2. Sterowniki PLC

Programowalny sterownik logiczny, PLC (od ang. *programmable logic controller*) – uniwersalne urządzenie mikroprocesorowe przeznaczone do sterowania pracą maszyny lub urządzenia technologicznego. PLC musi zostać dopasowany do określonego obiektu sterowania poprzez jego zaprogramowanie. Sterownik wyposaża się w odpowiednią liczbę układów wejściowych zbierających informacje o stanie obiektu i żądaniach obsługi oraz odpowiednią liczbę i rodzaj układów wyjściowych połączonych z elementami wykonawczymi, sygnalizacyjnymi lub transmisji danych.

Istnieją różne sposoby *pisania* programów na sterowniki PLC. Tak zapisany program przekształcany jest na tzw. język maszynowy i dopiero wtedy może być wykonany przez mikroprocesor. Sposoby (środowiska) programowania PLC:

- **LD (ladder diagram) logika drabinkowa** – schemat zbliżony do klasycznego rysunku technicznego elektrycznego
- FBD (function block diagram) – diagram bloków funkcyjnych, sekwencja linii zawierających bloki funkcyjne
- ST (structured text) tekst strukturalny – język zbliżony do Pascala
- IL (instruction list) lista instrukcji – rodzaj asemblera
- SFC (sequential function chart) sekwencyjny ciąg bloków – sekwencja bloków programowych z warunkami przejścia.

Logika drabinkowa jest szeroko stosowana do programowania sterowników PLC, w których wymagana jest sekwencyjna kontrola procesu lub operacji produkcyjnej. Logika drabinkowa jest przydatna w prostych, ale krytycznych systemach sterowania lub w przerobieniu starych obwodów przekaźników (*proszę przypomnieć sobie czym jest przekaźnik!*). Ponieważ programowalne sterowniki logiczne stały się bardziej wyrafinowane, stosuje się je również w bardzo złożonych systemach automatyki. Motywacją do przedstawienia logiki sterowania sekwencyjnego na schemacie drabinkowym było umożliwienie inżynierom i technikom produkcji oprogramowania bez dodatkowego szkolenia nauki języka programowania.

Implementacje logiki drabinkowej mogą mieć takie cechy, jak sekwencyjne wykonywanie i obsługa funkcji sterowania przepływem, które sprawiają, że analogia do sprzętu jest nieco niedokładna. Logikę drabinkową można traktować raczej jako język oparty na regułach niż język proceduralny.

Szczebel na drabinie stanowi regułę. Realizacja programu przez mikroprocesor polega na *odczytaniu* kolejno wszystkich *szczebli* drabiny od góry do dołu. Odczyt poszczególnych *szczebli* nastę-

puje bardzo szybko (nawet w mili- czy nanosekundach). Po zakończeniu odczytu mikroprocesor wraca na początek i ponownie wykonuje program, zapamiętując stan wyjść.

W logice drabinkowej występują trzy podstawowe oznaczenia: wejść (I), wyjść (Q) oraz markerów (M). Marker jest to zmienna (zapamiętująca wartość 1 lub 0) którą ustawia się jak wyjście lub odczytuje jak wejście. Po włączeniu sterownika wyjścia oraz markery ustawiane są na logiczne 0.

Rodzaj	Opis
I **I**nput (wejście)	W każdym sterowniku PLC mają takie samo oznaczenie, mogą być przypisywane **tylko** do symboli styków informują o stanie wejść na sterowniku. `--[ ]--` Wejście normalne (NO) `--[/]--` Wejście odwrócone (NC)
Q Output (wyjście)	W każdym sterowniku PLC mają takie samo oznaczenie, mogą być przypisywane **zarówno** do symboli cewek (wtedy ustawiają konkretne wyjście sterownika) jak i styków gdzie informują o stanie wyjść. `--( )--` Wyjście normalne (NO) `--(/)--` Wyjście odwrócone (NC)
M **M**arker	Inaczej zmienna wewnętrzna. Tym symbolem określa się zmienne wewnętrzne sterownika, wykorzystywane są jako cewki i styki. elementy pośrednie programu.

Markery oraz wyjścia

Wyjścia Q poza tym, że są fizycznie wyprowadzone ze sterownika i można je *gdzieś* podłączyć (do diody, przekaźnika, tranzystora, układu cyfrowego), można również wykorzystać jako informację wejściową. W tym celu wystarczy wykorzystać znacznik wejścia `--[ ]--`, i podpisać go symbolem wyjścia. Marker zachowuje się tak samo, z tą różnicą, że nie jest wyprowadzony ze sterownika – nie można go nigdzie podłączyć.

Odczyt *szczebla*

Szczeble odczytywane są od lewej. Lewą stronę drabinki możemy interpretować jako logiczną 1 i od niej zaczynamy. Idąc w prawo możemy spotkać skrzyżowanie lub jeden z symboli (I, M lub Q). Jeśli napotkamy wejście, to w zależności od jego typu (NO lub NC) oraz sygnału, po prawej stronie pojawi się 1 lub 0. Prosty przykład składający się z przycisku (podłączonego do wejścia I) oraz diody (podłączonej do wyjścia Q) oraz programu:

```
|----------[ ]----------------------------( )---|
|          I1                               Q    |
```

należy odczytywać tak: z lewej strony I1 jest 1, na wejściu I1 mogą pojawić się dwa stany: 0 lub 1.

Jeżeli przycisk nie zostanie wciśnięty (I1 = 0), *jedynka* z lewej nie może *przepłynąć* na stronę prawą, więc do Q dopływa 0 – dioda się nie świeci.

Jeżeli przycisk zostanie wciśnięty (I1 = 1), to do wyjścia Q dotrze 1 – dioda się zaświeci.

Przykład

Po uruchomieniu sterownika interpretujemy wszystkie wyjścia (Q) oraz markery (M) jako 0 oraz (dla ułatwienia) wszystkie wejścia jako *FAŁSZ* i *czytamy* program tak długo aż dojdziemy do wniosku, że nic się nie zmienia:

W pierwszym wierszu marker M ustawiony jest na 0, a więc sygnał od lewej strony nie *przechodzi* do Q1 – Q1 pozostaje wyłączone (na wyjściu pozostaje 0 logiczne).

W wierszu drugim wejście I1 jest odwrócone, a więc symbol *przepuszcza* gdy na wejściu jest logiczne 0, natomiast *blokuje* gdy wejście ustawione jest na 1. W tym przypadku (zakładamy na wejściu 0), I1 *przepuszcza*, więc logiczne 1 *dociera* do M. Od tego momentu marker M przyjmuje wartość 1.

W wierszu trzecim marker M już przyjmuje wartość 1, więc wyjście Q2 ustawiane jest na 1.

W tym miejscu musimy zapamiętać stany wyjść oraz markerów i zacząć czytać schemat od początku.

W pierwszym wierszu marker M przyjmuje wartość 1 (z poprzedniego przejścia), zatem Q1 ustawiany jest na 1.

W drugim wierszu I1 (nadal) nadal *przepuszcza* logiczną 1, więc M pozostaje 1.

W trzecim wierszu również nie następują żadne zmiany.

W tym miejscu widać, że kolejne interpretacje schematu nie powodują zmian wyjść ani markerów. Należy zatem przyjąć, że zmienia się któreś z wejść. Jedyne wejście jakie możemy tutaj znaleźć to I1.

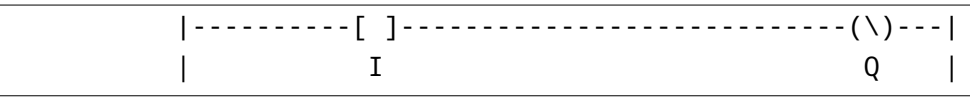
W pierwszym wierszu nie nastąpią zmiany, w drugim I1 przestanie *przepuszczać* sygnał, a więc marker M zostanie ustawiony na 0. W wierszu trzecim M przyjmuje wartość 0, więc Q2 również przyjmie wartość 0. Czytając ponownie: M w pierwszym wierszu przyjmie wartość 0, więc Q1 również. Itd., itd...

Oprogramowanie, które może pomóc: PLC Fiddle (przeglądarkowe), Do-more Designer Programming Software, PLC Ladder Simulator (Android), EKTS (Electrical Control Techniques Simulator)

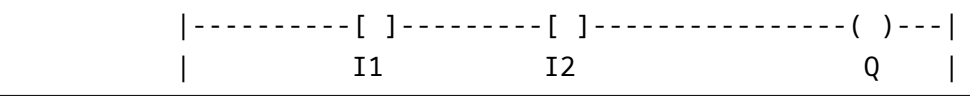
Zadanie 2.1

Jakim bramkom logicznym odpowiadają poniższe programy?

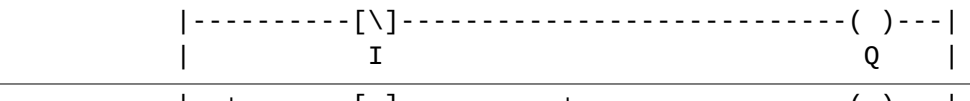
A



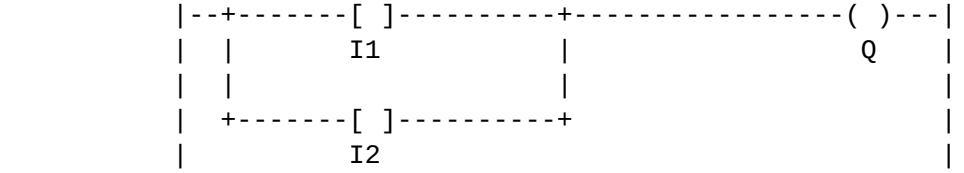
B



C



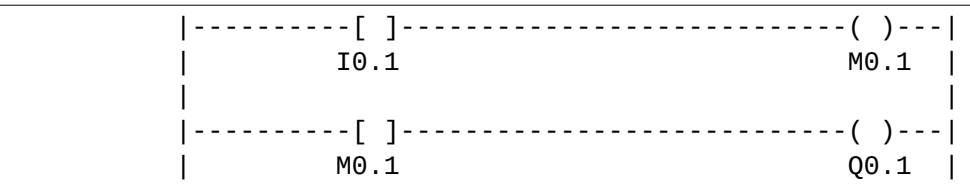
D



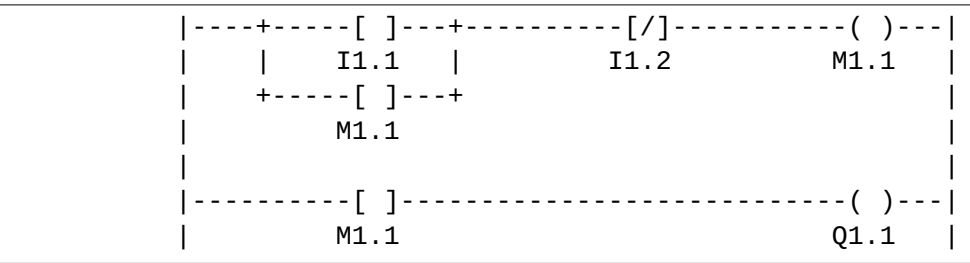
Zadanie 2.2

Zinterpretować działanie poniższych programów. Dla ułatwienia zadania można interpretować wejścia jako przyciski a wyjścia jako diody. Należy jednak pamiętać, że mogą to być inne urządzenia o wyjściach logicznych: czujniki, układy scalone, przekaźniki lub *odbiorniki* – np. przekaźniki.

A



B



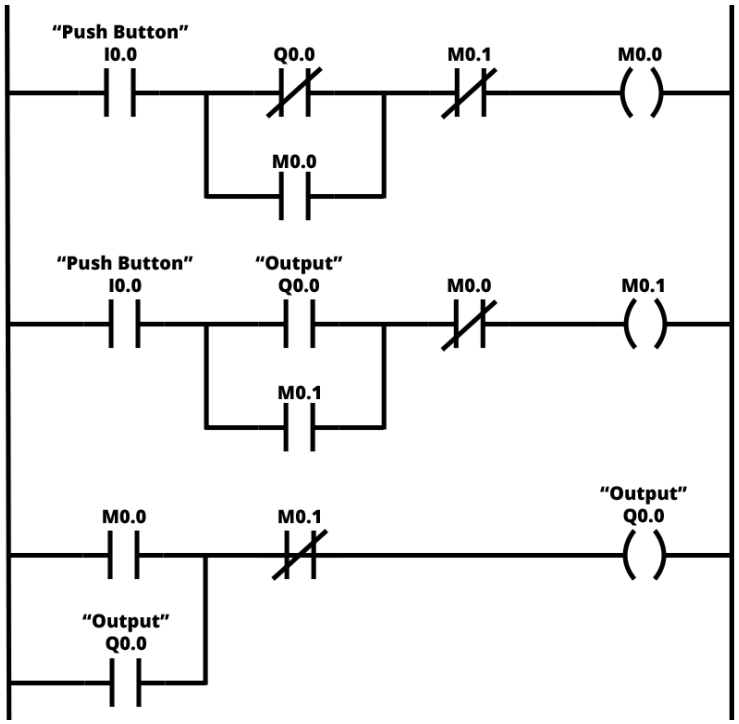
Podpowiedź do punktu B: co musi się stać, by na wyjściu Q1.1 pojawiła się 1? Co musi się stać by z powrotem zmieniła się na 0?

Zadanie 2.3

Zinterpretować poniższy program – pojedynczy przycisk włącz/wyłącz): ile jest wejść (fizycznych), wyjść oraz markerów, ile jest kombinacji wejść?

Rada: przeanalizuj schemat i ustal co pojawia się na wyjściu jeśli na wejściu nie ma sygnału. Co musi się stać, żeby na wyjściu pojawił się stan przeciwny? Co musi się stać, żeby ten stan znowu się zmienił? Czy program wraca w którymś momencie do *stanu początkowego*?

Co robi program?



Zadanie 2.4

Przy użyciu logiki drabinkowej napisać program obsługi wózka w taki sposób aby:

- Po pojawieniu się sygnału na wejściu I0.0 podaje sygnał ruchu w prawo (Q0.0 = 1) i nie zatrzymuje się po zniknięciu tego sygnału (propozycja: wykorzystać marker)
- Po dojechaniu do krańcówki prawej (I1.0) wózek zmienia kierunek podróży w lewo ustawiając wyjście Q0.1 na 1 oraz wyłączając ruch w prawo.
- Po dojechaniu do krańcówki lewej (I1.1) wózek kończy bieg.

Dodatkowe przyciski: I0.1 – zatrzymuje wózek (niezależnie od kierunku ruchu). I0.2 – wymuszający powrót wózka.

https://en.wikipedia.org/wiki/Ladder_logic
https://pl.wikipedia.org/wiki/Programowalny_sterownik_logiczny
http://www.plcacademy.com/ladder-logic-examples/?epik=0LyiXE_IWX-V-
http://home.agh.edu.pl/~aprzem/pliki/plc_1.pdf

Patryk Król

V1.5