Ćwiczenie 5

Realizacja prostych algorytmów sterowania z wykorzystaniem programu CONCEPT

1. Wprowadzenie

Wraz z wykorzystaniem komputerów typu IBM PC do programowania sterowników PLC oraz do sporządzenia dokumentacji tego oprogramowania znacznie rozwinęły się możliwości programowe i komunikacyjne sterowników pracujących niejednokrotnie w rozwiniętych sieciowych systemach komputerowych. Przyczyną był m. in. dynamiczny rozwój oprogramowania komputerów PC, a w szczególności oprogramowania do sterowania nadrzędnego i zbierania danych **SCADA** (*Supervisory Control and Data Acquisition*) przeznaczonego dla nadrzędnej warstwy sterowania.

Systemy **SCADA** dopełniają i rozszerzają możliwości sterowników, realizując m. in. następujące funkcje:

- zbierania i przetwarzania oraz archiwizacji danych pochodzących bezpośrednio z systemów sterownikowych
- opracowania raportów dotyczących bieżącego stanu, zużycia materiałów oraz stanu pracy maszyn i urządzeń
- wizualizacji w wielu formach graficznych wartości zmiennych procesowych (aktualnych i historycznych)
- generowania sygnałów alarmowych związanych z przekroczeniem wartości granicznych
- wypracowywania danych dla warstw sterowania operatywnego produkcją i warstwy zarządzania

Systemy **SCADA** zapewniły niezawodną komunikację ze sprzętem PLC, możliwość zmian jego oprogramowania w działających systemach sieciowych i dzięki temu zakres zastosowań i możliwości systemów sterownikowych wzrosły wielokrotnie.

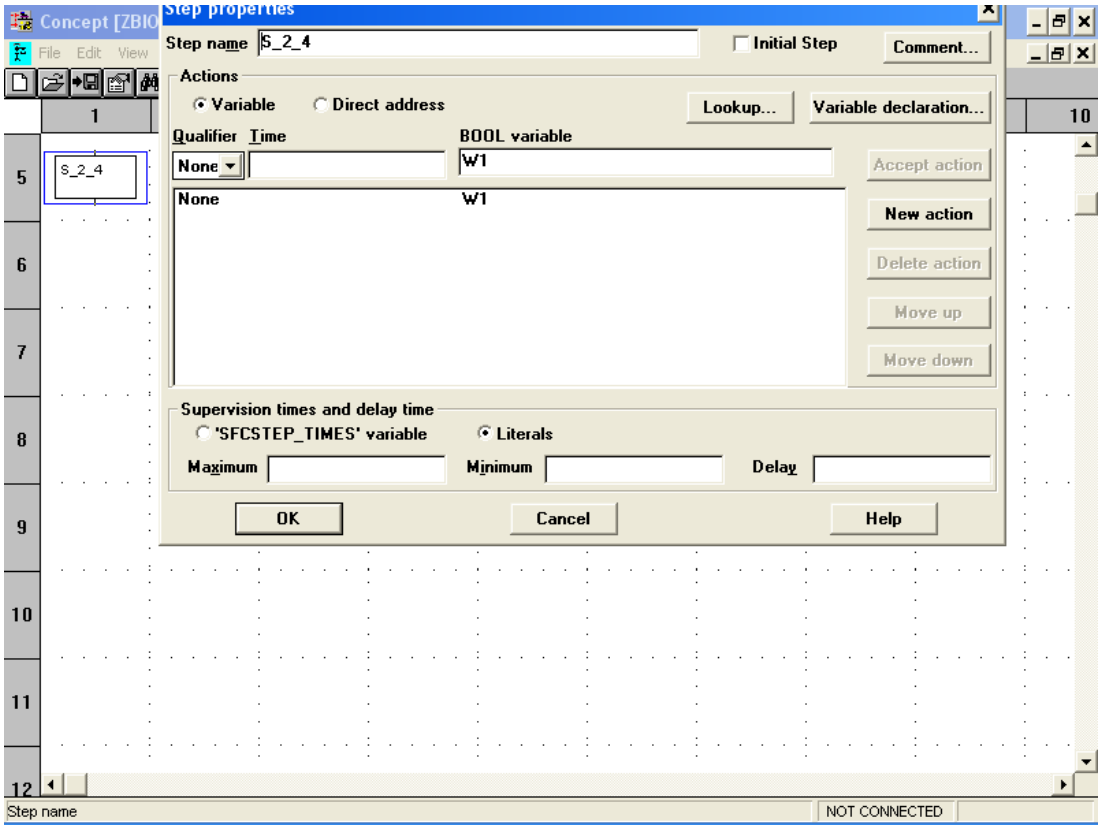
Język drabinkowy

Język drabinkowy jest językiem graficznym, którego zasady wywodzą się ze sposobu interpretacji schematów obwodów przekaźnikowo-stycznikowych.

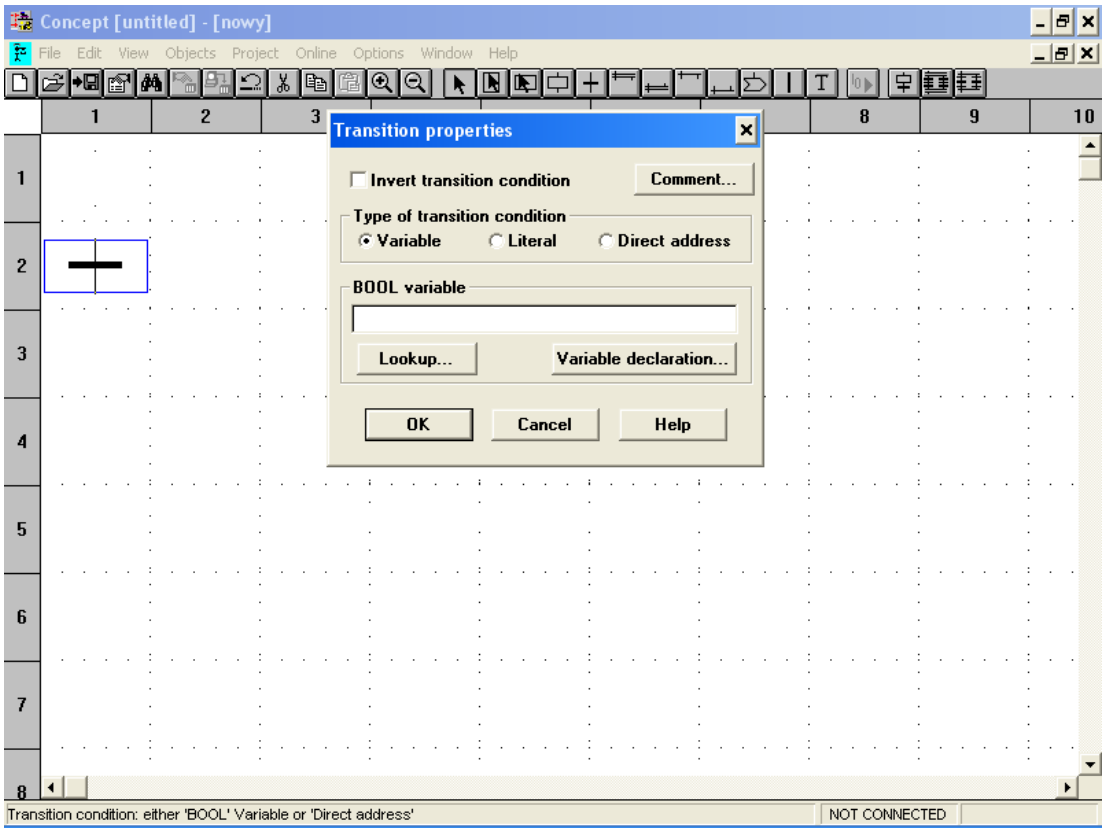
Algorytm interpretuje się jako przekazywanie „zasilania” poprzez zamknięte „styki” lub mostki do „cewek” i/lub bloków funkcyjnych (szyna „masy” nie jest w tej odmianie języka zaznaczana) w danym szczeblu drabinki algorytmu.

SFC (Sequential Function Chart)

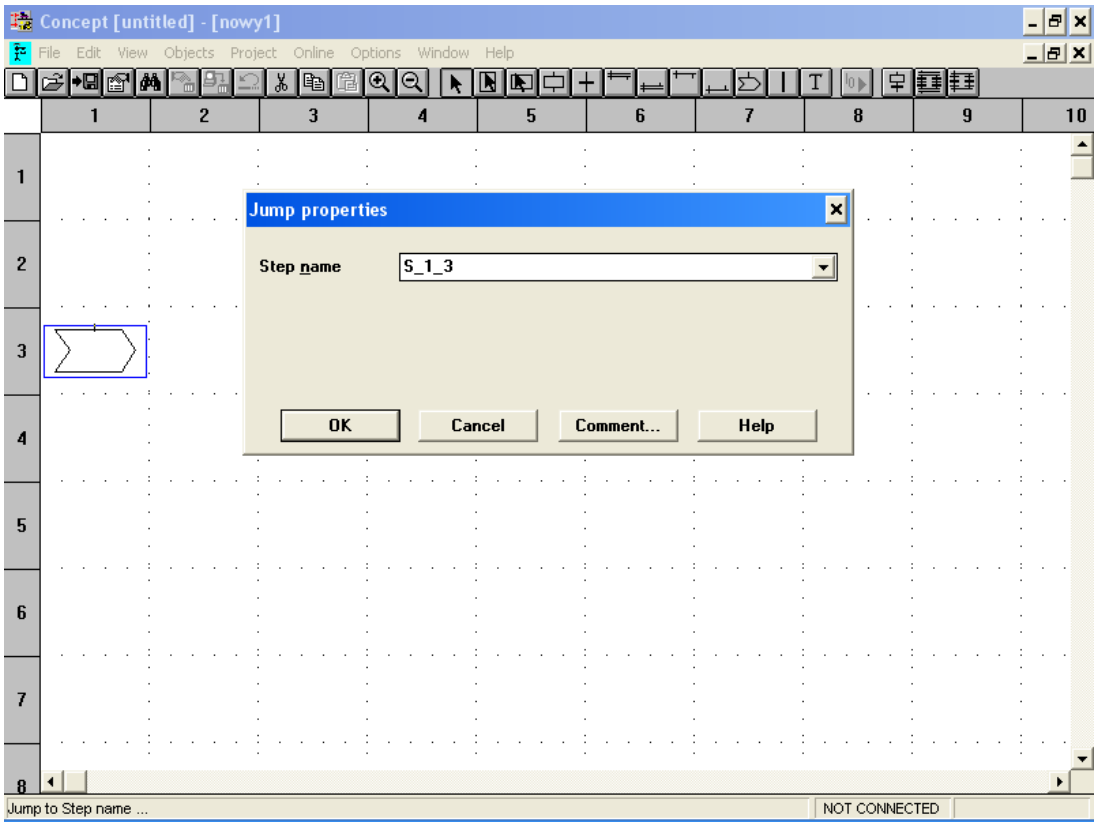
Przy programowaniu w oparciu o język sekwencyjny należy podzielić proces na dobrze zdefiniowane etapy (sekwencje) połączone przejściami (tranzycjami). Język ten jest rdzeniem standardu IEC61131-3. Znakomicie nadaje się do opisu procesów zachodzących równolegle. SFC składa się z wzajemnie sprzężonych etapów (steps) i przejść (transitions). Z każdym etapem jest skojarzony zbiór odpowiednich działań (actions) a każdemu przejściu między krokami towarzyszy warunek przejścia (transition condition). Po zadeklarowaniu zmiennych możemy przystąpić do pisania programu.



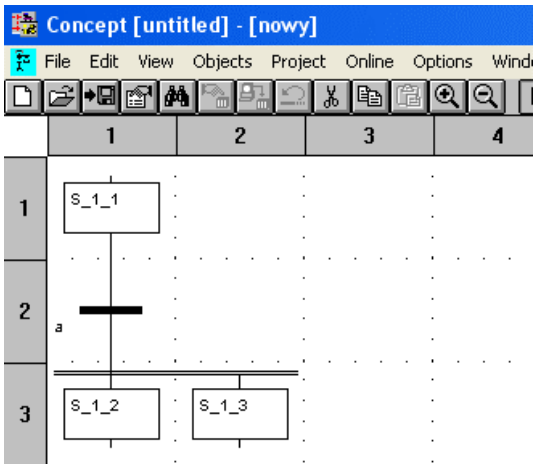
Dwukrotne kliknięcie na bloku etapu otwiera okno do kojarzenia z etapem wybranych akcji. Np. W celu skojarzenia z krokiem S_2_4 akcji W1 wciskamy Variable declaration… >wybieramy z wcześniej zadeklarowanych W1>zatwierdzamy OK. Wciskamy New action, zatwierdzamy OK. Dodawania i usuwanie poszczególnych akcji wykonuje się przy pomocy opcji New actoin, Delete action itd.



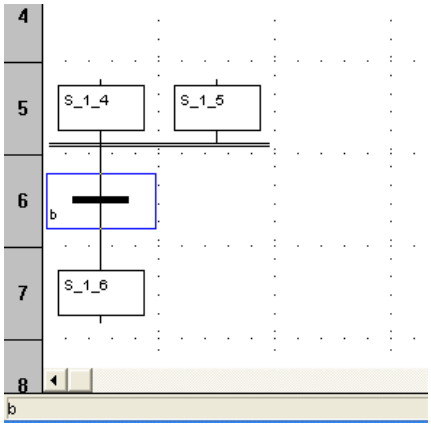
Dwukrotne kliknięcie na bloku przejścia otwiera okno własności przejścia. Wybieramy odpowiednio: zmienną zadeklarowaną, stałą znakową lub konkretny adres w pamięci sterownika.



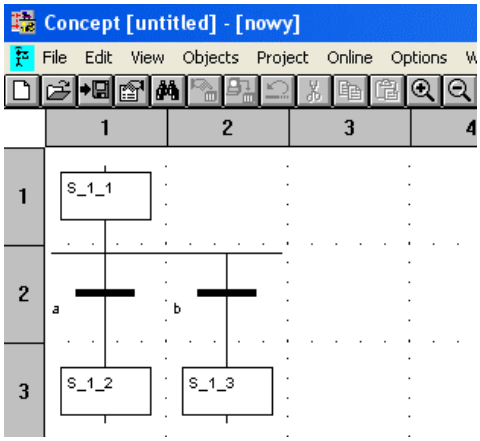
Blok skoku (jump) wykonuje funkcję przejścia do wybranego etapu. Etap wybieramy klikając dwukrotnie na bloku i wpisując nazwę wybranego etapu. Np. S_1_3



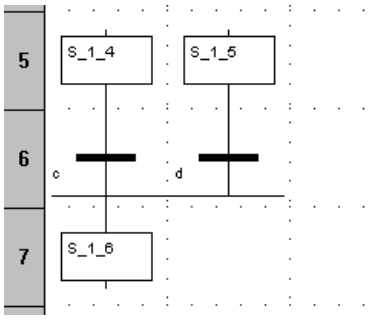
Początek sekwencji współbieżnych
Przejście z etapu S11 do etapów S12, S13, …następuje tylko wtedy, gdy S11 jest aktywny i jest spełniony wspólny warunek a. Po jednoczesnej aktywacji S12, S!13, … realizacja każdej sekwencji jest niezależna.



Zakończenie sekwencji współbieżnych
Przejście z etapów S14, S15, …do etapu S16 następuje tyko wtedy gdy wszystkie etapy dołączone do linii są aktywne i jest spełniony warunek b.



Realizacja wyboru sekwencji
Przejście z etapu S11 do S12 jest realizowane tylko wtedy, gdy S11 jest aktywny i jest spełniony warunek a, a przejście z S11 do S13 wtedy gdy S11 jest aktywny i jest spełniony warunek b a nie jest spełniony warunek a.



Zakończenie wyboru sekwencji
Przejście do etapu S16 jest możliwe wtedy, gdy etap S14 jest aktywny i jest spełniony warunek c lub gdy S15 jest aktywny i jest spełniony warunek d.

Program Concept

Jako narzędzie do programowania i uruchamiania programów w sterownikach PLC może służyć pakiet programowy Concept firmy Schneider Automation GmbH. Pakiet ten umożliwia programowanie sterowników Schneider–Modicon z rodzin: Quantum, TSX Compact, Momentum i Atrium. Podstawowe zalety tego narzędzia to:

- zasady programowania sterowników są zgodne z zaleceniami normy IEC 61131-3.
- pakiet jest łatwy w użytkowaniu – wykorzystano w nim wiele udogodnień dostępnych w aplikacjach pracujących w środowisku Windows. Wyposażony jest także w bardzo obszerny system pomocy (*Help*).
- korzystając z pakietu Concept można testować napisane programy bez konieczności posiadania fizycznego sterownika – pakiet został wyposażony w znakomity symulator sterowników wymienionych rodzin, który jest osobną aplikacją systemu Windows (sterownik wirtualny).

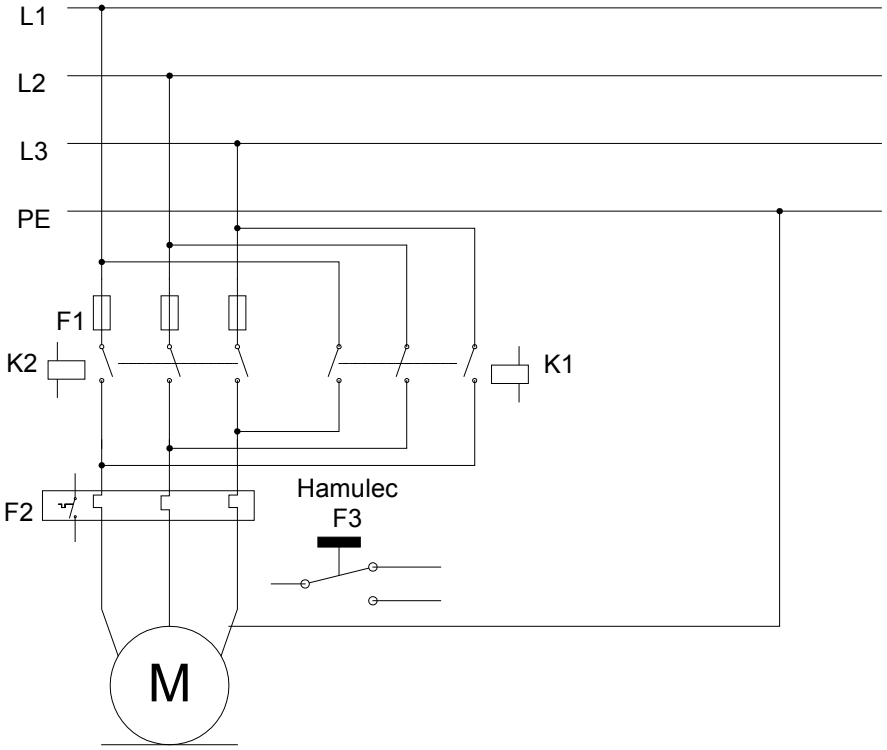
Pakiet *Concept* tworzy we współpracy z systemem *Windows 95, 98, 2000, NT, XP* środowisko programowe, które udostępnia sześć języków programowania przeznaczonych do tworzenia programów użytkownika dla sterowników programowalnych. Językami tymi są zdefiniowane w normie IEC 61131-3: *FBD*, *LD*, *IL*, *ST*, *SFC* oraz dodatkowo, dla zachowania ciągłości z oprogramowaniem *Modsoft*, jezyk *984 LL*, odpowiadający standardowi języka programowania *Modicon 984 Ladder Logic*.
Dla każdego z wymienionych języków programowania istnieje odpowiedni edytor jezyka. Ponadto, niezależnie od wybranego języka, Concept zawiera:

- edytor typów danych (ang. *Data type editor*),
- edytor zmiennych (ang. *Variables editor*),
- edytor informacji o danych (ang. *Reference data editor*).

2. Program ćwiczenia

Sterowanie silnikiem asynchronicznym

Obwód sterowania trójfazowym silnikiem asynchronicznym zasilanym z sieci o napięciu 380V przedstawia rys. 1

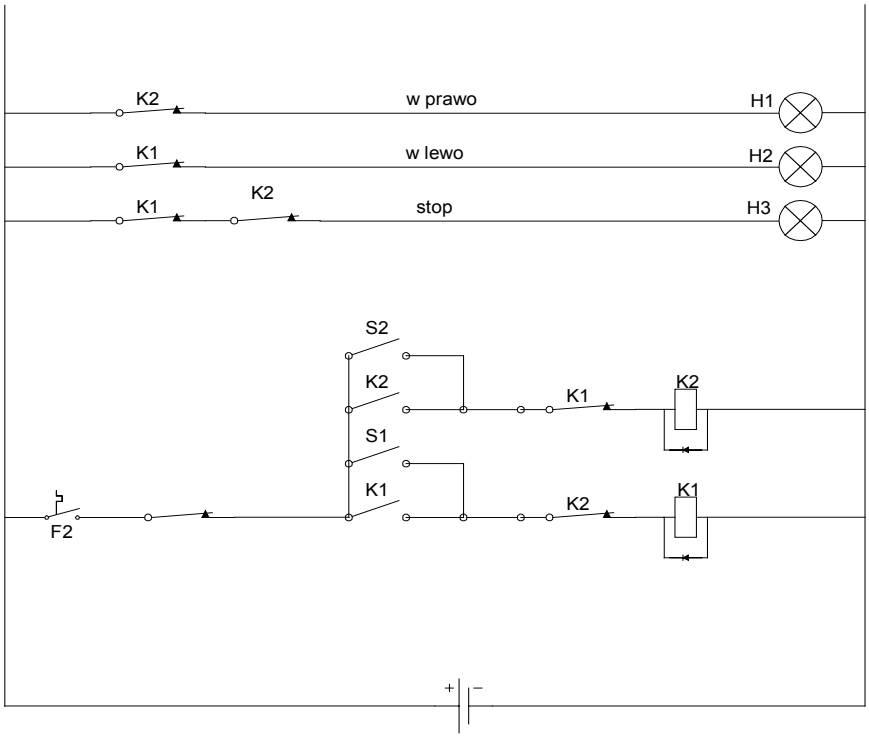


Rys. 1. Schemat obwodu mocy trójfazowego silnika prądu przemiennego

Silnik jest połączony ze źródłem napięcia przez styki przekaźników K1 albo K2, w zależności od wybranego przez przyciski sterownicze kierunku wirowania: S1 (w prawo) i S2 (w lewo), zaś jego wyłączenie następuje za pomocą przycisku wyłączającego S0 (stop). Załóżmy, że jest to silnik jednobiegowy z jednopoziomową ochroną termiczną. Ochrona ta działa pod wpływem nagrzania paska bimetalowego działającego na styk wyłączający F2. Styk ten umieszczony jest w obwodzie sterowania i zapewnia termiczne zabezpieczenie silnika. Rys. 2 przedstawia schemat stykowy układu sterowania takim silnikiem dla oznaczeń podanych w tabeli 1.

Oznaczenie	Opis	Oznaczenie wejść i wyjść	Liczba styków	
			rozwiernych	zwiernych
S0	Przycisk sterowniczy (stop)	%I3	1	
S1	Przycisk sterowniczy zał. kierunek prawo	%I1		1
S2	Przycisk sterowniczy zał. kierunek lewo	%I2		1
F1	Bezpieczniki topikowe			
F2	Styk wyłączający zabez. termicznego	%I4	1	
F3	Przełączny styk przerwowy hamulca	%I4	1	1
K1	Cewka i styki przekaźnika (prawo)	%Q1	2	2
K2	Cewka i styki przekaźnika (lewo)	%Q2	2	2
H1	Lampka sygnalizacji wirowania w prawo	%Q11		
H2	Lampka sygnalizacji wirowania w lewo	%Q12		
H3	Lampka sygnalizacji stopu	%Q13		

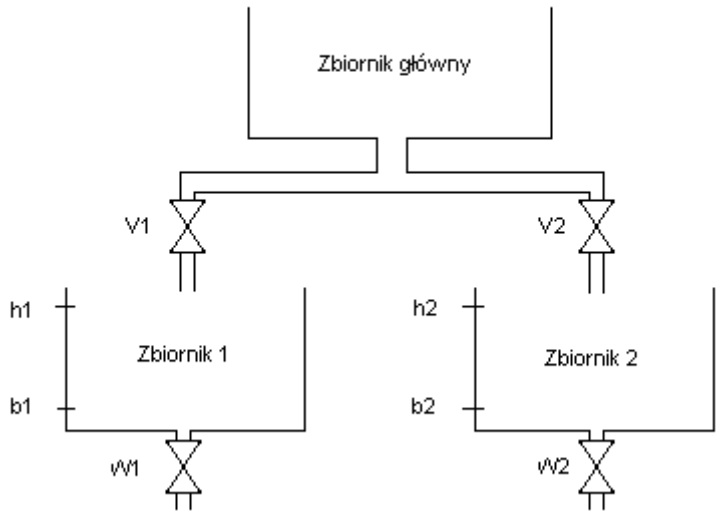
Tabela 1. Zestawienie oznaczeń do schematu sterowania



Rys. 2. Schemat stykowy obwodu sterowania

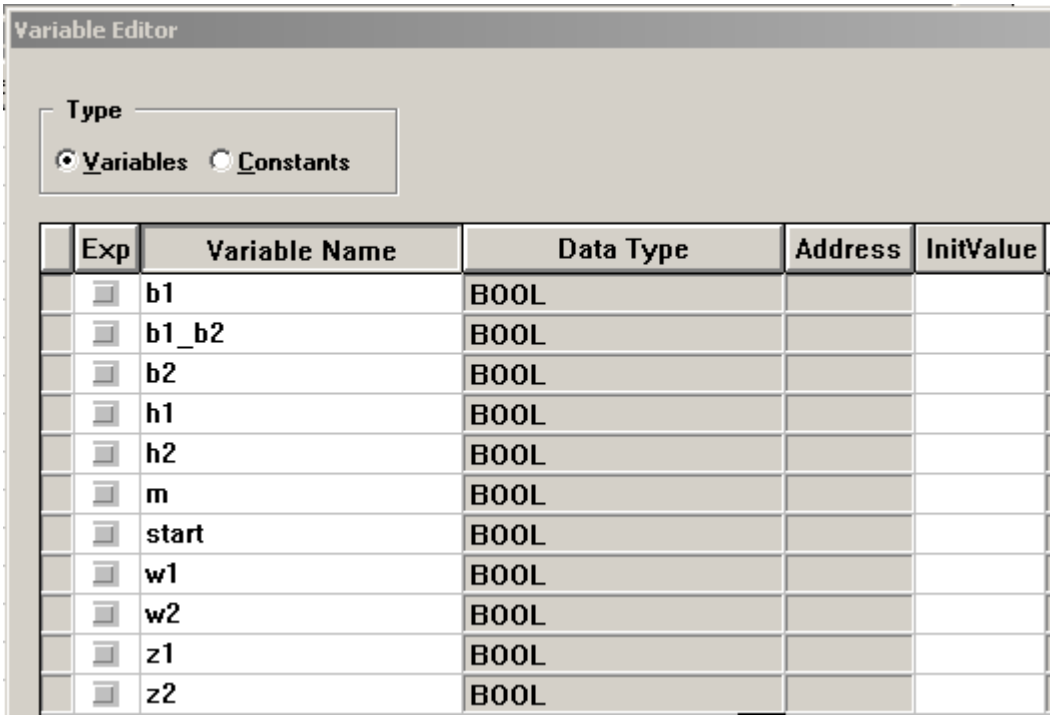
Na podstawie tabeli 1 oraz schematu stykowego (rys. 2) napisać program w języku drabinkowym realizujący wyżej opisany algorytm sterowania. Przeprowadzić animację algorytmu.

Sterowanie napełnianiem dwóch zbiorników



Przyjmuje się że zbiornik jest pusty jeżeli poziom płynu w zbiorniku znajduje się poniżej poziomu b, odpowiednio b1 w Zbiorniku 1 i b2 w Zbiorniku 2, co odpowiada wartościom b1=0 lub b2=0. Natomiast zbiornik jest pełny gdy poziom znajduje się powyżej czujnika h, czyli h1=1 lub h2=1. Zakłada się, że w stanie początkowym oba zbiorniki są puste. Jeśli zostanie naciśnięty przycisk m (m=1) to powinno rozpocząć się napełnianie

obu zbiorników przez otwarcie górnych zaworów V1 i V2 (V1=1 lub V2=1 oznacza, że odpowiedni zawór jest otwarty). Gdy zbiornik zostanie wypełniony, należy zamknąć zawór górny, a jego zawartość może być zużywana przez otwarcie zaworu dolnego (odpowiednio W1 i W2). Kiedy zbiornik się opróżni, to zawór dolny jest zamykany. Ponowne napełnienie może rozpocząć się dopiero wtedy, gdy oba zbiorniki będą puste i zostanie naciśnięty przycisk m. Poniżej przedstawiono definicje zmiennych wykorzystywanych w algorytmie sterowania zaworami. Dostęp do definicji zmiennych uzyskuje się z menu *Project\Variable declarations..* lub przyciskając F8.

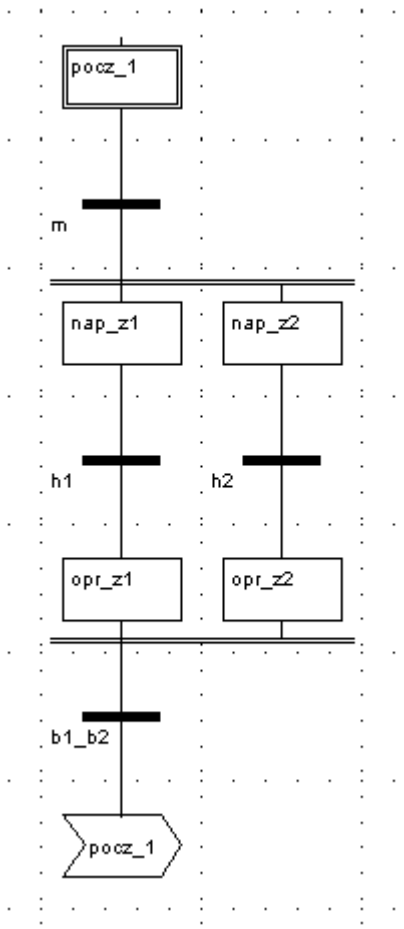


W tabeli 2 opisano znaczenie poszczególnych zmiennych:

Lp.	Nazwa zmiennej	Opis
1.	b1	Czujnik opróżnienia zaworu pierwszego
2.	b1_b2	b1 ***AND*** b2
3.	b2	Czujnik opróżnienia zaworu drugiego
4.	h1	Czujnik napełnienia zaworu drugiego
5.	h2	Czujnik napełnienia zaworu drugiego
6.	m	Przycisk rozpoczynający proces napełniania
7.	start	Zmienna niewykorzystywana w programie
8.	w1	Sterowanie zaworem dolnym (wypływ) zbiornika pierwszego
9.	w2	Sterowanie zaworem dolnym (wypływ) zbiornika drugiego
10.	z1	Sterowanie zaworem górnym (dopływ) zbiornika pierwszego
11.	z2	Sterowanie zaworem górnym (dopływ) zbiornika drugiego

Tabela 2. Opis zmiennych algorytmu sterowania procesem napełniania

Wprowadzić następujący schemat sterowania sekwencyjnego:



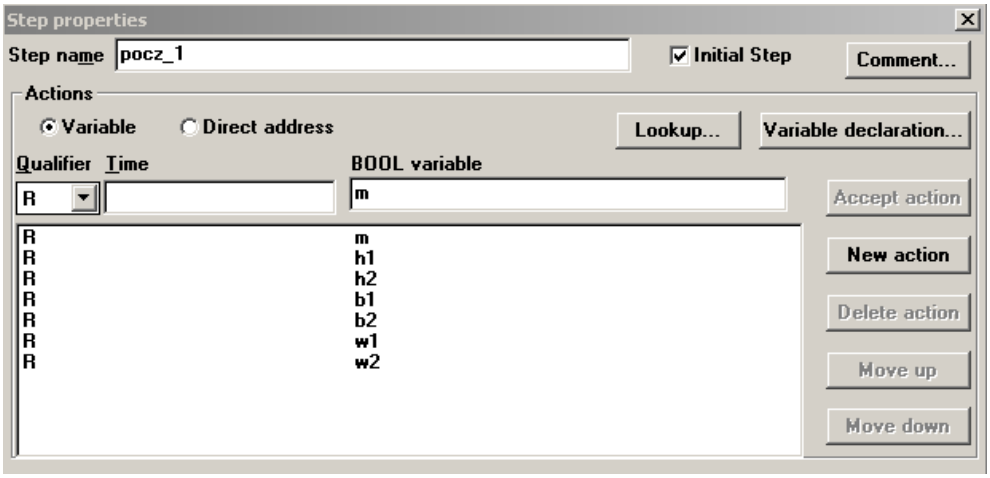
Schemat SFC sterowania procesem napełniania zbiorników
Etapy:
pocz_1 – etap początkowy
nap_z1 – napełnianie pierwszego zbiornika
nap_z1 – napełnianie drugiego zbiornika
opr_z1 – opróżnianie pierwszego zbiornika
opr_z2 – opróżnianie drugiego zbiornika

m, h1, h2, b1_b2 – tranzycje zależne od stanu odpowiednich zmiennych,

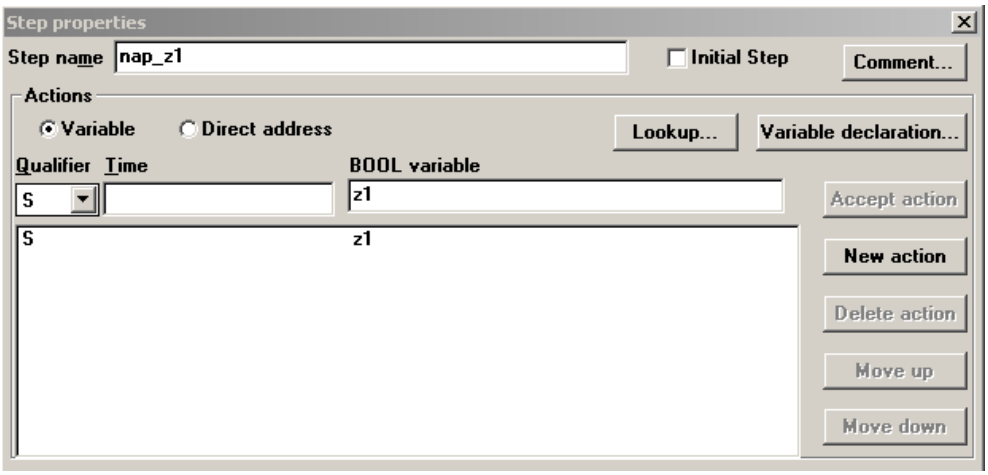
Poniżej przedstawiono definicje kolejnych etapów i tranzycji wykorzystanych w schemacie SFC sterowania procesem napełniania zbiorników

- **Etapy**

pocz_1



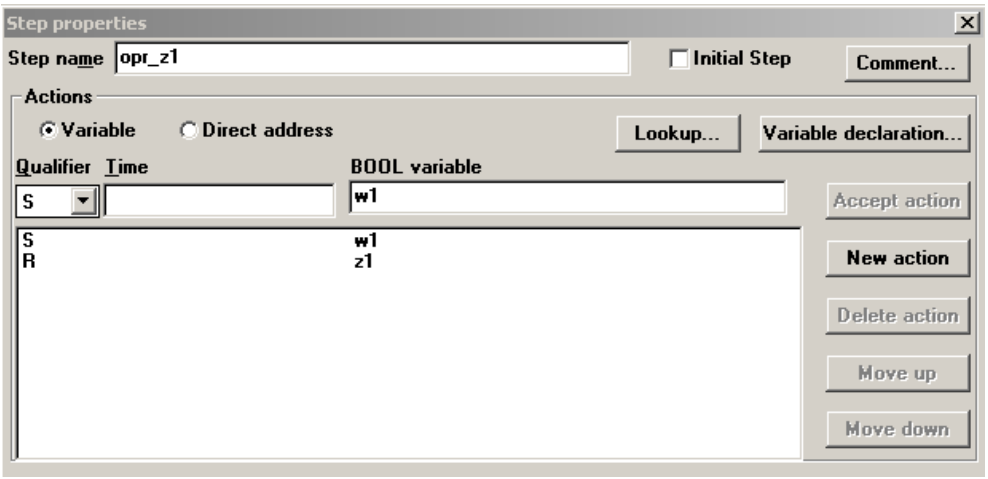
nap_z1



nap_z2

Etap **nap_z2** programuje się podobnie jak **nap_z1** przy czym ustawiana (S) jest zmienna **z2** a nazwa etapu (Step name) przyjmuje wartość *nap_z1*.

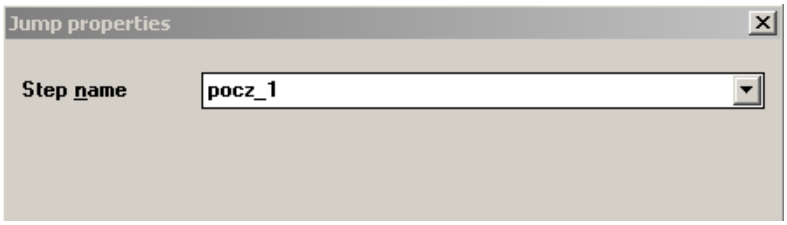
opr_z1



opr_z2

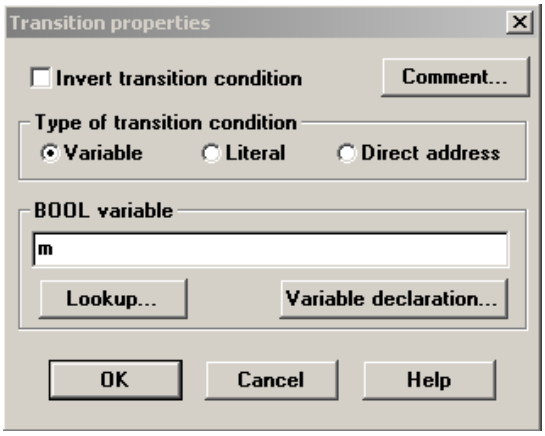
Etap **opr_z2** programuje się podobnie jak **opr_z1** przy czym ustawiana (S) jest zmienna **w2**, zerowana (R) zmienna **z2** a nazwa etapu (Step name) przyjmuje wartość o*pr_z1*.

Jump pocz_1



- **Tranzycje**

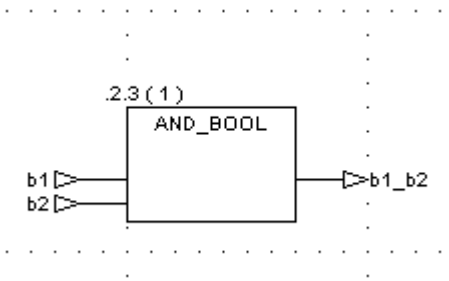
Tranzycja dla zmiennej m



Pozostałe tranzycje definiowane są podobnie przy czym w polach *BOOL variable* należy wstawić nazwy odpowiednich zmiennych (na podstawie schematu **SFC**).

UWAGA! W tabeli 2 występuje zmienna b1_b2. Jest ona obliczana w osobnej sekcji **FBD.**

Poniżej przedstawiono blok sekcji **FBD** realizujący operację ***AND***. W projekcie należy utworzyć nową sekcję w języku bloków funkcyjnych (**FBD**) oraz wprowadzić następujący blok funkcyjny.



Po zaprogramowaniu algorytmu sterowania napełnianiem zbiorników należy przeprowadzić animację. W tym celu niezbędne jest wprowadzanie wartości zmiennych wykorzystywanych w programie. Dostęp do edytora wartości zmiennych wykorzystywanego podczas animacji uzyskuje się z menu *Online/Reference Data Editor* lub ***CTRL+R*.**

Literatura

Legierski T., Wyrwał J., Kasprzyk J., Hajda J.: „Programowanie sterowników PLC”,
Pliki pomocy pakietu **Concept** firmy **AEG Schneider**