Krzysztof Falkowski

Sterowanie i programowanie układów mechatronicznych

Warszawa 2010

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

fundusze unijne dla oświaty

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Politechnika Warszawska
Wydział Samochodów i Maszyn Roboczych
Studia Podyplomowe dla Nauczycieli Przedmiotów Zawodowych
02-524 Warszawa, ul. Narbutta 84, tel. 22 849 43 07, 22 234 83 48
ipbmvr.simr.pw.edu.pl/spin/, e-mail: sto@simr.pw.edu.pl

Opiniodawca: dr inż. Jerzy GUSTOWSKI

Projekt okładki: Norbert SKUMIAŁ, Stefan TOMASZEK

Projekt układu graficznego tekstu: Grzegorz LINKIEWICZ

Skład tekstu: Krzysztof FALKOWSKI, Janusz BONAROWSKI

Publikacja bezpłatna, przeznaczona dla słuchaczy Studiów Podyplomowych dla Nauczycieli Przedmiotów Zawodowych.

ISBN 83-89703-36-X

Druk i oprawa: Drukarnia Expol P. Rybiński, J. Dąbek Spółka Jawna,
87-800 Włocławek, ul. Brzeska 4

Spis treści

Wstęp

Niniejsze materiały zostały opracowane w ramach realizacji projektu pn. „STUDIA PODYPLOMOWE DLA NAUCZYCIELI PRZEDMIOTÓW ZAWODOWYCH - mechatronika pojazdów i maszyn, komputerowo wspomagane projektowanie i wytwarzanie, bezpieczeństwo człowieka w środowisku pracy i ergonomia" finansowanego ze środków UNII EUROPEJSKIEJ w ramach PROGRAMU OPERACYJNEGO – KAPITAŁ LUDZKI. Materiały przeznaczone są dla słuchaczy tych studiów kierunku „Mechatronika pojazdów i maszyn" prowadzonych na Wydziale Samochodów i Maszyn Roboczych Politechniki Warszawskiej.

Niniejsze opracowanie przygotowano dla przedmiotu pt. "STEROWANIE I PROGRAMOWANIE UKŁADÓW MECHATRONICZNYCH". Jego zawartość merytoryczna w pełni odpowiada zakresowi opisanemu w sylabusie opracowanym dla tego przedmiotu.

Materiały uzupełniające i aktualizujące do przedmiotu będą udostępniane studentom za pośrednictwem systemu e-learning.

Całość opracowanych materiałów dydaktycznych dla ww. przedmiotu zawarta została w dwu częściach. W części pierwszej przedstawiono zagadnienia teoretyczne, natomiast w części drugiej znajdują się materiały do prowadzenia ćwiczeń ze sterownikami przemysłowymi.

Pierwsza część obejmuje trzy rozdziały. Rozdział 1 został poświęcony układom automatycznej regulacji. W rozdziale 2 przedstawione zostały kombinacyjne i sekwencyjne układy sterowania. Natomiast w rozdziale trzecim przedstawiona jest budowa sterowników PLC oraz metody adresowania, co jest niezbędną podstawą do przeprowadzenia ćwiczeń praktycznych zawartych w części drugiej.

Część druga obejmuje materiały dydaktyczne przeznaczone do prowadzenia ćwiczeń laboratoryjnych z wyżej wymienionego przedmiotu. Kolejne rozdziały zawierają opis teoretyczny i zadania do pięciu kolejnych ćwiczeń z programowania sterowników PLC.

Przedmiot prowadzony będzie w formie wykładów (8 godzin) i ćwiczeń (12 godzin). Głównym celem przedmiotu jest:

- zapoznanie ze strukturą układu sterowania,
- opis metod i praw sterowania w układach mechatronicznych,

- przedstawienie podstawowych algorytmów sterowania,
- zapoznanie z budową i programowaniem sterowników programowalnych.

Podczas wykładów poruszone zostaną następujące zagadnienia:

1. Podstawowe pojęcia i definicje w teorii sterowania.
2. Analogowe i cyfrowe systemy sterowania.
3. Projektowanie układów logicznych
4. Sterowanie kombinacyjne i sekwencyjne.
5. Sterowanie z zależnościami czasowymi.
6. Sterowanie zdarzeniami w układach mechatronicznych.
7. Sterowanie systemów mechatronicznych z wykorzystaniem regulatorów PID.
8. Budowa i struktura sterowników programowalnych.
9. Postawy programowania sterowników przemysłowych.

Przedmiot kończy się egzaminem sprawdzającym wiedzę teoretyczną w formie pisemnej. Podstawą dopuszczenia do egzaminu są obecności i zaliczenie ćwiczeń.

Poniższy materiał jest zbiorem podstawowych informacji niezbędnych do zrozumienia zagadnień związanych z postrzeganiem systemów mechatronicznych.

Część I
Podstawy Teoretyczne

1 Podstawy automatycznej regulacji

W tym rozdziale:

- Podstawowe pojęcia automatyki
- Transmitancja i model w przestrzeni stanu
- Regulator PID.

1.1. Wstęp

Jednym z podstawowych elementów systemu mechatronicznego jest układ sterowania. Układ ten zapewnia „inteligencję" systemu mechatronicznego. Obecnie większość wykonywanych produktów wyposażona jest w elementy sterujące. Współczesny samochód, który sukcesywnie ewoluuje w kierunku produktu mechatronicznego, posiada takie podsystemy jak układ sterowania pracą silnika i skrzyni biegów, układ przeciwpoślizgowy (ABS), układ stabilizacji toru jazdy, automatyczną klimatyzację, itd. Wymienione systemy posiadają układy odpowiedzialne za stabilizację parametrów (np. utrzymanie zadanej temperatury przez klimatyzację).

Do stabilizacji parametrów układu lub procesu (np. utrzymanie zadanej temperatury lub poziomu cieczy) wykorzystuje się układy automatycznej regulacji. Układ taki składa się ze obiektu regulacji, sprzężenia zwrotnego i regulatora. Na wyjściu obiektu regulacji (procesu regulacji) sygnał jest mierzony, a następnie przez sprzężenie zwrotne przekazany do elementu wyznaczającego różnicę między wartością zadaną a rzeczywistą. Różnica ta nazywa się uchybem regulacji i po przeliczeniu przez prawo sterowania regulatora wyznacza sygnał sterujący obiektem lub procesem.

W rozdziale przedstawiona zostanie budowa układów automatycznej regulacji. Podstawy modelowania układów oraz prawa sterowania regulatorów typu PID. W podrozdziale pierwszym omówione zostaną podstawowe pojęcia związane z automatyką i teorią sterowania.

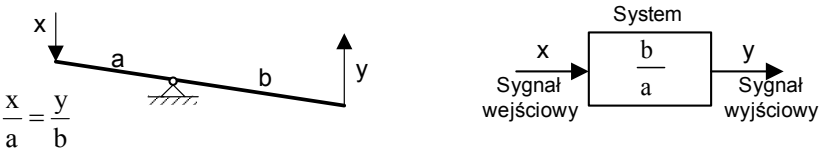
1.2. Podstawowe pojęcia

Proces sterowania w systemie machatronicznym bezpośrednio związany jest z takimi pojęciami jak sygnał, system, obiekt regulacji, itd. Przed omówieniem praw sterowania należy wprowadzić zbiór podstawowych pojęć.

Sygnał – w pojęciu ogólnym, przebieg w funkcji czasu dowolnej wielkości fizycznej występującej w procesie sterowania. Przykłady sy-

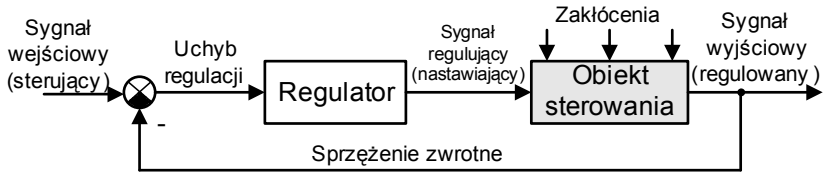
gnałów: siła, ciśnienie, przemieszczenie, napięcie, natężenie prądu, częstotliwość. Sygnał wykorzystywany jest do przekazywania informacji. Informacja może być zawarta w różnych cechach sygnału: amplituda, faza, szerokość impulsu itp. Sygnał może być wejściowy lub wyjściowy.

System – dowolny podzespół, zespół, przyrząd, urządzenie lub układ w którym wyróżniamy sygnał wejściowy i wyjściowy (rysunek 1.1).

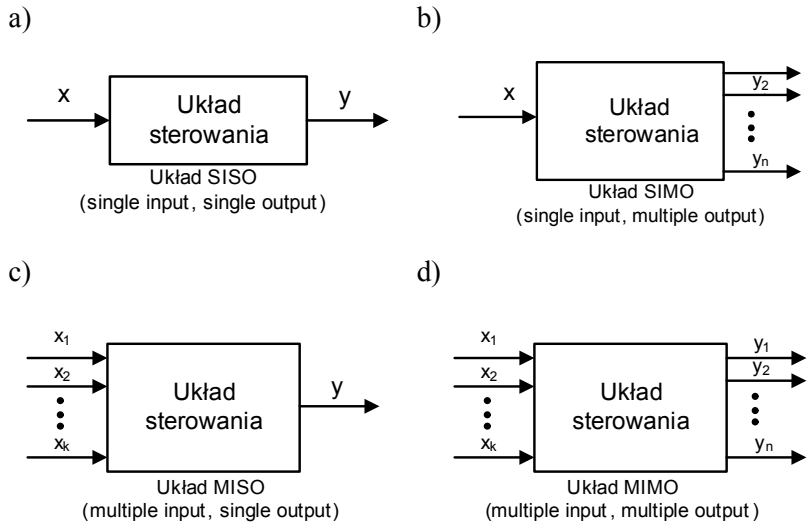


Rysunek 1.1. System

Układ sterowania (regulacji) – zespół elementów biorących bezpośredni udział w sterowaniu danym procesem, uporządkowany na zasadzie ich wzajemnej współpracy, tzn. zgodnie z kierunkiem przekazywania sygnałów (rysunek 1.2).



Rysunek 1.2. Układ sterowania (regulacji)



Rysunek 1.3. Klasyfikacja układu według kryterium liczby wejść i wyjść do systemu

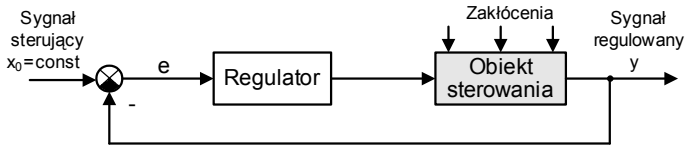
Regulator – układ, którego zadaniem jest zmiana sygnału regulującego (wejściowego do obiektu) w taki sposób, aby uchyb regulacji (sygnał błędu) był jak najmniejszy (rysunek 1.2).

Obiekt sterowania (regulacji) – obiekt fizyczny, którego stan (właściwości) chcemy zmieniać według ściśle określonej reguły (prawa sterowania).

Układy sterowania można podzielić według wielu kryteriów. Pierwszym kryterium jest ilość wejść i wyjść z układu. Według tego kryterium układy dzielimy na:

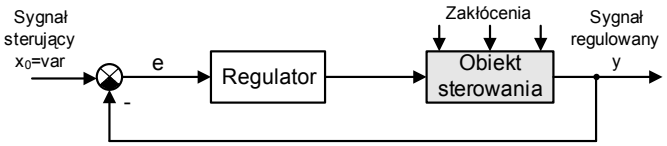
- układy (rysunek 1.3.a) o jednym wejściu i wyjściu (SISO – ang. single input and single output),
- układy (rysunek 1.3.b) o jednym wejściu i wielu wyjściach (SIMO – ang. single input and multiple output),
- układy (rysunek 1.3.c) o wielu wejściach i jednym wyjściu (MISO – ang. multiple input and single output),
- układy (rysunek 1.3.d) o wielu wejściach i wielu wyjściach (MISO – ang. multiple input and multiple output).

Na rysunku 1.3. przedstawiono poszczególne układy zgodnie z podaną klasyfikacją.



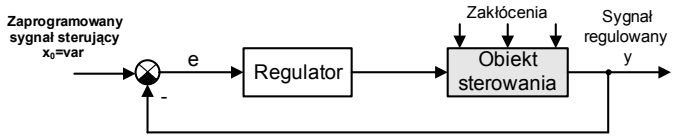
Rysunek 1.4. Układ stabilizacji automatycznej

Układy stabilizacji automatycznej – są to układy, w których sygnał sterujący ma stałą wartość (wartość zadana x_0 = const). W procesie regulacji podstawowym zadaniem układu jest utrzymanie stałej wartości sygnału regulowanego $y = x_0$.



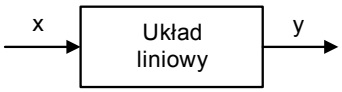
Rysunek 1.5. Układ regulacji nadążnej

Układy regulacji nadążnej – są to układy, w których sygnał sterujący jest **nieznaną** funkcją czasu (wartość zadana $\mathbf{x_0}$ = **var**). W procesie regulacji układ ma za zadanie tak działać, aby sygnał regulowany **y** nadążał za zmieniającą się wartością zadaną w taki sposób, aby uchyb regulacji dążył do zera.



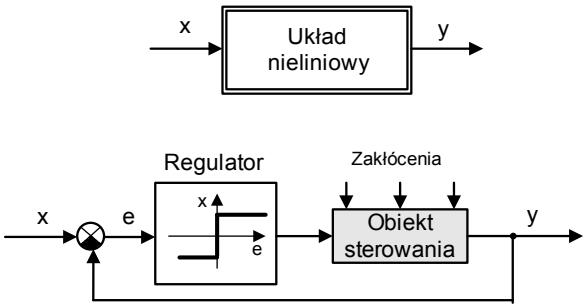
Rysunek 1.6. Układ regulacji programowalnej

Układy regulacji programowej – są szczególnym przypadkiem układów regulacji nadążnej, w których sygnał sterujący jest **znaną** i określoną według pewnego programu funkcją czasu (wartość zadana $\mathbf{x_0}$ = **var**.). W procesie regulacji układ ma za zadanie tak działać, aby sygnał regulowany **y** nadążał za zmieniającą się wartością zadaną w taki sposób, aby uchyb regulacji dążył do zera.



Rysunek 1.7. Układ liniowy

Układy liniowe – to układy, w których wszystkie elementy spełniają zasadę superpozycji.



Rysunek 1.8. Układ nieliniowy

Układy nieliniowe – to układy, w których przynajmniej jeden element nie spełnia zasady superpozycji.

Układy stacjonarne – to układy, w których elementy mają parametry **niezależne** od czasu. W przypadku elementów dynamicznych parametry (współczynniki równań różniczkowych) nie zależą od czasu.

$$a \cdot \frac{d^2y}{dt^2} + b \cdot \frac{dy}{dt} + c \cdot y = k \cdot x$$

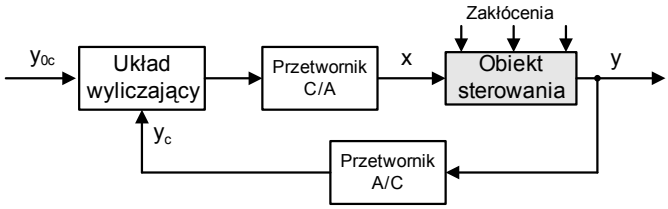
Układy niestacjonarne – to układy, w których elementy mają parametry **zależne** od czasu. W przypadku elementów dynamicznych parametry (współczynniki równań różniczkowych) zależą od czasu.

$$a(t) \cdot \frac{d^2y}{dt^2} + b(t) \cdot \frac{dy}{dt} + c(t) \cdot y = k \cdot x$$

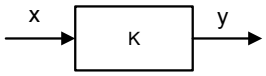
Układy analogowe – w układach analogowych wynik pomiaru przedstawiony jest w postaci wielkości fizycznej związanej z wielkością regulowaną zależnością funkcyjną.

Układy cyfrowe (rysunek 1.9) – w układach cyfrowych wielkości fizyczne przetwarzane są na postać cyfrową reprezentowaną przez liczby. Reprezentacje liczbowe wielkości fizycznych są odpowiednio kwantowane czyli dzielone na odpowiednią liczbę poziomów. W układach wyliczających realizowane jest odpowiednie prawo sterowania, w efekcie którego powstaje liczba odpowiadająca sygnałowi regulującemu (wejściowemu do obiektu regulacji). Liczba ta przekształcana jest

z postaci cyfrowej na postać analogową i przekazywana jest w tej postaci na wejście do obiektu regulacji.

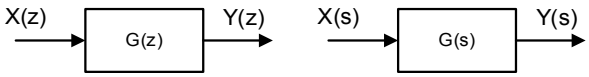


Rysunek 1.9. Układ sterowania cyfrowego



Rysunek 1.10. Układ statyczny

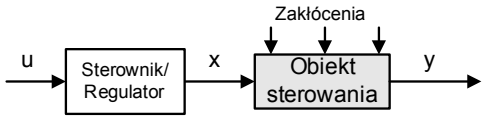
Układy statyczne (rysunek 1.10) – układy zbudowane z elementów opisanych równaniami algebraicznymi. Stosunek sygnału wyjściowego do wejściowego bez względu na ich częstotliwość daje w efekcie tą samą liczbę.



Rysunek 1.11. Dyskretny i ciągły układ dynamiczny

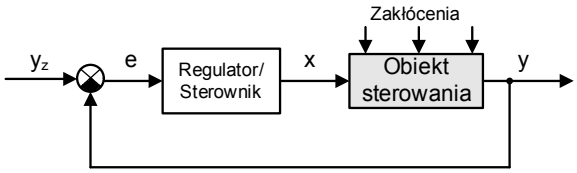
Układy dynamiczne (1.11) – układy zbudowane z elementów opisanych równaniami różniczkowymi w przypadku układów ciągłych lub równaniami różnicowymi w przypadku układów impulsowych. Stosunek amplitudy sygnału wyjściowego do wejściowego daje w efekcie dla różnych częstotliwości różne liczby. Zależność między sygnałem wyjściowym i wejściowym opisywana może być przez transmitancję operatorową (opis we współrzędnych stanu) lub transmitancję impulsową (impulsowy opis we współrzędnych stanu).

Układy otwarte – posiadają strukturę szeregową, bez sprzężenia zwrotnego, w których regulator (sterownik) nie ma informacji o bieżącej wartości wielkości regulowanej.



Rysunek 1.12. Układ otwarty

Układy zamknięte (rysunek 1.13) – posiadają strukturę ze sprzężeniem zwrotnym, w której regulator (sterownik) ma informację bezpośrednią lub pośrednią o parametrach stanu obiektu lub o wielkościach regulowanych.



Rysunek 1.13. Układ zamknięty

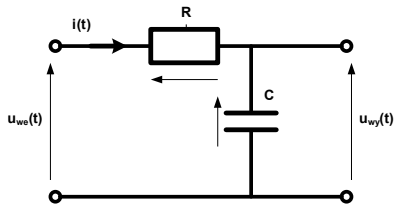
1.3. Transmitancja i model w przestrzeni stanu

W automatyce układy opisane są przez transmitancję operatorową i model w przestrzeni stanu. Transmitancja operatorowa opisuje układy dynamiczne w dziedzinie częstotliwości natomiast model w przestrzeni stanu opisuje właściwości układu w dziedzinie czasu.

Transmitancja operatorowa

Transmitancją operatorową jest zdefiniowana jako stosunek (iloraz) transformaty Laplace'a sygnału wyjściowego do transformaty Laplace'a sygnału wejściowego, wyznaczonych dla zerowych warunków początkowych.

Prześledzimy prosty przykład. Poniżej przedstawiony jest schemat układu RC.



Rysunek 1.14. Układ RC

Na wejściu układu RC podano napięcie wejściowe $u_{we}(t)$, którego wartość opisana jest zależnością:

$$u_{we}(t) = Ri(t) + \frac{1}{C}\int i(t)dt$$

Natomiast na wyjściu układu pojawi się napięcie $u_{wy}(t)$, które jest równe:

$$u_{wy}(t) = \frac{1}{C}\int i(t)dt$$

Zgodnie z przytoczoną definicją transmitancji operatorowej, wykonamy transformatę Laplace'a napięcia wyjściowego i napięcia wejściowego:

$$U_{we}(s) = RI(s) + \frac{1}{C}\frac{I(s)}{s}$$

$$U_{wy}(s) = \frac{1}{C}\frac{I(s)}{s}$$

Jeżeli wyznaczymy stosunek napięcia wyjściowego do napięcia wejściowego, to otrzymamy transmitancję układu RC:

$$G(s) = \frac{U_{wy}(s)}{U_{we}(s)} = \frac{\frac{1}{C}\frac{I(s)}{s}}{RI(s) + \frac{1}{C}\frac{I(s)}{s}}$$

$$G(s) = \frac{I(s)\frac{1}{sC}}{I(s)\left(R + \frac{1}{sC}\right)}$$

$$G(s) = \frac{\frac{1}{sC}}{\left(\frac{RCs+1}{sC}\right)} = \frac{1}{RCs+1}$$

Ostatecznie transmitancja operatorowa układu jest równa:

$$G(s) = \frac{1}{RCs + 1}$$

Jak wynika z przytoczonej zależności transmitancja jest związana z parametrami układu i opisuje jego właściwości. Przedstawioną analizę można przeprowadzić dla dowolnego układu. Jeżeli zastąpimy transformatę Laplace'a przez transformatę Z, otrzymamy transmitancje układu dyskretnego.

Model w przestrzeni stanu

Do opisu układów w dziedzinie czasu wykorzystuje się model w przestrzeni stanu.

Przed zdefiniowaniem modelu, zatrzymamy się przy pojęciu stanu układu. Wyjaśnienie tego pojęcia jest podstawą zrozumienia modelu w przestrzeni stanu. Często używamy stwierdzenia „że coś jest w stanie". Termin „stan" w odniesieniu do obiektu, oznacza gotowość do wykonania zadania. Jeżeli coś lub ktoś nie może zrealizować jakiegoś zadania, to stwierdzamy „że nie jest w stanie". Na podstawie znajomości stanu obiektu można przewidzieć jak obiekt będzie realizował zadanie.

Dlatego stanem układu nazywa się najmniej liczny zbiór wielkości, które należy określić w chwili $t = t_0$, aby można było jednoznacznie przewidzieć zachowanie się układu w każdej chwili $t \geq t_0$, dla każdego sygnału wymuszającego należącego do danego zbioru sygnałów wymuszających, przy założeniu, że wszystkie elementy zbioru wymuszeń są znane dla $t \geq t_0$. Wielkości te są nazywane zmiennymi lub współrzędnymi stanu. Wektor będący zbiorem n zmiennych stanu nazywa się wektorem stanu.

W ogólnym przypadku stan, w którym znajduje się układ zależy od sygnałów wejściowych oraz od wcześniejszego stanu, w którym znajdował się układ. Jest to coś w rodzaju pamięci układu. Należy zauważyć, że stan taki można rozpatrywać w kategoriach energii. Stan układu jest zdeterminowany przez ilość zgromadzonej w układzie energii.

Do wyznaczenia miary zmiany stanu układu w czasie będziemy wykorzystywać pochodną wektora stanu. Stan procesu dla $t \geq t_0$, zależy tylko od stanu w chwili początkowej t_0 oraz od wymuszenia u(t) dla $t \geq t_0$, co można opisać równaniem (równanie stanu):

$$\dot{x} = f(x, u, t)$$

z warunkiem początkowym $x(t_0)=x_0$.

Ponieważ nie wszystkie zmienne stanu są dostępne (bezpośrednio mierzalne), to wektor stanu x nie jest zarazem wektorem odpowiedzi y układu. Dla pełnego opisu procesu potrzebne jest jeszcze równanie wiążące wyjście układu „y" z wektorem stanu i wektorem wymuszeń (równanie wyjść):

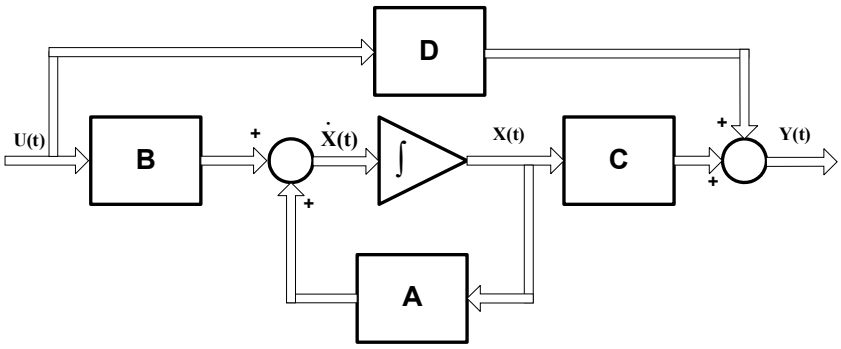
$$\boldsymbol{y = g(x, u, t)}$$

Dla układów liniowych funkcje wektorowe f i g są funkcjami liniowymi wektora stanu i wejść. Dla układów liniowych niestacjonarnych (o parametrach zależnych od czasu t) równania przyjmują postać:

$$\boldsymbol{\dot{x} = A(t)x(t) + B(t)u(t)}$$

$$\boldsymbol{y = C(t)x(t) + D(t)u(t)}$$

gdzie: **A**(t), **B**(t), **C**(t), **D**(t) – odpowiednio macierz stanu o wymiarze *nxn*, wymuszeń (wejść) **B**(t) o wymiarze *nxm*, odpowiedzi (wyjść) **C**(t) o wymiarze pxn, transmisyjna **D**(t) o wymiarze *pxm*.



Rysunek 1.15. Model w przestrzeni stanu

W układach linowych stacjonarnych (elementy macierzy nie zmieniają swoich wartości) układ opisany jest równaniami:

$$\boldsymbol{\dot{x} = Ax(t) + Bu(t)}$$

$$\boldsymbol{y = Cx(t) + Du(t)}$$

Wyznaczymy model w przestrzeni stanu dla układu RC z rysunku 1.14. Jak zaznaczono wcześniej napięcie wejściowe i wyjściowe opisane jest zależnością :

$$u_{we}(t) = Ri(t) + \frac{1}{C}\int i(t)dt$$

$$u_{wy}(t) = \frac{1}{C}\int i(t)dt$$

W podanych wyrażeniach występują całki z wartości prądu natomiast brakuje pochodnych. Dlatego wprowadzimy pomocniczą wielkość jaką jest ładunek elektryczny q. Pochodna po czasie ładunku równa jest natężeniu prądu elektrycznego:

$$i(t) = \frac{dq(t)}{dt}$$

Uwzględniając powyższą zależność w równaniach opisujących zmianę napięcia wejściowego i wyjściowego układu RC otrzymamy:

$$u_{we}(t) = R\frac{dq(t)}{dt} + \frac{1}{C}q(t)$$

$$u_{wy}(t) = \frac{1}{C}q(t)$$

Przekształcając powyższy układ do postaci, otrzymamy:

$$\frac{dq(t)}{dt} = \frac{u_{we}(t)}{R} - \frac{1}{RC}q(t)$$

$$u_{wy}(t) = \frac{1}{C}q(t)$$

Jeżeli sprowadzimy powyższe równania do postaci, to:

$$\underbrace{\frac{dq(t)}{dt}}_{\dot{x}(t)} = \underbrace{\left[-\frac{1}{RC}\right]}_{A}\underbrace{q(t)}_{x(t)} + \underbrace{\left[\frac{1}{R}\right]}_{B}\underbrace{u_{we}(t)}_{u(t)}$$

$$\underbrace{u_{wy}(t)}_{y(t)} = \underbrace{\left[\frac{1}{C}\right]}_{C}\underbrace{q(t)}_{x(t)}$$

Pierwsze równie jest równaniem stanu układu z rysunku 1.14 natomiast drugie jest równaniem wyjść. Układ opisany jest przez macierze:

$A = -\frac{1}{RC}$ - macierz stanu o wymiarach 1x1 ponieważ mamy tylko jeden element wektora stanu **x(t)**=q(t) – ładunek elektryczny,

$B = \frac{1}{R}$ - macierz wejść o wymiarach 1x1 ponieważ wektor stanu jest jednoelementowy i wektor wejść jest jednoelementowy $\mathbf{u}(t)=u_{we}(t)$,

$C = \frac{1}{C}$ - macierz wyjść o wymiarze 1x1 ponieważ występuje jeden element wektora stanu i jeden element wektora wyjść $\mathbf{y}(t)=u_{wy}(t)$,

$D = 0$ - macierz transmisyjna równa jest zero ponieważ nie występuje przenoszenie wartości wyjściowej na wyjście układu, a macierz ma wymiar 1x1 ponieważ występuje jedno wyjście i jedno wejście w układzie.

Między modelami występują ścisłe zależności. Model zapisany jako transmitancję można zapisać jako model w przestrzeni stanu natomiast model w przestrzeni stanu można przedstawić jako transmitancję:

$$G(s) = C[Is - A]^{-1}B + D$$

UWAGA!
Występuje odstępstwo w przypadku członu różniczkującego. Idealnego członu różniczkującego nie można przedstawić jako model w przestrzeni stanu. Idealny człon różniczkujący jest tworem matematycznym, który nie odzwierciedla procesów rzeczywistych zachodzących w przyrodzie.

1.4. Podstawowe struktury układów sterowania

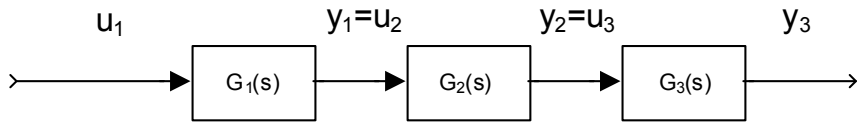
W procesie modelowania układów dynamicznych wykorzystuje się trzy rodzaje struktur:

- połączenia szeregowe,
- połączenia równoległe,
- dodatnie i ujemne sprzężenia zwrotne.

Dodatkowo w schematach blokowych wykonuje się przemieszczanie węzłów sumacyjnych oraz węzłów informacyjnych (rozgałęzienia).

Połączenie szeregowe

Szeregowym połączeniem (kaskada, łańcuch) nazywamy takie połączenie układów, w którym sygnał wyjściowy dowolnego członu ***i*** jest sygnałem wejściowym następnego w kierunku przepływu sygnału członu ***i+1.***



Rysunek 1.16. Szeregowe połączenie transmitancji

$$Y_1(s) = G_1(s)U_1(s)$$

$$Y_2(s) = G_2(s)U_2(s)$$

$$Y_3(s) = G_3(s)U_3(s)$$

$$Y_1(s) = U_2(s)$$

$$Y_2(s) = G_2(s)G_1(s)U_1(s)$$

$$Y_2(s) = U_3(s)$$

$$Y_3(s) = G_3(s)G_2(s)G_1(s)U_1(s)$$

Ostatecznie transmitancja połączenia szeregowego jest równa:

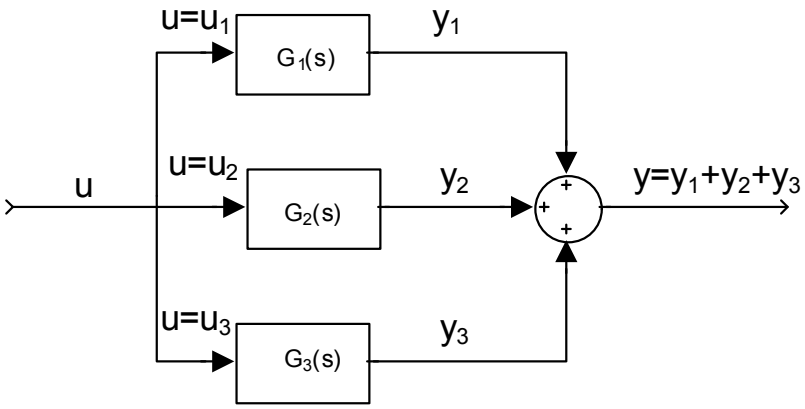
$$G_s(s) = G_3(s)G_2(s)G_1(s)$$

Dla szeregowo połączonych n transmitancji $G_i(s)$, otrzymamy

$$G_s(s) = \prod_{i=1}^{n-1} G_i(s)$$

Połączenie równoległe

Połączeniem równoległym nazywamy takie połączenie układów, w którym na wejście układów podana jest ten sam sygnał, a sygnałem wyjściowym jest suma algebraiczna sygnałów wyjściowych poszczególnych członów.



Rysunek 1.17. Połączenie równoległe transmitancji

$$Y_1(s) = G_1(s)U_1(s)$$

$$Y_2(s) = G_2(s)U_2(s)$$

$$Y_3(s) = G_3(s)U_3(s)$$

$$U(s) = U_1(s) = U_2(s) = U_3(s)$$

$$Y(s) = Y_1(s) + Y_2(s) + Y_3(s)$$

$$Y(s) = G_1(s)U(s) + G_2(s)U(s) + G_3(s)U(s)$$

$$Y(s) = [G_1(s) + G_2(s) + G_3(s)]U(s)$$

Ostatecznie transmitancja połączenia równoległego jest równa:

$$G_r(s) = G_1(s) + G_2(s) + G_3(s)$$

Dla szeregowo połączonych n transmitancji $G_i(s)$, otrzymamy

$$G_r(s) = \sum_{i=1}^{n-1} G_i(s)$$

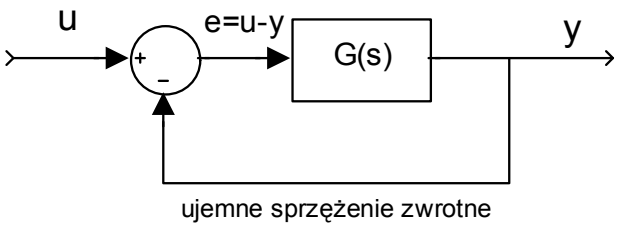
Sprzężenie zwrotne

W układzie ze sprzężeniem zwrotnym sygnał wyjściowy z układu umieszczonego w torze głównym podawany jest na wejście układu. Sygnał wejściowy może być wprowadzany na wejście ze znakiem minus, wtedy mamy do czynienia z ujemnym sprzężeniem zwrotnym,

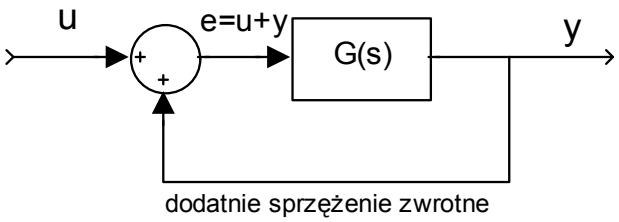
lub ze znakiem plus wtedy mamy do czynienia z dodatnim sprzężeniem zwrotnym.

Dodatnie sprzężenie zwrotne jest sporadycznie wykorzystywane w urządzeniach. W większości przypadków występowanie dodatniego sprzężenia zwrotnego powoduje utratę stabilności przez układ. Natomiast ujemne sprzężenie zwrotne poprawia stabilność i ułatwia sterowanie. Sprzężenie to jest powszechnie stosowane w układach automatycznej regulacji.

a)



b)



Rysunek 1.18. Ujemne sprzężenie zwrotne a), dodatnie sprzężenie zwrotne b)

Odpowiedź układu ze sprzężeniem zwrotnym jest równa:

$$Y(s) = G(s)E(s)$$

gdzie: E(s) – uchyb:

$$E(s) = U(s) \pm Y(s)$$

Stąd transmitancja układu zamkniętego jest równa:

$$Y(s) = G(s)[U(s) \pm Y(s)]$$

$$Y(s) = G(s)U(s) \pm G(s)Y(s)$$

$$Y(s) \mp G(s)Y(s) = G(s)U(s)$$

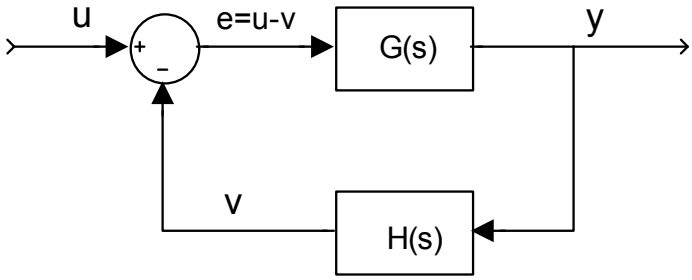
$$[1 \mp G(s)]Y(s) = G(s)U(s)$$

$$Y(s) = \frac{G(s)}{[1 \mp G(s)]}U(s)$$

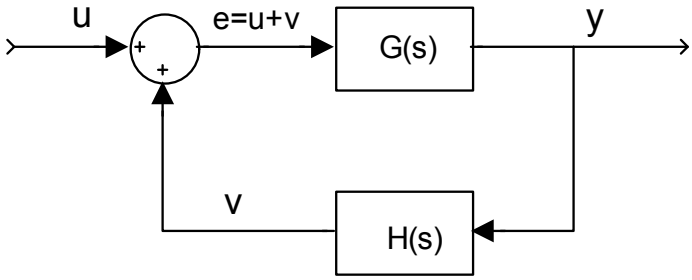
$$G_{sz}(s) = \frac{Y(s)}{U(s)} = \frac{G(s)}{[1 \mp G(s)]}$$

W torze sprzężenia zwrotnego, może wystąpić układ o transmitancji H(s) np. transmitancja czujnika mierzącego wartość wyjściową (rysunek 1.19).

a)



b)



Rysunek 1.19. Ujemne sprzężenie zwrotne a), dodatnie sprzężenie zwrotne b)

W tym przypadku wartość transmitancji jest równa:

$$Y(s) = G(s)E(s)$$

gdzie: E(s) – uchyb:

$$E(s) = U(s) \pm H(s)Y(s)$$

stąd transmitancja układu zamkniętego jest równa:

$$Y(s) = G(s)[U(s) \pm H(s)Y(s)]$$

$$Y(s) = G(s)U(s) \pm G(s)H(s)Y(s)$$

$$Y(s) \mp G(s)H(s)Y(s) = G(s)U(s)$$

$$[1 \mp G(s)H(s)]Y(s) = G(s)U(s)$$

$$Y(s) = \frac{G(s)}{[1 \mp G(s)H(s)]}U(s)$$

$$G_{zz}(s) = \frac{Y(s)}{U(s)} = \frac{G(s)}{[1 \mp G(s)H(s)]}$$

Korzystając z podanych powyżej zasad można wyznaczyć transmitancję dowolnego układu.

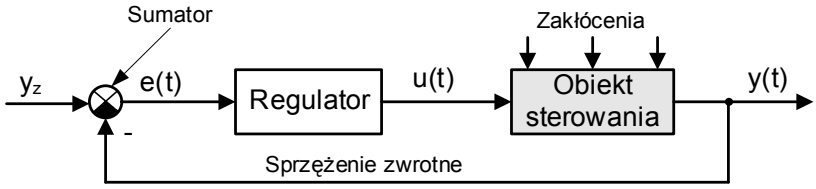
UWAGA!
Podczas wyznaczania transmitancji układu zamkniętego może zaistnieć potrzeba przesuwania węzłów informacyjnych i sumacyjnych. W węźle sumacyjnym wyliczana jest suma algebraiczna sygnałów. Natomiast w węźle informacyjnym sygnał ulega rozdzieleniu. Przesuwanie węzłów można wyznaczyć korzystając z przedstawionych powyżej zasad.

1.5. Regulacja PID

Na rysunku 1.19 przestawiona jest podstawowa struktura układu automatycznej regulacji. Układ taki składa się z obiektu regulacji, regulatora, sprzężenia zwrotnego. Na wejście obiektu podawany jest sygnał sterujący u(t), który jednocześnie jest sygnałem wyjściowym z regulatora. Na wyjściu obiektu regulacji wyprowadzony jest sygnał wyjściowy y(t), który jednocześnie podany jest przez sprzężenie zwrotne na wejście układu automatycznej regulacji. Do regulatora wprowadzona jest wartość uchybu regulacji e(t), który jest wyznaczony w elemencie

sumacyjnym (element sumacyjny wyznacza sumę algebraiczną sygnałów wejściowych). Wartość uchybu regulacji wyznaczona jest jako różnica między wartością zadaną sygnału wyjściowego a wartością rzeczywistą sygnału wyjściowego (różnica wyznaczona jest dla ujemnego sprzężenia zwrotnego):

$$e(t) = y_z - y(t)$$



Rysunek 1.19. Układ automatycznej regulacji

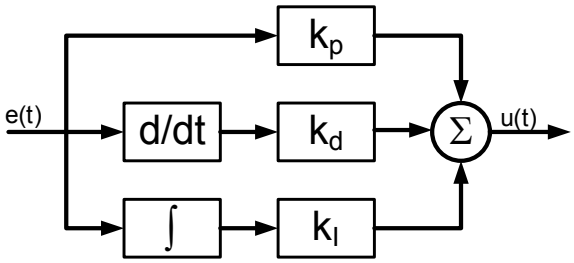
Wejściem układu automatycznej regulacji jest wartość zadana y_z, do której powinna dążyć wartość na wyjściu układu. Struktura układu przedstawiona na rysunku 1.19 zapewnia realizację celu sterowania jakim jest stabilizacja parametru y(t).

Do sterowania wykorzystuje się regulatory o różnej strukturze. Struktura regulatora wynika z założonego celu regulacji i przyjętych wskaźników jakości procesu sterowania.

W większości aplikacji można spotkać regulatory typu PID. Regulator taki wyznacza wartość sygnału sterującego jako sumę wartości proporcjonalnej, całki i pochodnej w czasie z uchybu regulacji:

$$u(t) = k_p e(t) + k_d \frac{de(t)}{dt} + k_I \int e(t)dt$$

gdzie: k_p, k_d, k_I – odpowiednio wzmocnienie członu proporcjonalnego, różniczkującego i całkującego.



Rysunek 1.20. Struktura regulatora PID

Przyjmując odpowiednie wzmocnienia równe zero, można zdefiniować regulator typu:

P – dla $k_d=k_I=0$,

$$u(t) = k_p e(t),$$

PI – dla $k_d=0$,

$$u(t) = k_p e(t) + k_I \int e(t)dt,$$

PD – dla $k_I=0$,

$$u(t) = k_p e(t) + k_d \frac{de(t)}{dt}.$$

2 Sterowanie kombinacyjne i sekwencyjne

W tym rozdziale:

- Zapis funkcji logicznych
- Minimalizacja funkcji logicznych
- Asynchroniczne i synchroniczne układy sekwencyjne

2.1. Wstęp

W pierwszym rozdziale przedstawiono układy automatycznej regulacji. Oprócz zadań związanych ze stabilizacją, utrzymaniem parametrów lub nadążaniem za parametrami istotne jest sterowanie procesem. Podczas realizacji procesu ważne jest wykonywanie pewnych stałych czynności, które są reakcją na pojawiające się zdarzenia.

Przykładem takiego systemu mogą być automatyczne systemy produkcyjne, gdzie pojawienie się detalu powoduje wykonanie czynności technologicznych według ustalonej kolejności.

W rozdziale przedstawione zostaną kombinacyjne i sekwencyjne układy sterowania.

2.2. Układy kombinacyjne

Układem kombinacyjnym nazywamy układ, w którym dla każdej kombinacji n-elementowego wektora wejściowego odpowiada jednoznaczna kombinacja m-elementowego wektora wyjściowego.

Rysunek 2.1. Układ kombinacyjny

W niektórych publikacjach układ kombinacyjny definiowany jest jako układ, który realizuje boolowskie funkcje logiczne. Obie podane definicje są słuszne i wyczerpująco przedstawiają cechy układu kombinacyjnego.

Układ taki charakteryzuje się:

- n – elementowym wektorem wejściowym n>1,

- m – elementowym wektorem wyjściowym m>1,
- realizuje operacje dwuargumentowe „and” i „or” i jednoargumentową „not” na argumentach „0” i „1” (algebra Boole'a).

Oprócz podanych powyżej operacji wykonywane są operacje „nand”, „nor”, „xor” i „nxor”. Operacje powyższe wnikają z praw algebry Boole'a (np. prawo de Morgana). Liczba wejść i wyjść w układzie kombinacyjnym może być różna. Ostatecznie funkcję logiczną można przedstawić jako:

$$Y = f(X)$$

gdzie:

Y – wektor wyjściowy m-elementowy,

X – wektor wejściowy n-elementowy.

Dla układu o trzech wejściach i dwu wyjściach można zapisać jako:

$$y_1 = f(x_1, x_2, x_3)$$

$$y_2 = f(x_1, x_2, x_3)$$

	x_1	x_2	x_3	y_1	y_2
0	000			01	
1	001			11	
2	010			11	
3	011			-0	
4	100			1-	
5	101			-0	
6	110			-0	
7	111			01	

Z podanej tabeli można wyznaczyć zbiór wektorów wejściowych, dla których otrzymamy na wyjściu „1”:

y_1=1 dla [001], [010],[100]

y_2=1 dla [000], [001],[010], [111].

Zbiór, dla których przyjmuje wartości „0”:

y_1=0 dla [000], [111]

y_2=0 dla [011], [101], [110].

W tabeli podano „-”, którą może przyjąć wektor wyjściowy. Kreska oznacza stan, który nie może zaistnieć z przyczyn technicznych lub wystąpienie stanu nie wpływa na pracę układu.

W przykładzie kreski występują dla wektorów wejściowych:

y_1=”-„ dla [101], [011], [110]

y_2=”-„ dla [100].

Funkcję logiczną można przedstawić w postaci sumacyjnej lub iloczynowej. Postać sumacyjna przestawiona jest jako suma iloczynów poszczególnych wektorów wejściowych, dla których iloczyn równy jest 1. Dla podanego powyżej przykładu otrzymamy:

$$y_1 = \overline{x_1}\overline{x_2}x_3 + \overline{x_1}x_2\overline{x_3} + x_1\overline{x_2}\overline{x_3}$$

Postać iloczynowa przedstawiana jest jako iloczyn sum elementów wektora stanu, dla których suma równa jest 0. Dla podanego powyżej przykładu postać iloczynowa jest równa:

$$y_1 = (\overline{x_1} + \overline{x_2} + \overline{x_3})(x_1 + x_2 + x_3)$$

Posługiwanie się wektorami w zapisie binarnym jest niewygodne, dlatego w dalszej analizie wprowadzimy zapis dziesiętny dla postaci sumacyjnej:

$$y_1 = \sum m(1,2,4) + d(3,5,6)$$

$$y_2 = \sum m(0,1,2,7) + d(4)$$

i dla postaci iloczynowej:

$$y_1 = \prod M(0,7) + d(3,5,6)$$

$$y_2 = \prod M(3,5,6) + d(4)$$

Dla postaci sumacyjne podaje się wektory wejściowe, dla których wyjścia przyjmują wartość „1” lub „-”, natomiast dla postaci iloczynowej podaje się wektory wejściowe, dla których wyjścia przyjmują wartość „0” lub „-”.

Rozpatrzymy układ sterowania procesem napełniania zbiornika. Poziom w zbiorniku określany jest przez trzy czujniki. Tym samym wektor wejść posiada postać:

[czujnik nr 1, czujnik nr 2, czujnik nr 3].

Jeżeli wszystkie trzy czujniki sygnalizują przekroczenie poziomu, to do zbiornika nie będzie pompowana ciecz. W przypadku, gdy czujnik nr 1 sygnalizuje poziom cieczy poniżej progu a dwa pozostałe powyżej to włączona jest pompa nr 1. Sygnalizowanie poziomu cieczy poniżej progu przez czujnik nr 1 i nr 2 powoduje włączenie pompy nr 2 i wyłączenie pompy nr 1. Jeżeli wszystkie trzy czujniki sygnalizują poziom cieczy poniżej progu, to włączone będą obie pompy (pompa nr 1 i nr 2).

Układ kombinacyjny sterujący pracą pomp posiada trzy wejścia i dwa wyjścia. Do przeprowadzenia pełnej analizy niezbędne jest wyznaczenie dwu funkcji logicznych:

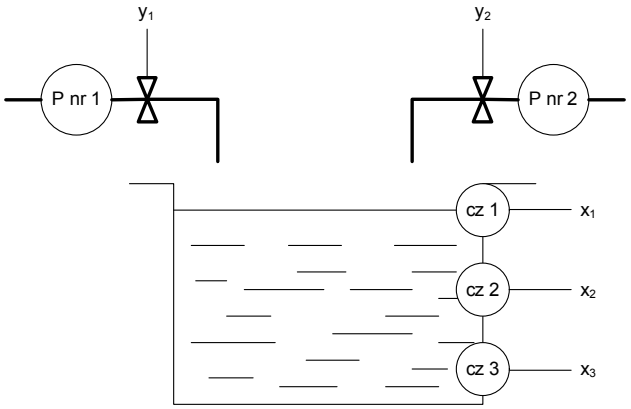
- sterowanie pompą pierwszą

$$y_1 = f(x_1, x_2, x_3)$$

- sterowanie pompą drugą:

$$y_2 = f(x_1, x_2, x_3)$$

gdzie: x_1 – sygnał z czujnika nr 1, x_2 – sygnał z czujnika nr 2, x_3 – sygnał z czujnika nr 3.



Rysunek 2.2. Układ do napełniania zbiornika

Dla trzech zmiennych wejściowych można wyznaczyć osiem kombinacji wektora wejściowego.

Wartość	Wektor wejściowy [x_1, x_2, x_3]
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Układ kombinacyjny sterujący pompą nr 1 i nr 2, dla kolejnych wektorów wejściowych posiada jednoznacznie zdefiniowane wektory sterujące (wyjściowe):

Wartość	Wektor wejściowy [x_1, x_2, x_3]	Wektor wyjściowy [y_1, y_2]
0	000	11
1	001	01
2	010	-
3	011	10
4	100	-
5	101	-
6	110	-
7	111	00

Jak można zauważyć w tabeli powyżej, są takie wektory wejściowe, które nie mogą zaistnieć np. 010, co oznacza przekroczenie poziomu w czujniku 1 i 3, natomiast w czujniku 2 poziom jest odpowiedni. Wystąpienie takiego wektora wejściowego jest niezgodne z zasadą działania układu sterowania. Układ posiada dwa wyjścia, dlatego do opisu układu kombinacyjnego należy wyznaczyć dwie funkcje logiczne.

Do opisu wykorzystamy system dziesiętny ponieważ posługiwania się długimi słowami binarnymi jest bardzo trudne. Dla podanego przykładu otrzymamy funkcje:

$$y_1 = \sum m(0,3) + d(2,4,5,6)$$

$$y_2 = \sum m(0,1) + d(2,4,5,6)$$

Pierwsza część (litera m) opisuje wektory dla których otrzymamy jedynki logiczne. Natomiast litera d opisuje kreski, czyli stany zabronione. Postać sumacyjna przedstawia funkcję logiczną jako sumę wektorów wejściowych. Każdy z wektorów jest traktowany jako iloczyn elementów wektora wejściowego. Elementy wektora, które mają wartość logiczną zero dla kombinacji zapisywane są ze znakiem negacji. Zapisana funkcja w postaci sumacyjnej jest równa:

$$y_1 = \underbrace{\overline{x_1}\,\overline{x_2}\,\overline{x_3}}_{0} + \underbrace{\overline{x_1}x_2x_3}_{3} + \underbrace{\overline{x_1}x_2\overline{x_3}}_{2} + \underbrace{x_1\overline{x_2}\overline{x_3}}_{4} + \underbrace{x_1\overline{x_2}x_3}_{5} + \underbrace{x_1x_2\overline{x_3}}_{6}$$

$$y_2 = \underbrace{\overline{x_1}\,\overline{x_2}\,\overline{x_3}}_{0} + \underbrace{\overline{x_1}\overline{x_2}x_3}_{1} + \underbrace{\overline{x_1}x_2\overline{x_3}}_{2} + \underbrace{x_1\overline{x_2}\overline{x_3}}_{4} + \underbrace{x_1\overline{x_2}x_3}_{5} + \underbrace{x_1x_2\overline{x_3}}_{6}$$

Podstawiając do powyższych wyrażeń wektor wejściowy 011 (wartość 3), otrzymamy:

$$y_1 = \overline{0}\ \overline{1}\ \overline{1} + \overline{0}\ 1\ 1 + \overline{0}\ 1\ \overline{1} + 0\ \overline{1}\ \overline{1} + 0\ \overline{1}\ 1 + 0\ 1\ \overline{1} = 0+1+0+0+0+0 = 1$$

$$y_2 = \overline{0}\ \overline{1}\ \overline{1} + \overline{0}\ \overline{1}\ 1 + \overline{0}\ 1\ \overline{1} + 0\ \overline{1}\ \overline{1} + 0\ \overline{1}\ 1 + 0\ 1\ \overline{1} = 0+0+0+0+0+0 = 0$$

Jak widać z podanego przykładu zaistnienie wektora o wartości 3 powoduje włączenie pompy nr 1, natomiast pompa nr 2 nie zostanie włączona.

Do opisu układów można również wykorzystać postać iloczynową, która jest wyznaczana dla wartości wyjściowej równej „0”. Suma logiczna obejmuje elementy wektora wejściowego. Jeżeli element wektora wejściowego dla danej kombinacji przyjmuje wartość logiczną „1”, to element ten zapisany jest jako negacja.

Postać iloczynowa dla podanego przykładu, w zapisie dziesiętnym, przyjmie postać:

$$y_1 = \prod M(1,7) + d(2,4,5,6)$$

$$y_2 = \prod M(3,7) + d(2,4,5,6)$$

Ostatecznie kanoniczną postać iloczynową można zapisać:

$$y_1 = \underbrace{(x_1 + x_2 + \overline{x_3})}_{1}\underbrace{(\overline{x_1} + \overline{x_2} + \overline{x_3})}_{7}\underbrace{(x_1 + \overline{x_2} + x_3)}_{2}$$
$$\underbrace{(\overline{x_1} + x_2 + x_3)}_{4}\underbrace{(\overline{x_1} + x_2 + \overline{x_3})}_{5}\underbrace{(\overline{x_1} + \overline{x_2} + x_3)}_{6}$$

$$y_2 = \underbrace{(x_1 + \overline{x_2} + \overline{x_3})}_{3}\underbrace{(\overline{x_1} + \overline{x_2} + \overline{x_3})}_{7}\underbrace{(x_1 + \overline{x_2} + x_3)}_{2}$$
$$\underbrace{(\overline{x_1} + x_2 + x_3)}_{4}\underbrace{(\overline{x_1} + x_2 + \overline{x_3})}_{5}\underbrace{(\overline{x_1} + \overline{x_2} + x_3)}_{6}$$

Minimalizacja funkcji logicznych

Przedstawiony sposób realizacji funkcji logicznych nie daje rozwiązań optymalnych. Dlatego opracowano liczne metody minimalizacji funkcji logicznych. Jedną z podstawowych metod są tablice Karnaugh'a.

Analiza układ, rozpoczyna się od zbudowania tablicy Karnaugh'a. Tablica posiada 2^n komórek, liczba wierszy i kolumn jest również potęgą liczby 2. W tablicy zapisane są wektory wejściowe jako potęga liczby dwa. Wektory wprowadzane są w kodzie Gray'a.

x_1 \ x_2, x_3	00	01	11	10
0				
1				

Powyżej przedstawiona jest tablica Karnaugh'a dla trzyelementowego wektora wejściowego. Podobnie można wyznaczyć tablicę dla większej ilość elementów wektora wejściowego. Przykład tablicy dla sześciu wejść podany jest poniżej.

abc \ def	000	001	011	010	110	111	101	100
000								
001								
011								
010								
110								
111								
101								
100								

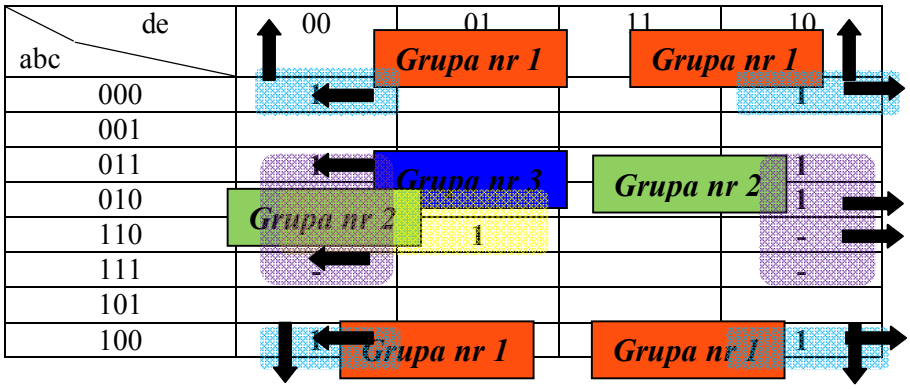
Tablica wyznaczana jest dla każdego wyjścia układu. Poniżej wyznaczone są tablice Karnaugh'a dla układu kombinacyjnego realizującego funkcję logiczną:

$$y = f(a,b,c,d,e) = \sum(0,2,8,9,1012,14,16,18,25) + d(24,26,28,30)$$

abc \ de	00	01	11	10
000	1			1
001				
011	1			1
010	1	1		1
110	-	1		-
111	-			-
101				
100	1			1

W polach tablicy umieszcza sie tylko wartości „1" i „kreski". Wartości logiczne odpowiadające „0" nie umieszcza się ze względu na przejrzystość tablicy. Jeżeli tablica jest zapisana dla funkcji logicznej w postaci iloczynowej, to w tablicy umieszcza się „0" i „kreski".

Po zbudowaniu tablicy można przystąpić do minimalizacji funkcji logicznej. W tablicy poszukujemy grup obejmujących jedynki i kreski. Grupa powinna posiadać wymiar odpowiadający potędze liczby dwa. Krawędzie tabeli dotykają się. Lewa krawędzi dotyka do prawej a górna dotyka do dolnej. Dlatego grupę można tworzyć z jedynek i kresek znajdujących się w dolnych i górnych wierszach oraz w lewej i prawej krawędzi.



W tabeli powyżej przedstawiono trzy grupy. Grupy, które obejmują sąsiadujące krawędzie zaznaczono strzałkami.

Grupa nr 1 obejmuje cztery jedynki, które znajdują się w rogach tablicy Karnaugh'a o wartościach:

abcde
00000
00010
10000
10010

Z analizy grupy wynika, że tylko elementy b, c i e posiadają tę samą wartość dla każdego elementu grupy. Dlatego do jednoznacznego opisu grupy można przyjąć funkcję logiczną:

$$Gr_1 = \bar{b}\bar{c}\bar{e}$$

Podobną analizę można przeprowadzić dla kolejnych grup.

Grupa nr 2	Grupa nr 3
abcde	abcde
01100	11000
01000	01000
11000	11010
11100	01010
01110	
01010	
11010	
11110	
$b\bar{e}$	$bc\bar{e}$

W wyniku minimalizacji funkcji logicznej, otrzymamy implikanty proste, których suma jest minimalną realizacją funkcji logicznej, w postaci:

$$y = f(a,b,c,d,e) = \bar{b}\bar{c}\bar{e} + b\bar{e} + \bar{a}b\bar{e}$$

Następnie wyznaczona zostanie minimalna postać iloczynowa dla kombinacyjnego układu sterującego pracą pomp napełniających zbiornik (rysunek 2.2). Funkcje iloczynowe, które opisują układ są następujące:

$$y_1 = f(x_1,x_2,x_3) = \prod(1,7) + d(2,4,5,6)$$

$$y_2 = f(x_1, x_2, x_3) = \prod(3,7) + d(2,4,5,6)$$

Zgodne z podaną wcześniej zasadą, dla postaci iloczynowej podajemy „0" i „kreski".

Tablica Karnaugh'a dla y_1.

x_1 \ x_2,x_3	00	01	11	10
0		0		-
1	-	-	0	-

W tabeli występują dwie grupy przedstawione poniżej.

Grupa nr 1	Grupa nr 2
x_1,x_2,x_3	x_1,x_2,x_3
001	100
101	101
	111
	110

Dla postaci iloczynowej wyznaczane są implicjenty proste, które dają minimalną postać iloczynową:

$$y_1 = f(x_1, x_2, x_3) = (x_2 + \overline{x_3})\overline{x_1}$$

Tablica Karnaugh'a dla y2.

x_1 \ x_2,x_3	00	01	11	10
0			0	-
1	-	-	0	-

Funkcja y_2 opisana jest przez jedną grupę czteroelementową.

x_1,x_2,x_3
011
010
111
110

Minimalna postać funkcji y_2 jest następująca:

$$y_2 = f(x_1, x_2, x_3) = \overline{x_3}$$

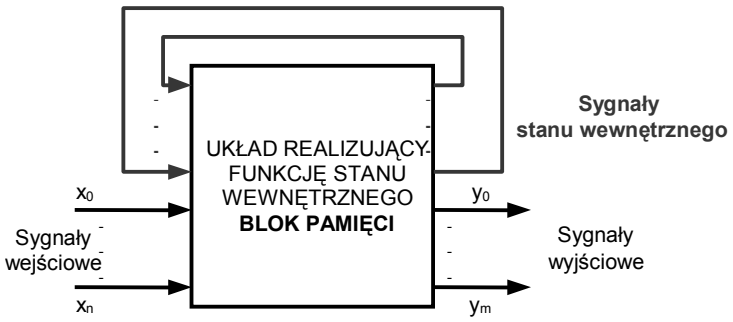
2.3. Układy sekwencyjne

Układ cyfrowy, w którym aktualny stan wyjść zależy nie tylko od aktualnego stanu wejść, ale również zależy od stanu,w którym układ znajdował się wcześniej, nazywamy układem sekwencyjnym lub układem z pamięcią (można spotkać określenie z pamięcią stanu).

Przykładem elementarnego układu sekwencyjnego jest układ przełączający, który służy do zaświecenia lampy stołowej z ręcznie uruchamianym przyciskiem o jednym tzw. położeniu stabilnym. Jeśli lampa nie świeci się, to naciśnięcie przycisków powoduje jej zaświecenie. W przypadku gdy lampa jest włączona, to naciśniecie przycisku powoduje wyłączenie lampy. Przyciśniecie przycisku powoduje włączenie lub wyłączenie lampy zależnie od tego czy wcześniej była włączona czy wyłączona.

W układach sekwencyjnych zależności między wejściami i wyjściami stają się niejednoznaczne. Tym samym wektorom wejściowym mogą odpowiadać różne wektory wyjść. Wartość na wyjściu zależy od „historii" układu - „pamięć stanu".

Pamięć realizowana jest przez wprowadzenie sprzężenia zwrotnego. Informacja o stanie, w którym znajduje się układ jest przekazywana na wejście układu. Układ ten nazywa się blokiem pamięci. Blok pamięci odpowiedzialny jest za realizację funkcji stanu układu.



Rysunek 2.3. Blok pamięci układu sekwencyjnego

UWAGA!
W układzie automatycznej regulacji występuje sprzężenie zwrotne do wyznaczenia uchybu regulacji. W układzie sekwencyjnym nie rozróżnia się dodatniego i ujemnego sprzężenia zwrotnego. Sprzężenie zwrotne w układzie sekwencyjnym, to rozszerzenie wektora wejść o dodatkowe elementy, którymi są wyjścia bloku pamięci.

Modelem układu sekwencyjnego jest automat skończony. Rozróżnia się dwa podstawowe typy automatów:

- automat Mealy'ego,
- automat Moore'a.

Automatem skończonym Mealy'ego nazywać będziemy układ:

$$M=<X,S,Y,\delta,\lambda>$$

gdzie:

$X=\{x_0,x_1,x_2, \ldots,x_n\}$ – wektor sygnałów wejściowych,

$S=\{s_0,s_1,s_2, \ldots,s_r\}$ – wektor stanów wewnętrznych,

$Y=\{y_0,y_1,y_2, \ldots,y_r\}$ – wektor sygnałów wyjściowych,

δ – funkcja przejść automatu Mealy'ego $\delta=S \times X$,

λ– funkcja wyjść automatu Mealy'ego $\lambda=S \times X$.

Równanie stanu automatu Mealy'ego:

$$S(t+1)=\delta(S(t),X(t)),$$

Równanie wyjść automatu Mealy'ego:

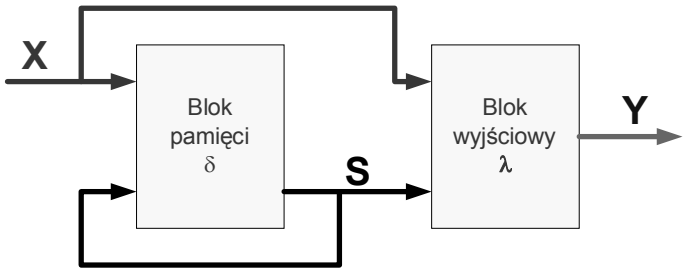
$$Y(t)=\lambda(S(t),X(t)).$$

W automacie Mealy'ego wartość stanu wewnętrznego zależy od bieżącej wartości stanu, w którym znajduje się automat oraz od sygnałów wejściowych:

$S(t+1)=\delta(S(t),X(t))$ - funkcja realizowana przez blok pamięci.

Wartość na wyjściu automatu zależy od stanu, w którym znajduje się automat oraz od wartości wyjściowej:

Y(t)=λ(S(t),X(t)) – funkcja realizowana przez blok wyjściowy.



Rysunek 2.4. Automat Mealy'ego

Automatem skończonym Moore'a nazywać będziemy układ:

$$\mu=<X,S,Y,\delta,\lambda>$$

gdzie:

X={x0,x1,x2, …,xn} – wektor sygnałów wejściowych,

S={s0,s1,s2, …,sr} – wektor stanów wewnętrznych,

Y={y0,y1,y2, …,yr} – wektor sygnałów wyjściowych,

δ – funkcja przejść automatu Moore'a,

λ– funkcja wyjść automatu Moore'a,

Równanie stanu automatu Moore'a:

$$S(t+1)=\delta(S(t),X(t)),$$

Równanie wyjść automatu Moore'a:

$$Y(t)=\lambda(S(t)).$$

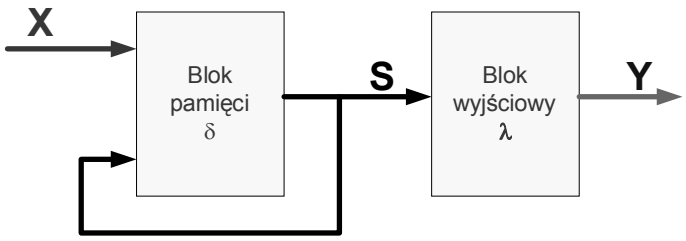
W automacie Moore'a wartość stanu wewnętrznego zależy od bieżącej wartości stanu, w którym znajduje się automat oraz od sygnałów wejściowych:

S(t+1)**=δ(S**(t)**,X**(t)**), -** funkcja realizowana przez blok pamięci.

Wartość na wyjściu automatu zależy od stanu, w którym znajduje się automat:

Y(t)=**λ(S**(t)**)** – funkcja realizowana przez blok wyjściowy.

Jest to podstawowa cecha odróżniająca automat Moore'a od automatu Mealy'ego.



Rysunek 2.5. Automat Moore'a

Układy sekwencyjne dzielimy na:

- układy sekwencyjne asynchroniczne,
- układy sekwencyjne synchroniczne.

W układach sekwencyjnych asynchronicznych zmiana stanu wewnętrznego następuje bezpośrednio i wyłącznie pod wpływem zmiany stanu wejść. Nowy stan wewnętrzny ustala się po pewnym czasie t określonym przez opóźnienie elementów, z których zbudowany jest układ realizujący funkcję stanu δ.

W układach synchronicznych zmiana stanu wewnętrznego może nastąpić tylko w ściśle określonych chwilach czasu, wyznaczonych przez sygnał doprowadzony do specjalnego wejścia układu. Wejście to, nazywane jest taktującym lub zegarowym i oznaczane jest literą C (ang. Clock). Stan wejść oddziaływuje na stan wewnętrzny automatu tylko w chwilach czasu, gdy wejście zegarowe jest aktywne. Zmiana stanu wejść, gdy wejście zegarowe jest nieaktywne nie powoduje zmiany stanu wewnętrznego układu.

Układ sekwencyjny opisywany jest przez:

- opis słowny,
- wykres czasowy,
- graf przejść i wyjść,
- tablica przejść i wyjść.

Opis słowny jest opisem działania układu, w którym podane są charakterystyczne informacje o wektorze wejściowym, stanach wewnętrznych układu i wektorze wyjściowym.

Wykres czasowy określa wzajemne zależności pomiędzy sygnałami wejściowymi i wyjściowymi. Każdemu sygnałowi przyporządkowane są wartości 0 lub 1. Oś czasu nie jest skalowana najczęściej przedstawia tylko zależności pomiędzy odpowiednimi sygnałami wejściowymi i wyjściowymi.

Tablica przejść opisuje funkcję przejść δ. W odpowiednich polach tabeli wpisuje się wartości następnych stanów. Pole określone jest przez wartość wektora wejściowego oraz stan bieżący.

Tablica wyjść, opisuje funkcję wyjść λ i jest różna zależnie od typu automatu. W automacie Mealy'ego wartość wektora wyjść wpisywana jest w te same pola, co tabela przejść, ponieważ wartość wyjściowa zależy od wektora wejść oraz od stanu układu. W automacie Moore'a generuje się oddzielną tabelę, w której umieszcza się wartości wyjściowe automatu odpowiadające odpowiednim stanom.

Należy zauważyć, że zawsze pierwotna tabela stanu i wyjść jest generowana dla automatu Moore'a i dopiero po wprowadzeniu kolejnych przekształceń wyznacza się tabele Mealy'ego albo pozostaje się przy automacie Moore'a.

Graf przejść i wyjść zawiera pełną informację o układzie. W grafie umieszczone są informacje o liczbie stanów wewnętrznych układu S i wektorze wejść oraz wyjść.

Wierzchołki grafu odpowiadają stanom wewnętrznym układu. Gałęzie grafu odpowiadają wektorowi wejść i opisują przejście pomiędzy dwoma stanami. Gałąź jest wyposażona w zwrot, który określa kierunek przechodzenia z bieżącego stanu do następnego. Tak opisywana jest funkcja przejść δ.

Stan wyjść w automacie opisuje się zależnie od typu automatu. W automacie Moore'a wartości wyjściowe zależą bezpośrednio od stanu, w którym znajduje się automat. Wartości wyjściowe bezpośrednio przyporządkowane są wierzchołkom grafu.

W automacie Mealy'ego wartości wyjściowe zależą od stanu, w którym znajduje się automat i od wektora wejściowego. Dlatego, w tego typu automatach wartości wektora wyjść umieszczone są w gałęziach obok wektora wejściowego.

Najprostszymi układami sekwencyjnymi są przerzutniki asynchroniczne. Przerzutnik tego typu posiada dwa wejścia:

- wejście wpisujące set (s),
- wejście zerujące reset (r).

Układ posiada wyjście Q oraz wyjście zanegowane notQ. Przerzutnik realizuje funkcję:

s	r	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	-

Q(t)→Q(t+1)		s	r
0	0	0	-
0	1	1	0
1	0	0	1
1	1	-	0

Projektowanie układów sterowania sekwencyjnego

Punktem wyjścia do projektowania układu asynchronicznego jest opis słowny, przebieg czasowy sygnałów wejściowych i wyjściowych, graf lub tabela przejść i wyjść.

Proces projektowania realizowany jest zgodnie z następującymi etapami:

1. Wyznaczenie grafu przejść i wyjść na odstawie opisu słownego lub przebiegów czasowych sygnałów wejściowych i wyjściowych.
2. Sporządzenie pierwotnej tabeli przejść i wyjść.
3. Redukcja pierwotnej tabeli przejść i wyjść.
4. Wyznaczenie funkcji przejść.
5. Wyznaczenie funkcji wyjść.

AB	Q^n		Q^{n+1}
	0	1	
00	1	1	1
01	0	1	Q^n
11	0	0	0
10	1	1	1

Proces projektowania zostanie przedstawiony na przykładzie układu, którego tabela przejść i wyjść podana jest powyżej. Układ będzie pracować synchronicznie zgodnie z taktami zegarowymi podawanymi na wejście dodatkowe „Clock". W układzie wartości wyjść równe są stanowi układu Q^n:

$$y = Q^n$$

W tabeli Q^{n+1} oznacza stan następny względem stanu Q^n. Zapis ten przedstawia następstwo stanów.

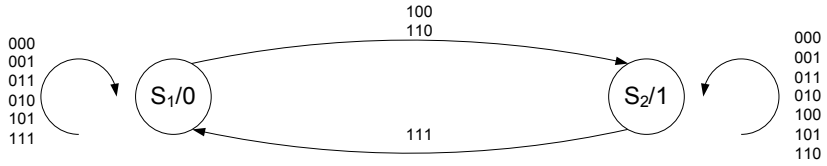
Po wprowadzeniu wejścia zegarowego otrzymamy tabelę:

Q^n	„Clock" A B							
	000	001	011	010	110	111	101	100
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**	**1**	**0**	**1**	**1**

Zgodnie z tabelą, jeżeli sygnał zegarowy ma wartość „0" to jest utrzymywany aktualny stan niezależnie od stanu wejść A i B. Jeżeli sygnał zegarowy „clock" przyjmuje wartość „1", to występują trzy przypadki:

1. dla wektora wejściowego (A, B) [1 0] i [0 0] na wyjściu układu będzie wartość „1",
2. dla wektora wejściowego (A, B) [1 1] na wyjściu układu będzie wartość „0",
3. dla wektora wejściowego (A, B) [0 1] na wyjściu układu będzie utrzymana wartość stanu, tak jak dla clock=0.

Układ posiada dwa stany wewnętrzne, które odpowiednio przyjmują wartość „0" i „1".



Rysunek 2.6. Graf

Na rysunku 2.6 przedstawiony jest graf przedstawiający przejścia między poszczególnymi stanami.

Korzystając z tabeli kodowania stanów dla przerzutnika asynchronicznego RS, można wyznaczyć funkcje wzbudzeń dla wejścia „set" i „reset".

Tabela dla wejścia ustawiającego S (set) jest następująca:

Q^n	„Clock" A B							
	000	001	011	010	110	111	101	100
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**
1	**-**	**-**	**-**	**-**	**-**	**0**	**-**	**-**

Funkcja wzbudzeń dla wejścia „set" jest następująca:

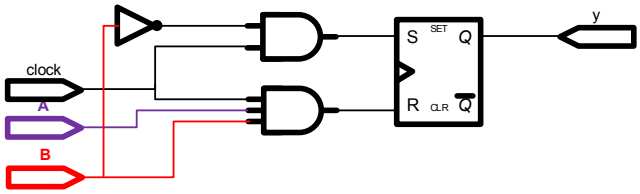
$$S^n = (Clock)\,\overline{B}$$

Tabela dla wejścia ustawiającego R (Reset) jest następująca:

Q^n	„Clock" A B							
	000	001	011	010	110	111	101	100
0	**-**	**-**	**-**	**-**	**0**	**-**	**-**	**0**
1	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**

Funkcja wzbudzeń dla wejścia „reset" jest następująca:

$$R^n = (Clock)\,AB$$



Rysunek 2.7. Realizacja układu sterowania

Podany przykład można zrealizować jako układ elektroniczny lub zaprogramować w sterowniku PLC (rozdział 3 – część II).

3 Podstawy programowania sterowników PLC

W tym rozdziale:

- Budowa sterowników PLC
- Adresowanie pamięci sterownika
- Języki programowania

3.1. Wstęp

W systemach automatyki wykorzystuje się sterowniki programowalne PLC (ang. *Programmable Logic Controller*).

Sterownik, to cyfrowy system elektroniczny do stosowania w środowisku przemysłowym, który posługuje się pamięcią programowalną do przechowywania zorientowanych na użytkownika instrukcji w celu sterowania przez cyfrowe lub analogowe wejścia i wyjścia szeroką gamą maszyn i procesów. Program zapisany w jego pamięci i wykonywany jest w sposób cykliczny.

Poniżej podano najważniejsze wydarzenia związane z opracowaniem i wdrażaniem sterowników przemysłowych.

1. Prace nad zbudowaniem sterowników rozpoczęto w latach 60-tych.
2. Pierwszy sterownik opracowano w 1969 roku.
3. Pierwszy komercyjny zbudowano w 1973 roku.
4. W 1976 roku wprowadzono sterowniki z wyposażone w kasety umożliwiające sterowanie rozproszone.
5. W 1970 roku opracowano pierwszy język typu LD.
6. W 1977 zastosowano w sterownikach PLC procesory 8080 z koprocesorami do wykonywania operacji logicznych.
7. Lata 80-te sukcesywne zastępowanie przekaźnikowych struktur logicznych przez sterowniki PLC.
8. Lata 90-te normalizacja terminologii i języków programowania przez wprowadzenie normy IEC 1131 a następnie 61131 (trzy części).
9. Wprowadzenie systemów SCADA (ang. Supervisory Control and Data Acquisition) w latach 90-tych.

Obecnie na rynku dostępnych jest kilkudziesięciu dostawców sterowników przemysłowych. Każdy z producentów oferuje własne oprogramowanie i sprzęt. Pomimo pewnych różnic ogólne zasady programowania

sterowników są takie same. Pewne uporządkowanie składni języków wprowadziła norma IEC 1131 a potem 61131, jednak drobne różnice szczególnie w adresowaniu i definiowaniu pojęć występują nadal.

Zawarty w podręczniku materiał wykładowy i ćwiczeniowy przygotowano w oparciu o sterownik PLC Simatic S7-200 i program Step7MicroWIN.

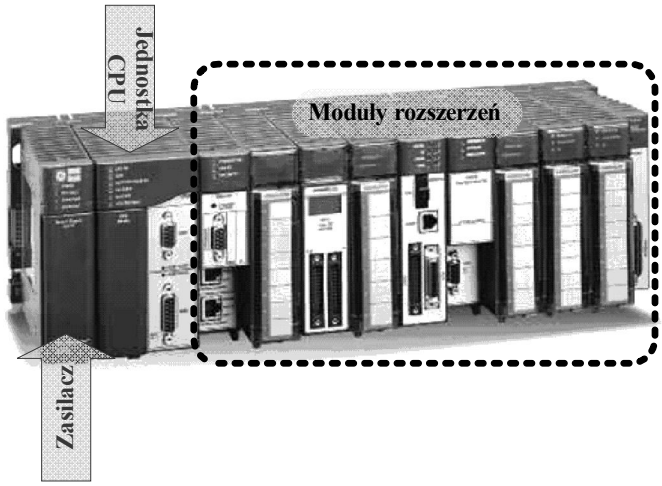
3.2. Budowa sterownika przemysłowego PLC

Sterowniki przemysłowe dzielą się na:

- sterowniki zintegrowane (Compact), które w ramach jednej obudowy zawierają niezbędne układy do prawidłowego funkcjonowania (rysunek 3.1),
- sterowniki modułowe, które składają się z wielu modułów współpracujących ze sobą (rysunek 3.2),
- sterowniki/sterowanie rozproszone, w którym poszczególne moduły systemu pracują niezależnie a dane wymieniane są przez sieci komputerowe (tzw. wyspy z modułami wejść i wyjść dyskretnych i analogowych oraz moduły specjalizowane).

Sterowanie rozproszone umożliwia nadzorowanie i automatyzację procesów rozłożone na bardzo dużej przestrzeni. Przykładem są systemy sterujące pracą instalacji wodno-kanalizacyjnych w miastach i rurociągi. Podobne rozwiązania wprowadza się w nowoczesnych systemach automatyzacji produkcji. Wprowadzenie tzw. wysp, czyli urządzeń, które zawierają moduły I/O i komunikacyjne, umożliwia rejestrację sygnałów analogowych i dyskretnych. Sygnały te przesyłane są przez sieć komputerową (również przesyłanie sygnałów falami elektromagnetycznymi np. w systemie UMTS) do jednostki CPU, która znajduje się w znacznej odległości od sterowanego elementu. Po przetworzeniu danych, jednostka CPU wysyła sygnały sterujące do modułów wyjściowych sterujących lokalnie urządzeniami wykonawczymi. W hierarchii systemu może być więcej jednostek CPU (jednostki nadrzędne, podrzędne oraz nadmiarowe). Systemy tego typu wykorzystują sieci Ethernet, systemy telekomunikacji cyfrowej (UMTS, HSDPA) i satelitarnej.

Rysunek 3.1. Sterownik zintegrowany



Rysunek 3.2. Sterownik modułowy

Sterownik przemysłowy zbudowany jest z trzech podstawowych elementów:

- jednostka CPU
- zasilacz
- moduły rozszerzające.

Sterowniki zintegrowane wyposażony jest we wszystkie niezbędne elementy do samodzielnego wykonywania zadania. Rozszerzeniu podlegają tylko dodatkowe wejścia, wyjścia i moduły komunikacyjne.

Moduł jednostki centralnej przeznaczony jest do:

- wykonania programu sterowania, przygotowanego przez użytkownika,
- zarządzania infrastrukturą sterownika, czyli wszystkimi modułami wchodzącymi w skład sterownika,
- komunikacji z innymi urządzeniami systemu sterownia procesem.

Budowa jednostki CPU odpowiada budowie klasycznego systemu mikrokomputerowego. W strukturze sytemu możemy wyróżnić mikroprocesor, pamięć danych i programu, układy wejść-wyjść, magistralę i oprogramowanie systemowe. Jednostka CPU i moduły dodatkowe zasilane są przez moduł zasilacza, który zapewnia niezbędne wartości napięć i moc. Sterowniki standardowo przygotowane są do zasilania z sieci prądu przemiennego 230 V 50 Hz lub prądu stałego 24 V.

W grupie modułów możemy wyróżnić następujące elementy:

- moduły wejść dyskretnych i analogowych,
- moduły wyjść dyskretnych i analogowych,
- moduły komunikacyjne,
- moduły pozycjonowania przeznaczone do sterowania napędami elektrycznymi (falowniki),
- moduły do obsługi termometrów rezystancyjnych i termoelektrycznych (termopary).

Można spotkać inne moduły np. specjalistycznych regulatorów, sterowania rozmytego, komunikacji radiowej. W ofercie producentów sterowników można znaleźć różne moduły dedykowane dla danego typu sterownika.

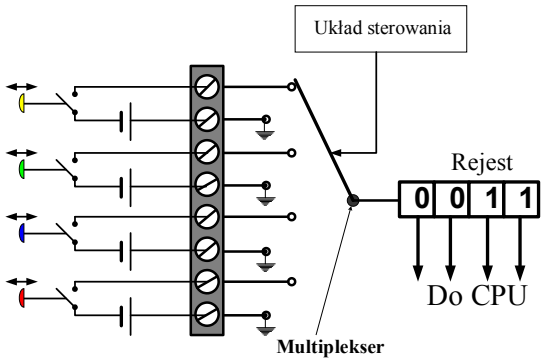
Moduł wejść dyskretnych

Moduł wejść dyskretnych DIM (Digital Input Modules) sterownika odpowiedzialny jest za dostarczanie do jednostki CPU informacji o sygnałach dwustanowych występujących w sterowanym procesie (napięcie lub prąd). Wartość zarejestrowana na wejściu zapisywana jest w polu bitowym rejestru wejściowego modułu DI. Rejestr ten znajduje się w odpowiednim miejscu przestrzeni adresowej. Na rysunku 3.3 przedstawio-

ny jest schemat modułu DI. Przedstawiony na rysunku multiplekser może być wyeliminowany przez zapis równoległy wartości poszczególnych wejść w rejestrze.

Moduł współpracuje z przełącznikami/włącznikami (ang. switch) – podstawowe elementy pulpitów sterowniczych. Jego stan zmieniany jest przez operatora. Przełączniki mają styki normalnie otwarte lub normalnie zamknięte. Konstrukcyjnie mogą znacznie się od siebie różnić (np. występuje lampka sygnalizacyjna, klucz włączający itd.). Odmianą wyłączników są wyłączniki krańcowe, których stan jest zmieniany w wyniku przemieszczenia lub siły wymuszonej przez układ stanowiący element procesu sterowania. Najczęściej są to przemieszczenia skrajne dlatego nazywa się je wyłącznikami krańcowymi.

Inną grupą z która współpracują moduły DIM są czujniki zbliżeniowe (ang. proximity switch), są to odpowiedniki wyłączników krańcowych, w których proces przełączenia następuje na drodze bezstykowej. Możemy wyróżnić czujniki indukcyjne, ultradźwiękowe, optyczne i pojemnościowe.



Rysunek 3.3. Moduł wejść dyskretnych

W modułach wejść binarnych wykorzystywane są dwustanowe sygnały napięcia:

- sygnał 0 V lub 24 V dla DC,
- sygnał 0 V lub 120 V 50 Hz,
- sygnał 0 V lub 120/230 V 50 Hz,

oraz dwustanowe sygnały prądu:

- sygnał 0 lub 4 mA DC,

- sygnał 4 lub 20 mA DC.

Moduł wejść analogowych

Zadaniem modułu wejść analogowych AI (ang. Analog Input Modules) jest rejestrowanie wielkości ciągłych występujących w procesie przemysłowym. Proces rejestracji wielkości ciągłych nie jest tak prosty jak w przypadku wielkości dyskretnych. Muszą w strukturze układu wejściowego pojawić się przetworniki analogowo-cyfrowe, które dokonają konwersji sygnału analogowego na sygnał cyfrowy.

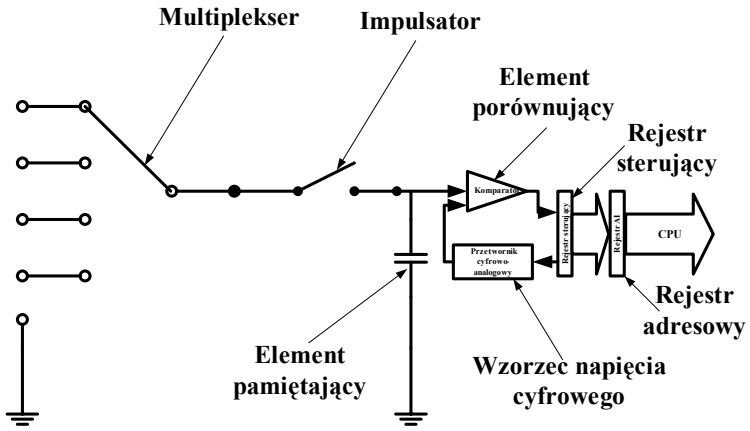
Wynik konwersji jest wpisywany do odpowiedniego rejestru, któremu przyporządkowany jest odpowiedni moduł analogowy. Każdy z modułów ma swój własny rejestr. Rejestry umieszczone są w przestrzeni adresowej sterownika. Każdorazowa komunikacja procesora z modułem odbywa się przez rejestr. W programie sterownika odwoływanie się do rejestru realizowane jest przez identyfikatory, które mogą być oznaczone:

- AIB - Analog Input Byte,
- AIW - Analog Input Word,
- AID - Analog Input Double Word.

Proces przetwarzania realizowany jest przez przetwornik analogowo - cyfrowy. Proces ten składa się z trzech etapów:

1. Próbkowanie.
2. Kwantowanie.
3. Kodowanie.

Ze względu na występowanie w układzie wielu wejść, ale tylko jednego przetwornika, występuje Multiplekser, który przełącza kolejne wejścia (rysunek 3.4).



Rysunek 3.4. Moduł wejść analogowo-cyfrowych

Najczęściej sygnałami wejściowymi modułów wejść analogowych są:

- sygnały napięciowe od 0 do 10 V DC,
- sygnały napięciowe od -5 do 5 V DC,
- napięcia rzędu kilku mV pochodzące np. z czujników termoelektrycznych.
- sygnały prądowe 4 do 20 mA DC,
- sygnały prądowe 0 do 20 mA DC,
- zmienne rezystancje.

Konstrukcja toru pomiarowego wejścia analogowego jest złożona, ze względu na dopasowanie poziomów napięć i prądów sygnałów wejściowych oraz częstotliwości sygnału wejściowego. Jednym z podstawowych ograniczeń jest częstotliwość graniczna przetwarzanego sygnału, która związana jest z częstotliwością próbkowania przetwornika analogowo-cyfrowego.

Moduł wyjść dyskretnych

Zadaniem modułu wyjść dyskretnych DO (Digital Outputs Modules) jest dostarczanie sygnałów dwustanowych do sterowanego procesu technologicznego. Sterowanie realizowane jest na

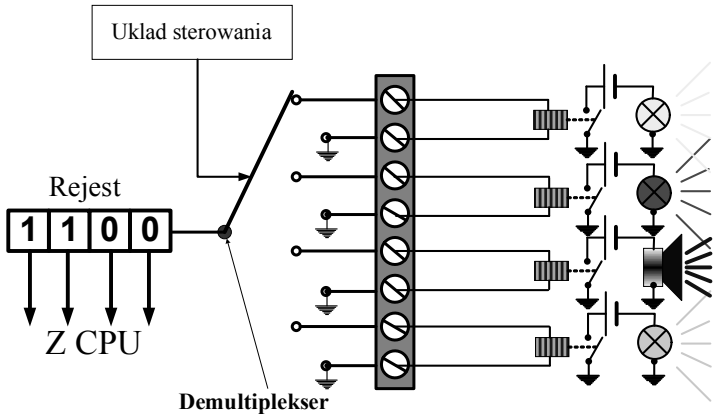
zasadzie włączenie/wyłączenie. Na rysunku 3.5 przedstawiony jest schemat modułu DO. W większości aplikacji demultiplekser zastąpiony jest przez przesłanie równoległe zawartości rejestrów na wyjście modułu DO.

Podczas pisania programu sterującego możliwe jest odwoływanie się do rejestrów modułu za pomocą identyfikatorów. Najczęściej spotyka się identyfikatory QB (Output Byte), QW (Output Word) i QD (Output Double Word).

Wyjścia binarne realizowane są przez łączniki, które zbudowane są z:

- tranzystorów,
- tyrystorów,
- przekaźników dwupołożeniowych.

Parametry elektryczne łącznika decydują o rodzaju i mocy urządzenia, którym można sterować.



Rysunek 3.5. Moduł wyjść dyskretnych

Moduł wyjść analogowych

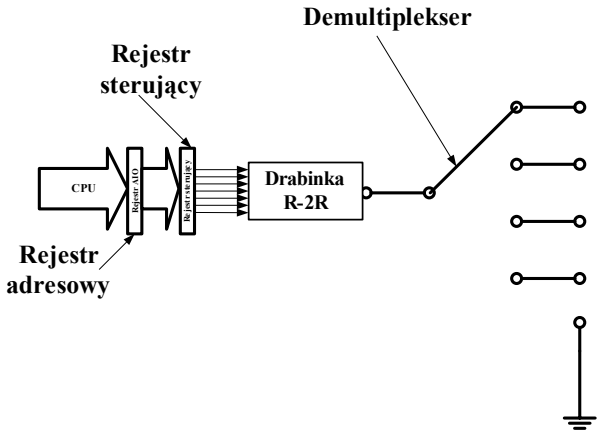
Zadaniem modułu wyjść analogowych AO (ang. Analogo Outputs Modules) jest wysyłanie sygnałów analogowych sterującym proce-

sem. Do wysyłania tych sygnałów wykorzystywane są przetworniki cyfrowo-analogowe. Przetwornik tego typu przetwarza kod binarny na sygnał ciągły. Kod binarny zapisany jest w rejestrze modułu wyjść analogowych. Rejestr związany z obsługą modułu znajduje się w przestrzeni adresowej sterownika. Odwołanie do rejestru realizowane jest przez identyfikatory.

Identyfikatory te to AQB (Analog Output Byte), AQW (Analog Output Word) i AQD (Analog Output Double Word). Na wyjściu modułu analogowego wykorzystuje się:

- sygnały napięciowe – od 0 do 10 V DC i -5 do 5 V DC,
- sygnały prądowe – od 4 mA do 20 mA i od 0 do 20 mA DC.

Sygnał wyjściowy może być demultipleksowany w module z kilkoma wyjściami.



Rysunek 3.6. Moduł wyjść analogowych

Cykl pracy sterownika

Sterownik przemysłowy pracuje w cyklu. Cykl obejmuje stałe czynności. Czynności te obejmują:

1. Odczytanie wejść. Sterownik kopiuje stan fizycznych wejść do rejestru wejściowego.

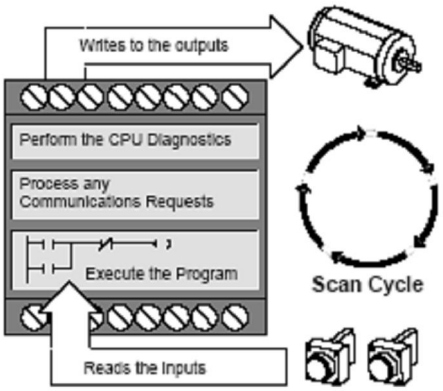
2. Wykonanie programu. Sterownik wykonuje program i gromadzi zmienne w przestrzeni adresowej.

3. Proces komunikacji na żądanie. Sterownik wykonuje zadania związane z komunikacją.

4. Wykonanie programu autodiagnostycznego. Sterownik uruchamia oprogramowanie firmowe, którego zadaniem jest przetestowanie sterownika, pamięci oraz modułów zewnętrznych.

5. Zapis wyjść. Wartości gromadzone w pamięci jako wartości wyjściowe przepisywane są z rejestru wyjściowego do fizycznych wyjść.

Sterowniki przemysłowe po zapisaniu programu w pamięci sterownika i uruchomieniu (tryb RUN), wykonują kolejne linie programu. Jeżeli program dojdzie do ostatniej linii programu, to sterownik powraca do pierwszej linii i dalej realizuje program. Taki cykl powtarza się do momentu kiedy sterownik przełączony zostanie do trybu STOP.



Rysunek 3.7. Cykl pracy sterownika

3.3. Przestrzeń adresowa sterownika PLC

W sterowniku występują obszary pamięci programu oraz danych. Obszar pamięci danych przydzielony jest do obsługi poszczególnych wejść, wyjść i dodatkowych modułów. Każdy obszar jest identyfikowany przez adres.

Reprezentacja liczby w sterowniku PLC

W sterowniku PLC posługujemy się liczbami które mogą być zapisane w systemie binarnym jako:

- Bajt – (B) Byte (8 – bitów),
- Słowo – (W) Word (16 – bitów),
- Słowo podwójnej precyzji - (D) DoubleWord (32 – bity),

Poszczególne liczby mogą reprezentować wartości:

- całkowite dodatnie (unsigned integer – bez znaku),
- całkowite (signed integer – ze znakiem),
- rzeczywiste (real).

W tabeli poniżej podano wartości jakie mogą przyjmować liczby w systemie dziesiętnym.

	Bajt (B)	Słowo (W)	Słowo podwójne (D)
Całkowita dodatnia	0÷255	0÷65535	0÷4294967295
Całkowita	-128÷127	-32768÷32767	-2147483648÷2147483647
Rzeczywista	Nie występuje	nie występuje	$\pm 3.402823e^{38}$

Adres w przestrzeni pamięci sterownika

W sterowniku przemysłowym S7-200 podstawową jednostką jest bajt podając adres zawsze podajemy numer bajtu. Adres składa się trzech pól:

- pierwsze pole oznacza obszar pamięci, do której odnosi się adres,

- drugie definiuje format adresu (B, W, D),

- trzecie podaje numer bajtu.

Adresowanie bajtu **VB100**

V	B	100
Obszar pamięci	Rozmiar8 - bitów	Numer bajtu

Oznacza przestrzeń pamięci przechowywania zmiennych. Adres odnosi się do bajtu o numerze 100.

Bajt 98
Bajt 99
Bajt 100
Bajt 101

Adresowanie słowa **VW100**

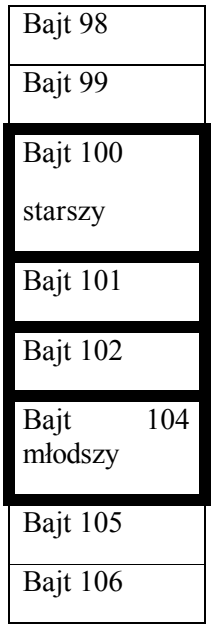
V	W	100
Obszar pamięci	Rozmiar16-bitów	Numer bajtu

Oznacza przestrzeń pamięci przechowywania zmiennych. Adres odnosi się do słowa, którego starszy bajt posiada adres 100, natomiast młodszy posiada adres 101.

Adresowanie słowa podwójnej precyzji **VD100**

V	D	100
Obszar pamięci	Rozmiar 32- bity	Numer bajtu

Oznacza przestrzeń pamięci przechowywania zmiennych. Adres odnosi się do słowa o podwójnej precyzji, którego najstarszy bajt posiada adres 100. Kolejne bajty posiadają adresy 101, 102 i 103.



Zawsze podajemy numer bajtu, natomiast format decyduje, które kolejne bajty będą przyporządkowane do adresu.

Jeżeli istnieje konieczność adresowania pojedynczego bitu, to format adresu ulega pewnej modyfikacji. Adres składa się z czterech elementów:

- pierwsze pole to obszar pamięci do której odnosi się adres,

- drugie podaje numer bajtu,

- trzecie pole to separator „ . „ (kropka),

- czwarte to numer bitu w bajcie.

Przy adresowaniu bitu nie podajemy formatu liczby.

V100.1

Adres odwołuje się do obszaru pamięci V, w której znajduje się bajt o adresie 100 i bit 1. Bity posiadają numery od 0 do 7. Bit o numerze 0 jest najmniej znaczącym (LS) natomiast o numerze 7 jest najbardziej znaczącym w bajcie (MS).

V	100	.	1
Obszar pamięci	Numer bajtu	separator	Nimer bitu

7	6	5	4	3	2	1	0
Bajt 99							
7 MS	6	5	4	3	2	1	0 LS
Bajt 100							
7	6	5	4	3	2	1	0
Bajt 101							

Wejścia dyskretne I

Sterownik S7-200 próbkuje fizyczne wejścia sterownika i zapisuje wyniki do rejestru wejściowego na początku każdego cyklu skanowania. Dostęp do rejestru wejściowego realizowany jest przez bit, bajt, słowo i podwójne słowo.

Bit	I[bajt].[bit]	I0.2
Byte, word, double word	I[size – rozmiar][początkowy bajt adresu]	IB4

Wyjścia dyskretne Q

Na koniec każdego cyklu, sterownik S7-200 przepisuje zgromadzone wartości w rejestrze wyjściowym do fizycznych wyjść sterownika.

Dostęp do rejestru wyjściowego realizowany jest przez bit, bajt, słowo i podwójne słowo.

Bit	Q[bajt].[bit]	Q0.2
Byte, word, double word	Q[size – rozmiar][początkowy bajt adresu]	QB4

Przestrzeń zmiennych V

Przestrzeń pamięci zmiennych V jest przeznaczona do gromadzenia zmiennych pośrednich między wynikami operacji będących wynikiem sterowania logicznego realizowanego przez program. Można użyć obszaru V do gromadzenia innych danych odnoszących się do procesu lub zadania. Dostęp do obszaru V pamięci realizowany jest przez bit, bajt, słowo i podwójne słowo.

Bit	V[bajt].[bit]	V10.2
Byte, word, double word	V[size – rozmiar][początkowy bajt adresu]	VW100

Przestrzeń zmiennych M

Przestrzeń pamięci M przechowuje bity, które sterują relacjami między pośrednimi wartościami operacji. Dostęp do obszaru M pamięci realizowany jest przez bit, bajt, słowo i podwójne słowo.

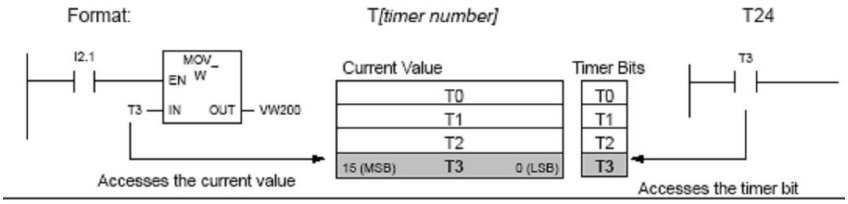
Bit	M[bajt].[bit]	M20.7
Byte, word, double word	M[size – rozmiar][początkowy bajt adresu]	MD20

Przestrzeń adresowa Timera T

Sterownik wyposażony jest w rejestry przechowujące zawartość timerów. Dostęp do timera jest przez składnie:

T[nr zegara]

Timer w przestrzeni adresowej opisany jest przez rejestr 16-to bitowy, który przechowuje bieżącą wartość czasu. Dodatkowo każdy timer posiada jeden bit, który sygnalizuje jego stan. Bit ten sygnalizuje przekroczenie zadeklarowanego przedziału czasu. Jednocześnie bit ten używany jest do sterowania. Więcej informacji o programowaniu timerów dostępne jest w materiałach ćwiczeniowych (Cześć II).



Rysunek 3.8. Rejestr i adres Timera

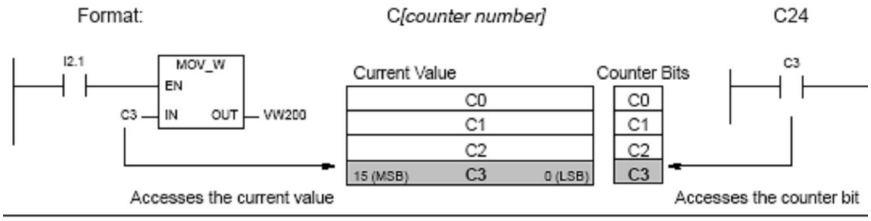
Przestrzeń adresowa licznika C

Sterownik wyposażony jest w liczniki, które zliczają zdarzenia. Każdy licznik ma przypisany w przestrzeni adresowej 16-to bitowy rejestr. Liczniki są adresowane zgodnie ze składnią:

C[nr licznika]

Licznik w przestrzeni adresowej opisany jest przez rejestr 16-to bitowy, który przechowuje bieżącą wartość liczby zarejestrowanych zdarzeń. Dodatkowo każdy licznik posiada jeden bit, który sygnalizuje jego stan. Bit ten sygnalizuje przekroczenie zadeklarowanej liczby zdarzeń.

Jednocześnie bit ten używany jest do sterowania. Więcej informacji o programowaniu liczników dostępne jest w materiałach ćwiczeniowych (Część II).



Rysunek 3.9. Rejestr i bit licznika

Przestrzeń adresowa wejść analogowych AI

Sterownik S7-200 zamienia wartości analogowe (napięcie, temperaturę) na 16-bitowe słowo. Dostęp do tych danych jest realizowany przez identyfikator AI oraz daną W i wartość startową adresu. Należy pamiętać, że mamy do czynienia ze słowami, które adresujemy przez bajty. Dlatego odwołania są do adresów parzystych (0, 2, 4,…).

AIW[adres początkowy byte] AIW4

Przestrzeń adresowa wyjść analogowych AQ

Sterownik S7-200 zamienia 16-bitowe słowo na wartość analogową. Dostęp do tych danych jest realizowany przez identyfikator AQ oraz daną W i wartość startową adresu. Należy pamiętać, że mamy do czynienia ze słowami, które adresujemy przez bajty. Dlatego odwołania są do adresów parzystych (0, 2, 4,…).

AQW[adres początkowy byte] AQW4

Oprócz podanych obszarów w pamięci sterownika można wyróżnić obsługę szybkich liczników, akumulatora, itp. Informacje o adresowaniu tych elementów przekraczają zakres podręcznika.

3.4. Języki programowania

Do programowania sterowników wykorzystuje się języki programowania, które dzielą się na dwie podstawowe grupy:

- tekstowe języki programowania,
- graficzne języki programowania.

W grupie języków tekstowych wyróżnia się język STL (Statement List), który jest w pewnym sensie analogiem asemblera. Język ten zawiera instrukcje logiczne, arytmetyczne, operacje relacyjne oraz złożone funkcje.

W grupie języków graficznych wyróżnia się:

- Język LAD (Ladder Diagram) – który swoją formą zapisu przypomina schemat układu stykowego. Umożliwia on tworzenie programu, w którym symbole graficzne zastępują instrukcje programu i operandy. W tej metodzie wykorzystuje się zdefiniowane bloki np. regulatora typu PID.
- Język FBD (Function Block Diagram) – program dla sterownika przygotowany jest wyłącznie z wykorzystaniem bloków funkcyjnych realizujących operacje matematyczne, logiczne oraz specjalne.
- Język SFC (Sequential Function Chart), gdzie wykorzystuje się metodę grafu do bezpośredniego programowania sterowników PLC.

W sterowniku S7-200 wraz z oprogramowaniem Step7/MicroWIN dostępne są języki programowania STL, LD i FBD. Zasady programowania w języku LD podane są w materiałach do ćwiczeń (Część II).

Literatura do części I

[1] Podręcznik SIMATIC S7-200, SIEMENS, Warszawa 2006.

[2] K. Kamiński: Programowanie w Step7 MicroWIN, 2006.

[3] W. Pełczewski „Teoria sterowania", WNT, Warszawa 1980.

[4] J. Kowal „Podstawy automatyki" Tom I i II, Uczelniane Wydawnictwo Naukowo – Dydaktyczne, Kraków 2007.

[5] W. Kwiatkowski „Podstawy teorii sterowania", BEL Studio, Warszawa 2007.

[6] Z. Seta „Wprowadzenie do zagadnień sterowania- wykorzystanie programowalnych sterowników logicznych PLC", Mikom, Warszawa 2002.

[7] T. Legierski, J. Wyrwał, J. Kasprzyk, J. Hajda, „Programowanie sterowników PLC", Pracownia Komputerowa Jacka Skalmierskiego, 2008.

[8] J. Kasprzyk, „Programowanie sterowników przemysłowych", WNT, Warszawa 2007.

Część II

Podstawy programowania sterowników PLC - ćwiczenia

1 Wprowadzenie do programu Step7/MicroWIN

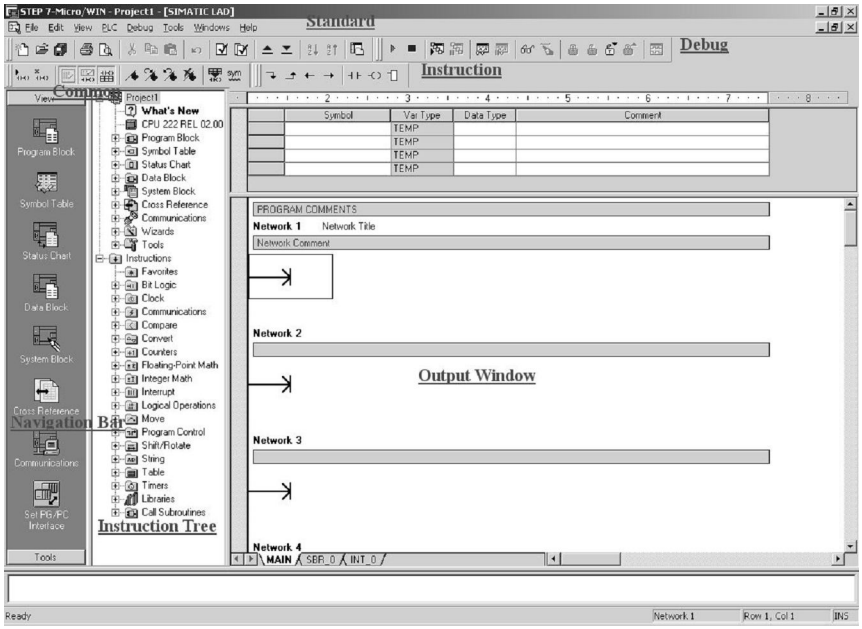
W tym rozdziale:

- Podstawy użytkowania programu STEP7/MicroWIN
- Wprowadzenie do języka programowania LD

1.1. Program Step7/MicroWIN

Do programowania sterowników PLC rodziny S7-200 wykorzystuje się program Step7/MicroWIN. Uruchomienie programu następuje po wybraniu ikony na pulpicie (rysunek 1.1). Po podwójnym kliknięciu powinno pojawić się okno programu (rysunek 1.2).

Rysunek1.1. Ikona programu.



Rysunek 1.2. Okno programu

W oknie programu (rysunek 1.2) występuje standardowe rozwijane menu oraz cztery paski narzędziowe i trzy ramki. Wyróżniamy paski:

- Standard,

- Debug,
- Common,
- Instruction.

Standard – pasek służy do wykonania takich czynności jak zapis i odczyt plików, cofanie czynności itd. Są to standardowe polecenia wykonywane w aplikacjach Windows. Na pasku znajdują się również narzędzia do kompilacji programu, zapisu i odczytu programu ze sterownika przemysłowego.

Debug – sterowanie i testowane programu. Korzystając z tego paska można prowadzić podgląd zmiennych procesowych.

Common – polecenia ułatwiające edytowanie i przygotowanie programu. Korzystając z tego paska można prowadzić podgląd zmiennych i adresów, co w określonych okolicznościach może ułatwić przygotowanie programu.

Instruction – pasek ułatwia wprowadzanie poleceń do programu. Wygląd paska zależy od języka programowania (tekstowy, drabinkowy i bloków funkcyjnych).

Programowanie sterowników ułatwiają trzy ramki:

- Navigation Bar,
- Instruction Tree,
- Output Window.

Navigation Bar – ramka nawigacyjna służy do przełączania zawartości ramki Output Window. Ponadto ramka służy do szybkiego uruchamiania narzędzi.

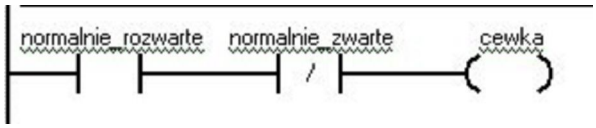
Instruction Tree – drzewo poleceń. Rozwijane menu, które umożliwia zarządzanie poleceniami. Zawartość ramki zależnie od używanego języka programowania.

Output Window – okno wyjściowe. Zawartość okna zmienia się zależnie od języka programowani oraz ustawień w ramce nawigacyjnej.

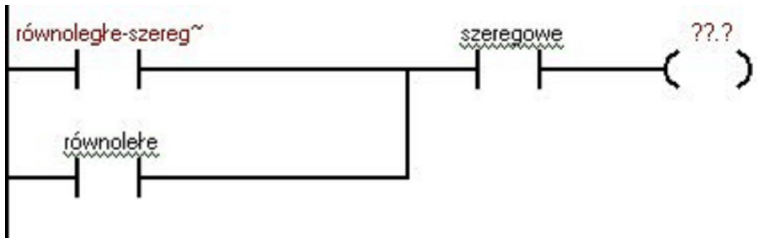
1.2. Programowanie w języku drabinkowym (LD)

Program napisany w języku LD przypomina schemat elektryczny, w którym prąd przepływa od lewej strony do prawej. Lewa krawędź okna można traktować jako źródło zasilania/listwę zasilającą. Do listwy przyłączone są elementy stykowe. Każdą linie utworzona przez elementy stykowe zamykamy cewką (rysunek 1.3). W linii może być dowolna liczba elementów stykowych, natomiast powinna być zamknięta przez jedną cewkę. Jeżeli chcemy wysterować więcej niż jedną cewką, to każdą kolejną dołączamy równolegle do linii.

Elementy stykowe dzielimy na kilka typów. Podstawowy podział to styki normalnie rozwarte i normalnie zwarte (rysunek 1.3). Kolejne elementy będą definiowane w następnych ćwiczeniach.



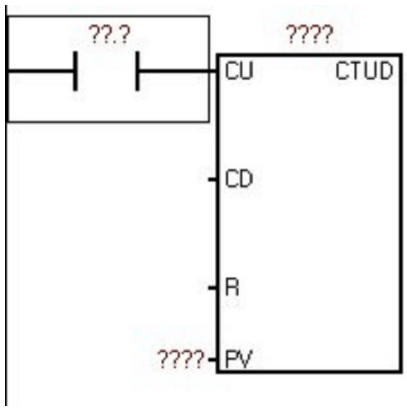
Rysunek 1.3. Normalnie zwarte i normalnie rozwarte styki



Rysunek 1.4. Złożone struktury układów stykowych

Elementy stykowe mogą być łączone w struktury szeregowe i równoległe. W gałęzi oprócz elementów stykowych i cewek występują bloki funkcyjne. Blok funkcyjny realizuje złożone funkcje np. licznika, timera, operacji matematycznej itd. Blok posiada wejścia i wyjścia. Zależnie od typu bloku liczba wejść i wyjść jest różna. W bloku występuje co najmniej jedno wejście. Wejście pierwsze, to zawsze sygnał **Enable**, który uaktywnia blok funkcyjny.

Dlatego wejście pierwsze musi być poprzedzone elementem stykowym. Brak elementu stykowego zostanie zasygnalizowany jako błąd podczas kompilacji programu.



Rysunek 1.5. Wejścia i wyjścia bloku funkcyjnego

W bloku funkcyjnym występują dwa typy wejść. Wejścia logiczne IN, które sterowane jest przez element stykowy. Tego typu wejście nie jest poprzedzone polem. Drugi typ wejść w bloku funkcyjnym poprzedzony jest ciągiem znaków zapytania. Oznacza to, że w takie pole należy wpisać adres, wartość lub nazwę zmiennej, którą można modyfikować przez manipulację w programie. Wejście tego typu określona jest jako IN_OUT i może być typu bajt, słowo, podwójne słowo, wartość rzeczywista. Typ wpisanej zmiennej zależy od bloku funkcyjnego. Podobnie definiowane są wyjścia z bloku. Dla wyjścia typu OUT, obowiązują zasady jak dla wejścia typu IN. Drugi typ wyjść jest typu IN_OUT.

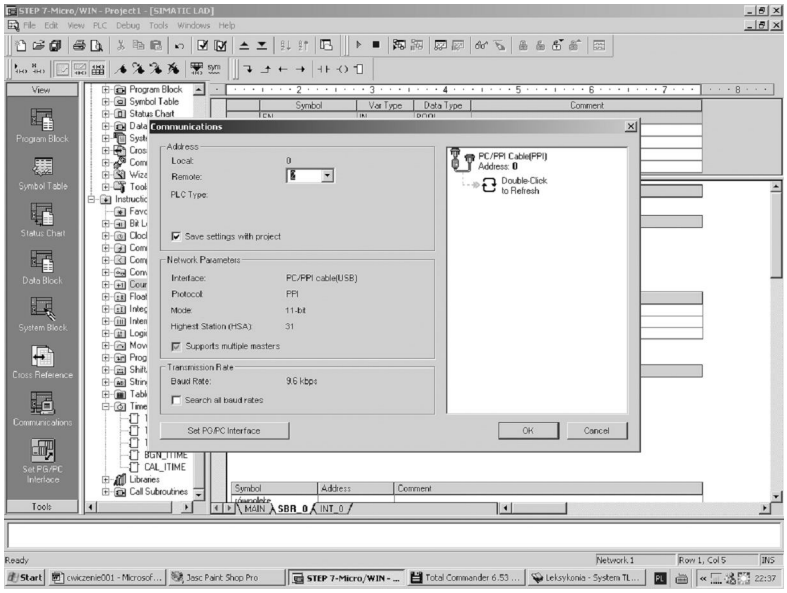
W programie każda gałąź określana jest jako NETWORK o odpowiednim numerze. Poszczególne NETWORKI odpowiadają kolejnym liniom programu.

1.3. Komunikacja ze sterownikiem i zapis programu w pamięci sterownika

Przygotowany program zapisujemy w pamięci sterownika. Przed zapisaniem programu należy nawiązać połączenie miedzy hostem (komputerem nadrzędnym) a sterownikiem.

Sterownik powinien być połączony kablem PPI z Hostem, w którym przygotowany jest program. Kabel jest przyłączony do portu USB. W starszych wersjach jest to port RS232.

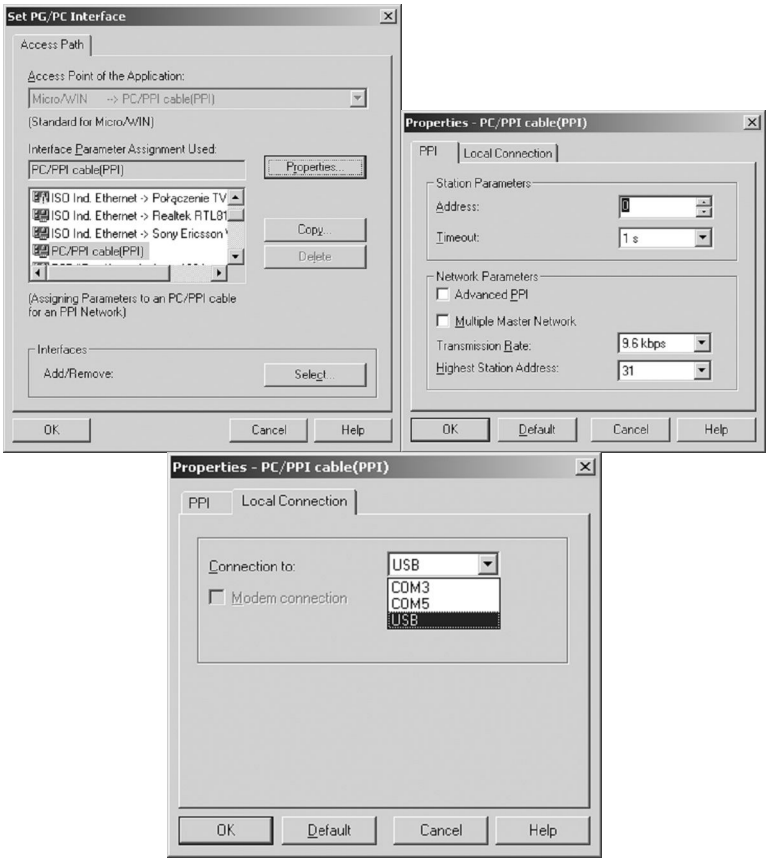
Z ramki nawigacyjnej wybieramy ikonę **Communications**. Jeżeli jest skonfigurowane połączenie to wystarczy kliknąć dwukrotnie na ikonę a nawiązane zostanie połączenie ze sterownikiem.



Rysunek 1.6. Ustanowienie połączenia

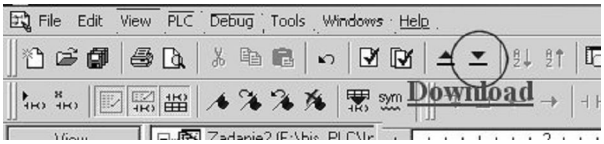
Połączenie można skonfigurować przez wybór pola **„Set PG/PC Interface”**. Następnie wybieramy protokół PC/PPI i uruchamiamy właściwości protokołu **„Properties”**. Następnie konfigurujemy parametry połączenie. Należy zmienić ustawienie portu na USB

ponieważ domyślnie ustawione jest połączenie przez port COM (rysunek 1.7).



Rysunek 1.7. Konfiguracja komunikacji PPI

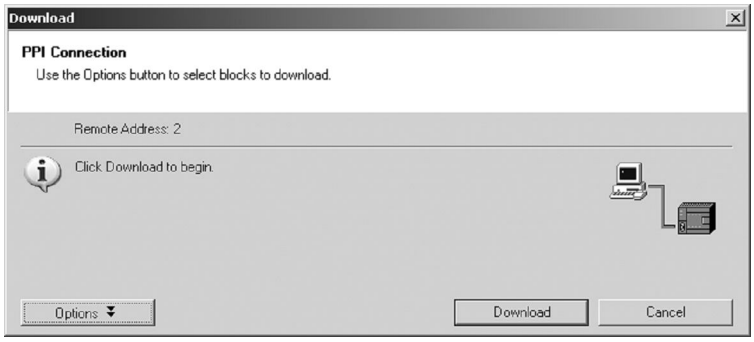
Po nawiązaniu połączenia można zapisać program w pamięci sterownika. Wybieramy polecenie zapis do sterownika z paska narzędziowego **Standard** (rysunek 1.8).



Rysunek 1.8. Zapis programu w sterownika

Po wybraniu narzędzia **Download** pojawi się okno, które umożliwia zapis programu w sterowniku. Ponownie wybieramy przycisk

Download program zostanie zapisany w pamięci sterownika (rysunek 1.9).



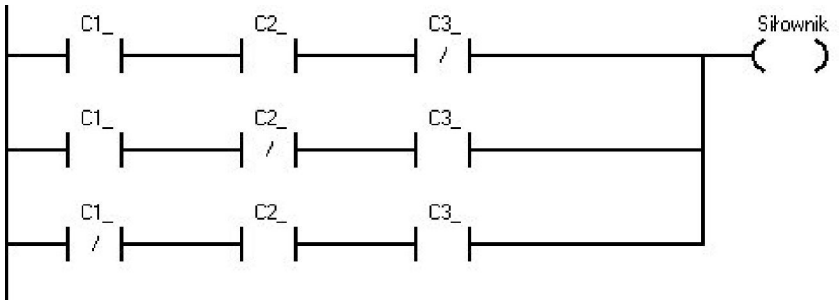
Rysunek 1.9. Zapis programu

1.4. Uruchomienie programu

Przygotujemy program dla sterownika, który ma realizować proces sterowania ruchem wykrojnika. Praca układu polega na wysunięciu i cofaniu tłoczyska siłownika jednostronnego działania po spełnieniu określonych warunków. Siłownik ma za zadanie przesunąć wykrojnik, jeżeli pod wykrojnikiem znajduje się detal. Detal podstawiany może być z trzech kierunków. Wykrycie detalu sygnalizują dwa z trzech czujników. Włączenie czujnika C1 i C2 oznacza kierunek I, C2 i C3 to kierunek II, natomiast C1 i C3 to kierunek III.

			Symbol	Address	Comment
1			C1_	I0.0	czujnik 1
2			C2_	I0.1	czujnik 2
3			C3_	I0.2	czujnik 3
4			Siłownik	Q0.0	starowanie siłownika
5					

Rysunek 1.10. Wejścia i wyjścia oraz ich adresy

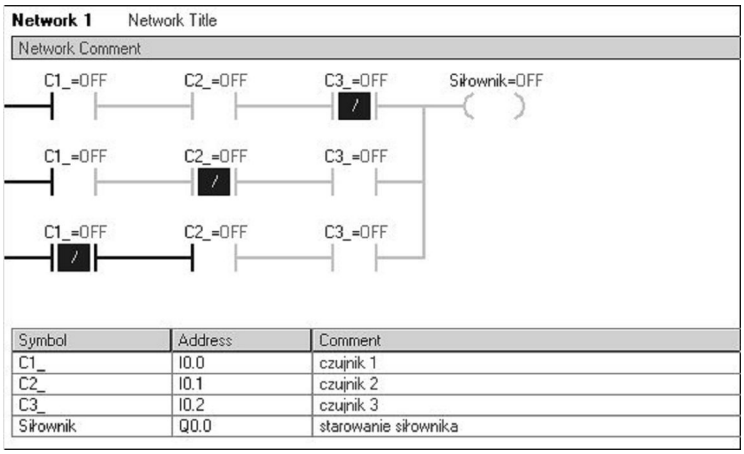


Rysunek 1.11. Program realizujący zadanie

Program realizujący powyższe zadanie przedstawiony jest na rysunku 1.11. Po zapisaniu programu w pamięci sterownika należy uruchomić sterownik. W pasku narzędziowym **Debug** znajduje się polecenie **RUN**. Obok znajduje się polecenie **STOP** (rysunek 1.12).

Rysunek 1.12. Polecenie RUN

Sterownik zacznie realizować program. Program można przetestować analizując zmienne procesowe w trybie on-line (rysunek 1.13). Prace w trybie on-line można włączyć wybierając przycisk „**Program Status**", który znajduje się obok przycisku „**STOP"** (rysunek 1.12).



Symbol	Address	Comment
C1_	I0.0	czujnik 1
C2_	I0.1	czujnik 2
C3_	I0.2	czujnik 3
Siłownik	Q0.0	starowanie siłownika

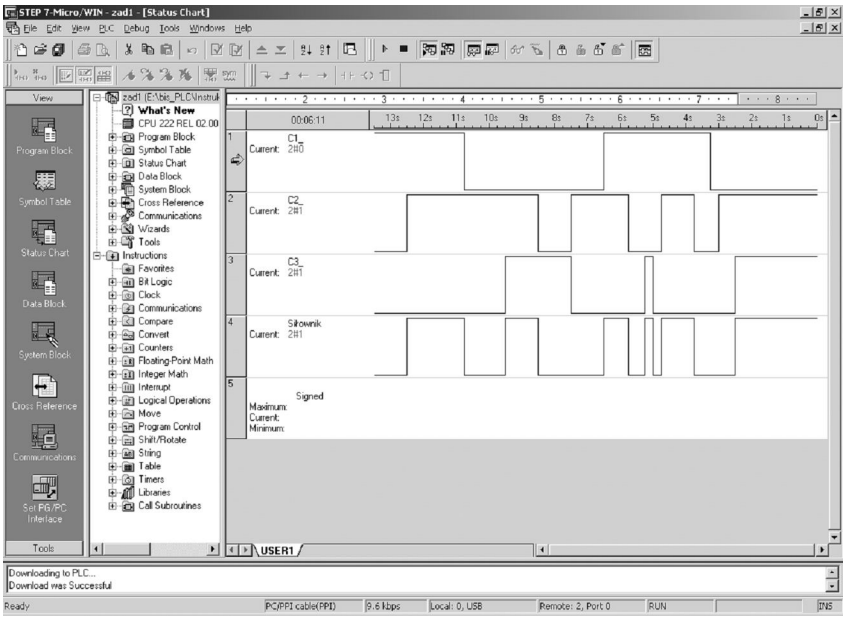
Rysunek 1.13. Podgląd parametrów „Program Status"

Można prowadzić podgląd parametrów i rejestrować ich zmianę w czasie. W tym celu z ramki nawigacyjnej wybieramy **„Status Chart**". W **Output Window** pojawi się okno, w którym zaznaczamy zmienne, które poddamy analizie (rysunek 1.14).

	Address	Format	Current Value	New Value
1	C1_	Bit		
2	C2_	Bit		
3	C3_	Bit		
4	Siłownik	Bit		
5		Signed		

Rysunek 1.14. Status Chart

W oknie (rysunek 1.14) można odczytać aktualne wartości parametrów oraz wymuszać ich zmianę (Force). Ponadto możemy obserwować zmianę parametrów wykorzystując do tego „**Trend View**" (rysunek 1.15).



Rysunek 1.15. Status Chart – Trend View

1.5. Zadanie do rozwiązania

Proszę przeprowadzić analizę programu, który steruje ruchem szczęki imadła. Prac polega na wysunięciu i cofaniu tłoczyska siłownika dwustronnego działania przy odpowiednim ustawieniu przycisków. Siłownik dociska szczękę, kiedy naciśnięto przycisk START i puszcza, gdy naciśnięto przycisk STOP. Program musi tak działać, aby wyższy priorytet posiadał przycisk START. Oznacza to, że jak długo jest włączony przycisk START tak długo naciśnięcie przycisku STOP nie zmniejsza docisku szczęk.

2 Układy kombinacyjne

W tym rozdziale:

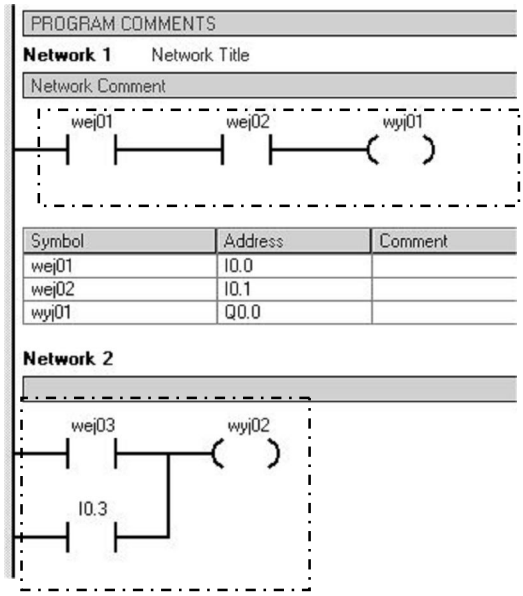
- Podstawy programowania funkcji logicznych
- Minimalizacja funkcji logicznych
- Sterowanie kombinacyjne

2.1. Wprowadzenie

Każdy program składa się z szeregu funkcji logicznych (kombinacji styków) umieszczonych w pamięci sterownika. Styki mogą być połączone:

- szeregowo (AND),
- równolegle (OR).

Na rysunku 2.1 przedstawiono szeregowe połączenie styków w **Network 1** oraz równoległe w **Network 2**.



Rysunek 2.1. Połączenie styków szeregowe i równolegle

Prawie wszystkie zadania sterujące można zrealizować za pomocą podstawowych funkcji logicznych. Do najczęściej realizowanych funkcji logicznych w układach sterujących należą funkcje koniunkcji i al.terna-tywy.

Koniunkcja AND (i) – stosowana w tych układach, w których pojawia się wartość na wyjściu Y, jeśli pojawiły się obydwa sygnały wejściowe X_1 i X_2.

a	**b**	**a** AND **b**
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

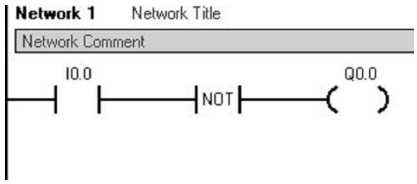
Alternatywa OR (lub) – stosowana w tych układach, w których pojawia się wartość na wyjściu Y, jeśli pojawił się którykolwiek z sygnałów wejściowych X_1 lub X_2.

a	**b**	**a** or **b**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

Funkcja logiczna AND odpowiada szeregowemu podłączeniu styków na schemacie drabinkowym. W tradycyjnym obwodzie elektrycznym funkcja iloczynu logicznego AND jest równoważna szeregowemu połączeniu elementów.

Funkcja logiczna OR odpowiada równoległemu podłączeniu styków na schemacie drabinkowym. W tradycyjnym obwodzie elektrycznym funkcja sumy logicznej OR jest równoważna równoległemu połączeniu elementów.

Oprócz funkcji logicznych AND i OR w algebrze Boole'a występuje operacja NOT. W sterownikach S7-200 jest realizowana jako specjalny styk.



a	NOT(a)
0	1
1	0

Rysunek 2.2. Funkcja NOT

Operacja logiczna NOT jest operacją jednoargumentową. Na rysunku 2.2 przedstawiony jest sposób zanegowania wejścia o adresie I0.0. Jeżeli na wejściu I0.0 jest wartość logiczna 1, to na wyjściu otrzymamy wartość logiczną 0.

Korzystając z przedstawionych powyżej funkcji podstawowych i praw funkcji logicznych (prawo de Morgana, suma modulo 2 i równoważność) można wyznaczyć operacje NAND, NOR, XOR i NXOR.

NAND: $\overline{a} + \overline{b} = \overline{a \cdot b}$

a	**b**	**a** NAND **b**
0	**0**	**1**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

NOR: $\overline{a} \cdot \overline{b} = \overline{a + b}$

a	**b**	**a** NOR **b**
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

XOR: $a \cdot \overline{b} + \overline{a} \cdot b = a \oplus b$

a	**b**	**a** XOR **b**
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**0**

NXOR: $\overline{a} \cdot \overline{b} + a \cdot b = \overline{a \oplus b}$

a	**b**	**a** NXOR **b**
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

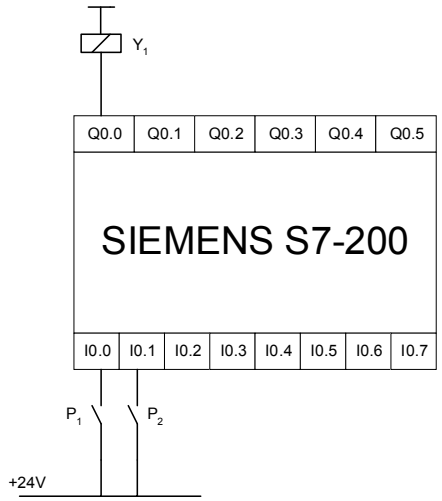
2.2. Zadania do przeanalizowania

Zadanie 1

Zawór elektromagnetyczny Y1 prasy hydraulicznej otwiera dopływ cieczy po włączeniu przycisków P1 i P2.

Rozwiązanie:

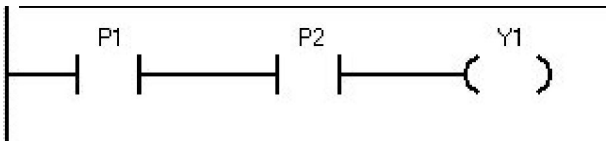
1. Schemat połączenia elementów ze sterownikiem:



2. Lista zmiennych:

Adres	NAZWA	KOMENTARZ
I0.0	P1	przycisk 1
I0.1	P2	przycisk 2
Q0.0	Y1	zawór elektromagnetyczny

3. Program w języku drabinkowym LD

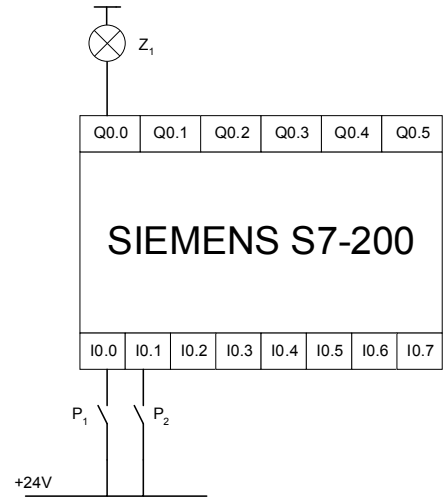


Zadanie 2

Żarówka Z1 zaświeci się po włączeniu przycisku P1 lub P2.

Rozwiązanie:

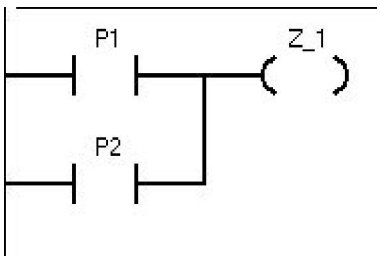
1. Schemat układu sterowania:



2. Lista zmiennych:

Adres	NAZWA	KOMENTARZ
I0.0	P1	przycisk 1
I0.1	P2	przycisk 2
Q0.0	Z1	żarówka

3. Program w języku drabinkowym LD

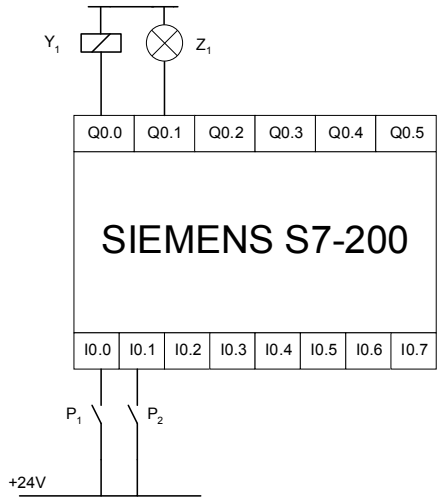


Zadanie 3

Zawór elektromagnetyczny Y1 prasy hydraulicznej otwiera dopływ cieczy po włączeniu przycisku P1 i P2. Załączenie obu przycisków jest zasygnalizowane zapaleniem żarówki Z1.

Rozwiązanie:

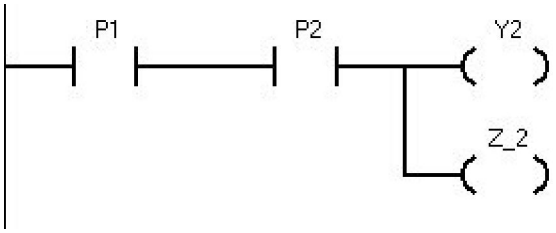
1. Schemat połączenia elementów ze sterownikiem:



2. Lista zmiennych:

Adres	NAZWA	KOMENTARZ
I0.0	P1	przycisk 1
I0.1	P2	przycisk 2
Q0.0	Y1	zawór elektromagnetyczny
Q0.1	Z1	żarówka

3. Program w języku drabinkowym LD



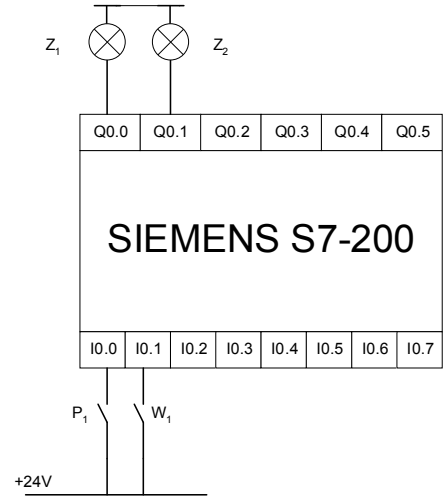
Zadanie 4

Wciśnięcie i ciągłe przytrzymanie przycisku P1 powoduje wysuw tłoczyska siłownika. Osiągnięciu przez tłoczysko siłownika położenia krańcowego sygnalizowane jest zapaleniem żarówki Z1. W przypadku, gdy tłoczysko siłownika nie osiągnie położenia krańcowego zapala się żarówka Z2.

Krańcowe położenie tłoczyska siłownika wyznacza łącznik krańcowy W1.

Rozwiązanie:

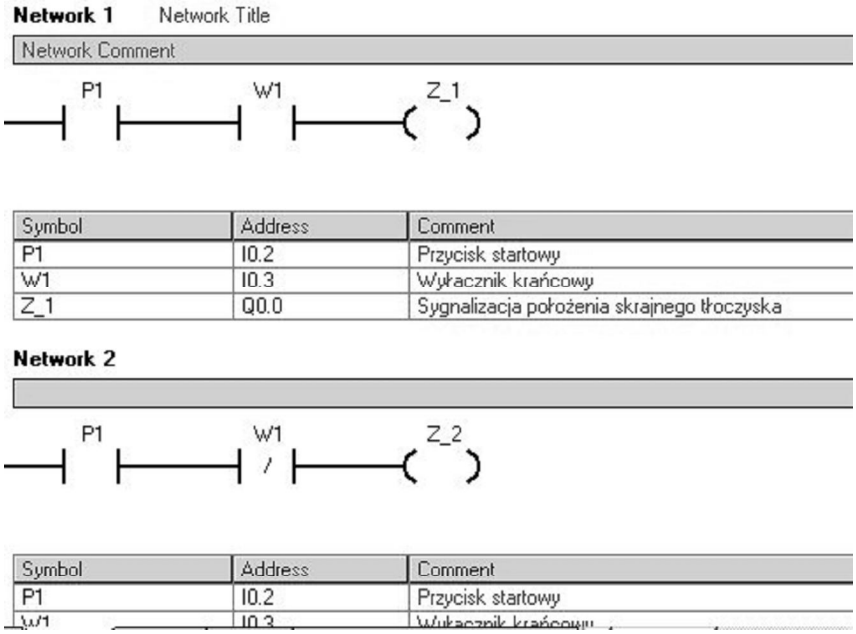
1. Schemat układu sterowania:



2. Lista zmiennych:

Adres	NAZWA	KOMENTARZ
I0.0	P1	przycisk 1
I0.1	W1	włącznik krańcowy 2
Q0.0	Z1	żarówka 1
Q0.1	Z2	żarówka 2

3. Program w języku drabinkowym LD



2.3. Zadania do zrealizowania

Zadanie 1

Poziom cieczy w zbiorniku jest mierzony przez czujnik progowy C1. Po przekroczeniu dopuszczalnego poziomu cieczy w zbiorniku, czujnik C1 wysyła sygnał sterujący, który włącza pompę Pom1. Pompa usuwa wodę ze zbiornika. Jeżeli poziom wody w zbiorniku spadnie poniżej pewnego poziomu pompa zostanie wyłączona (czujnik posiada histerezę). Proszę napisać program realizujący powyższe zadanie.

Zadanie 2

W zbiorniku mierzony jest poziom cieczy przez czujnik C1 podobnie jak w zadaniu 1. Silnik pompy jest zabezpieczony przez wyłącznik termiczny, który zabezpiecza pompę przed uszkodzeniem T1. Proszę zmodyfikować tak program, aby silnik pompy został wyłączony przez

zabezpieczenie termiczne (Termik). Sygnał z termika podawany jest na wejście sterownika. Uruchomienie pompy i zadziałanie zabezpieczenia termicznego sygnalizowane jest przez lampki.

Zadanie 3

Układ regulacji poziomu cieczy w zbiorniku jest wyposażony w cztery czujniki poziomu C1, C2, C3, C4 oraz cztery pompy o różnej mocy Pom1, Pom2, Pom3, Pom4. Czujniki poziomu są umieszczone, co 5 [cm]. Jeśli zadziała pierwszy czujnik C1 to włączana jest najsłabsza pompa Pom1. Gdy zadziałają jednocześnie czujniki C1 i C2 załączana jest mocniejsza pompa Pom2 a wyłączana zostanie słabsza pompa Pom1. Kiedy zadziałają trzy czujniki C1, C2 i C3 włączana zostanie trzecia pompa Pom3 pozostałe są wyłączone, itd. Należy napisać program realizujący powyższe zadanie.

Zadanie 4

Napisać program, który umożliwi zamianę liczby załączanych przycisków P1, P2, P3 na numer zapalanej żarówki Z1, Z2 lub Z3 (np. gdy załączymy dwa dowolne przyciski, to fakt ten powinien zostać zasygnalizowany zapaleniem żarówki Z2).

Zadanie 5

Proszę napisać program, który na wyjściu podaje liczbę binarną (poczynając od najemnie znaczącego bitu Q0.0) oznaczającą liczbę załączonych wejść.

Zadanie 6

Proszę napisać program, który na wyjściu sterownika wyświetla słowo binarne będące sumą dwu czterobitowych liczb zapisanych w reprezentacji uzupełnienia do dwóch (pierwsze słowo {4 bity} I0.3 ÷I0.0 i drugie słowo {4 bity} I0.7÷I0.4). Na wyjściu należy uwzględnić bit przeniesienia.

Zadanie 7

Proszę zrealizować następujące funkcje logiczne:

Zestaw 1

$$f(a,b,c,d,e,f) = \sum m(1,5,9,13,21,23,29,31,37,45,53,61)$$
$$f(a,b,c,d,e,f) = \sum m(2,4 \div 6,12 \div 21,28 \div 31,34,38,50,51,60 \div 63)$$

Zestaw 2

$$f(a,b,c,d,e,f) = \sum m(0,2,4,6 \div 8,10 \div 14,16,18,19,29,30)$$
$$f(a,b,c,d,e,f) = \sum m(4,5,10,12,13,16,17,,21,25 \div 27,29)$$

Zestaw 3

$$f(a,b,c,d,e,f) = \prod M(0,1,3,4,9,10,13,14,16,19,20,23,25,26,29,30)$$
$$f(a,b,c,d,e,f) = \sum m(0,1,3,5,7,15,19,21) + d(31)$$

Zestaw 4

$$f(a,b,c,d,e,f) = \sum m(4,6,7,9,11 \div 15,20,22,25,27,28,30) + d(1,5,29,31)$$
$$f(a,b,c,d,e,f) = \sum m(0,1,3,7,15,16,18,19,23) + d(31)$$

Zestaw 5

$$f(a,b,c,d,e,f) = \sum m(0 \div 4,6,8,11,12,27,28) + d(9,16 \div 19)$$

$$f(a,b,c,d,e,f) = \prod M(0,1,3,4,9,10,13,14,16,19,20,23,25,26,29,30)$$

Zestaw 6

$$f(a,b,c,d,e,f) = \prod M(0,1,3,4,9,10,13,14,16,19,20,23,25,26,29,30)$$
$$f(a,b,c,d,e,f) = \sum m(4,5,10,12,13,16,17,,21,25 \div 27,29)$$

Zestaw 7

$$f(a,b,c,d,e,f) = \sum m(0,2 \div 6,12,19,20,24,28) + d(1,13,16,29,31)$$

$$f(a,b,c,d,e,f) = \sum m(2,4 \div 6,12 \div 21,28 \div 31,34,38,50,51,60 \div 63)$$

Zestaw 8

$$f(a,b,c,d,e,f) = \sum m(5,7,13,15,16,20,25,27,29,31)$$

$$f(a,b,c,d,e,f) = \sum m(0,1,3,5,7,15,19,21) + d(31)$$

3 Układy z pamięcią stanu

W tym rozdziale:

- Podstawy programowania układów sekwencyjnych.
- Układy sterowania sekwencyjnego.

3.1. Wprowadzenie

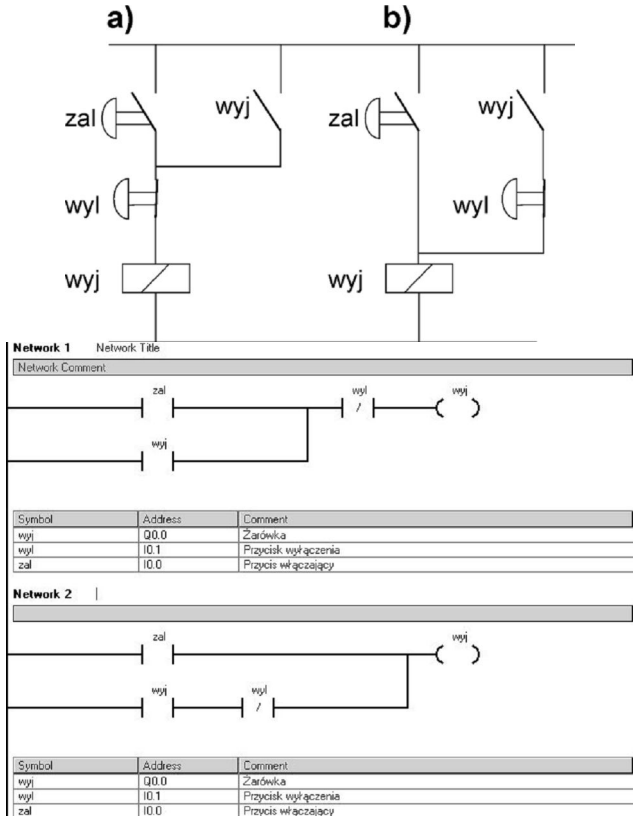
Układy pamięci, czyli układy z samopodtrzymaniem stosuje się do zapamiętania stanu załączenia cewki stycznika, przekaźnika czyli elementów wykonawczych. Na rysunku 3.1 przedstawiono dwie podstawowe realizacje układu z samopodtrzymaniem:

a. układ stosowany dla pierwszeństwa wyłączenia – pamięć RS z priorytetem kasowania,

b. układ stosowany dla pierwszeństwa załączenia – pamięć RS z priorytetem zapisu.

Adres	NAZWA	KOMENTARZ
%I0.0	zal	przycisk włączający
%I0.1	wyl	przycisk wyłączający
%Q0.0	wyj	żarówka

Adres	FUNKCJA
%I0.0	wejście zapisujące SET
%I0.1	wejście kasujące RESET
%Q0.0	wyjście pamiętające stan wysoki na %I0.0



Rysunek 3.1. Realizacja układu z samopodtrzymaniem

W momencie naciśnięcia przycisku zal, w wersji b) następuje bezpośrednie zapalenie żarówki wyj, natomiast w wersji a) pośrednie poprzez przycisk rozwierny wyl (styk normalnie zamknięty).

Naciśnięcia przycisku zal powoduje pojawienie się stanu wysokiego (”1”) na wyjściu Q0.0 a co za tym idzie zapalenie żarówki wyj. Styk pomocniczy Q0.0 normalnie otwarty zmienia swój stan na zwarty (zamyka się) dzięki czemu umożliwia dalszy przepływ prądu na wyjście Q0.0, pomimo że przestaliśmy naciskać przycisk zal. Zaniknięcie stanu wysokiego na wejściu I0.0 nie wpływa na stan wyjścia Q0.0. Działanie tego typu nazywamy „samopodtrzymaniem” wyjścia Q0.0. Wyłączenie żarówki wyj nastąpi po naciśnięciu przycisku wyl. Jednoczesne włączenie przycisków zal i wyl nie włączy żarówki wyj.

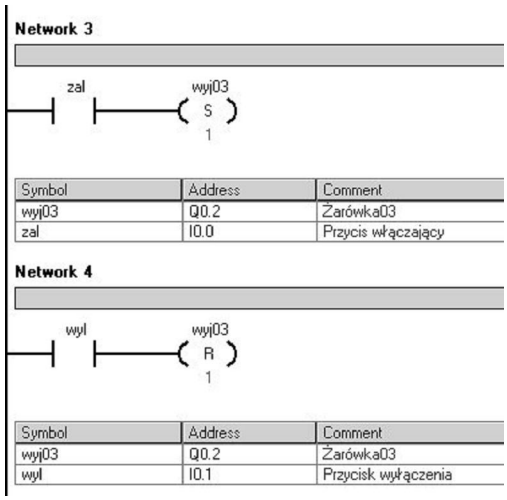
Działanie układu z samopodtrzymaniem z priorytetem zapisu jest analogiczne do działania układu z priorytetem kasowania. Jednak w tym

przypadku jednoczesne włączenie przycisków zal i wyl spowoduje włączenie żarówki wyj.

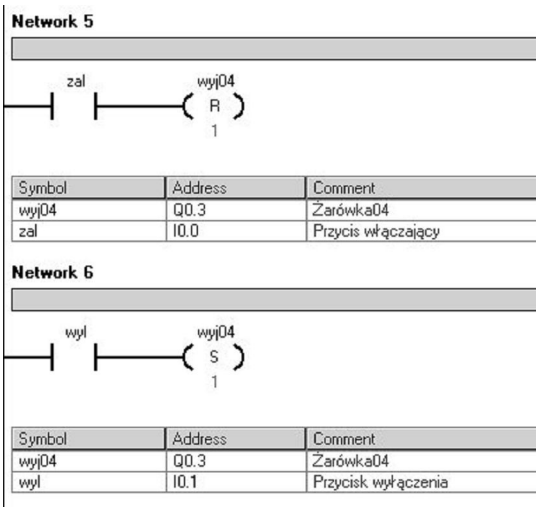
W celu zwiększenia niezawodności działania oraz ze względu na bezpieczeństwo obsługi urządzeń, przyciski wyłączające (awaryjne) w instalacjach ze sterownikami powinny być stykami rozwiernymi (normalnie zamkniętymi). Jakiekolwiek uszkodzenie tego styku, rozkręcenie, przerwanie, przecięcie przewodu doprowadzającego prąd do tego styku, powinno wyłączyć element wykonawczy.

Podstawową instrukcją niezbędną do realizacji funkcji sekwencyjnych jest układ pamięciowy z przerzutnikiem R-S. Jest to instrukcja dwuargumentowa z wejściem ustawiającym (S-set) i wejściem kasującym (R - reset). W układzie pamięci R-S (rysunek 3.2a, b) zmiana sygnału z 0 na 1 na wejściu ustawiającym zal, ustawia układ pamięci (pojawienie się stanu wysokiego (”1”) na wyjściu Q0.2, zaś zmiana sygnału z 0 na 1 na wejściu zerującym (kasującym) wyl, zeruje układ pamięci (pojawienie się stanu niskiego (”0”) na wyjściu Q0.2).

Jeżeli obydwa wejścia mają równocześnie stan ”1”, to o stanie wyjścia decyduje kolejność argumentów. Jeżeli ostatnim argumentem jest kasowanie (rysunek 3.2a), to przerzutnik jest zerowany – na wyjściu Q0.2 otrzymujemy ”0”. Jeżeli ostatnim argumentem jest ustawianie (rysunek 3.2b), to przerzutnik jest ustawiony – na wyjściu Q0.3 otrzymujemy ”1”. Rozkaz znajdujący się bliżej końca programu jest rozkazem dominującym (priorytetowym).

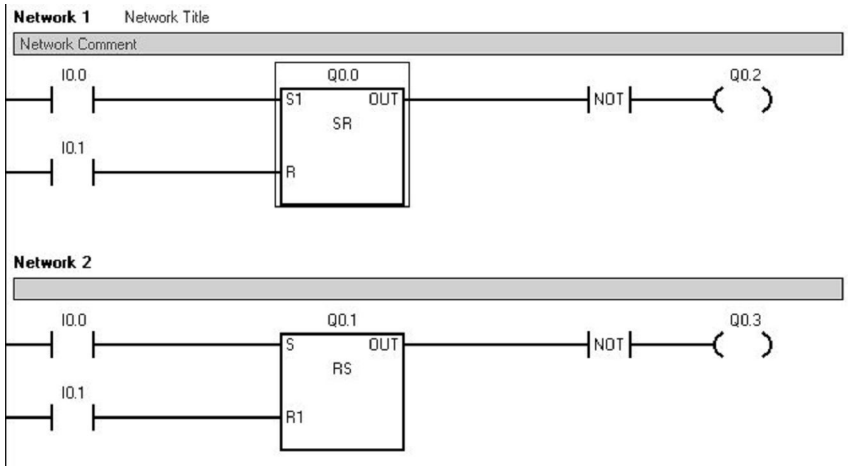


Rysunek 3.2a. Pamięć RS z wykorzystaniem elementu wyjściowego z pamięcią, priorytet kasowania



Rysunek 3.2b. Pamięć RS z wykorzystaniem elementu wyjściowego z pamięcią, priorytet zapisu

W programie Step7Micro/WIN dostępne są bloki realizujące funkcje przerzutnika SR z priorytetem wejścia zapisującego S1-R oraz z priorytetem wejścia zerującego S-R1 (kasującego).



Rysunek 3.3. Przerzutnik z priorytetem wejścia ustawiającego S1 i priorytetem wejścia zerującego R1

Działanie przerzutnika jest takie samo jak układu przedstawionego na rysunku 3.2a i 3.2b. Układy realizują funkcję przerzutnika asynchronicznego SR. Przerzutnik tego typu zakłada, że podanie na wejście S i R przerzutnika jednocześnie 1 jest zabronione. Ponieważ w sterowniku

cewka zerująca i ustawiająca może wystąpić w dowolnej kolejności, to może pojawić się problem jednoczesnego podania wartości 1 na wejście S i R. Podobnie jak w cewkach wyróżnia się przerzutniki z dominacja wejścia S lub R. Różnice w działaniu cewek przedstawione są w tabeli poniżej.

Przerzutnik z priorytetem wejścia ustawiającego S1		
S1	**R**	**Q**
0	0	Stan poprzedni
0	1	0
1	0	1
1	1	1
Przerzutnik z priorytetem wejścia zerującego R1		
S	**R1**	**Q**
0	0	Stan poprzedni
0	1	0
1	0	1
1	1	0

3.2. Zadania do przeanalizowania

Napiszemy program, który realizuje działanie układu przedstawionego w tabeli.

AB	**Q^n**		**Q^{n+1}**
	0	**1**	
00	**1**	**0**	$\overline{Q}^n$
01	**1**	**1**	**1**
11	**0**	**1**	**Q^n**
10	**0**	**0**	**0**

Powyższą tabelę zapiszemy w postaci:

Q^n	**AB**			
	00	**01**	**11**	**10**
0	**1**	**1**	**0**	**0**
1	**0**	**1**	**1**	**0**

Tabela kodowania dla przerzutnika SR jest następująca:

$Q^n \rightarrow Q^{n+1}$		S	R
0	0	0	-
0	1	1	0
1	0	0	1
1	1	-	0

Tabela dla wejścia S i R jest następująca:

Wejście S				
Q^n	AB			
	00	01	11	10
0	1	1	0	0
1	0	-	-	0

Wejście R				
Q^n	AB			
	00	01	11	10
0	0	0	-	-
1	1	0	0	1

Stąd funkcja wzbudzeń jest następująca:

$$S = \overline{QA}$$
$$R = Q\overline{B}$$

UWAGA!
Program powinien zawierać generator, który będzie synchronizował zmianę zboczem zmianę stanów sterownika.

3.3. Zadania do zrealizowania

Zadanie 1

Zrealizować sterowanie silnikiem elektrycznym bez samopodtrzymania. Silnik jest włączany przyciskiem P1. Silnik pracuje dopóty, dopóki przycisk P1 jest wciśnięty. Stan pracy silnika sygnalizowany jest optycznie poprzez lampki kontrolne: L1 – silnik włączony; L2 – silnik wyłączony. Silnik jest dodatkowo zabezpieczony przed przeciążeniem przez użycie przekaźnika termicznego T1.

Zadanie 2

Zrealizować sterowanie silnikiem elektrycznym z samopodtrzymaniem. Silnik jest włączany przyciskiem P1. Silnik pracuje bez względu na stan przycisku P1. Stan pracy silnika sygnalizowany jest optycznie poprzez lampki kontrolne: L1 – silnik włączony; L2 – silnik wyłączony. Silnik jest dodatkowo zabezpieczony przed przeciążeniem przez użycie przekaźnika termicznego T1. Wyłączenie silnika realizuje przycisk P2.

Zadanie 3

Zrealizować sterowanie silnikiem elektrycznym z samopodtrzymaniem realizowane z dwóch pulpitów sterujących. Silnik jest włączany przyciskiem P1 na pulpicie I lub P2 na pulpicie II. Silnik pracuje bez względu na stan przycisku P1 i P2. Stan pracy silnika sygnalizowany jest optycznie poprzez lampki kontrolne: L1 (L2 - pulpit II) – silnik włączony; L3 (L4 - pulpit II) – silnik wyłączony. Silnik jest dodatkowo zabezpieczony przed przeciążeniem przez użycie przekaźnika termicznego T1. Wyłączenie silnika realizuje przycisk P3.

Zadanie 4

Proszę napisać program realizujący sumowanie szeregowe dwu liczb dwójkowych. W sumatorze szeregowym wprowadzane są bit po bicie dwa bajty. Na wyjściu sumatora wyprowadzana jest suma powyższych bajtów, również bit po bicie. Bajty wprowadzane są w kolejności od najmniej do najbardziej znaczącego.

Zadanie 5

Proszę napisać program realizujący komparator szeregowy dwu liczb binarnych. W komparatorze liczby mogą być porównywane bit po bicie, poczynając od najmniej znaczącego lub porównywanie może odbywać się od najbardziej znaczącego bitu. Liczby mogą być: A=B, A>B, A<B, co powinno być sygnalizowane na odpowiednich wyjściach.

Zadanie 6

Proszę napisać program realizujący funkcję układu asynchronicznego zgodnie z tabelą:

AB	Q^n		Q^{n+1}
	0	1	
00	**0**	**0**	**0**
01	**1**	**1**	**1**
11	**0**	**0**	**0**
10	**0**	**0**	**0**

Zadanie 7

Proszę napisać program realizujący funkcję układu synchronicznego zgodnie z tabelą:

AB	Q^n		Q^{n+1}
	0	1	
00	**1**	**0**	$\overline{Q}^n$
01	**1**	**0**	$\overline{Q}^n$
11	**1**	**1**	**1**
10	**0**	**0**	**0**

AB	Q^n		Q^{n+1}
	0	1	
00	0	1	Q^n
01	0	0	0
11	1	0	$\overline{Q}^n$
10	1	1	1

AB	Q^n		Q^{n+1}
	0	1	
00	1	1	1
01	0	0	0
11	1	0	$\overline{Q}^n$
10	1	0	$\overline{Q}^n$

AB	Q^n		Q^{n+1}
	0	1	
00	1	1	1
01	0	1	$\overline{Q}^n$
11	0	1	$\overline{Q}^n$
10	0	1	Q^n

AB	Q^n		Q^{n+1}
	0	1	
00	1	1	1
01	1	1	1
11	0	0	0
10	1	0	$\overline{Q}^n$

AB	Q^n		Q^{n+1}
	0	1	
00	1	1	1
01	0	1	Q^n
11	0	0	0
10	1	1	1

4 Układy z zależnościami czasowym

W tym rozdziale:

- Programowanie timerów.
- Układy sterowania z zależnościami czasowymi.

4.1. Wprowadzenie

Układy czasowe pozwalają na wprowadzenie zależności czasowych dla zdarzeń. Zależności czasowe mogą mieć charakter:

1. opóźnienia czasowego,
2. generacji zadanego przebiegu,
3. odmierzania czasu.

Do realizacji układów czasowych w sterowniku S7-200 wykorzystuje się następujące przekaźniki czasowe:

TONR	*przekaźnik czasowy z pamięcią*	*Zlicza czas, przez który doprowadzany jest sygnał. Po doprowadzeniu sygnału do wejścia Reset wartość przekaźnika jest zerowana.*
TOF	*przekaźnik czasowy z zanegowanym wejściem, bez pamięci*	*Zlicza czas, przez który nie jest doprowadzany sygnał. Po upływie zadanej wartości czasu następuje wyzerowanie wartości przerzutnika. Po podaniu wartości wysokiego poziomu przerzutnik nie pamięta ustalonej wartości.*
TON	*przekaźnik czasowy bez pamięci*	*Zlicza czas, przez który doprowadzany jest sygnał. Po zaniku sygnału na wejściu zliczona wartość jest utracona.*

W sterownikach S7-200 wyróżnia się trzy rozdzielczości przekaźników czasowych.

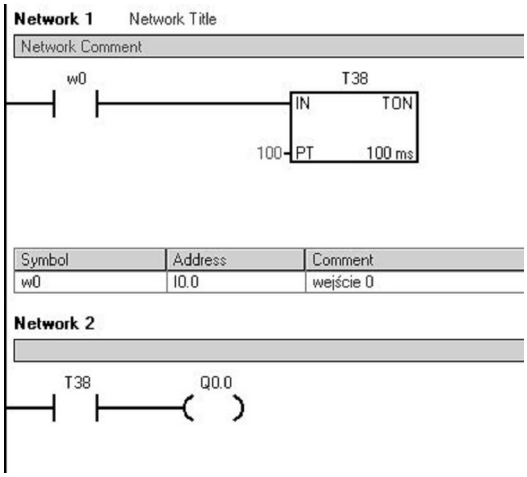
Timer	Rozdzielczość	Adres Timera
TONR	1 ms	T0, T64
	10 ms	T1÷T4, T65÷T68
	100 ms	T5÷T31, T69÷T95
TON TOF	1 ms	T32 ,T96
	10 ms	T33÷T36, T97÷T100
	100 ms	T37÷T63, T101÷T255

Przekaźnik jest sterowany przez dwa wejścia:

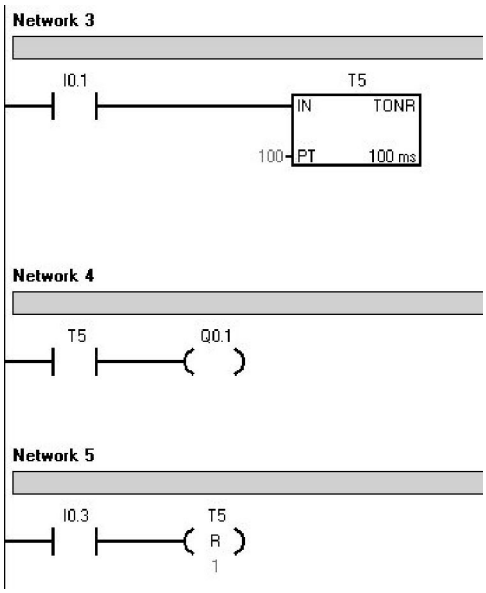
IN – wejście zezwalające na pracę,

PT – wartość określająca przedział czasowy (przedział jest określony jako iloczyn wartości PT i rozdzielczości).

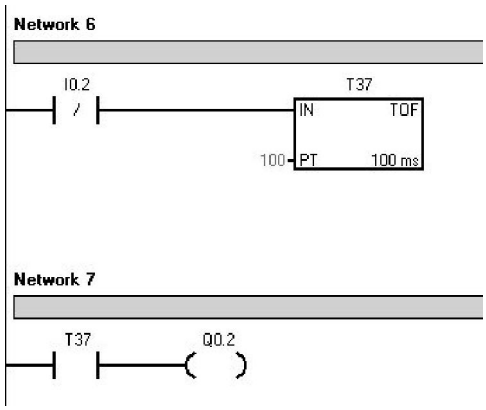
Oprócz zdefiniowania powyższych wartości należy podać adres Timera. Program dla podanego adresu automatyczne przypisuje rozdzielczość.



Rysunek 4.1. Przerzutnik czasowy TON



Rysunek 4.2. Przerzutnik czasowy TONR

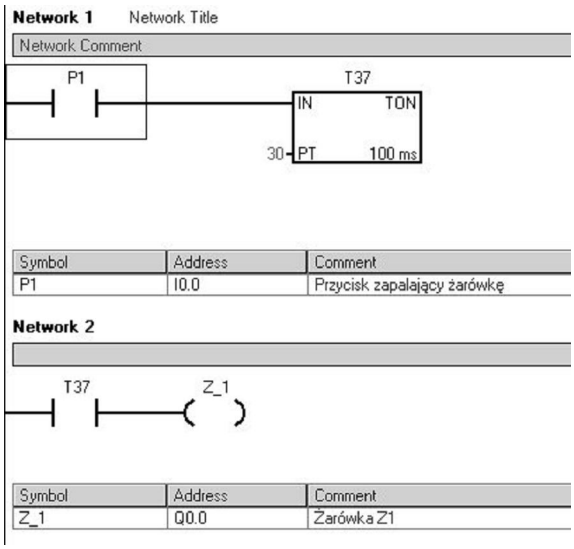


Rysunek 4.3. Przerzutnik czasowy TOF

4.2. Zadania do przeanalizowania

Zadanie 1

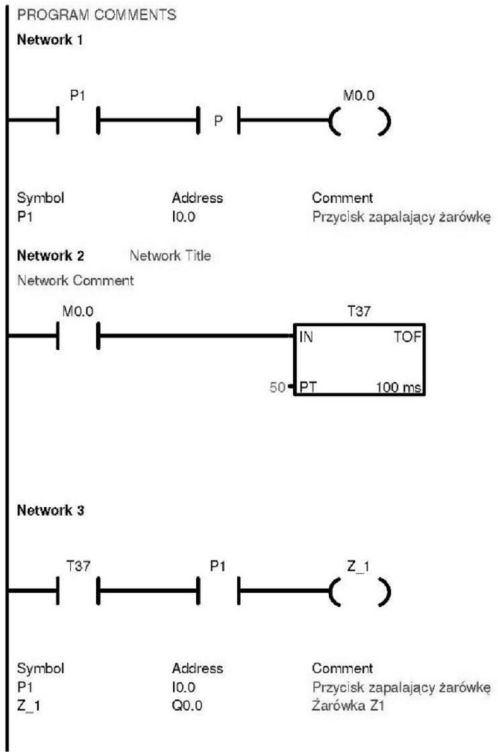
Żarówka Z1 zapali się po 3 sekundach od momentu włączenia przycisku P1. Po wyłączeniu przycisku P1 żarówka Z1 zgaśnie.



Rysunek 4.4. Realizacja zadania 1

Zadanie 2

Wciśnięcie przycisku P1 zapali żarówkę Z1 na 5 sekund. Żarówkę Z1 można wyłączyć w dowolnej chwili wyłączając przycisk P1.

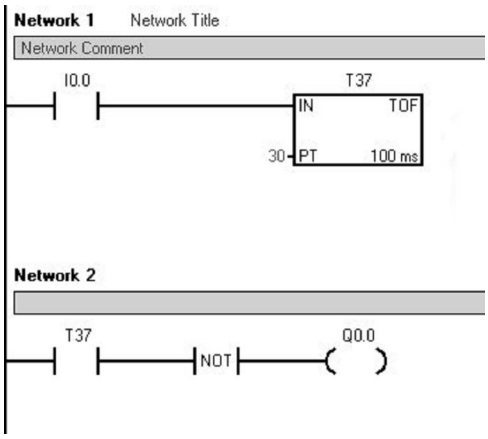


Rysunek 4.5. Realizacja zadania 2

M0.0 – zmienna wewnętrzna (marker, znacznik, flaga) służy do zapamiętania stanów pośrednich i tymczasowych wyników obliczeń nie oddziałujących bezpośrednio na urządzenia wyjściowe oraz sygnalizowania pewnych kroków w sekwencji sterującej.

Zadanie 3

Przekaźnik czasowy timer ustawia na wyjściu Q0.0 wartość 0 zawsze, ilekroć wartość zmiennej związanej z wejściem I0.0 zmienia wartość na 1. Sygnał wyjściowy jest ponownie ustawiany na 1 po upływie 3 sekund od momentu ustawienia wejścia na 0.

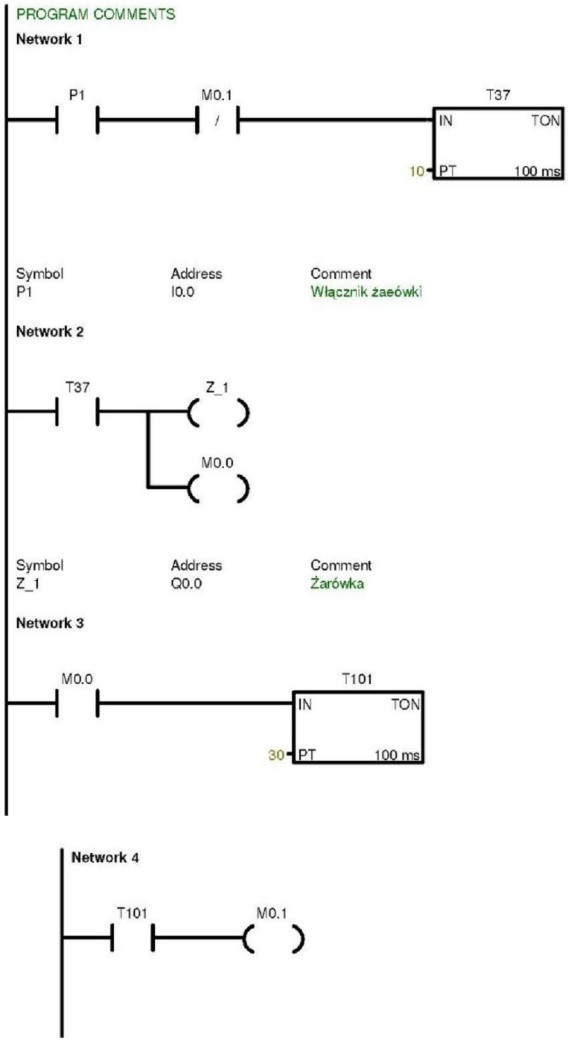


Rysunek 4.6. Realizacja zadania 3

Zadanie 4

Po wciśnięciu przycisku P1 program realizuje świecenie żarówki Z1 w następującym cyklu:

świecenie – 3 sekundy; przerwa - 1 sekunda.



Rysunek 4.7. Realizacja zadnia 4

4.3. Zadania do zrealizowania

Zadanie 1

Żarówka Z1 jest zapalana po wciśnięciu przycisku P1. Żarówka pali się przez 5 sekund bez względu na stan przycisku P1.

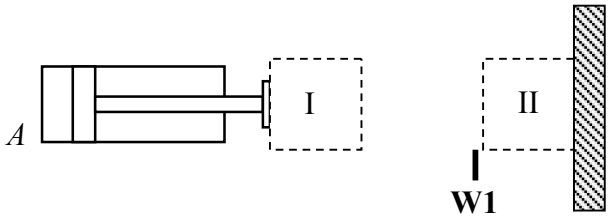
Zadanie 2

Żarówka Z1 zapali się po 7 sekundach od momentu włączenia przycisku P1 na 5 sekund bez względu na stan przycisku P1. Układ zrealizuj w oparciu o dwa przekaźniki czasowe.

Zadanie 3

Tłoczysko siłownika pneumatycznego dwustronnego działania A wysuwa się po wciśnięciu przycisku **W0**. Po osiągnięciu przez tłoczysko siłownika położenia krańcowego (sygnał optyczny L1) następuje docisk detalu przez 6 sekund, po czym tłoczysko siłownika wycofuje się w tylne skrajne położenie (rysunek 4.8).

Krańcowe położenie tłoczyska siłownika określa wyłącznik krańcowy **W1**.



Rysunek 4.8. Rysunek do zadania 3

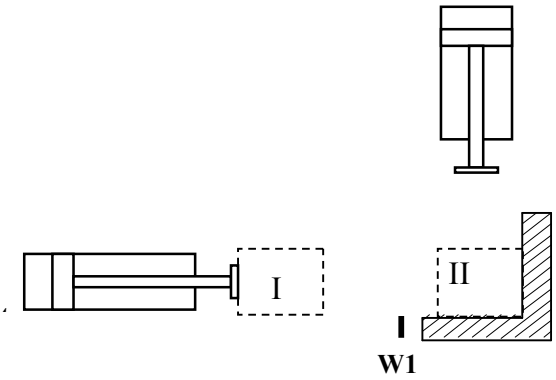
Zadanie 4

Zrealizować sterowanie chłodzeniem prądnicy z chwilą jej uruchomienia i 5 sekund po zatrzymaniu prądnicy. Uruchomienie prądnicy sygnalizowane jest optycznie – lampka L1, włączenie chłodnicy sygnalizowane jest optycznie – lampka L2.

Zadanie 5

Tłoczysko siłownika pneumatycznego dwustronnego działania A wysuwa się po wciśnięciu przycisku **W0**. Po osiągnięciu przez tłoczysko siłownika A położenia krańcowego (położenie II detalu sygnalizowane optycznie – lampka L1) następuje wysuw tłoczyska B (lampka L2). Po upływie 4 sekund następuje powrót tłoczyska siłownika B w tylne położenie. Następnie po upływie 3 sekund następuje powrót tłoczyska siłownika A (rysunek 4.9).

Krańcowe położenie tłoczyska siłownika A określa wyłącznik krańcowy **W1**.



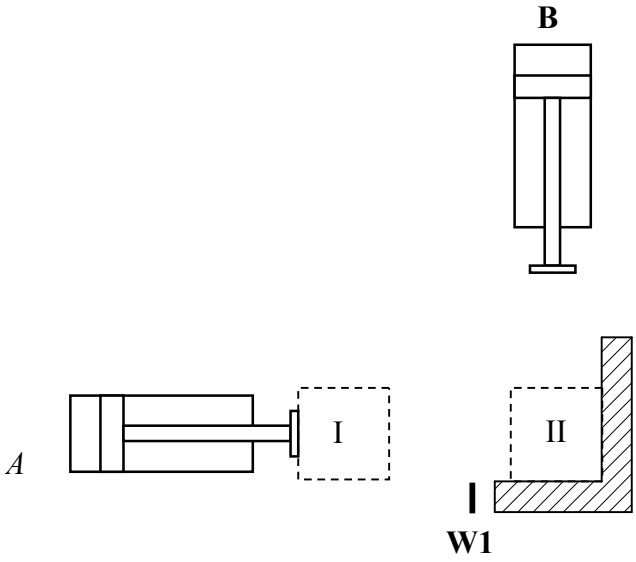
Rysunek 4.9. Rysunek do zadania 5

Zadanie 6

Tłoczysko siłownika pneumatycznego dwustronnego działania A wysuwa się po wciśnięciu przycisku **W0**. Po osiągnięciu przez tłoczysko siłownika A położenia krańcowego (położenie II detalu sygnalizowane optycznie – lampka L1) i upłynięciu 2 sekund następuje wysuw tłoczyska B (lampka L2). Następnie po upływie 4 sekund następuje

powrót tłoczyska siłownika B w tylne położenie, po czym po upływie 2 sekund następuje powrót tłoczyska siłownika A (rysunek 4.10).

Krańcowe położenie tłoczyska siłownika A określa wyłącznik krańcowy **W1**.



Rysunek 4.610. Rysunek do zadania 6

5 Układy sterowania z licznikami zdarzeń

W tym rozdziale:

- Programowanie liczników.
- Układy sterowania z licznikami zdarzeń.

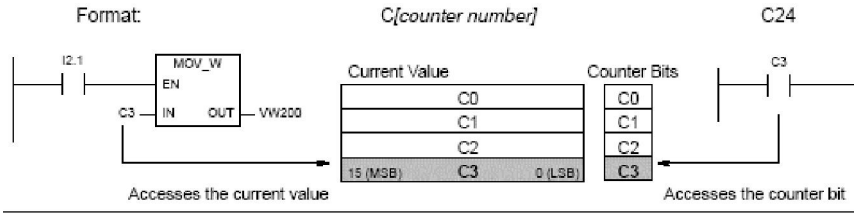
5.1. Wprowadzenie

Liczniki wykorzystuje się do określania liczby zaistniałych zdarzeń. Zawartość timera zmieniała się w funkcji czasu z dokładnie określonym inkrementem czasu. Wartość inkrementu zależy od zastosowanego timera. W liczniku zmiana zawartości licznika zależy od wystąpienia zdarzenia. Przykładowo jeżeli na wejściu sterownika przychodzi impuls to następuje zmian zawartości licznika.

Liczniki wykorzystuje się do:

1. zliczania zdarzeń,
2. sterowania zdarzeniami.

Każdy z licznik ma przypisany w przestrzeni adresowej 16-to bitowy rejestr. Liczniki są adresowane zgodnie ze składnią C[nr licznika].



Rysunek 5.1. Dostęp do adresu licznika

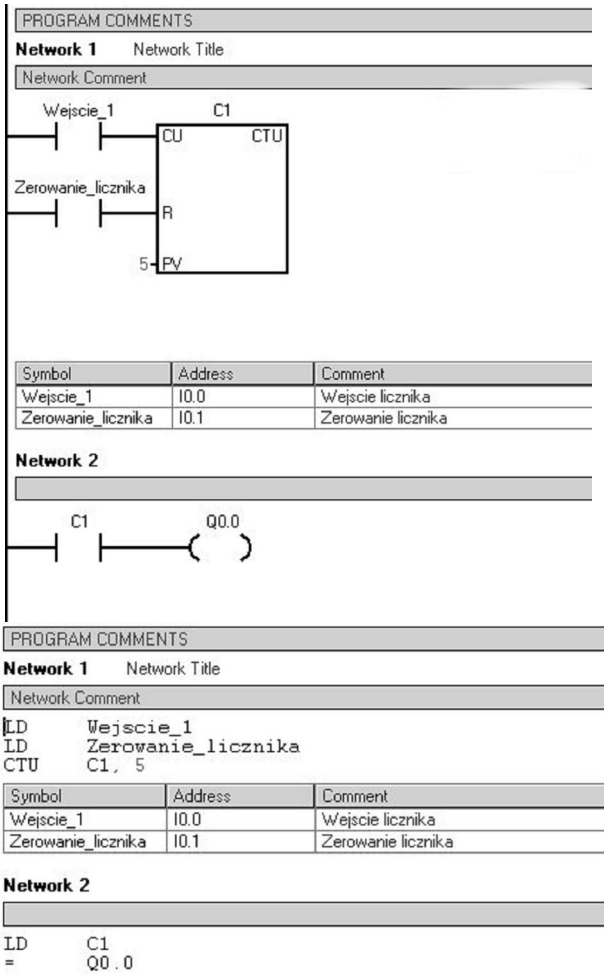
Do realizacji funkcji zliczania w sterowniku SIMANTIC S7-200 wykorzystuje się następujące liczniki:

CTU	*Licznik zliczający w górę*	*Każde doprowadzenie sygnału do tego licznika powoduje zwiększenie jego wartości o 1 - inkrementacja*
CTD	*Licznik zliczający w dół*	*Każde doprowadzenie sygnału do tego licznika powoduje zmniejszenie jego wartości o 1 - dekrementacja*
CTUD	*Licznik zliczający w górę i w dół (rewersyjny)*	*Licznik posiada dwa wejścia. Przyjście zdarzenia na wejście pierwsze CU powoduje zwiększenie zawartości licznika. Natomiast przyjście zdarzenia na wejście CD następuje zmniejszenie zawartości licznika.*

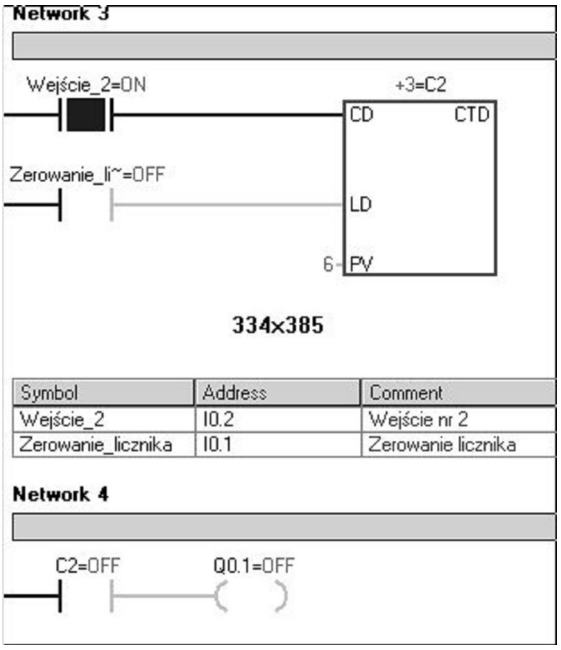
Licznik zliczający w górę CTU posiada trzy wejścia:

1. wejście na które przychodzi zdarzenie zliczające - CU,
2. wejście zerujące zawartość licznika - R,
3. wejście określające liczbę dla której licznik generuje licznik przechodzi w stan włączenia on – PV (wartość zmienia się od 0 do 32767).

Przyjście kolejnych zdarzeń na wejście CU powoduje zwiększenie zawartości licznika (rysunek 5.2). Po osiągnięciu wartości równej PV następuje ustawienie bitu licznika na wartość 1. Stan licznika jest wyświetlany i dostępny jest pod słowem umieszczonym w pamięci o adresie C1 (rysunek 5.1). Jeżeli w trakcie pracy wskazane jest wyzerowanie licznika to należy podać wartość na wejście zerujące R.



Rysunek 5.2. Licznik zliczający w górę



Symbol	Address	Comment
Wejście_2	I0.2	Wejście nr 2
Zerowanie_licznika	I0.1	Zerowanie licznika

Network 3

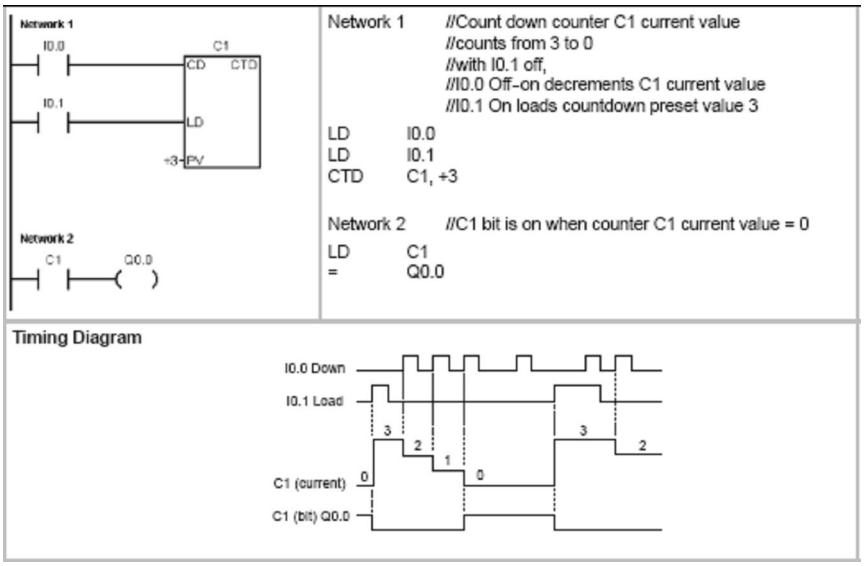
```
LD      Wejście_2
LD      Zerowanie_licznika
CTD     C2, 6
```

Symbol	Address	Comment
Wejście_2	I0.2	Wejście nr 2
Zerowanie_licznika	I0.1	Zerowanie licznika

Network 4

```
LD      C2
=       Q0.1
```

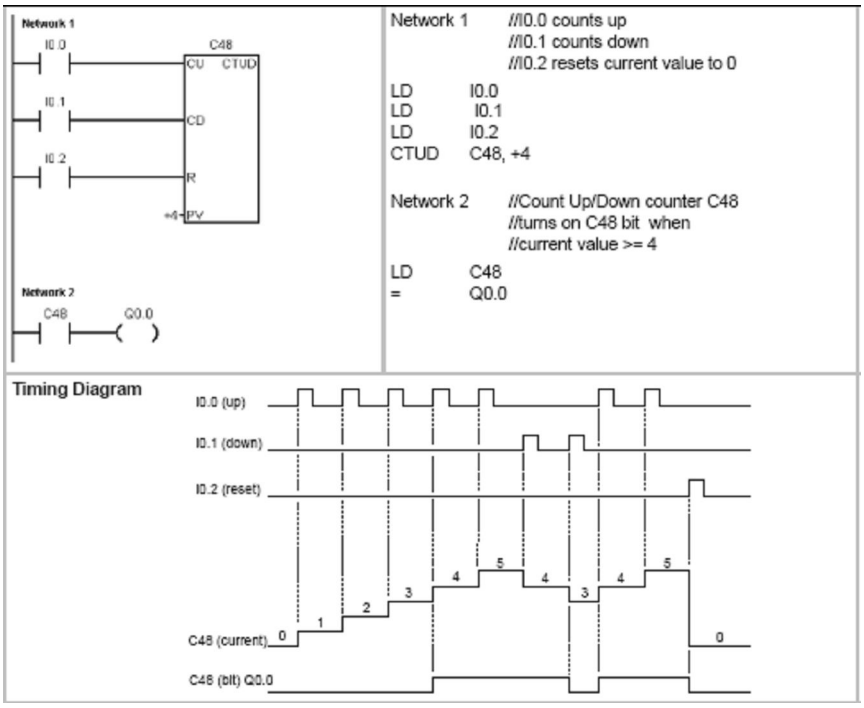
Rysunek 5.3. Realizacja licznika CTD



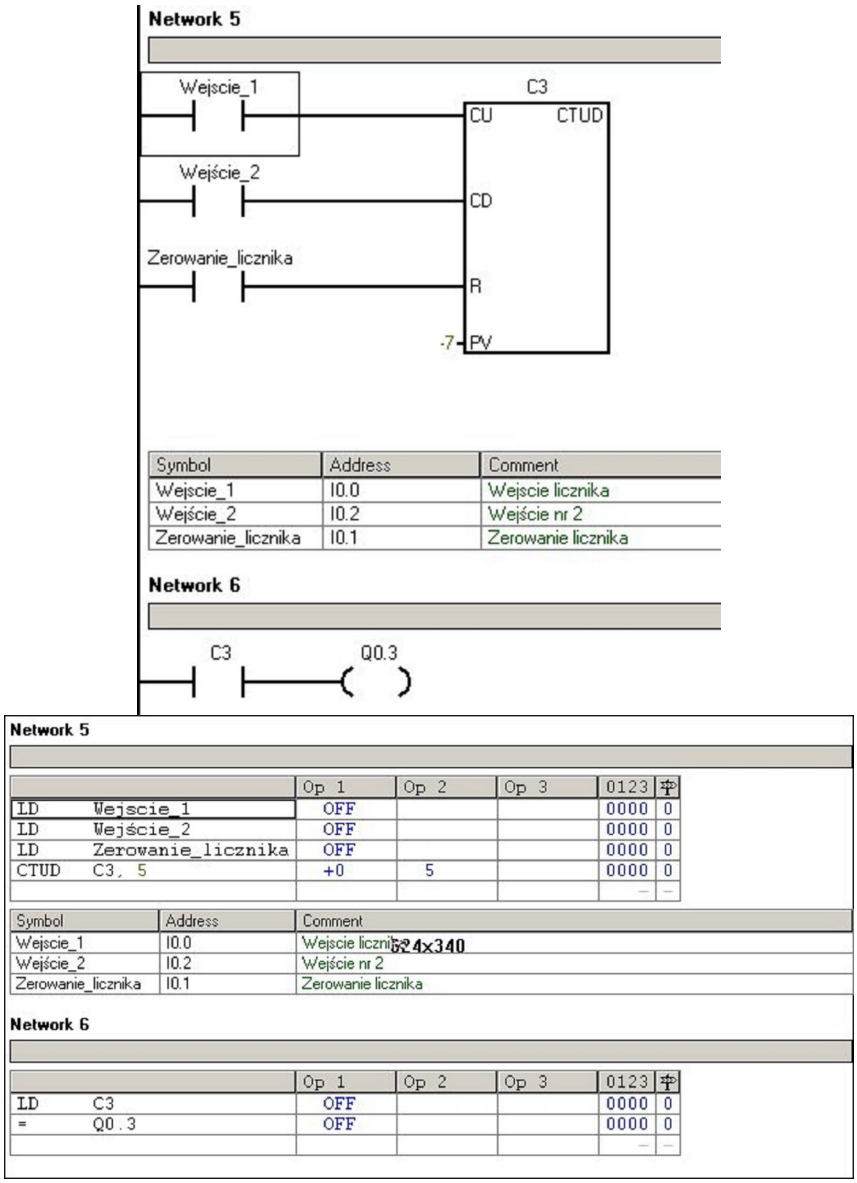
Rysunek 5.4. Licznik zliczający w dół

Licznik zliczający w dół posiada trzy wejścia (rysunek 5.3). Pierwsze wejście (CD) rejestruje zdarzenie. Wejście PV określa maksymalną zawartość licznika, od której zmniejsza się zawartość licznika. Przyjście zdarzenia powoduje zmniejszenie zawartości licznika. Po osiągnięciu wartości 0 licznik ustawia bit licznika na wartość 1 (rysunek 5.4). W liczniku CTD nie występuje wejście zerujące, tylko wejście ustawiające wartość początkową LD, która jest równa PV. Przyjście zdarzenia na wejście LD powoduje zapisanie do rejestru licznika wartości PV.

Licznik zliczający w górę i dół CUD działa podobnie jak licznik zliczający w górę. W liczniku występuje dodatkowo czwarte wejście CD. Przyjście zdarzenia na wejście CU zwiększa zawartość licznika, natomiast przyjście zdarzenia na wejście CU powoduje jego obniżenie. Po osiągnięciu lub przekroczeniu wartości podanej na wejście PV bit licznika ustawiany jest na 1. Jeżeli na wejście CD podano zdarzenia, które obniżą zawartość licznika poniżej wartości PV, to bit licznika ustawi się na wartość 0. Zawartość rejestru licznika można wyzerować, przez podanie zdarzenia na wejście zerujące R (rysunek 5.5). Zawartość licznika może zmieniać się od -32767 do 32767. Na rysunku 6 przedstawiona jest realizacja licznika CTUD.



Rysunek 5.5. Licznik zliczający w górę i dół CTUD

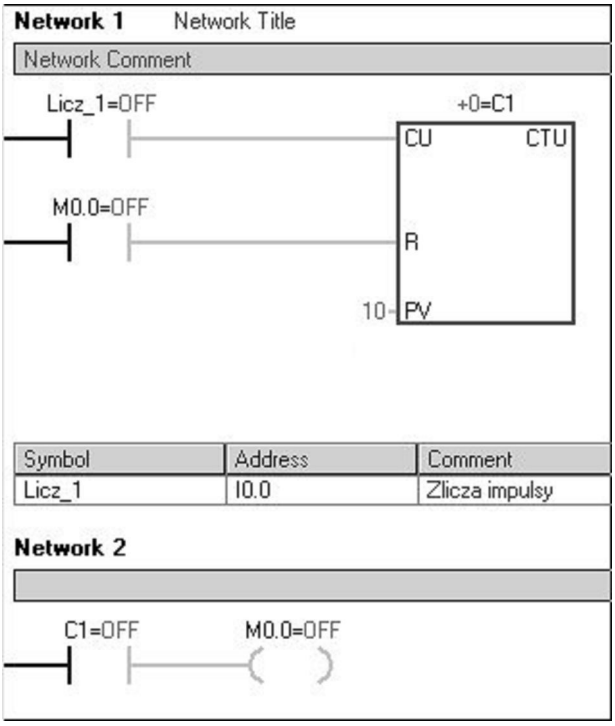


Rysunek 5.6. Realizacja licznika CTUD

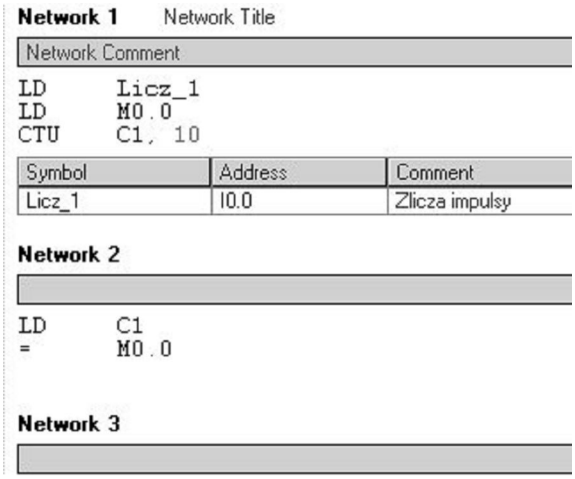
5.2. Zadania do przeanalizowania

Zadanie 1

Każde wciśnięcie przycisku P1 powoduje zwiększenie wartości licznika LICZ_1 o 1. Po odliczeniu 10 impulsów sygnał wyjściowy licznika jest doprowadzany do wewnętrznego przekaźnika M0.0. Ustawienie wartości zmiennej M0.0 na 1 powoduje wyzerowanie wartości zliczonej przez licznik.



Rysunek 5.7. Rozwiązanie zadania nr 1 w języku LD



Rysunek 5.7. Rozwiązanie zadania nr 1 w języku tekstowym

5.3. Zadania do zrealizowania

Zadanie 1

Napisać program na licznik impulsów przychodzących do wejścia W1. Licznik ma liczyć do 3 (podanie czterech impulsów powoduje powrót do stanu wyjściowego). Przycisk RESET służy do zerowania licznika.

Zadanie 2

Po wciśnięciu przycisku START, ze pomocą przenośnika taśmowego napędzanego silnikiem, wazony są transportowane do pudełek. W pudełku mieści się 6 wazonów (wazony wykrywane są przez optyczny czujnik cz_wa. Wypełnione pudełka (sygnał świetlny L1) za pomocą zwrotnicy są skierowane do samochodu (sterownik wysyła sygnał sterujący zwrotnicą). Po minięciu przez paczkę zwrotnicy i ustawieniu w pozycji wyjściowej zwrotnicy lampa L1 gaśnie. Mniecie zwrotnicy przez paczkę wykrywa czujnik cz1 Po załadowaniu 5 pudełek (sygnał świetlny L2) samochód odjeżdża. Lampa L2 gaśnie po minięciu samochodu przez czujniki w odpowiedniej kolejności cz2 i cz3. Kolejność zadziałania czujników jest istotna ze względu na rozróżnienie czy samochód odjeżdża czy dojeżdża do rampy.

Zadanie 3

Po 1 sekundzie od wciśnięcia przycisku W0 następuje 4 – krotne zapalenie się lampki.

(świecenie – 1 sekunda; przerwa - 2 sekundy).

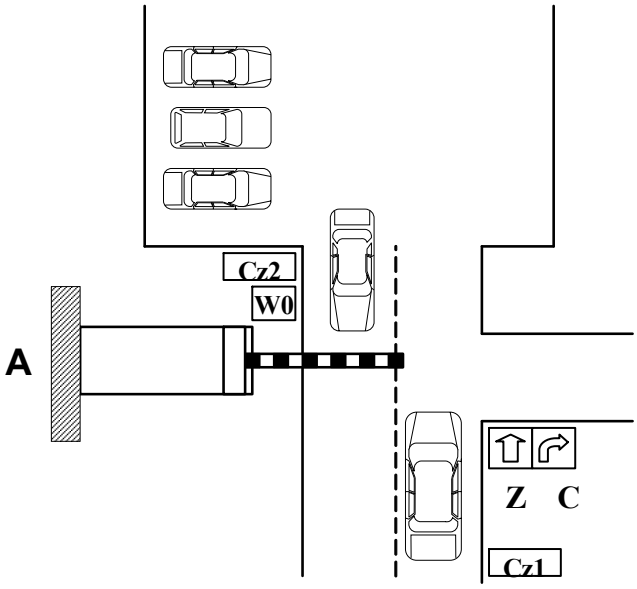
Zadanie 4

Parking mieści maksymalnie 10 samochodów. Czujnik Cz1 rejestruje liczbę wjeżdżających samochodów. Czujnik Cz2 rejestruje liczbę samochodów wyjeżdżających. Szlaban A jest zamknięty.

Każdy wyjeżdżający kierowca zatrzymuje samochód w celu opłacenia postoju (wciśnięcie przycisku W0). Po uiszczeniu opłaty szlaban A otwiera się na 4 sekundy.

W przypadku, gdy w garażu znajduje się 10 samochodów zostaje włączone czerwone światło dla wjeżdżających samochodów.

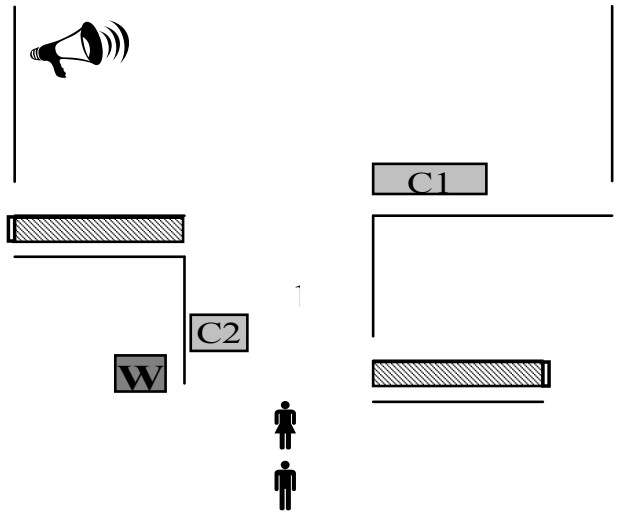
Gdy liczba zaparkowanych samochodów nie przekracza 10 pali się światło zielone.



Rysunek 5.8. Schemat do zadania 4

Zadanie 5

Napisać program realizujący zliczanie liczby osób wchodzących do sądu – czujnik C1. Jeżeli w trakcie przejścia, czujnik C2 wykryje metal, włączony zostaje sygnał alarmowy (5 x po 1 sekundzie) i zamknięte pomieszczenie 1. Przycisk W otwiera drzwi.

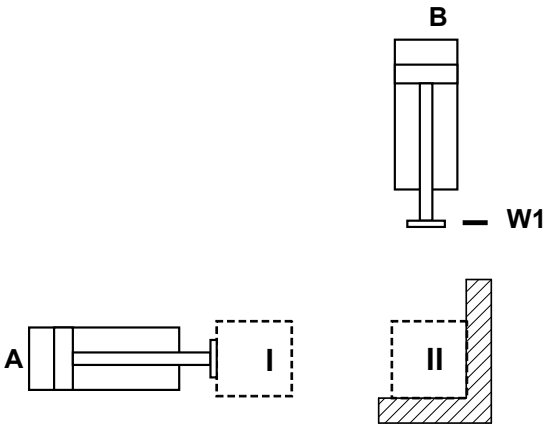


Rysunek 5.9. Schemat do zadania 5

Zadanie 6

Po wciśnięciu przycisku W0 tłoczysko siłownika pneumatycznego dwustronnego działania A wysuwa się i detal zostaje przesunięty z pozycji I na pozycję II. Po upływie 1 sekundy następuje 3 – krotny docisk detalu w pozycji II przy pomocy tłoczyska siłownika B. W położeniu wysuniętym tłoczysko siłownika B pozostaje każdorazowo 1 sekundę. Następnie następuje powrót tłoczyska siłownika A.

Skrajne tylne położenie tłoczyska siłownika B sygnalizuje włącznik krańcowy W1.



Rysunek 5.10. Rysunek układu do zadania 6

Literatura do części II

[1] Podręcznik SIMATIC S7-200, SIEMENS, Warszawa 2006.

[2] K. Kamiński: Programowanie w Step7 MicroWIN, 2006.