8. Sterowniki PLC – język LD

Język *schematów drabinkowych LD* jest odpowiednikiem schematów drabinkowych (obwodowych) układów cyfrowych realizowanych w technologii stykowo-przekaźnikowej. Program w języku *LD* jest rysowany w postaci zbioru elementów łączonych w obwody. Symbole graficzne rozmieszczane są w obwodach w sposób podobny do szczebli schematów drabinkowych przekaźnikowych układów sterowania, a wykonanie programu polega na przepływie sygnału przez kolejne obwody programu.

8.1. Opis języka

Podstawowymi elementami obwodów języka *LD* są:

- połącznia (poziome i pionowe),
- styki i cewki.

Stany połączeń, styków oraz cewek są intepretowane jako wartości logiczne. Ze względu na to, że język, podobnie jak język *FBD*, pozwala na wykorzystywanie funkcji oraz bloków funkcjonalnych, bloki te muszą mieć co najmniej po jednym wejściu i wyjściu typu `BOOL`, żeby umożliwić przepływ prądu przez blok (w przypadku ich braku można korzystać z wejścia i wyjścia `EN` i `ENO`, zob. p. 6.3.3).

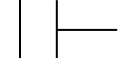
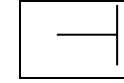
Podczas wykonania programu napisanego w języku *LD* obowiązują podobne zasady, jak podczas realizacji programu w *FBD*:

- wartość wyjściowa elementu obwodu wyznaczana jest dopiero po wyznaczeniu wszystkich wejść tego elementu,
- przetwarzanie elementu obwodu jest zakończone po wyznaczeniu wartości wszystkich jego wyjść,
- przetwarzanie obwodu jest zakończone po wyznaczeniu wyjść wszystkich jego elementów,
- domyślnie obwody wykonywane są od góry do dołu (porządek ten można zmienić z pomocą dodatkowych symboli języka).

8.1.1. Szyny prądowe

Obwód języka *LD* jest ograniczony z lewej i prawej strony *szynami prądowymi*. Szyny nie są właściwie elementami obwodu. Są one rysowane w postaci linii pionowych (tab. 8.1), przy czym tylko lewa szyna prądowa musi być narysowana, prawa szyna może być pomijana, w takim przypadku przyjmuje się, że występuje ona niejawnie. Szyny prądowe są „pod napięciem", a prąd przepływa w obwodzie od szyny lewej do prawej. Stan lewej szyny jest interpretowany jako `ON`, co odpowiada wartości zmiennej boolowskiej `1` (`TRUE`), stan prawej szyny nie jest definiowany.

Tab. 8.1. Szyny prądowe

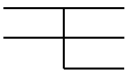
Symbol	*Opis*
	lewa szyna prądowa (z dołączonym połączeniem poziomym)
	prawa szyna prądowa (z dołączonym połączeniem poziomym)

8.1.2. Połączenia

Połączenie poziome rysowane w postaci linii poziomej przesyła stan elementu znajdującego się po lewej stronie połączenia na jego prawą stronę. *Połączenie pionowe* rysowane jest w postaci linii pionowej z dołączonym jednym lub kilkoma połączeniami poziomymi. Połączenie pionowe realizuje funkcję

logicznej alternatywy (OR) sygnałów poziomych znajdujących się po lewej stronie połączenia i ustawia jej wartość w połączeniach poziomych po prawej stronie.

Tab. 8.2. Rodzaje połączeń

Symbol	*Opis*
———	połączenie poziome
	połączenie pionowe (z dołączonymi połączeniami poziomymi, liczba połączeń poziomych może być dowolna, przykładowe połączenie realizuje alternatywę 2 sygnałów z lewej strony połączenia i ustawia obliczony wynik w 3 połączeniach po prawej stronie)

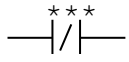
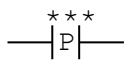
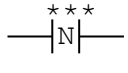
8.1.3. Styki

Styk realizuje funkcję *logicznej koniunkcji* (AND) sygnału z połączenia poziomego po jego lewej stronie ze stanem samego styku i ustawia jej wartość w połączeniu poziomym po jego prawej stronie. Stan styku zależy od jego rodzaju i wartości skojarzonej z nim zmiennej.

Na działanie styku można spojrzeć inaczej, biorąc pod uwagę jego stan. Z algebry Boole'a wiadomo, że elementem neutralnym koniunkcji jest 1 ($p \cdot 1 = p$), więc jeżeli stan styku jest równy 1, to można powiedzieć, że styk kopiuje wartość z połączenia po swojej lewej stronie do połączenia po prawej stronie. Z prawa elementów neutralnych ($p \cdot 0 = 0$) wynika zerowanie wartości połączenia po prawej, w przypadku gdy stan styku wynosi 0.

Styki, w odróżnieniu od cewek, nie modyfikują wartości skojarzonych z nimi zmiennych. Standardowe styki języka *LD* zostały zestawione w tab. 8.3.

Tab. 8.3. Rodzaje styków

Symbol	*Opis*
styki statyczne	
	*styk zwierny (normalnie otwarty)* stan styku jest określany jako 1, gdy skojarzona z nim zmienna ma wartość 1 (w przeciwnym przypadku stan jest równy 0)
	*styk rozwierny (normalnie zamknięty)* stan styku jest określany jako 1, gdy skojarzona z nim zmienna ma wartość 0 (w przeciwnym przypadku stan jest równy 1)
styki impulsowe	
	*styk wrażliwy na zbocze narastające* stan styku jest określany jako 1, gdy skojarzona z nim zmienna zmienia wartość z 0 na 1 (w przeciwnym przypadku stan jest równy 0)
	*styk wrażliwy na zbocze opadające* stan styku jest określany jako 1, gdy skojarzona z nim zmienna zmienia wartość z 1 na 0 (w przeciwnym przypadku stan jest równy 0)

gdzie (***) to nazwa zmiennej skojarzonej ze stykiem.

8.1.4. Cewki

Cewka przesyła bez zmiany sygnał z połączenia po jej lewej stronie do połączenia po prawej stronie, ustawiając jednocześnie wartość skojarzonej z nią zmiennej. Zmienna skojarzona z cewką otrzymuje wartość na podstawie wartości sygnału przepływającego przez cewkę oraz jej typ. Standardowe cewki języka *LD* zostały zestawione w tab. 8.4.

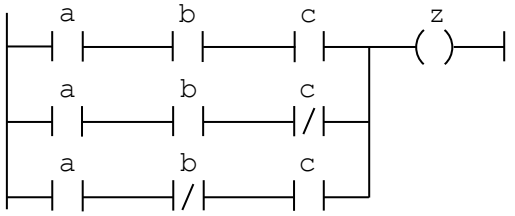
Tab. 8.4. Rodzaje cewek

Symbol	*Opis*
cewki zwykłe	
*** —( )—	*cewka* skojarzona z cewką zmienna dostaje wartość odpowiadającą stanowi połączenia po jej lewej stronie
*** —(/)—	*cewka negująca* skojarzona z cewką zmienna dostaje zanegowaną wartość stanu połączenia po jej lewej stronie
cewki zatrzaskiwane	
*** —(S)—	*cewka ustawiająca* skojarzona z cewką zmienna dostaje wartość `1`, jeżeli połączenie z jej lewej strony jest w stanie `1`, wyzerowanie zmiennej następuje dopiero po uaktywnieniu *cewki kasującej*
*** —(R)—	*cewka kasująca* skojarzona z cewką zmienna dostaje wartość `0`, jeżeli połączenie z jej lewej strony jest w stanie `1`, ustawienie zmiennej następuje dopiero po uaktywnieniu *cewki ustawiającej*
cewki impulsowe (reagują na zbocze sygnału połączenia z lewej strony)	
*** —(P)—	*cewka wrażliwa na zbocze narastające* skojarzona z cewką zmienna dostaje wartość `1` na czas jednego cyklu, jeżeli na połączeniu z jej lewej strony pojawiło się zbocze narastające
*** —(N)—	*cewka wrażliwa na zbocze opadające* skojarzona z cewką zmienna dostaje wartość `1` na czas jednego cyklu, jeżeli na połączeniu z jej lewej strony pojawiło się zbocze opadające

gdzie (***) to zmienna skojarzona z cewką.

8.1.5. Proste obwody w LD

Na rys. 8.1 przedstawiony został obwód realizujący funkcję logiczną zapisaną w postaci dysjunkcyjnej, która wcześniej była zapisana w postaci obwodu języka *FBD* (zob. rys. 7.1).



Rys.8.1. Obwód języka LD realizujący funkcję $z = a \cdot b \cdot c + a \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot c$

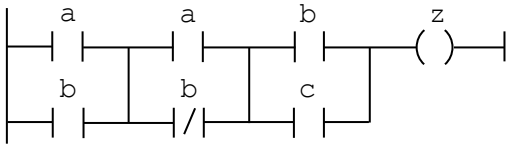
Trzy połączenia poziome dołączone do lewej szyny prądowej przesyłają stan tej szyny (stan `on`, który odpowiada wartości `1`) do trzech styków skojarzonych ze zmienną `a`. Po przetworzeniu tych

sygnałów przez styki (sygnały po prawej stronie styków odpowiadają koniunkcji $1 \cdot a = a$), przetwarzane są one przez styki skojarzone ze zmienną b. Po ich przetworzeniu, z prawej strony styków, otrzymuje się koniunkcje: $a \cdot b$, $a \cdot b$ i $a \cdot \overline{b}$, a po przetworzeniu przez styki skojarzone ze zmienną c koniunkcje: $a \cdot b \cdot c$, $a \cdot b \cdot \overline{c}$ i $a \cdot \overline{b} \cdot c$.

Połączenie pionowe przed symbolem cewki realizuje ostatecznie alternatywę wyznaczonych koniunkcji, a cewka przypisuje tę wartość zmiennej z, tzn.:

$$z = a \cdot b \cdot c + a \cdot b \cdot \overline{c} + a \cdot \overline{b} \cdot c.$$

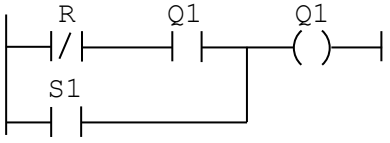
Rys. 8.2 przedstawia tę samą funkcję, przekształconą do postaci koniunkcyjnej (por. z zapisem funkcji w języku *FBD* na rys. 7.2).



Rys.8.2. Obwód języka LD realizujący funkcję $z = (a+b) \cdot (a+\overline{b}) \cdot (b+c)$

W tym przypadku dwa połączenia poziome przesyłają stan lewej szyny prądowej do styków skojarzonych ze zmiennymi a i b. Po przetworzeniu, po prawej stronie styków, otrzymuje się wartości zmiennych skojarzonych ze stykami, następnie połączenie pionowe realizuje alternatywę $(a + b)$ i przesyła ją dalej dwoma połączeniami po prawej stronie połączenia pionowego. Alternatywa przetwarzana jest następnie przez styk zwierny skojarzony ze zmienną a i styk rozwierny skojarzony ze zmienną b. Styki realizują funkcję koniunkcji, więc po przetworzeniu w górnej gałęzi otrzymuje się sygnał
$(a + b) \cdot a$, natomiast w dolnej $(a + b) \cdot \overline{b}$. W następnym kroku kolejne połączenie pionowe realizuje alternatywę, a więc po jego prawej stronie otrzymuje się sygnał $(a + b) \cdot a + (a + b) \cdot \overline{b}$, co można, wykorzystując prawo rozdzielności, zapisać w równoważnej postaci jako $(a + b) \cdot (a + \overline{b})$. Na koniec sygnał ten jest przetwarzany przez styki skojarzone ze zmiennymi b i c. W rezultacie w górnej gałęzi otrzymuje się sygnał $(a + b) \cdot (a + \overline{b}) \cdot b$, a w dolnej sygnał $(a + b) \cdot (a + \overline{b}) \cdot c$. Po przetworzeniu sygnałów przez ostatnie połączenie pionowe otrzymuje się sygnał $(a + b) \cdot (a + \overline{b}) \cdot (b + c)$, a cewka na końcu obwodu przypisuje jego wartość zmiennej z.

W przypadku układów sekwencyjnych pętle sprzężeń zwrotnych w języku *LD* nie mogą być rysowane w postaci jawnej, tak jak było to możliwe w języku *FBD*. Pętle realizowane są z pomocą zmiennych związanych z tymi sprzężeniami. Rysunek 8.3 przedstawia realizację *przerzutnika SR z dominującym wejściem ustawiającym* (por. z zapisem w języku *FBD* na rys. 7.3).



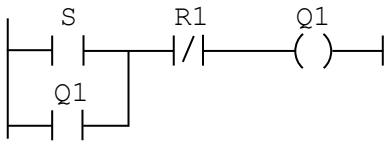
Rys.8.3. Przerzutnik SR z dominującym wejściem ustawiającym

$Q1 = \overline{R} \cdot Q1 + S1$

W następnych punktach omówione zostały przykładowe programy których realizacja w języku *FBD* została przedstawiona w punkcie poprzednim.

8.2. Programy z przerzutnikami

Przerzutniki w języku *LD* można budować, bazując na odpowiednich wyrażeniach logicznych. Realizacja *przerzutnika SR z dominującym wejściem ustawiającym* została pokazana na rys. 8.3, a *przerzutnika SR z dominującym wejściem zerującym* na rys. 8.4.

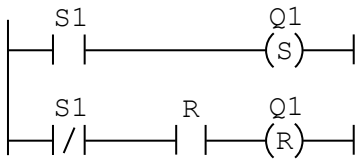
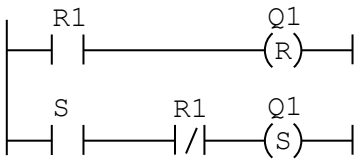
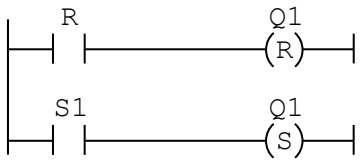
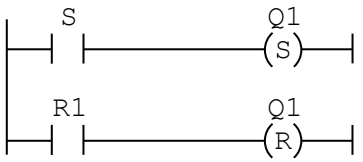


Rys.8.4. Przerzutnik SR z dominującym wejściem zerującym

$$Q1 = (S+Q1) \cdot \overline{R1}$$

Przerzutniki można również budować, wykorzystując *cewki zatrzaskiwane*: *ustawiającą* i *kasującą*. W tab. 8.5 przedstawiono przykładowe metody realizacji przerzutnika SR w wersji z dominującym wejściem ustawiającym i dominującym wejściem zerującym z użyciem cewek.

Tab. 8.5. Realizacje przerzutników SR w LD

SR z dominującym ustawianiem	*SR z dominującym zerowaniem*
	
	

Dodatkowo, podobnie jak w *FBD*, w języku *LD* można używać bloki funkcjonalne `SR` i `RS`, więc przykładowe programy z punktu 7.2 mogą być zbudowane bardzo podobnie.

8.2.1. Przykład 1a, 1b

W punkcie 7.2.1 rozważono zadanie sterowania pracą urządzenia za pomocą dwóch łączników bistabilnych. Zadanie rozwiązano, wykorzystując *przerzutnik SR z dominującym wejściem ustawiającym* z warunkami:

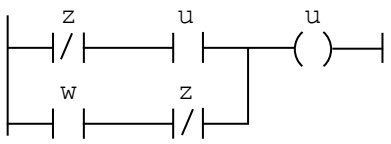
- na wejściu ustawiającym:

$$w \cdot \overline{z},$$

- na wejściu zerującym:

$$z.$$

W pierwszej wersji programu w języku *LD* przerzutnik został zbudowany tak jak to pokazano na rys. 8.3. Wyrażenia dla wejścia ustawiającego i zerującego wprowadzone zostały w miejsce oryginalnych sygnałów `S1` i `R`, sygnał sterujący pracą urządzenia zastąpił oryginalny sygnał wyjściowy przerzutnika `Q1`. Rozwiązanie w tej wersji zostało przedstawione na rys. 8.5. Zmienne skojarzone z wejściami i wyjściem sterownika są zadeklarowane tak jak w punkcie 7.2.1.



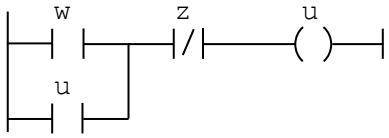
Rys.8.5. Sterowanie pracą urządzenia wersja 1

Z analizy wyrażenia logicznego reprezentowanego przez obwód języka *LD* przedstawiony na rys. 8.5 wynika, że styk rozwierny skojarzony ze zmienną z może być przeniesiony bezpośrednio przed symbol cewki. Na rys. 8.6 dodatkowo styk zwierny skojarzony ze zmienną w został przeniesiony do górnej gałęzi, a styk skojarzony ze zmienną u do dolnej gałęzi obwodu. Zamiana ta została wykonana, żeby pokazać, że prostszym rozwiązaniem rozważanego zadania byłoby wykorzystanie *przerzutnika SR z dominującym wejściem zerującym*. Obwód z rys. 8.6 odpowiada właściwie schematowi *przerzutnika SR* przedstawionemu na rys. 8.4 z warunkami:

- na wejściu ustawiającym:

  w,
- na wejściu zerującym:

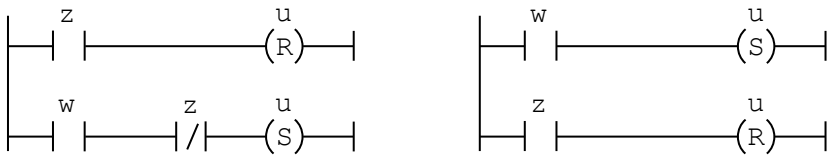
  z.



Rys.8.6. Sterowanie pracą urządzenia wersja 2

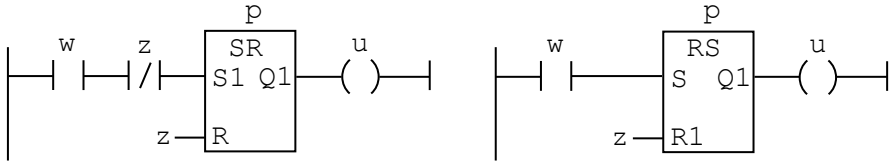
Rozwiązanie z rys. 8.6 odpowiada więc rozwiązaniu przedstawionemu w punkcie 7.2.2, gdzie w języku FBD przerzutnik SR z dominującym wejściem zerującym został wykorzystany do rozwiązania zadania.

Z opisu przedstawionego na początku punktu 8.2 wynika, że zadanie można również rozwiązać, wykorzystując *cewki zatrzaskiwane*. Podstawiając warunki dla wejść ustawiającego i zerującego przerzutników, których realizacje zostały zebrane w tab. 8.5, można uzyskać równoważne obwody realizujące rozważane zadanie (rys. 8.7).



Rys.8.7. Sterowanie pracą urządzenia wersja z cewkami zatrzaskiwanymi

W języku *LD* można wykorzystywać również bloki przerzutników, więc program sterujący pracą urządzenia może być zrealizowany w podobny sposób jak w punktach 7.2.1 i 7.2.2 z wykorzystaniem *przerzutnika SR z dominującym wejściem ustawiającym* i *przerzutnika SR z dominującym wejściem zerującym* (rys. 8.8).



Rys.8.8. Sterowanie pracą urządzenia wersja z blokami przerzutników

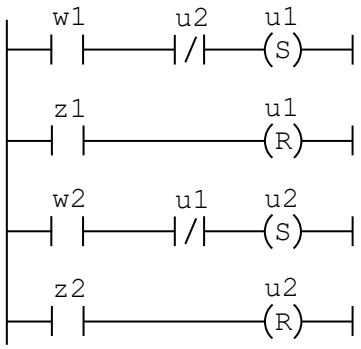
Kolejne przykładowe problemy, do których rozwiązania niezbędne są przerzutniki SR można rozwiązać, wykorzystując wszystkie omówione tutaj podejścia. W większości przykładów tego typu pokazane zostanie jednak jedno rozwiązanie, w którym przerzutniki będą budowane z wykorzystaniem cewek zatrzaskiwanych.

8.2.2. Przykład 1c

W punkcie 7.2.3 został przedstawiony przykład sterowania pracą dwóch urządzeń za pomocą dwóch zestawów łączników bistabilnych umożliwiających włączanie i wyłączanie urządzeń z zastrzeżeniem możliwości ich jednoczesnej pracy. Zadanie rozwiązano, wykorzystując dwa *przerzutniki SR z dominującym wejściem zerującym*. Warunki ustawiane na wejściach przerzutników zostały zebrane w tab. 8.6. Na rys. 8.9 pokazano obwody języka *LD* realizujące przyjęte zasady sterowania. Zmienne są zadeklarowane w taki sam sposób jak w punkcie 7.2.3.

Tab. 8.6. Warunki w przykładzie 1c

Przerzutnik	*Wejście ustawiające*	*Wejście zerujące*
1	$w1 \cdot \overline{u2}$	z1
2	$w2 \cdot \overline{u1}$	z2



Rys.8.9. Sterowanie pracą dwóch urządzeń wersja 1

8.3. Programy z detekcją zbocza

W języku *LD* można wykorzystywać bloki funkcjonalne, a więc przykładowe programy z punktu 7.3 wykorzystujące *detektor zbocza narastającego* mogą być zbudowane bardzo podobnie w języku *LD*. Dodatkowo norma IEC 61131 opisuje *styki impulsowe* reagujące na zbocze sygnału połączenia z lewej strony, które również mogą być wykorzystane do rozwiązania przykładowych zadań.

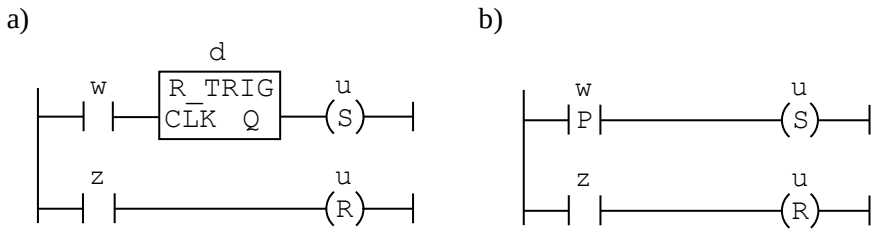
8.3.1. Przykład 2a

W przykładzie przedstawionym w punkcie 7.3.1 urządzenie jest uruchamiane i zatrzymywane za pomocą dwóch łączników bistabilnych: włączającego i wyłączającego. *Detektor zbocza narastającego* został wykorzystany do blokowania włączania urządzenia, w przypadku gdy po włączeniu obydwu łączników zwolniony został łącznik wyłączający. W pierwotnej wersji programu (zob. p. 7.2.1 i 7.2.2) urządzenie w takiej sytuacji było włączane.

W programie pokazanym na rys. 7.7 zadanie rozwiązano, wykorzystując *przerzutnik SR z dominującym wejściem zerującym*. Warunki ustawiane na wejściach przerzutnika zostały zebrane w tab. 8.7. Na rys. 8.10 pokazano obwody języka *LD* z przerzutnikiem zrealizowanym z pomocą *cewek zatrzaskiwanych*, deklaracje zmiennych nie zostały przedstawione – przyjęto, że zadeklarowane są w taki sam sposób jak w punkcie 7.3.1.

Tab. 8.7. Warunki w przykładzie 2a

Wejście ustawiające	*Wejście zerujące*
↑w	z



Rys.8.10. Sterowanie pracą urządzenia wersja 3
a) z detektorem zbocza, b) ze stykiem impulsowym

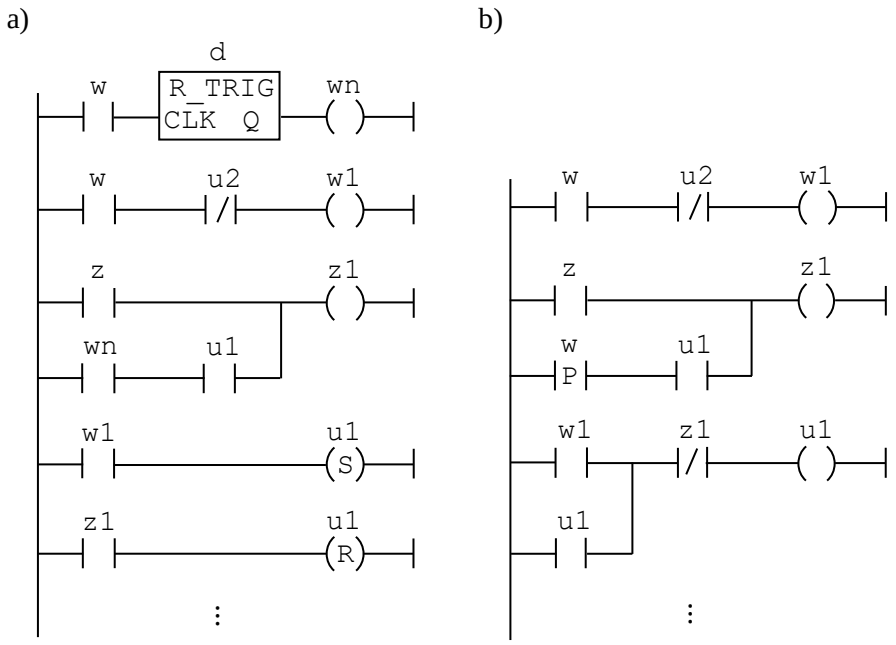
8.3.2. Przykład 2b

W punkcie 7.3.2 detekcję zbocza wykorzystano do sterowania pracą dwóch urządzeń, które mają jeden zestaw łączników wykorzystywanych do włączania i wyłączania. Pierwsze włączenie łącznika włączającego uruchamia urządzenie pierwsze, po wyłączeniu i ponownym włączeniu łącznika urządzenie pierwsze jest zatrzymywane, a uruchamiane jest urządzenie drugie, operacja zatrzymywania i uruchamiania jest powtarzana dla urządzenia pierwszego i drugiego. Wyłączanie urządzeń następuje po włączeniu łącznika wyłączającego.

Program w języku *LD* został napisany podobnie jak w punkcie 7.3.2 z wykorzystaniem dwóch *przerzutników SR z dominującym wejściem zerującym*. W programie wykorzystano zmienne pomocnicze (`w1`, `z1`, …), którym przypisano wartości na podstawie omówionych w punkcie 7.3.2 warunków dla wejść przerzutników (tab. 8.8). Przedstawione zostały dwie wersje rozwiązania (rys. 8.11): w pierwszej przerzutniki zbudowane zostały z pomocą *cewek zatrzaskiwanych*, w drugiej wykorzystany został schemat przedstawiony na rys. 8.4, a wyrażenia dla wejść przerzutników wprowadzone zostały w miejsce oryginalnych sygnałów `S` i `R1`.

Tab. 8.8. Warunki w przykładzie 2b

Przerzutnik	*Wejście ustawiające*	*Wejście zerujące*
1	$w \cdot \overline{u2}$	$z + \uparrow w \cdot u1$
2	$w \cdot \overline{u1}$	$z + \uparrow w \cdot u2$



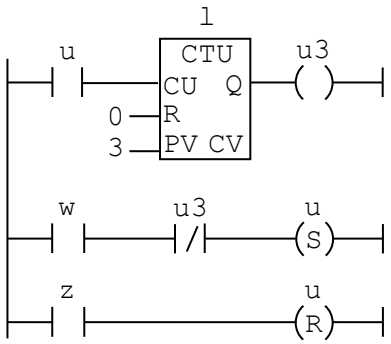
Rys.8.11. Sterowanie pracą dwóch urządzeń wersja 2
a) z detektorem zbocza, b) ze stykami impulsowymi

W obydwu wersjach pokazano jedynie fragment programu pozwalający na sterowanie pracą pierwszego urządzenia, obsługa drugiego urządzenia może być zrealizowana analogicznie. Zmienne pomocnicze `w1`, `z1`, … przechowują wartości wyrażeń określających warunki włączenia i zatrzymania urządzeń, zwiększając czytelność obydwu programów. Dodatkowo warto zauważyć, że pierwsza wersja programu zapisana bez użycia zmiennych pomocniczych nie pozwalałaby na sterowanie pracą urządzeń zgodnie z przyjętymi zasadami. Gdyby obwody ustawiające zmienne pomocnicze zakończone były cewkami zatrzaskiwanymi ustawiającymi i zerującymi wartości zmiennych `u1`, `u2`, to włączenie urządzenia nie byłoby możliwe – po ustawieniu zmiennej w obwodzie pierwszym byłaby ona natychmiast zerowana w obwodzie drugim. W tego typu przypadkach, gdy warunek w drugim obwodzie jest uzależniony od wartości ustawianej zmiennej, prawidłowe działanie programu zapewnia odpowiednia zmienna pomocnicza ustawiona przed zmianą wartości zmiennej w obwodzie pierwszym. W rozwiązaniach z blokami funkcjonalnymi `SR` i `RS` oraz w przypadku użycia schematów z rys. 8.3 czy 8.4 zmienne pomocnicze nie są niezbędne do prawidłowego działania programu – zmienna sterująca pracą urządzenia otrzymuje wartość za pomocą jednego obwodu.

8.4. Program z licznikiem

8.4.1. Przykład 3

W przedstawionym w punkcie 7.4.1 programie licznik został wykorzystany do blokowania włączania urządzenia, umożliwiając maksymalnie jego trzykrotne uruchomienie. Program ten można podobnie zapisać w języku *LD*. W rozwiązaniu przedstawionym na rys. 8.12 wykorzystana została pomocnicza zmienna `u3`, której wartość odpowiada stanowi licznika.

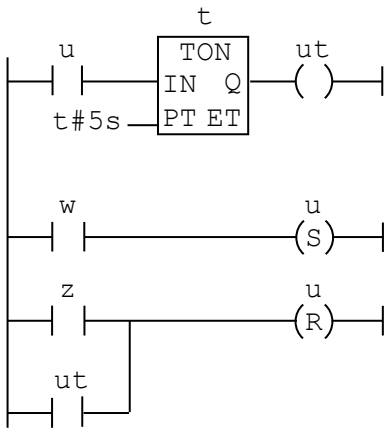


Rys.8.12. Sterowanie pracą urządzenia wersja 4

8.5. Programy z timerami

8.5.1. Przykład 4a

W punkcie 7.5.1 w programie pozwalającym na włączanie i wyłączanie urządzenia dwoma łącznikami dodano funkcję automatycznego wyłączania po 5 sekundach od uruchomienia urządzenia. Na rys. 8.13 pokazane zostało rozwiązanie w języku *LD*. Zmienna pomocnicza `ut`, której wartość odpowiada stanowi timera, wykorzystywana jest w obwodzie odpowiedzialnym za wyłączanie urządzenia. Podobnie jak w rozwiązaniu z rys. 8.11a ustawienie wartości zmiennej pomocniczej przed zmianą wartości zmiennej `u` w obwodzie zakończonym cewką ustawiającą, zapewnia działanie programu zgodne z przyjętymi założeniami.

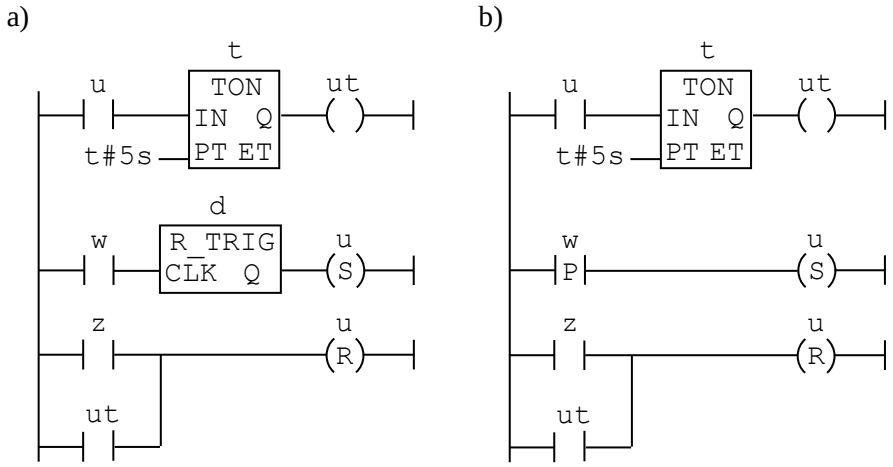


Rys.8.13. Sterowanie pracą urządzenia wersja 5

8.5.2. Przykład 4b

W programie opisanym w punkcie 7.5.2 urządzenie jest uruchamianie w reakcji na zbocze narastające sygnału związanego z łącznikiem włączającym. Takie rozwiązanie blokuje ponowne uruchomienie urządzenia po jego wyłączeniu przy włączonym łączniku włączającym.

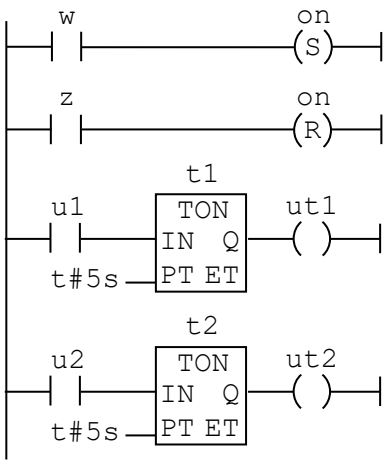
Detekcję zbocza można zrealizować podobnie jak w punkcie 8.3, wykorzystując blok funkcjonalny `R_TRIG` lub *styk impulsowy*. Obydwie wersje rozwiązania zostały pokazane na rys. 8.14.



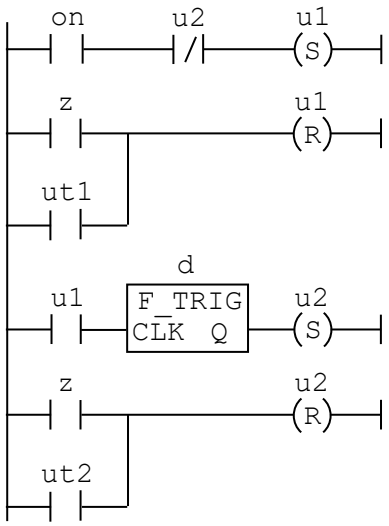
Rys.8.14. Sterowanie pracą urządzenia wersja 6

8.5.3. Przykład 4c

W punkcie 7.5 timery wykorzystane zostały również do kaskadowego włączania dwóch urządzeń. Obydwa urządzenia stanowią w przykładzie niepodzielny układ sterowany z pomocą dwóch łączników – włączającego i wyłączającego. Po włączeniu łącznika włączającego uruchamiane jest urządzenie pierwsze, po 5 sekundach urządzenie jest wyłączane i uruchamiane jest urządzenie drugie, proces wyłączania i uruchamiania urządzeń jest powtarzany do momentu włączenia łącznika wyłączającego. Program w języku *LD* realizujący rozważane zadanie został przedstawiony na rys. 8.15 i 8.16.



Rys.8.15. Sterowanie pracą dwóch urządzeń wersja 3 (fragment 1)

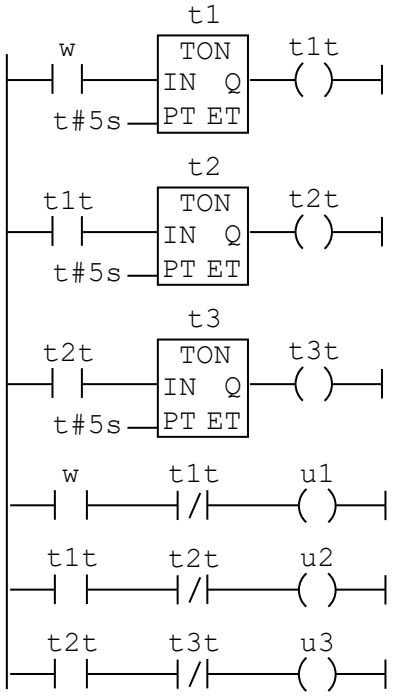


Rys.8.16. Sterowanie pracą dwóch urządzeń wersja 3 (fragment 2)

W pierwszym fragmencie programu (rys. 8.15), w celu uproszczenia postaci wyrażenia włączającego urządzenie pierwsze, wykorzystany został *przerzutnik SR* ustawiający zmienną pomocniczą `on` określającą stan układu (układ włączony albo wyłączony), a do kontroli czasu aktywności urządzeń wykorzystane zostały dwa timery ustawiające zmienne pomocnicze odpowiadające stanom timerów. W drugim fragmencie programu (rys. 8.16) odbywa się właściwe uruchamianie i wyłączanie urządzeń.

8.5.4. Przykład 4d

W przykładzie, podobnie jak w punkcie 7.5.4, pokazane zostanie rozwiązanie zadania kaskadowego włączania trzech urządzeń. Proces uruchamiania jest inicjowany po włączeniu łącznika bistabilnego, przerywanie pracy urządzeń następuje po wyłączeniu tego łącznika.

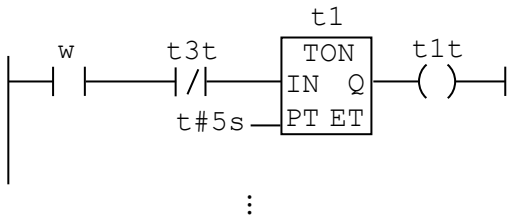


Rys.8.17. Sterowanie pracą trzech urządzeń wersja 1

Program w tej wersji do ponownego rozpoczęcia cyklu pracy urządzeń wymaga wyłączenia i włączania łącznika włączającego. Wyłączenie łącznika zeruje wejście i w konsekwencji wyjście pierwszego timera, co dalej prowadzi do wyzerowania wejścia i wyjścia kolejno drugiego i trzeciego timera. Włączenie łącznika powoduje uruchomienie pierwszego timera, który po 5 sekundach ustawia swoje wyjście, co uruchamia timer drugi, a po kolejnych 5 sekundach timer trzeci. Ostatnie trzy obwody programu ustalają stany sygnałów sterujących w zależności od stanu łącznika włączającego i od stanu odpowiednich timerów.

8.5.5. Przykład 4e

Podobnie jak w punkcie 7.5.5, w celu wyeliminowania konieczności działań operatora niezbędnych do rozpoczęcia kolejnego cyklu pracy urządzeń w programie z przykładu poprzedniego należy zmodyfikować jedynie pierwszy obwód odpowiedzialny za pracę pierwszego timera.

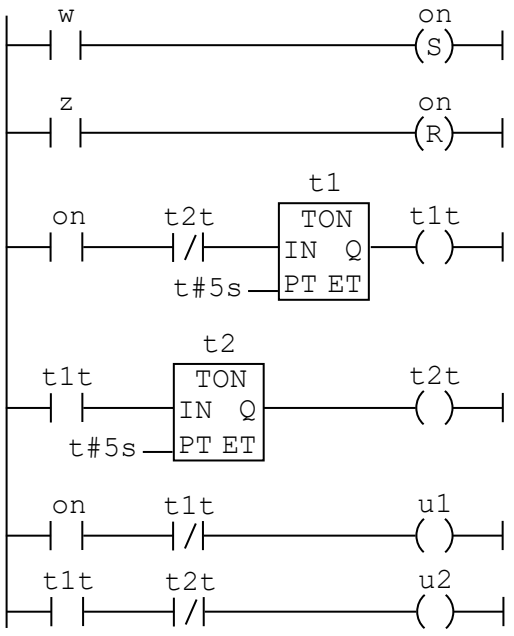


dalsza część programu jak na rys. 8.17

Rys.8.18. Sterowanie pracą trzech urządzeń wersja 2

8.5.6. Przykład 4f

Na koniec zadanie z przykładu 4c zostało rozwiązane z wykorzystaniem metody opisanej w punktach 8.5.4 i 8.5.5. Przedstawione rozwiązanie jest właściwie tłumaczeniem na język *LD* programu zaprezentowanego w punkcie 7.5.6.



Rys.8.19. Sterowanie pracą dwóch urządzeń wersja 4

8.6. Zadania

8.1. Zapisz programy z punktu 7.6 w języku LD.

8.2. Napisz program, którego zadaniem jest sterowanie ruchem siłownika dwustronnego działania. Siłownik powinien cyklicznie wykonywać ruchy pomiędzy skrajnymi położeniami sygnalizowanymi przez 2 czujniki, w które jest wyposażony. Ruch roboczy należy rozpocząć, jeżeli tłoczysko siłownika jest całkowicie wsunięte (sygnalizowane stanem wysokim pierwszego czujnika). Po całkowitym wysunięciu tłoczyska (sygnalizowane stanem wysokim drugiego czujnika) należy rozpocząć ruch powrotny.

8.3. Zmodyfikuj program z zadania 8.2 tak, aby siłownik mógł rozpocząć pracę również w sytuacji, gdy w chwili początkowej znajduje się pomiędzy położeniami skrajnymi (sygnały z obydwu czujników są niskie).

8.4. Zmodyfikuj program z zadania 8.2 tak, aby siłownik rozpoczynał pracę po włączeniu układu przyciskiem START i kończył po wciśnięciu przycisku STOP.

8.5. Zmodyfikuj program z zadania 8.4 tak, aby po całkowitym wysunięciu tłoczyska siłownik pozostawał w tym stanie przez określony wcześniej czas. Podobnie należy zatrzymać ruch siłownika po całkowitym wsunięciu tłoczyska.

8.6. Zmodyfikuj program z zadania 8.5 tak, aby siłownik wykonywał tylko określoną wcześniej liczbę cykli. Ponowne uruchomienie siłownika powinno być możliwe po ponownym wciśnięciu przycisku START.