AUTOMATYKA PRZEMYSŁOWA

Literatura:

1. Aleksander M., Borys W.: Elementy techniki cyfrowej. PWSZ, Nowy Sącz 2002.

2. Barczyk J.: Automatyzacja procesów dyskretnych. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 2003.

3. Brzózka J.: Regulatory cyfrowe w automatyce. MIKOM, Warszawa 2002.

4. Materiały szkoleniowe opracowane przez: Grzegorza Faracika, Grzegorza Dubiela, Sławomira Dzierżka i Tomasza Michałka. Zbiór zadań dla sterowników GE-Fanuc serii 90-30/VersaMax/Micro wraz z przykładami rozwiązań, ASTOR sp. z o.o. 31-112 Kraków.

5. Kamiński K.: Programowanie sterownika S7 – NORCOM, Gdańsk 2000.

6. Legierski T.: Programowanie sterowników PLC. Wydawnictwo pracowni komputerowej J. Skalmierskiego, Gliwice 1998.

7. Kasprzyk J.: Programowanie sterowników przemysłowych. Wydawnictwo Naukowo – Techniczne, Warszawa 2006.

Kryteria zaliczenia przedmiotu

- Wykład: Uczęszczanie na wykłady. Zaliczenie końcowe z wykładu.
- Laboratorium: student wykonał i zaliczył wszystkie zajęcia laboratoryjne, zgodnie z planem studiów.
- Ocena końcowa średnia z ocen za wiadomości teoretyczne, z pracy w laboratorium i ze sprawozdań. Ocena do indeksu po pozytywnym zaliczeniu 2 form zajęć z oceną średnią z otrzymanych ocen z wykładu i laboratorium.

TEMATY WYKŁADÓW Z AUTOMATYKI PRZEMYSŁOWEJ

L.p.	Temat zajęć	L. godz.	Data	Podpis
1.	Algebra Boole'a i podstawowe funkcje logiczne.	1	05.10	
2.	Rozkład logiczny zera i jedynki, zapis funkcji logicznej i jej minimalizacja	1	12.10	
3.	Bramki logiczne i układy scalone.	1	19.10	
4.	Układy logiczne kombinacyjne.	1	26.10	
5.	Układy logiczne sekwencyjne, synchroniczne i asynchroniczne.	1	09.11	
6.	Układy logiczne i systemy dwójkowe.	1	16.11	
7.	Budowa pamięci, liczników impulsów i procesora	1	23.11	
8.	Przemysłowe układy sterowania binarnego.	1	30.11	
9.	Budowa sterownika PLC i jego moduły.	1	07.12	
10.	Programowanie sterownika w języku drabinkowym	1	14.12	
11.	Sygnały i zmienne binarne	1	21.12	
12.	Realizacja układów logicznych w oparciu o sterowniki PLC.	1	04.01	
13.	Sterowanie cykliczne i regulacja dwupołożeniowa	1	11.01	
14.	Przemysłowe systemy alarmowe i sterowania.	1	18.01	
15.	Zaliczenie.	1	25.01	

TEMATY LABORATORIUM Z AUTOMATYKI PRZEMYSŁOWEJ
Studiów stacjonarnych 23/24

Temat zajęć	L. godz.	Data
1. Synteza układów logicznych kombinacyjnych	2	13.10.23
2. Synteza układów logicznych sekwencyjnych	2	27.10.23
3. Projektowanie układów logicznych	2	17.11.23
4. Programowalne sterowniki PLC. Budowa i konfigurowanie	2	01.12.23
5. Programowanie sterowników PLC- przykłady	2	15.12.23
6. Koncepcje projektowe układów sterujących- przykłady	2	12.01.24
7. Budowa i uruchomienie projektu na PLC	2	26.01.24
Zaliczenie	1	29.01.24

Algebra Boole’a

Przedmiotem algebry Boole’a są działania logiczne: negacja, alternatywa i koniunkcja wykonane na zmiennych i liczbach binarnych (0,1).

Algebra Boole'a

- Działania na zmiennych (obiektach), które przyjmują jedynie dwie wartości
  - w logice formalnej „prawda" lub „fałsz" (inaczej „T" lub „F")
  - w systemach cyfrowych „włączony" lub „wyłączony" inaczej „1" lub „0"
- Wyrażenia boolowskie powstają w wyniku połączenia zmiennych i działań (operacji) na nich
  - podstawowe operatory: AND, OR, NOT

Wyrażenia boolowskie

- Operator Boole'a można zdefiniować przy pomocy tabeli lub matrycy prawdy (ang. truth table) (inaczej tabelki prawdziwościowej)
  - kolumny danych wejściowych
  - kolumna danych wyjściowych
- Operator AND nazywamy koniunkcją lub iloczynem logicznym
- Operator OR nazywamy alternatywą lub sumą logiczną

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Wyrażenia boolowskie (c.d.)

- Operator NOT nazywamy negacją lub zaprzeczeniem
- Zapisujemy przy pomocy kreski nad argumentem (ē), poprzedzamy znakiem ~ (~e) lub ¬ (¬e)
- Czytamy jako „nie e”

NOT x

x	$\overline{x}$
0	1
1	0

Przykład

- Tablica prawdy dla funkcji boolowskiej

$$F(x,y,z) = x\bar{z}+y$$

- Dla obliczenia skomplikowanych funkcji boolowskich wygodnie jest rozbić ją kilka kolumn reprezentujących poszczególne kroki wyliczeń

$F(x,y,z) = x\bar{z}+y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

Tożsamości boolowskie

- Komputery składają się z obwodów implementujących funkcje boolowskie
- Im prostsza funkcja boolowska tym mniejszy obwód
  - łatwiejsze w budowie, mniejsze zużycie energii i szybsze niż obwody implemetujące złożone funkcje
- Aby zredukować funkcje boolowską można posłużyć się tożsamościami boolowskimi

Tożsamości boolowskie (c.d.)

- przemienność, łączność, rozdzielność (drugie i pierwsze prawo rozdzielności)

Identity Name	AND Form	OR Form
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x+(y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$

Tożsamości boolowskie (c.d.)

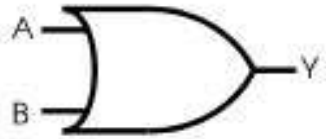
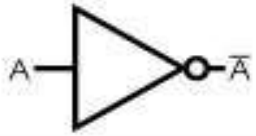
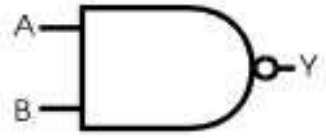
- absorpcja, prawo De Morgana, podwójne zaprzeczenie

Identity Name	AND Form	OR Form
Absorption Law	$x(x+y) = x$	$x + xy = x$
DeMorgan's Law	$\overline{(xy)} = \bar{x} + \bar{y}$	$\overline{(x+y)} = \bar{x}\bar{y}$
Double Complement Law	$\overline{(\bar{x})} = x$	

Tożsamości boolowskie (c.d.)

- element neutralny, własności „zera” i „jedynki” (prawa pochłaniania), idempotentność (prawa tautologii), uzupełnienie (prawo sprzeczności i wyłączonego środka)

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0 + x = x$
Null Law	$0x = 0$	$1 + x = 1$
Idempotent Law	$xx = x$	$x + x = x$
Inverse Law	$x\bar{x} = 0$	$x + \bar{x} = 1$

FUNKCJA LOGICZNA	SYMBOL LOGICZNY	WYRAŻENIE ALGEBRAICZNE	TRUTH TABLE		
			Inputs		Output
			A	B	Y
AND		$A \cdot B = Y$	0 0 1 1	0 1 0 1	0 0 0 1
OR		$A + B = Y$	0 0 1 1	0 1 0 1	0 1 1 1
NOT *(Inverter)*		$A = \overline{A}$	0 1		1 0
NAND		$\overline{A \cdot B} = Y$	0 0 1 1	0 1 0 1	1 1 1 0
NOR		$\overline{A + B} = Y$	0 0 1 1	0 1 0 1	1 0 0 0
XOR		$A \oplus B = Y$	0 0 1 1	0 1 0 1	0 1 1 0
XNOR		$\overline{A \oplus B} = Y$	0 0 1 1	0 1 0 1	1 0 0 1

Program symulacyjny:

https://sourceforge.net/projects/logisim/

http://www.softronix.com/download/mmlogic14.exe

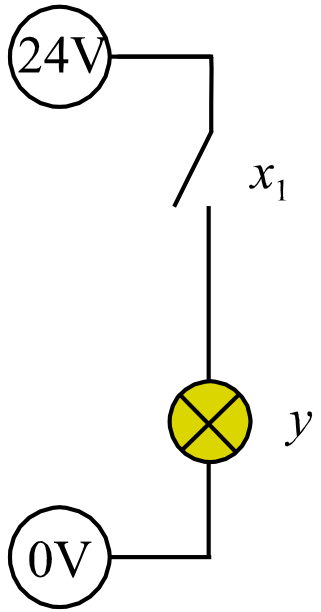
Realizacja funkcji przełączających

Elementy przełączające, z których zbudowany jest układ przełączający, łączą lub przerywają przepływ energii w obwodzie, np.:

- przekaźniki i przełączniki elektryczne załączają przepływ energii elektrycznej,
- rozdzielacze pneumatyczne zmieniają kierunek przepływu sprężonego powietrza,
- rozdzielacze hydrauliczne sterują kierunkiem przepływu cieczy hydraulicznej.

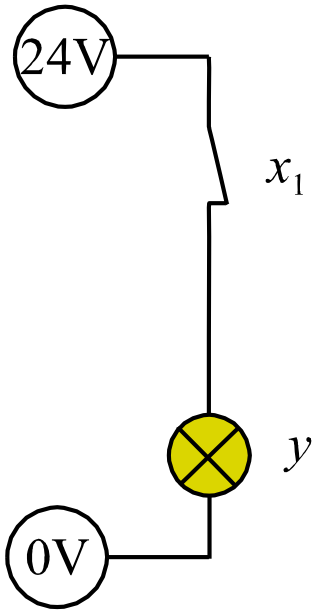
Realizacja funkcji przełączających

Elektryczne elementy stykowe



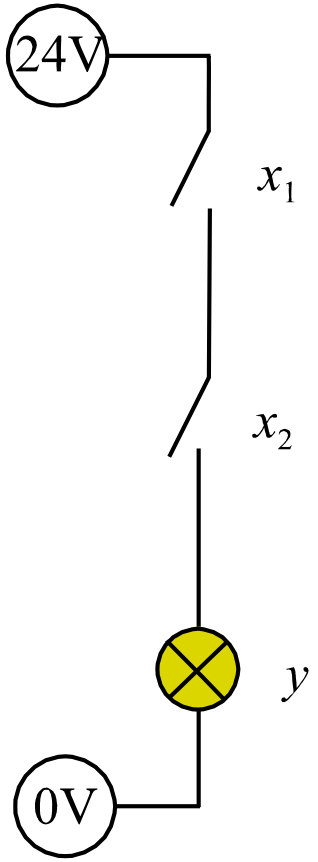
$y = x_1$

Funkcja powtórzeń



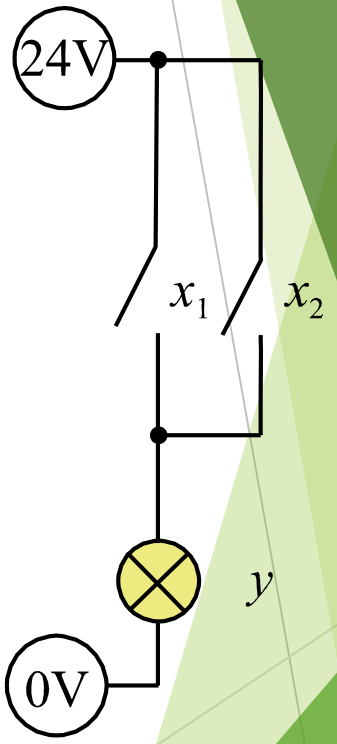
$y = \bar{x}_1$

Funkcja negacji



$y = x_1 \cdot x_2$

Funkcja iloczynu

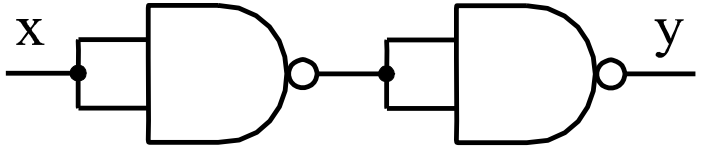


$y = x_1 + x_2$

Funkcja sumy

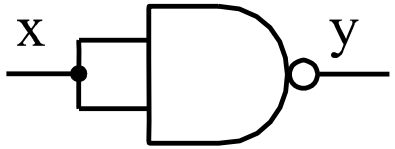
Realizacja funkcji przełączających

Elektryczne elementy bezstykowe



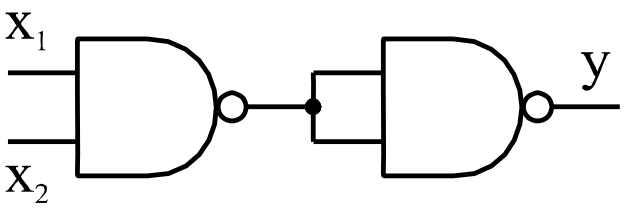
$y = x$

Funkcja powtórzeń



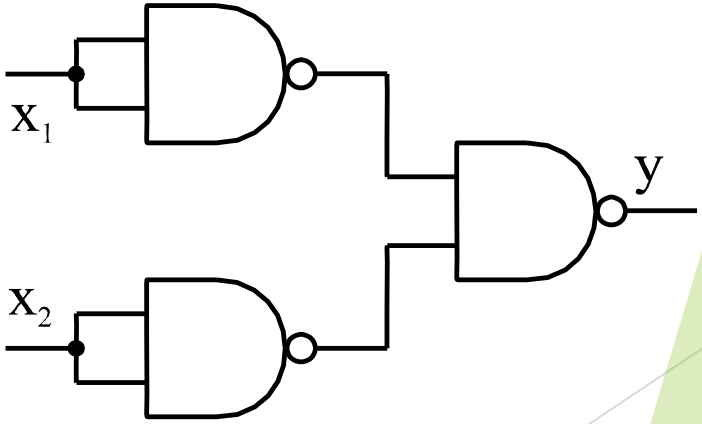
$y = \bar{x}$

Funkcja negacji



$y=x_1x_2$

Funkcja iloczynu



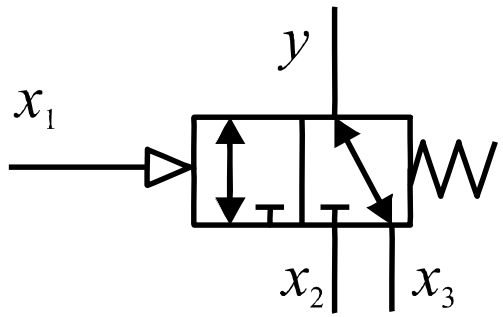
$y=x_1+x_2$

Funkcja sumy

Realizacja funkcji przełączających

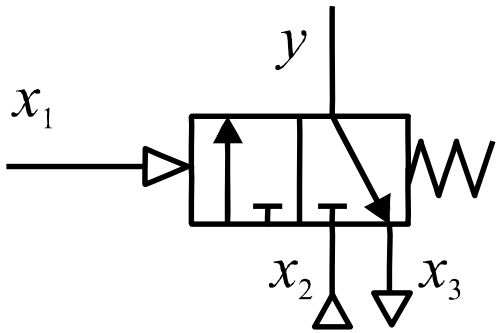
Elementy płynowe

Przykład z wykorzystaniem zaworu rozdzielającego 3/2



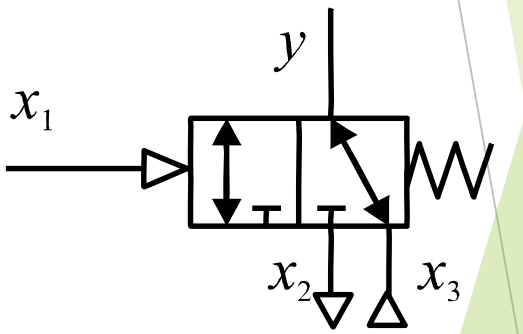
$y = x_1 x_2 + \bar{x}_1 x_3$

a) Ogólna postać funkcji



$y = x_1$

b) Funkcja powtórzeń



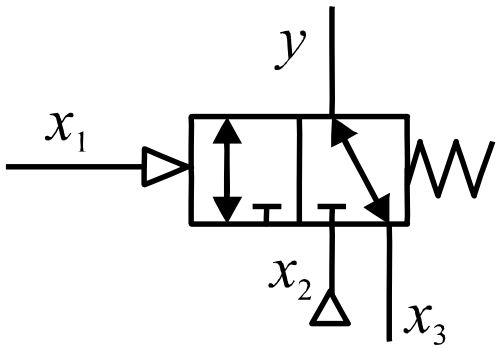
$y = \bar{x}_1$

c) Funkcja negacji

Realizacja funkcji przełączających

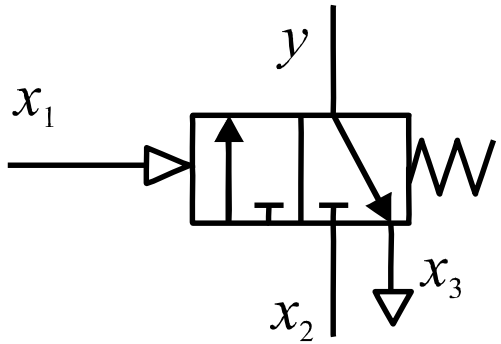
Elementy płynowe

Przykład z wykorzystaniem zaworu rozdzielającego 3/2



$y=x_1+x_3$

d) Funkcja sumy



$y=x_1x_2$

e) Funkcja iloczynu

Sposoby przedstawienia informacji w układach cyfrowych

- Informacje w układach cyfrowych są przedstawione w przyjętym systemie liczenia zwanym systemem pozycyjnym.
- System pozycyjny jest systemem, w którym przedstawiamy liczby za pomocą ciągu znaków (cyfr), przy czym z każdą pozycją w ciągu związana jest określona waga. Wagi poszczególnych pozycji w ciągu są odpowiednimi potęgami podstawy.
- Każdą liczbę naturalną przedstawiamy w tym systemie przez podane ciągu cyfr stanowiących współczynniki, przez które mnożone są wagi poszczególnych pozycji.

.

- Jeśli podstawę systemu liczenia przyjmiemy liczbę naturalną P, to każdą liczbę L możemy przedstawić w tym systemie jako:

$$L = \sum_{i=0}^{n} a_i . P^i \; gdzie \; a_i \in \{0,1,....,P-1\}$$

Przykład:
W systemie dziesiętnym, wartość liczby zapisanej w postaci ciągu cyfr 0, 1,…, 9 jest określona jako suma

$$L = \sum_{i=0}^{n} a_i . 10^i$$

W układach cyfrowych powszechnie stosuje się dwójkowy system liczenia (podstawa 2). Cyfra binarna nosi nazwę bitu. Wada dwójkowego systemu liczenia jest fakt, że liczba pozycji wymagana dla przedstawienia dużych liczb jest kilkakrotnie większa w porównaniu w systemie dziesiętnym.
Przykład:

$$1011010 = 1.2^6 + 0.2^5 + 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 = 64 + 16 + 8 + 2 = 90$$

Operacja arytmetyczne na liczbach w systemie dwójkowym są analogiczne jak w systemie dziesiętnym.

W technice cyfrowej duże znaczenia mają system ósemkowy i szesnastkowy. Niżej podane są oznaczenie i znaki stosowane w systemach pozycyjnych:

B – binarny $\{0,1\}$

Q – ósemkowy $\{0,1,2,3,4,5,,6,7\}$

D – dziesiętny $\{0,1,2,3,4,5,,6,7,8,9\}$

H – szesnastkowy $\{0,1,2,3,4,5,,6,7,8,9,A,B,C,D,E,F\}$

Przykład:

$63201\ Q = 6.8^4 + 3.8^3 + 2.8^2 + 1.8^0 = 6.\ 4096 + 3.512 + 2.64 + 1 = 26241\ D$

$3E82\ H = 3.16^3 + 14.16^2 + 8.16^1 + 2.16^0 = 3.\ 4096 + 14.256 + 8.16 + 2 = 16002\ D$

Zamiana liczb:

Liczba ósemkowa na 3 bitową liczbę dwójkową

3057 Q = (011 000 101 111) = 011000101111 B

Liczba szesnastkowa na 4-bitową liczbę dwójkową

AF9 H = (1010 1111 1001) = 101011111001 B

Odwrotnie

1011101000101 1 B = (010 111 010 001 011) B = 27213 Q

1011101000101 1 B = (0010 1110 1000 1011) B = 2E8B H

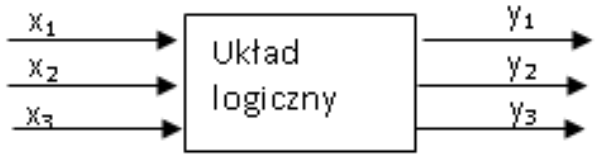
Układy logiczne

- Układy przetwarzające informacje, których sygnały są dwustanowe, nazywamy układami logicznymi (układami przełączającymi, cyfrowymi).
- Dwa stany wielkości fizycznej występującej w tych układach przyjęto oznaczać symbolami 0 i 1.
- Istnieją dwie grupy układów logicznych: kombinacyjne (jednotaktowe) i sekwencyjne (wielotaktowe).
- Układem kombinacyjnym jest taki układ, w którym wartości sygnałów wyjściowych zależą tylko od wartości sygnałów wejściowych w danej chwili.
- Układem sekwencyjnym jest układ, w którym wartości sygnałów wejściowych zależą od wartości sygnałów wejściowych w danej chwili oraz od ich wartości w chwilach poprzednich.

Układy logiczne kombinacyjne

- ✓ Sformułowanie zadania układu logicznego może być w postaci słownej lub tablicy wartości funkcji realizowanej przez układ.
- ✓ Jest to tablica podająca wartości sygnału wyjściowego dla poszczególnych kombinacji wartości sygnałów wejściowych.
- ✓ Proces tworzenia algebraicznego zapisu funkcji, którą ma realizować układ, nazywa się syntezą funkcji. Działanie zmierzające do uzyskania schematu blokowego układu realizującego daną funkcję z uwzględnieniem zastosowanych do jego budowy elementów, nazywa się syntezą strukturalną.
- ✓ Układ kombinacyjny ogólnie może być przedstawiony na rys. 1 i opisany układem równań:

$$y_1 = f_1(x_1, x_2, x_3)$$
$$y_2 = f_2(x_1, x_2, x_3)$$
$$y_3 = f_3(x_1, x_2, x_3)$$



Rys. 1. Schemat układu logicznego.

Dla funkcji logicznej $y = f(x_1, x_2, ..., x_n)$, gdzie argumenty przyjmują tylko dwie wartości 0 albo 1, liczba kombinacji wartości argumentów jest skończona. Każdą kombinację wartości argumentów nazwiemy stanem argumentów. Liczba stanów funkcji n argumentów będzie 2^n. Stan argumentów zapisany w postaci ciągu cyfr, z których każda reprezentuje wartość kolejnego argumentu, tworzy liczbę binarną (w systemie dwójkowym). Liczbę tę nazwiemy numerem stanu. Np. dla funkcji czteroargumentowej numer stanu $(x_1, x_2, x_3, x_4) = (1,1,1,1)$ jest 15.

Funkcja logiczna jest w pełni określona, jeśli znane są wartości zmiennej zależnej dla wszystkich stanów argumentów.

- Każdą funkcję logiczną przedstawić można w tzw. kanonicznej postaci alternatywnej lub równoważnej kanonicznej postaci koniunkcyjnej.
- Kanoniczna postać alternatywna jest sumą składników jedności K_i utworzonych dla stanów argumentów, przy których zmienna zależna jest jedynką.
- Kanoniczna postać koniunkcyjna jest iloczynem czynników zera D_i utworzonych dla stanów argumentów, przy których zmienna zależna jest zerem.
- Tablica 1 podaje czynniki zera i składniki jedności funkcji trzech argumentów i sposób ich numeracji.

Tab. 1. Składniki jedności K_i i czynniki zera D_i funkcji trzyargumentowych

Nr	X_1	X_2	X_3	D_i	K_i
0	0	0	0	$D_0 = x_1 + x_2 + x_3$	$K_0 = \overline{x_1}.\overline{x_2}.\overline{x_3}$
1	0	0	1	$D_1 = x_1 + x_2 + \overline{x_3}$	$K_1 = \overline{x_1}.\overline{x_2}.x_3$
2	0	1	0	$D_2 = x_1 + \overline{x_2} + x_3$	$K_2 = \overline{x_1}.x_2.\overline{x_3}$
3	0	1	1	$D_3 = x_1 + \overline{x_2} + \overline{x_3}$	$K_3 = \overline{x_1}.x_2.x_3$
4	1	0	0	$D_4 = \overline{x_1} + x_2 + x_3$	$K_4 = x_1.\overline{x_2}.\overline{x_3}$
5	1	0	1	$D_5 = \overline{x_1} + x_2 + \overline{x_3}$	$K_5 = x_1.\overline{x_2}.x_3$
6	1	1	0	$D_6 = \overline{x_1} + \overline{x_2} + x_3$	$K_6 = x_1.x_2.\overline{x_3}$
7	1	1	1	$D_7 = \overline{x_1} + \overline{x_2} + \overline{x_3}$	$K_7 = x_1.x_2.x_3$

Przykład: Zbudować układ logiczny sterujący systemem alarmowym w danym pomieszczeniu. System składa się z 3 czujników temperaturowych. Układ uruchamia alarm w przypadku załączenia co najmniej 2 czujników.

Tab. 2. Tabela wartości przykładowej funkcji logicznej y trzyargumentowej

Nr	X_1	X_2	X_3	Y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Przykładowo wypisano postacie kanoniczne funkcji zdefiniowanej w tab. 2.

$$y = K_3 + K_5 + K_6 + K_7 = \overline{x_1}.x_2.x_3 + x_1.\overline{x_2}.x_3 + x_1.x_2.\overline{x_3} + x_1.x_2.x_3$$

$$y = D_0.D_1.D_2D_4 = (x_1 + x_2 + x_3)(x_1 + x_2 + \overline{x_3})(x_1 + \overline{x_2} + x_3)(\overline{x_1} + x_2 + x_3)$$

Podany system numeracji czynników zera i składników jedności umożliwia stosowanie uproszczonego zapisu postaci kanonicznych funkcji. Np. dla funkcji podanej w tab.2 zapisujemy

$$y(x_1, x_2, x_3) = \sum(3,5,6,7) \quad \text{lub} \quad y(x_1, x_2, x_3) = \prod(0,1,2,4)$$

Metoda Karnaugh’a

Do ułatwienia procesu przekształceń algebraicznych i minimalizacji funkcji logicznych służą tablice Karnaugh’a. Wnętrze tablicy wypełnia się wartościami funkcji *y* dla poszczególnych stanów argumentów x_i, wypisanych na obrzeżu. Stany argumentów są tak ułożone, aby składniki jedności (lub czynniki zera) utworzonych dla sąsiednich pól różniły się tylko jednym znakiem negacji, a więc nadają się do sklejania wg wzorów.

$$xy + x\bar{y} = x \quad \text{lub} \quad (x + y)(x + \bar{y}) = x$$

W celu ułatwienia wypełniania tablic na podstawie tablicy wartości funkcji, poszczególnym pola odpowiadające numerom stanu zostały ponumerowane.

Tab. 3. Tabela Karnaugha dla podanej funkcji trzyargumentowej

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

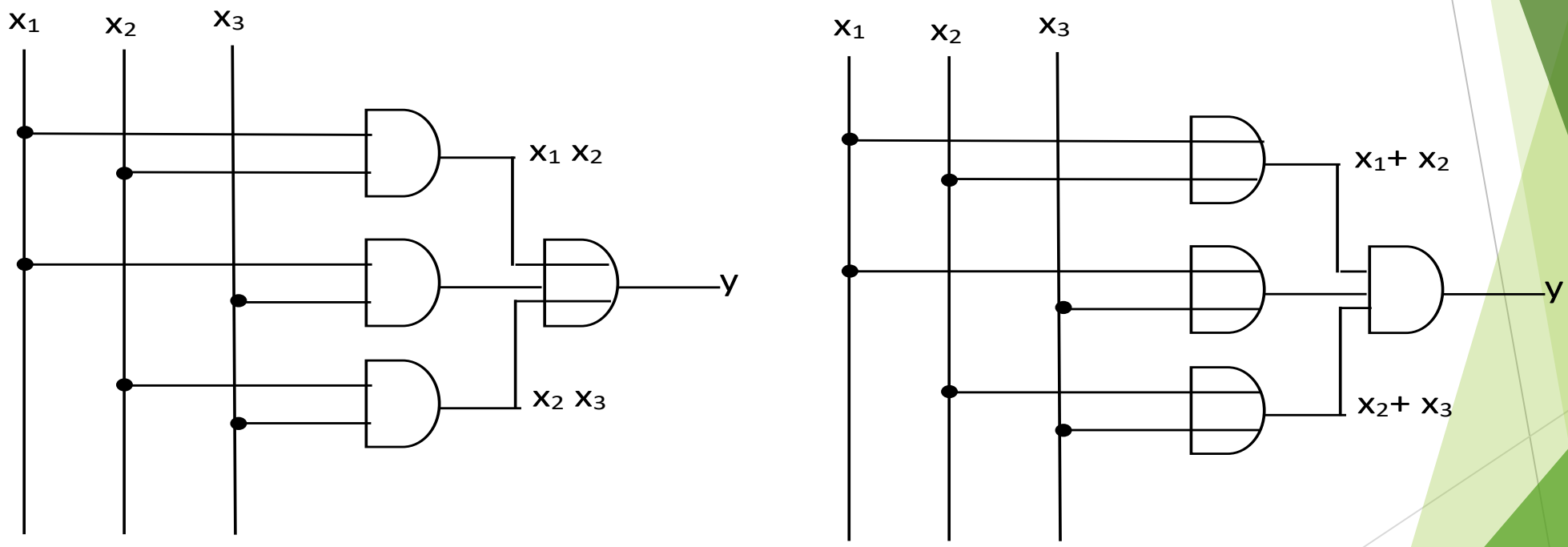
Postać kanoniczna alternatywna funkcji y

$$y = K_3 + K_5 + K_6 + K_7 = \overline{x_1}.x_2.x_3 + x_1.\overline{x_2}.x_3 + x_1.x_2.\overline{x_3} + x_1.x_2.x_3$$

$$= (K_3 + K_7) + (K_5 + K_7) + (K_6 + K_7) = x_2.x_3 + x_1.x_3 + x_1.x_2$$

Postać kanoniczna koniunkcyjna funkcji y

$$y = (D_0.D_1).(D_0.D_2).(D_0.D_4) = (x_1 + x_2)(x_1 + x_3)(x_2 + x_3)$$

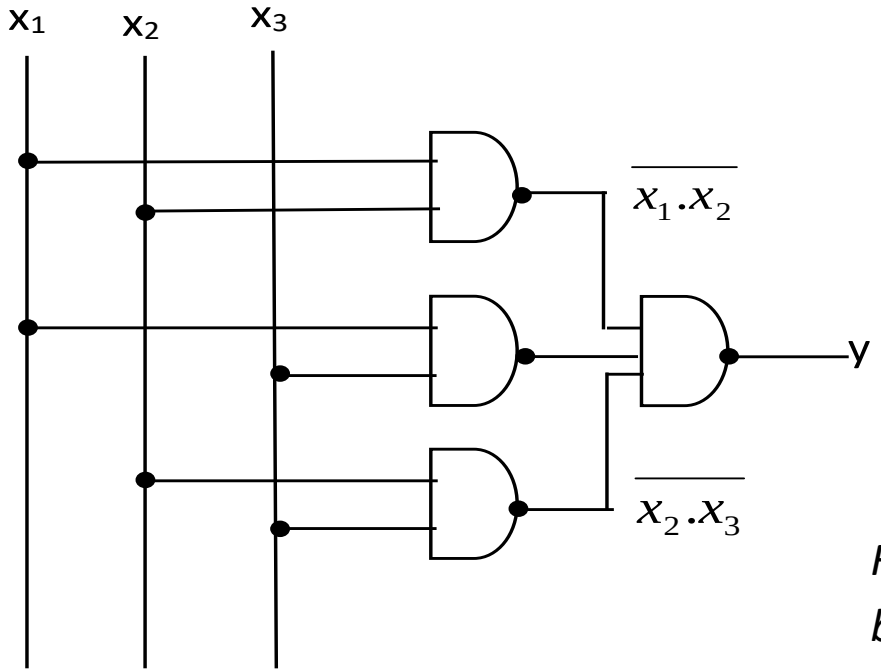


Rys.2. Schemat blokowy budowy układu logicznego: a) zminimalizowanego metodą sklejania jedynek, b) zminimalizowanego metodą sklejania zer.

W przypadku stosowania wyłącznie jednego typu bramek logicznych jak NAND lub NOR do budowy układu logicznego, postać zminimalizowaną funkcji y trzeba przekształci za pomocą prawa Morgan'a. Np.:

$$y = x_2.x_3 + x_1.x_3 + x_1.x_2 = \overline{\overline{x_2.x_3 + x_1.x_3 + x_1.x_2}} = \overline{(\overline{x_2.x_3})(\overline{x_1.x_3})(\overline{x_1.x_2})} \text{ lub}$$

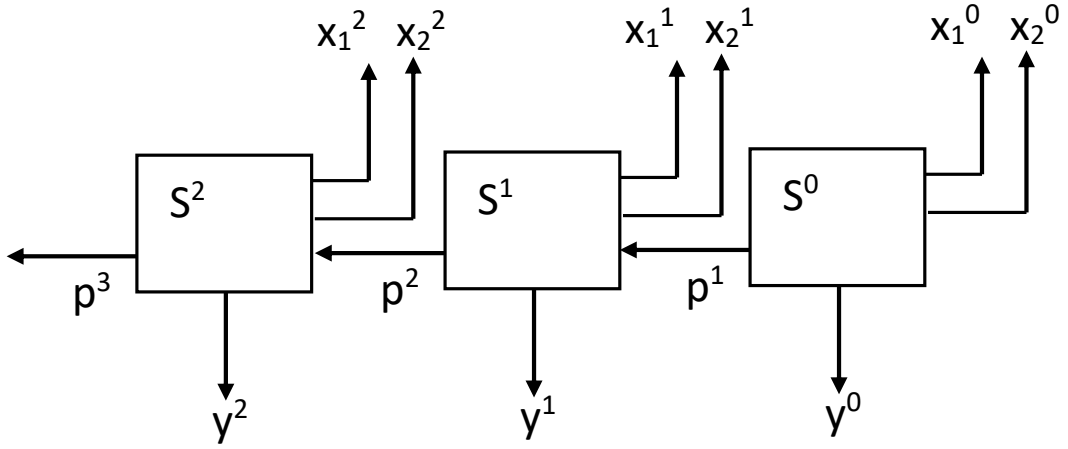
$$y = (x_1 + x_2)(x_1 + x_3)(x_2 + x_3) = (\overline{\overline{x_1 + x_2}})(\overline{\overline{x_1 + x_3}})(\overline{\overline{x_2 + x_3}}) = \overline{(\overline{x_1}\,\overline{x_2})(\overline{x_1}\,\overline{x_3})(\overline{x_3}\,\overline{x_2})}$$



Rys.3. Schemat blokowy układu logicznego zbudowanego z bramek NAND.

Sumator liczb binarnych

- Jedną z metod maszynowego dodawania liczb binarnych jest sumowanie równoległe. Sumator wykonujący operację dodawania składa się z połączonych szeregowo, tzw. sumatorów jednopozycyjnych, oznaczonych na rysunku 4 kwadratami.
- Sygnałami wejściowymi sumatora jednopozycyjnego są wartości cyfr danej pozycji składników sumy x_1, x_2 oraz przeniesienie p^i z pozycji niższej. Sygnałami wyjściowymi są sygnał sumy danej pozycji oraz sygnał przeniesienia do pozycji wyższej.



Rys.4. Schemat blokowy równoległego sumatora dwójkowego.

Sumator liczb binarnych

x_1	x_2	p^{n-1}	y^n	p^n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tab. 5. Tabela wartości dla sumatora jednopozycyjnego

p^{n-1} \ x_1,x_2	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Tab. 6. Tabela Karnaugha dla funkcji y^{n+1}

$$y^{n+1} = \bar{x}_1\bar{x}_2p^n + \bar{x}_1x_2\bar{p}^n + x_1\bar{x}_2\bar{p}^n + x_1x_2p^n$$
$$= (\bar{x}_1\bar{x}_2 + x_1x_2)p^n + (\bar{x}_1x_2 + x_1\bar{x}_2)\bar{p}^n$$
$$= \left(\overline{x_1 \bigoplus x_2}\right)p^n + \left(x_1 \bigoplus x_2\right)\bar{p}^n$$
$$= \left(x_1 \bigoplus x_2\right) \bigoplus p^n$$

p^{n-1} \ x_1,x_2	00	01	11	10
0	0	0	1	0
1	0	1	1	1

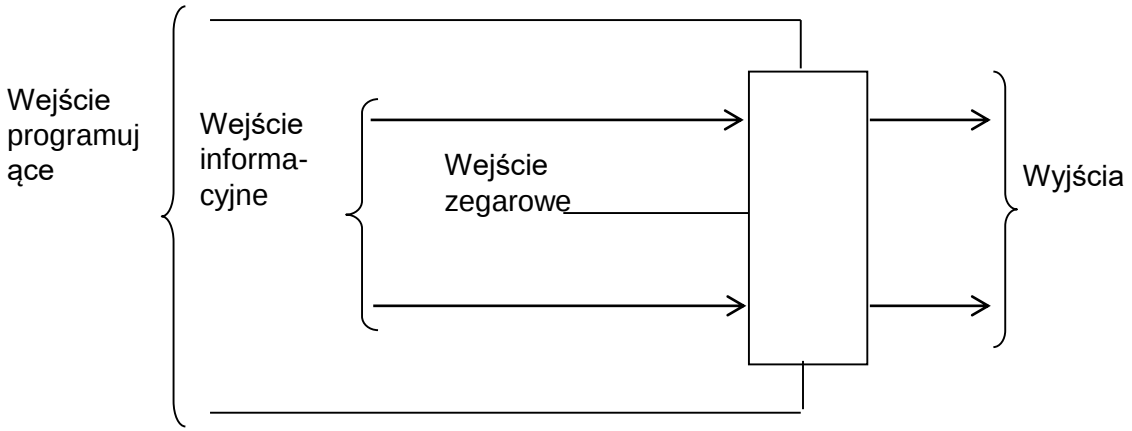
Tab. 7. Tabela Karnaugh'a dla funkcji p^{n+1}

$$p^{n+1} = x_1x_2 + x_2p^n + x_1p^n = \overline{\overline{x_1x_2}.\overline{x_2p^n}.\overline{x_1p^n}}$$

Układy logiczne sekwencyjne

- W układach sekwencyjnych stany sygnałów wyjściowych zależą od stanów sygnałów wejściowych w danej chwili i w chwilach poprzednich. Stan sygnałów przechowujących informacje o poprzednich stanach wejść nazywamy stanem wewnętrznym. Stan wewnętrzny układu jest zapamiętany w urządzeniach zwanych pamięcią lub przerzutnikiem.
- Przerzutnik jest podstawowym elementem układów sekwencyjnych. Jego funkcja polega na zapamiętaniu jednego bitu informacji. Przerzutnik posiada dwa stany wewnętrzne z możliwością przejść w obu kierunkach (z 1 na 0 i z 0 na 1).
- Zmiana stanu przerzutnika występuje pod wpływem zmiany sygnałów wejściowych. Ze względu na moment zmiany stanu przerzutnika, dzielą się na:
  - asynchroniczne,
  - synchroniczne.

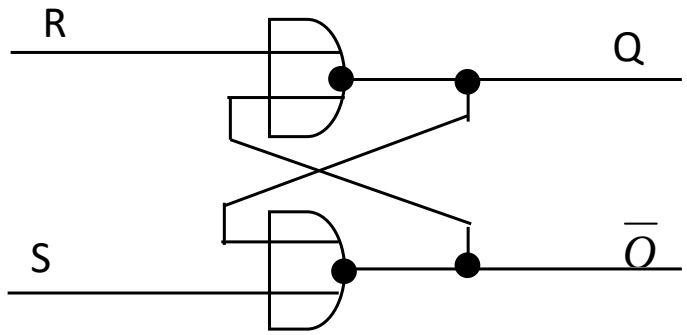
- Przerzutniki asynchroniczne są to układy, w których zmiana stanu następuje natychmiast po zmianie wartości sygnałów wejściowych. Przerzutnik synchroniczny jest układem mającym dwa rodzaje wejść:
  - wejście informacyjne,
  - wejście zegarowe.
- Zmiana stanu przerzutnika może nastąpić tylko w czasie trwania sygnału zegarowego. Stan przerzutnika zależy od wartości sygnałów informacyjnych oraz od stanu poprzedniego. W przerzutniku synchronicznym mogą występować asynchroniczne wejścia programujące pozwalające na asynchroniczne ustalenia stanu przerzutnika. Wejścia asynchroniczne mają zawsze priorytet w stosunku do pozostałych wejść przerzutnika.



Rys.1. Symbol graficzny przerzutnika

Schemat budowy przerzutnika RS wykonanego z dwóch bramek NOR odpowiednio połączonych ze sobą przedstawia rys.2. Przerzutnik ten posiada dwa wejścia programujące R i S oraz dwa wyjścia, Q i zanegowane $\overline{Q}$. Stany wejść R i S oddziałują asynchronicznie na stan przerzutnika. Kombinacja sygnałów wejściowych R=0 i S=1 ustawia na wyjściu stan 0, kombinacja R=1 i S=0 powoduje ustawienie na wyjściu przerzutnika stan 1, natomiast podczas trwania kombinacji R=S=0 pamiętany jest stan przerzutnika, który został ostatnio ustawiony. Kombinacja sygnałów wejściowych R=S=1 jest zabroniona ponieważ stan na wyjściu przerzutnika jest nieokreślony

$$Q_{n+1} = \overline{R + \overline{Q}_n} = \overline{R}\left(S + Q_n\right)$$
$$\overline{Q}_{n+1} = \overline{S + Q_n} = \overline{S}\left(R + \overline{Q}_n\right)$$



R	0	0	0	0	1	1	1	1
S	0	0	1	1	0	0	1	1
Q_n	0	1	0	1	0	1	0	1
Q_{n+1}	0	1	1	1	0	0	-	-

Rys.2. Asynchroniczny przerzutnik typu RS zbudowany z bramek NOR:

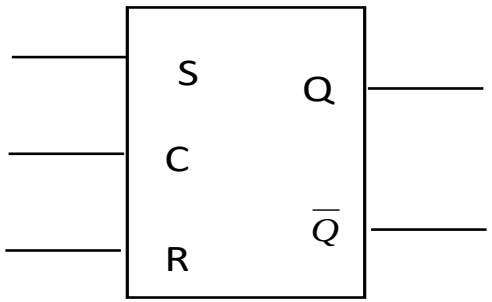
a) schemat logiczny, b) tablica stanów.

2. Synchroniczne przerzutnik typu RS

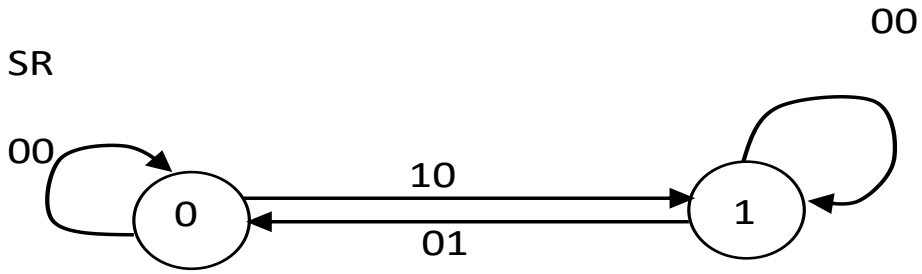
- Symbol graficzny synchronicznego przerzutnika RS i jego działanie logiczne zilustrowano na rys. 3. Wejściami informacyjnymi są synchronizowane wejścia R i S. Wejście C jest wejściem dla impulsów zegarowych. Przerzutnik podczas wyzwalania go impulsem zegarowym:
  - nie zmienia stanu, jeśli R=S=0,
  - ustawi się w stanie 0, jeśli R=1 i S=0,
  - ustawi się w stanie 1, jeśli R=0 i S=1.
  - Stan przerzutnika będzie nieokreślony dla R=S=1.
- Równanie logiczne przerzutnika RS otrzymane na podstawie tablicy Karnaugh'a ma postać:

$$Q_{n+1} = S_n + \overline{R_n} \cdot Q_n$$

a)



b)



c)

$R_n\ S_n$

Q_n	00	01	11	10
0	0	0	-	1
1	1	0	-	1

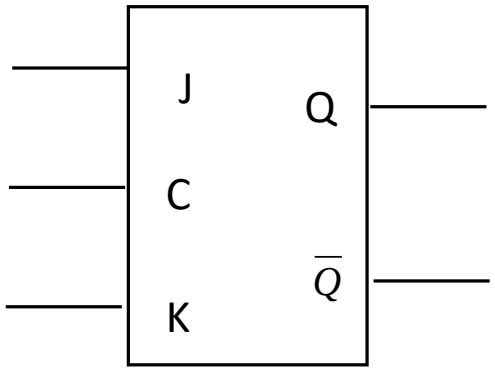
Rys.4. Synchroniczny przerzutnik typu RS: a) symbol graficzny, b) graf, c) tablica Karnaugh'a dla Q_{n+1}.

3. Synchroniczne przerzutnik typu JK

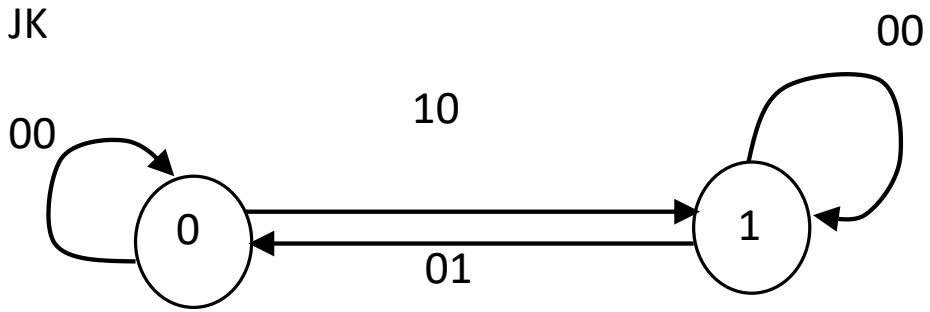
- Wejścia informacyjne tego przerzutnika oznaczone J i K odpowiadają wejściom S i R przerzutnika RS.
- Jedyna różnica w działaniu obu typów polega na tym, że w przypadku R=S=1 stan przerzutnika RS jest nieokreślony, natomiast dla J=K=1 stan następny przerzutnika JK będzie negacją stanu aktualnego.
- Przerzutnik JK jest funkcjonalnie najbardziej uniwersalny. Równanie logiczne przerzutnika JK ma postać:

$$Q_{n+1} = J_n \overline{Q_n} + \overline{K_n} \cdot Q_n$$

a)



b)



c)

$J_n\ K_n$

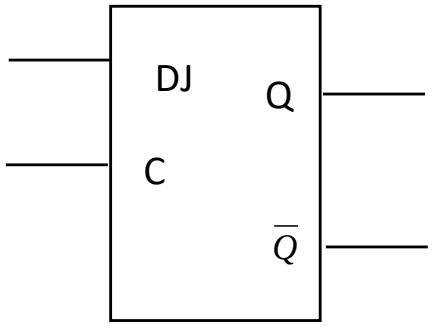
Q_n	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Rys.6. Synchroniczny przerzutnik typu JK: a) symbol graficzny, b) graf, c) tablica Karnaugh'a dla Q_{n+1}.

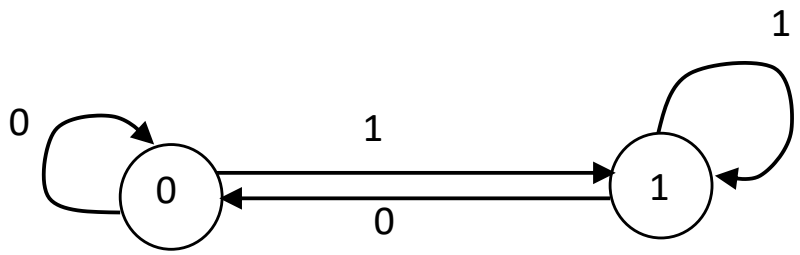
4. Synchroniczne przerzutnik typu D

Symbol graficzny synchronicznego przerzutnika typu D i jego działanie logiczne przedstawiono na rys. 7. Równanie logiczne przerzutnika D ma postać: $Q_{n+1} = D_n$

a)



b)



c)

Q_{n+1}

Q_n \ D_n	*0*	*1*
0	0	1
1	0	1

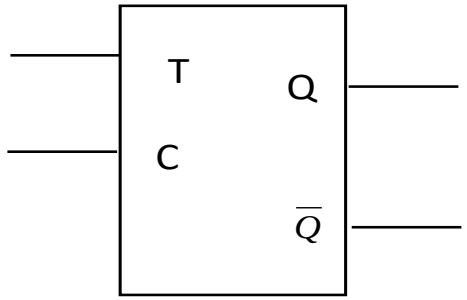
Rys.7. Synchroniczny przerzutnik typu D: a) symbol graficzny, b) graf, c) tablica Karnaugh'a dla Q_{n+1}.

5. Synchroniczne przerzutnik typu T

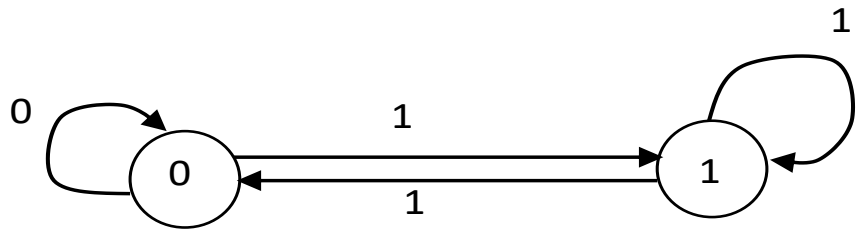
Symbol graficzny synchronicznego przerzutnika typu T i jego działanie logiczne przedstawiono na rys. 8. Równanie logiczne przerzutnika T ma postać:

$$Q_{n+1} = T_n \overline{Q_n} + \overline{T_n} Q_n$$

a)



b)



c)

Q_{n+1}

Q_n \ T_n	*0*	*1*
0	0	1
1	1	0

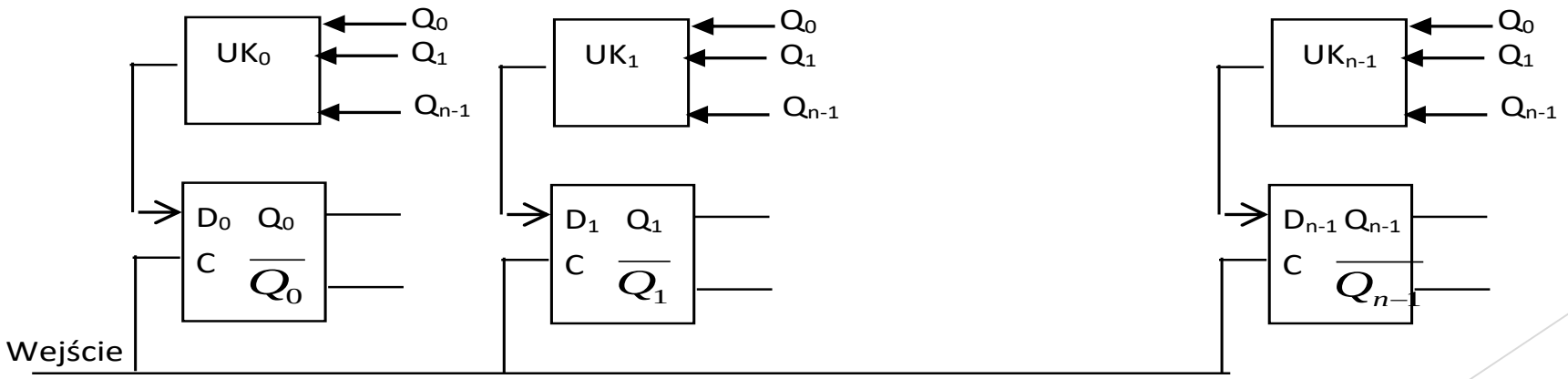
Rys.8. Synchroniczny przerzutnik typu T: a) symbol graficzny, b) graf, c) tablica Karnaugh'a dla Q_{n+1}.

Synteza układów sekwencyjnych

- Najbardziej typowe układy sekwencyjne to liczniki i rejestry. Strukturę bardzie złożonych układów sekwencyjnych można na ogół tak rozdzielić, aby wyodrębnić w nich te dwa typy układów.

Licznik synchroniczne.

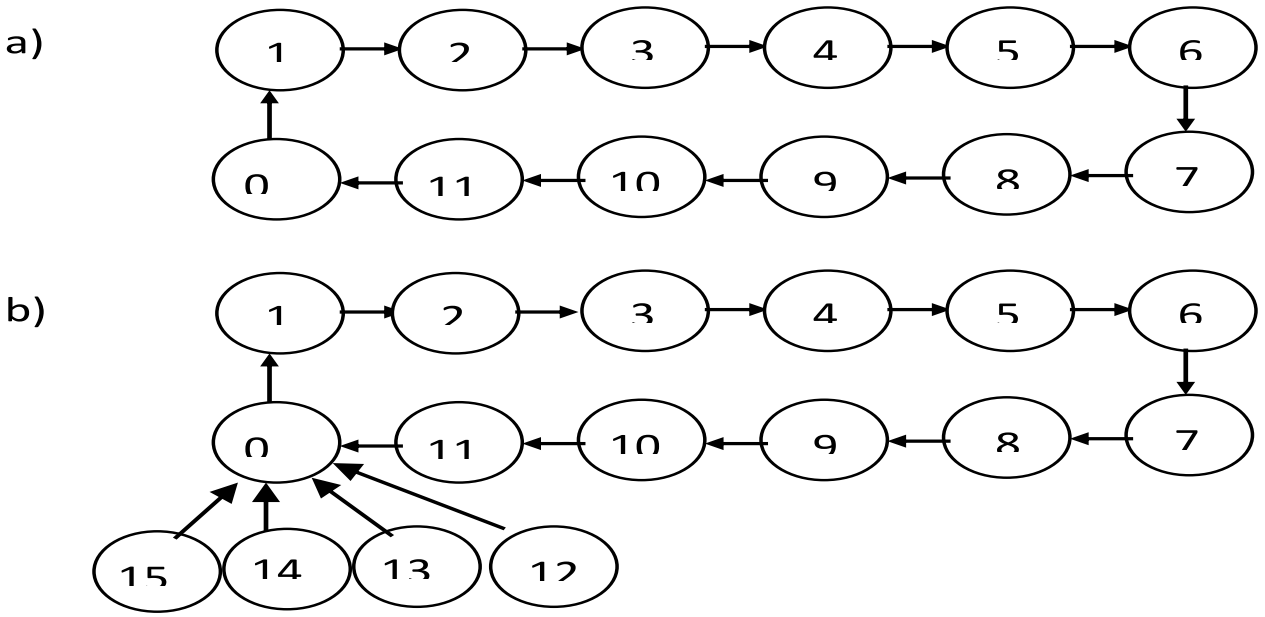
- Liczniki synchroniczne wykonuje się w ten sposób, że wejścia zegarowe wszystkich przerzutników wchodzących w skład licznika są połączone razem i wysterowane jednym sygnałem. Wejścia informacyjne przerzutnika są wysterowane przez układy kombinacyjne tak dobrane, aby licznik zmieniał swoje stany według pożądanego grafu przejść. Struktura licznika synchronicznego zbudowanego z przerzutników typu D jest okazana na rys.9.



Rys.9. Ogólna struktura licznika synchronicznego zbudowanego na przerzutnikach typu D.

Licznik synchroniczne.

Na rys. 10a pokazany jest graf przejść licznika modulo 12, natomiast na rys. 10b graf tego samego układu, ale z uwzględnieniem założenia, że licznik zeruje się dla stanów większych od 11, przechodząc do stan 0 i powtarza pętlę podstawową.



Rys.2. Graf przejść licznika synchronicznego modulo 12: a) bez zerowania, b) z zerowaniem stanów przypadkowych.

Licznik synchroniczne.

- ✓ Realizacja projektu licznika wg grafu 10a będzie prostsza.
- ✓ Poniżej przedstawiono metodę wyznaczania funkcji wzbudzeń dla poszczególnych wejść. Założono wykorzystanie przerzutników typu D. Stąd wynika konieczność wyznaczenia 4 funkcji D_0, D_1, D_2, D_3 na wejściach informacyjnych tych przerzutników.
- ✓ W tym celu należy przenieść graf do tablicy Karnaugh'a i posługując się tablicą wzbudzeń przerzutnika typu D, wypełnić cztery tablice Karnaugh'a dla wszystkich wejść. Następnie należy zminimalizować funkcje wejściowe.
- ✓ Ten sam układ zrealizowany na przerzutnikach JK będzie wymagał znacznie prostszych funkcji sterujących wejściami informacyjnymi przerzutników.

Licznik synchroniczne.

Przy projektowaniu układów synchronicznych należy wykonać w kolejności następujących czynności:

- Określić liczbę przerzutników k na podstawie wartości najwyższego stanu n: $k \leq \log_2(n+1)$
- Na podstawie grafu i tablicy wzbudzeń przerzutnika zrealizować tablicę Karnaugh'a dla każdego z wejść informacyjnych przerzutników i zminimalizować je,
- Otrzymane funkcje logiczne zrealizować za pomocą bramek logicznych.

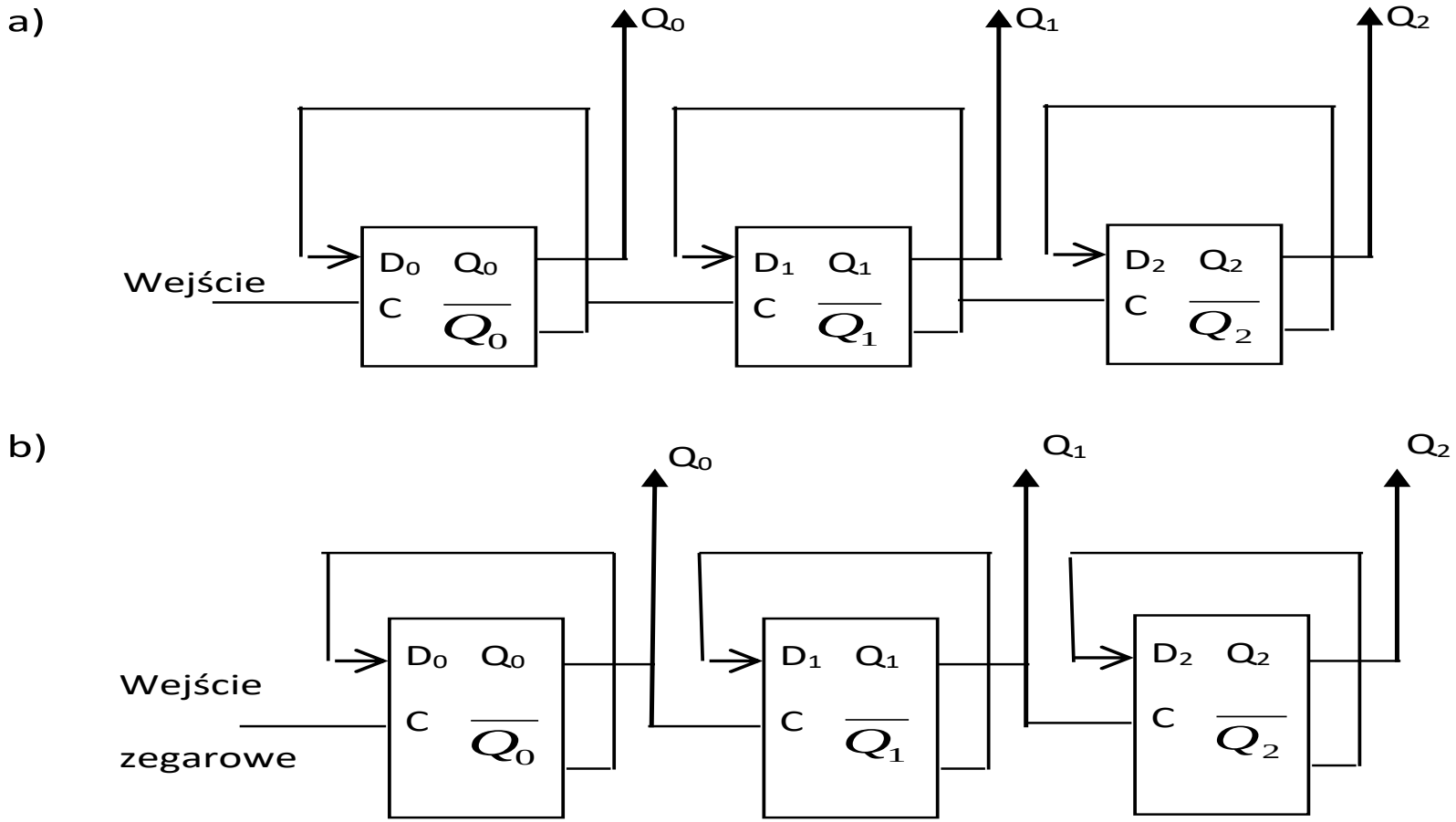
Szybkość pracy liczników synchronicznych jest duża. Częstotliwość graniczną licznika wyznacza czas propagacji jednego przerzutnika powiększony o czas propagacji sygnału przez najdłuższą ścieżkę w układzie kombinacyjnym.

2. Liczniki asynchroniczne

W licznikach asynchronicznych zliczane impulsy są podawane na jeden lub na niektóre z wejść zegarowych przerzutników. Najprostsze liczniki asynchroniczne są zbudowane przez kaskadowe połączenie, tzw. dwójek liczących, czyli liczników modulo 2, tworząc licznik modulo 2^n liczący do przodu czy do tyłu, w zależności od sposobu sprzęgania poszczególnych przerzutników ze sobą.

Aby otrzymać licznik liczący do przodu należy na wejście zegarowe kolejnego przerzutnika podać sygnał z wyjścia $\overline{Q}$ poprzedzającego przerzutnika. W przypadku licznika liczącego do tyłu należy na wejście zegarowe podać sygnał z wyjścia Q poprzedzającego przerzutnika. Pokazuje to rysunek 4, który przedstawia dwa liczniki modulo 8, liczący do przodu i do tyłu.

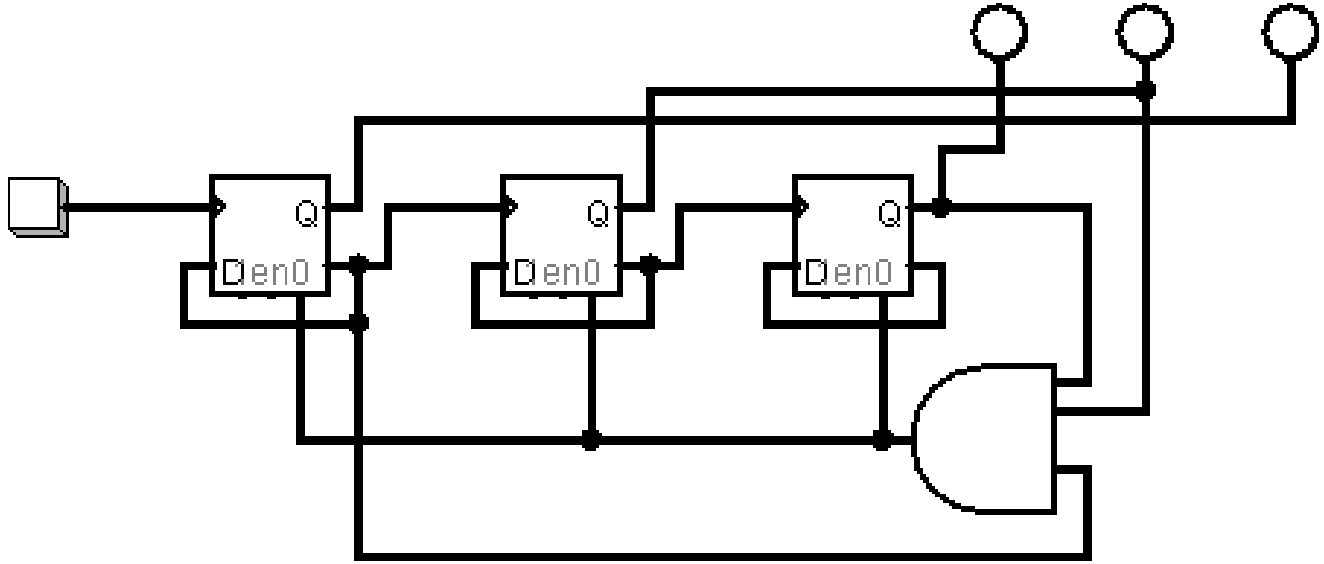
2. Liczniki asynchroniczne



Rys.4. Licznik asynchroniczny modulo 8: a) liczący do przodu, b) liczący do tyłu.

2. Liczniki asynchroniczne

W przypadku licznika modulo n, gdzie $n \neq 2^k$, naturalną pętlę licznika asynchronicznego należy skrócić. Do tego celu stosuje się układ kombinacyjny wykrywający stan n, którego wyjście podawane jest na wejścia zerujące wszystkich przerzutników. Przykład licznika asynchronicznego modulo 6 przedstawia rysunek 5. Układem kombinacyjnym wykrywającym stan 5 (110) jest tu bramka AND.



Rys.5. Licznik asynchroniczny modulo 6 z układem zerującym przy stanie nr 5

3. Liczniki synchroniczne

W licznikach synchronicznych zliczane impulsy są podawane na wszystkie wejścia zegarowe przerzutników. Na podstawie tablicy stanów licznika i tablicy wzbudzeń przerzutników jest zrealizowana tablica Karnaugh'a dla każdego z wejść informacyjnych przerzutników i zminimalizowana. Otrzymane funkcje logiczne zrealizowane są za pomocą bramek logicznych (rys. 6).

Przykład 6. Zbudować licznik realizujący sekwencję liczenia

$0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 7$, zbudowany z przerzutników typu D.

W tym przypadku tablica wzburzeń jest taka sama jak tablica wyjściowa licznika.

3. Liczniki synchroniczne

Tablica stanów i tablica wzburzeń

Stan	Q(t)			Q(t+1)					
	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	D_2	D_1	D_0
0	0	0	0	0	0	1	0	0	1
1	0	0	1	0	1	0	0	1	0
2	0	1	0	1	0	0	1	0	0
4	1	0	0	1	0	1	1	0	1
5	1	0	1	1	1	1	1	1	1
7	1	1	1	0	1	0	0	1	0

3. Liczniki synchroniczne

Tablice Karnaugh'a

Dla D_2

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	0	0	X	1
1	1	1	0	X

$$D_2 = \overline{Q_2}Q_1 + \overline{Q_1}Q_2 = Q_2 \oplus Q_1$$

Dla D_1

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	0	1	X	0
1	0	1	1	X

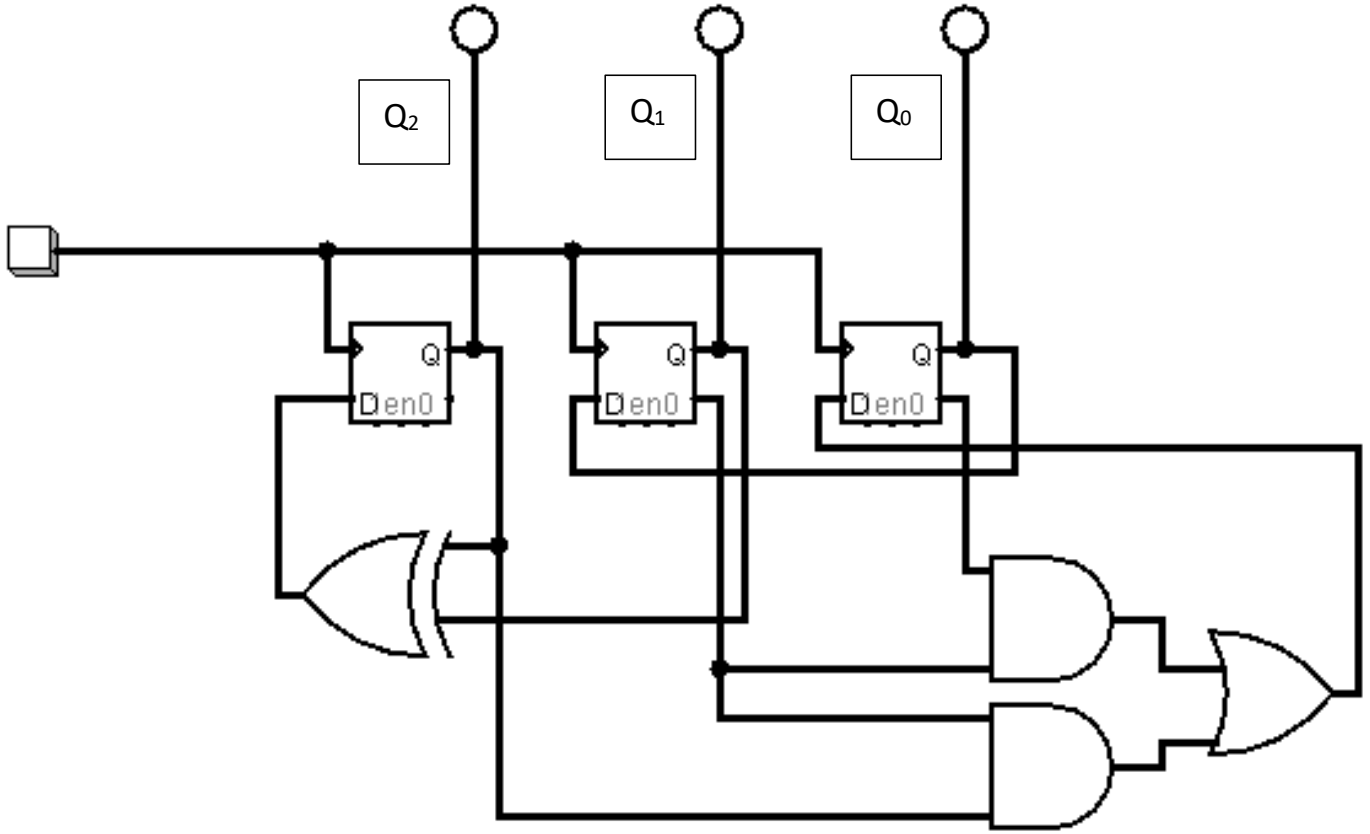
$$D_1 = Q_0$$

Dla D_0

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	1	0	X	0
1	1	1	0	X

$$D_0 = \overline{Q_0}\,\overline{Q_1} + \overline{Q_1}Q_2$$

3. Liczniki synchroniczne



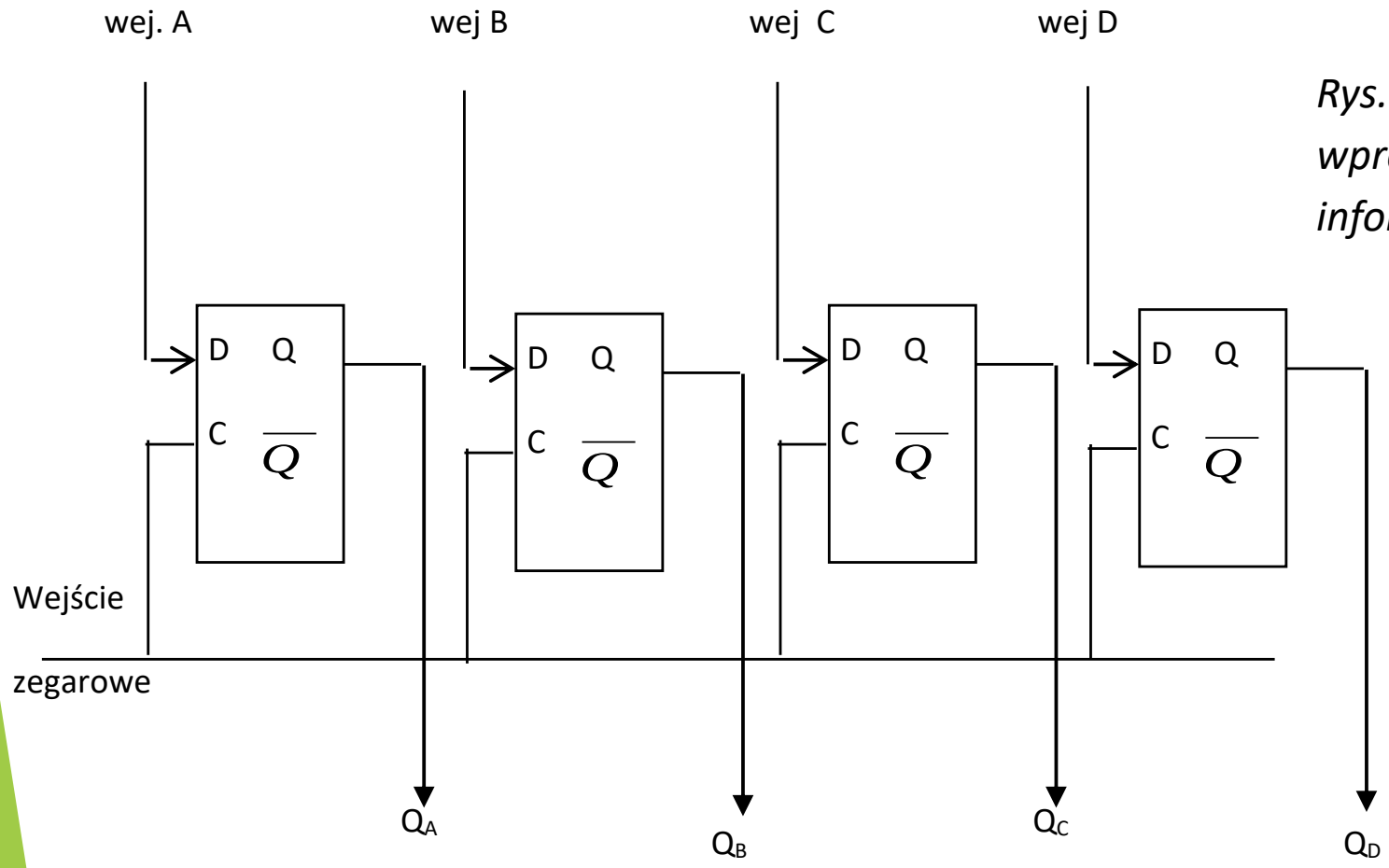
Rys.6. Licznik synchroniczny przykładu 6.

3. REJESTRY

- Rejestrami są układy służące do przechowywania informacji kilkubitowej.
- Informacja może być do rejestru wprowadzana i wyprowadzana równolegle lub szeregowo. W tym drugim przypadku mamy do czynienia z rejestrem przesuwnym. Jest to układ, w którym informacja podlega przemieszczeniu w lewo lub w prawo w takt impulsów zegarowych.
- Do budowy rejestrów najwygodniej jest używać przerzutników typu D, których istota działania polega na zapamiętaniu informacji podane na wejścia D. Na rys. 1 pokazano przykładowe rozwiązania rejestrów.

3. REJESTRY

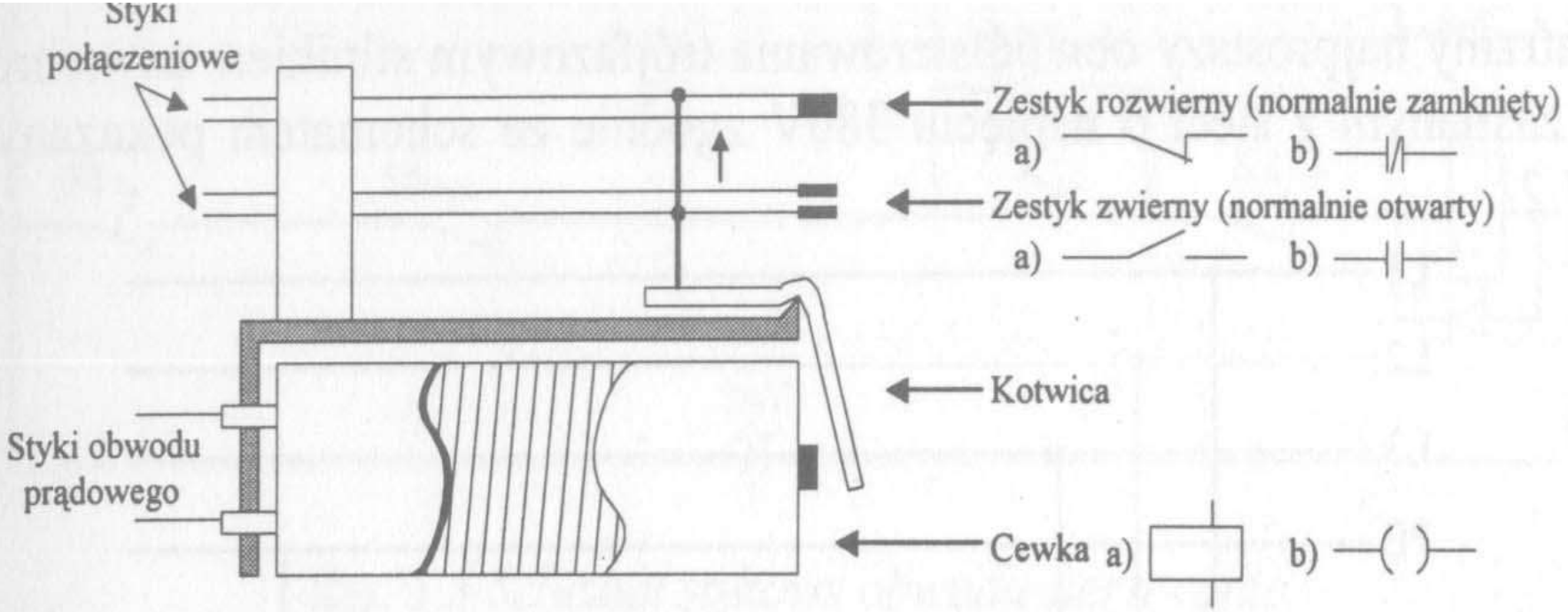
- Do budowy rejestrów najwygodniej jest używać przerzutników typu D, których istota działania polega na zapamiętaniu informacji podane na wejścia D. Na rysunku 6 pokazano przykładowe rozwiązania rejestrów.



Rys.6. Rejestr czterobitowy z równoległym wprowadzaniem i wyprowadzaniem informacji.

Wykład 5. Przemysłowe układy sterowania binarnego

Konstrukcja przekaźnika elektromechanicznego pokazana na Rys. 1.1.
Od tego czasu zmniejszyły się jego fizyczne wymiary i pobór mocy, wzrosła szybkość działania, wytrzymałość prądowa i napięciowa, a przede wszystkim wzrosła jego niezawodność.



a) oznaczenie na schematach stykowych
b) oznaczenie w języku schematów drabinkowych

Rys. 1.1 Konstrukcja przekaźnika elektromechanicznego

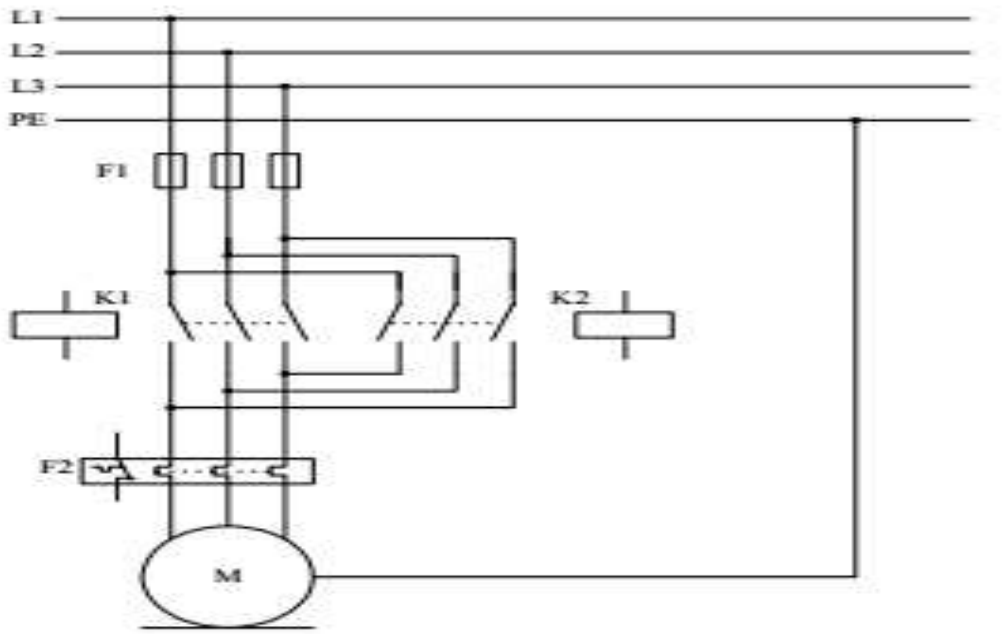
Wykład 5. Przemysłowe układy sterowania binarnego

- Główną część takiego przekaźnika stanowi nadal ruchoma kotwica zapewniająca elektryczne połączenie w obwodach, w których umieszczono styki tego przekaźnika. Działanie tych styków (zwiernych lub rozwiernych) jest zależne od sygnału w obwodzie wzbudzenia cewki przekaźnika.
- Cewki przekaźników umieszczone w obwodach wejściowych są pobudzane w zależności od położenia styków obiektowych (czujników), załączników i wyłączników. Zgodnie z zaprojektowanymi warunkami logicznymi i czasowymi zapewniają one pobudzanie cewek przekaźników, a przy większych obciążeniach prądowych, przekaźników pośredniczących lub styczników, których styki umieszczone w obwodach mocy zapewniają włączanie i wyłączanie urządzeń wykonawczych takich jak napędy sterujące, regulacyjne, urządzenia grzewcze, itp.
- Chociaż obecnie stosuje się znacznie częściej łączniki elektroniczne z wykorzystaniem tranzystorów i tyrystorów, zarówno w niskoprądowych obwodach sterujących jak i w wysokoprądowych obwodach mocy, to dla zrozumienia zasad programowania sterowników konieczne jest podanie podstawowych danych o przekaźnikach. Stosowane przez dziesiątki lat przekaźnikowe systemy sterowania były bowiem wykonywane niejednokrotnie z wielu tysięcy takich przekaźników.
- Wszystkie funkcje podobnych systemów, a często znacznie bardziej złożonych, realizują obecnie programy sterowników. Programy profesjonalne powinny być więc pisane zgodnie z zasadami projektowania i w oparciu o doświadczenia praktyczne, które przez wiele lat wypracowano przy konstruowaniu układów przekaźnikowych.

Wykład 5. Przemysłowe układy sterowania binarnego

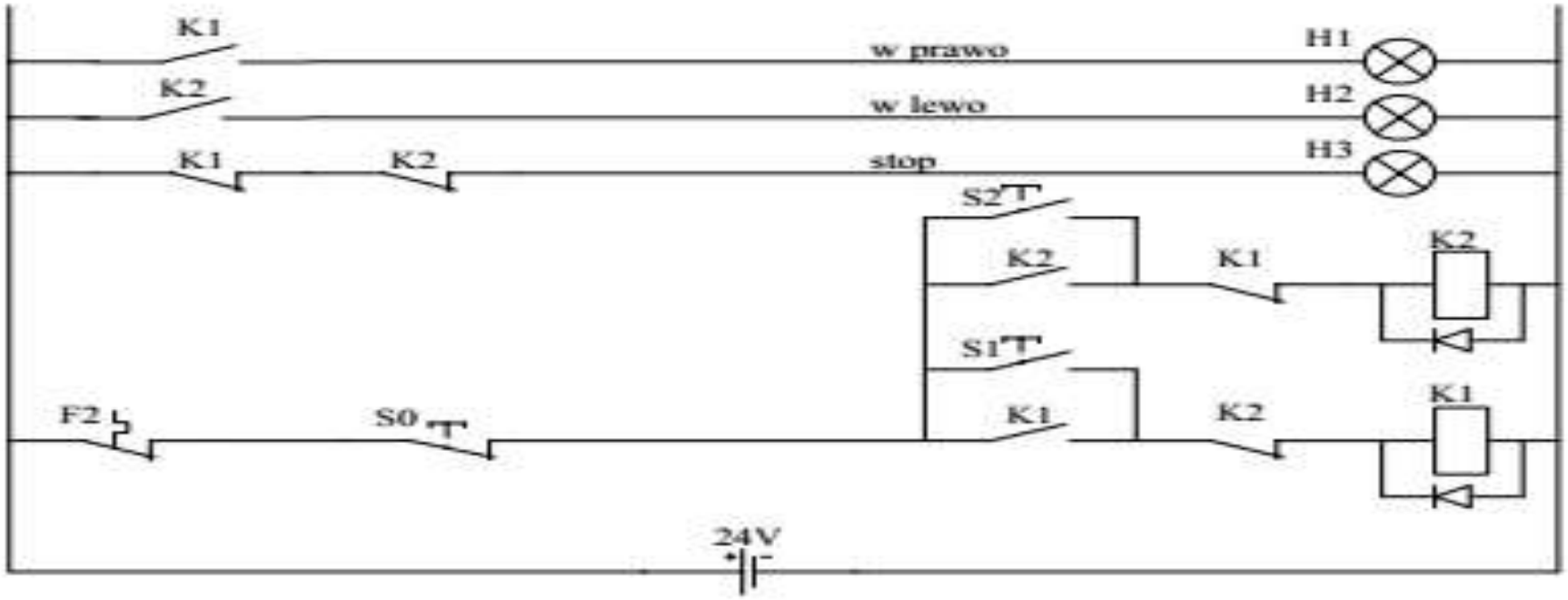
Rozpatrzmy najprostszy obwód sterowania trójfazowym silnikiem asynchronicznym, zasilanym z sieci o napięciu 380V zgodnie ze schematem pokazanym na Rys. 1.2.
Silnik jest połączony ze źródłem napięcia przez styki przekaźników K1 albo K2, w zależności od wybranego przez przyciski sterownicze kierunku wirowania: S1 (w prawo) i S2 (w lewo), zaś jego wyłączenie następuje za pomocą przycisku wyłączającego S0 (stop). Załóżmy, że jest to silnik jednobiegowy z jednopoziomową ochroną termiczną. Ochrona ta działa pod wpływem nagrzania paska bimetalowego działającego na styk wyłączający F2. Styk ten umieszczony jest w obwodzie sterowania i zapewnia termiczne zabezpieczenie silnika.

1.4 Przykład zadania sterowania napędem rewersyjnym



Wykład 5. Przemysłowe układy sterowania binarnego

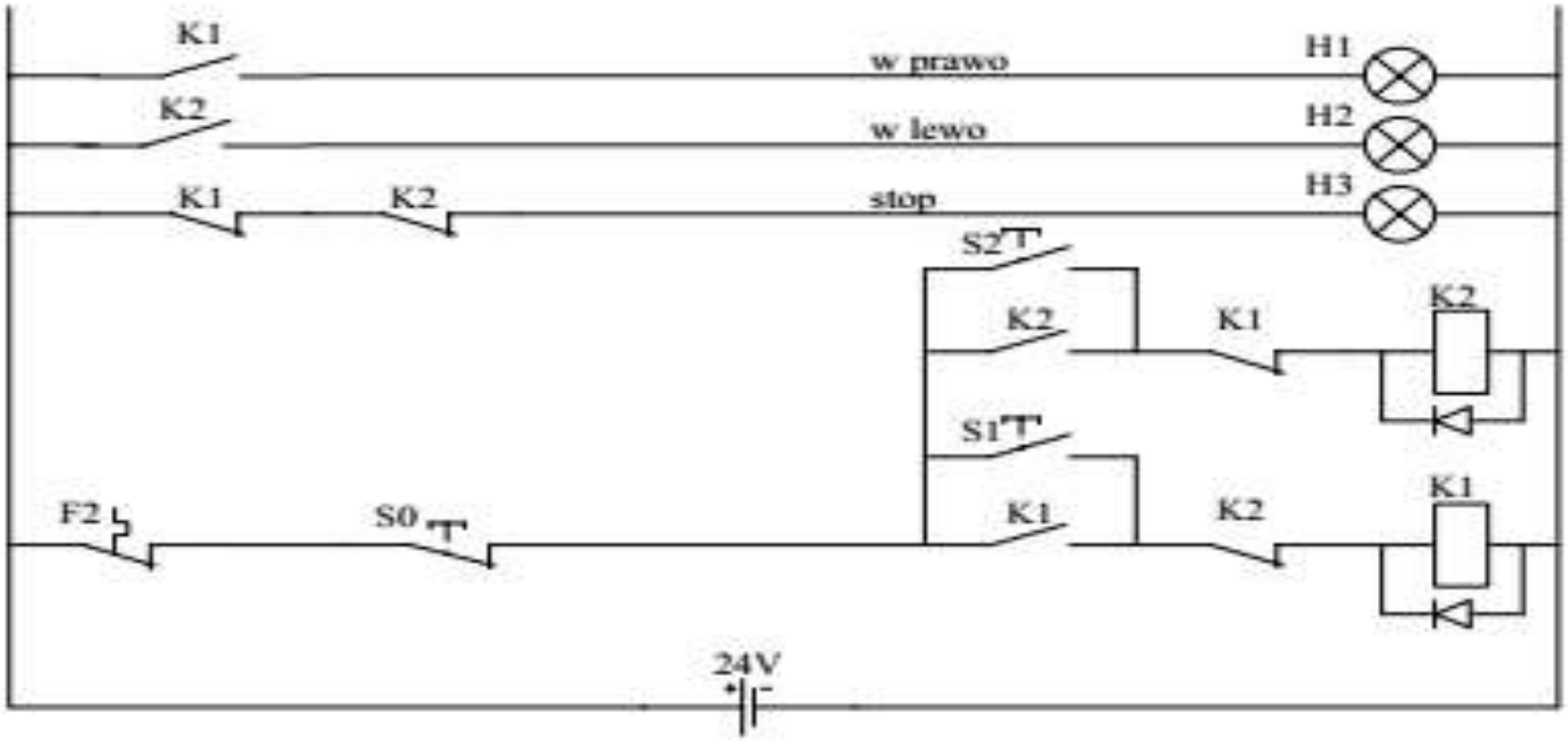
Rys. 1.3 przedstawia schemat stykowy układu sterowania silnikiem. W układzie tym załączenie silnika następuje przyciskami sterowniczymi załączającymi kierunek wirowania w prawo przez S1, podobnie w lewo przez S2. Po zwolnieniu przycisku S1 napięcie w obwodzie cewki K1 przekaźnika załączającego silnik jest podtrzymywanie przez styk tego przekaźnika (oznaczony również przez K1). Ponowne naciśnięcie S1 w tym układzie sterowania nic nie zmienia. Podobnie działają przycisk S2 i przekaźnik K2. Zanik napięcia w obwodach cewek następuje po naciśnięciu przycisku wyłączającego stop (styk rozwierny S0). Styk rozwierny F2 zabezpieczenia termicznego silnika może przerwać obwód sterowania, podobnie jak styk przycisku S0. Natomiast styki rozwierne K1 i K2 umieszczone w obwodach obu cewek uniemożliwiają jednoczesne ich pobudzenie.



Rys. 1.3 Schemat stykowy obwodu sterowania

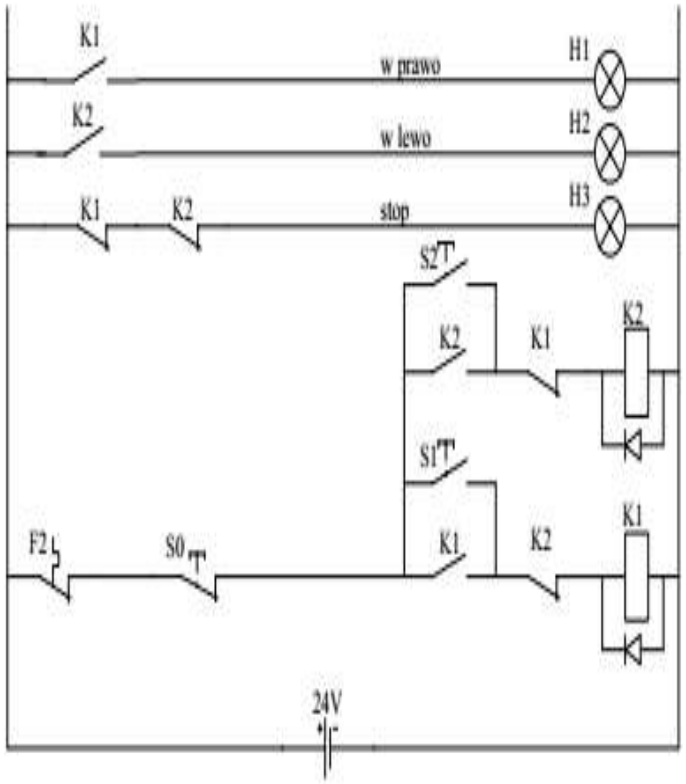
Wykład 5. Przemysłowe układy sterowania binarnego

W układzie tym zapewniono sygnalizację pracy i postoju silnika za pomocą lampek, a ponadto wprowadzono zabezpieczenia przed napięciami, które indukują się w cewkach w chwili rozwierania styków wyłączających. Zabezpieczenie to stanowią odpowiednio włączone diody, które dla prądów indukcyjnych zamykają obwód cewki.

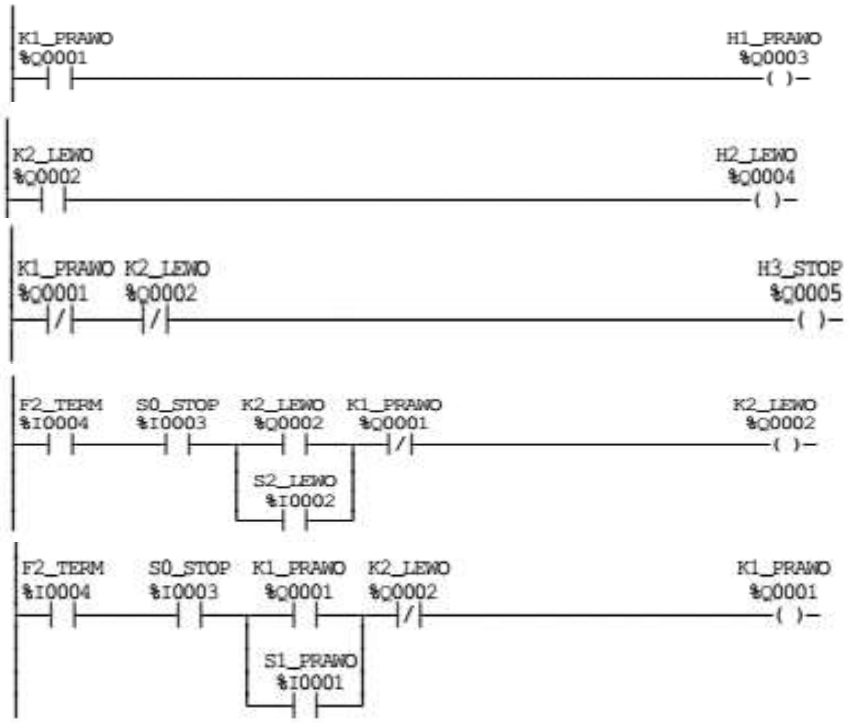


Rys. 1.3 Schemat stykowy obwodu sterowania

Programowanie sterowników programowalnych PLC



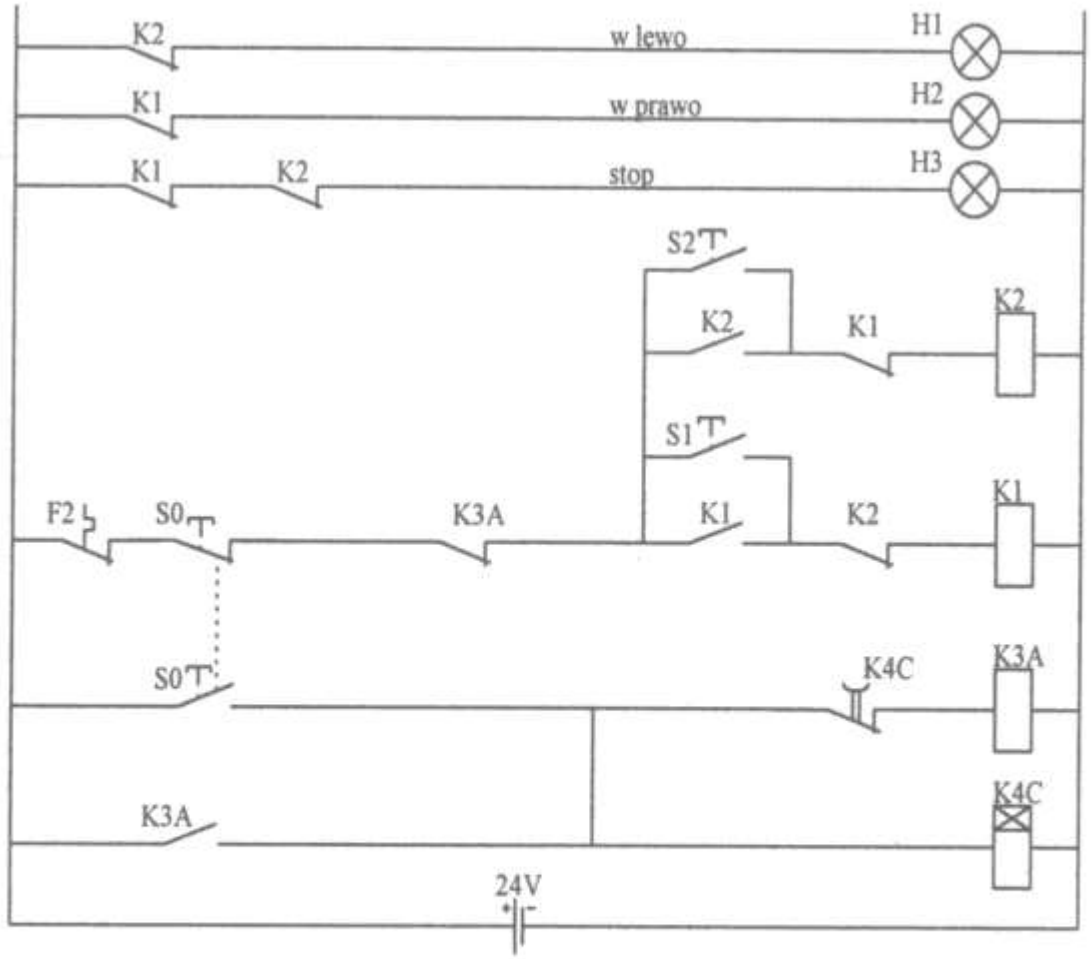
Rys. 1.3 Schemat stykowy obwodu sterowania



Wykład 5. Przemysłowe układy sterowania
Przełączanie silnika z opóźnieniem

Zakładając, że w układzie sterowania silnikiem, po zatrzymaniu silnika za pomocą przycisku S0 (stop) można go ponownie uruchomić dopiero po zadanym czasie, należy zaprojektować schemat stykowy obwodu sterowania, który będzie realizował jego funkcje.

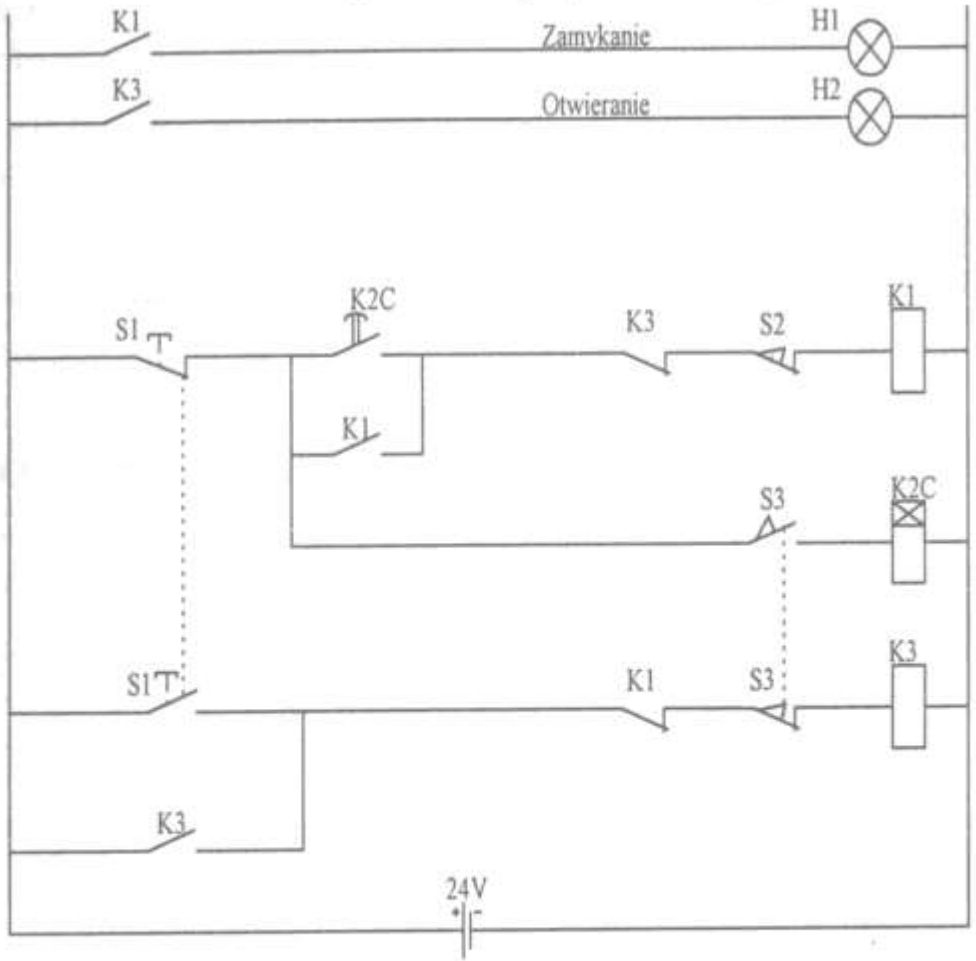
Rozwiązanie pokazane na schemacie Rys. 1.10 różni się od poprzedniego jedynie ostatnimi dwoma szczeblami, których zadaniem jest spowodowanie opóźnienia włączenia silnika po chwili naciśnięcia przycisku S0 (stop) w zależności od nastawy przekaźnika czasowego K4A.



Rys. 1.10 Schemat stykowy obwodu sterowania silnikiem z opóźnionym przełączaniem.

Wykład 5. Przemysłowe układy sterowania
Automatyczne otwieranie i zamykanie drzwi

- Zakładając, że silnik pokazany na Rys. 1.2 ma być wykorzystywany do automatycznego otwierania i zamykania drzwi, należy zaprojektować schemat stykowa obwodu sterowania i napisać program dla sterownika, który będzie realizował je go funkcje.
- W rozwiązaniu pokazanym na Rys. 1.11 przyjęto, że działanie układu rozpoczyna się w chwili włączenia przez wchodzącego lub wychodzącego styku S1 (np. fotokomórki). Następnie przekaźnik K3 włącza otwieranie drzwi aż do najechani na wyłącznik krańcowy S3. Po czym, po przytrzymaniu drzwi otwartych przez przekaźnik czasowy K2C, drzwi zaczynają się zamykać dzięki zadziałaniu przekaźnika K1 aż do napotkania wyłącznika krańcowego S2. Zamykanie to następuje o ile w międzyczasie nie został włączony przez fotokomórkę S1.



Rys. 1.11 Schemat stykowy obwodu sterowania silnikiem wykorzystywanym do automatycznego otwierania i zamykania drzwi

Wykład 6. Budowa i działanie sterowników programowalnych PLC

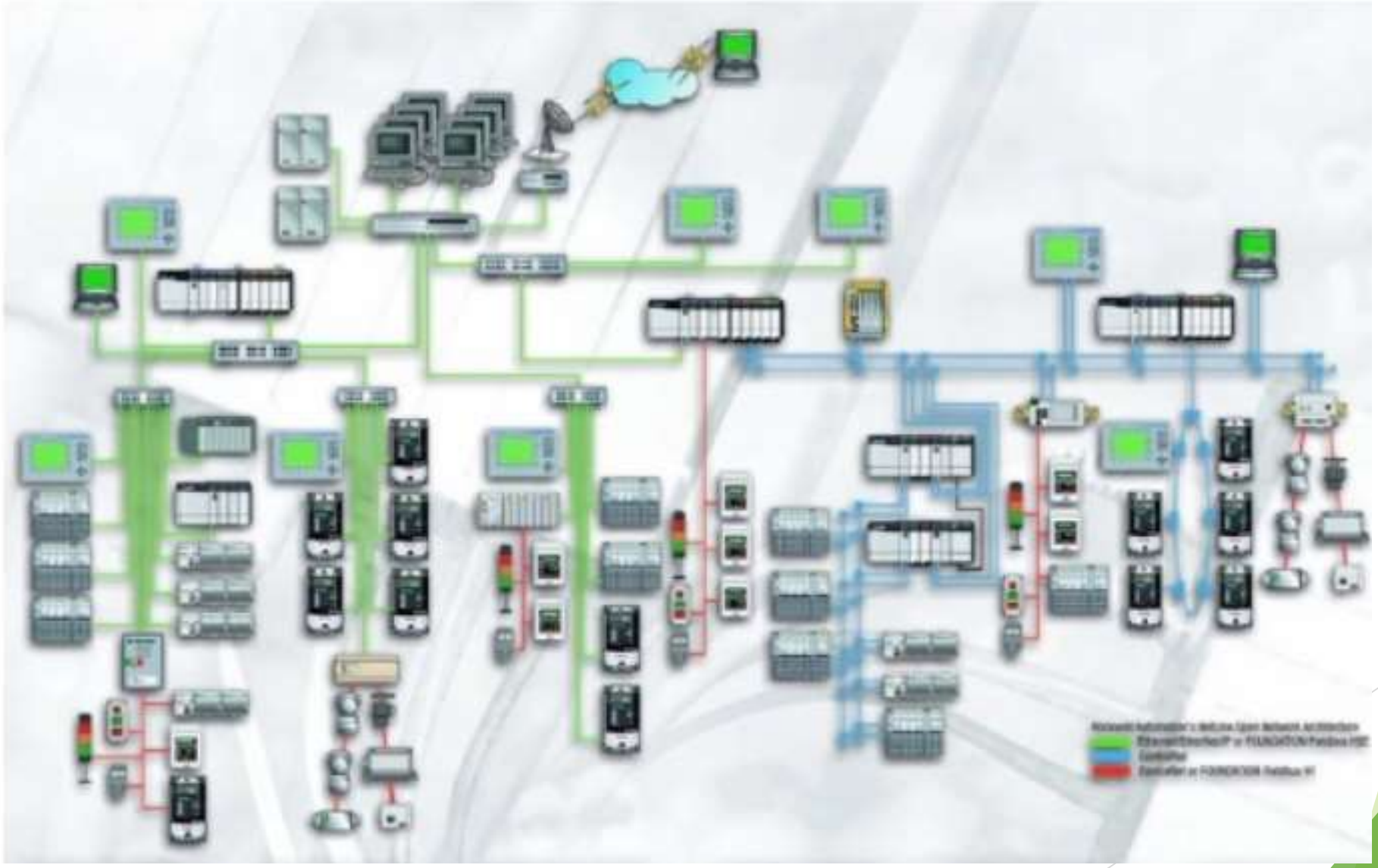
1. WPROWADZENIE

Hierarchiczna struktura systemu *sterowania i zarządzania*:

- warstwa czujników i elementów wykonawczych (w tym rozproszone systemy we/wy, urządzenia inteligentne) − interfejs z procesem,
- warstwa sterowania bezpośredniego (sterowanie napędami, układy regulacji itd.) – sterowniki programowalne (PLC),
- warstwa sterowania nadrzędnego i operatorskiego − systemy *SCADA* na komputerach PC, urządzenia interfejsu operatora (MMI),
- warstwa zarządzania (sterowanie produkcją, zasobami, bazy danych, harmonogramowanie itp.).

Hierarchia sieci LAN: od sieci polowych (*fieldbus*) i przemysłowych po Ethernet (także Internet)

Kierunek rozwoju: *Totally Integrated Automation, Totally Distributed Control*:



1.1 Sterowniki programowalne PLC

Sterowniki programowalne (*PLC*, ang. *Programmable Logic Controllers*) są komputerami przemysłowymi, które pod kontrolą systemu operacyjnego czasu rzeczywistego:

- *zbierają pomiary* za pośrednictwem modułów wejściowych z cyfrowych i analogowych czujników oraz urządzeń pomiarowych;
- korzystając z uzyskanych danych o sterowanym procesie lub maszynie *wykonują programy użytkownika*, zawierające zakodowane algorytmy sterowania i przetwarzania danych;
- *generują sygnały sterujące* odpowiednie do wyników obliczeń tych programów i przekazują je poprzez moduły wyjściowe do elementów i urządzeń wykonawczych;

a ponadto mają możliwość:

- *transmitowania danych* za pomocą modułów i łączy komunikacyjnych;
- *realizacji funkcji diagnostyki* programowej i sprzętowej.

Wartości pomiarów zmiennych procesowych stanowią *wejścia sterownika*, zaś obliczone zmienne sterujące są jego *wyjściami*.

Wykład 6. Budowa i działanie sterowników programowalnych PLC

Norma IEC 61131-1 określa sterownik programowalny jako:

> *„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."*

Rozróżnienia:

- algorytm sterowania realizowany sprzętowo a realizowany programowo
- sterownik programowalny a sterownik dedykowany

Wykład 6. Budowa i działanie sterowników programowalnych PLC

Historia *sterowników programowalnych* sięga roku 1968, gdy w firmie *General Motors* grupa inżynierów rozpoczęła prace projektowe nad nową generacją sterowników, przyjmując następujące założenia:

Łatwość programowania i przeprogramowywania, stosownie do zmieniających się warunków pracy.

Łatwość utrzymania w ruchu produkcyjnym, z możliwością napraw przez wymianę instalowanych modułów (ang. *plug-in modules*).

Większa niezawodność w warunkach przemysłowych, przy mniejszych gabarytach niż sprzęt przekaźnikowy.

Koszty porównywalne ze stosowanymi panelami przekaźnikowymi i szafami sterowniczymi.

Do rozszerzenia produkcji i zastosowań sterowników PLC przyczyniły się głównie:

- łatwość programowania przy użyciu języka schematów drabinkowych podobnego do schematów stykowo-przekaźnikowych;
- zwiększenie niezawodności komputerów przemysłowych na tyle, aby mogły działać w zanieczyszczonym środowisku;
- wprowadzenie programowej kontroli obwodów wejściowych i wyjściowych, oraz innych możliwości diagnostyki systemowej i obiektowej;
- zapewnienie komunikacji z gniazdami przemysłowymi, panelami operatorskimi, wyświetlaczami, komputerami osobistymi oraz innymi urządzeniami stanowiącymi łącze operatora (*MMI*, ang. *Man Machine Interface*).

Rodziny sterowników charakteryzują się tym, że poszczególne modele:

- mogą być programowane w tym samym języku i przy użyciu tego samego pakietu programowego;
- posiadają takie same zmienne programowe oraz taką samą strukturę modułów I/O (moduły, płyty łączeniowe, drajwery, kable łączeniowe itp.);
- istnieje możliwość przenoszenia programów między modelami oraz korzystania z tych samych opcji w każdym modelu.

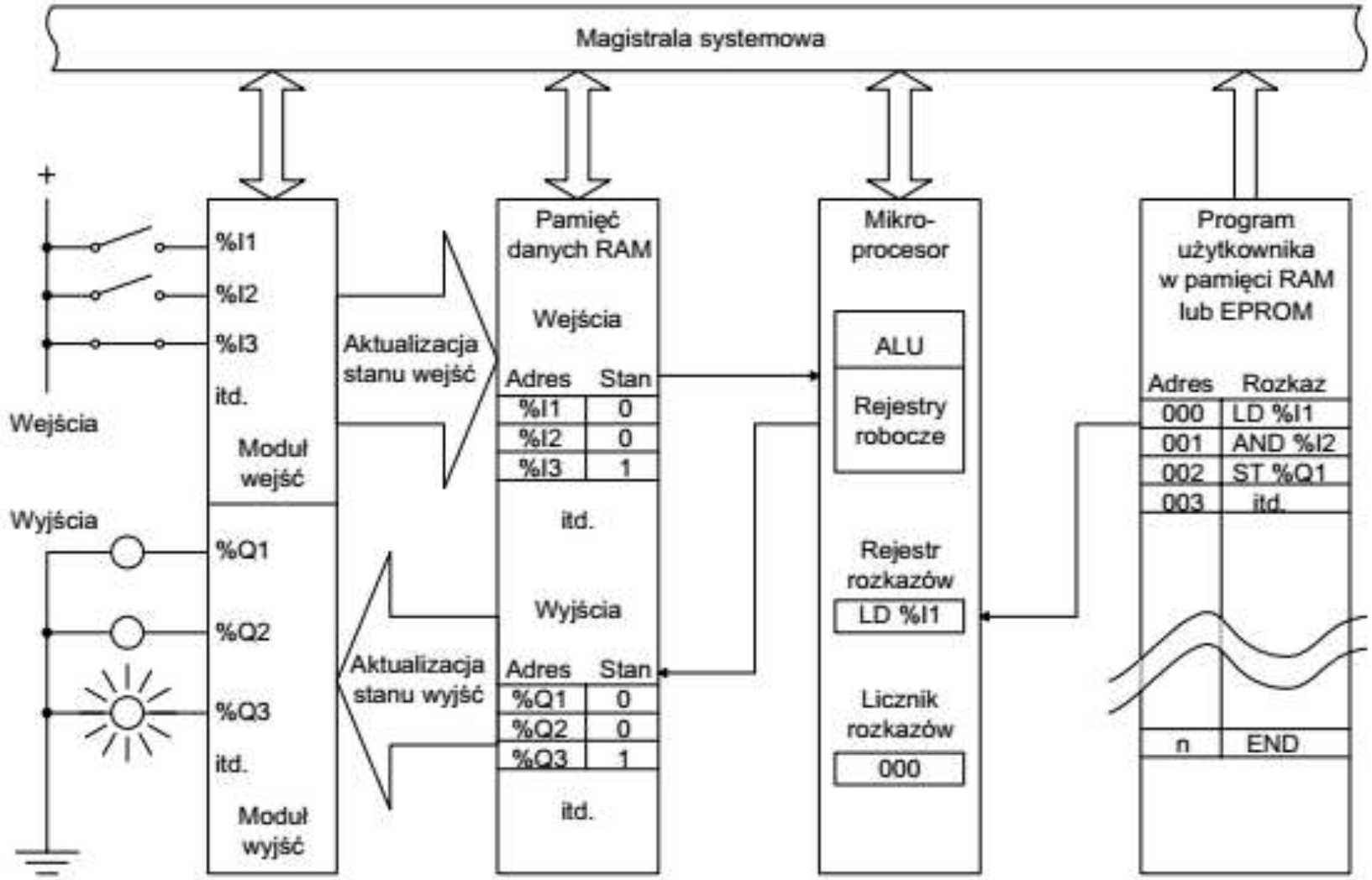
Wykład 6. Budowa i działanie sterowników programowalnych PLC

Systemy *SCADA* (ang. *Supervisory Control and Data Acquisition*) dopełniają i rozszerzają możliwości sterowników, realizując w warstwie sterowania nadrzędnego następujące funkcje:

- zbieranie i przetwarzanie oraz archiwizacja danych pochodzących bezpośrednio z systemów sterownikowych;
- opracowywanie raportów dotyczących bieżącego stanu procesu, zużycia materiałów oraz stanu pracy maszyn i urządzeń;
- wizualizacja wartości zmiennych procesowych (aktualnych i historycznych) w różnych formach graficznych;
- generowanie sygnałów alarmowych związanych z przekroczeniem wartości granicznych;
- wypracowywanie danych dla warstw sterowania operatywnego produkcją oraz warstwy zarządzania.

W celu podwyższenia *niezawodności* systemów sterownikowych wielu producentów wprowadziło sprzętowe i programowe rozwiązania *redundancyjne*.

1.3 Budowa i zasada działania sterownika



Rys. 1.1 Schemat ideowy sterownika

Wykład 6. Budowa i działanie sterowników programowalnych PLC

Sterownik pracuje w *cyklu programowym* (ang. *Program Sweep*), w którym:

- W fazie aktualizacji stanu wejść występuje przepisanie wartości wejść z modułów wejściowych do odpowiadających im obszarów w pamięci danych sterownika (oznaczonych tu jako *%In*, gdzie *n* jest numerem wejścia);
- W fazie wykonania programu realizowany jest jeden przebieg programu użytkownika – kolejne instrukcje programu przekazywane są z pamięci programu do mikroprocesora, który je dekoduje, wykonuje odpowiednie działania i zapisuje wynik obliczeń w pamięci danych. Program użytkownika kończy się instrukcją *END*;
- W fazie aktualizacji wyjść następuje przepisanie obliczonych wartości wyjść (oznaczonych tu jako *%Qn*, gdzie *n* jest numerem wyjścia) z odpowiedniego obszaru danych do modułów wyjściowych, które generują sygnały sterujące.

Pamięć przeznaczona na dane użytkownika dzieli się na obszary:

- dane wejściowe, zawierające obraz wejść sterownika, zwykle adresowane są za pomocą przedrostka *%I*;
- dane wyjściowe, zawierające obraz wyjść sterownika, zwykle adresowane są za pomocą przedrostka *%Q*;
- dane użytkownika, nie związane ani z wejściami, ani z wyjściami, zwykle adresowane za pomocą przedrostka *%M*.

Ze względu na architekturę wyróżnia się sterowniki:

- kompaktowe (małe, przekaźniki inteligentne),
- modułowe (średnie i duże).

Tab. 1.3 Rodziny sterowników największych producentów

Producent	Inteligentne przekaźniki	Małe	Średnie	Duże
Siemens	Logo	SIMATIC S7-200	SIMATIC S7-300	SIMATIC S7-400
Schneider Electric	Zelio	Nano, Micro, Twido	Premium, Compact, Momentum	Quantum
GE Fanuc	VersaMax-Nano	VersaMax-Micro	90-30, VersaMax, PACSystems RX3i	90-70, PACSystems RX7i
Mitsubishi Electric	ALPHA	MELSEC FX1, FX2	MELSEC QnAS	MELSEC QnA, MELSEC System Q
Omron		CPM1, CPM2, CQM1H	C200H-alpha, CJ1, CS1	CVM1
Rockwell Automation (Allen-Bradley)	Pico	MicroLogix	SLC500, FlexLogix, ControlLogix	PLC-5

1.8 Sterowniki modułowe

Rys. 1.8. Przykład sterownika modułowego

Elementy sterownika modułowego:

- Płyta łączeniowa (ang. *baseplate*), zwana także kasetą (ang. *rack*), posiada gniazda (ang. *slots*) do podłączenia wybranych modułów,
- Moduły podstawowe: *zasilacz* (*PS*, ang. *Power Supply*) oraz moduł *jednostki centralnej* (*CPU*, ang. *Central Proccessing Unit*).
- Moduły wejść i wyjść cyfrowych (ang. *Digital Input, Digital Output*),
- Moduły wejść i wyjść analogowych (ang. *Analog Input, Analog Output*),
- Moduły komunikacyjne, do podłączenia sterownika do sieci lokalnej w określonym standardzie, np. *Modbus, Profibus, ControlNet, Genius* itp. lub do sieci *Ethernet*,

Inne moduły (moduły inteligentne i dodatkowe):

- Moduły szybkich liczników (*HSC*, ang. *High-Speed Counter*),
- Moduły pozycjonowania osi (*APM*, ang. *Axis Positioning Module*),
- Moduły wejściowe dla czujników temperatury,
- Moduły regulatora PID lub regulatory rozmyte,
- Moduły akwizycji kodu paskowego itd.

Sterownik 90-70

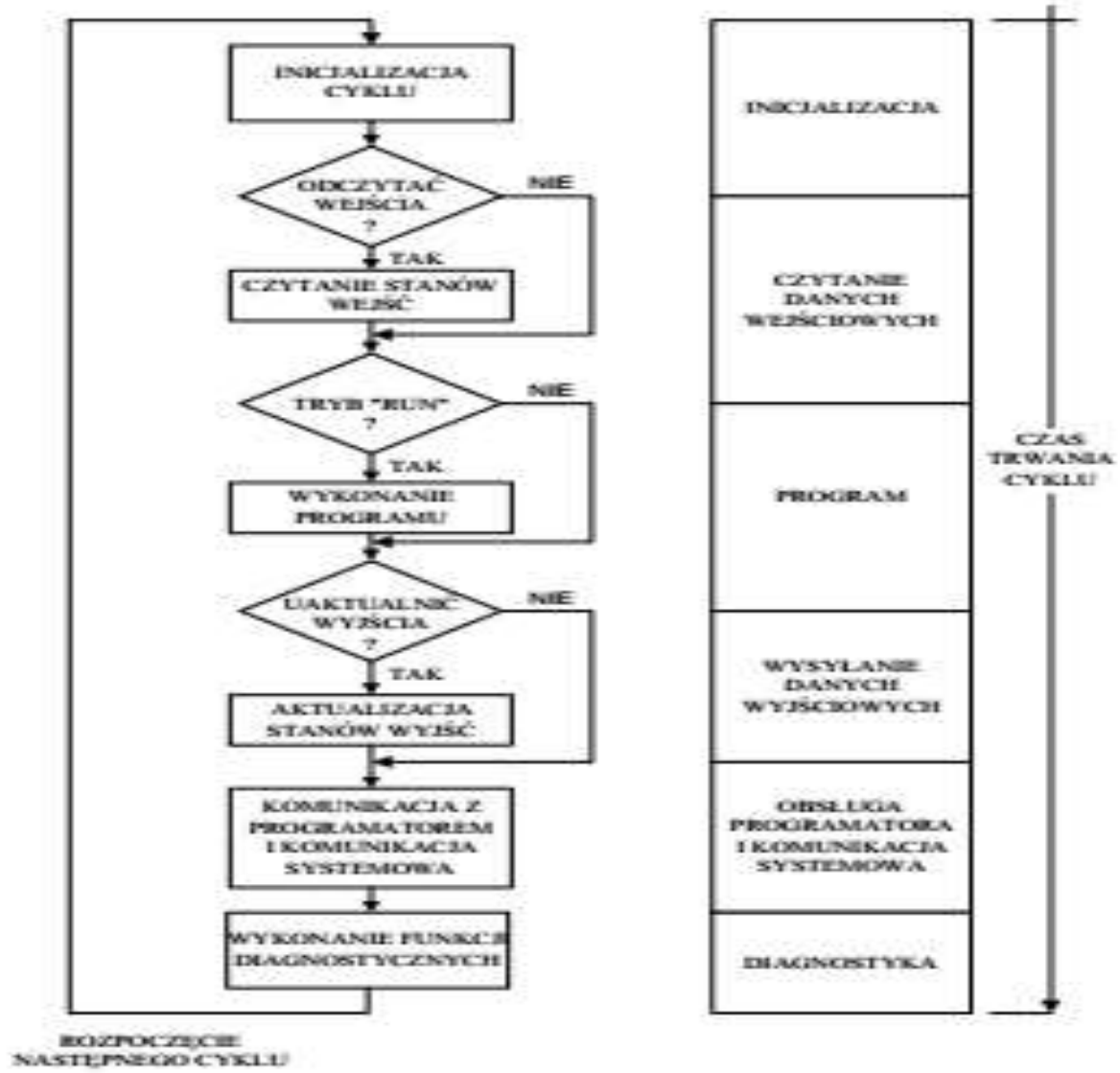
GE Fanuc - Durus

Sterowniki Siemens – S7-200

S7-200 jest sterownikiem dedykowanym automatyzacji maszyn i urządzeń oraz przeznaczonym do tworzenia zdecentralizowanych struktur sterowania dla małych obiektów typu przepompownie, oczyszczalnie ścieków. Sterownik ma budowę modułową, dzięki czemu może być łatwo dopasowany do wymagań użytkownika.

Sterownik LOGO! I S7-300

1.7 Cykl programowy sterownika i tryby pracy



Rys. 1.9 Fazy cyklu programowego sterownika PLC

Wykład 6. Budowa i działanie sterowników programowalnych PLC

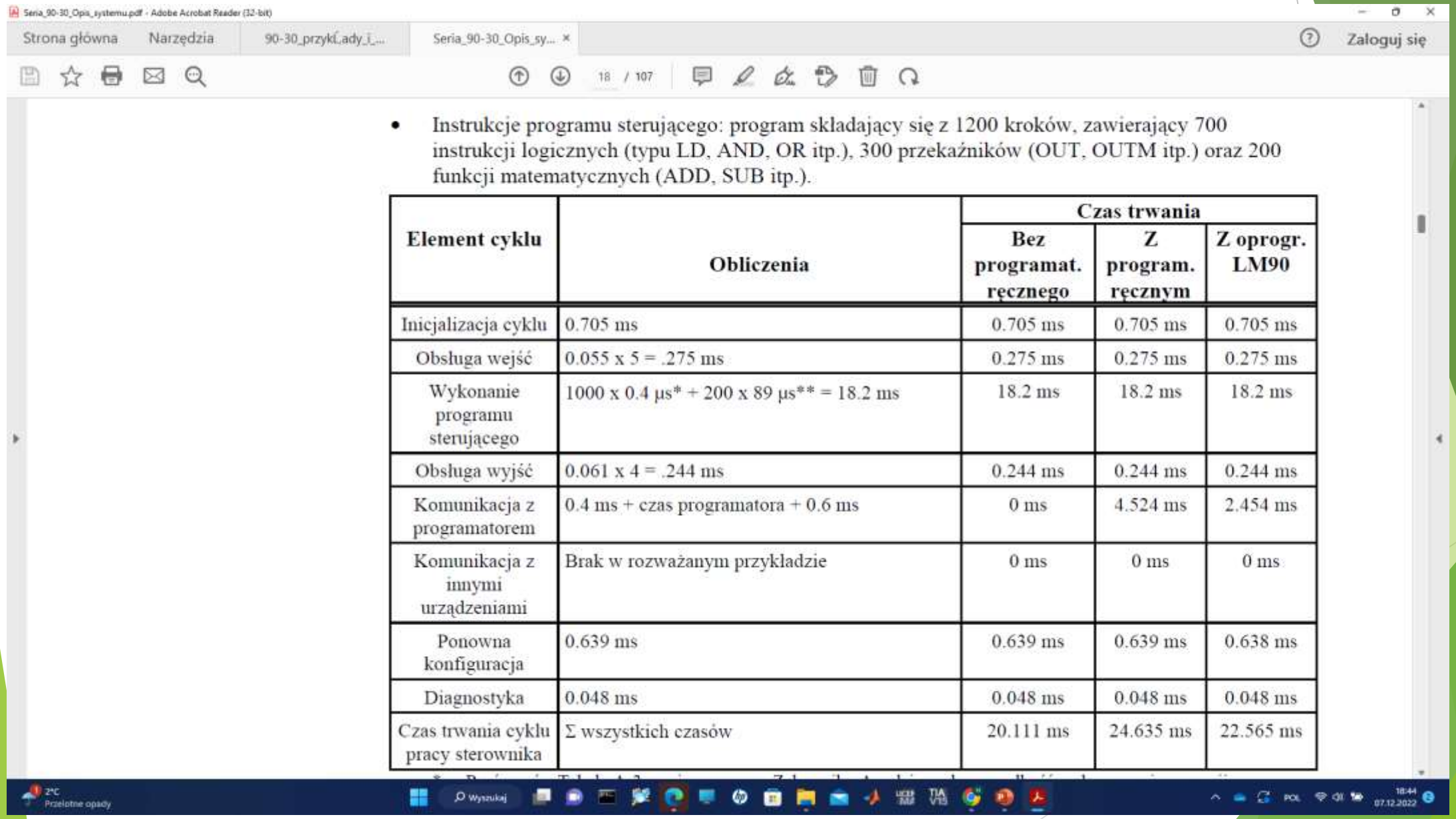
Podstawowe tryby pracy sterownika:

- Tryb *RUN* (tryb *wykonywania*) – jest to właściwy tryb pracy sterownika, w którym wykonywane są wszystkie fazy cyklu;
- Tryb *STOP* – tryb zatrzymania sterownika.

Ponadto: praca w pojedynczym cyklu (ang. *Single Sweep*), praca krokowa (wykonanie tylko jednego rozkazu) lub praca z normalnym cyklem, ale bez aktywizacji sygnałów wyjściowych w modułach wyjść cyfrowych (tryby testowania).

Czas trwania cyklu sterownika (ang. *Scan Time*) nie może być dowolnie długi. W CPU istnieje układ zegara (*watchdog*), zabezpieczającego przed zawieszeniem sterownika.

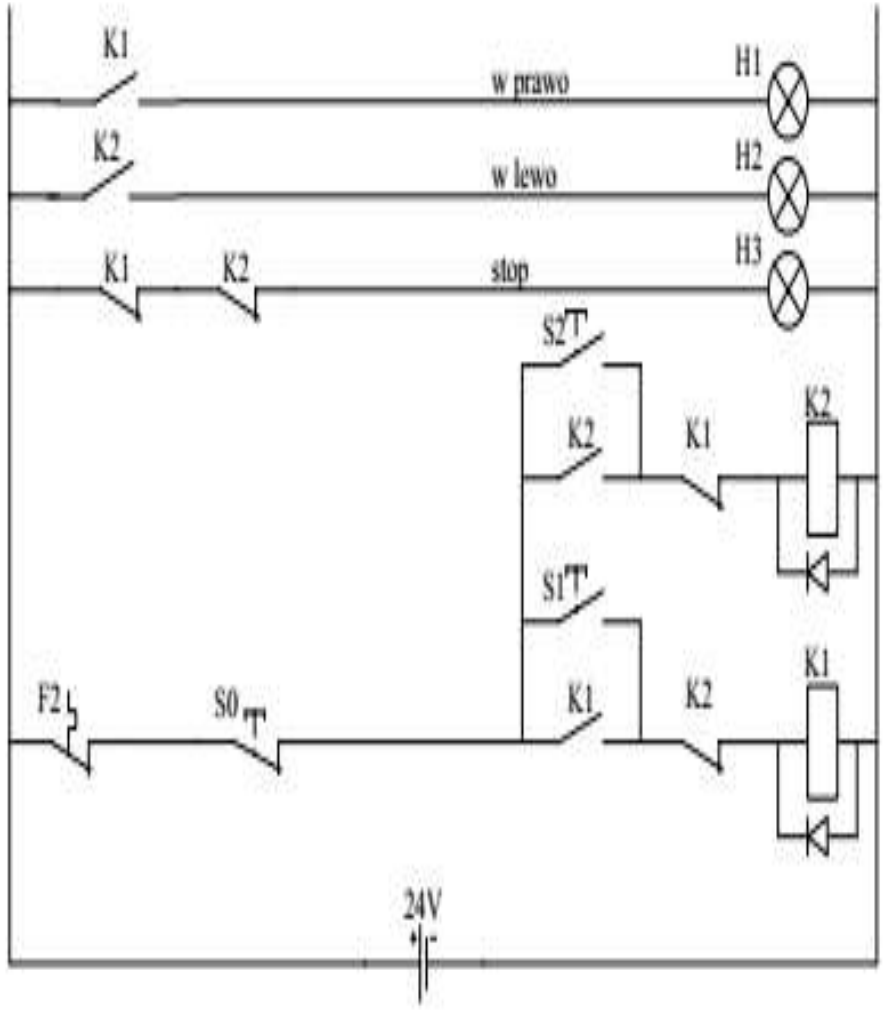
Wykład 6. Budowa i działanie sterowników programowalnych PLC



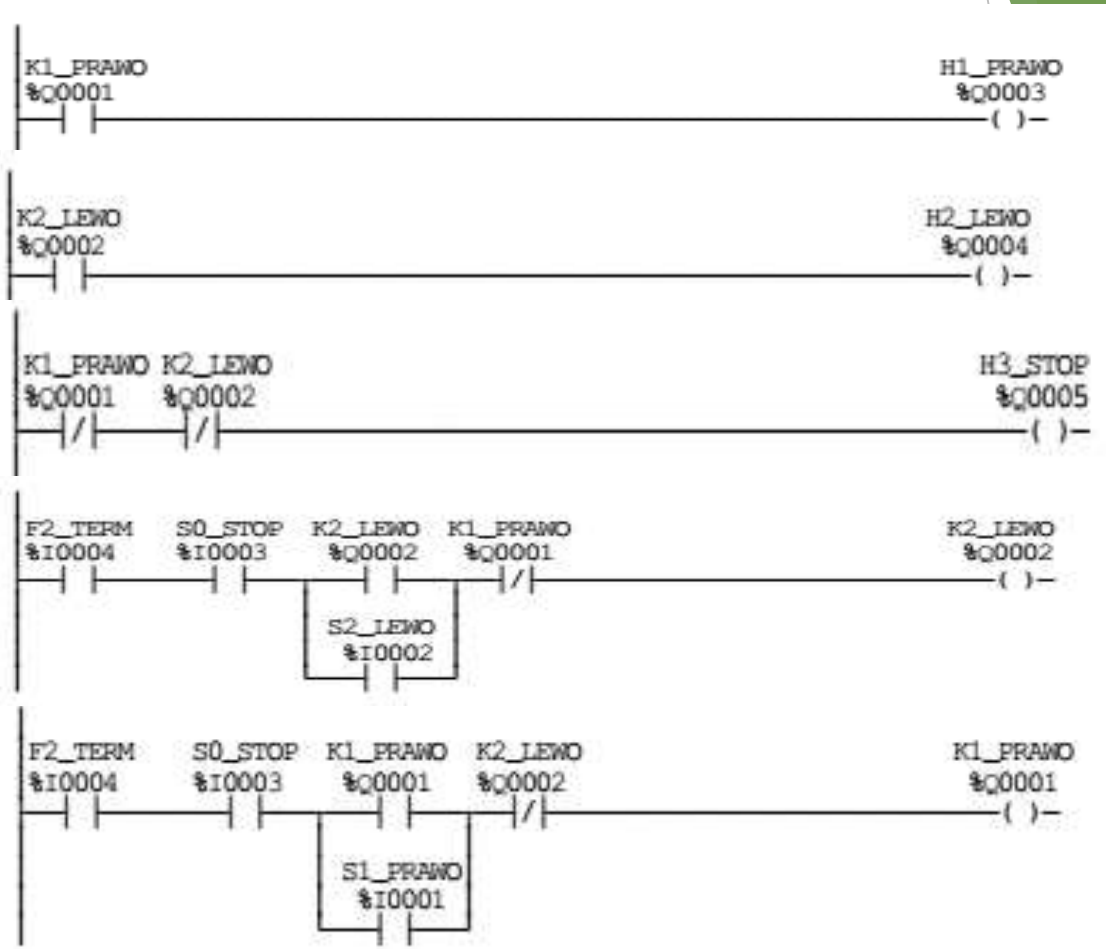
- Instrukcje programu sterującego: program składający się z 1200 kroków, zawierający 700 instrukcji logicznych (typu LD, AND, OR itp.), 300 przekaźników (OUT, OUTM itp.) oraz 200 funkcji matematycznych (ADD, SUB itp.).

Element cyklu	Obliczenia	Czas trwania		
		Bez programat. ręcznego	Z program. ręcznym	Z oprogr. LM90
Inicjalizacja cyklu	0.705 ms	0.705 ms	0.705 ms	0.705 ms
Obsługa wejść	0.055 x 5 = .275 ms	0.275 ms	0.275 ms	0.275 ms
Wykonanie programu sterującego	1000 x 0.4 µs* + 200 x 89 µs** = 18.2 ms	18.2 ms	18.2 ms	18.2 ms
Obsługa wyjść	0.061 x 4 = .244 ms	0.244 ms	0.244 ms	0.244 ms
Komunikacja z programatorem	0.4 ms + czas programatora + 0.6 ms	0 ms	4.524 ms	2.454 ms
Komunikacja z innymi urządzeniami	Brak w rozważanym przykładzie	0 ms	0 ms	0 ms
Ponowna konfiguracja	0.639 ms	0.639 ms	0.639 ms	0.638 ms
Diagnostyka	0.048 ms	0.048 ms	0.048 ms	0.048 ms
Czas trwania cyklu pracy sterownika	Σ wszystkich czasów	20.111 ms	24.635 ms	22.565 ms

Wykład 7. Programowanie sterowników programowalnych PLC



Rys. 1.3 Schemat stykowy obwodu sterowania

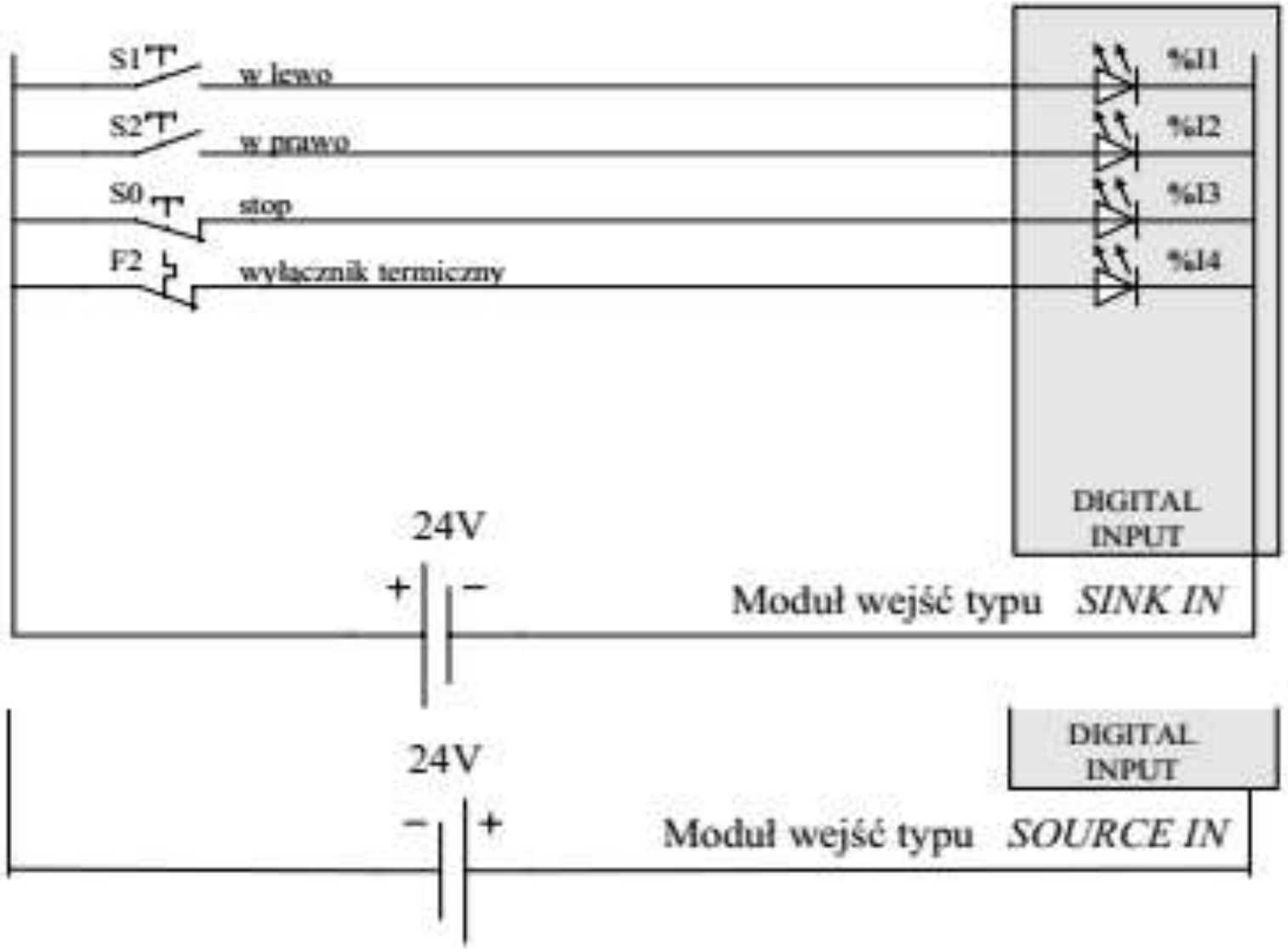


Wykład 7. Programowanie sterowników programowalnych PLC

Tab. 1.1 Deklaracje zmiennych dla programu sterowania silnikiem

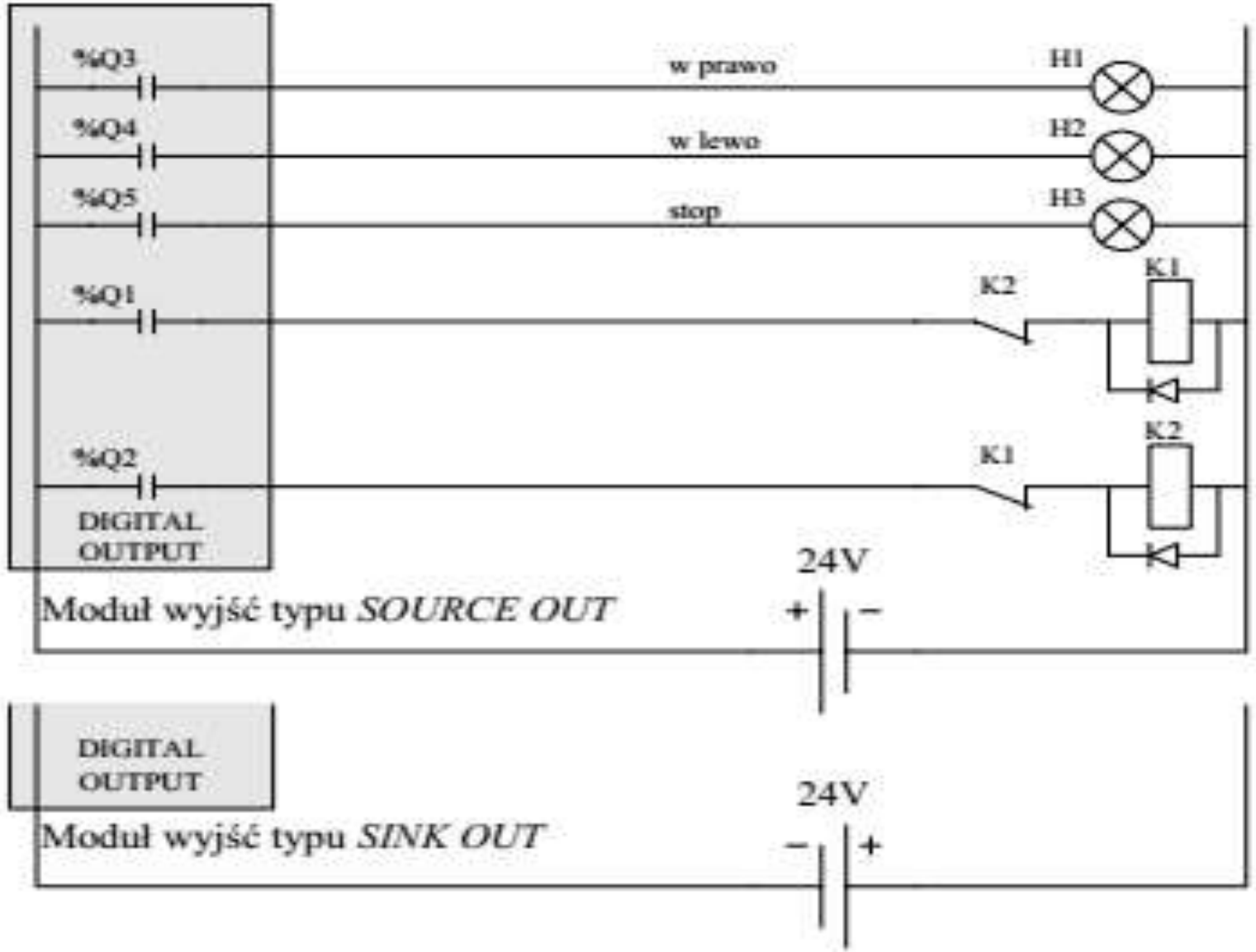
Adres w pamięci	Nazwa zmiennej	Symbol na schemacie	Opis
%I0001	*S1_PRAWO*	*S1*	przycisk załączenia w prawo (zwierny)
%I0002	*S2_LEWO*	*S2*	przycisk załączenia w lewo (zwierny)
%I0003	*S0_STOP*	*S0*	przycisk wyłączenia stop (rozwierny)
%I0004	*F2_TERM*	*F2*	zabezpieczenie termiczne (rozwierny)
%Q0001	*K1_PRAWO*	*K1*	przekaźnik załączający silnik w prawo
%Q0002	*K2_LEWO*	*K2*	przekaźnik załączający silnik w lewo
%Q0003	*H1_PRAWO*	*H1*	lampka sygnalizacji wirowania w prawo
%Q0004	*H2_LEWO*	*H2*	lampka sygnalizacji wirowania w lewo
%Q0005	*H3_STOP*	*H3*	lampka sygnalizacji stopu

Wykład 7. Programowanie sterowników programowalnych PLC

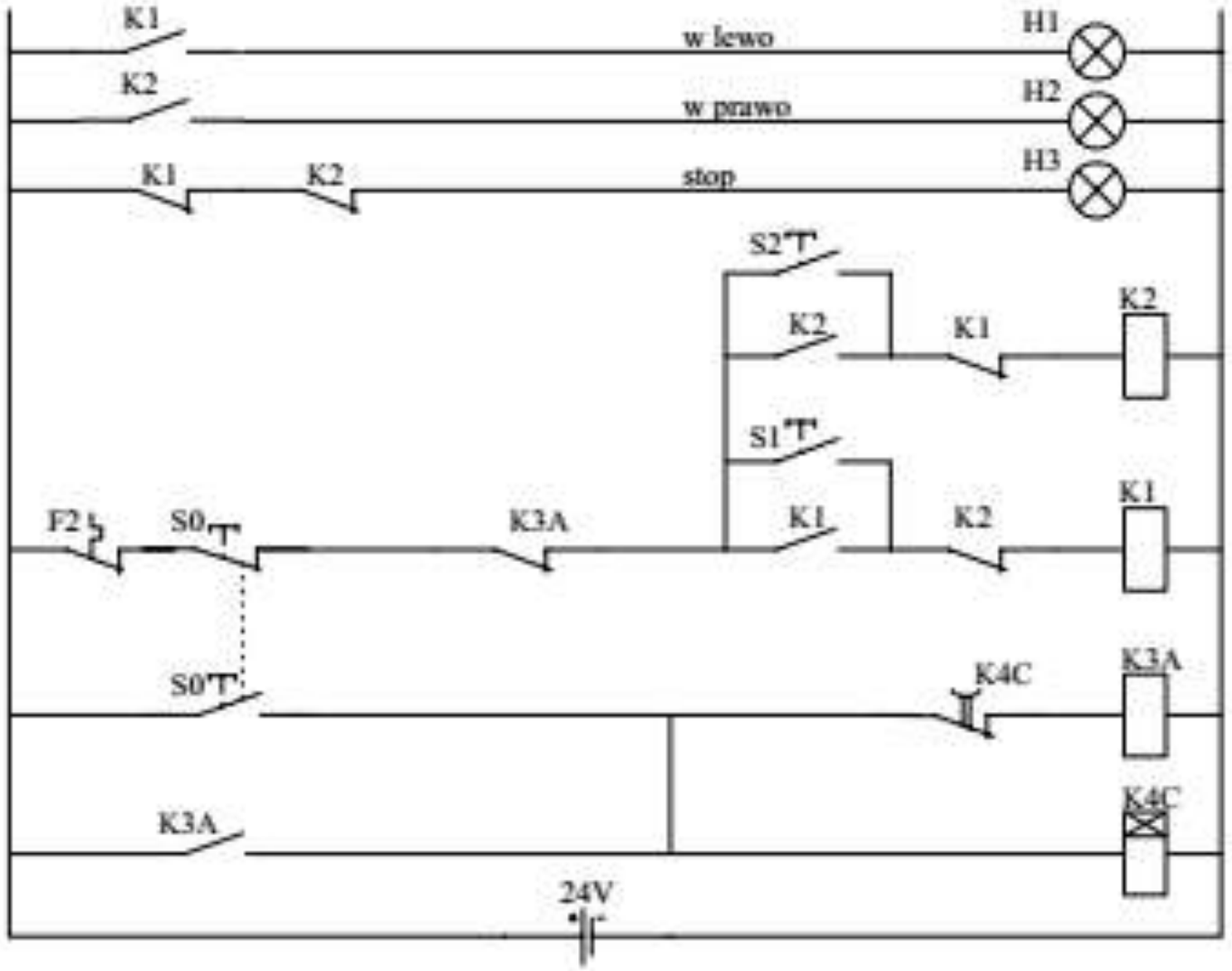


Rys. 1.4 Schemat połączeń obwodów wejściowych z modułem wejść cyfrowych

Wykład 7. Programowanie sterowników programowalnych PLC



Rys. 1.5 Schemat połączeń obwodów wyjściowych z modułem wyjść cyfrowych

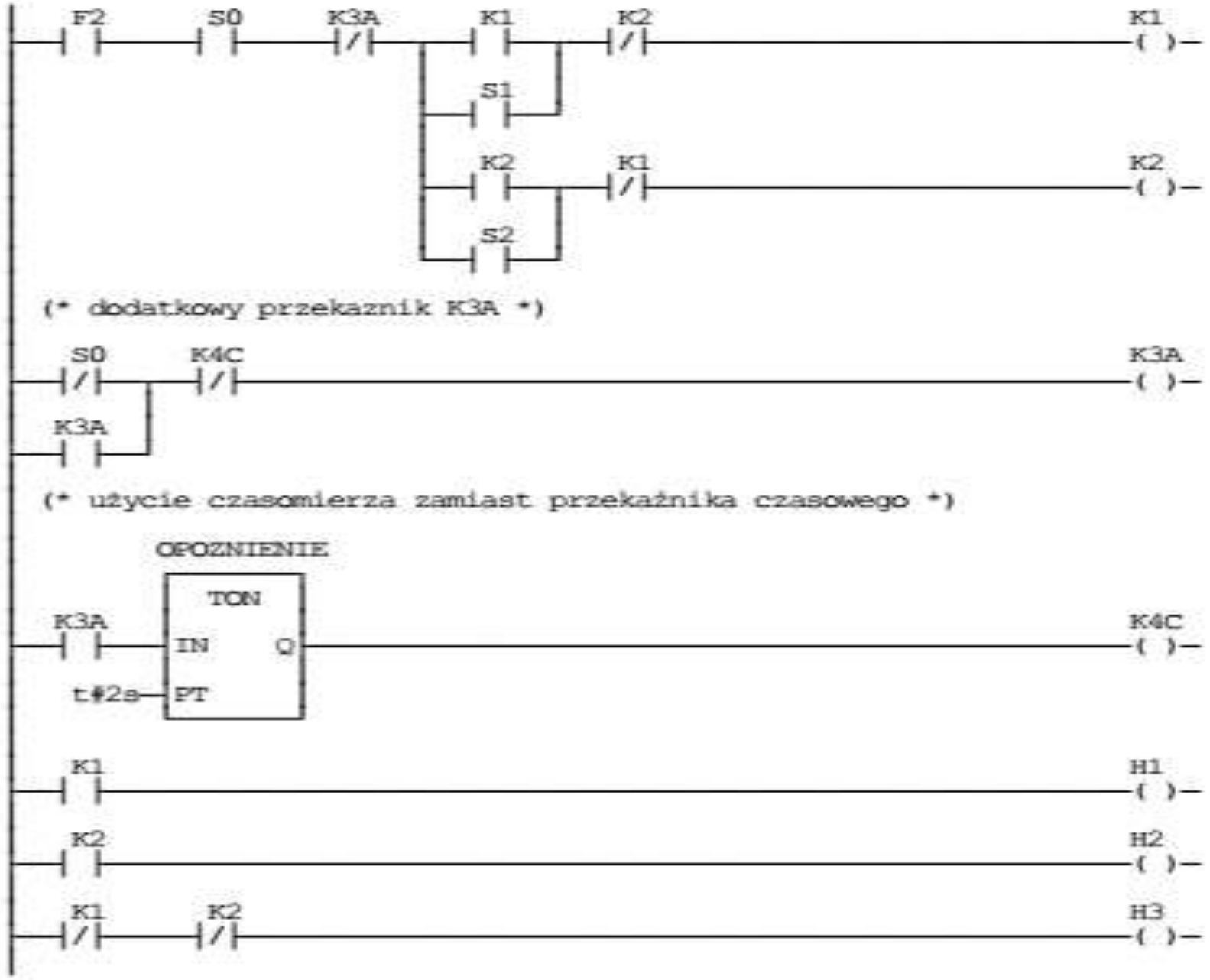


Rys. 1.6 Schemat stykowy obwodu sterowania silnikiem z opóźnionym przełączaniem

Tab. 1.2 Deklaracje zmiennych dla programu sterowania z opóźnionym przełączaniem

Adres w pamięci	Nazwa zmiennej	Opis
%I0001	*S1*	przycisk załączenia w prawo (zwierny)
%I0002	*S2*	przycisk załączenia w lewo (zwierny)
%I0003	*S0*	przycisk wyłączenia stop (rozwierny)
%I0004	*F2*	zabezpieczenie termiczne (rozwierny)
%Q0001	*K1*	przekaźnik załączający silnik w prawo
%Q0002	*K2*	przekaźnik załączający silnik w lewo
%Q0003	*H1*	lampka sygnalizacji wirowania w prawo
%Q0004	*H2*	lampka sygnalizacji wirowania w lewo
%Q0005	*H3*	lampka sygnalizacji stopu
%M0001	*K3A*	zmienna pomocnicza w pamięci sterownika
%M0002	*K4C*	zmienna pomocnicza w pamięci sterownika

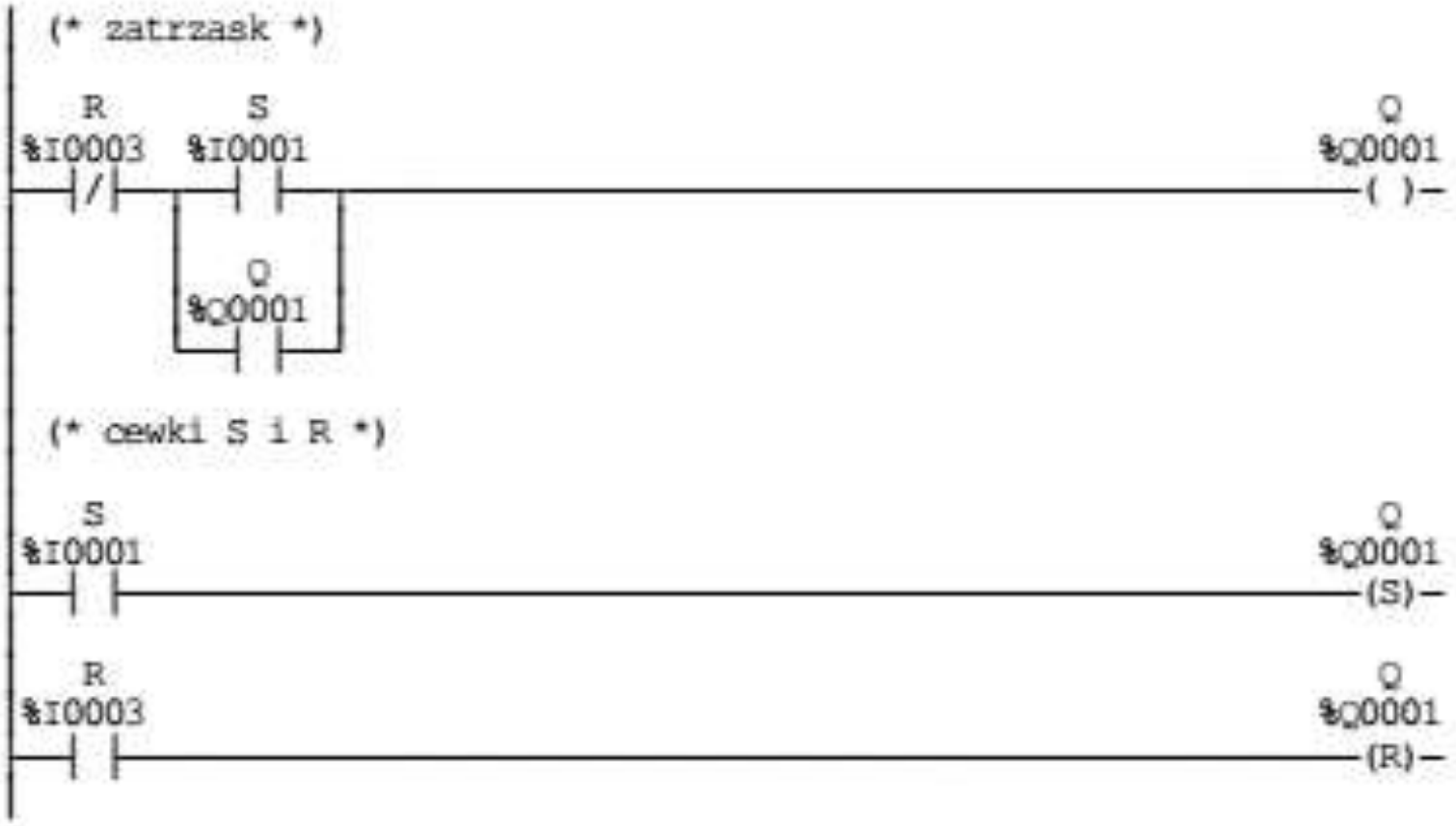
Program sterowania silnikiem z opóźnionym przełączaniem



Wykład 7. Programowanie sterowników programowalnych PLC

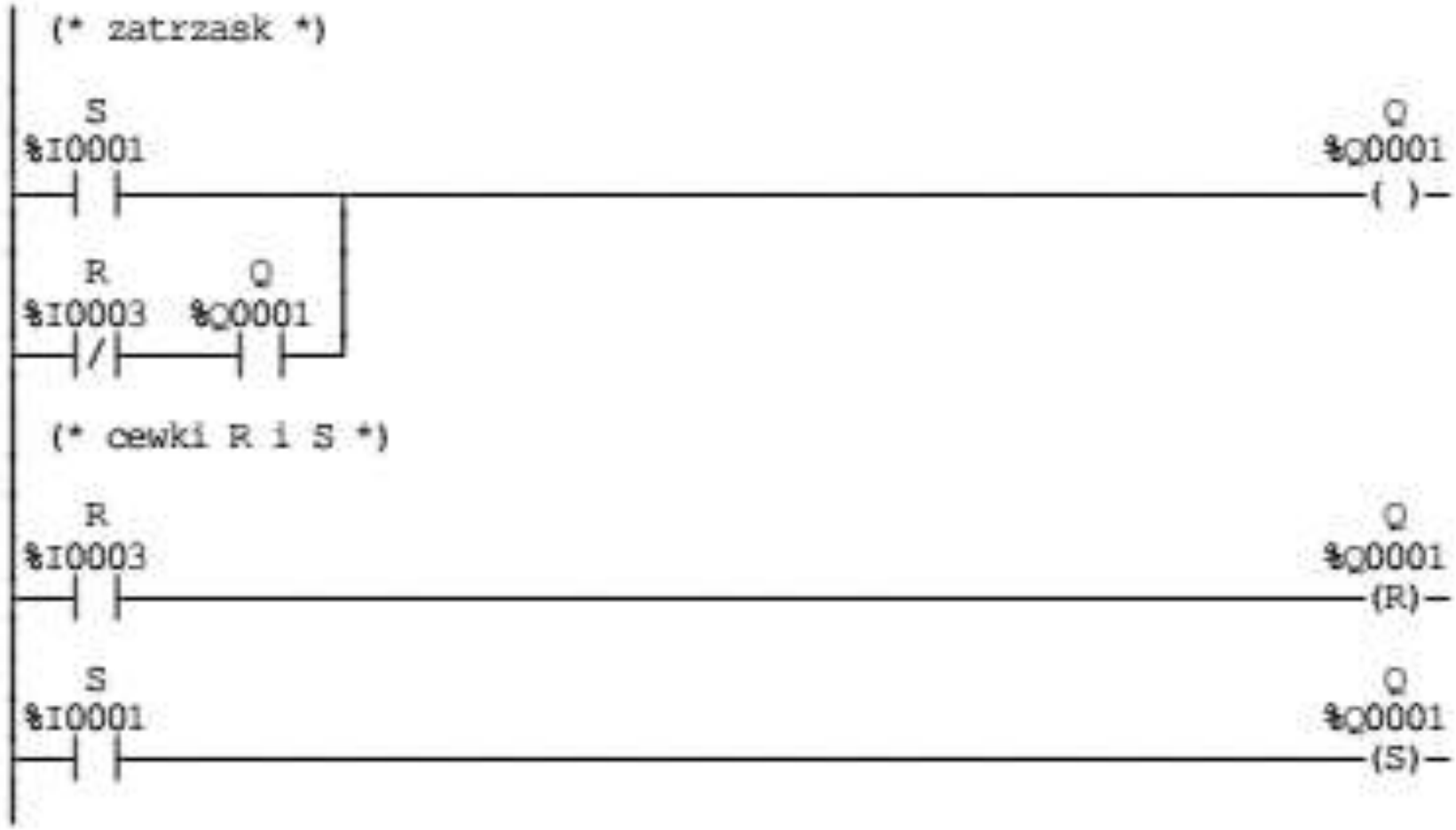
1.5 Programowanie przerzutników

Dwa sposoby programowania przerzutnika RS w języku schematów drabinkowych



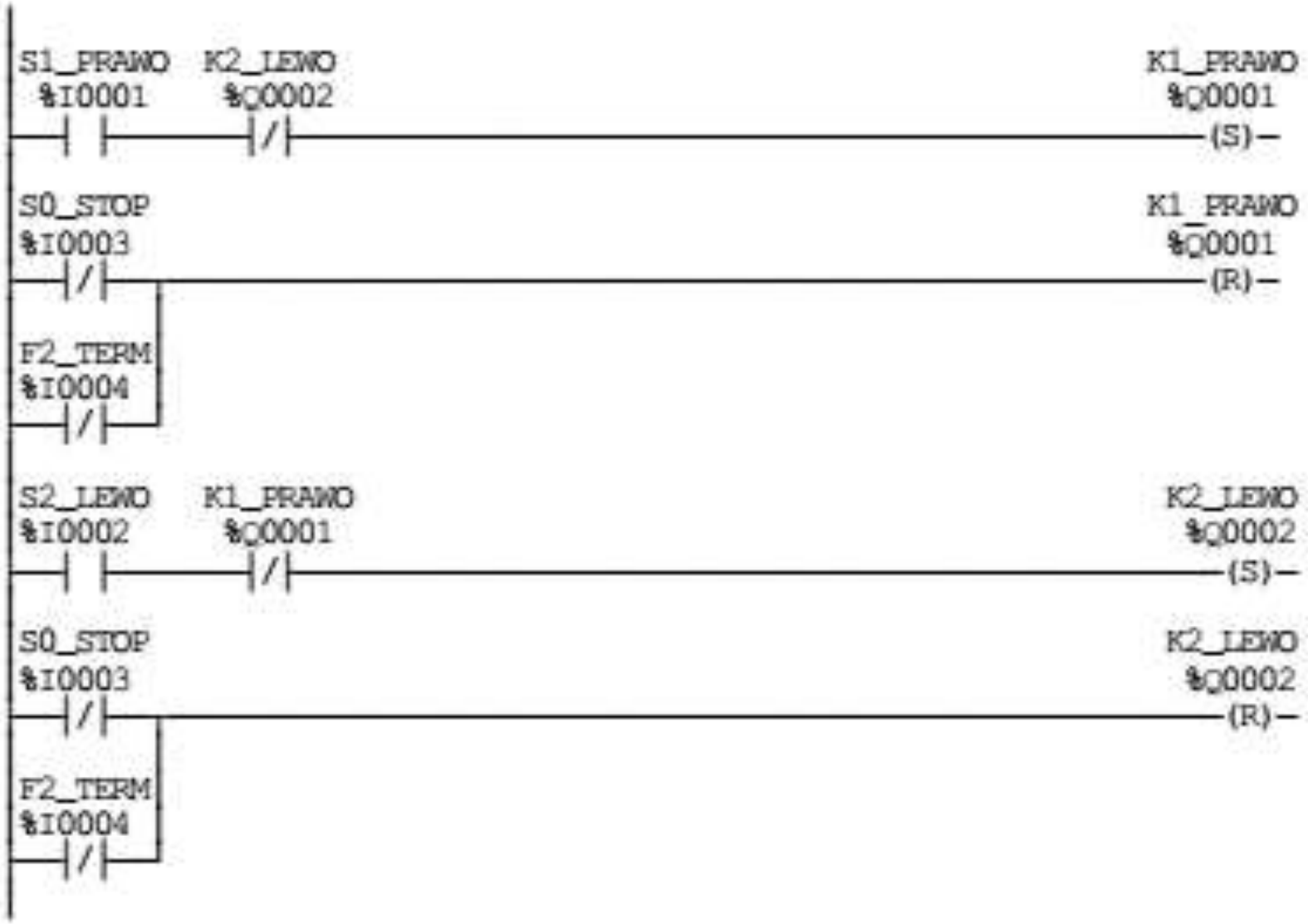
Wykład 7. Programowanie sterowników programowalnych PLC

Dwa sposoby programowania przerzutnika SR w języku schematów drabinkowych

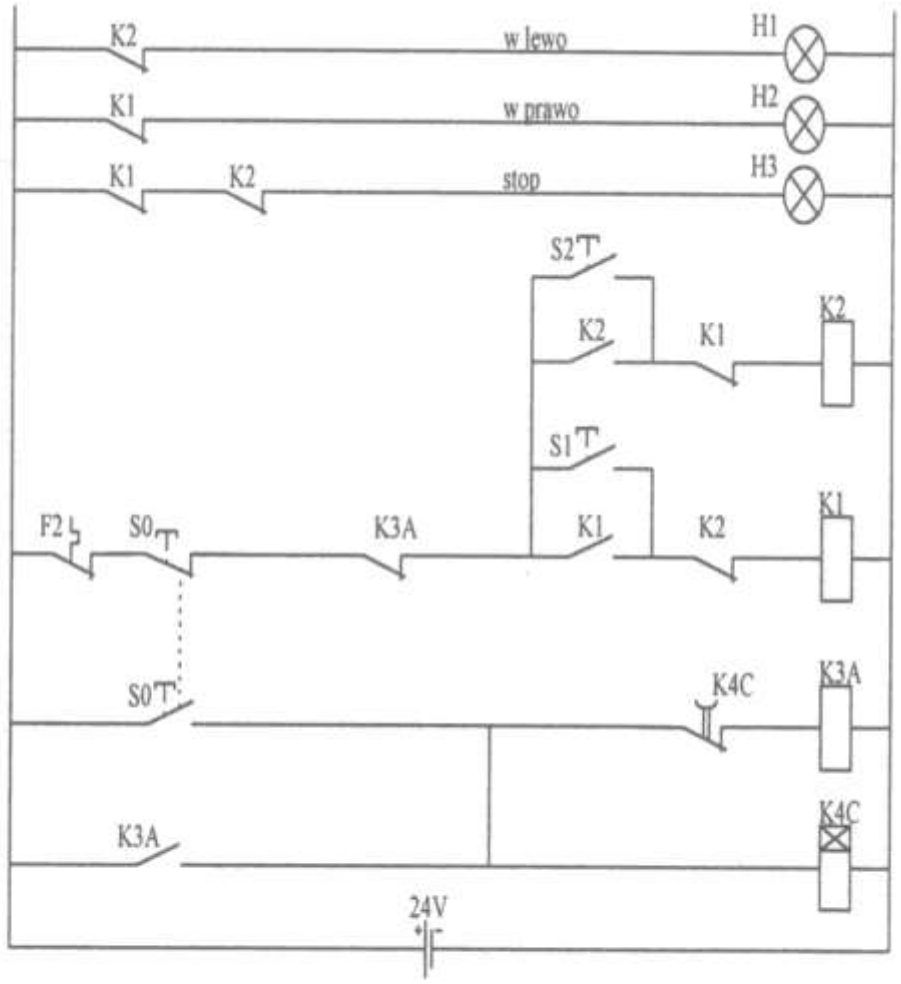


Wykład 7. Programowanie sterowników programowalnych PLC

Program sterowania silnikiem w języku schematów drabinkowych (wersja 2)



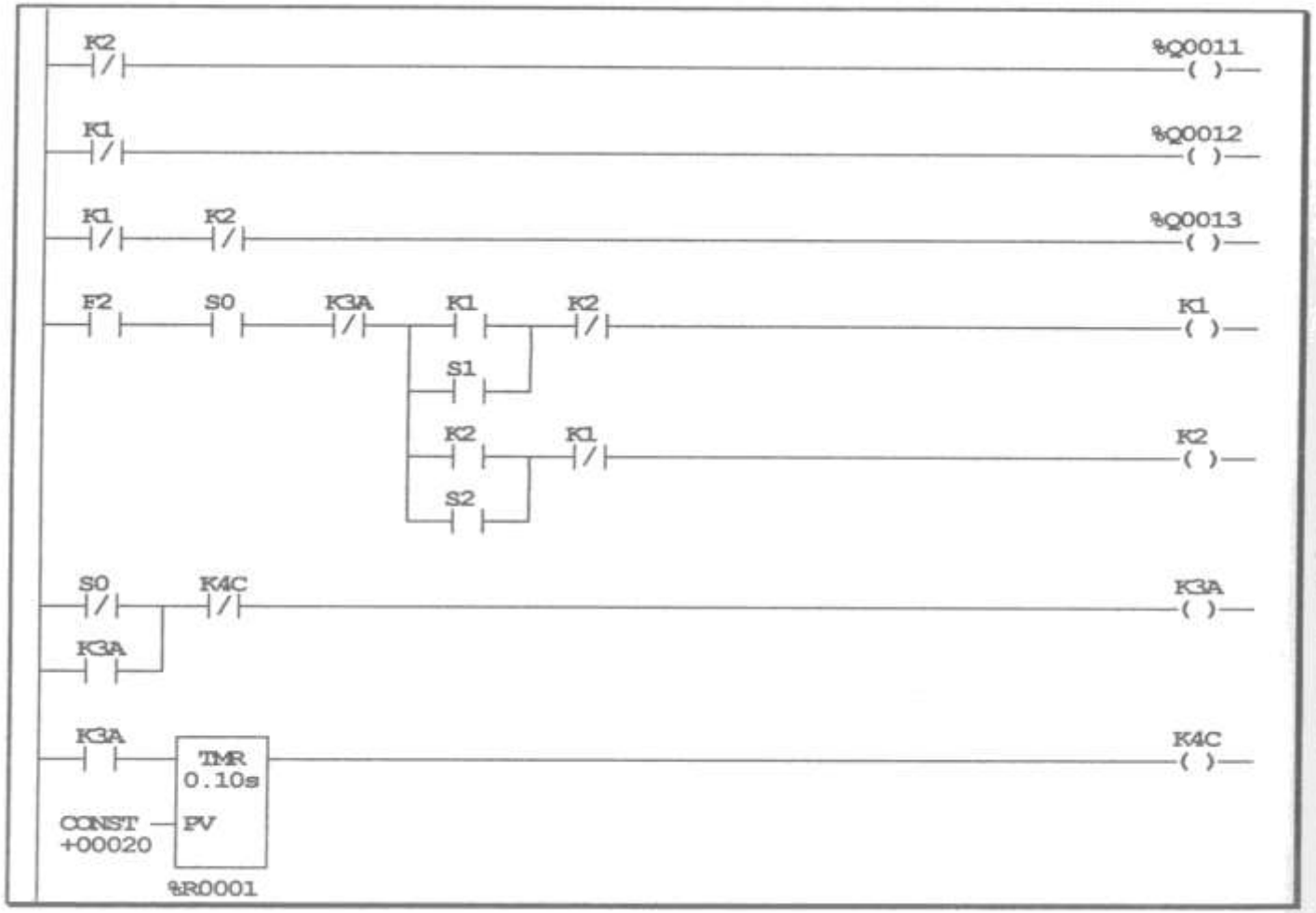
Przykład układów sterowania silnikiem



Rys. 1.10 Schemat stykowy obwodu sterowania silnikiem z opóźnionym przełączaniem.

%I0001	S0	przycisk stopu
%I0002	S1	przycisk wirowania w lewo
%I0003	S2	przycisk wirowania w prawo
%I0004	F2	styk bezpiecznika termicznego
%Q0001	K1	cewka załączająca wirowania w lewo
%Q0002	K2	cewka załączająca wirowania w prawo
%Q0003	H1	lampka syg. wirowania w lewo
%Q0004	H2	lampka syg. wirowania w prawo
%Q0005	H3	lampka stopu
%M0001	K3A	przekaźnik pomocniczy
%M0002	K4C	przekaźnik czasowy

Tab. 1.4 Deklaracje zmiennych dla silnika z opóźnionym przełączaniem



Języki programowania sterowników

- Z punktu widzenia użytkownika możliwość programowania sterowników PLC jest najbardziej interesującym elementem systemu sterowania realizowanego za ich pomocą, gdyż to właśnie w ten sposób wprowadza się do systemu odpowiedni algorytm sterowania.

- Norma IEC 61131 „Programmable Controllers” składa się z pięciu części, a jej trzecia część dotyczy języków programowania i stanowi jej najważniejszą część.

Języki programowania sterowników

Norma IEC 61131-3 definiuje pojęcia podstawowe, zasady ogólne, model programowy i model komunikacyjny (wymiana danych między elementami oprogramowania) oraz podstawowe typy i struktury danych. Określono w niej dwie grupy języków programowania: języki tekstowe i graficzne.

- W grupie języków tekstowych zdefiniowane zostały następujące języki:
- Język **listy instrukcji IL** (Instruction List)
- Język **strukturalny ST** (Structured Text)
- Do grupy języków graficznych należą:
- Język **schematów drabinkowych LAD** (Ladder Diagram)
- Język **schematów blokowych FBD** (Function Block Diagram)

Ponadto w normie IEC 61131-3 przedstawiono sposób tworzenia struktury wewnętrznej programu w postaci **grafu sekwencji SFC** (Sequential Function Chart), który pozwala na opisywanie zadań sterowania sekwencyjnego za pomocą grafów zawierających etapy (kroki) i warunki przejścia (tranzycji) między tymi etapami.

Język **listy instrukcji IL** (Instruction List)

Język listy instrukcji IL, będący odpowiednikiem języka typu assembler, którego zbiór instrukcji obejmuje operacje logiczne, arytmetyczne, operacje relacji, jak również funkcje przerzutników, czasomierzy, liczników itp..

Język **strukturalny ST** (Structured Text)

Język strukturalny ST, który jest odpowiednikiem języka algorytmicznego wysokiego poziomu, zawierającego struktury programowe takie, jak:

If...then...else...end_if

Case...of...end_case

For...to...do...end_for

While...do...end_while

Repeat...end_repeat

Język **schematów drabinkowych LAD** (Ladder Diagram)

Język schematów drabinkowych LAD (lub LD), podobny do stykowych obwodów przekaźnikowych, w którym dopuszcza się użycie także funkcji: arytmetycznych, logicznych, porównań i relacji jak również bloków funkcyjnych: przerzutników, czasomierzy, liczników, regulatora PID czy bloków programowych.

Język **schematów blokowych FBD** (Function Block Diagram)

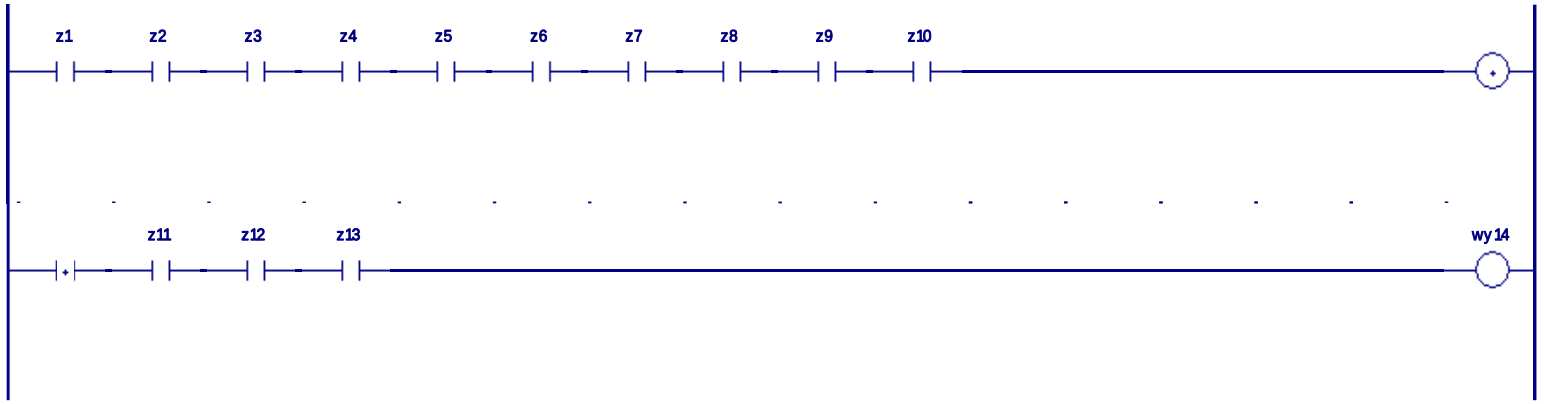
Język schematów blokowych FBD, będący odpowiednikiem schematów przepływu sygnału dla obwodów logicznych przedstawionych w formie połączonych bramek logicznych oraz bloków funkcyjnych takich jak w języku LAD.

Język C

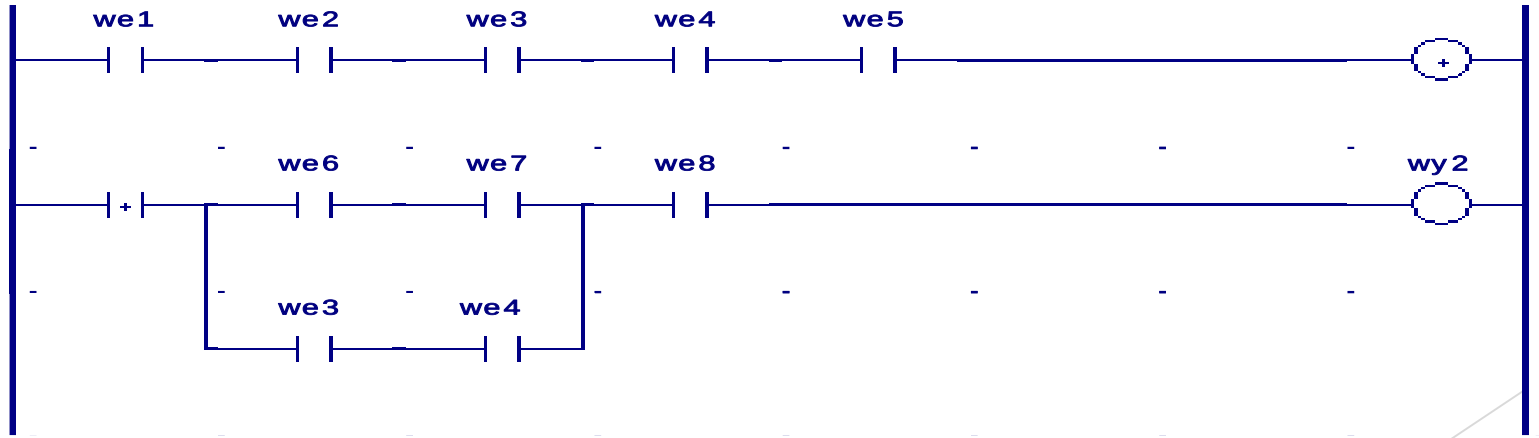
Język C jest od niedawna stosowany do programowania sterowników programowalnych PLC. Taką możliwość daje nam m. in. firma GE Fanuc w systemach 90-70 czy PACSystems RX3i/RX7i.

ZASADY KONSTRUKCJI SZCZEBLA

1. Szczebel może zawierać w jednej linii maksymalnie 29 styków w ostatniej kolumnie szczebla może się znaleźć cewka, skok lub blok funkcyjny.

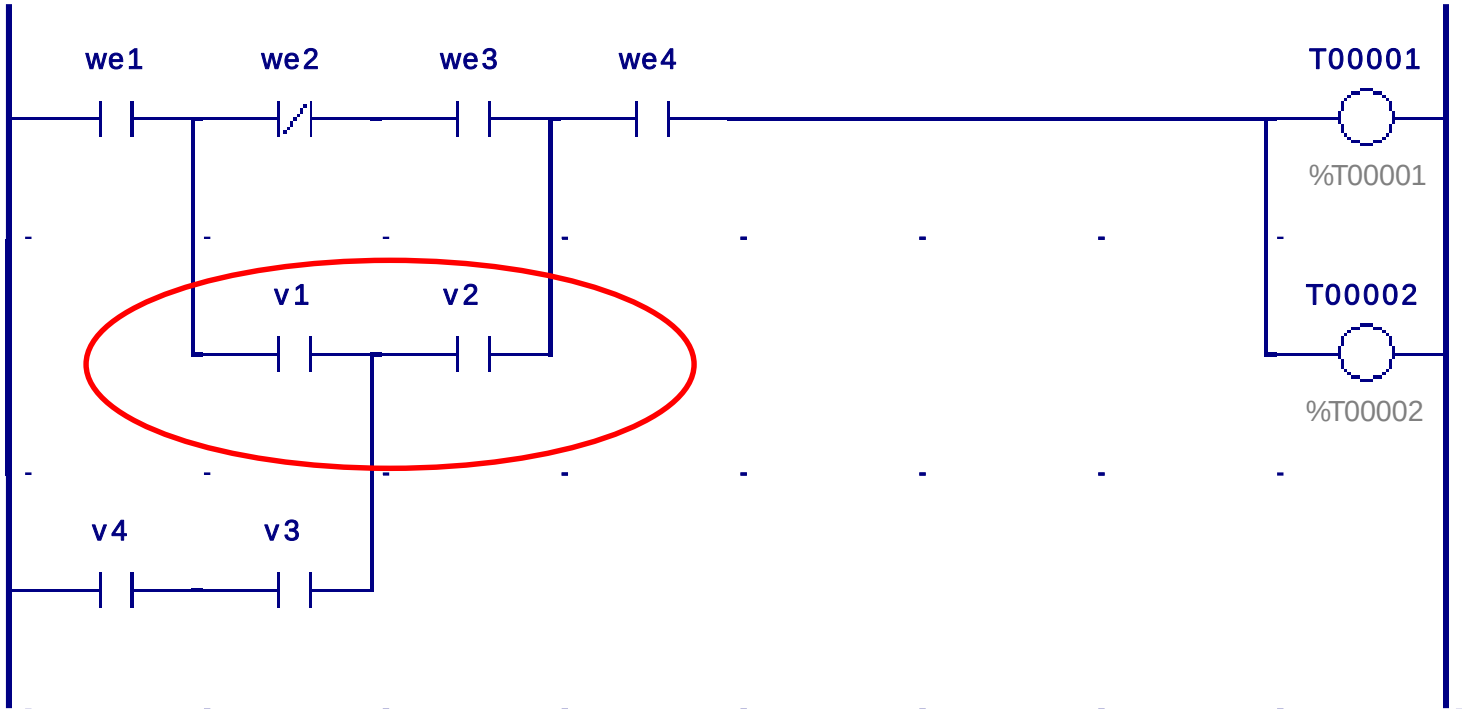


2. Styk kontynuacji musi znajdować się w pierwszej kolumnie szczebla.



ZASADY KONSTRUKCJI SZCZEBLA

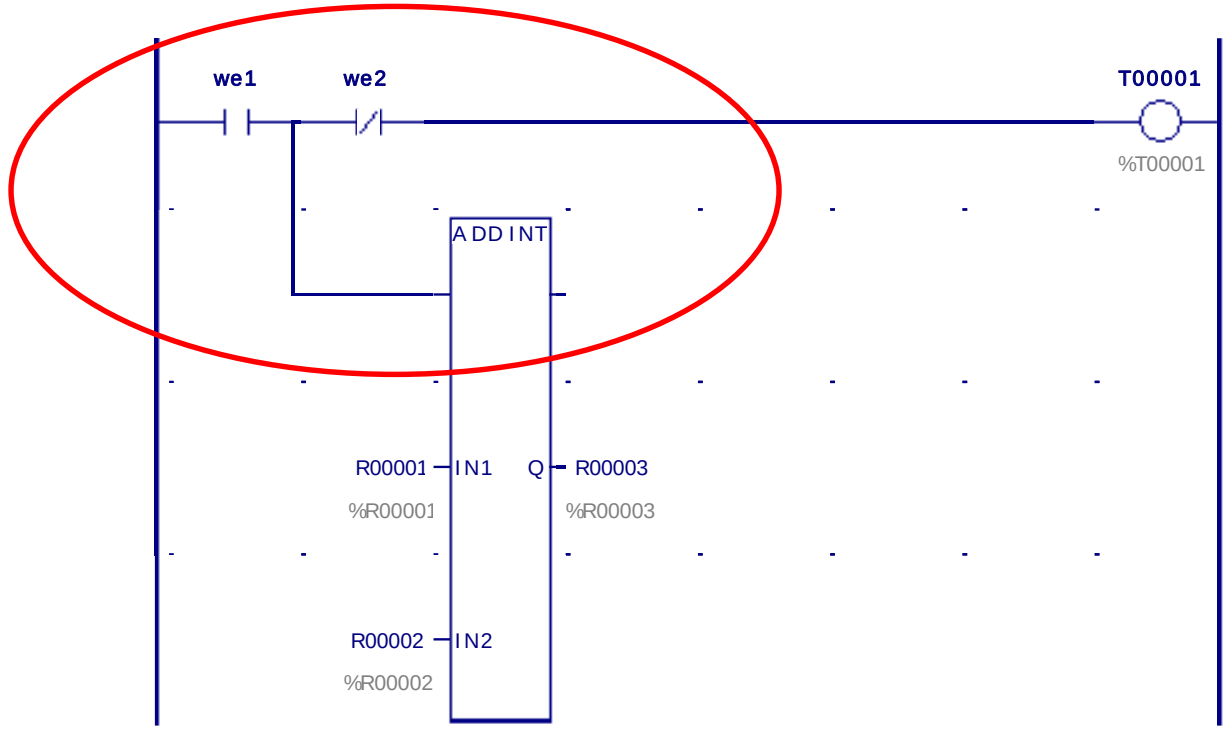
4. Nie może wystąpić rozgałęzienie mające początek lub koniec wewnątrz innego rozgałęzienia.



Uwaga !!! Rozgałęzienia muszą mieć początek i koniec na tym samym poziomie.

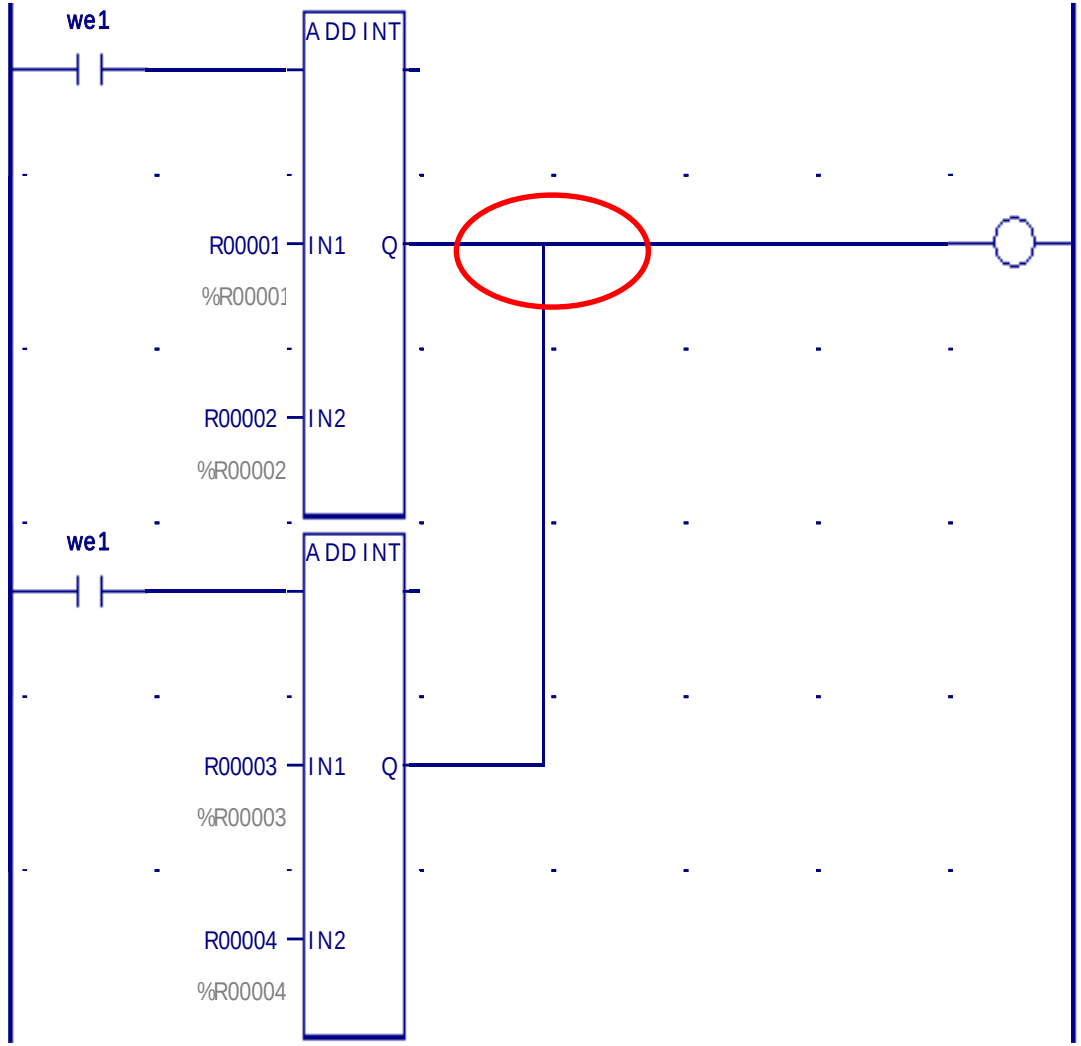
ZASADY KONSTRUKCJI SZCZEBLA

5. Jeżeli w szczeblu występuje blok funkcyjny, to poniżej ani powyżej tego bloku nie mogą występować rozgałęzienia.



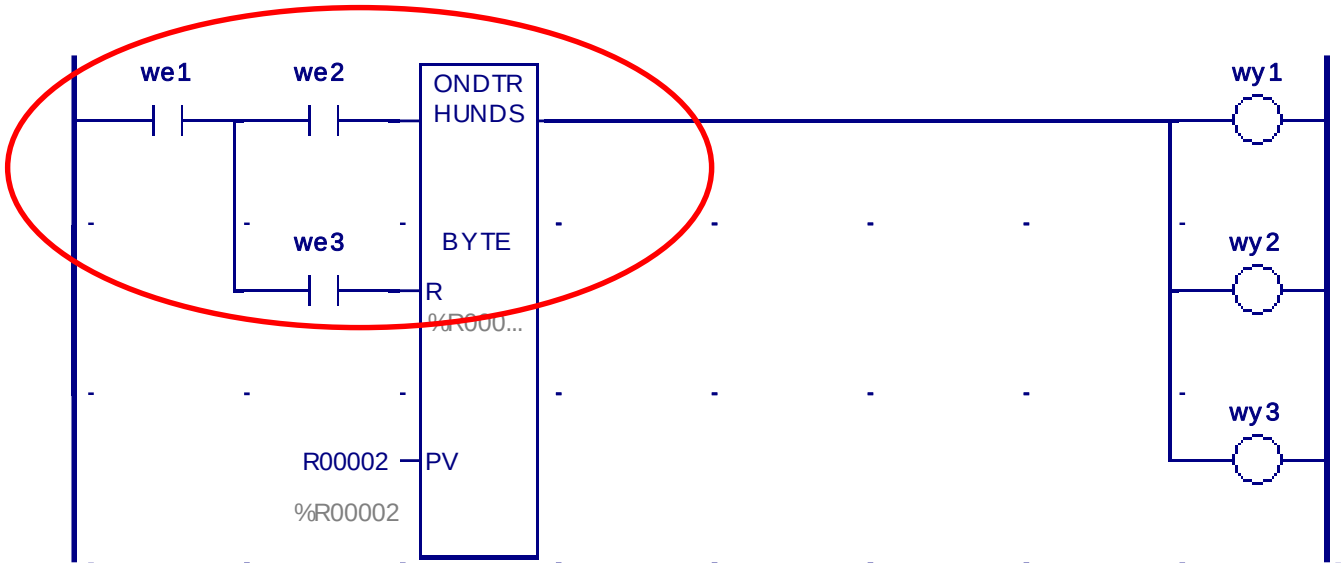
ZASADY KONSTRUKCJI SZCZEBLA

5. Jeżeli w szczeblu występuje blok funkcyjny, to poniżej ani powyżej tego bloku nie mogą występować rozgałęzienia.



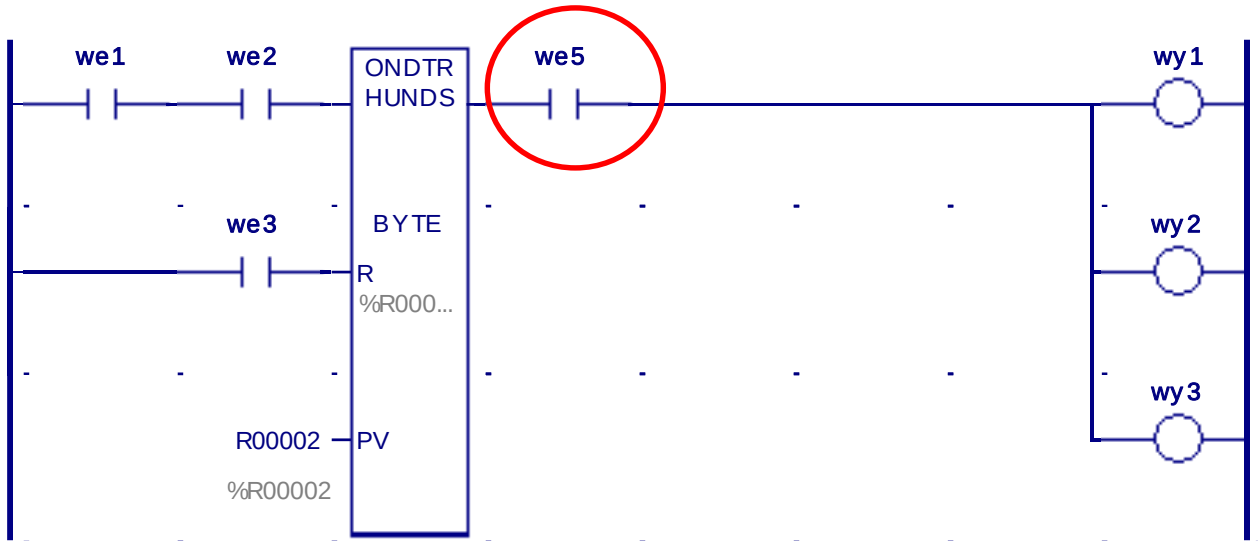
ZASADY KONSTRUKCJI SZCZEBLA

6. Jeżeli w szczeblu występuje blok funkcyjny, to nie może być żadnych rozgałęzień z wyjątkiem prowadzących bezpośrednio do cewek przekaźników.



ZASADY KONSTRUKCJI SZCZEBLA

7. Za blokiem funkcyjnym nie może być żadnych styków.
8. Do styków lub cewek można przypisać jedynie referencje binarne. Do cewki nie można przypisać referencji %I.



Zmienne dyskretne

- %I –reprezentujące fizyczne wejścia dyskretne
- %Q - reprezentujące fizyczne wyjścia dyskretne
- %M – reprezentujące wewnętrzne (pomocnicze) zmienne programu sterującego
- %T – tymczasowe (tracące swój stan po zaniku zasilania lub zatrzymaniu/uruchomieniu sterownika) zmienne pomocnicze
- %S - zmienne systemowe informacyjne (tylko do odczytu)
- %G – zmienne globalne

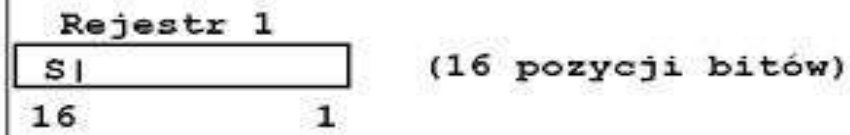
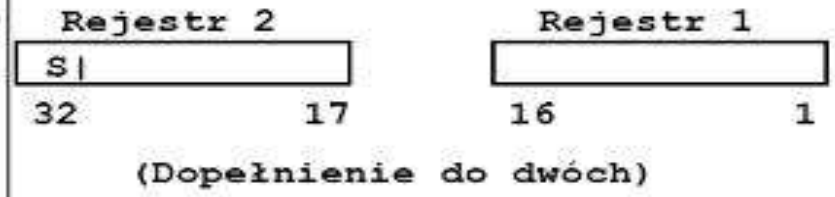
Zmienne rejestrowe

%R – zmienna 16 bitowa, rejestr w którym można przechowywać dane programu sterującego

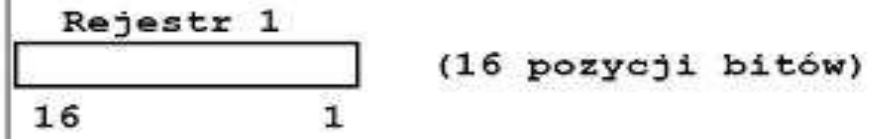
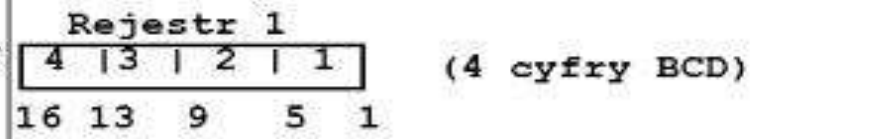
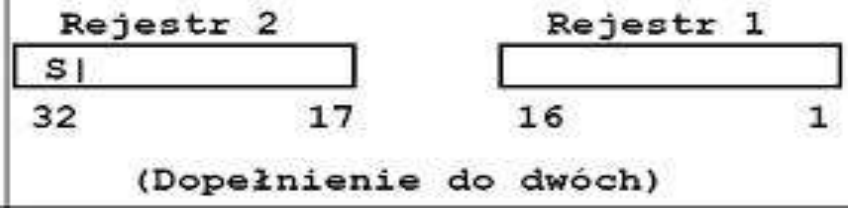
%AI –zmienna 16 bitowa, rejestr przeznaczony do wczytania wartości wejścia analogowego

%AQ –zmienna 16 bitowa, rejestr przeznaczony do zapisu wartości wyjścia analogowego

Typy danych

Typ	Nazwa	Opis	Format zapisu
INT	Liczby całkowite ze znakiem	Liczby całkowite ze znakiem zajmują 16 bitów pamięci i są zapisywane w formacie dopełnienia do dwóch dla liczb ujemnych. Zakres: od -32768 do +32768.	
DINT	Liczby całkowite podwójnej precyzji ze znakiem	Liczby całkowite podwójnej precyzji ze znakiem są przechowywane w 32 bitach pamięci i są zapisywane w formacie dopełnienia do dwóch dla liczb ujemnych. (Bit 32 to bit znaku.) Zakres: od 2,147,483,648 do +2,147,483,867.	
BIT	Bit	Dana zajmująca najmniejsza komórkę pamięci. Może przyjmować wartość 1 lub 0. Słowo bitowe może mieć długość N.	
BYTE	Bajt	Dana zawierająca 8 bitów. Zakres: od 0 do 255 (0 do FF w systemie heksadecymalnym).	

Typy danych cd.

WORD	Słowo	Słowo - wykorzystuje 16 kolejnych bitów pamięci sterownika, ale w przeciwieństwie do ciągu bitów reprezentującego w pamięci liczbę. bity mogą być niezależne od siebie. Każdy bit posiada swój własny stan logiczny (1 lub 0). Zakres wartości: 0 do FFFF (w systemie heksadecymalnym)	
BCD-4	Czterocyfrowa liczba dziesiętna zakodowana w formacie BCD (Four-Digit Binary Coded Decimal)	Czterocyfrowe liczby dziesiętne zakodowane w formacie BCD zajmują 16 bitów pamięci. Każda z czterech cyfr tej liczby jest zakodowana w czterech bitach i może reprezentować cyfrę z zakresu od 0 do 9. Zakres wartości: od 0 do 9999.	
REAL	Liczba rzeczywista	Liczby rzeczywiste zajmują 32 kolejne bity pamięci (w rzeczywistości są to dwie kolejne komórki pamięci po 16 bitów każda). Zakres wartości: ± 1.401298E-45 do ± 3.402823E+38.	

Deklaracje zmiennych binarnych i analogowych.

Tab.3.4. Przykładowe sposoby reprezentacji danych liczbowych, łańcuchowych i czasowych.

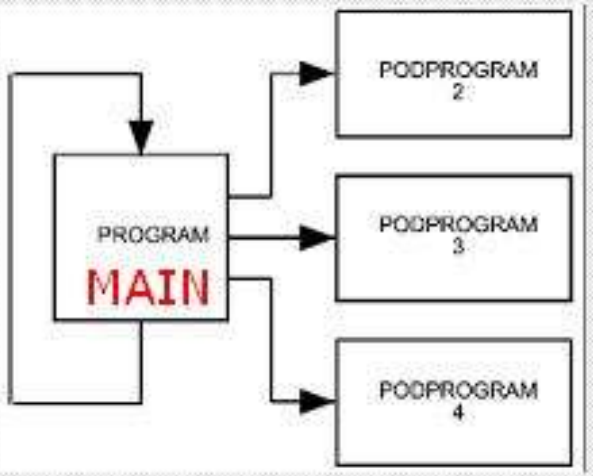
Typ danej	**Przykład**
Liczba całkowita	*123 ; -95 ; 100_000*
Liczba rzeczywista	*10.0 ; -123.456 ; 0.0 ; -1.5E-3*
Liczba binarna	*2#11110000 ; 2#0101_0101*
Liczba szesnastkowa	*16#A8 ; 16#FFFF*
Liczby i wyrażenia boolowskie	*0 ; 1 ; TRUE ; FALSE*
Ciąg znaków	*'ABCD' ; 'TEXTRL'*
Czas trwania	*T#10s ; T#3h_15m ; TIME#30ms*
Data	*DATE#2010-12-31 ; D#1970-01-01*
Godzina dnia	*TIME_OF_DAY#14:15:59.25*

Podprogramy

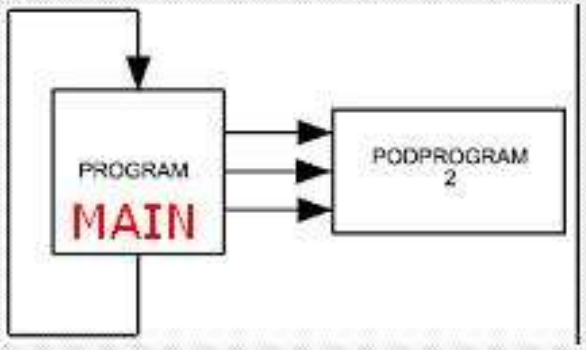
- Program może w trakcie wykonywania wywołać podprogram.
- Podprogram musi zostać zadeklarowany w projekcie dopiero wtedy podprogram ten można wywołać za pomocą instrukcji CALL.
- Maksymalnie w programie mogą zostać zadeklarowane 64 podprogramy, a dla każdego z bloków programu sterującego dozwolone są 64 instrukcje CALL.
- Maksymalny rozmiar podprogramów to 16 kB lub 3000 szczebli, lecz program główny wraz ze wszystkimi podprogramami musi zmieścić się w granicach obowiązujących dla poszczególnych jednostek centralnych.

Podprogramy
(różne możliwości wywołania)

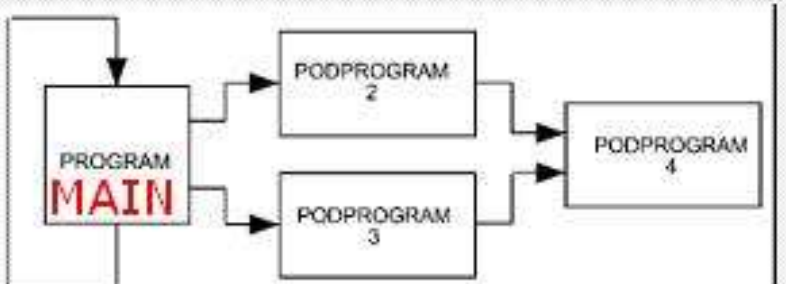
a) Wywołanie rożnych podprogramów



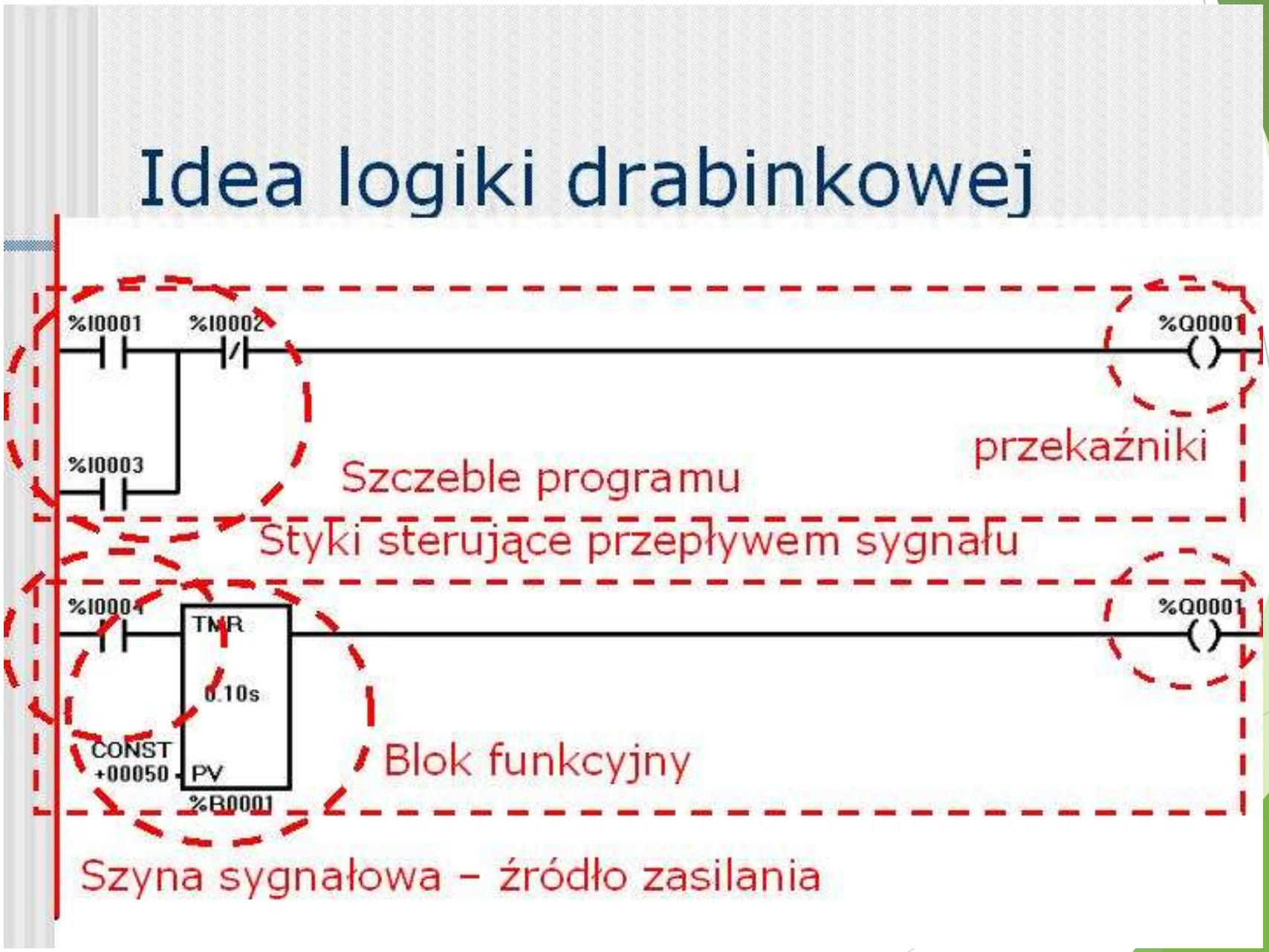
b)wywołanie 1 podprogramu



c) Podprogram wywołuje inny podprogram



Programowanie w języku drabinkowym. D.

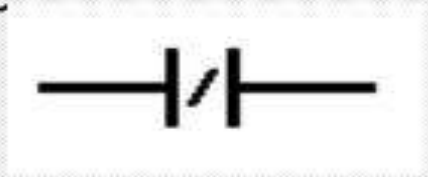


Programowanie w języku drabinkowym. D.

Styki

- Służą do sterowania przepływem sygnału w programie drabinkowym
- Wyróżniamy styki
  - Otwarte

  - Zamknięte

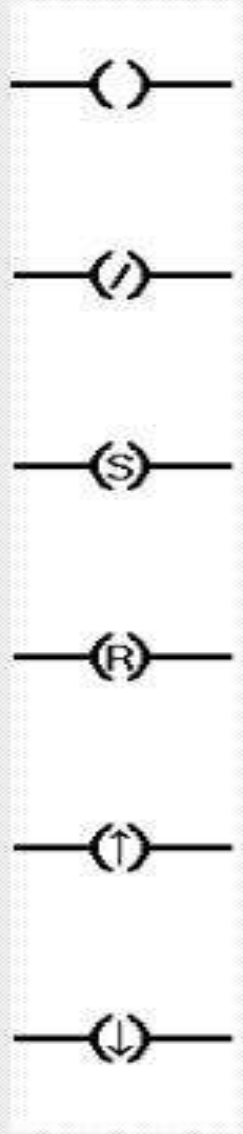


  - Stykom przypisuje się zmienne dyskretne (%I, %Q, %M, %S, %T, %G)

Programowanie w języku drabinkowym.

Przekaźniki

- Przekaźniki stosowane są w celu wpływania na stan zmiennych dyskretnych (nie dotyczy np. zmiennych %S)
- Przekaźniki
  - normalny
  - zanegowany
  - SET
  - RESET
  - ze zboczem narastającym
  - ze zboczem opadającym



Programowanie w języku drabinkowym.

Przekaźniki cd.

- Dopływ sygnału do przekaźnika musi być sterowany przez inne elementy logiczne (np. styki lub bloki funkcyjne)
- Jeżeli określony stan zmiennej przypisanej przekaźnikowi ma decydować o wykonaniu pewnej części programu sterującego, należy tam zastosować zmienną wewnętrzną
- Przekaźniki są zawsze umieszczane skrajnie, po prawej stronie linii programu sterującego.

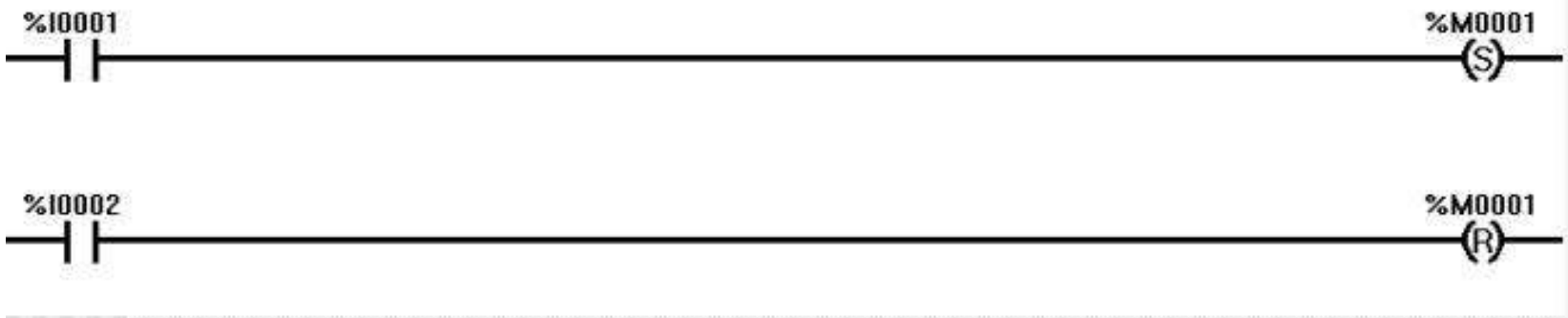
Programowanie w języku drabinkowym.

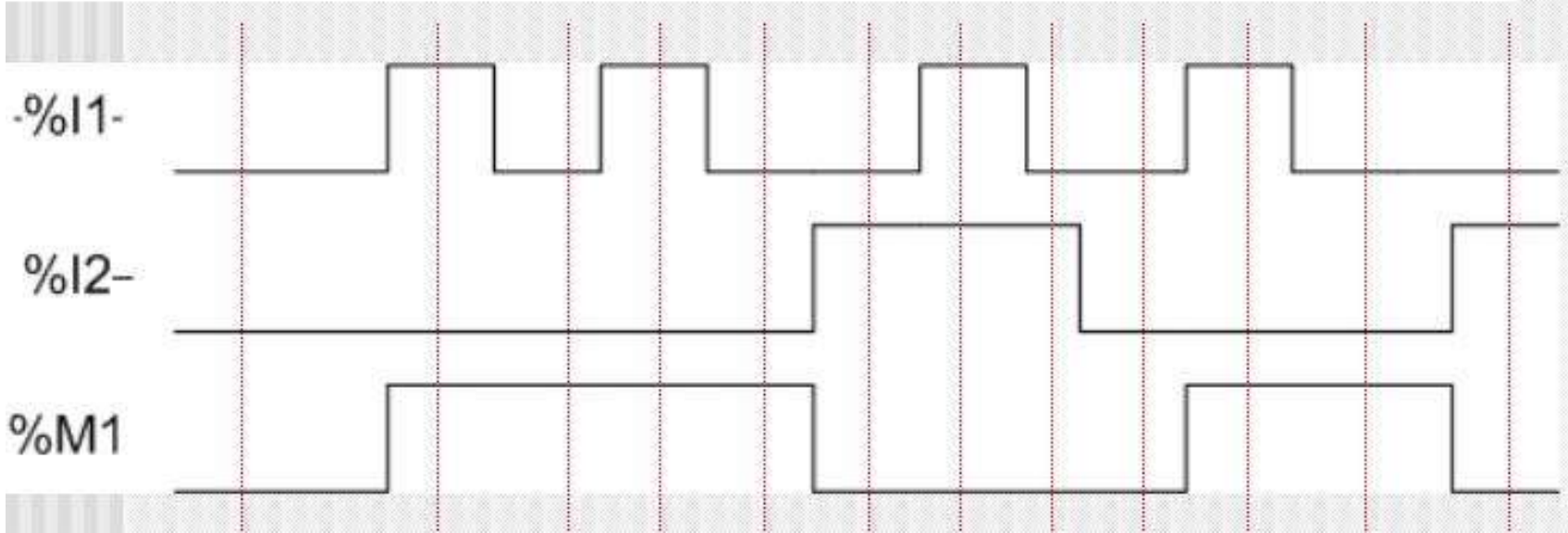
Przekaźniki sprzężone SET, RESET

- Gdy do przekaźnika SET dopłynie (choćby na chwilę) sygnał to przypisana do niego zmienna jest ustawiana w stan wysoki.
- Zmienna ta podtrzymywana jest w stanie wysokim, aż do zadziałania sprzężonego (przez nazwę zmiennej) przekaźnika

Programowanie w języku drabinkowym.

Przekaźniki SET, RESET cd.

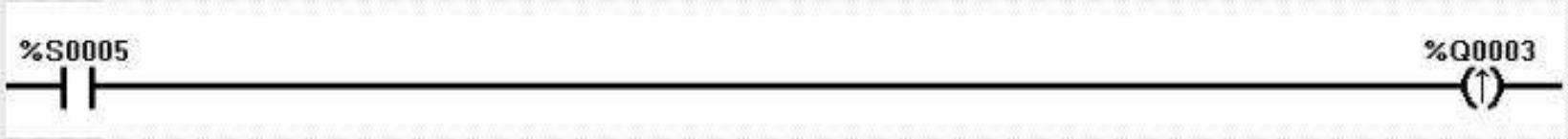


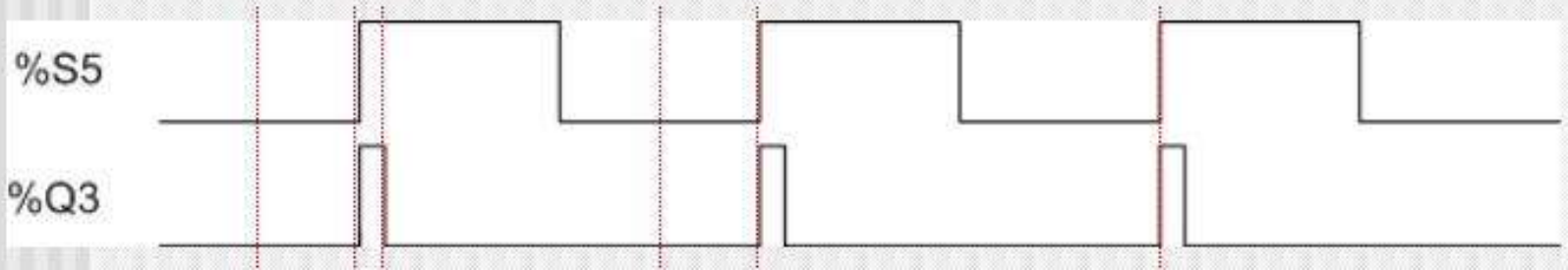


Programowanie w języku drabinkowym.

Przekaźnik ze zboczem narastającym

- Gdy do przekaźnika nie dopływał sygnał, a w bieżącym cyklu zaczął dopływać to na czas jednego cyklu przypisana do niego zmienna ustawiana jest w stan wysoki

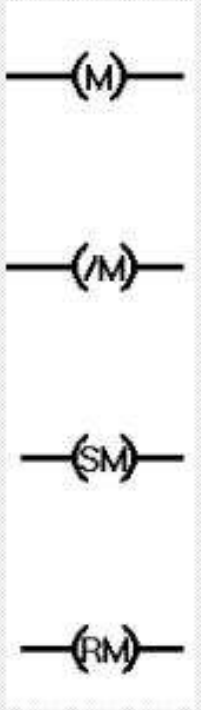




Programowanie w języku drabinkowym.

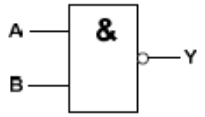
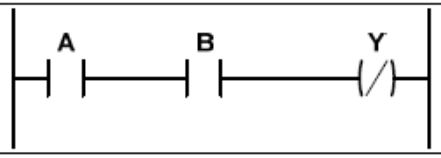
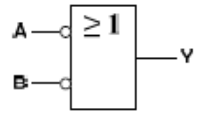
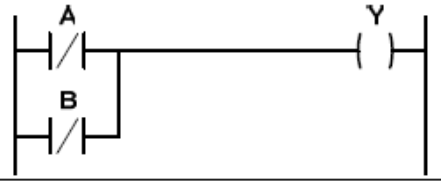
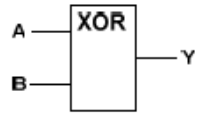
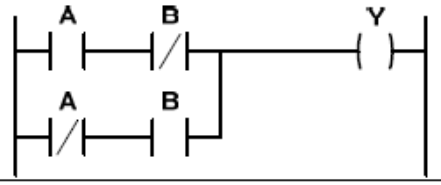
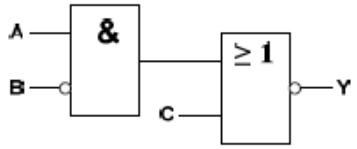
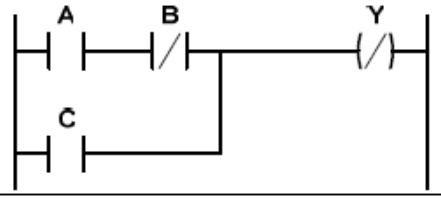
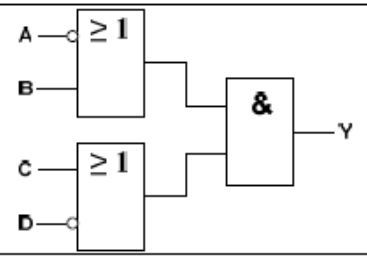
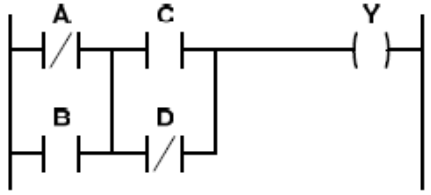
Przekaźniki z pamięcią

- Przekaźniki te działają analogicznie do odpowiadających im typom przekaźników bez pamięci, jednak stan zmiennej przypisanej do takiego przekaźnika zostaje zachowany nawet po wyłączeniu zasilania (za wyjątkiem %T i oczywiście %S)



Programowanie w języku drabinkowym. Stosowanie bloków funkcyjnych. Blok PID.

Przykłady realizacji funkcji logicznych w językach FBD i LD.

Wyrażenie logiczne	Realizacja w języku FBD	Realizacja w języku LD
$Y = \overline{A \cdot B}$		
$Y = \overline{A} + \overline{B}$		
$Y = A \oplus B$		
$Y = \overline{A + \overline{B} + C}$		
$Y = (\overline{A} + B) \cdot (C + \overline{D})$		

Programowanie….
Standardowe funkcje i bloki funkcjonalne.

W części trzeciej normy IEC 61131 określono grupę standardowych funkcji i bloków funkcjonalnych, które mogą występować we wszystkich zdefiniowanych językach programowania sterowników PLC. Przyjęto, że funkcjami będą nazywane elementy realizujące podstawowe operacje logiczne i arytmetyczne, natomiast do bloków funkcjonalnych zaliczać się będą elementy, które charakteryzują się pamięcią stanu (np. przerzutniki, czasomierze, liczniki).
Według normy, funkcje standardowe zostały podzielone na 7 grup:

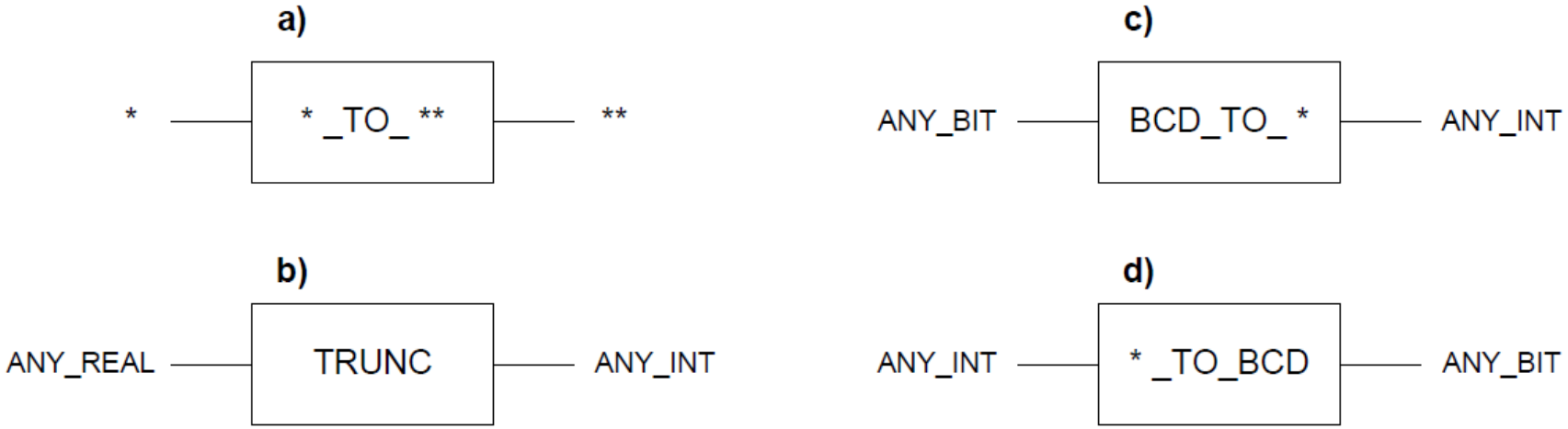
- Funkcje konwersji typów,
- Funkcje liczbowe,
- Funkcje na ciągach bitów,
- Funkcje wyboru i porównania,
- Funkcje na ciągach znaków,
- Funkcje na typach czasowych,
- Funkcje na typach wyliczeniowych.

Spośród standardowych bloków funkcjonalnych wyróżnić można następujące grupy:

- Elementy bistabilne (dwustanowe),
- Detektory zbocza,
- Liczniki,
- Czasomierze.

Standardowe funkcje konwersji typów

- Ogólna postać funkcji konwersji typu przedstawiona jest na rysunku. Symbolem (*) zaznaczono typ zmiennej wejściowej, zaś symbol (**) przedstawia wyjściowy typ zmiennej, po przetworzeniu. Przykładowe funkcje konwersji typów to: *INT_TO_REAL*, *BYTE_TO_WORD, BCD_TO_INT,* itp.
- Do funkcji konwertującej typ danej należy także funkcja *TRUNC*, która obcina część ułamkową liczby typu rzeczywistego, przekształcając ją do postaci liczby całkowitej.

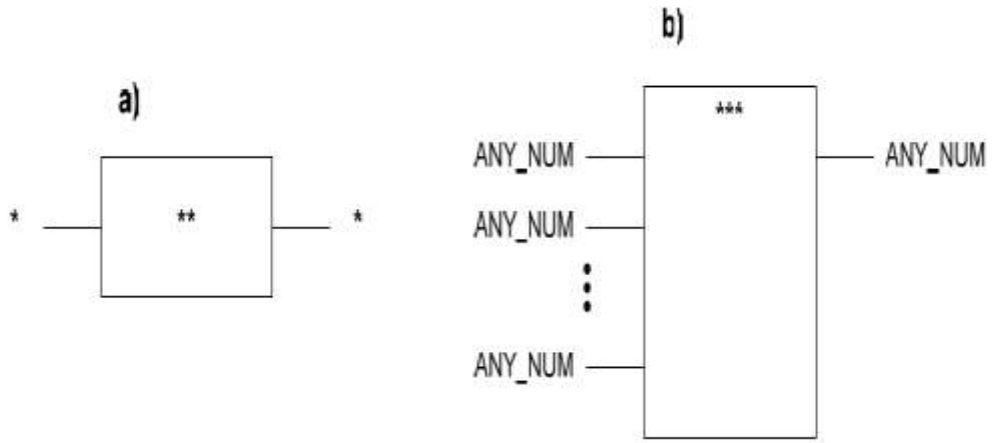


Rys.3.9. Przykładowe funkcje konwersji typów.

Standardowe funkcje liczbowe

Funkcje liczbowe wykonują operacje na liczbach o różnych typach danych.
Ogólnie dzielą się na dwie grupy:

- funkcje jednej zmiennej (np. pierwiastek kwadratowy, funkcje logarytmiczne, trygonometryczne),
- funkcje wielu zmiennych (funkcje arytmetyczne takie jak: dodawanie, odejmowanie, mnożenie, dzielenie itp.).



Tab.3.7. Standardowe funkcje liczbowe.

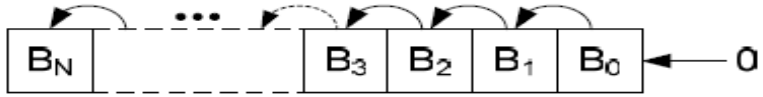
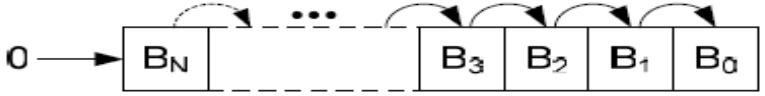
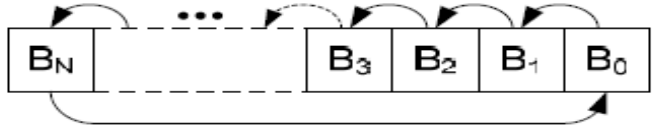
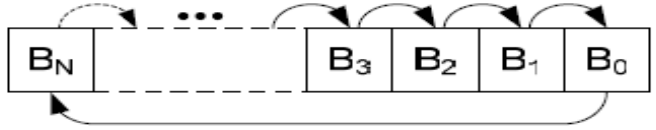
Nazwa funkcji	Opis
ABS	Wartość bezwzględna
SQRT	Pierwiastek kwadratowy
LN	Logarytm naturalny
LOG	Logarytm dziesiętny
EXP	Funkcja wykładnicza ex
SIN	Sinus [rad]
COS	Cosinus [rad]
TAN	Tangens [rad]
ASIN	Arcus sinus
ACOS	Arcus cosinus
ATAN	Arcus tangens
ADD	Dodawanie
MUL	Mnożenie
SUB	Odejmowanie
DIV	Dzielenie
MOD	Reszta dzielenia całkowitego

Standardowe funkcje na ciągach bitów

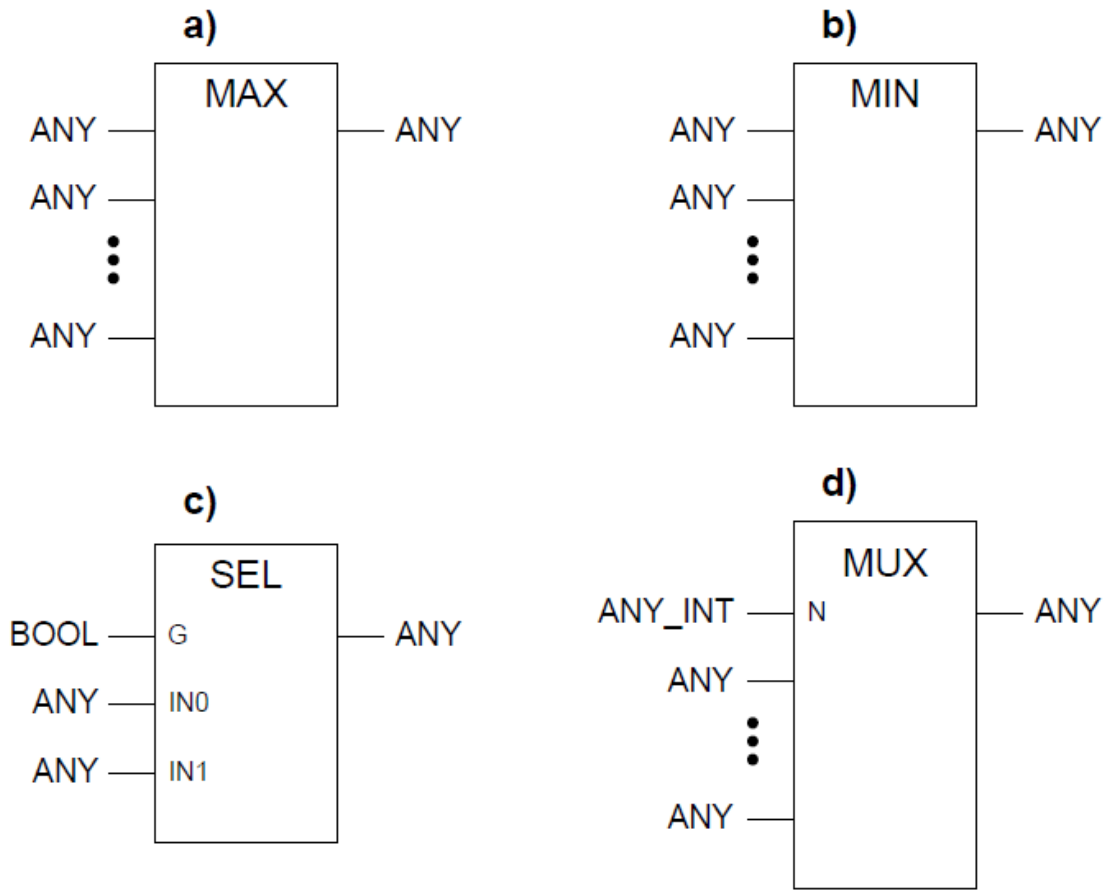
Funkcje standardowe operujące na ciągach bitów przeprowadzają operacje bitowe na danych należących do dowolnego typu binarnego *ANY_BIT* (np. *BYTE*, *WORD*, *DWORD*). Funkcje te dzielą się na:

- funkcje przesuwania bitów,
- funkcje logiczne.

Standardowe funkcje przesuwania bitów

SHL	Przesunięcie logiczne bitów w lewo o N pozycji 
SHR	Przesunięcie logiczne bitów w prawo o N pozycji 
ROL	Przesunięcie cykliczne (rotacja) bitów w lewo o N pozycji 
ROR	Przesunięcie cykliczne (rotacja) bitów w prawo o N pozycji 

Funkcje wyboru



Funkcje porównania (kompatatory)

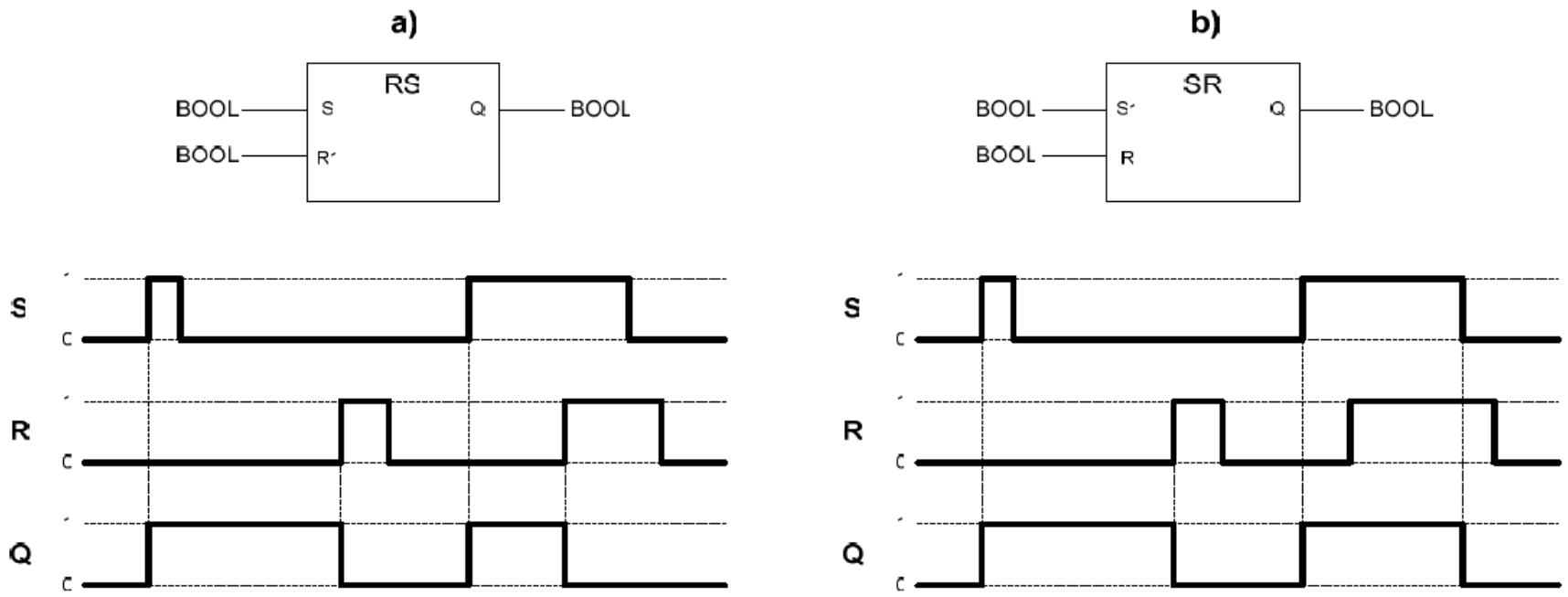
Funkcje te operują na danych wejściowych dowolnego typu. Wyjściem komparatorów jest sygnał logiczny, który przyjmuje wartość logiczna 1, wtedy gdy warunek porównania jest spełniony.

Tab.3.9. Standardowe funkcje porównania.

Nazwa	Symbol	Opis działania
GT	>	Komparator „większy niż” (ang. *Greater Than*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 > WE_2 > ... > WE_N$
GE	>=	Komparator „większy lub równy” (ang. *Greater or Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 \geq WE_2 \geq ... \geq WE_N$
EQ	=	Komparator „równy” (ang. *Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 = WE_2 = ... = WE_N$
LE	<=	Komparator „mniejszy lub równy” (ang. *Less or Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 \leq WE_2 \leq ... \leq WE_N$
LT	<	Komparator „mniejszy niż” (ang. *Less Than*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 < WE_2, < ... < WE_N$
NE	<>	Komparator „nie równy” (ang. *Not Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 \neq WE_2 \neq ... \neq WE_N$

Przerzutniki RS i SR

Różnica pomiędzy tymi elementami dotyczy specyfiki działania, przy jednoczesnym podaniu wartości logicznej „1” na oba wejścia (R i S). Otóż w przerzutniku RS dominującym wejściem jest wejście kasujące R, zaś w przerzutniku SR priorytetowym jest sygnał ustawiający S.



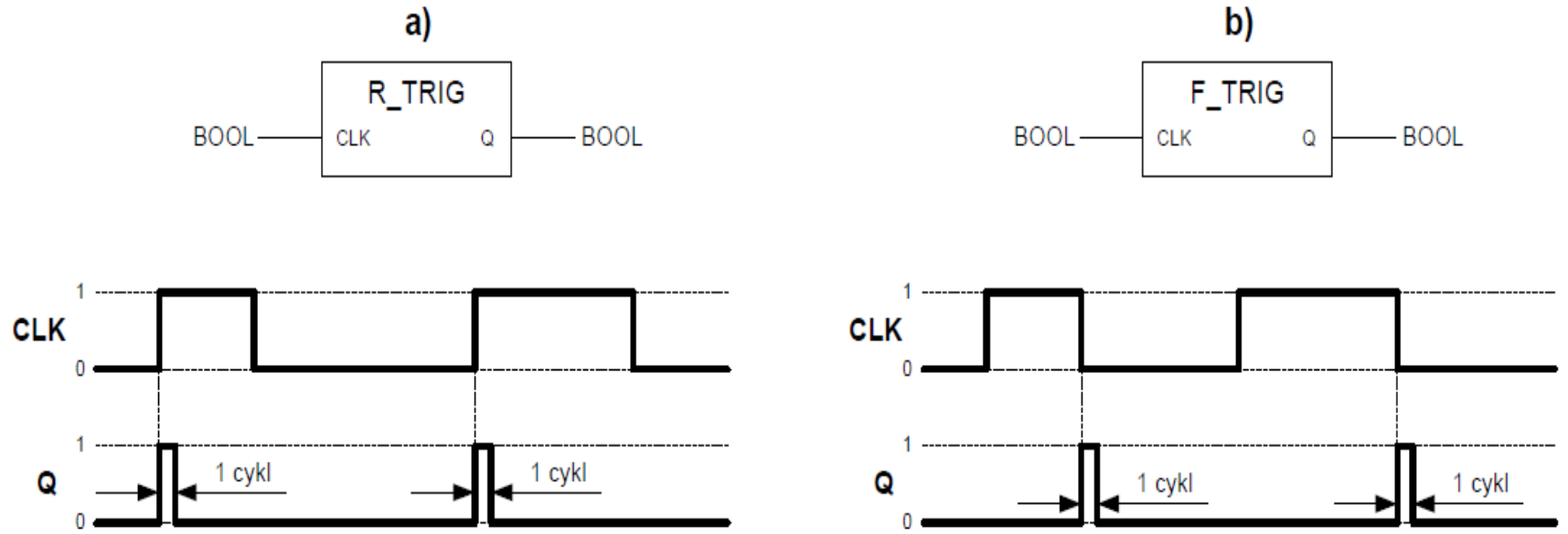
Rys.3.13. Symbole graficzne i zasada działania przerzutników RS (a) i SR (b).

Detektory zboczy

Do standardowych bloków funkcjonalnych, umożliwiających detekcje zbocza sygnałów cyfrowych należą:

- detektor zbocza narastającego R_TRIG (ang. *Rising Trigger*),
- detektor zbocza opadającego F_TRIG (ang. *Falling Trigger*).

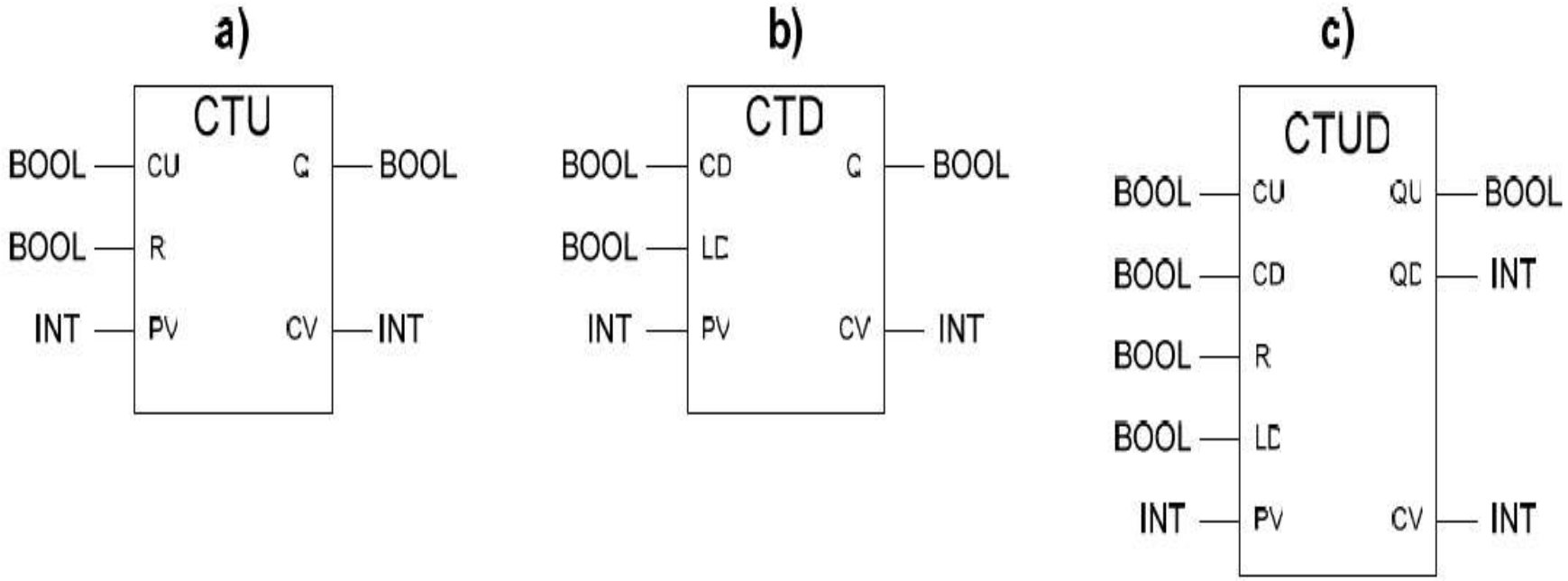
Detektor wykrywa zmianę stanu logicznego na wejściu CLK, powodując ustawienie na wyjściu wysokiego stanu logicznego na czas jednego cyklu przetwarzania sterownika. Pomimo krótkiego czasu trwania umożliwia ustawienie lub skasowanie przerzutników, taktowanie liczników czy wyzwolenie czasomierzy.



Timery i liczniki

Liczniki należą do standardowych bloków funkcjonalnych, umożliwiających zliczanie impulsów. W grupie tych elementów wyróżnia się:

- licznik zliczający w góre CTU (ang. Counter Up),
- licznik odliczający w dół CTD (ang. Counter Down),
- licznik dwukierunkowy CTUD (ang. Counter Up-Down).



Rys 3.15. Symbole graficzne liczników: a) – CTU, b) – CTD, c) – CTUD

Przekaźniki czasowe *(timers)* i liczniki *(counters)*

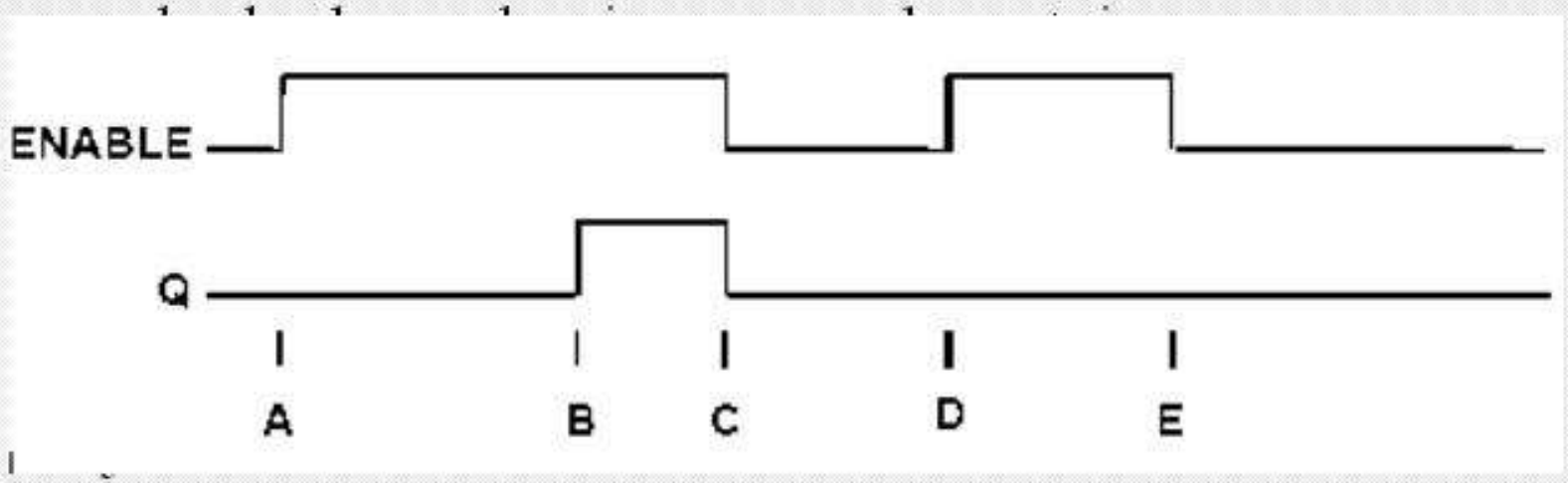
- Służą do odmierzania czasu i zliczania impulsów
- Potrzebują do swojej pracy trzech rejestrów pamięci %R

wartość bieżąca (CV)	słowo 1
wartość zadana (PV)	słowo 2
słowo sterujące	słowo 3

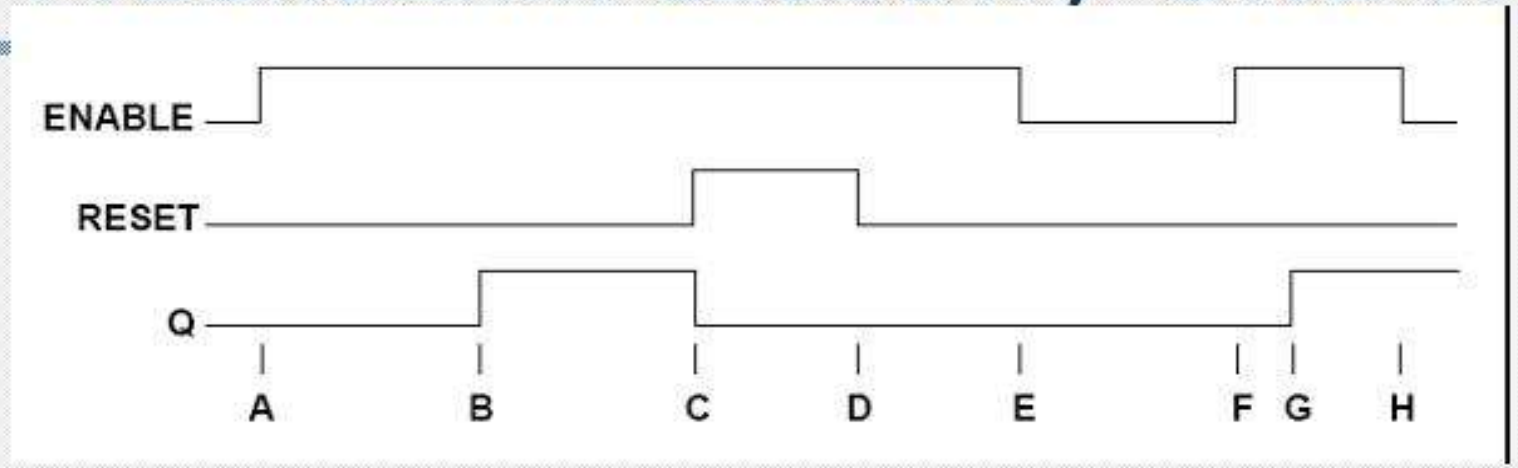
- Czas może być zliczany w dziesiątych, setnych lub tysięcznych częściach sekundy. Zakres zmierzonej wartości wynosi od 0 do +32767 jednostek czasu.Zatem zakres 0d 0.001 sek. do 3276,7 sek.

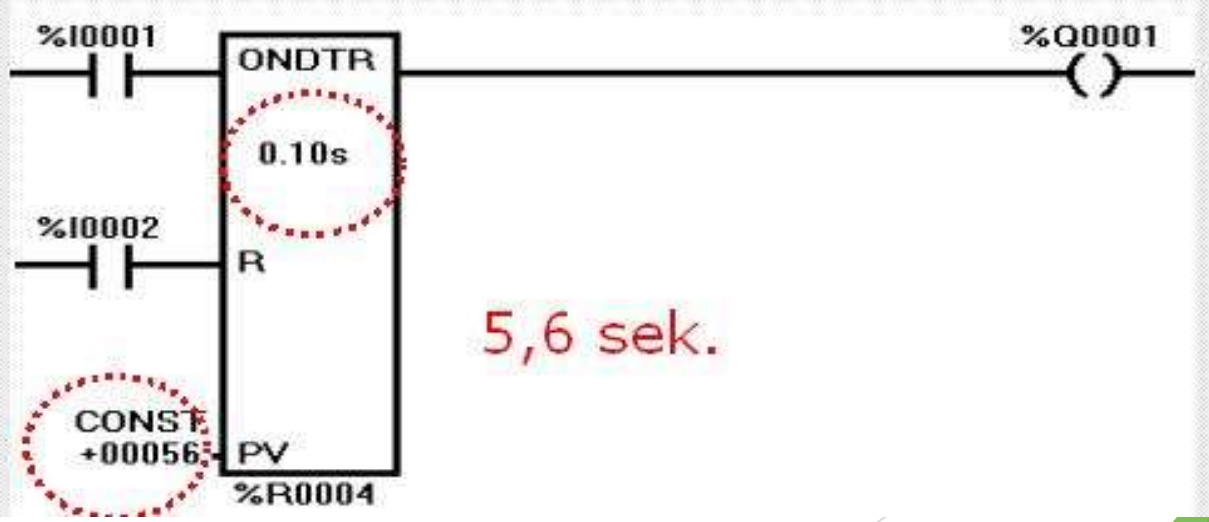
Przekaźnik czasowy TMR

- Przekaźnik czasowy bez pamięci (TMR) zlicza czas,



Przekaźnik czasowy ONDTR

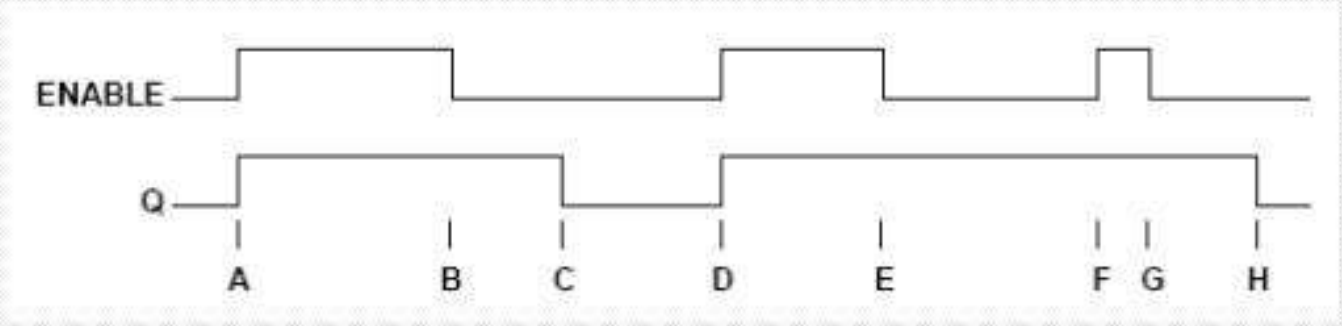




5,6 sek.

Przekaźnik czasowy OFDT

- Przekaźnik czasowy bez pamięci, z zanegowanym wejściem (OFDT) zlicza czas, gdy nie dopływa do niego sygnał i zostaje wyzerowany, gdy sygnał zacznie dopływać.



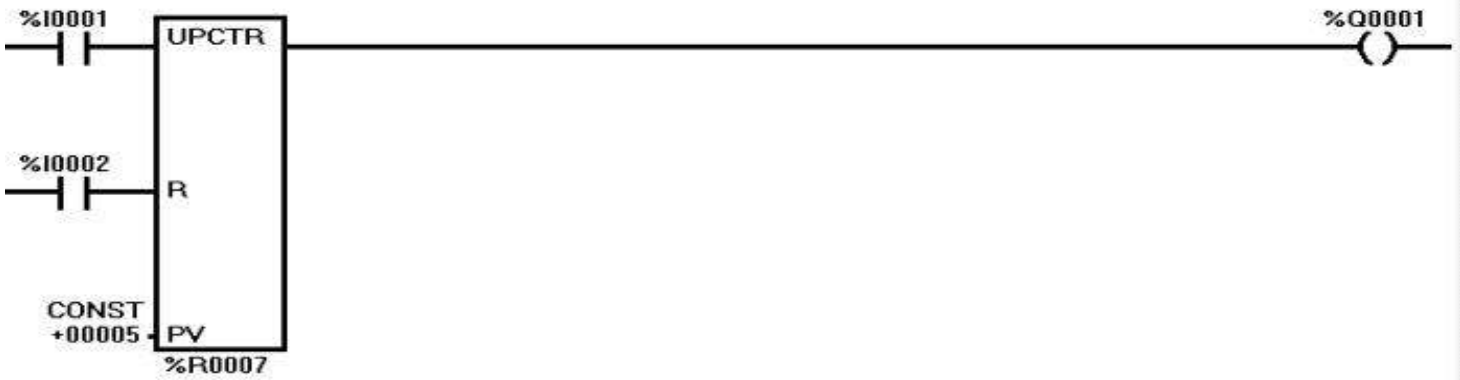
Liczniki

Licznik zliczający w górę UPCTR

- Licznik zliczający w górę służy do zliczania impulsów sygnału od 0 do zadanej wartości.
- Zakres licznika wynosi od 0 do +32767 impulsów. Podanie sygnału na wejście zerujące powoduje ustawienie wartości bieżącej licznika na 0.
- Zbocze narastające sygnału wejściowego (zmiana stanu sygnału wejściowego z 0 na 1) powoduje zwiększenie wartości bieżącej o 1. Wartość ta może być zwiększana ponad wartość zadaną PV.
- Sygnał wyjściowy jest wysyłany zawsze, gdy wartość bieżąca jest większa lub równa od wartości zadanej.

Liczniki

UPCTR



Liczniki

Licznik zliczający w dół DNCTR

- Licznik zliczający w dół (DNTCTR) służy do odliczania impulsów sygnału od zadanej wartości do 0.
- Minimalna wartość zadana może być równa zero, a maksymalna +32 767 impulsów. Minimalna wartość bieżąca wynosi -32 768. Podanie sygnału na wejście zerujące powoduje skopiowanie wartości bieżącej do rejestru, w którym przechowywana jest wartość zadana. Zbocze narastające sygnału wejściowego (zmiana stanu sygnału wejściowego z 0 na 1) powoduje zmniejszenie wartości bieżącej o 1. Sygnał wyjściowy jest wysyłany, gdy wartość bieżąca jest większa lub równa zeru.

Funkcje matematyczne

Funkcje matematyczne

Oznaczenie skrótowe	Funkcja	Opis
ADD	Dodawanie	Dodawanie dwóch liczb.
SUB	Odejmowanie	Odejmowanie dwóch liczb
MUL	Mnożenie	Mnożenie dwóch liczb.
DIV	Dzielenie bez reszty	Część całkowita z dzielenia dwóch liczb.
MOD	Dzielenie modulo	Reszta z dzielenia dwóch liczb.
SQRT	Pierwiastek kwadratowy	Obliczanie pierwiastka kwadratowego z liczby całkowitej lub rzeczywistej.
SIN, COS, TAN, ASIN, ACOS, ATAN	Funkcje trygonometryczne*	Obliczenie odpowiedniej funkcji trygonometrycznej dla parametru będącego liczbą rzeczywistą, podanego w parametrze IN.
LOG, LN EXP, EXPT	Funkcje logarytmiczne/ wykładnicze*	Obliczenie odpowiedniej funkcji trygonometrycznej dla parametru będącego liczbą rzeczywistą, podanego w parametrze IN.
RAD, DEG	Konwersja wartości kąta	Obliczenie odpowiedniej funkcji dla parametru będącego liczbą rzeczywistą, zadanego wejściu IN.

CPU >= 350

Funkcje matematyczne

- Po doprowadzeniu sygnału do funkcji, wykonywane jest odpowiednie działanie matematyczne na dwóch liczbach I1 i I2, które są parametrami wejściowymi bloku funkcyjnego.
- Obydwa parametry wejściowe muszą być takiego samego typu. Parametr wyjściowy Q jest też tego samego typu.

Typ	Opis
INT	Liczba całkowita ze znakiem (16 bitowa)
DINT	Liczba całkowita podwójnej precyzji ze znakiem (32 bitowa).
REAL	Liczba rzeczywista

Funkcje matematyczne

- %Q1 w stanie wysokim gdy %I1 w stanie wysokim oraz gdy wynik działania bloku jest poprawny

Relacje matematyczne

Oznaczenie skrótowe	Funkcja	Opis
EQ	"równe"	Sprawdzenie czy dwie liczby są równe.
NE	"nierówne"	Sprawdzenie czy dwie liczby mają różne wartości.
GT	"większe"	Sprawdzenie czy jedna liczba jest większa od drugiej.
GE	"większe lub równe"	Sprawdzenie czy jedna liczba jest większa lub równa drugiej.
LT	"mniejsze"	Sprawdzenie czy jedna liczba jest mniejsza od drugiej.
LE	"mniejsze lub równe"	Sprawdzenie czy jedna liczba jest mniejsza lub równa drugiej.
RANGE	Przedział	Sprawdzenie czy liczba mieści się w zadanym przedziale (funkcja dostępna w jednostkach centralnych wersja 4.5 lub wyższe).

Relacje matematyczne – przykład

%Q1 w stanie wysokim gdy %I1 w stanie wysokim oraz gdy wartość zmiennej %R1 jest większa lub równa wartości zmiennej %R3

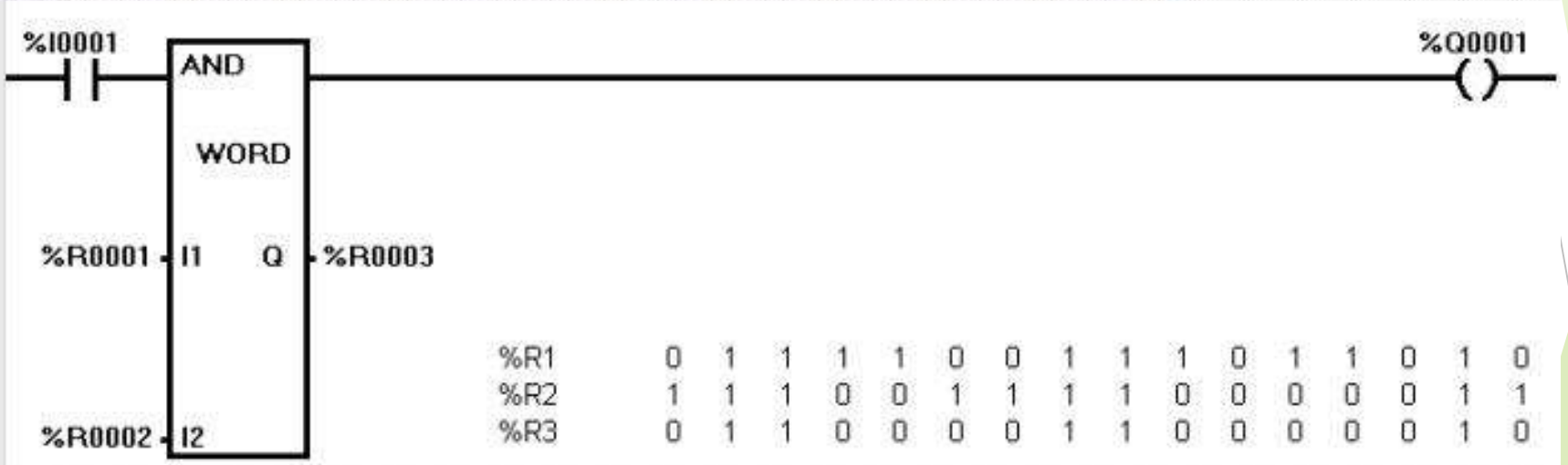
Operacje bitowe

- Bloki funkcyjne z tej grupy wykonują operacje logiczne na ciągach bitów. Funkcje AND, OR, XOR i NOT wykonują operacje na pojedynczym słowie.
- Pozostałe funkcje z tej grupy mogą wykonywać działania na ciągu słów, długość takiego ciągu nie może przekraczać 256 słów.
- Wszystkie funkcje do operacji bitowych wymagają danych typu WORD.

Operacje bitowe cd.

Oznaczenie skrótowe	Funkcja	Opis
AND	Logiczne AND	Jeżeli zarówno bit parametru wejściowego I1 jak i odpowiadający mu bit parametru wejściowego I2 mają wartość 1, wartość odpowiedniego bitu parametru wyjściowego Q jest ustawiana na 1.
OR	Logiczne OR	Jeżeli bit parametru wejściowego I1 ma wartość 1 i/lub odpowiadający mu bit parametru wejściowego I2 ma wartość 1, wartość odpowiedniego bitu parametru wyjściowego Q jest ustawiana na 1.
XOR	Alternatywa wyłączająca	Jeżeli bit parametru wejściowego I1 i odpowiadający mu bit parametru wejściowego I2 mają różne wartości, wartość odpowiedniego bitu parametru wyjściowego Q jest ustawiana na 1.
NOT	Negacja logiczna	Każdy bit w parametrze wyjściowym Q jest ustawiany na wartość przeciwną w stosunku do wartości odpowiedniego bitu parametru I1.

Operacje bitowe AND



Za każdym razem, po doprowadzeniu sygnału, funkcje AND i OR porównują każdy bit parametru I1 z odpowiednim bitem parametru I2, począwszy od najmniej znaczących bitów. W przypadku funkcji AND, wartość każdego bitu parametru wyjściowego Q jest ustawiana na 1, jeśli
odpowiednie bity pierwszego i drugiego parametru wejściowego (tzn. słów I1 oraz I2) mają wartość 1. Jeśli jeden lub obydwa bity mają wartość 0, to wartość odpowiedniego bitu słowa Q ustawiana jest na 0.

Operacje bitowe

- Argumenty mogą być typu mieszanego np. AND %R1 i %I17
- W przypadku zmiennych dyskretnych adres zmiennej jest pierwszym bitem (najmniej znaczącym) argumentu
- W powyższym przypadku wykonywane jest AND na zmiennych ze słowa %R1 i zmiennych od %I17 do %I32
- Wyrównywanie do pełnego bajtu!

Operacje bitowe

- Dane są wprawdzie pogrupowane w 16-bitowe słowa, lecz traktowane jako nieprzerwany ciąg bitów, z pierwszym bitem pierwszego słowa stanowiącym bit najmniej znaczący (LSB), i ostatnim bitem ostatniego słowa stanowiącym bit najbardziej znaczący (MSN)
- Przykładowo, jeśli funkcja ma wykonać operację na trzech słowach o adresie początkowym %R0100, to wykona ją na 48 kolejnych bitach,

%R0100	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1 ← 1 bit (LSB)
%R0101	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
%R0102	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33

↑

(MSB)

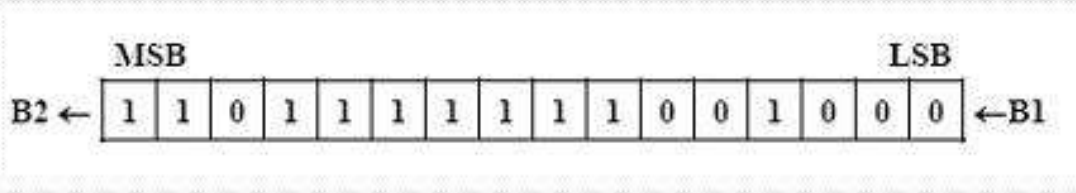
Operacje bitowe cd.

Oznaczenie skrótowe	Funkcja	Opis
SHL	Przesunięcie w lewo	Przesunięcie wszystkich bitów jednego słowa bitowego lub ciągu kilku słów bitowych w lewo, o wyszczególniona liczbę miejsc.
SHR	Przesunięcie w prawo	Przesunięcie wszystkich bitów jednego słowa bitowego lub ciągu kilku słów bitowych w prawo, o wyszczególniona liczbę miejsc.
ROL	Przesunięcie w lewo w obiegu zamkniętym	Przesunięcie wszystkich bitów jednego słowa bitowego lub ciągu kilku słów bitowych w lewo, o wyszczególniona liczbę miejsc, przy czym najbardziej znaczące bity (z lewej strony słowa) "wypchnięte" ze słowa bitowego zostają wpisane na puste miejsca z prawej strony słowa.
ROR	Przesunięcie w prawo w obiegu zamkniętym	Przesunięcie wszystkich bitów jednego słowa bitowego lub ciągu kilku słów bitowych w prawo, o wyszczególniona liczbę miejsc, przy czym najmniej znaczące bity (z prawej strony słowa) "wypchnięte" ze słowa bitowego zostają wpisane na puste miejsca z lewej strony słowa.

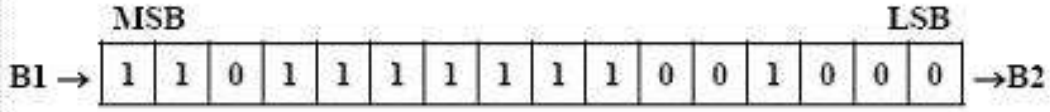
Operacje bitowe - przesunięcie

Funkcję SHL można wykorzystać do przesunięcia wszystkich bitów jednego słowa bitowego lub ciągu kilku słów bitowych w lewo, o wyszczególnioną liczbę miejsc. Wyższe bity (z lewej strony słowa) zostają "wypchnięte" ze słowa bitowego. Na puste miejsca zostają wpisane zadane wartości.

SHL (Shift Left)



SHR (Shift Right)

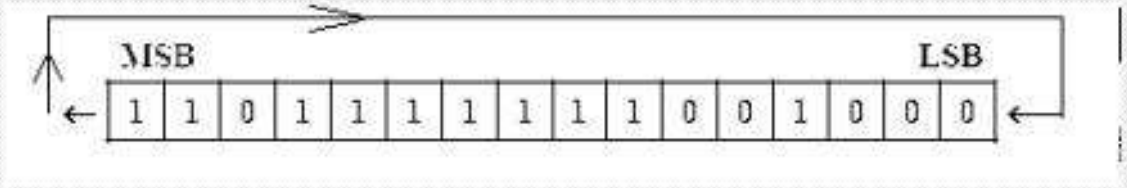


Od 1go do 256 słów bitowych

Operacje bitowe - rotacje

Funkcję ROL można wykorzystać do przesunięcia wszystkich bitów jednego słowa bitowego lub ciągu kilku słów bitowych w lewo, o wyszczególnioną liczbę miejsc. Najbardziej znaczące bity (z lewej strony słowa), "wypchnięte" ze słowa bitowego zostają wpisane na puste miejsca z prawej strony słowa.

ROL – ROtation Left



Operacje bitowe cd.

Oznaczenie skrótowe	Funkcja	Opis
BTST	Sprawdzanie wartości pojedynczego bitu	Sprawdzanie wartości (0 lub 1) jednego z bitów słowa bitowego.
BSET	Ustawianie wartości pojedynczego bitu na 1	Ustawianie wartości danego bitu w słowie bitowym na 1.
BCLR	Ustawianie wartości pojedynczego bitu na 0	Ustawienie wartości danego bitu w słowie bitowym na 0.
BPOS	Lokalizowanie pierwszego bitu o wartości 1	Przeszukiwanie słowa bitowego (lub ciągu słów) do napotkania pierwszego bitu o wartości równej 1.
MSKCMP	Porównanie dwóch słów bitowych z możliwością maskowania bitów	Porównanie kolejnych bitów dwóch ciągów bitów, z możliwością maskowania wybranych bitów (funkcja dostępna w jednostkach centralnych 4.5 lub wyższych).

Operacje przemieszczania danych

Oznaczenie skrótowe	Funkcja	Opis
MOVE	Przemieszczenie bitu, liczby lub słowa bitowego	Kopiowanie danych, traktowanych jako pojedyncze bity. Maksymalna, dopuszczalna długość wynosi 256 słów, za wyjątkiem MOVE_BIT, dla której wielkość ta wynosi 256 bitów. Dane mogą być przemieszczane do zmiennych trzech różnych typów, bez uprzedniej konwersji.
BLKMOV	Przemieszczanie grupy wartości	Kopiowanie grupy siedmiu stałych wartości do określonego obszaru pamięci sterownika. Wartości stałe stanowią część parametrów wejściowych tej funkcji
BLKCLR	Zerowanie fragmentu pamięci	Zerowanie określonego bloku pamięci sterownika. Funkcja ta może zostać wykorzystana do zerowania pamięci zmiennych dyskretnych (%I, %Q, %M, %G, or %T) i pamięci zmiennych rejestrowych (%R, %AI i %AQ). Maksymalna, dopuszczalna długość wynosi 256 słów.
SHFR	Rejestr przemieszczający	Wstawienie jednego lub większej liczby słów w określone miejsce pamięci. Maksymalna, dopuszczalna długość wynosi 256 słów.
BITSEQ	Przemieszczanie jedynki	Przemieszczanie sekwencji bitów w ciągu bitów. Maksymalna, dopuszczalna długość wynosi 256 bitów.
COMMREQ	Inicjalizacja komunikacji z jednym z modułów sterownika	Funkcja ta pozwala na nawiązanie komunikacji sterownika z jednym z wyspecjalizowanych modułów dodatkowych, jak np. z modułem komunikacyjnym GENIUS lub z modułem programowalnego koprocesora.

Operacje tablicowe

Oznaczenie skrótowe	Funkcja	Opis
ARRAY_MOVE	Kopiowanie danych	Kopiowanie określonej liczby danych z tablicy źródłowej do tablicy docelowej
SRCH_EQ	Szukanie wartości zadanej	Przeszukiwanie tablicy danych w celu znalezienia wartości równej wartości zadanej.
SRCH_NE	Szukanie wartości różnej	Przeszukiwanie tablicy danych w celu znalezienia wartości różnej od wartości zadanej.
SRCH_GT	Szukanie wartości większej	Przeszukiwanie tablicy danych w celu znalezienia wartości większej od wartości zadanej.
SRCH_GE	Szukanie wartości większej lub równej	Przeszukiwanie tablicy danych w celu znalezienia wartości większej lub równej wartości zadanej.
SRCH_LT	Szukanie wartości mniejszej	Przeszukiwanie tablicy danych w celu znalezienia wartości mniejszej od wartości zadanej.
SRCH_LE	Szukanie wartości mniejszej lub równej	Przeszukiwanie tablicy danych w celu znalezienia wartości mniejszej lub równej wartości zadanej.

Funkcje konwersji

Oznaczenie skrótowe	Funkcja	Opis
BCD-4	Konwersja na kod BCD	Konwersja danych typu INT na kod BCD.
INT	Konwersja na dane typu INT	Konwersja danych typu BCD-4 na dane typu INT.
DINT	Konwersja na dane typu DINT	Konwersja danych typu REAL na dane typu DINT.
REAL	Konwersja na dane typu REAL	Konwersja danych typu INT, DINT, BCD-4 lub WORD na dane typu REAL.
WORD	Konwersja na dane typu WORD	Konwersja danych typu REAL na dane typu WORD
TRUN	Obcinanie części dziesiętnej	Zaokrąglenie liczby rzeczywistej poprzez odrzucenie części dziesiętnej.

Funkcje sterujące

Funkcja	Opis
CALL	Wywołanie podprogramu w danym miejscu programu sterującego.
DOIO	Natychmiastowe uaktualnienie stanu wybranych wejść lub wyjść. (Funkcja ta obsługuje wszystkie wejścia i wyjścia modułu wyszczególnione jako parametry jej wywołania. Nie jest możliwe uaktualnienie wybranych (częściowe) wejść i wyjść modułu.) Opcjonalnie można umieścić kopię obsługiwanych wejść i wyjść w pamięci wewnętrznej, a nie w standardowej pamięci wejść dyskretnych.
SER	Szybki rejestrator zdarzeń do zapisu sygnałów. Blok sterujący tej funkcji zawiera informacje o sposobie wykonywania tego bloku funkcyjnego, wprowadzone przez użytkownika, opisy kanałów oraz parametry robocze.
END	Koniec programu. Program wykonywany jest od pierwszego szczebla drabiny logicznej aż do ostatniego szczebla, jeżeli jednak napotkana zostanie instrukcja END, wykonanie programu zostanie bezwarunkowo przerwane. Funkcja END jest użyteczna podczas uruchamiania programu, nie jest jednak dozwolone korzystanie z niej przy programowaniu SFC (proszę porównać z uwagą na stronie 4-198).

Funkcje sterujące

JUMP i JUMPN	Przejście do innego miejsca w programie sterującym (oznaczonego instrukcją LABEL, proszę porównać z zamieszczoną poniżej uwagą). Oprogramowanie Logicmaster 90-30/Micro udostępnia dwa rodzaje funkcji JUMP: bez możliwości pokrywania się zakresów działania (JUMP) i z możliwością pokrywania się zakresów działania (JUMPN).
LABEL i LABELN	Miejsce docelowe dla instrukcji skoku (JUMP). Oprogramowanie Logicmaster 90-30/Micro udostępnia dwa rodzaje funkcji LABEL: bez możliwości pokrywania się zakresów działania (LABEL) i z możliwością pokrywania się zakresów działania (LABELN).
COMMENT	Wstawienie komentarza (objaśnienia danego szczebla programu sterującego). Tekst komentarza można wprowadzić oraz wyświetlić na ekranie komputera w celu dokonania w nim ewentualnych zmian po zaakceptowaniu szczebla z wstawionym blokiem funkcyjnym COMMENT, poprzez ustawienie kursora w miejscu, w którym znajduje się blok i naciśnięcie klawisza F10 (Zoom).

Funkcje sterujące

SVCREQ	Uruchomienie specjalnych funkcji sterownika: • Odczyt/ zmiana liczby słów sumy kontrolnej programu sterującego. • Odczyt/ zmiana wskazań zegara podtrzymującego aktualną datę i czas • Zatrzymanie sterownika na końcu bieżącego cyklu. • Wymazanie komunikatów z tabeli błędów działania sterownika i układów wejść/wyjść. • Odczyt ostatnio zarejestrowanego w odpowiedniej tabeli komunikatu o błędzie działania sterownika oraz układów wejść/wyjść. • Odczyt wskazań zegara odmierzającego czas pracy sterownika • Kontrola występowania wymuszonej zmiany wartości zmiennych wejściowych i wyjściowych. • Odczyt sumy kontrolnej programu sterującego i konfiguracji. • Porównanie rzeczywistej konfiguracji modułów wejść/wyjść sterownika ze zdefiniowaną. Odczyt czasu trwania ostatniej przerwy w zasilaniu sterownika.
PID	Dwa regulatory proporcjonalno- całkowo- różniczkowe: • Standardowy regulator PID ISA (PIDISA). • Regulator PID o niezależnych wyrazach (PIDIND).

Blok funkcyjny regulatora PID z możliwością ręcznego zadawania parametrów

Blok funkcyjny regulatora PID z możliwością ręcznego zadawania parametrów

- Styki I1 zezwala na pracę regulatora. Gdy styki I2 jest wyłączony, to regulator pracuje w trybie automatycznym.
- Wartość SP zawiera wartość zadaną wielkości regulowanej (punkt pracy regulatora), PV jest wielkością regulowaną.
- Jeżeli natomiast chcemy zadać ręcznie wartość sygnału wyjściowego, to należy przejść w tryb pracy „manual” - przekaźnik I2 załączony. Teraz mamy możliwość zwiększania wartości wyjściowych regulatora - I3 lub zmniejszania - I4.
- Pamiętać trzeba, że regulator PID zajmuje 40 kolejnych rejestrów, więc nie powinniśmy ich używać przez inne bloki funkcyjne.
- Blok funkcyjny PID wysyła sygnał potwierdzający zrealizowanie algorytmu bez przeszkód do przekaźnika M1. Sygnał sterujący, wypracowany przez PID w podanym przykładzie przesyłany jest do pamięci od lokacji M97.
- Aby regulator PID zaczął działać należy zadać mu przynajmniej podstawowe parametry, tj. współczynnik proporcjonalności różny od zera oraz zakres wartości sygnału wyjściowego większy od zera. Można tego dokonać edytując blok funkcyjny PID (ustawić kursor na bloku funkcyjnym i nacisnąć F10).

Wskazówki do programowania sterowników

Wskazówka 1

W przypadku korzystania z przekaźników czasowych jak i liczników pamiętać należy o tym, że:

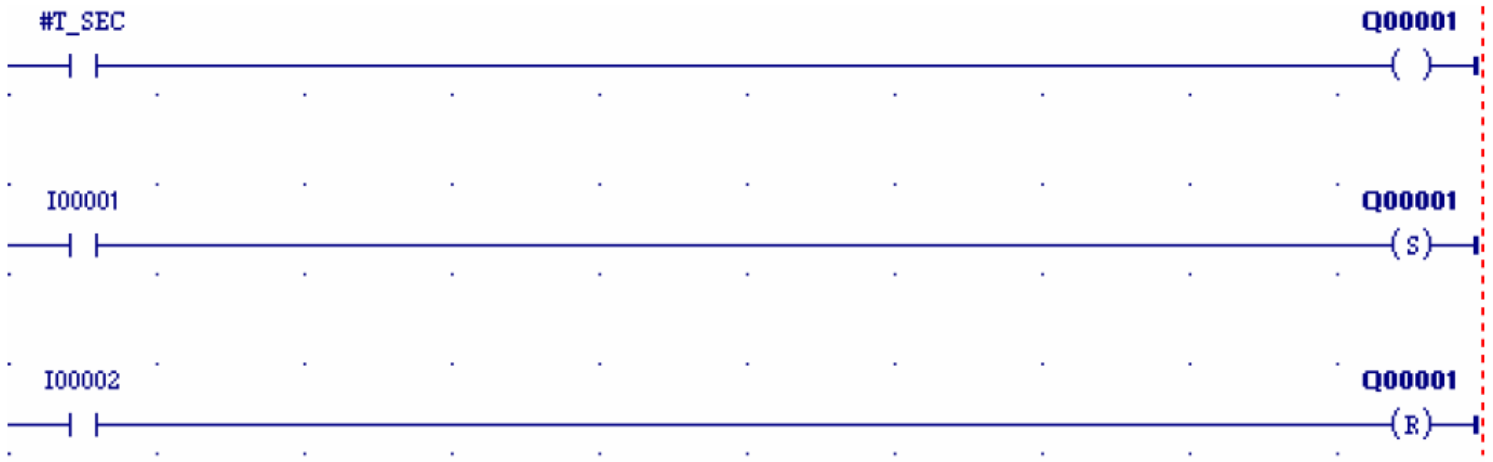
1. Odstęp w ich adresowaniu powinien być nie mniejszy jak 3 rejestry.
2. Niepodanie wartości zadanej PV spowoduje, że wyjście tego bloku funkcyjnego będzie cały czas aktywne.
3. Każdorazowy zanik sygnału zezwalającego *Enable* spowoduje wyzerowanie TMR.

Dla próby proponujemy sprawdzić zasadę pierwszą - ulokować jeden przekaźnik czasowy np. w rejestrze R2, a drugi przekaźnik czasowy np. w rejestrze R3 i sprawdzić, jaki to ma wpływ na pracę tego typu bloków funkcyjnych.

Wskazówki do programowania sterowników

Wskazówka 2

Z dużą rozwagą należy podchodzić do sytuacji, gdy stosujemy różne typy zmiennych dla tej samej komórki rejestru:



W zaprezentowanym przykładzie można jeszcze kontrolować przebieg wykonywania programu. Niestety, w praktyce spotkać się można z o wiele bardziej rozbudowanymi strukturami, zawierającymi instrukcje skoku czy też podprogramy. Wtedy doprowadzenie do konfliktu typów zmiennych powoduje, że przestajemy kontrolować przebieg programu i nie jesteśmy w stanie przewidzieć stanu, w jakim znajdzie się sterownik.

Wskazówki do programowania sterowników

- Wskazówka 3

Stosując bloki MOVE możemy dokonywać przemieszczenia bitu, liczby lub słowa bitowego. Przemieszczenia mogą się odbywać pomiędzy wejściami, wyjściami i rejestrami, z uwzględnieniem typów zmiennych. Przesyłając wartości z określonych rejestrów do pamięci bitowej pamiętać trzeba o tym, że jeden rejestr zawiera 16 bitów - więc np. przesłanie trzech rejestrów spowoduje zwiększenie indeksacji adresów obszaru bitowego o 48 (a nie o 3).

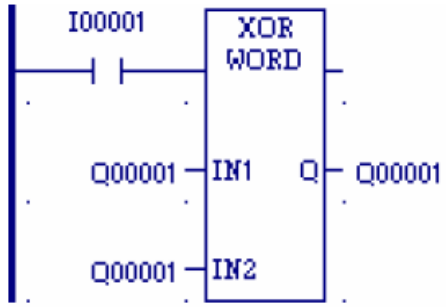
Wskazówki do programowania sterowników

Wskazówka 4

Do wyzerowania bitu pamięci w sterowniku może służyć przekaźnik:

a do wyzerowania słowa bitowego blok XOR, AND, BLK CLR, itp.

Przykładowo podano sposób wyzerowania słowa bitowego zaczynającego się w Q1:



Najczęściej popełniane błędy logiczne (nie konfiguracja, nie składnia)

- Kilka przekaźników w programie z tym samym adresem zmiennej
- Brak sprzężenia przekaźników SET RESET tą samą zmienną
- Nadpisanie rejestrów licznika lub przekaźnika czasowego
- Niewłaściwa kolejność szczebli

Najczęściej popełniane błędy

- Nadpisanie drugiego słowa zmiennej REAL
- Przekroczenie zakresu zmiennych
- Niewłaściwe wykorzystanie procedur
- Brak inicjalizacji zmiennych
- BRAK ZEROWANIA PAMIĘCI przy rozpoczęciu testowania programu

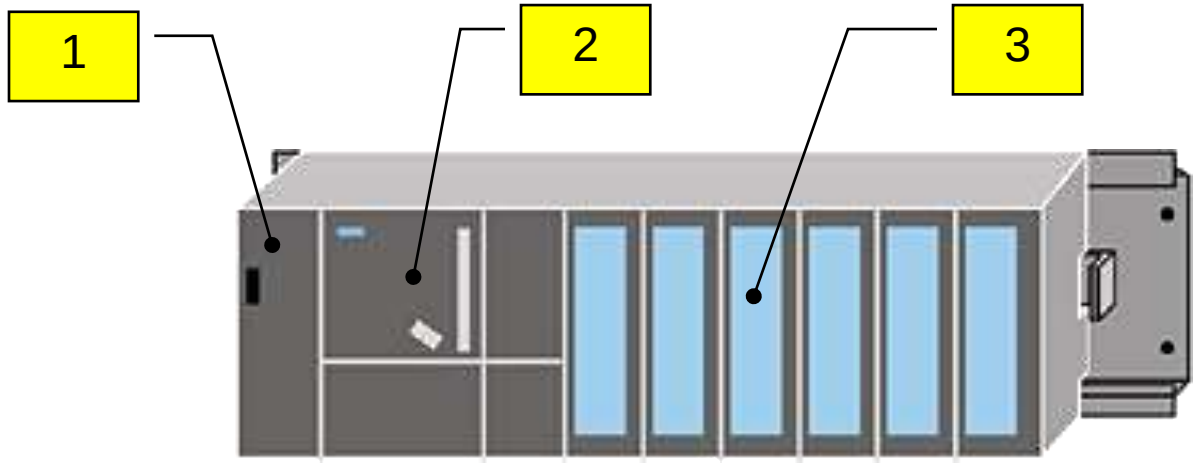
Sterowniki programowalne PLC

firmy Siemens: S5-95U i S7-314IFM

Dane techniczne

	S5-95U	S7-314IFM
Język programowania	STEP 5	STEP7
Pamięć	16 kB	24 kB
Czas obróbki instrukcji binarnej	2µs	1µs
Znaczniki	2048	2048
Timery	128 0,01 ÷ 9990s	128 0,01 ÷ 9990s
Liczniki	128 0 ÷ 999	128 0 ÷ 999
Wejścia cyfrowe (wbudowane)	16	16 + 4
Wyjścia cyfrowe (wbudowane)	16	16
Wejścia analogowe (wbudowane)	8 0 ÷ 10 V	4 0 ÷ 10 V
Wyjścia analogowe (wbudowane)	1 0 ÷ 10 V ; 0 ÷20 mA	1 0 ÷ 10 V ; 0 ÷20 mA
Możliwości rozbudowy	32 moduły	32 moduły

Montaż sterowników

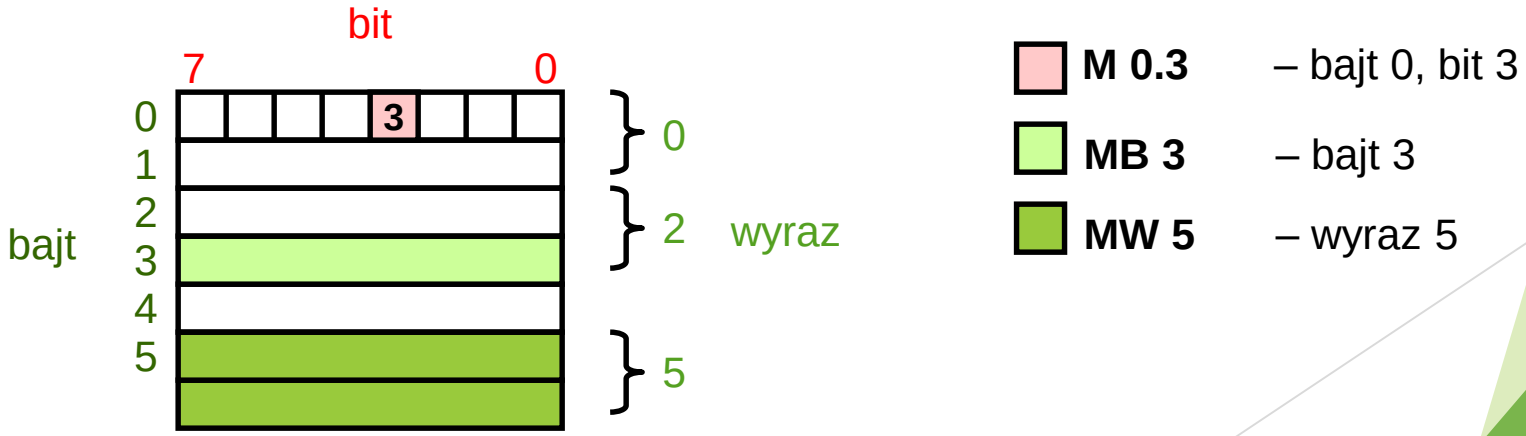


Sterowniki PLC są produkowane w postaci modułów montowanych na szynie montażowej w następującej kolejności:

1. Zasilacz.
2. Jednostka sterująca.
3. Moduły I/O (wejścia i wyjścia).

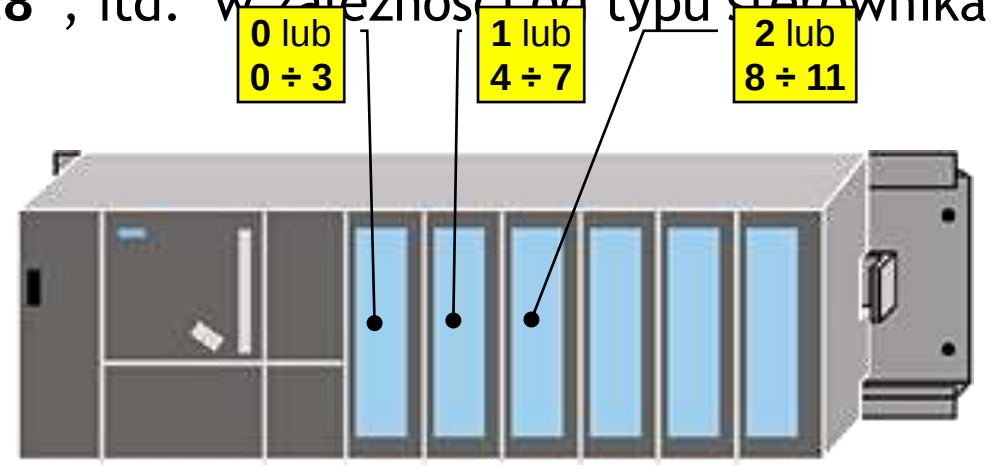
Adresowanie pamięci

W pamięci sterownika wyodrębniona jest pewna ilość miejsc do przechowywania chwilowych wyników operacji. W sterownikach PLC rozróżniamy 4 tryby adresowania: bitowo, bajtowo, wyrazowo oraz przy pomocy dwóch słów. Adresując słownie operujemy na 16-tu bitach i przy pomocy dwóch słów na 32-ch bitach.



Adresowanie modułów

Adresowanie modułów przebiega podobnie jak adresowanie pamięci. W przypadku wejść i wyjść binarnych podajemy numer modułu i po kropce numer zacisku a w przypadku modułów analogowych tylko numer zacisku (adresowanie wyrazowe). Numer modułu zależy od jego umiejscowienia na szynie. Pierwszy moduł otrzymuje adres „**0**” a następne „**1**”, „**2**” itd. lub „**4**”, „**8**”, itd. w zależności od typu sterownika i modułów.



Np. aby odczytać czujnik podłączony do modułu nr „**0**” i zacisku „**3**” wpisujmy: **I 0.3**

Operandy

Przed opisem numerycznym wejścia, wyjścia lub pamięci dodaje się symbol literowy identyfikujący dany adres. Ponieważ dopuszczalny jest zapis w języku angielskim i niemieckim poniżej przedstawiono zapis niektórych operandów.

	oznaczenie		**bitowo**	**bajtowo**	**wyrazowo**
	ang.	**niem.**			
wejścia	I	E	×	×	×
wyjścia	Q	A	×	×	×
flagi	F	M	×	×	×
dane	D	D		×	×
timer	T	T		×	
licznik	C	Z		×	
stałe	K	K		×	×

Moduł czasowy (timer)

Działanie modułu czasowego odpowiada sposobowi działania przekaźnika czasowego z opóźnionym załączaniem lub wyłączaniem. Maksymalnie można zaprogramować 128 modułów czasowych oznaczonych instrukcją T0 do T127. W sterownikach Simatic możemy korzystać z 5-ciu różnie działających układów czasowych:

SP – **Pulse Timer**: Daje na wyjściu sygnał o określonej długości tylko przy aktywnym sygnale START.

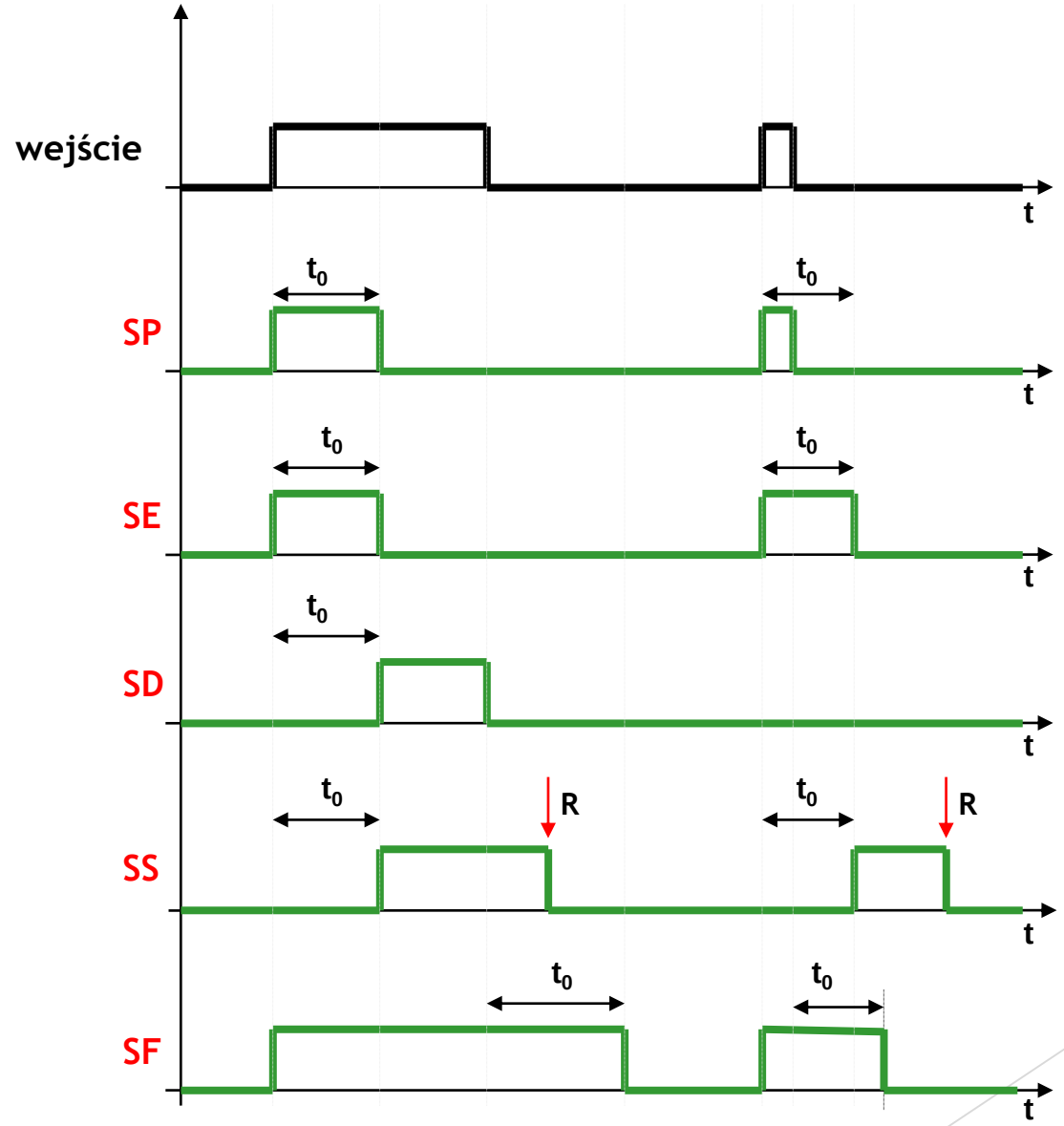
SE – **Extended Pulse Timer**: Daje na wyjściu sygnał o określonej długości przy krótkiej aktywacji sygnału START.

SD – **On-Delay Timer**: Ustawienie timera jako timer z opóźnionym załączaniem.

SS – **Retentiv On-Delay Timer**: Uaktywnia się przez krótką aktywację sygnału START. Kasowanie jest możliwe tylko wejściem kasującym.

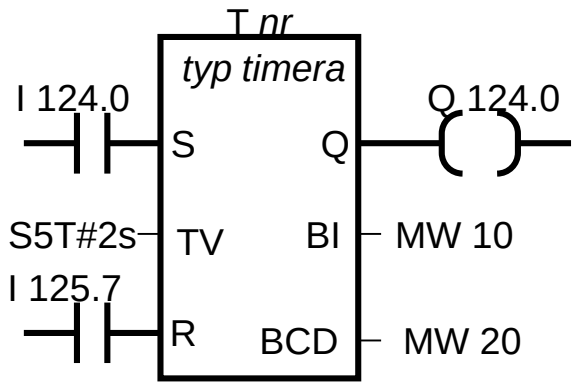
SF – **Off-Delay Timer**: Ustawienie timera jako timer z opóźnionym wyłączaniem.

Wykresy czasowe



Wykorzystanie timera

LAD



STL

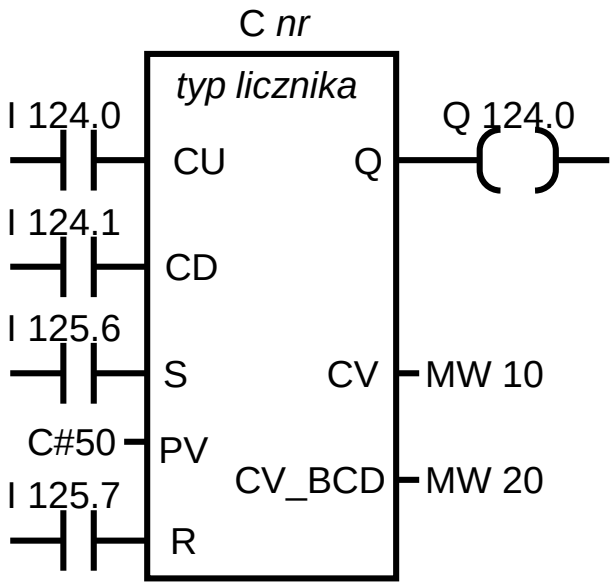
```
A     I  124.0
L     S5T#2s
SP    T1
A     I  125.7
R     T1
L     T1
T     MW  10
LC    T1
T     MW  20
A     T1
=     Q124.0
```

Typ timera	STEP 5	STEP 7
SP	1_-_	S_PULSE
SE	1_-_V	S_PEXT
SD	T!_!0	S_ODT
SS	T!_!S	S_ODTS
SF	0!_!T	S_OFFDT

Liczniki

Licznik może zliczać sygnały zarówno w przód jak i do tyłu. Zakres liczenia zawiera się w przedziale od 0 do 999. Maksymalnie można zaprogramować 128 liczników.

LAD



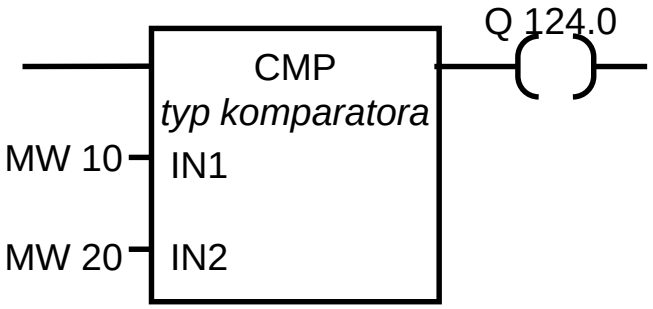
STL

```
A     I  124.0
CU    C1
A     I  124.1
CD    C1
A     I  125.6
L     C# 50
S     C1
A     I  125.7
R     C1
L     C1
T     MW10
LC    C1
T     MW20
A     C1
=     Q124.0
```

Komparatory

Komparator służy do porównywania ze sobą dwóch wartości 16-bitowych lub 32-bitowych.

LAD



STL

```
L     MW10
L     MW20
==I
=     Q124.0
```

typ komparatora	STEP 5 16-bit	STEP 7		
		16-bit	32-bit	rzeczywiste
równy	!=F	==I	==D	==R
różny	><F	<>I	<>D	<>R
większy	>F	>I	>D	>R
mniejszy	<F	<I	<D	<R
większy lub równy	>=F	>=I	>=D	>=R
mniejszy lub równy	<=F	<=I	<=D	<=R

Przykład

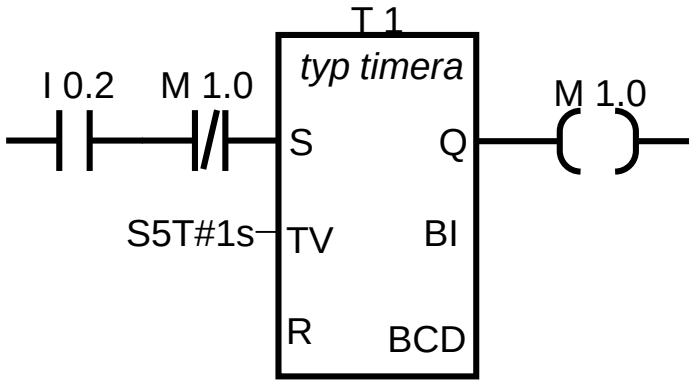
Programowanie i działanie sterownika PLC najlepiej zobrazować na przykładzie.

Problem:
Przy załączonym wejściu I 0.2 na wyjściu Q 2.5 ma być generowany sygnał taktujący o stałym i równym czasie trwania impulsu i pauzy.

Rozwiązanie:
Zadanie można rozwiązać programując dwie gałęzie:
1. Timer typu SD generujący w pamięci impuls co 1 sekundę w czasie gdy naciśnięty jest przycisk startujący (I 0.2).
2. Układ przełączający stan lampki (Q 2.5) w momencie wystąpienia impulsu.

Rozwiązanie

1



2