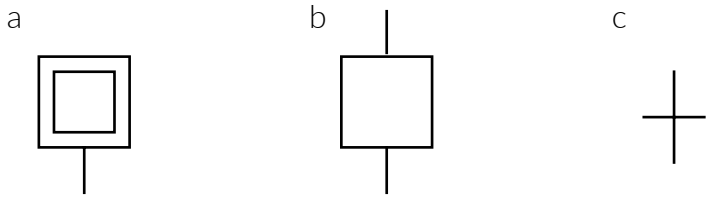
Układy sekwencyjne

Układ sekwencyjny to układ, w którym poziomy sygnałów wyjściowych zależą od poziomów sygnałów wejściowych oraz od stanu wewnętrznego w którym układ znajdował się poprzednio.

Klasyczne metody syntezy układów sekwencyjnych, wykorzystują elementarne układy logiczne, takie jak bramki i przerzutniki. W praktyce stosowanie tych metod ograniczone jest liczbą sygnałów wejściowych i stanów wewnętrznych układu. Bardziej złożone układy sekwencyjne wymagają zastosowania technik dekompozycji i wspomagania komputerowego.

Synteza układu sekwencyjnego przy użyciu sterownika PLC, sprowadza się do opracowania algorytmu sterowania sekwencyjnego, jego implementacji z wykorzystaniem wybranego języka programowania, kompilacji i załadowania do sterownika. W celu usprawnienia procesu syntezy i analizy pracy układu sekwencyjnego, opracowano język **SFC** (ang. Sequential Function Chart). Język ten jest językiem graficznym, który składa się z kilku elementów wykorzystywanych do budowy sieci działań układu sekwencyjnego. Elementy języka SFC:



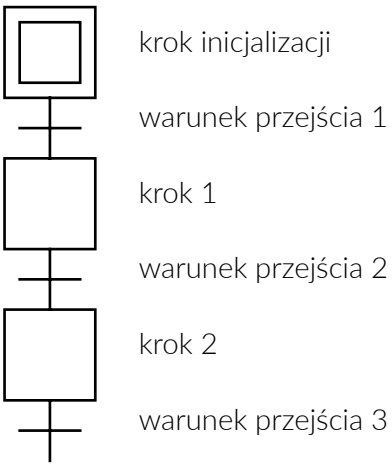
Rys. 1. Elementy języka SFC, a - krok inicjalizacji (ang. initialization step), b - krok (ang. step), c - przejście (ang. transition)

Krok pełni funkcje elementu wykonawczego. Zawiera instrukcje jakie mają zostać wykonane w danym stanie.

Krok inicjalizacji definiuje instrukcje, które należy wykonać inicjując stan początkowy układu sekwencyjnego.

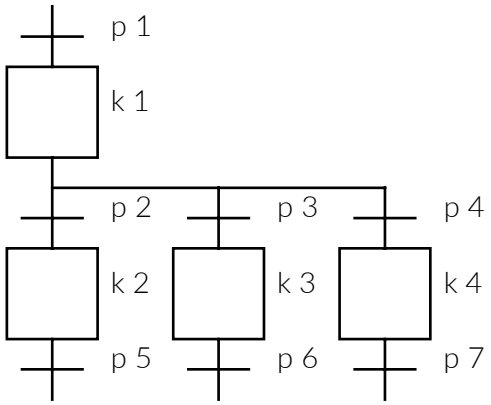
Przejście pełni funkcje instrukcji warunkowej. Zawiera warunek logiczny, którego spełnienie powoduje przejście do kolejnego kroku.

Język SFC zakłada, że każdy element kroku musi być poprzedzony elementem przejścia. Wyjątkiem jest element kroku inicjalizacji, który może lecz nie musi być poprzedzony warunkiem przejścia.



Rys. 2. Przykładowa sekwencja prosta

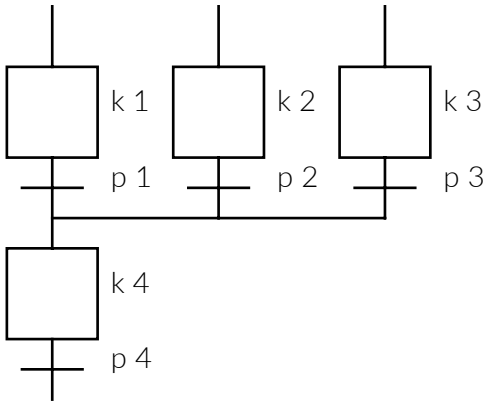
W przypadku, gdy zachodzi konieczność wybrania jednej z kilku sekwencji, należy zastosować **rozgałęzienie alternatywne**. Wybór danej sekwencji uzależniony jest od spełnienia jednego z kilku zdefiniowanych warunków.



Rys. 3. Rozgałęzienie alternatywne

Dla przykładu z rysunku 3, sekwencja w której pierwszym krokiem jest krok k 2, wybrana zostanie tylko wtedy gdy spełniony zostanie warunek p 2. Gdy spełniony zostanie warunek p 3, wybrana zostanie sekwencja w której pierwszym krokiem jest krok k 3. Jeżeli spełniony zostanie warunek p 4, to wybrana zostanie sekwencja w której pierwszym krokiem jest krok k 4. W przypadku gdy wszystkie warunki (p 2, p 3, p 4) są spełnione wybrana zostanie pierwsza sekwencja od lewej.

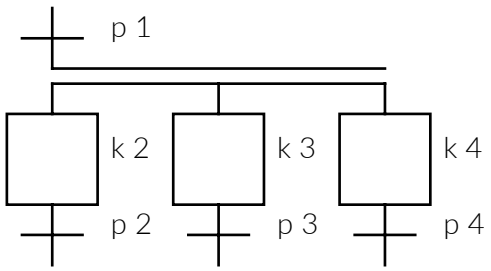
Alternatywne sekwencje muszą zostać połączone. Do tego celu służy **połączenie alternatywne**.



Rys. 4. Połączenie alternatywne

Dla przykładu z rysunku 4, układ przejdzie do kroku k 4 wtedy gdy spełniony zostanie jeden z warunków: p 1, p 2 lub p 3.

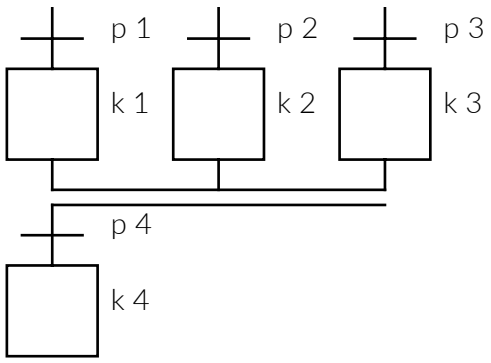
Język SFC zakłada możliwość współbieżnego (równoległego) przetwarzania sekwencji. Uruchomienie równoległych sekwencji wymaga użycia **rozgałęzienia równoległego**.



Rys. 5. Rozgałęzienie równoległe

Spełnienie warunku p 1 spowoduje aktywację wszystkich elementów kroku (k 2, k 3, i 4), które dołączone są do rozgałęzienia równoległego.

Złączenie sekwencji równoległych realizowane jest przy pomocy operacji **połączenia równoległego**.

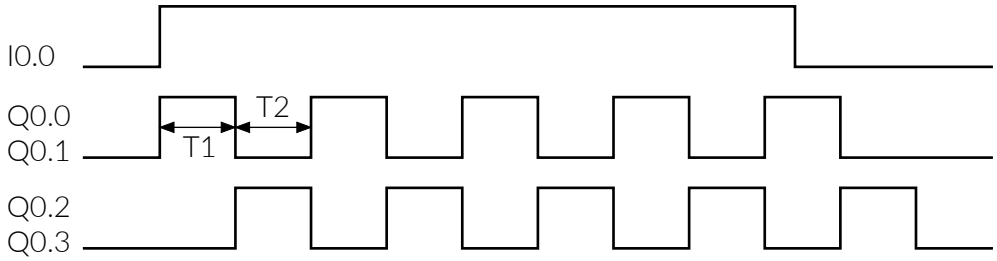


Rys. 6. Połączenie równoległe

Dla sekwencji z rysunku 6, układ przejdzie do kroku k 4 tylko wtedy gdy spełniony zostanie warunek p 4. Operacja połączenia równoległego pełni rolę operacji synchronizacji.

Przykład 1

Napisz program realizujący sekwencję połączania wyjść przedstawioną na rysunku 7.



Rys. 7. Generator fali prostokątnej

Po załączeniu wejścia I0.0, układ powinien naprzemiennie załączać i wyłączać wyjścia Q0.0, Q0.1, Q0.2 oraz Q0.3. Czas trwania załączenia wyjść Q0.0 i Q0.1, określony jest czasem pracy timera T1. Czas trwania załączenia wyjść Q0.2 i Q0.3, określony jest czasem pracy timera T2. Uwaga po wyłączeniu wejścia I0.0, układ powinien dokończyć sekwencję.

Układ może znaleźć się w trzech stanach:

STAN0 - inicjacja, zerowy stan układu, realizowane są instrukcje z kroku K0, czyli wyłączone są wyjścia Q0.0, Q0.1, Q0.2 oraz Q0.3.

STAN1 - stan pierwszy, realizowane są instrukcje z kroku K1, czyli załączone są wyjścia Q0.0 i Q0.1 oraz wyłączone wyjścia Q0.2 i Q0.3.

STAN2 - stan drugi, realizowane są instrukcje z kroku K2, czyli załączone są wyjścia Q0.2 i Q0.3 oraz wyłączone wyjścia Q0.0 i Q0.1.

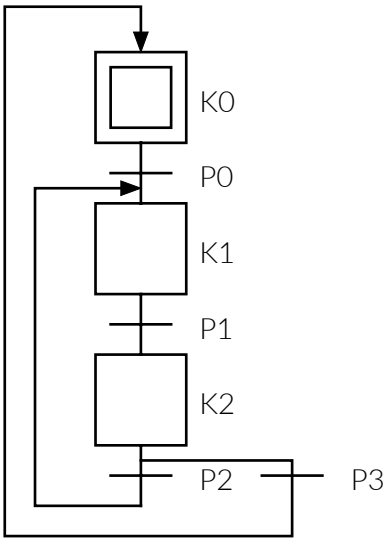
Zdefiniowane są przejścia pomiędzy krokami:

P0 - przejście z kroku K0 do K1, realizowane gdy zmieni się stan wejścia I0.0 z 0 na 1.

P1 - przejście z kroku K1 do K2, realizowane gdy timer T1 skończy odmierzać czas.

P2 - przejście z kroku K2 do K1, realizowane gdy timer T2 skończy odmierzać czas i wejście I0.0 jest w stanie wysokim.

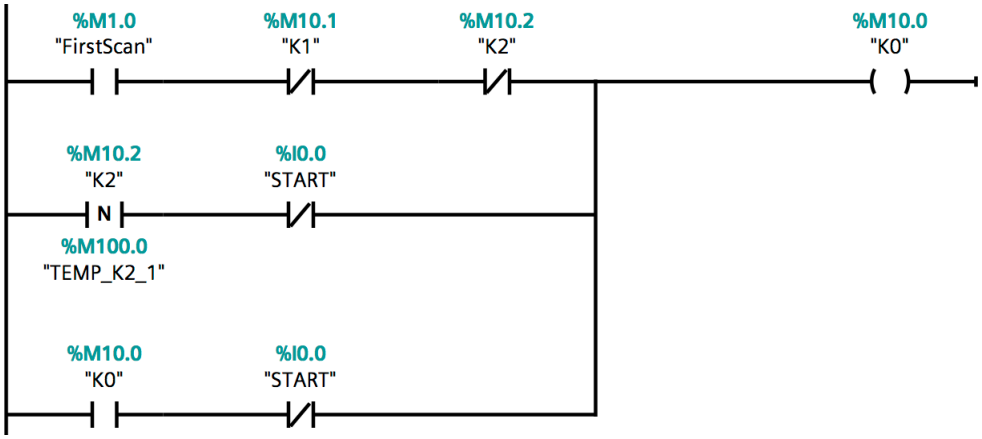
P3 - przejście z kroku K2 do K0, realizowane gdy timer T2 skończy odmierzać czas i wejście I0.0 jest w stanie niskim.



Rys. 8. Schemat SFC

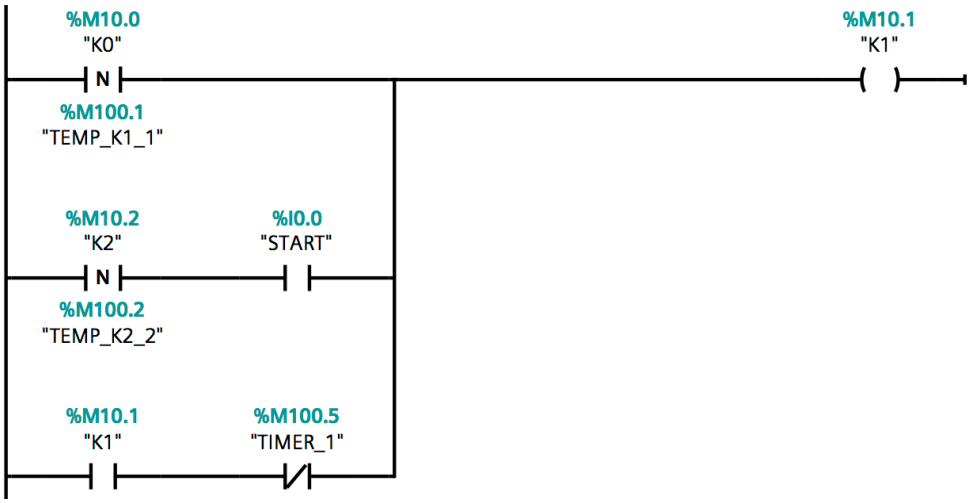
Implementacja w języku drabinkowym (LAD).

Siatka 1 - Sekwencer, krok K0.



Algorytm siatki 1 określa kiedy program znajduje się w kroku 0 (K0). Pierwsza gałąź uaktywnia element K0, tylko podczas pierwszej aktywacji siatki (FirstScan) a więc po uruchomieniu programu i wtedy gdy program nie znajduje się ani w pierwszym (K1) ani drugim (K2) kroku. Gałąź ta umożliwia zainicjowanie układu sekwencyjnego czyli wprowadzenie go do kroku K0 po uruchomieniu sterownika. W trakcie kolejnych aktywacji siatki, pierwsza gałąź jest stale nieaktywna a aktywacja elementu K0 wymaga aktywacji drugiej lub trzeciej gałęzi siatki. Druga gałąź siatki, definiuje warunek przejścia z kroku 2 do kroku 0. Jest to warunek P3 z rysunku 8. Jest on spełniony wtedy gdy zakończył się krok 2 i przełącznik START (I0.0) jest wyłączony. Aktywacja drugiej gałęzi ma charakter impulsowy. Gałąź trzecia realizuje funkcję podtrzymania aktywnego kroku 0. Wraz z załączeniem przełącznika START podtrzymanie jest przerywane (warunek przejścia P0) i K0 zmienia stan z wysokiego na niski.

Siatka 2 - Sekwencer, krok K1.



Algorytm siatki 2 określa kiedy program znajduje się w kroku 1 (K1). Pierwsza gałąź jest załączana wtedy gdy program wychodzi z kroku 0, generowany jest wtedy impuls, który podawany jest na element K1. Druga gałąź, definiuje warunek przejścia do kroku 1 z kroku 2. Jest to warunek P2 z rysunku 8, który jest spełniony wtedy gdy program wyszedł z kroku 2 i załączony jest przełącznik START. Podobnie jak w przypadku pierwszej gałęzi, aktywacja drugiej gałęzi ma charakter impulsowy. Trzecia gałąź realizuje funkcję podtrzymania wysokiego stanu elementu K1 przez czas pracy timera 1. Po tym czasie podtrzymanie jest przerywane i program wychodzi z kroku 1. Jest to warunek przejścia P1 z rysunku 8.

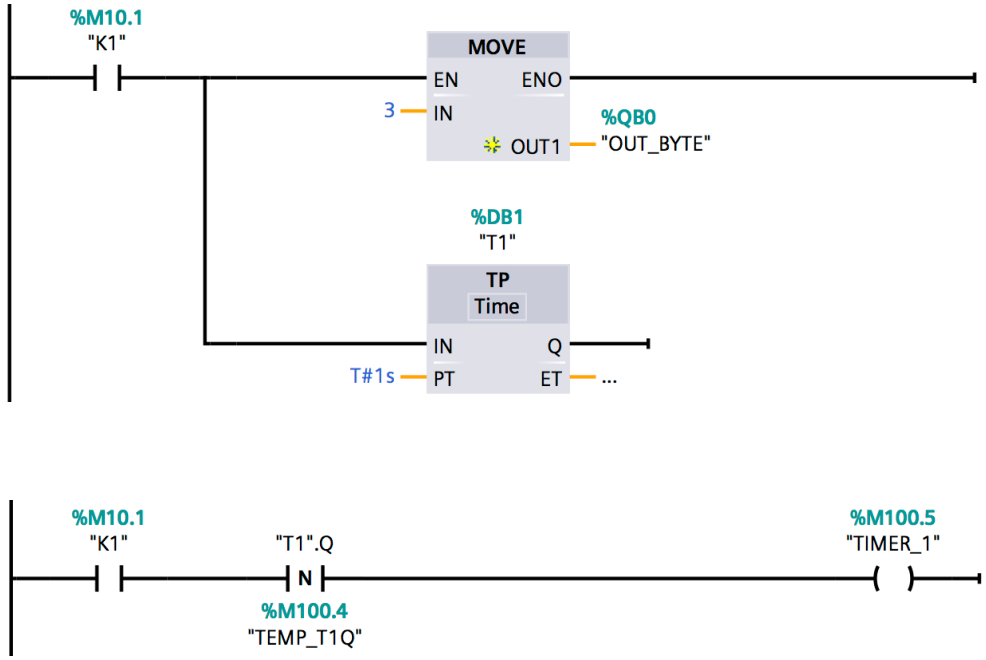
Siatka 3 - Sekwencer, krok K2.

Algorytm siatki 3 określa kiedy program znajduje się w kroku 2 (K2). Pierwsza gałąź jest załączana wtedy gdy program wychodzi z kroku 1. Generowany jest wtedy impuls, który podawany jest na element K2. Druga gałąź realizuje funkcję podtrzymania wysokiego stanu elementu K2 przez czas pracy timera 2. Po tym czasie podtrzymanie jest przerywane i program wychodzi z kroku 2.

Siatka 4 - Akcje w kroku 0.

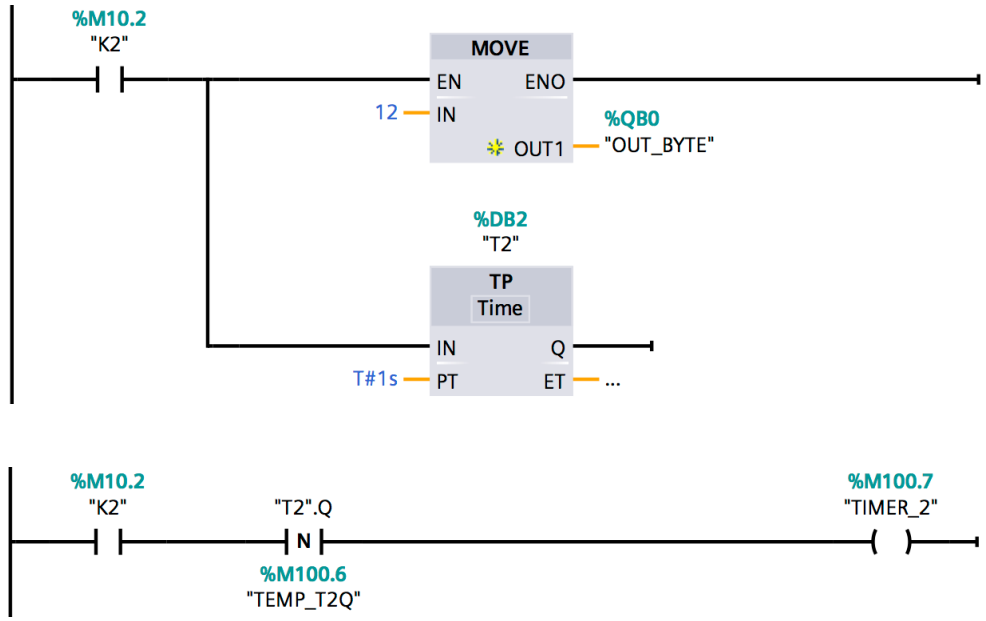
Algorytm siatki 4 określa działania które realizowane są wtedy gdy program znajduje się w kroku 0. W tym kroku zerowane są wszystkie bity portu QB0.

Siatki 5 i 6 - Akcje w kroku 1.



Algorytm zrealizowany w siatkach 5 i 6 określa działania jakie wykonywane są wtedy gdy program znajduje się w kroku 1. W siatce 5, program ustawia pierwsze dwa bity portu QB0 (pozostałe są zerowane) oraz uruchamia timer 1 (T1). W siatce 6, program sprawdza czy timer 1 zakończył pracę. Jeżeli tak to generowany jest impuls, który podawany jest na element TIMER_1. Impuls ten przerywa podtrzymanie stanu wysokiego elementu K1 (siatka 2) a tym samym kończy krok 1.

Siatki 7 i 8 - Akcje w kroku 2.



Algorytm realizowany w siatkach 7 i 8 określa działania jakie wykonywane są wtedy gdy program znajduje się w kroku 2. W siatce 7, program ustawia 3 i 4 bit portu QB0 (pozostałe bity są zerowane)

oraz uruchamia timer 2 (T2) . W siatce 8, program sprawdza czy timer 2 skończył pracę. Jeżeli ta to generowany jest impuls, który podawany jest na element TIMER_2. Impuls ten przerywa podtrzymanie stanu wysokiego na elementu K2 (siatka 3) a tym samym kończy krok 2.

Zadanie 1

Rozbudować program z przykładu 1 tak aby możliwe było wyłączenie sekwencji przełączenia wyjść układu zarówno w kroku pierwszym jak i w drugim. Określić stany układu, zdefiniować kroki oraz warunki przejścia pomiędzy nimi, utworzyć schemat SFC i na jego podstawie utworzyć program w języku drabinkowym. Uruchomić program i sprawdzić jego działanie.

Zadanie 2

Zaprojektować i uruchomić układ sterowania świateł na skrzyżowaniu.

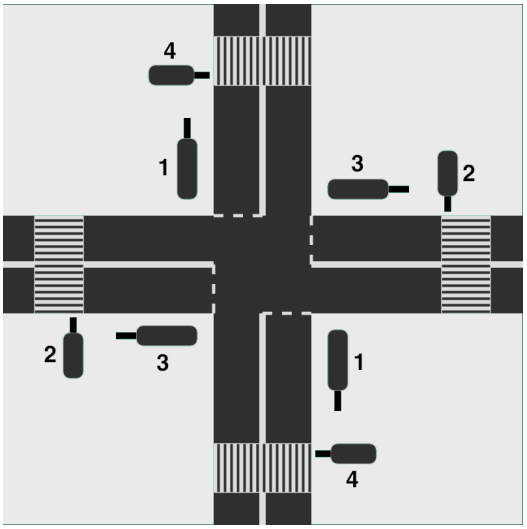


Tabela sekwencji zmiany świateł:

Sygnalizator 1	Sygnalizator 2	Sygnalizator 3	Sygnalizator 4
żółty		żółty	
czerwony	czerwony	czerwony	czerwony
czerwony	czerwony	czerwony / żółty	czerwony
czerwony	czerwony	zielony	zielony
czerwony	czerwony	zielony	zielony (pulsowanie)
czerwony	czerwony	żółty	czerwony
czerwony	czerwony	czerwony	czerwony
czerwony / żółty	czerwony	czerwony	czerwony
zielony	zielony	czerwony	czerwony
zielony	zielony (pulsowanie)	czerwony	czerwony
żółty	czerwony	czerwony	czerwony

Oświetlenie skrzyżowania sterowane jest przy pomocy pierwszych sześciu bitów portu wyjścia QB0.

Sprawdzić jaka kombinacja bitów portu QB0 uaktywnia daną kombinację sygnalizatorów (wiersz tabeli sekwencji zmiany świateł). Określić stany w których może znajdować się układ, zdefiniować kroki i warunki przejścia pomiędzy nimi. Utworzyć schemat SFC, na jego podstawie utworzyć program w języku drabinkowym. Uruchomić program i sprawdzić jego działanie.

Uwaga. Czas trwania danej kombinacji świateł na skrzyżowaniu, dobrać samodzielnie.