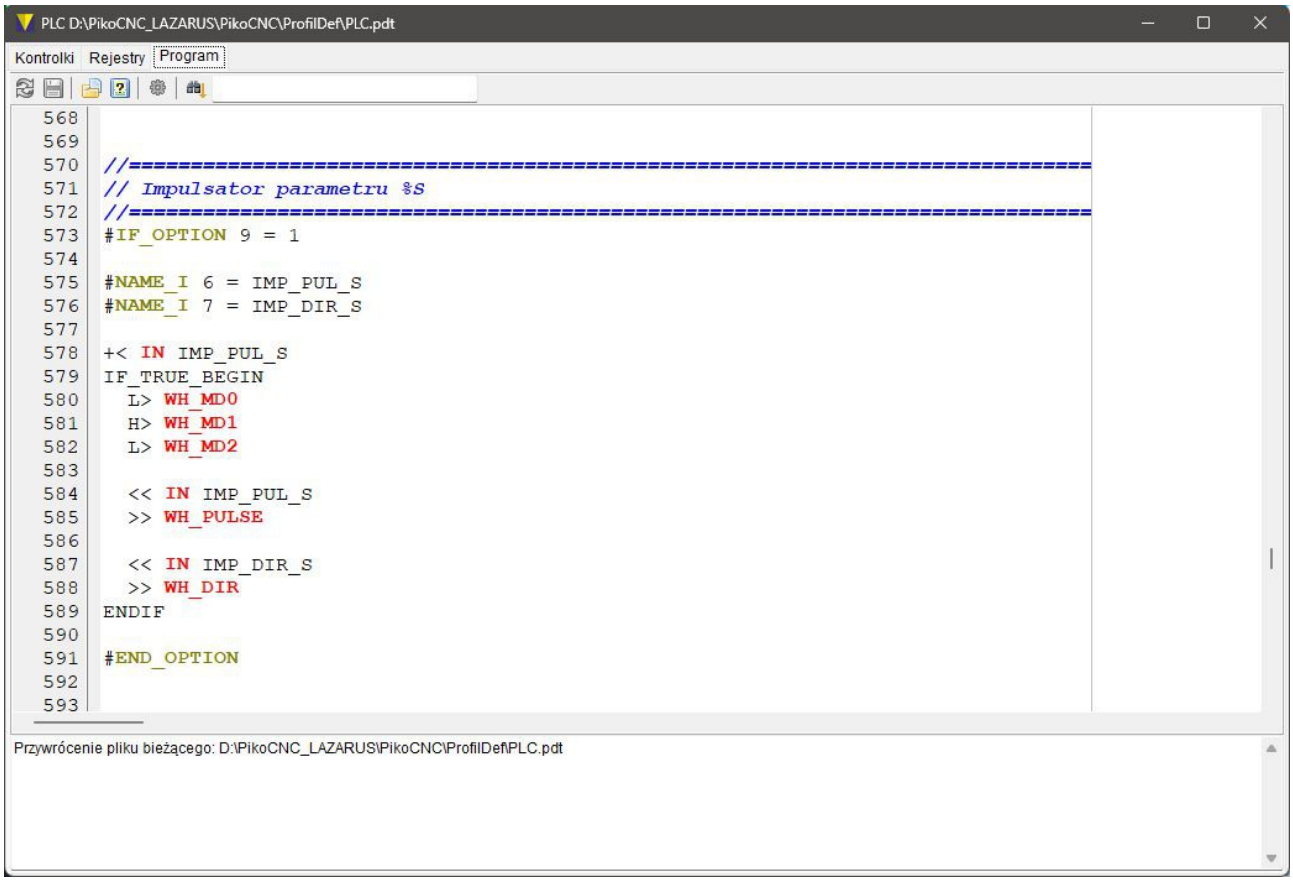
ADC – wartości zmierzone przez ADC

Zakładka „Rejestry” [v2]

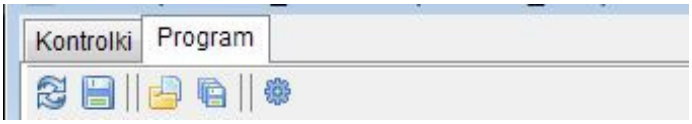


Zakładka ta jest widoczna tylko wówczas, gdy w naszym programie PLC użyjemy dyrektywy #PLC_V2 przełączającą kompilator na nowszą wersje. Możemy monitorować stan wszystkich 22 rejestrów „R”. Przyciskami DEC / HEX / BIN wybieramy tryb wyświetlania wartości dla podświetlonego rejestru.

Zakładka „Program”



Jest to zakładka zawierająca nasz program PLC. Klikają na oknie RMB mamy menu z narzędziami ułatwiającymi nawigację i edycję programu. Pasek ikon:



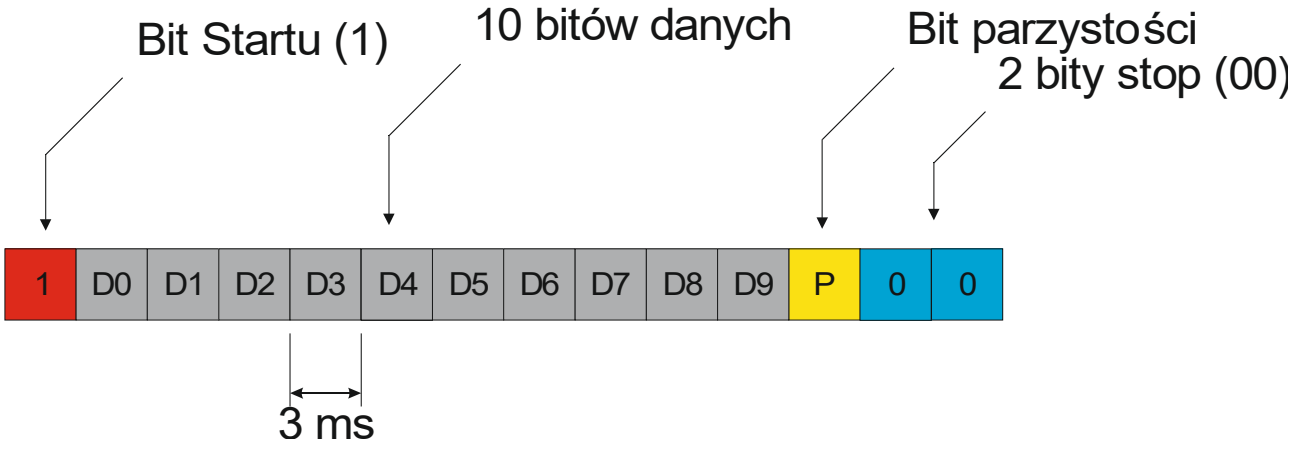
1 2 3 4 5

Ikona	Opis	Funkcja
1	Przywróć bieżący	Otwiera w edytorze plik bieżący czyli plik o nawie PLC.pdt z katalogu aktualnego profilu - zazwyczaj jest to katalog ProfilDef.
2	Zapisz bieżący	Zapisuje program jako plik PLC.pdt w katalogu aktualnego profilu. Program przed zapisem jest zawsze kompilowany dlatego nie da się zapisać programu który ma np. błędy składniowe. Program możemy zapisać tylko wówczas, gdy rozłączona jest komunikacja z kontrolerem!
3	Otwórz dowolny	Otwiera plik z programem z dowolnej lokalizacji
4	Zapisz jako	Zapisuje plik w dowolnej lokalizacji.
5	Kompiluj	Kompilacja programu. Umożliwia na bieżąco sprawdzanie poprawności programu.

Moduł zdalnych wejść.

Z myślą o zwiększeniu liczby wejść na takie potrzeby jak JOG czy przyciski sterujące - PLC obsługuje prosty protokół komunikacyjny na wejściu REMOTE_IN (rejestr TIMER).

Budowa ramki danych.



Ramka składa się z jednego bitu startu (=1), 10-ciu bitów danych, bitu parzystości oraz dwóch bitów stopu (=00). Bit parzystości ustawiany jest gdy w polu danych (D0-D9) ustawiona jest nieparzysta liczba bitów. Ramki muszą być nadawane non-stop jedna za drugą, ale co 10 ramkę należy zrobić przerwę w nadawaniu na czas trwania jednej ramki (42 ms). Przerwa ma na celu dać gwarancję synchronizacji transmisji. Przerwa w nadawaniu powyżej 70 ms jest traktowana jako zakończenie transmisji i zerowane są wtedy bity ramki. Czas trwania jednego bitu to 3 ms.

Obsługa na poziomie PLC.

Aby odbierać dane od modułu REMOTE musimy najpierw podłączyć wejście do którego podpięty jest moduł do wejścia REMOTE_IN (rejestr TIMER)

```
<< IN 5       // Moduł zdalnych wejść podpięty pod wejście 5
>> REMOTE_IN
```

Zależnie od typu kontrolera odebrane dane znajdują się w różnych miejscach. Należy tych informacji szukać w dokumentacji płyt (np. w płytach typu A, B widoczne są w rejestrze IN jako bity 22-31).

Aktualnie [V2] Standardowym sposobem jest odczyt za pomocą komendy STAT.

```
STAT. R10 60 // Załadownie odebranej ramki do R10
```

PPHU ELCOSIMO
Andrzej Woźniak
ul. Zielona 1B
62-110 Damasławek