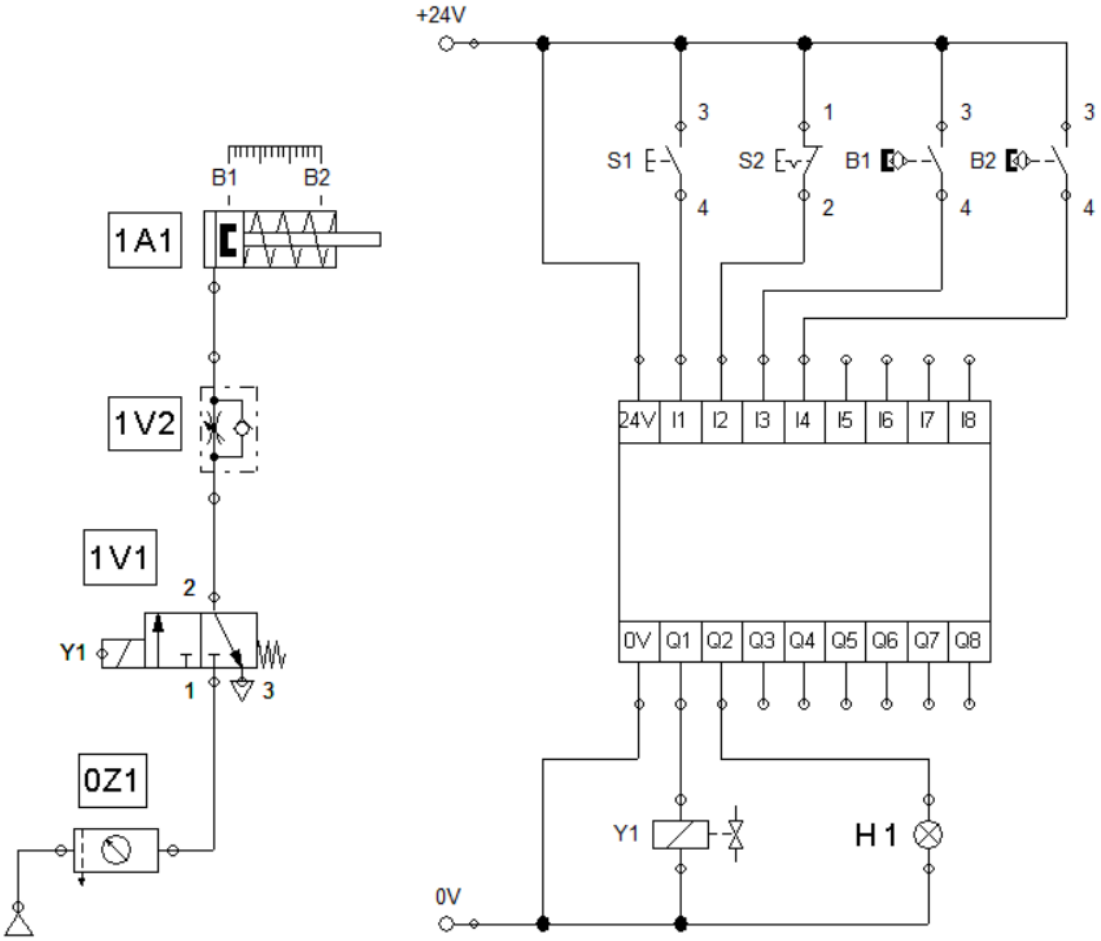
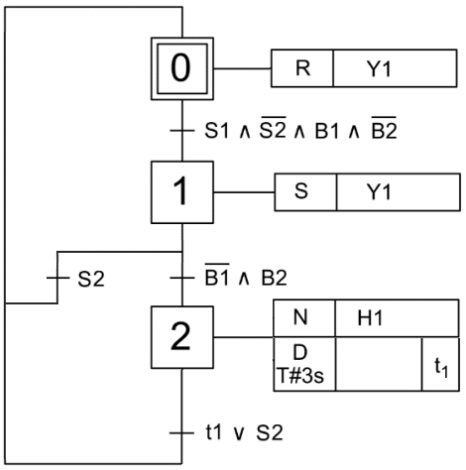
ZAPIS TRANZYCJI Z ALGORYTMU SFC W PROGRAMIE W JĘZYKU LD

Schemat części pneumatycznej i elektrycznej układu mechatronicznego

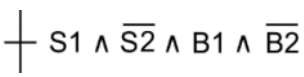


Graficzny opis pracy układu mechatronicznego za pomocą algorytmu SFC



**tranzycie w algorytmie zostały zapisane z zastosowaniem logiki, w której np. S2 oznacza przycisk naciśnięty, a $\overline{S2}$ oznacza przycisk nienaciśnięty, niezależnie od tego, czy posiada on zestyk NO czy NC*

1. Na początku należy zinterpretować zapis danej tranzycji (warunku przejścia) w algorytmie SFC. Dla przykładu weźmy pod uwagę tranzycję między krokiem 0 a krokiem 1:



S1 i S2 to przyciski, a B1 i B2 to czujniki, co widać po symbolach na schemacie elektrycznym. Zapis bez negacji, czyli S1, B1 oznacza, że do spełnienia tego warunku przejścia przycisk S1 należy nacisnąć, a czujnik B1 musi zostać aktywowany.
Zapis z negacją, czyli $\overline{S2}$, $\overline{B2}$ oznacza, że do spełnienia tego warunku przejścia przycisk S2 nie może być naciśnięty, a czujnik B2 nie może być aktywowany.
Dodatkowo w warunkach przejść mogą występować symbole oznaczające związek logiczny między stanami poszczególnych elementów. W analizowanym warunku mamy symbol ∧ oznaczający funkcję logiczną AND (i). **Zatem interpretacja zapisu powyższej tranzycji jest następująca: naciśnięty przycisk S1 i nienaciśnięty przycisk S2 i aktywowany czujnik B1 i nieaktywowany czujnik B2.**

UWAGA:
Funkcja AND (i) może być w zapisie tranzycji oznaczona symbolami: ∧ ∩ •
Funkcja OR (lub) może być w zapisie tranzycji oznaczona symbolami: ∨ ∪ +

2. Po zinterpretowaniu zapisu tranzycji przechodzimy do schematu elektrycznego i symulujemy spełnienie analizowanego warunku przejścia. Zatem naciskamy przycisk S1, nie naciskamy przycisku S2, aktywujemy czujnik B1 i nie aktywujemy czujnika B2. Naciśnięty przycisk i aktywowany czujnik zmieniają stan swoich zestyków: zestyk normalnie otwarty zamknie się, a zestyk normalnie zamknięty otworzy się. Przycisk, który nie jest naciskany i czujnik, który nie jest aktywowany nie zmieniają stanu swoich zestyków: zestyk normalnie otwarty pozostanie otwarty, a zestyk normalnie zamknięty pozostanie zamknięty. **Zatem w analizowanym przypadku spełnienie warunku przejścia spowoduje, że zestyk przycisku S1 zamknie się, zestyk przycisku S2 pozostanie zamknięty, zestyk czujnika B1 zamknie się, a zestyk czujnika B2 pozostanie otwarty.**

3. Po przeprowadzeniu symulacji należy ocenić poziom napięć na poszczególnych wejściach sterownika PLC w momencie spełnienia warunku przejścia. Ponownie przechodzimy do schematu elektrycznego i oceniamy, czy po spełnieniu warunku przejścia napięcie zasilające z zacisku +24V dociera do poszczególnych wejść sterownika. Pod uwagę bierzemy tylko te wejścia, do których dołączone są elementy występujące w analizowanej tranzycji. W naszym warunku mamy elementy S1, S2, B1 i B2, więc pod uwagę musimy wziąć wejścia I1, I2, I3 i I4. **W momencie spełnienia warunku przejścia zestyki elementów S1, S2 i B1 są zamknięte, więc na wejściach I1, I2 i I3 pojawia się napięcie +24V. Mówimy wtedy, że wejścia I1, I2 i I3 są w stanie „1". Natomiast zestyk elementu B2 w momencie spełnienia warunku przejścia jest otwarty, więc na wejściu I4 nie występuje napięcie +24V (wejście I4 jest odcięte od napięcia zasilającego +24V przez otwarty zestyk czujnika B2). Mówimy wtedy, że wejście I4 jest w stanie „0".**

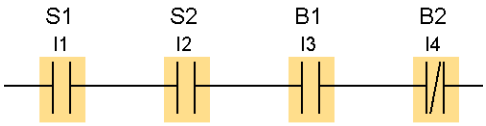
4. Gdy wiemy już które wejścia w momencie spełnienia analizowanego warunku przejścia są w stanie „1”, a które są w stanie „0” możemy podjąć decyzję o rodzaju zestyku programowego jaki należy wstawić w programie dla poszczególnych wejść. **Zasada wyboru zestyku programowego jest następująca:**
   - **jeżeli w wyniku przeprowadzonej symulacji oceniliśmy, że dane wejście sterownika jest w stanie „1”, to w zapisie warunku przejścia w programie należy zastosować zestyk programowy NO powiązany z tym wejściem,**
   - **jeżeli w wyniku przeprowadzonej symulacji oceniliśmy, że dane wejście sterownika jest w stanie „0”, to w zapisie warunku przejścia w programie należy zastosować zestyk programowy NC powiązany z tym wejściem.**

5. Gdy mamy już wybrane zestyki programowe należy zastanowić się nad sposobem ich połączenia, aby w programie realizowana była funkcja logiczna zapisana w tranzycji. **Jeżeli w tranzycji występuje funkcja AND (symbole: $\wedge$, $\cap$, $\bullet$) to zestyki programowe łączymy ze sobą szeregowo. Jeżeli w tranzycji występuje funkcja OR (symbole: $\vee$, $\cup$, $+$) to zestyki programowe łączymy ze sobą równolegle.**

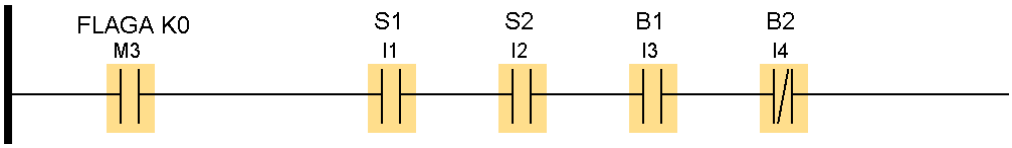
   Zatem prawidłowy zapis analizowanego warunku:

   $$S1 \wedge \overline{S2} \wedge B1 \wedge \overline{B2}$$

   wygląda następująco:

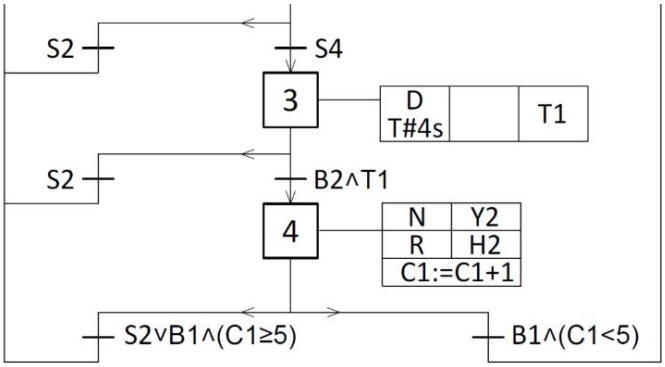


6. Należy oczywiście pamiętać, że przejście do danego kroku może nastąpić tylko w momencie, gdy układ znajduje się w kroku bezpośrednio poprzedzającym ten krok, czyli np. do kroku 1 można przejść tylko wtedy, gdy układ jest w kroku 0 (patrz algorytm SFC na pierwszej stronie) i spełniony jest warunek przejścia znajdujący się między tymi krokami. Zatem do zapisanego w programie warunku przejścia należy jeszcze dołączyć szeregowo zestyk programowy NO flagi kroku poprzedniego (w naszym przypadku jest to flaga kroku 0, czyli FLAGA K0).



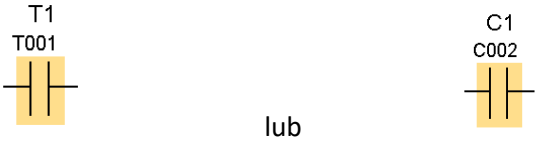
   Wyjątkiem od tej reguły jest zapis warunku natychmiastowego przerwania pracy układu i powrotu do kroku początkowego (kroku 0), np. po naciśnięciu przycisku S2. Wówczas nie ma znaczenia, w którym kroku aktualnie znajduje się układ – po naciśnięciu przycisku S2 układ natychmiast musi wrócić do kroku początkowego. W takim przypadku nie ma potrzeby uzupełniania warunku przejścia o flagę kroku poprzedniego.

7. Jeżeli w tranzycji występuje zmienna odnosząca się do stanu timera (przykładowe oznaczenia tego typu zmiennych w algorytmie SFC to „t”, „T”, „t1”, „T1” itp.) lub zmienna odnosząca się do stanu licznika (przykładowe oznaczenia tego typu zmiennych w algorytmie SFC to „k”, „n”, „C1” itp.), jak to widać poniżej na fragmencie algorytmu SFC:



to w programie dla tych zmiennych należy zastosować:

- zestyk programowy NO powiązany z użytym w programie blokiem timera lub licznika, jeżeli zapis tranzycji oznacza, że czas odmierzany przez timer minął (w tranzycji mamy wtedy np. T1, czyli zmienną bez negacji) lub licznik zliczył nastawioną ilość zdarzeń (w tranzycji mamy wtedy np. C1 $\geq$ 5, czyli zmienną o aktualnej wartości co najmniej równej nastawie licznika):



- zestyk programowy NC powiązany z użytym w programie blokiem timera lub licznika, jeżeli zapis tranzycji oznacza, że czas odmierzany przez timer jeszcze nie minął (w tranzycji mamy wtedy np. $\overline{T1}$, czyli zmienną z negacją) lub licznik nie zliczył jeszcze nastawionej ilości zdarzeń (w tranzycji mamy np. C1 < 5, czyli zmienną o aktualnej wartości mniejszej od nastawy licznika):

