Uniwersytet Morski w Gdyni

Katedra Podstaw Techniki

Laboratorium Automatyki Cyfrowej

Instrukcja do ćwiczenia

Ćwiczenie nr	1
Temat :	**Programowanie cyfrowe Symulator**
Stanowisko laboratoryjne	Symulator PLC
Opracował :	A. Mielewczyk

UNIWERSYTET MORSKI
Gdynia 2022

Instrukcja nr.1

1. Temat ćwiczenia:

Podstawy programowania w języku drabinkowym LAD.

2. Cel ćwiczenia:

Celem ćwiczenia jest zapoznanie się z podstawowymi elementami w programowaniu drabinkowym sterownika PLC i wykonanie symulacji załączania wyjść sterownika.

3. Zakres wymaganych wiadomości:

- sygnały cyfrowe i ich adresowanie,
- sygnały analogowe i ich adresowanie,
- elementy czasowe (timery),
- elementy zliczające i porównujące,

4. Przebieg ćwiczenia:

Zapisać program załączania wyjść cyfrowych i analogowych według wskazanych przykładów.

5. Pomoce i urządzenia:

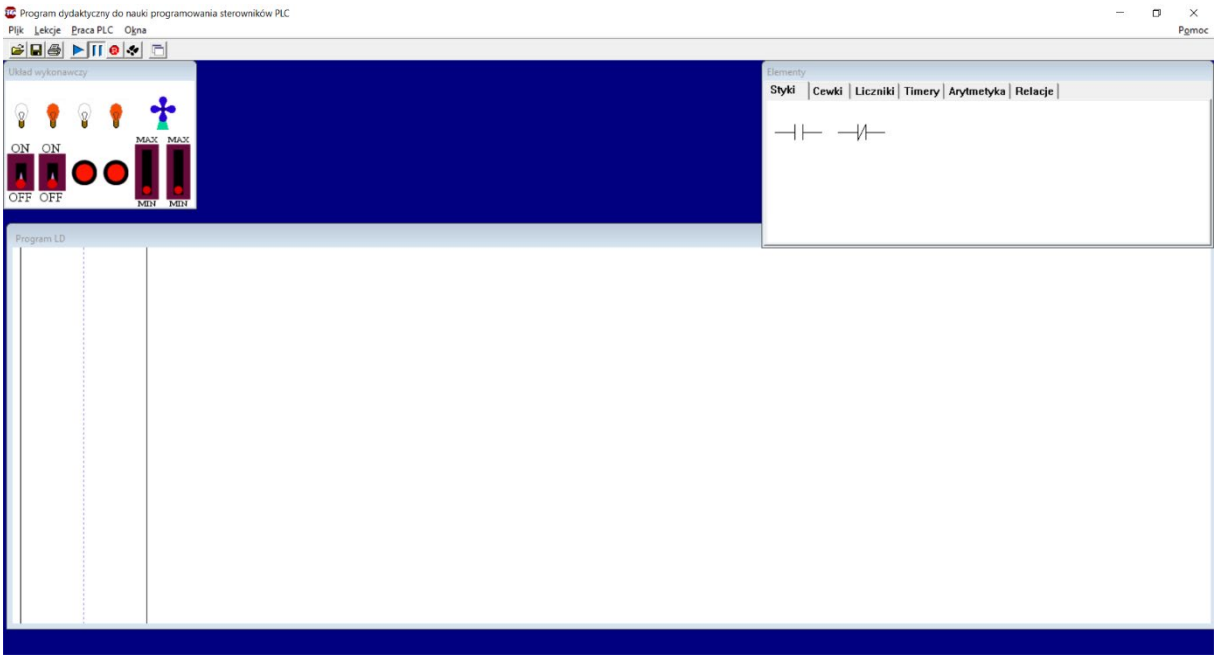
Symulator komputerowy sterownika PLC.

6. Treść sprawozdania:

Część wstępna, opis zadania, realizacja w języku drabinkowym.

WPROWADZENIE

Prosty symulator do programowania w języku drabinkowym przedstawiono na rys. 1.

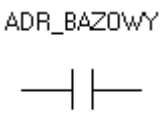


Rys. 1 Symulator komputerowy sterownika PLC

Podstawowe elementy programowania sterowników PLC w języku drabinkowym.

Styk zwierny

Symbol rozkazu:



Opis działania:

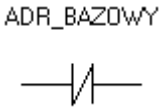
Styk zwierny zachowuje się jak przełącznik, który przewodzi prąd wtedy, gdy element pod wybranym adresem bazowym jest zamknięty. Przełącznik może być w wersji bistabilnej lub astabilnej.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	tak	tak	tak	tak	tak	tak	nie

Styk rozwierny

Symbol rozkazu:



Opis działania:

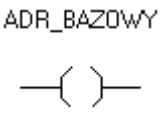
Styk rozwierny zachowuje się jak przełącznik, który przewodzi prąd wtedy, gdy element pod wybranym adresem bazowym jest otwarty. Przełącznik może być w wersji bistabilnej lub astabilnej.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	tak	tak	tak	tak	tak	tak	nie

Cewka zwierna

Symbol rozkazu:



Opis działania:

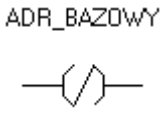
Cewka zwierna zamyka obwód wyjściowy, czyli przyjmuje wartość 1. Jeśli prąd do niej nie dopływa, to przyjmuje wartość 0.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	tak	tak	tak	nie	tak	nie

Cewka rozwierna

Symbol rozkazu:



Opis działania:

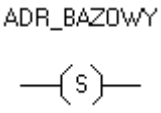
Cewka rozwierna otwiera obwód wyjściowy, czyli przyjmuje wartość 0. Jeśli prąd do niej nie dopływa, to przyjmuje wartość 1.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	tak	tak	tak	nie	tak	nie

Cewka S

Symbol rozkazu:



Opis działania:

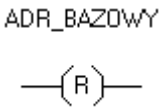
Cewka typu S działa w połączeniu z cewką typu R. Ich działanie jest identyczne jak w przypadku przerzutnika typu RS. Jeśli cewka typu S jest zasilana, to komórka pamięci pod wybranym adresem bazowym ustawi się na 1. Jeśli cewka ta nie jest zasilana, to zawartość komórki pod adresem bazowym nie ulega zmianie.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	tak	tak	tak	nie	tak	nie

Cewka R

Symbol rozkazu:



Opis działania:

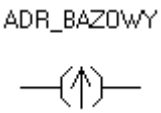
Cewka typu R działa w połączeniu z cewką typu S. Ich działanie jest identyczne jak w przypadku przerzutnika typu RS. Jeśli cewka typu R jest zasilana, to komórka pamięci pod wybranym adresem bazowym ustawi się na 0. Jeśli cewka ta nie jest zasilana, to zawartość komórki pod adresem bazowym nie ulega zmianie.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	tak	tak	tak	nie	tak	nie

Cewka z narastającym zboczem

Symbol rozkazu:



Opis działania:

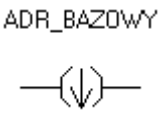
Cewka ta reaguje na narastające zbocze sygnału wejściowego. W momencie, gdy podczas poprzedniego cyklu nie była ona zasilana, a w tym cyklu dopływa do niej prąd, ustawia ona zawartość komórki pod adresem bazowym na 1, ale tylko na ten jeden cykl. We wszystkich innych przypadkach ustawia ona 0.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	tak	tak	tak	nie	tak	nie

Cewka z opadającym zboczem

Symbol rozkazu:



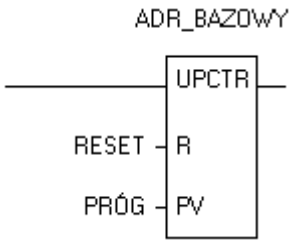
Opis działania:

Cewka ta reaguje na opadające zbocze sygnału wejściowego. W momencie, gdy podczas poprzedniego cyklu była ona zasilana, a w tym cyklu nie dopływa do niej prąd, ustawia ona zawartość komórki pod adresem bazowym na 1, ale tylko na ten jeden cykl. We wszystkich innych przypadkach ustawia ona 0.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	tak	tak	tak	nie	tak	nie

Licznik zliczanie w górę



Symbol rozkazu:

Opis działania:

Licznik UPCTR (Up Counter) używany jest do zliczania impulsów. Za każdym razem, gdy wejście licznika zmienia stan z niskiego na wysoki zwiększana jest liczba aktualnie zliczonych impulsów. Wejście RESET służy do wyzerowania licznika. Wyjście licznika ustawia się w stan wysoki, gdy liczba zliczonych impulsów jest równa lub większa niż wartość na wejściu PV.

Licznik w czasie pracy przechowuje swój stan w pamięci sterownika. Do zachowania całego stanu potrzebuje trzech komórek pamięci. Adres bazowy pokazuje na pierwszą z tych trzech komórek. Istotną sprawą przy programowaniu sterowników jest to, aby żadne dwa bloki funkcyjne nie trzymały swoich danych w tych samych obszarach. Jeśli np. dla jednego licznika

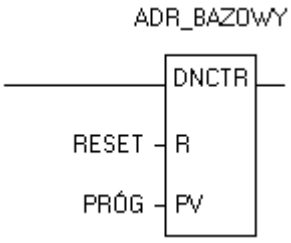
wybierzemy jako adres bazowy komórkę %R0005, oznacza to, że komórki %R0006 i %R0007 są zajęte przez ten licznik. Inna funkcja może użyć dopiero adresu %R0008.
Jeśli zajdzie konieczność odczytania ilości aktualnie zliczonych impulsów, można odczytać ją spod adresu równego adresowi bazowemu.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	nie	tak	nie	nie	nie	nie
R	tak	nie	tak	tak	nie	nie	nie
PV	nie	nie	tak	tak	tak	nie	tak

Licznik – zliczanie w dół

Symbol rozkazu:



Opis działania:

Licznik DNCTR (Down Counter) używany jest do zliczania impulsów w dół. Za każdym razem, gdy wejście licznika zmienia stan z niskiego na wysoki zmniejszana jest liczba aktualnie zliczonych impulsów. Wejście RESET służy do ustawienia ilości impulsów do zliczenia na wartość z wejścia PV. Wyjście licznika ustawia się w stan wysoki, gdy liczba zliczonych impulsów jest równa lub mniejsza niż zero.

Licznik w czasie pracy przechowuje swój stan w pamięci sterownika. Do zachowania całego stanu potrzebuje trzech komórek pamięci. Adres bazowy pokazuje na pierwszą z tych trzech komórek. Istotną sprawą przy programowaniu sterowników jest to, aby żadne dwa bloki funkcyjne nie trzymały swoich danych w tych samych obszarach. Jeśli np. dla jednego licznika wybierzemy jako adres bazowy komórkę %R0005, oznacza to, że komórki %R0006 i %R0007 są zajęte przez ten licznik. Inna funkcja może użyć dopiero adresu %R0008.

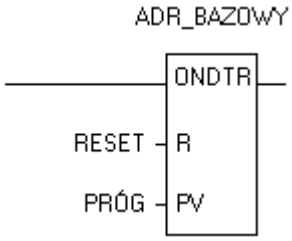
Jeśli zajdzie konieczność odczytania ilości aktualnie zliczonych impulsów, można odczytać ją spod adresu równego adresowi bazowemu.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	nie	tak	nie	nie	nie	nie
R	tak	nie	tak	tak	nie	nie	nie
PV	nie	nie	tak	tak	tak	nie	tak

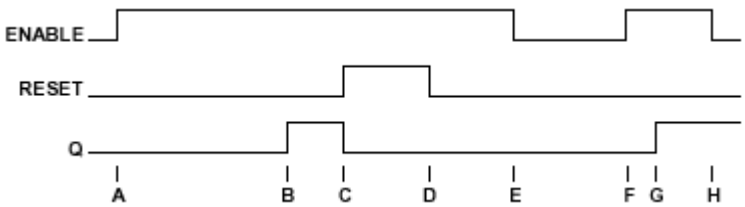
Timer załączający ONDTR

Symbol rozkazu:



Opis działania:

Timer ONDTR (on-delay timer) zwiększa zawartość wewnętrznego licznika o jeden, jeśli dopływa do niego zasilanie. W przypadku przerwy w dopływie zasilania wartość wewnętrznego licznika jest zachowywana. Mierzony czas wyrażany jest w dziesiątkach milisekund. Jeśli ilość zmierzonego czasu będzie równa lub większa od wartości podawanej na wejście PV, na wyjściu pojawi się stan 1. W przeciwnym wypadku na wyjściu obecny jest stan 0. Stan wewnętrznego licznika można wyzerować podając wartość różną od zera na wejście RESET. Przykładowe przebiegi czasowe timera ONDTR pokazano na rys. 2.



Rys. 2 Przykładowy przebieg czasowy timera ONDTR

A - na wejściu ENABLE pojawia się stan wysoki; timer zaczyna zliczać czas.
B - zliczony czas osiągnął wartość PV; wyjście przyjmuje stan wysoki.
C - na wejściu RESET pojawia się stan wysoki; wyjście przyjmuje stan niski, czas zliczany jest ustawiany na 0.
D - RESET przechodzi w stan niski; timer znów zaczyna zliczać czas.

E - wejście ENABLE przyjmuje stan niski; timer przestaje zliczać, ale zapamiętuje ilość zliczonego czasu.
F - wejście znów przechodzi w stan wysoki; timer kontynuuje zliczanie czasu.
G - zliczona wartość zrównuje się z PV; wyjście przyjmuje stan wysoki. Timer kontynuuje zliczanie czasu dopóki ENABLE nie przyjmie stanu niskiego albo RESET nie przyjmie stanu wysokiego.
H - ENABLE przyjmuje stan niski; timer przestaje zliczać czas.

Timer w czasie pracy przechowuje swój stan w pamięci sterownika. Do zachowania całego stanu potrzebuje trzech komórek pamięci. Adres bazowy pokazuje na pierwszą z tych trzech komórek. Istotną sprawą przy programowaniu sterowników jest to, aby żadne dwa bloki funkcyjne nie trzymały swoich danych w tych samych obszarach. Jeśli np. dla jednego timera wybierzemy jako adres bazowy komórkę %R0005, to oznacza to, że komórki %R0006 i %R0007 są zajęte przez ten timer. Inna funkcja może użyć dopiero adresu %R0008.

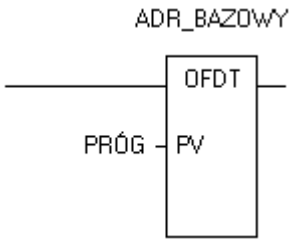
Jeśli zajdzie konieczność odczytania ilości aktualnie zliczonego czasu, można odczytać ją spod adresu równego adresowi bazowemu.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	nie	tak	nie	nie	nie	nie
R	tak	nie	tak	tak	nie	nie	nie
PV	nie	nie	tak	tak	tak	nie	tak

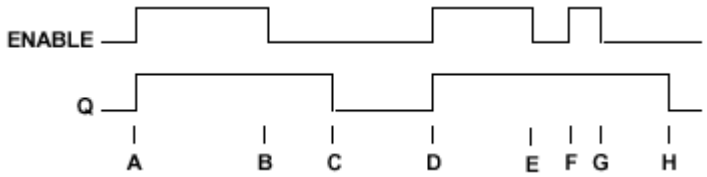
Timer wyłączający OFDT

Symbol rozkazu:



Opis działania:

Timer OFDT (off-delay timer) ma za zadanie opóźniać opadające zbocze sygnału wejściowego. Ilość czasu, o jaką chcemy opóźnić zbocze, podaje się na wejście PV (= wartość * 10 ms). Przykładowe przebiegi czasowe timera OFDT przedstawiono na rys. 3.



Rys. 3 Przykładowy przebieg czasowy timera OFDT

A - wejście przechodzi w stan wysoki; wyjście przechodzi w stan wysoki, timer jest resetowany.
B - wejście przechodzi w stan niski; timer rozpoczyna zliczanie czasu.
C - zliczony czas osiągnął wartość PV; wyjście przechodzi w stan niski i timer przestaje zliczać czas.
D - wejście przechodzi w stan wysoki; timer jest resetowany.
E - wejście przechodzi w stan niski; timer rozpoczyna zliczanie czasu.
F - wejście przechodzi w stan wysoki; timer jest resetowany.
G - wejście przechodzi w stan niski; timer rozpoczyna zliczanie czasu.
H - zliczony czas osiągnął wartość PV; wyjście przechodzi w stan niski i timer przestaje zliczać czas.

Timer w czasie pracy przechowuje swój stan w pamięci sterownika. Do zachowania całego stanu potrzebuje trzech komórek pamięci. Adres bazowy pokazuje na pierwszą z tych trzech komórek. Istotną sprawą przy programowaniu sterowników jest to, aby żadne dwa bloki funkcyjne nie trzymały swoich danych w tych samych obszarach. Jeśli np. dla jednego timera wybierzemy jako adres bazowy komórkę %R0005, to oznacza to, że komórki %R0006 i %R0007 są zajęte przez ten timer. Inna funkcja może użyć dopiero adresu %R0008.

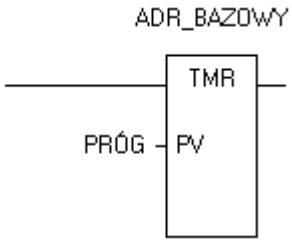
Jeśli zajdzie konieczność odczytania ilości aktualnie zliczonego czasu, można odczytać ją spod adresu równego adresowi bazowemu.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	nie	tak	nie	nie	nie	nie
PV	nie	nie	tak	tak	tak	nie	tak

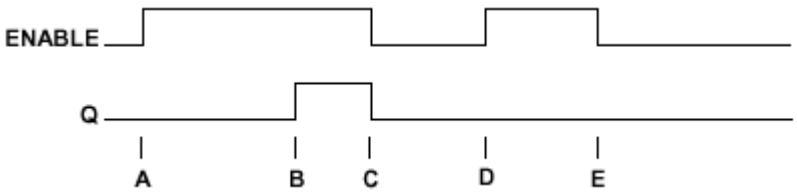
Timer

Symbol rozkazu:



Opis działania:

Timer TMR ma za zadanie odmierzać podaną ilość czasu. Na wejściu PV podaje się żądany czas (= wartość * 10 ms). Jeśli na wejściu przez dany okres czasu utrzyma się stan wysoki, to wyjście timera przyjmie stan wysoki. Pojawienie się na wejściu stanu niskiego powoduje zresetowanie timera. Przykładowe przebiegi czasowe timera TMR pokazano na rys. 4.



Rys. 4 Przykładowy przebieg czasowy timera TMR

A - wejście przyjmuje stan wysoki; timer rozpoczyna zliczanie czasu.
B - zliczona wartość czasu zrównuje się z PV; wyjście przechodzi w stan wysoki i timer kontynuuje zliczanie czasu.
C - wejście przybiera stan niski; wyjście również przyjmuje stan niski i timer zostaje zresetowany.
D - wejście przechodzi w stan wysoki; timer rozpoczyna odliczanie czasu.
E - wejście przechodzi w stan niski zanim timer osiągnie wartość PV; wyjście pozostaje w stanie niskim, timer jest resetowany.

Timer w czasie pracy przechowuje swój stan w pamięci sterownika. Do zachowania całego stanu potrzebuje trzech komórek pamięci. Adres bazowy pokazuje na pierwszą z tych trzech komórek. Istotną sprawą przy programowaniu sterowników jest to, aby żadne dwa bloki funkcyjne nie trzymały swoich danych w tych samych obszarach. Jeśli np. dla jednego timera wybierzemy jako adres bazowy komórkę %R0005, to oznacza to, że komórki %R0006 i %R0007 są zajęte przez ten timer. Inna funkcja może użyć dopiero adresu %R0008.

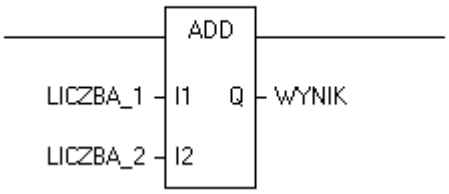
Jeśli zajdzie konieczność odczytania ilości aktualnie zliczonego czasu, można odczytać ją spod adresu równego adresowi bazowemu.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
Adres bazowy	nie	nie	tak	nie	nie	nie	nie
PV	nie	nie	tak	tak	tak	nie	tak

Dodawanie ADD

Symbol rozkazu:



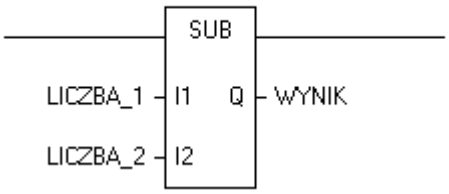
Opis działania:

Funkcja ta służy do dodawania dwóch liczb całkowitych, podawanych na wejścia I1 i I2. Wynik dodawania zapisywany jest na wyjście Q. Operacja dodawania wykonywana jest tylko wtedy, gdy na wejściu bloku jest stan wysoki. Wyjście bloku ustawia się w stan wysoki wtedy, gdy operacja dodawania wykonana zostanie pomyślnie.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak
Q	tak	tak	tak	tak	tak	tak	tak

Odejmowanie SUB



Symbol rozkazu:

Opis działania:

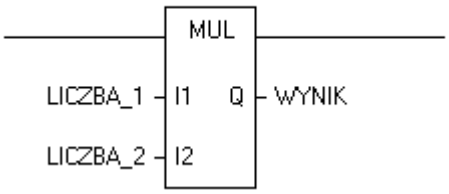
Funkcja ta służy do odejmowania dwóch liczb całkowitych, podawanych na wejścia I1 i I2. Wynik odejmowania zapisywany jest na wyjście Q. Operacja odejmowania wykonywana jest tylko wtedy, gdy na wejściu bloku jest stan wysoki. Wyjście bloku ustawia się w stan wysoki wtedy, gdy operacja odejmowania wykonana zostanie pomyślnie.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak
Q	tak	tak	tak	tak	tak	tak	tak

Mnożenie MUL

Symbol rozkazu:



Opis działania:

Funkcja ta służy do mnożenia dwóch liczb całkowitych, podawanych na wejścia I1 i I2. Wynik mnożenia zapisywany jest na wyjście Q. Operacja mnożenia wykonywana jest tylko

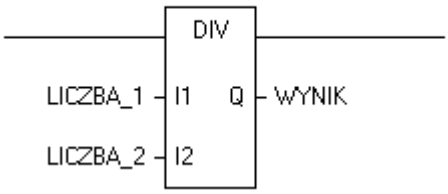
wtedy, gdy na wejściu bloku jest stan wysoki. Wyjście bloku ustawia się w stan wysoki wtedy, gdy operacja mnożenia wykonana zostanie pomyślnie.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak
Q	tak	tak	tak	tak	tak	tak	tak

Dzielenie DIV

Symbol rozkazu:



Opis działania:

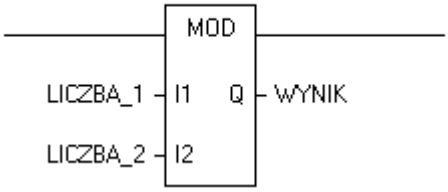
Funkcja ta służy do dzielenia dwóch liczb całkowitych, podawanych na wejścia I1 i I2. Wynik dzielenia zapisywany jest na wyjście Q. Operacja dzielenia wykonywana jest tylko wtedy, gdy na wejściu bloku jest stan wysoki. Wyjście bloku ustawia się w stan wysoki wtedy, gdy operacja dzielenia wykonana zostanie pomyślnie. Wynik jest liczbą całkowitą z pominięciem reszty dzielenia.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak
Q	tak	tak	tak	tak	tak	tak	tak

Dzielenie z resztą MOD

Symbol rozkazu:



Opis działania:

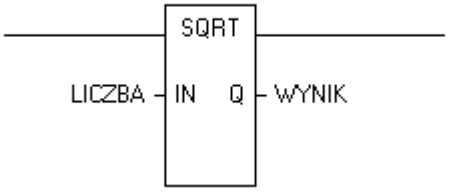
Funkcja ta służy do obliczania reszty z dzielenia dwóch liczb całkowitych, podawanych na wejścia I1 i I2. Reszta z dzielenia zapisywana jest na wyjście Q. Operacja dzielenia wykonywana jest tylko wtedy, gdy na wejściu bloku jest stan wysoki. Wyjście bloku ustawia się w stan wysoki wtedy, gdy operacja dzielenia wykonana zostanie pomyślnie. Wynik jest liczbą całkowitą jako reszta z dzielenia.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak
Q	tak	tak	tak	tak	tak	tak	tak

Pierwiastek SQRT

Symbol rozkazu:



Opis działania:

Funkcja ta służy do obliczania pierwiastka z liczby podawanej na wejście IN. Wynik pierwiastkowania zapisywany jest na wyjście Q. Operacja pierwiastkowania wykonywana jest

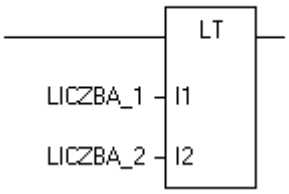
tylko wtedy, gdy na wejściu bloku jest stan wysoki. Wyjście bloku ustawia się w stan wysoki wtedy, gdy operacja pierwiastkowania wykonana zostanie pomyślnie.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
IN	tak	tak	tak	tak	tak	tak	tak
Q	tak	tak	tak	tak	tak	tak	tak

Relacja mniejszy LT

Symbol rozkazu:



Opis działania:

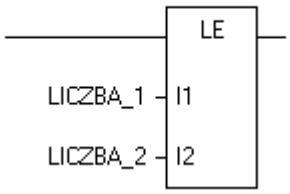
Ten blok funkcyjny służy do porównywania dwóch wartości liczbowych, podawanych na wejścia I1 i I2. Na wyjściu pojawi się stan wysoki wtedy i tylko wtedy, gdy I1 < I2. LT to skrót od "less than", tzn. "mniejszy niż".

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak

Relacja mniejszy równy LE

Symbol rozkazu:



Opis działania:

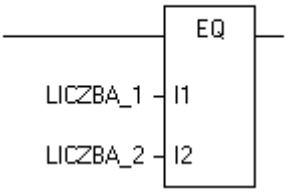
Ten blok funkcyjny służy do porównywania dwóch wartości liczbowych, podawanych na wejścia I1 i I2. Na wyjściu pojawi się stan wysoki wtedy i tylko wtedy, gdy I1 <= I2. LT to skrót od "less than or equal", tzn. "mniejszy lub równy".

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak

Relacja równy EQ

Symbol rozkazu:



Opis działania:

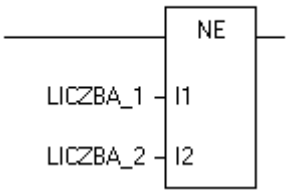
Ten blok funkcyjny służy do porównywania dwóch wartości liczbowych, podawanych na wejścia I1 i I2. Na wyjściu pojawi się stan wysoki wtedy i tylko wtedy, gdy I1 = I2. EQ to skrót od "equal", tzn. "równe".

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak

Relacja różny NE

Symbol rozkazu:



Opis działania:

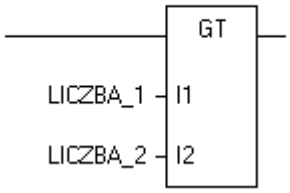
Ten blok funkcyjny służy do porównywania dwóch wartości liczbowych, podawanych na wejścia I1 i I2. Na wyjściu pojawi się stan wysoki wtedy i tylko wtedy, gdy I1 <> I2. NE to skrót od "not equal", tzn. "nierówne".

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak

Relacja większy GT

Symbol rozkazu:



Opis działania:

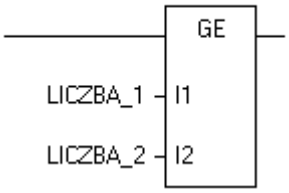
Ten blok funkcyjny służy do porównywania dwóch wartości liczbowych, podawanych na wejścia I1 i I2. Na wyjściu pojawi się stan wysoki wtedy i tylko wtedy, gdy I1 > I2. GT to skrót od "greater than", tzn. "większy niż".

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak

Relacja większy równy GE

Symbol rozkazu:



Opis działania:

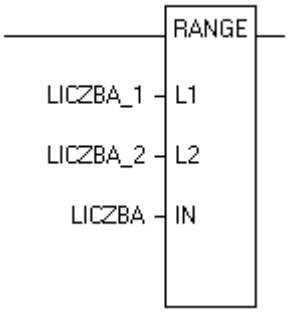
Ten blok funkcyjny służy do porównywania dwóch wartości liczbowych, podawanych na wejścia I1 i I2. Na wyjściu pojawi się stan wysoki wtedy i tylko wtedy, gdy I1 >= I2. GE to skrót od "greater than or equal", tzn. "większy lub równy".

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak

Relacja zakres RANGE

Symbol rozkazu:



Opis działania:

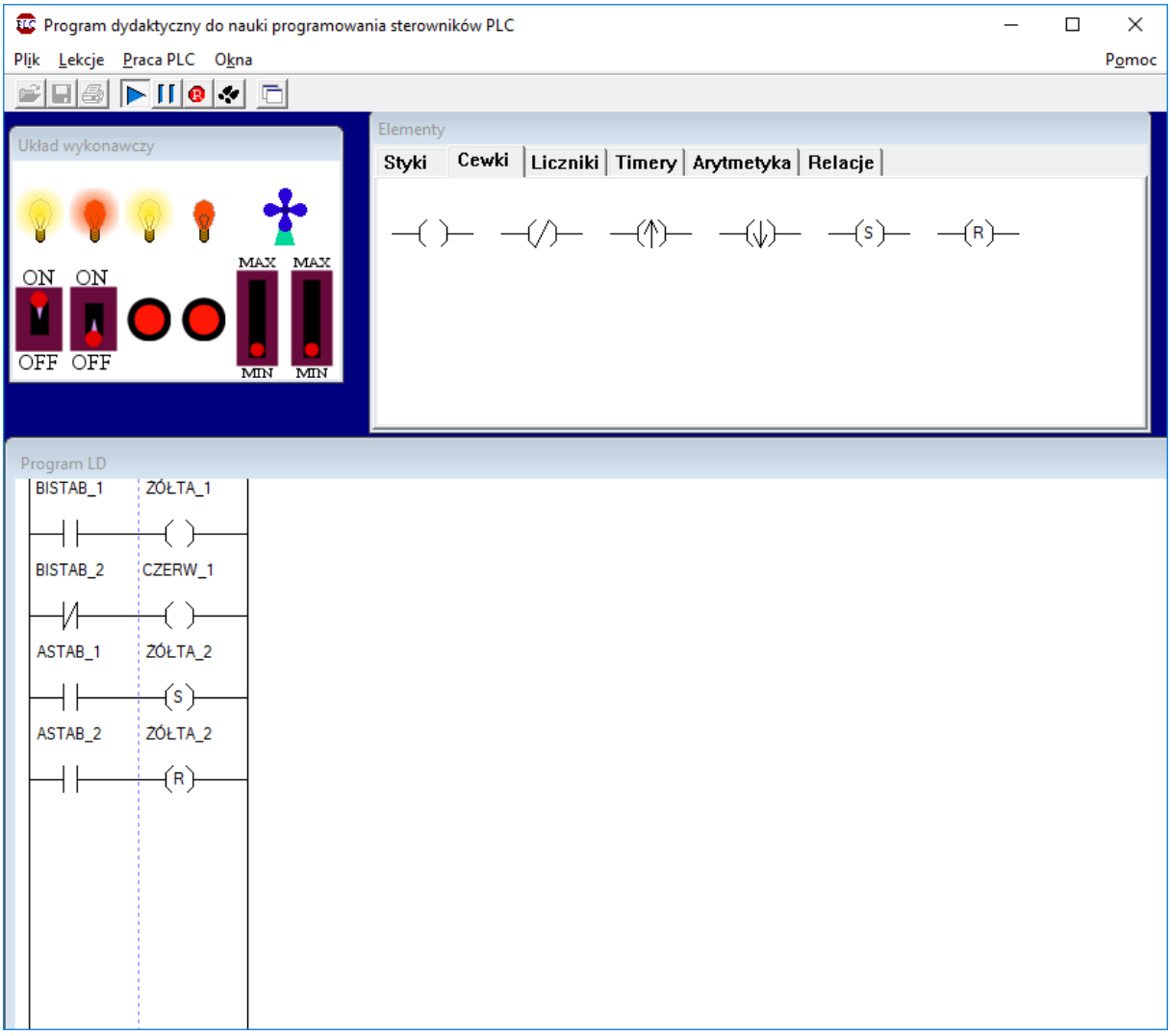
Ten blok funkcyjny służy do sprawdzenia, czy liczba dana na wejście IN zawiera się pomiędzy liczbami danymi na wejścia I1 oraz I2. Na wyjściu pojawi się stan wysoki wtedy i tylko wtedy, gdy I1 <= IN <= I2 lub I2 <= IN <= I1. RANGE oznacza zakres.

Dopuszczalne typy pamięci:

Parametr	%I	%Q	%R	%M	%AI	%AQ	const
I1	tak	tak	tak	tak	tak	tak	tak
I2	tak	tak	tak	tak	tak	tak	tak
IN	tak	tak	tak	tak	tak	tak	tak

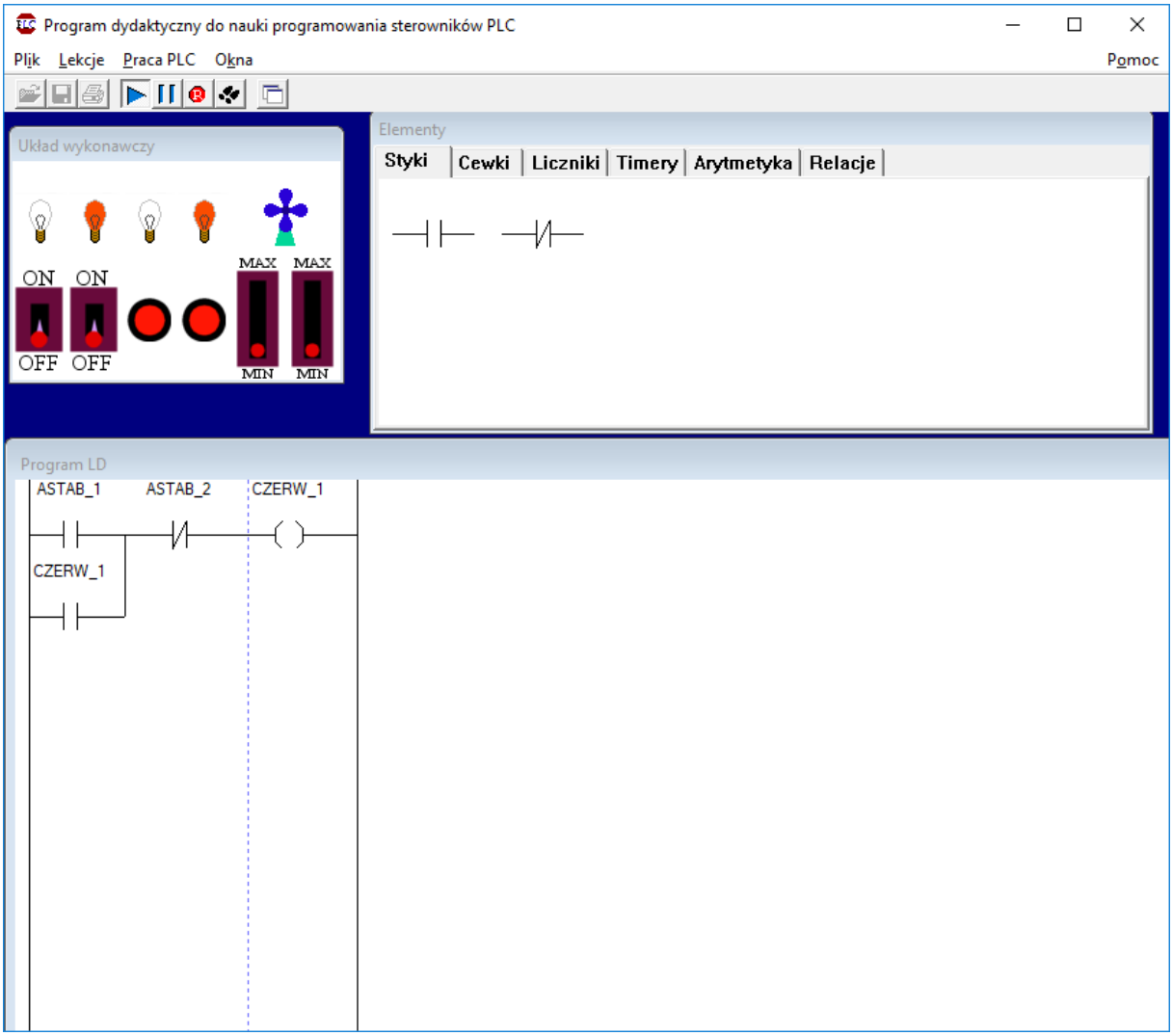
Programowanie sterownika

W obwodach elektrycznych zawsze występuje załączanie urządzeń elektrycznych. Proste układy mają załączanie ręczne i bezpośrednie. W układach sterowania załączanie realizuje sterownik wraz z oprogramowaniem. Przykłady załączania pokazano na rys. 5.



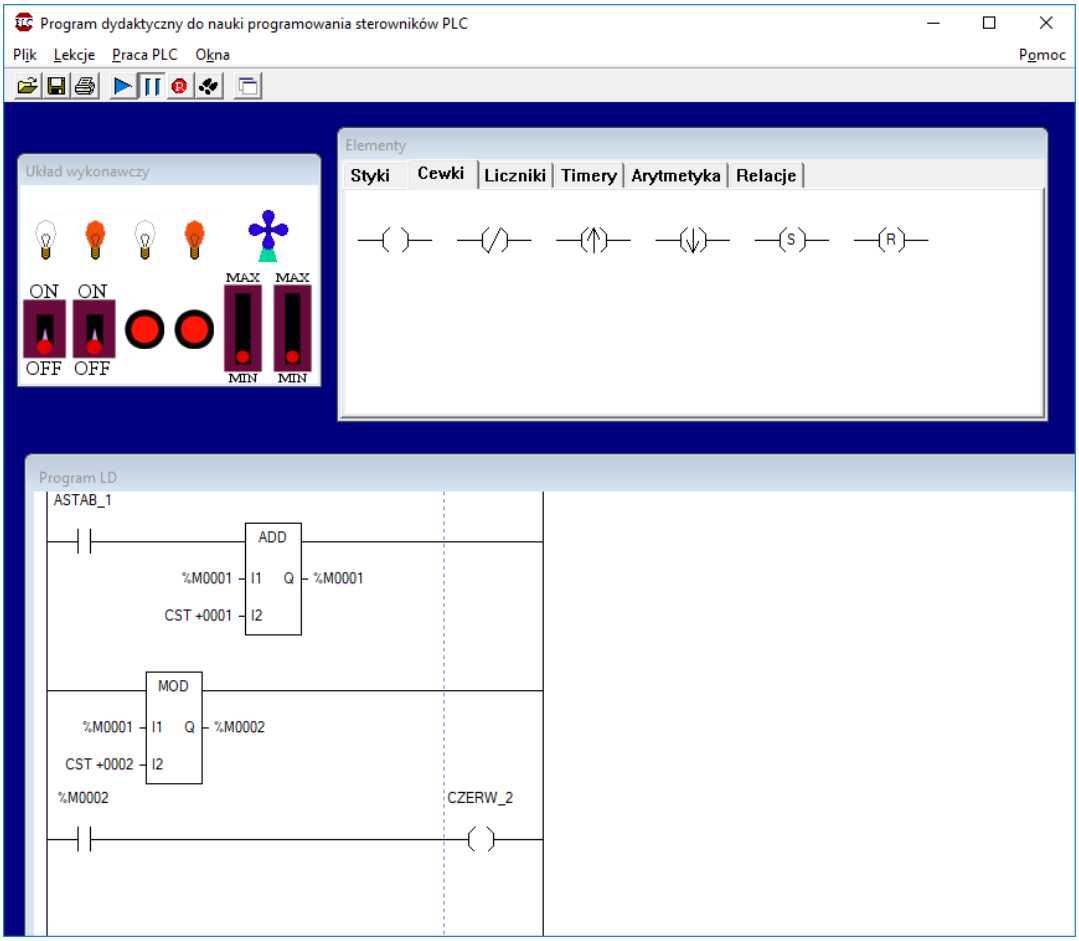
Rys. 5 Załączanie – proste elementy wejściowe i wyjściowe

Inną formą załączania stosowane w technice jest załączanie impulsowe z podtrzymaniem przedstawione na rys. 6.

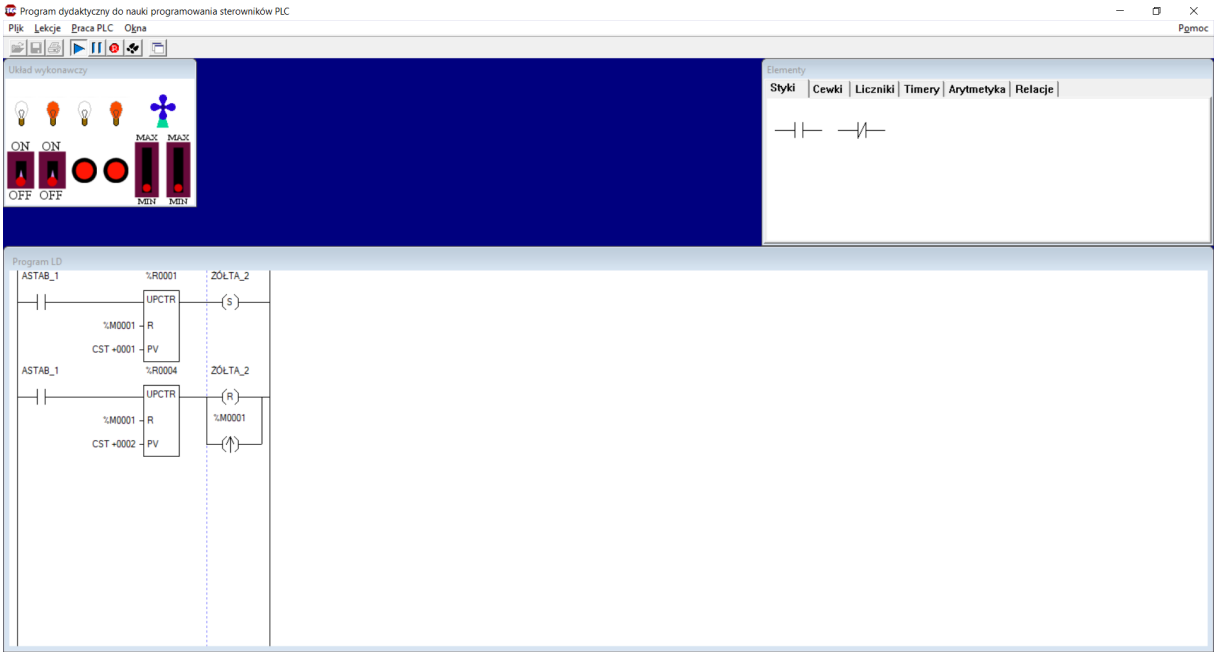


Rys. 6 Załączanie impulsowe z podtrzymaniem

Załączanie można realizować pojedynczym przyciskiem – jednorazowe wciśnięcie powoduje załączenie wyjścia, a kolejne wciśnięcie wyłącza element. Wymaga to określenia bitu parzystości jak na rys. 7 i 8.

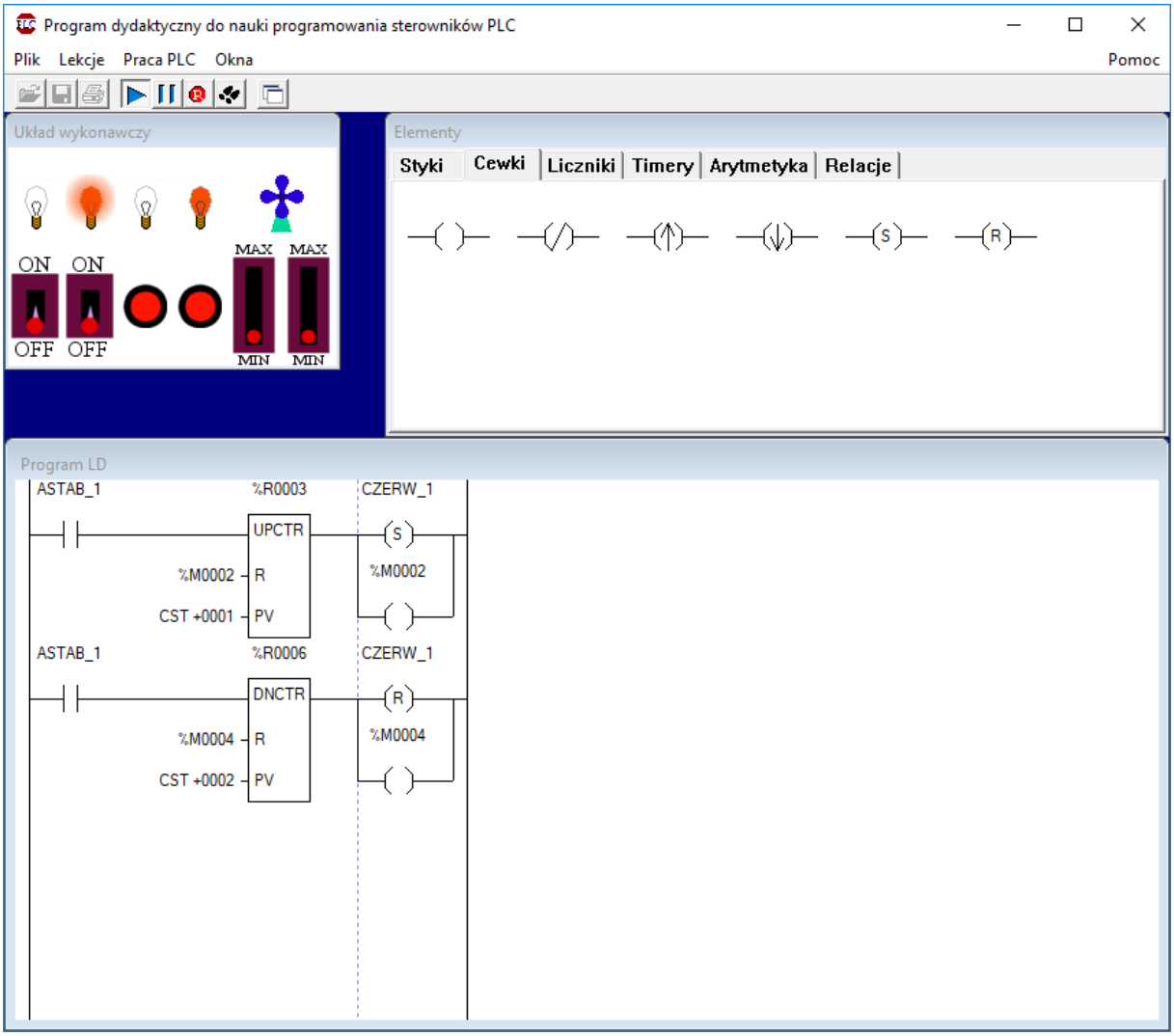


Rys. 7 Załączanie pojedynczym przyciskiem, z określeniem bitu parzystości



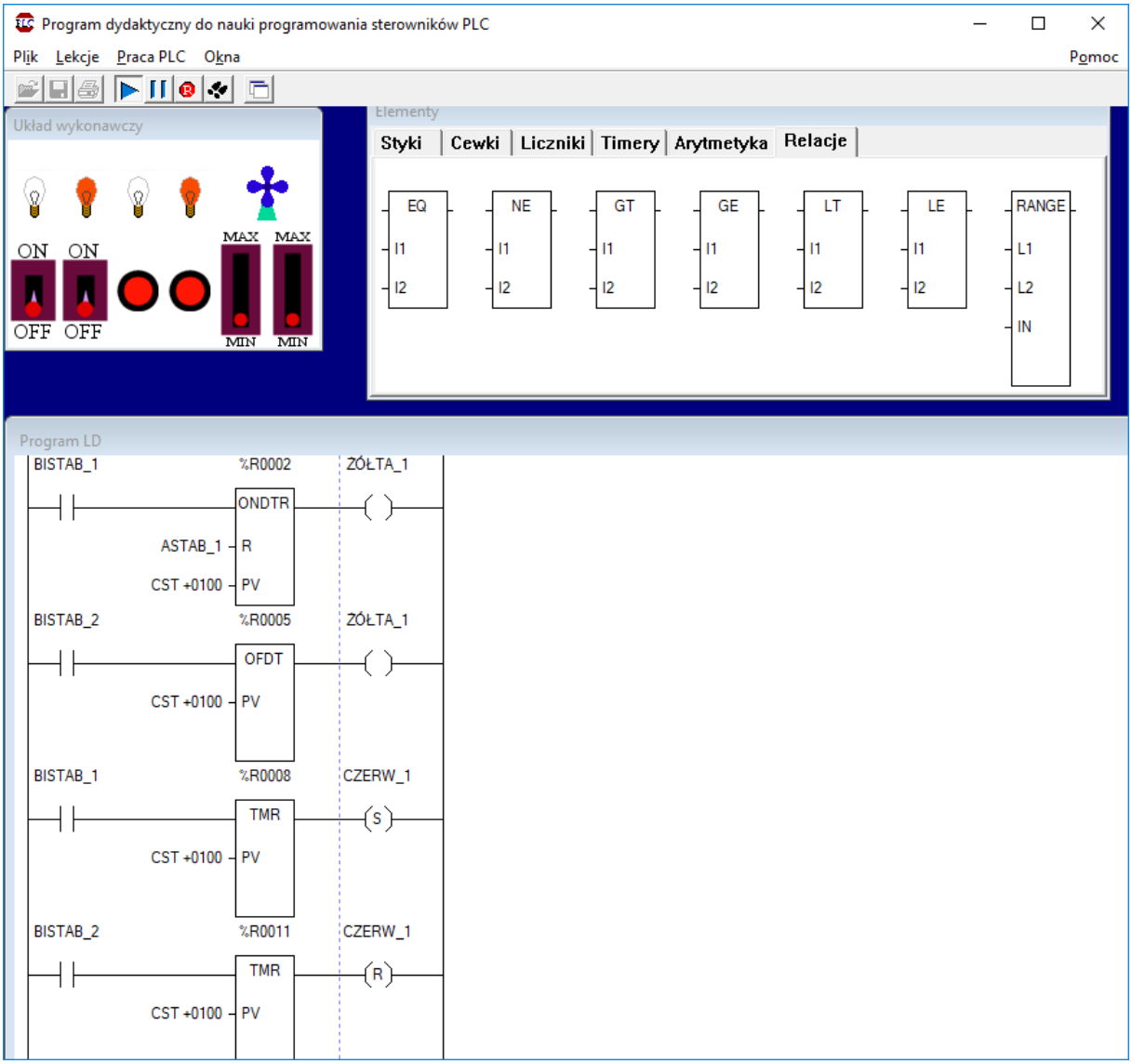
Rys. 8 Załączanie pojedynczym przyciskiem, z zastosowaniem licznika impulsów

Ważnym elementem w sterowaniu są timery. Przykład pokazano na rys. 9.

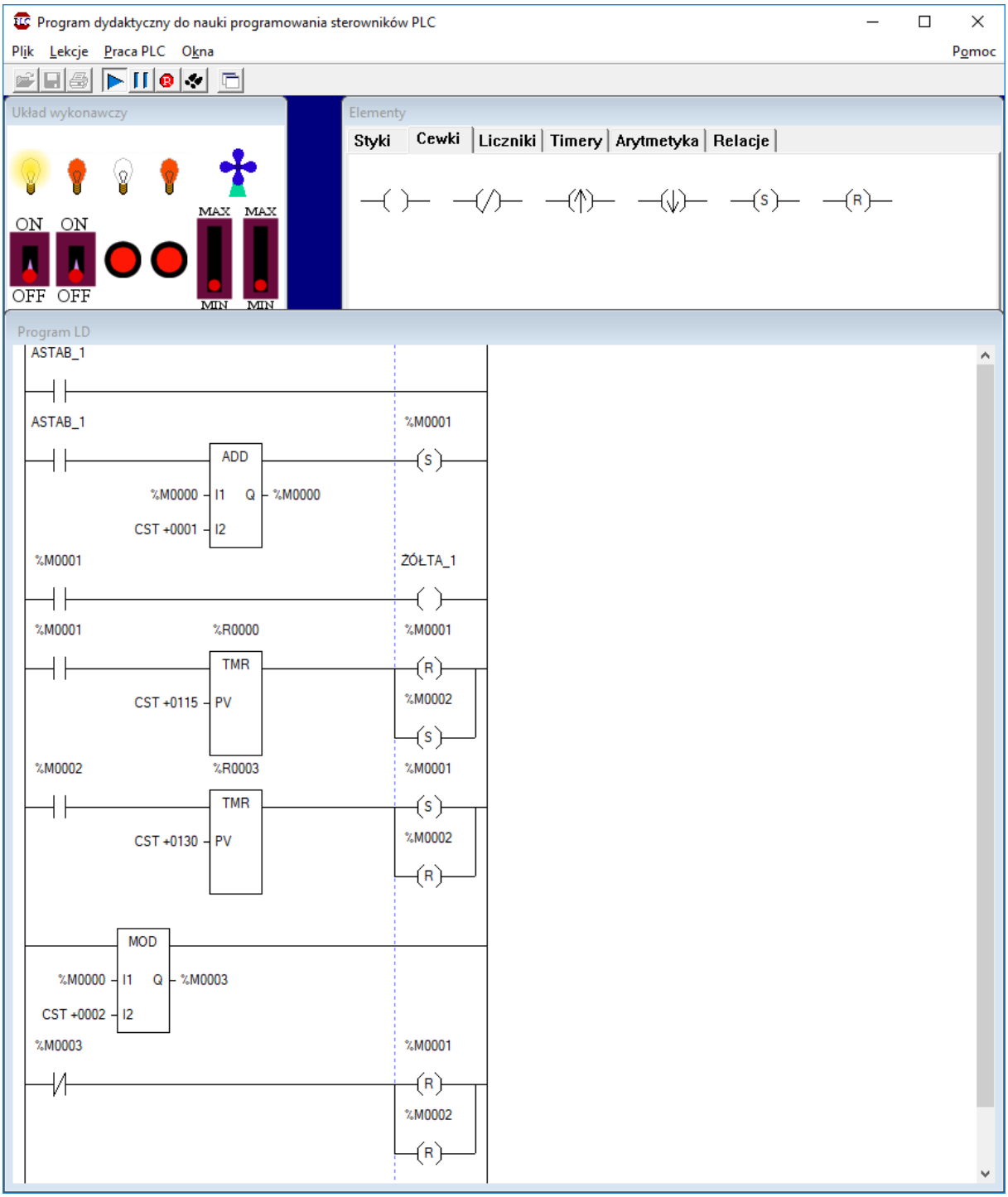


Rys. 9 Załączanie z użyciem timera

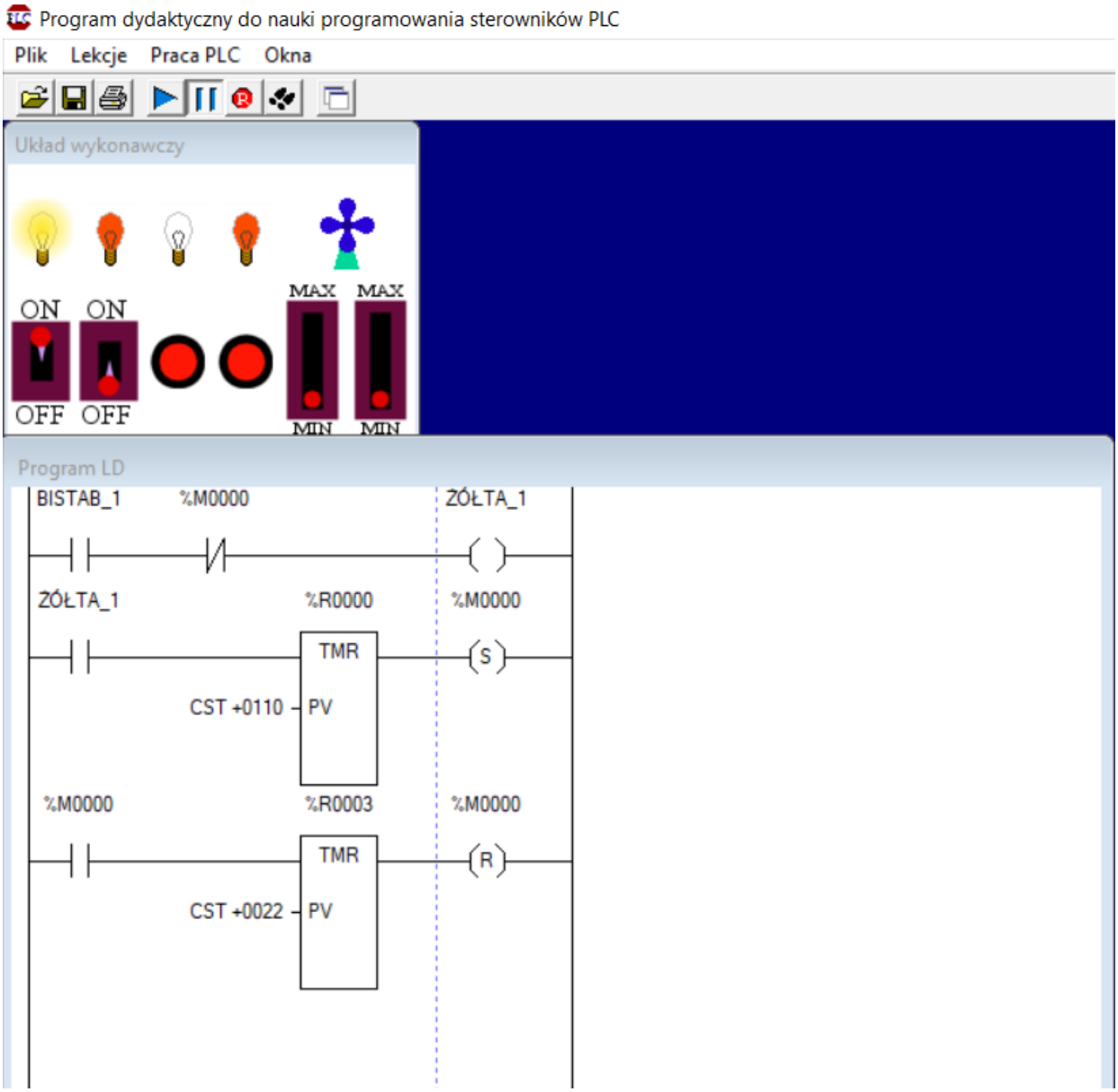
Innym rozwiązaniem stosowanym w sterowaniu jest pulsowanie na wyjściu jako ostrzeżenie lub podkreślenie znaczenia informacji. Przykład pokazano na rys. 10, 11, 12 i 13.



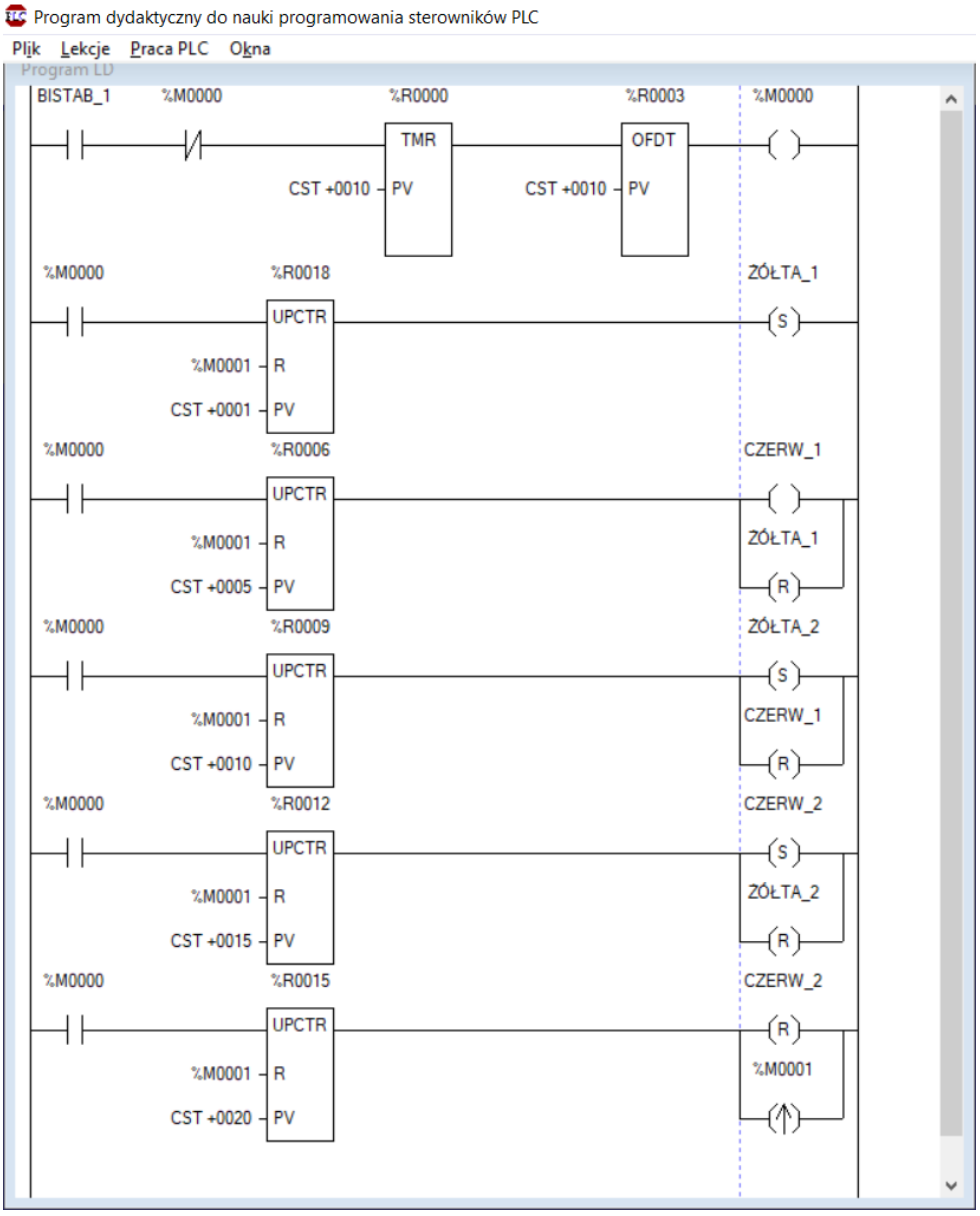
Rys. 10 Załączanie z użyciem timera – migotanie



Rys. 11 Załączanie przełączające

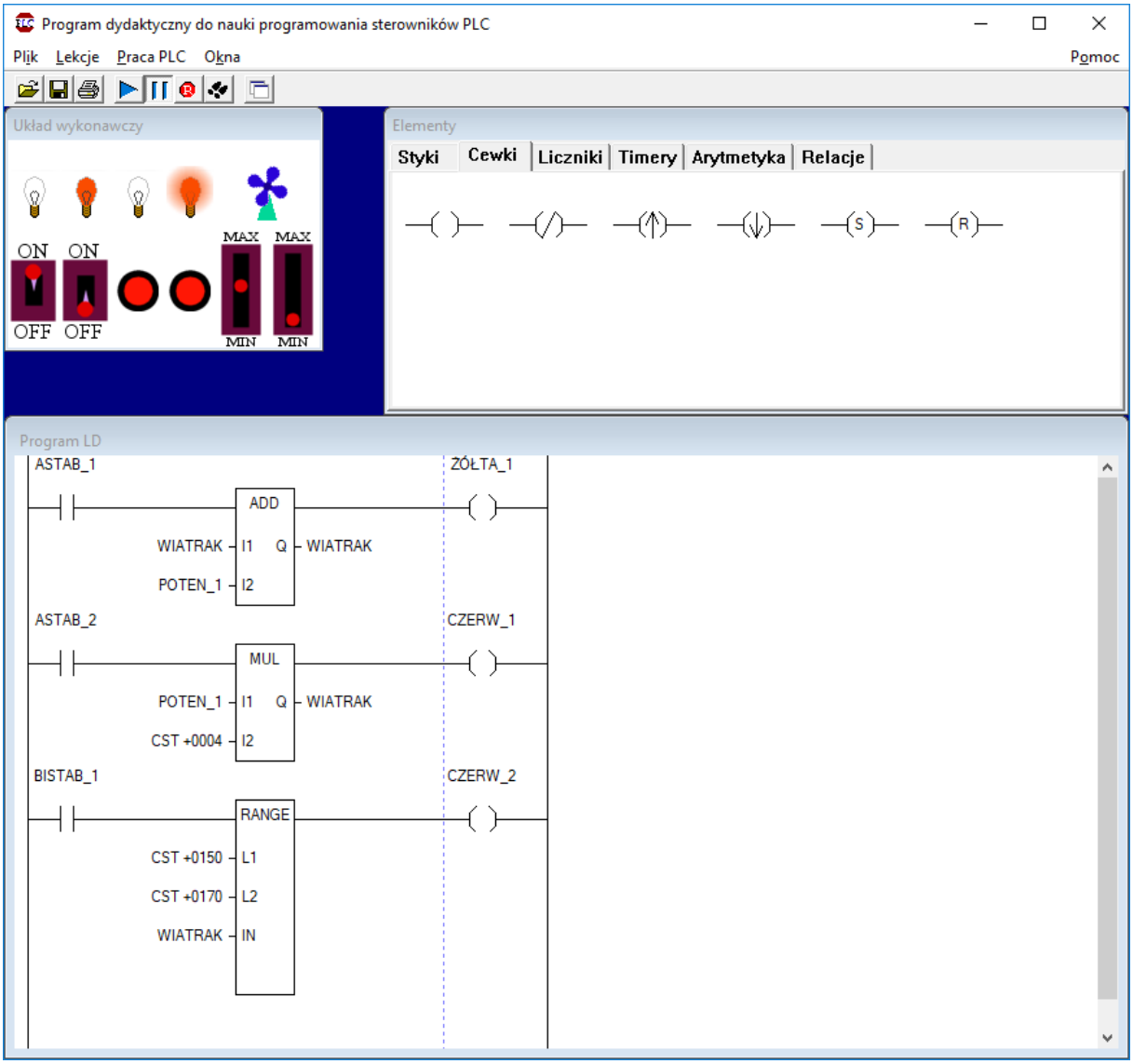


Rys. 12 Załączanie naprzemienne



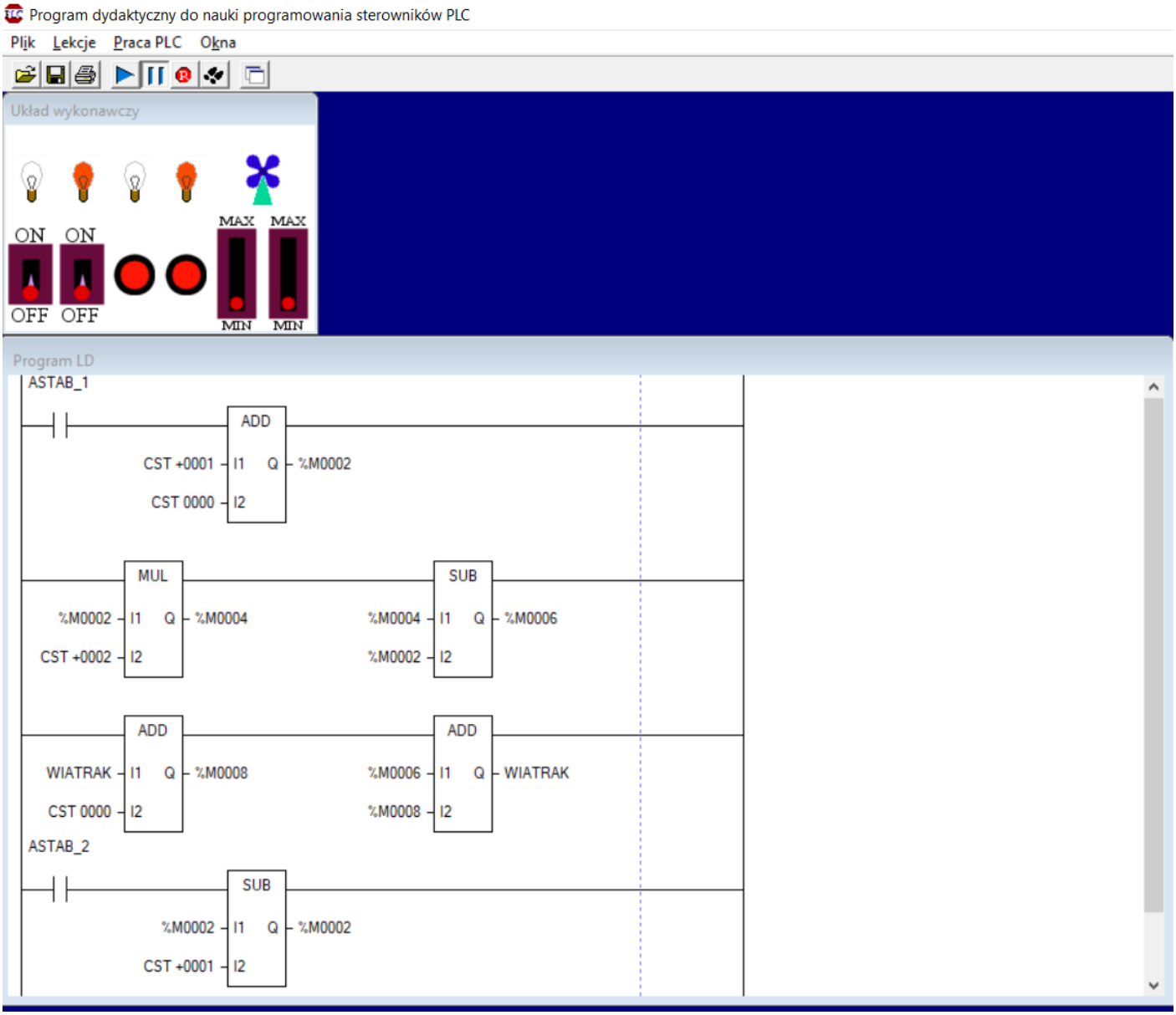
Rys. 13 Załączanie kolejne wyjść po sobie, np. sygnalizacja świetlna na skrzyżowaniu

W pracy sterownika jest też przetwarzanie sygnału analogowego w zakresie liczb całkowitych. Przykład pokazano na rys. 14.



Rys. 14 Praca ze sygnałem analogowym na sterowniku PLC

Przykład programowania równania różnicowego $Y_R=2*e(n)-e(n-1)+Y_R(n-1)$



Rys. 15 Programowanie równania różnicowego regulatora P na sterowniku PLC

7. Pytania kontrolne

a. Bloki i programowanie sterowników PLC w języku drabinkowym.
b. Sygnały wejścia i wyjścia cyfrowe i analogowe sterownika PLC.
c. Sygnały standardowe i przetworniki cyfrowe.
d. Bloki obsługi czasu.
e. Bloki zliczania impulsów.

Instrukcja do ćwiczenia

Ćwiczenie nr	2
Temat :	**Programowanie cyfrowe** **Regulator P**
Stanowisko laboratoryjne	Symulator PLC
Opracował :	A. Mielewczyk

UNIWERSYTET MORSKI
Gdynia 2022

Instrukcja nr. 2

1. **Temat ćwiczenia:**

   Programowanie regulatora P.

2. **Cel ćwiczenia:**

   Celem ćwiczenia jest napisanie programu regulatora cyfrowego P na symulatorze PLC i wykonanie symulacji procesu regulacji.

3. **Zakres wymaganych wiadomości:**

   - równanie regulatora P,
   - charakterystyka regulatora P,
   - równanie cyfrowe regulatora P,
   - zapis równania regulatora P w języku drabinkowym na sterownik PLC,

4. **Przebieg ćwiczenia:**

   Zapisać program regulatora P w symulatorze sterownika PLC, wprowadzić równanie uchybu regulacji, zamknąć sprzężenie zwrotne i wykonać symulację.

5. **Pomoce i urządzenia:**

   Symulator sterownika PLC.

6. **Treść sprawozdania:**

   Część wstępna, równanie regulatora P, charakterystyki regulatora P, równanie cyfrowe regulatora P, regulator P w zapisie drabinkowym.

WPROWADZENIE

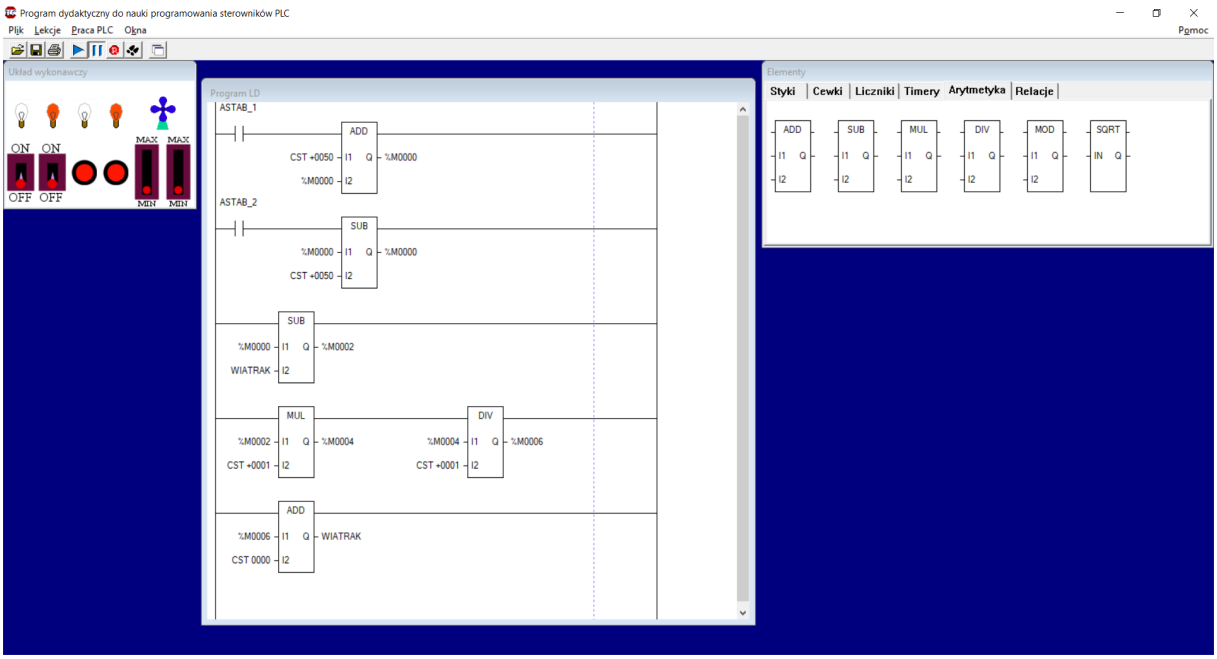
Równanie regulatora P jest następujące:

$$y(t) = k_p e(t)$$

Równanie regulatora P po dyskretyzacji:

$$y(n) = k_p e(n)$$

Przykład programowania regulatora P pokazano na rys. 1.



Rys. 1 Przykład programowania regulatora P w języku drabinkowym

Wprowadzenie wartości zadanej i jej wzrost

Linia nr. 2

Zmniejszanie wartości zadanej

Linia nr. 3

Obliczenie uchybu regulacji e(n) i zapisanie do komórki M2

Linia nr. 4

Wyliczenie działania k*e(n) o wartości całkowitej wzmocnienia i zapisanie wyniku w komórce M4, kolejne dzielenie umożliwia wprowadzenie wzmocnienia ułamkowego i zapisanie wyniku w komórce M6

Linia nr. 6

Zapisanie aktualnej odpowiedzi do obiektu.

7. Pytania kontrolne

1. Równanie i charakterystyki regulatora P.
2. Dyskretyzacja równań różniczkowych.
3. Bloki i programowanie sterowników PLC w języku drabinkowym.
4. Sygnały wejścia i wyjścia analogowe sterownika PLC.
5. Programowanie uchybu regulacji i sprzężenia zwrotnego.

Instrukcja do ćwiczenia

Ćwiczenie nr	3
Temat :	**Programowanie cyfrowe Regulator PI**
Stanowisko laboratoryjne	Symulator PLC
Opracował :	A. Mielewczyk

UNIWERSYTET MORSKI
Gdynia 2022

Instrukcja nr. 3

1. Temat ćwiczenia:

Programowanie regulatora PI.

2. Cel ćwiczenia:

Celem ćwiczenia jest napisanie programu regulatora cyfrowego PI i wykonanie symulacji procesu regulacji.

3. Zakres wymaganych wiadomości:

- równanie regulatora PI,
- charakterystyka regulatora PI,
- równanie cyfrowe regulatora PI,
- zapis równania regulatora PI w języku drabinkowym na sterownik PLC,

4. Przebieg ćwiczenia:

Zapisać program regulatora PI w symulatorze sterownika PLC, wprowadzić równanie uchybu regulacji, zamknąć sprzężenie zwrotne i wykonać symulację.

5. Pomoce i urządzenia:

symulator sterownika PLC.

6. Treść sprawozdania:

część wstępna, równanie regulatora, charakterystyki regulatora, równanie cyfrowe regulatora PI, regulator PI w zapisie drabinkowym.

WPROWADZENIE

Równanie regulatora PI jest następujące:

$$y(t) = k_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt\right)$$

Równanie regulatora PI po dyskretyzacji:

$$y(n) = k_p\left(e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i)\right)$$

Równanie regulatora PI w wersji przyrostowej:

$$y(n) - y(n-1) = k_p\left(e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i)\right) - k_p\left(e(n-1) + \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)\right)$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + k_p\left(\frac{1}{T_i}\sum_{i=0}^{n} e(i) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)\right)$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + k_p\left(\frac{1}{T_i}\sum_{i=0}^{n-1} e(i) + \frac{1}{T_i}e(n) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)\right)$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + k_p\frac{1}{T_i}e(n)$$

$$y(n) - y(n-1) = k_p\left(e(n) + \frac{1}{T_i}e(n)\right) - k_p e(n-1)$$

Końcowa postać równania regulatora PI jest następująca:

$$y(n) = k_p e(n)\left(1 + \frac{1}{T_i}\right) - k_p e(n-1) + y(n-1)$$

Równanie regulatora PI o rozdzielonych funkcjach:

$$y(t) = k_p e(t) + \frac{1}{T_i}\int_0^t e(t)dt$$

Równanie regulatora PI po dyskretyzacji:

$$y(n) = k_p e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i)$$

Równanie regulatora PI w wersji przyrostowej:

$$y(n) - y(n-1) = k_p e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i) - k_p e(n-1) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + \frac{1}{T_i}\sum_{i=0}^{n} e(i) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + \frac{1}{T_i}\sum_{i=0}^{n-1} e(i) + \frac{1}{T_i}e(n) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + \frac{1}{T_i}e(n)$$

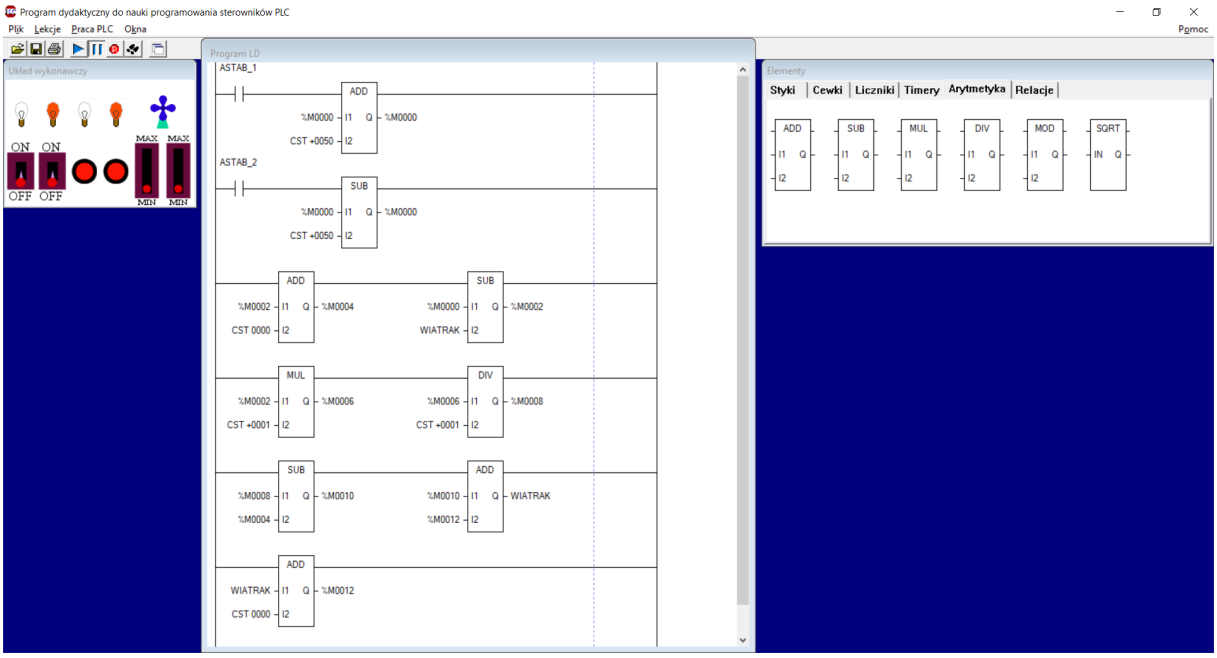
$$y(n) - y(n-1) = \left(k_p + \frac{1}{T_i}\right) e(n) - k_p e(n-1)$$

Końcowa postać równania regulatora PI o rozdzielonych funkcjach jest następująca:

$$y(n) = \left(k_p + \frac{1}{T_i}\right) e(n) - k_p e(n-1) + y(n-1)$$

Przykład regulatora PI w języku drabinkowym

Regulator PI o równaniu $y(n) = e(n) - e(n-1) + y(n-1)$ przedstawiono na rys. 1.



Rys. 1 Przykład programowania regulatora PI w języku drabinkowym

Linia nr. 1

Wprowadzenie wartości zadanej i jej wzrost

Linia nr. 2

Zmniejszanie wartości zadanej

Linia nr. 3

Zapisanie uchybu regulacji e(n-1) do komórki pamięci M4 oraz obliczenie uchybu regulacji e(n) i zapisanie do komórki M2

Linia nr. 4

Wyliczenie działania k*e(n) o wartości ułamkowej wzmocnienia i zapisanie w komórce M8

Linia nr. 5

Odjęcie uchybu regulacji e(n-1) z komórki M4 oraz dodanie odpowiedzi z chwili poprzedniej y(n-1) i zapisanie odpowiedzi aktualnej do obiektu

Linia nr. 6

Zapisanie aktualnej odpowiedzi do komórki M12, która będzie odpowiedzią poprzednią y(n-1) w kolejnym przejściu programu.

Pytania kontrolne

a. Równanie i charakterystyki regulatora PI.
b. Dyskretyzacja równań różniczkowych.
c. Programowanie akcji całkującej.
d. Sygnały wejścia i wyjścia analogowe sterownika PLC.
e. Programowanie uchybu regulacji i sprzężenia zwrotnego.
f. Przepełnienie i nasycenie

Instrukcja do ćwiczenia

Ćwiczenie nr	4
Temat :	**Programowanie cyfrowe Regulator PID**
Stanowisko laboratoryjne	Symulator PLC
Opracował :	A. Mielewczyk

UNIWERSYTET MORSKI
Gdynia 2022

Instrukcja nr. 4

1. Temat ćwiczenia:

Programowanie regulatora PID.

2. Cel ćwiczenia:

Celem ćwiczenia jest napisanie programu regulatora cyfrowego PID i wykonanie symulacji procesu regulacji.

3. Zakres wymaganych wiadomości:

- równanie regulatora PID,
- charakterystyka regulatora PID,
- równanie cyfrowe regulatora PID,
- zapis równania regulatora PID w języku drabinkowym na sterownik PLC,

4. Przebieg ćwiczenia:

Zapisać program regulatora PID w symulatorze sterownika PLC, wprowadzić równanie uchybu regulacji, zamknąć sprzężenie zwrotne i wykonać symulację.

5. Pomoce i urządzenia:

Symulator sterownika PLC.

6. Treść sprawozdania:

część wstępna, równanie regulatora, charakterystyki regulatora, równanie cyfrowe regulatora PID, regulator PID w zapisie drabinkowym.

WPROWADZENIE

Równanie regulatora PID jest następujące:

$$y(t) = k_p\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right)$$

Równanie regulatora PID po dyskretyzacji:

$$y(n) = k_p\left(e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i) + T_d\frac{\Delta e(n)}{T}\right)$$

Równanie regulatora PID w wersji przyrostowej:

$$\begin{aligned} y(n) - y(n-1) &= k_p\left(e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i) + T_d\frac{\Delta e(n)}{T}\right) \\ &\quad - k_p\left(e(n-1) + \frac{1}{T_i}\sum_{i=0}^{n-1} e(i) + T_d\frac{\Delta e(n-1)}{T}\right) \end{aligned}$$

$$\begin{aligned} y(n) - y(n-1) &= k_p\big(e(n) - e(n-1)\big) + k_p\left(\frac{1}{T_i}\sum_{i=0}^{n} e(i) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)\right) \\ &\quad + k_p\left(T_d\frac{\Delta e(n)}{T} - T_d\frac{\Delta e(n-1)}{T}\right) \end{aligned}$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + k_p\left(\frac{1}{T_i}\sum_{i=0}^{n-1} e(i) + \frac{1}{T_i}e(n) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i)\right) + k_p\frac{T_d}{T}\left(e(n) - 2e(n-1) + e(n-2)\right)$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + k_p\frac{1}{T_i}e(n) + k_p\frac{T_d}{T}\left(e(n) - 2e(n-1) + e(n-2)\right)$$

$$y(n) - y(n-1) = k_p e(n)\left(1 + \frac{1}{T_i} + \frac{T_d}{T}\right) - k_p e(n-1)\left(1 + 2\frac{T_d}{T}\right) + k_p\frac{T_d}{T}e(n-2)$$

Końcowa postać równania regulatora PID jest następująca:

$$y(n) = k_p e(n)\left(1 + \frac{1}{T_i} + \frac{T_d}{T}\right) - k_p e(n-1)\left(1 + 2\frac{T_d}{T}\right) + k_p\frac{T_d}{T}e(n-2) + y(n-1)$$

Równanie regulatora PID o rozdzielonych funkcjach jest następujące:

$$y(t) = k_p e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}$$

Równanie regulatora PID po dyskretyzacji:

$$y(n) = k_p e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i) + T_d\frac{\Delta e(n)}{T}$$

Równanie regulatora PID w wersji przyrostowej:

$$y(n) - y(n-1) = k_p e(n) + \frac{1}{T_i}\sum_{i=0}^{n} e(i) + T_d\frac{\Delta e(n)}{T} - k_p e(n-1) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i) - T_d\frac{\Delta e(n-1)}{T}$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + \frac{1}{T_i}\sum_{i=0}^{n} e(i) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i) + T_d\frac{\Delta e(n)}{T} - T_d\frac{\Delta e(n-1)}{T}$$

$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + \frac{1}{T_i}\sum_{i=0}^{n-1} e(i) + \frac{1}{T_i}e(n) - \frac{1}{T_i}\sum_{i=0}^{n-1} e(i) + \frac{T_d}{T}(e(n) - 2e(n-1) + e(n-2))$$

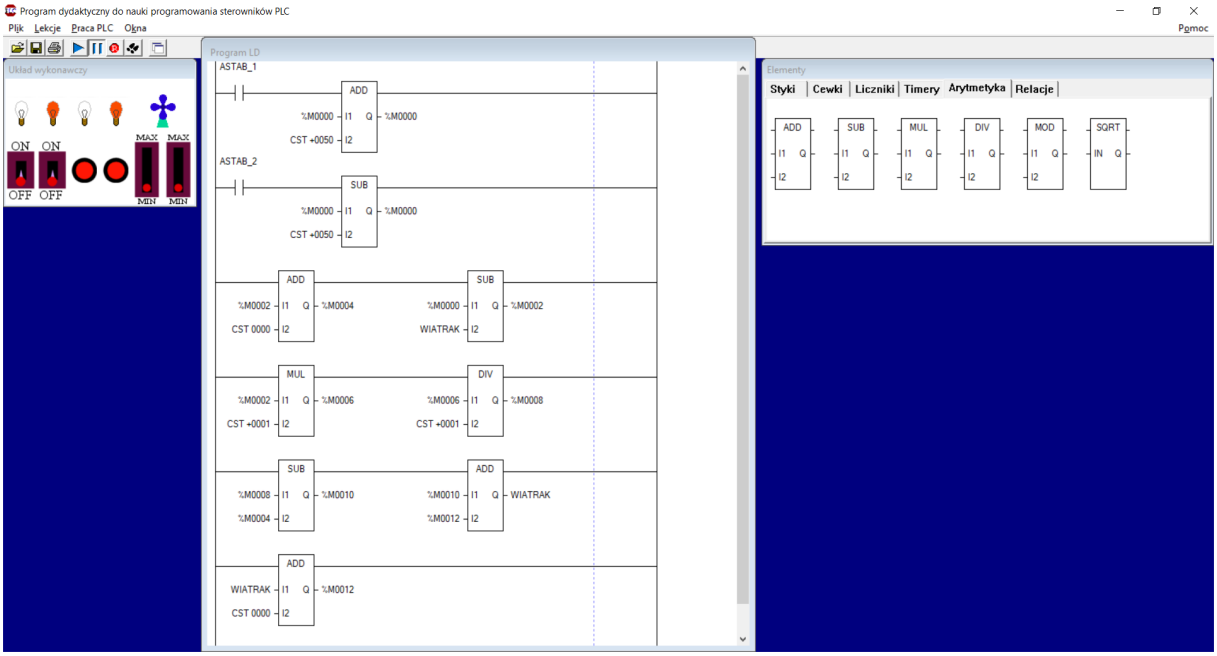
$$y(n) - y(n-1) = k_p\big(e(n) - e(n-1)\big) + \frac{1}{T_i}e(n) + \frac{T_d}{T}(e(n) - 2e(n-1) + e(n-2))$$

$$y(n) - y(n-1) = e(n)\left(k_p + \frac{1}{T_i} + \frac{T_d}{T}\right) - e(n-1)\left(k_p + 2\frac{T_d}{T}\right) + \frac{T_d}{T}e(n-2)$$

$$y(n) = e(n)\left(k_p + \frac{1}{T_i} + \frac{T_d}{T}\right) - e(n-1)\left(k_p + 2\frac{T_d}{T}\right) + \frac{T_d}{T}e(n-2) + y(n-1)$$

Przykład regulatora PID w języku drabinkowym

Przykład regulator PID o równaniu $y(n) = e(n) - e(n-1) + e(n-2) + y(n-1)$ przedstawiono na rys. 1.



Rys. 1 Przykład programowania regulatora PID w języku drabinkowym

Linia nr. 1

Wprowadzenie wartości zadanej i jej wzrost

Linia nr. 2

Zmniejszanie wartości zadanej

Linia nr. 3

Zapisanie uchybu regulacji e(n-1) do komórki pamięci M4 oraz obliczenie uchybu regulacji e(n) i zapisanie do komórki M2

Linia nr. 4

Wyliczenie działania k*e(n) o wartości ułamkowej wzmocnienia i zapisanie w komórce M8

Linia nr. 5

Odjęcie uchybu regulacji e(n-1) z komórki M4 oraz dodanie odpowiedzi z chwili poprzedniej y(n-1) i zapisanie odpowiedzi aktualnej do obiektu

Linia nr. 6

Zapisanie aktualnej odpowiedzi do komórki M12, która będzie odpowiedzią poprzednią y(n-1) w kolejnym przejściu programu.

7. Pytania kontrolne

a) Równanie i charakterystyki regulatora PID.
b) Dyskretyzacja równań różniczkowych.
c) Programowanie akcji całkującej i różniczkującej.
d) Sygnały wejścia i wyjścia analogowe sterownika PLC.
e) Programowanie uchybu regulacji i sprzężenia zwrotnego.
f) Przepełnienie i nasycenie