

KATEDRA TECHNIK WYTWARZANIA I AUTOMATYZACJI

Przedmiot:	Automatyzacja procesów obróbkowych	
Temat ćwiczenia:	T1: Programowanie sterowników PLC, język drabinkowy	Numer ćwiczenia: 1 lub 2
	T2: Programowanie sterowników PLC, język bloków logicznych	

1.1. Cel ćwiczenia.

Celem ćwiczenia jest zapoznanie się z zasadami programowania sterowników PLC z wykorzystaniem języka schematów drabinkowych LD (ladder diagram), będącego jednym ze standardowych narzędzi programowania sterowników PLC.

1.2. Wstęp teoretyczny.

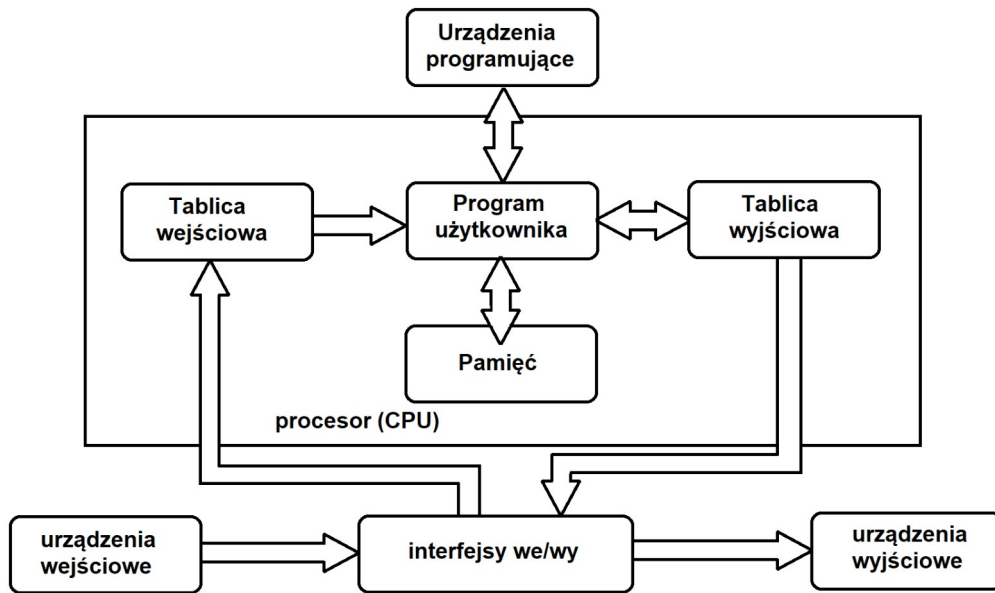
Programowalny sterownik logiczny, PLC (ang. programmable logic controller) – uniwersalne urządzenie mikroprocesorowe przeznaczone do sterowania pracą maszyny lub urządzenia technologicznego. PLC musi zostać dopasowany do określonego obiektu sterowania poprzez wprowadzenie do jego pamięci żądanego algorytmu działania obiektu.

Sterownik PLC zbudowany jest z:

- jednostki centralnej CPU,
- pamięci ROM, PROM, EPROM, EEPROM
- bloków wejść cyfrowych
- bloków wejść analogowych
- bloków komunikacyjnych
- bloków wyjść cyfrowych
- bloków wyjść analogowych
- bloków specjalnych

Sterownik wykonuje program cyklicznie, tzn. wykonuje po kolei sekwencję zapisaną w programie. Na początku cyklu skanowania sprawdza stany na wejściach, wykonuje sekwencję programu, a następnie w zależności od jego struktury zmienia odpowiednio

stany na wyjściach, czyli najprościej mówiąc steruje urządzenia wykonawczymi. Na rys. 1. przedstawiono schemat blokowy PLC.

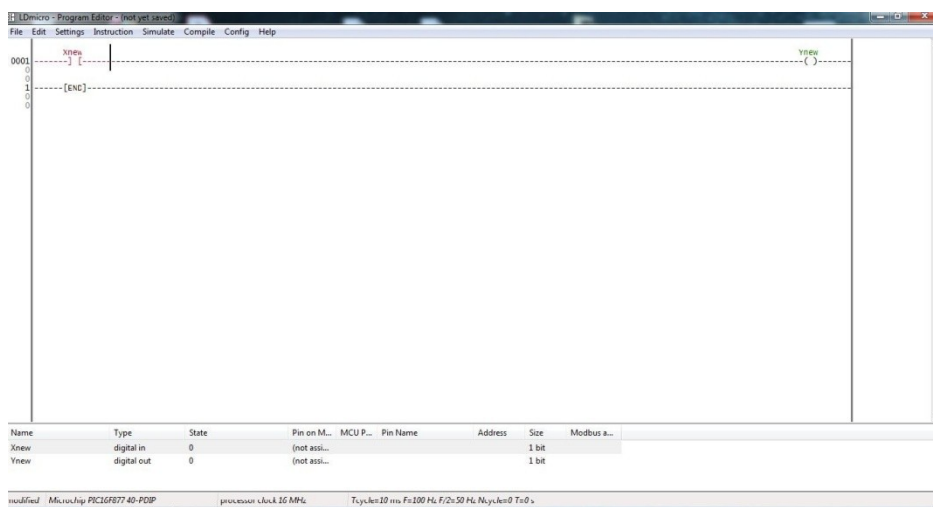


Rys. 1. Schemat blokowy sterownika PLC.

1.3. Opis opracowanego programu sterującego w języku drabinkowym wraz z komentarzami.

Programowanie sterowników odbywa się za pomocą różnych języków programowania. W tym ćwiczeniu wykorzystano język LD (Ladder Diagram), jest to schemat zbliżony do klasycznego rysunku technicznego elektrycznego. Program sterujący w języku drabinkowym wykonano w programie LDmicro. Pozwala on napisać i kompilować programy dedykowane dla popularnych mikrokontrolerów serii ATmega, PIC16 i innych. Okno główne przedstawiono na rysunku 2.

Rys. 2.

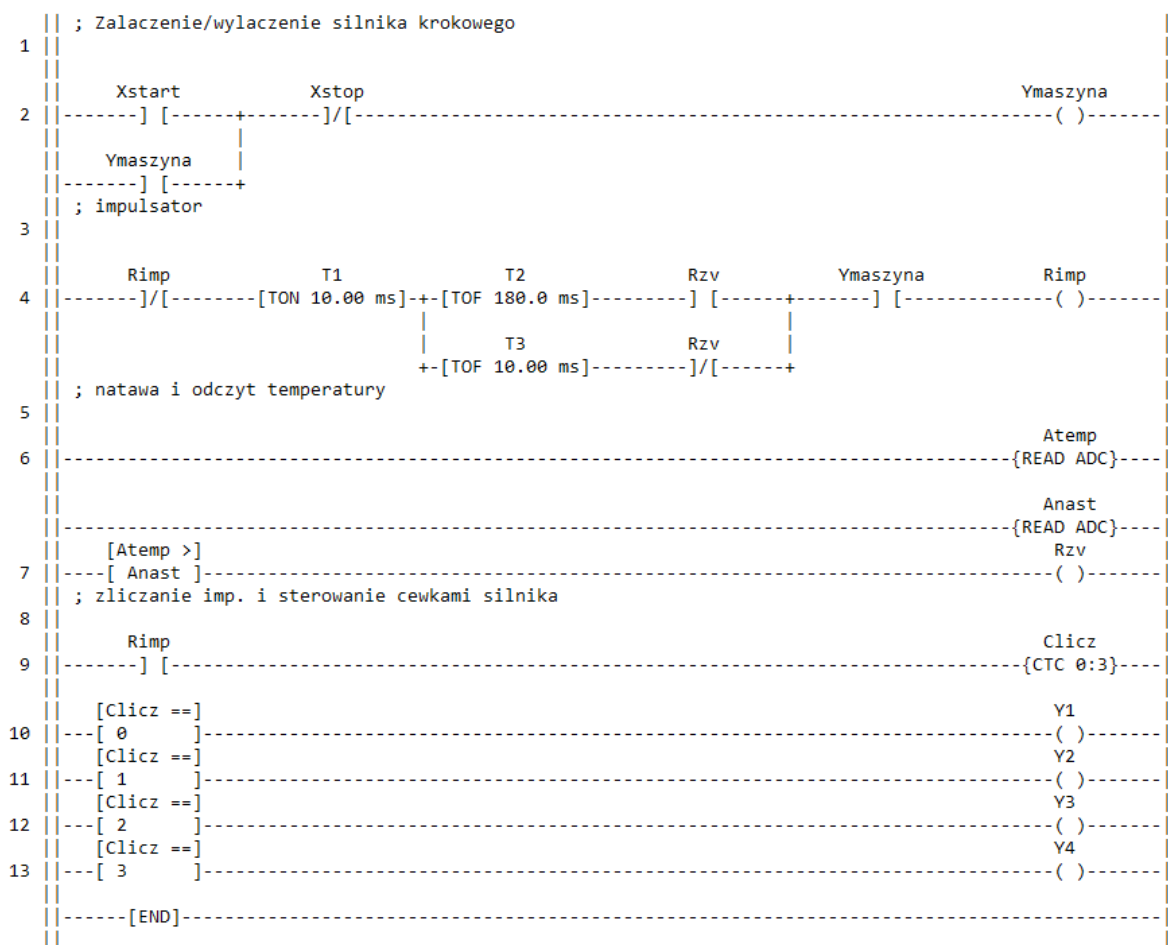


Okno

programu LDmicro.

Tekst programu:

LADDER DIAGRAM:



I/O ASSIGNMENT:

Name	Type	Pin
Xstart	digital in	1
Xstop	digital in	3
Ymaszyna	digital out	10
Y1	digital out	(not assigned)
Y2	digital out	(not assigned)
Y3	digital out	(not assigned)
Y4	digital out	(not assigned)
Anast	adc input	(not assigned)
Atemp	adc input	(not assigned)
Rimp	int. relay	
Rzv	int. relay	
T1	turn-on delay	
T2	turn-off delay	
T3	turn-off delay	
Clicz	counter	

Komentarz:

- Pierwszy sześciel drabinka odpowiada za start/stop obrabiarzki. Skonystano z funkcji contacts, gdzie w zmiennych Source można wybrać stan sygnału: „X” oznacza wejście, „R” oznacza stan sygnału z wewnętrznej komórki pomiaru, „Y” oznacza stan wyjścia. Funkcja symuluje pracę tutaj przycisku włączenia i wyłączenia maszyny.
- W sześcielu drugim wykorzystano wykł wewnętrzną funkcję oraz funkcje TON i TOF, które opóźniają załączenie i wyłączenie. Ten sześciel odpowiada za zmianę prędkości obrotowej silnika ^{SP} kółkowego.
- Ten sześciel odpowiada za pomiar napięcia. Od tego momentu zaczyna się nastawa i służy temperatury. Sygnał przechodzi jeżeli jest większy od wartości nastawionej.
- Kolejny blok to wykorzystanie funkcji Circular Counter. Jej działanie polega na zmianie do wartości nastawionej; zerowaniu i zliczeniu od początku.
- Następną decyzją to sprawdzenie, czy wartości sygnału z Circular Counter odpowiadają przypisanym w kolejnych sześcielach. Jeżeli tak, poprzez funkcję Coil wystawiamy sygnał, który powoduje następną rozłączenie maszyny.

1.4. Wnioski.

Celem ćwiczenia było zapoznanie się z zasadami programowania sterowników PLC z wykorzystaniem języka - schematów drabinkowych LD (ladder diagram).
Korzystając z programu LDmuw wykonałem znane komendy występujące w tym języku. Wykorzystanie polega na odpowiednim łąceniu bloków, dzięki ich poprawnemu utworzeniu program może wykonywać skomplikowane operacje, sterować zadanymi programami.
Budowa programu sugeruje, że nie jest on przeznaczony bezpośrednio dla programistów, potrzebne funkcje wyglądem przypominają bardziej schematy obwodów.
Symulacja w czasie rzeczywistym umożliwia szybsze testowanie poprawności utworzonego programu, może on służyć do sterowania np. windą w budynku, czego dowiedzieliśmy na co dzień.

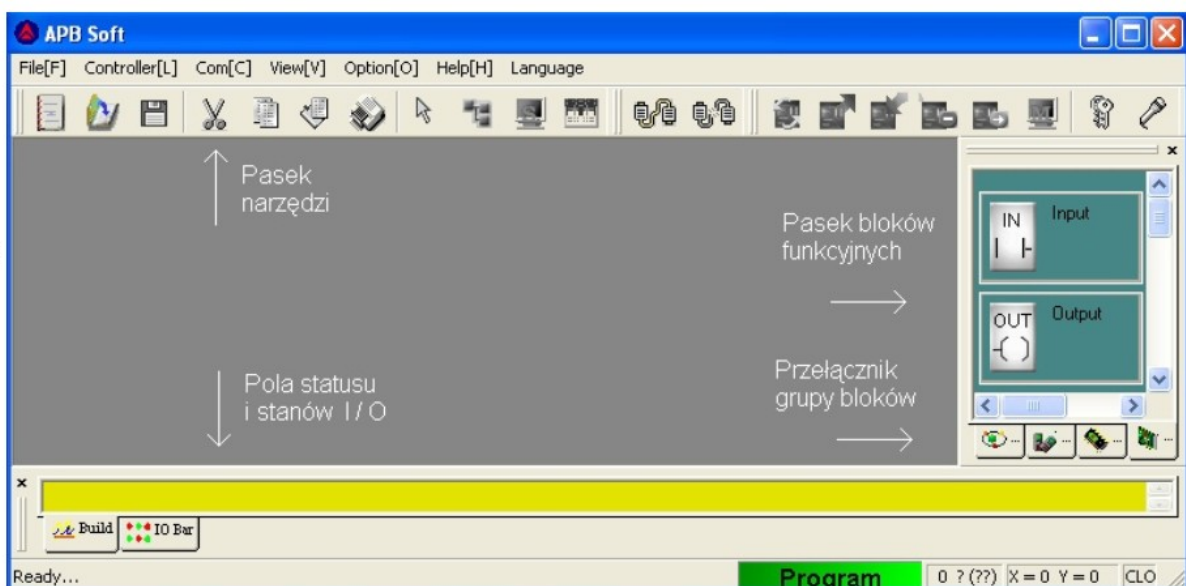
2.1. Cel ćwiczenia.

Celem ćwiczenia jest zapoznanie się z zasadami programowania sterowników PLC z wykorzystaniem języka bloków logicznych, będącego jednym ze standardowych narzędzi programowania sterowników PLC.

2.2. Wstęp teoretyczny.

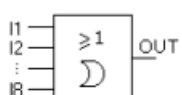
Sterowniki serii APB stanowią uniwersalne urządzenia programowalne, tj. takie, w których sam instalator określa sposób ich działania. Do programowania sterowników APB wykorzystuje się zestaw bloków funkcyjnych o zaawansowanych możliwościach, łączonych w schemat (diagram). Rysowanie diagramu FBD, symulowanie działania, ustawianie parametrów i w końcu przesyłanie kodu do sterownika wykonywane jest za pomocą bezpłatnego programu komputerowego APB Soft.

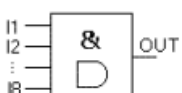
APB software program ten jest darmowym środowiskiem przeznaczonym do tworzenia programów sterujących blokowo. Pozwala on napisać i kompilować programy dedykowane dla popularnych sterowników serii APB. Na rys. 3. przedstawiono widok programu po uruchomieniu.

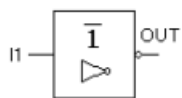


Rys. 3. Widok środowiska APB Software.

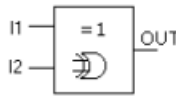
Podstawowe bloki funkcyjne w programie APB software:

 Alternatywa, tj. suma logiczna, która przepuszcza sygnał tylko wtedy, gdy co najmniej jeden argument jest jedyneką logiczną.

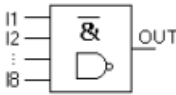
 Koniunkcja, tj. iloczyn logiczny, który przepuszcza sygnał tylko wtedy, gdy oba argumenty mają wartość logiczną 1.



Negacja, zamienia sygnał wejściowy na przeciwny na wyjściu.

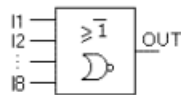


XOR, sygnał na wyjściu ma wartość logiczną 1 tylko wtedy, gdy wejściowe mają różną wartość logiczną.



NAND, tj. negacja koniunkcji, daje na wyjściu wartość 1, jeżeli chociaż jedna wartość wejściowa jest równa 0.

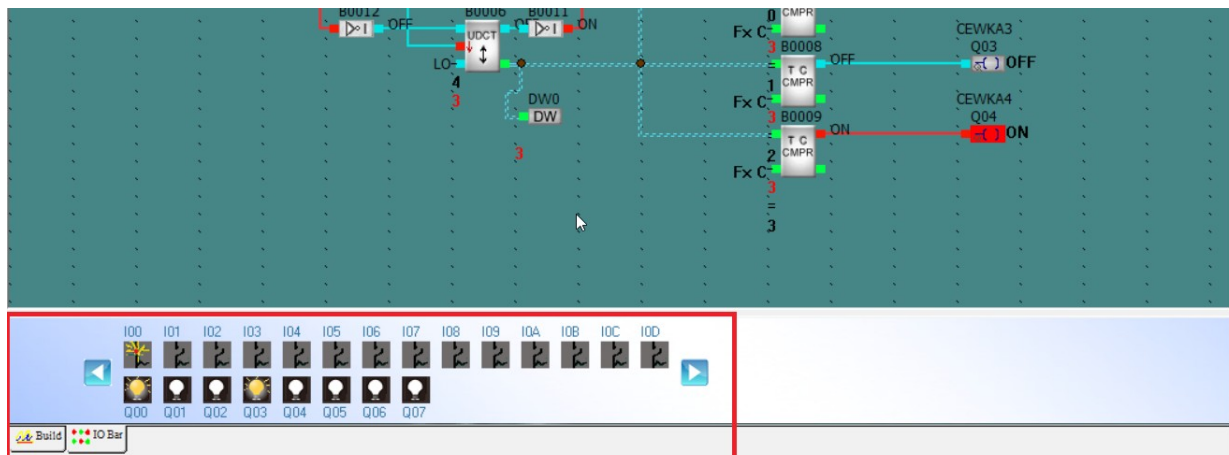
NOR, tj, chociaż jedna wartość



negacja alternatywy, daje na wyjściu wartość 0, jeżeli wejściowa jest równa 1.

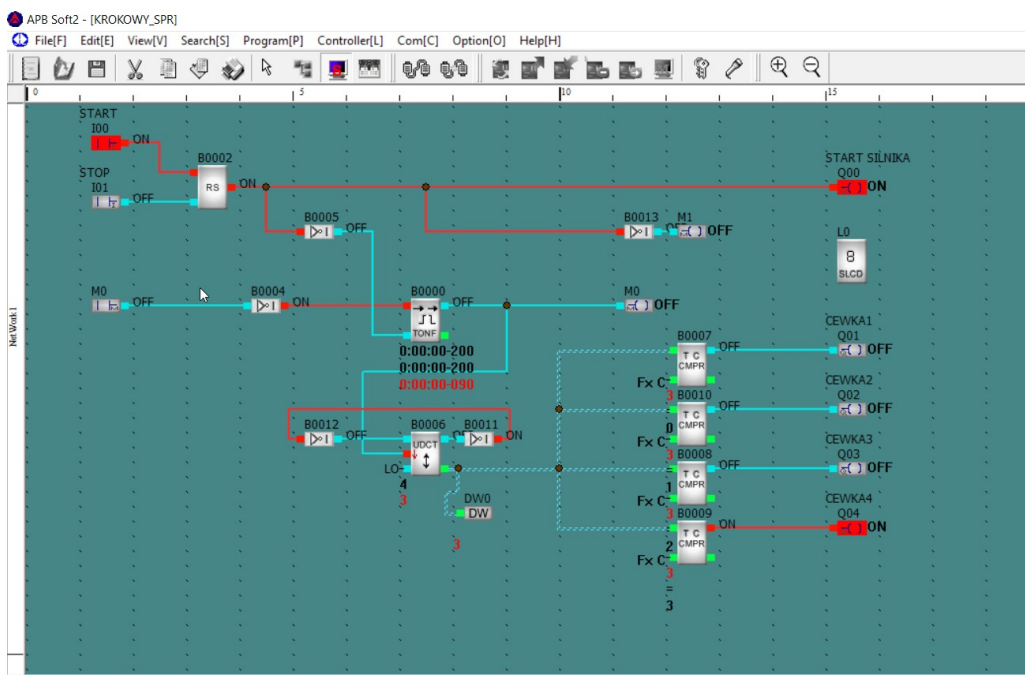
2.3. Opis opracowanego programu sterującego w języku bloków logicznych wraz z komentarzami.

Funkcjonowanie programu sprawdzamy za pomocą zakładki Program ->Simulation -> Start. W kolorze czerwonym wyświetlane są aktywne połączenia, tj. stan 1. Stany sygnałów wejściowych i wyjściowych są także wyświetlane w zakładce IO Bar (rys. 4). Na rysunku 5. przedstawiono natomiast widok w czasie symulacji programu.

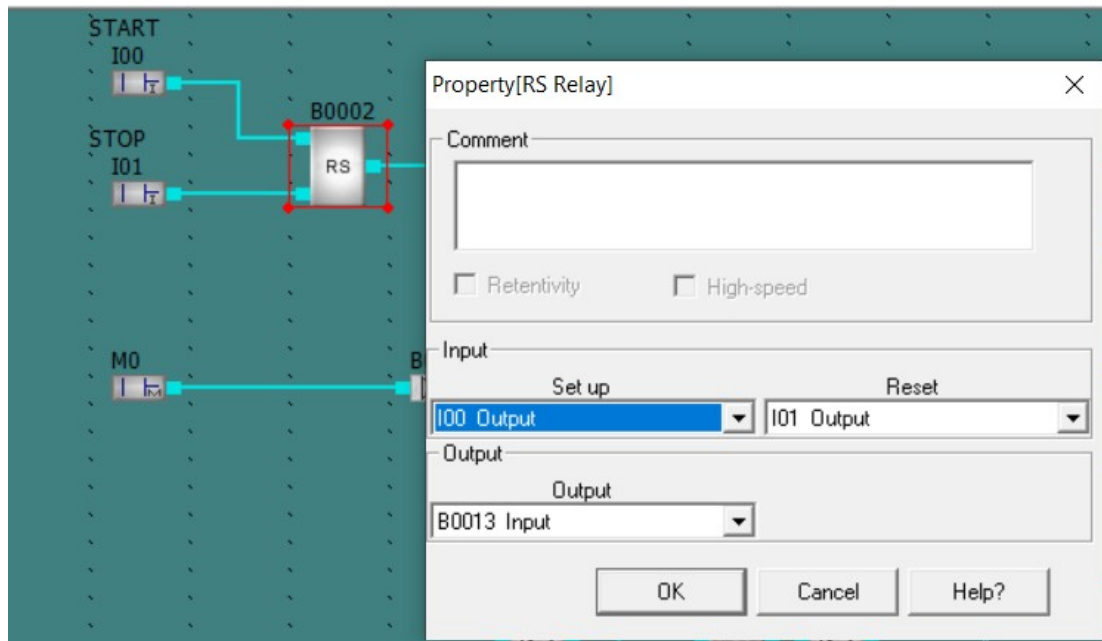


Rys. 4. Stany sygnałów wejściowych i wyjściowych.

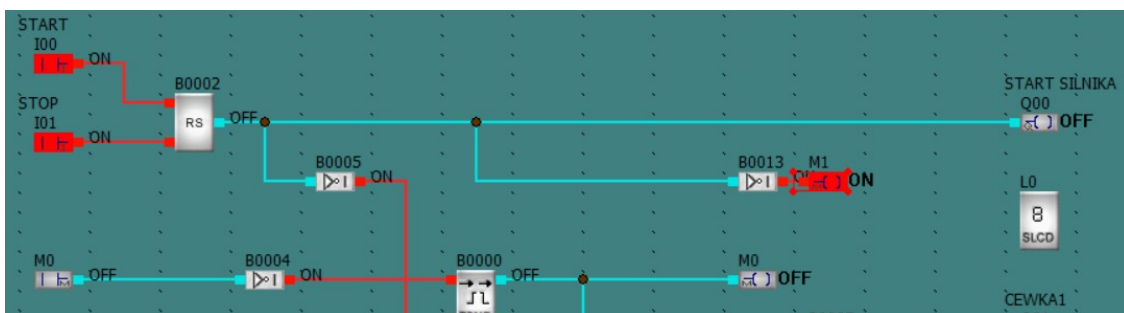
Program odpowiada za sterowanie cewkami 1 do 4, start silnika sygnał startowy ma oznaczenie 100, natomiast sygnał stopujący to 101. Analizując schemat od góry, na początku zastosowano przerzutnik RS, tj. Reset set (zemy, ustaw). Pokazano go na rys. 6. Na wyjściu z tego przerzutnika ustawiony jest sygnał B0013. Silnik jest włączony dopóki na wyjściu 101 nie pojawi się wartość logiczna 1. W przypadku pracy silnika wyjście Q00 silnika nie ma uzerwono. Wyłączenie silnika, czyli doprowadzenie sygnału 101 o wartości 1 powoduje negację sygnału na dojściu do M01 (przez B0013) i zasilenie nie M1. Sytuację tą przedstawiono na rysunku 7.



Rys. 5. Widok z symulacji programu wykonanego w APB Soft2.

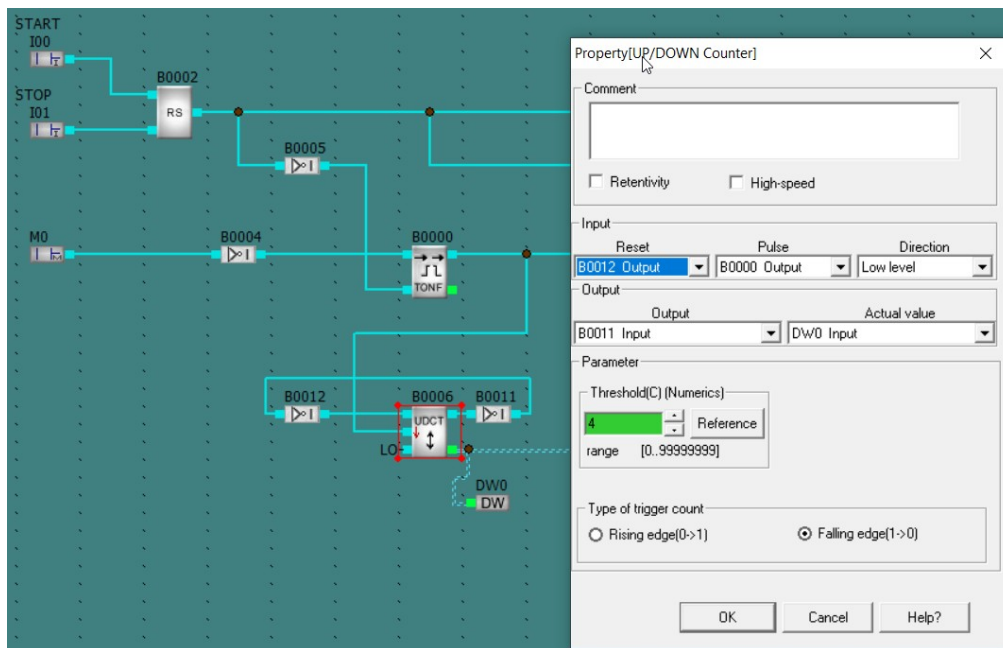


Rys. 6. Przerzutnik RS.



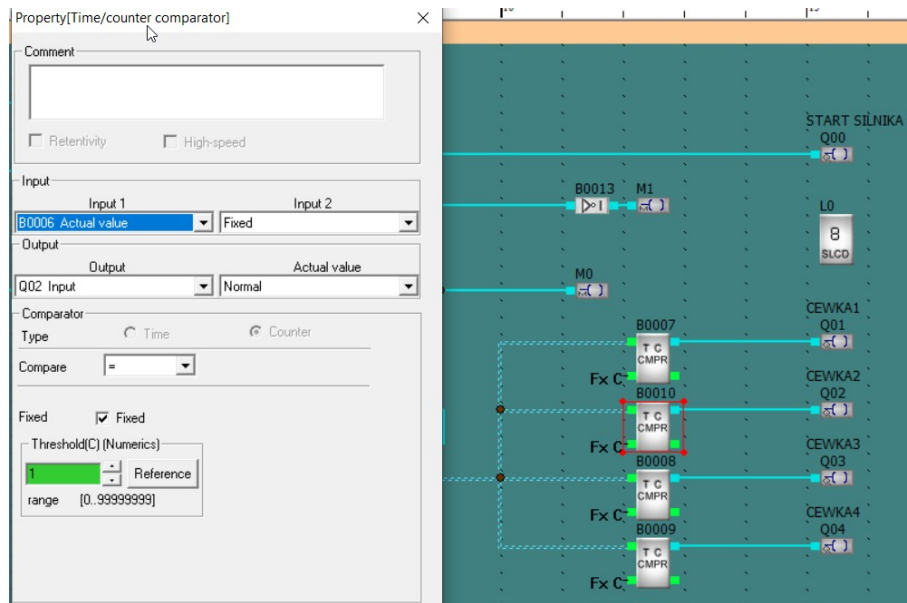
Rys. 7. Widok fragmentu programu w czasie zatrzymania pracy.

Sygnal z przerzutnika trafia kolejno do bloku TONF (ON/OFF Delay), który odpowiada za opóźnione włączenia i wyłączenia. Posiada on 2 wejścia (TRG, RES) oraz 2 wyjścia. Wejście TRG odpowiada za opóźnione włączenia i wyłączenia, sygnał stały narastającego (0-1 na TRG) o czas TON i spadającego (1-0 na TRG) o czas TOFF. Po zmianie w TRG od strony B002 ma to sygnał wyjściowy zmienia się na 1 dopiero po zadany opóźnieniu (TON). Analogiczny sygnał na wyjściu będzie miał wartość 0 dopiero po opóźnieniu 1-0 na TRG. Sygnał pochodzący od B005 i konfigury się do RES ma większy priorytet od TRG. Oznacza to, że niezależnie od TRG sygnał podany na RES zresetuje zegar i ustawi wyjście 0 na wartość 0. Na drugim wyjściu z bloku TONF można zamieścić informację o aktualnym stanie sygnału (rys. 8)



Rys. 9. Ustawienia licznika UP/DOWN Counter.

Poza rejestrem DW sygnał z linka jest przekonywany do komparatora czasu T/C-CMPR. Blok ten pozwala porównywać czas dwóch bloków czasowych albo stanów dwóch kontaktów. Umozliwia też porównywanie z wartością stałą (wpisaną lub pobieraną). Dwa wejścia muszą być połączone do tego samego typu bloków, w przypadku tego programu jedno z wejść jest zablokowane poprzez Fixed. Wynika to z tego, że nie porównujemy pierwszego sygnału wejściowego z innym. Wejścia 1 oraz 2 mogą być połączone do bloku czasowego albo linka (tak jak w tym programie). Numerem porównania wykorzystanym w bloku jest warunek równania. Kiedy z bloków komparatora ma ustawioną wartość 0-3, odpowiadająca kolejno numerom cewek 1-4 (rys 10). Sygnał jest następnie kierowany na wyjścia f. cewki tylko po spełnieniu warunku (wartość na wejściu równa ustawionej).



Rys. 10. Ustawienia time/counter komparator.

2.4. Wnioski.

Celem ćwiczenia było zapoznanie się z zasadami programowania sterowników PLC z wykorzystaniem języka bloków logicznych. Na podstawie laboratorium można stwierdzić, że język ten jest bardziej przystępny dla osób mających styczność z układami cyfrowymi, ponieważ do języka LD, który przypomina schemat elektryczny. Bezpośrednie połączenia między kolejnymi blokami w interfejsie programu AFB sprawiają, że program ten jest bardziej przejrzysty. Dodatkowo w schemacie blokowym mogą pojawiać się sprzężenia zwrotne, gdy sygnał wyjściowy ma odzwierciedlenie do bloku występującego na wcześniejszym etapie schematu. Oba z tych języków programowania PLC należą natomiast do grupy języków graficznych, co prowadzi na pewno do łatwiejszej wizualizacji wykonanych schematów na poziomie programu.