Politechnika Wrocławska

Katedra Mechatroniki, Automatyki i Systemów Sterowania

Pracownia Automatyki, Modelowania i Mechatroniki

Sterowniki PLC – Allen-Bradley
Konfiguracja i programowanie w języku Ladder

Autor: Jacek Jagodziński
Sygnatura: 1/sir/021/01001
Wersja: 1.1

Laboratorium urządzeń i układów automatyki

Laboratorium Systemów Automatyki i Mechatroniki

Wrocław – 2018

Tablica 1. Kontrola wersji

Wersja	Autorzy	Data
1/SIR/021/01001 *Wersja:* 1.0	Jacek Jagodziński	30/09/2018
1/SIR/021/01001 *Wersja:* 1.1 Raport PREPRINTY W04/2018/P-022	Jacek Jagodziński	10/10/2018

Tablica 2. Historia zmian

Wersja	Opis	Autorzy
1/SIR/021/01001 *Wersja:* 1.0	Wersja podstawowa	–
1/SIR/021/01001 *Wersja:* 1.1	Poprawiony rys. 20 Modyfikacja zad. 5	–

Spis treści

1769 **CompactLogix**

1769 CompactLogix to rodzina sterowników występująca w jednej z pięciu wersji (*L35CR*, *L35E*, *L32C*, *L32E*, *L31*). Wersja determinuje między innymi ilocść portów oraz wielkość pamięci. Seria 1769 CompactLogix wspiera sieci *Etherner/IP*, *ControlNet*, *DH-485*, *RS-232*, *Modbus* oraz *DeviceNet* (poprzez moduły) [1].

FX5U CPU

Jednostki centralna 1769-L35E:

- pozwala na podłączenie do 30 modułów *I/O* (w trzech bankach),
- pozwala na odczyt dancyh z modułów co, 1ms przy 1 . . . 4 modłułach oraz 2ms przy 5 . . . 30 modułów,
- pobiera 90mA przy zasilaniu 24VDC,
- posiada 1 port *EtherNet/IP*,
- posiada 1 port szeregowy *RS-232* [1].

Wprowadzenie

Cele ćwiczenia:

- Zapoznanie się ogólnym schematem konfiguracji sterowników na przykładzie sterowników Allen-Bradley'a.
- Nauka podstaw języka drabinkowego (LD) (proste funkcje).
- Wykorzystanie wejść/wyjść sterownika do praktycznych zastosowań.

Sterowniki PLC (Programmable Logix Controller) stanowią ogromną grupę regulatorów cyfrowych, przeznaczonych do realizacji układów sterowania w praktyce przemysłowej. Zasadniczo w środowiskach udostępnionych przez producentów PLC, można zrealizować dowolny specjalizowany algorytm/układ regulacji, przy ograniczeniach związanych z pamięcią sterownika oraz wymaganiami czasu rzeczywistego. Obecnie, ze względu na ogromną funkcjonalność omawianych urządzeń, niektórzy producenci nazywają je sterownikami PAC (Programmable Automation Controller).

W normie IEC-1131-3 można odnaleźć wytyczne dotyczące programowania sterowników PLC, opisano między innymi:

- model oprogramowania sterowników PLC,
- elementy organizacyjne oprogramowania,
- modele komunikacji,
- typy danych w PLC,
- języki programowania PLC [10].

Wymienione wyżej elementy stanowią elementarną bazę niezbędną dla prawidłowej konfiguracji i programowania sterowników PLC. Uszczegółowiając w sterownikach korzysta się najczęściej z pięciu różnych języków programowania. Są to: j. drabinkowy (LD), j. schematów blokowych (FBD), listy instrukcji (IL), tekst strukturalny (ST), graf sekwencji (SFC), niniejsza instrukcja stanowi wprowadzenie tylko do jednego – LD – dlatego z pewności nie wyczerpuje tematyki, a stanowi jedynie zalążek większego zbioru zagadnień. Dla szerszego spojrzenia niezbędna jest pogłębiona analiza materiałów źródłowych.

Sterowniki Allen-Bradley

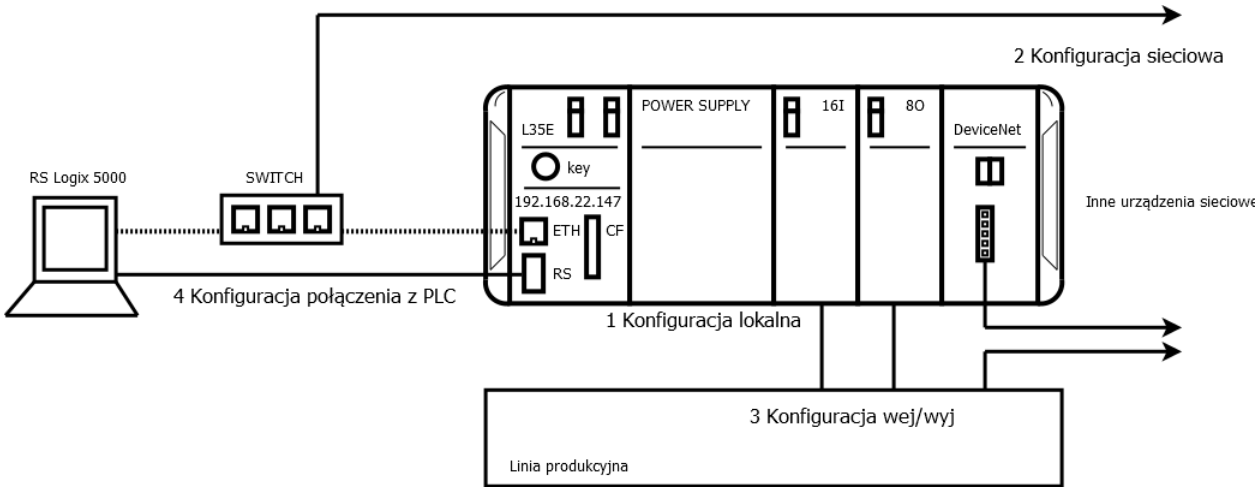
Allen Bradley to amerykańskie przedsiębiorstwo produkujące sprzęt elektroniczny założone na początku ubiegłego wieku. Rockwell Automation zakupił markę 1985 roku, obecnie jest to jeden największych producentów automatyki przemysłowej na świecie [6]. Autoryzowanym dystrybutorem produktów Rockwell Automation na rynek polski jest RAControls (stan na 2018) [4]. Sterowniki Allen-Bradley podzielone są w zależności od wielkości aplikacji na trzy grupy:

- ContorlLogix (GuardLogix spełniający wyższe wymogi bezpieczeństwa) – duże aplikacje – wielkie linie produkcyjne.
- CompactLogix – średnie-małe aplikacje – sterowanie maszyn, napędów i rozwiązań współpracujących z sieciami przemysłowymi EtherNet/IP, ControlNet, DeviceNet. Ta seria będzie poznawana na laboratorium.
- MicroLogix i Micro800 – micro aplikacje – sterowniki kompaktowe z możliwością rozbudowy, z zastosowań można wymienić automatykę budynkową (sterowanie oświetleniem itp.), systemy bezpieczeństwa, czy sterowanie proste maszynami (taśmociągi itp.) [8].

Konfiguracja sterownika

W przypadku niniejszej instrukcji omawiane zagadnienia dotyczą sterowników Allen Bradley, natomiast sposób postępowania przy konfiguracji urządzeń tego typu jest na tyle uniwersalne, że może być zastosowane do także do innych typów urządzeń.

Czynności podstawowe niezbędne do prawidłowego uruchomienia sterownika zostały pokazane na rysunku 1.



Rysunek 1. Etapy konfiguracji sterownika realizującego obsługę linii produkcyjnej

Niezbędnymi etapami uruchomienia sterownika PLC są:

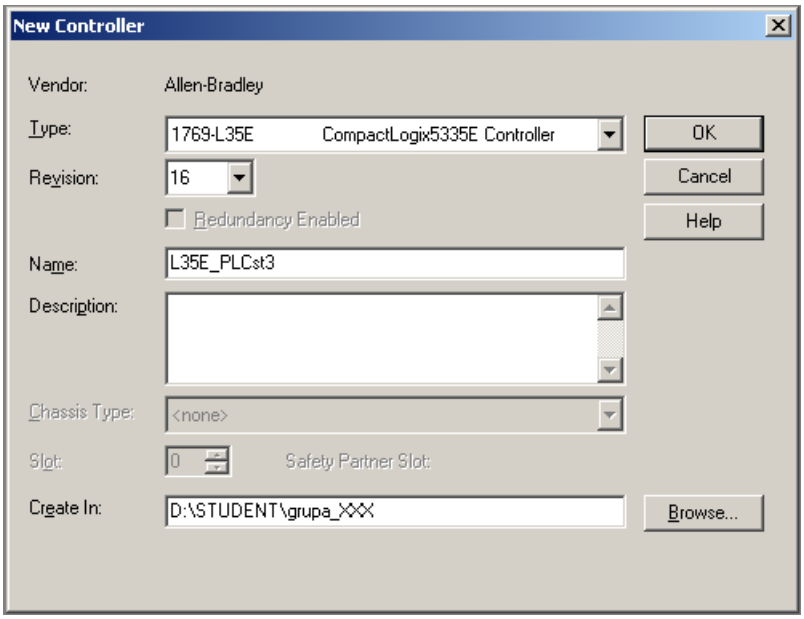
1. Konfiguracja lokalna – w oprogramowaniu producenta należy zdefiniować jakie moduły posiada jednostką, którą projektant zamierza programować (wskazać jednostkę centralną, moduły wejść/wyjść, itp.).
2. Konfiguracja sieciowa – należy zdefiniować i ustawić parametry dla wszystkich modułów sieciowych oraz interfejsów, które znajdują się w urządzeniu. W przypadku jednostki pokazanej na rysunku dysponujemy portem RS-232 i EtherNet/IP w jednostce centralnej oraz modułem wyposażonym w DeviceNet (do konfiguracji będzie należało np. zdefiniowanie parametrów transmisji, czy adresu IP urządzenia). Bardziej szczegółowe informacje w tym zakresie są omawiane na *Sieciach przemysłowych*.
3. Dostosowanie się do sposobu podłączenia wejść oraz wyjść (cyfrowych i analogowych) – konfigurację taką w wielu sterownikach realizuje się już w samym programie poprzez nadanie nazw odpowiednim wejściom i wyjściom. W przypadku Allena Bradleya, należy zdefiniować tagi, które będą odnosiły się do wejść/wyjść sterownika PLC, a w dalszej kolejności rzeczywistym urządzeniom ze linii produkcyjnej.
4. Konfiguracja połączenia z PLC – fizyczny sposób połączenia sterownika z komputerem, na rysunku zostały pokazane dwa sposoby takiego połączenia przez RS oraz przez sieć Ethernet. Należy zatem zwrócić uwagę, przez jaki interfejs projektant łączy się ze sterownikiem (fizyczne podłączenie). Pamiętając, że formalnie powinniśmy zdefiniować adres PLC, z którym ustanawia się połączenie oraz jaki adres nadaje się PLC w konfiguracji. Na przykład jeżeli zostanie dokonane połączenie z urządzeniem o adresie 192.168.22.178, a w konfiguracji zostanie wysłane 192.168.22.179, to po takiej rekonfiguracji adres 192.168.22.178 będzie już nie aktywny, a nowy adres urządzenia to 192.168.22.179.
5. Utworzenie programu realizującego postawione zadanie.

Konfiguracja w *RSLogix 5000*

Tworzenie projektu

Uruchom program z pulpitu *RSLogix 5000* lub z menu *Start/Programy/Rockwell Software/RSLogix 5000 Enterprise Series/RSLogix 5000*.

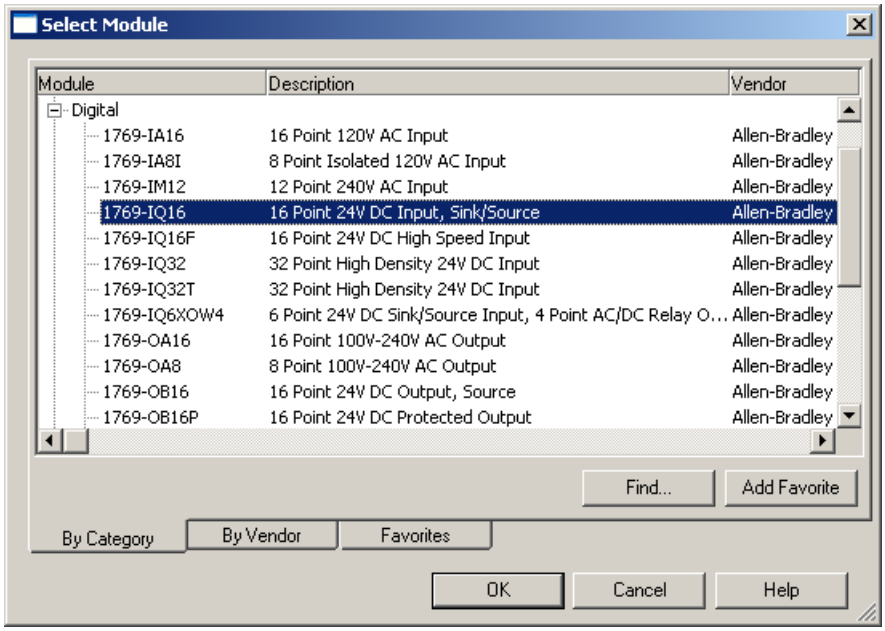
Aby utworzyć nowy projekt wybierz *File/New*.



Rysunek 2. Menu *New Controller*

W menu *New Controller* (rys. 2) wybrać następujące elementy.
Typ jednostki centralnej – należy odczytać ze sterownika (np. *1769-L35E*).
Rewizje – wersje oprogramowania na sterowniku (np. 16).
Następnie należy ustalić nazwę sterownika – nazwa powinna jednoznacznie identyfikować urządzenie, w praktyce nazwy pomagają także określić położenie sterownika) na przykład *L35E_PLC_st3*.
Utwórz projekt w katalogu *STUDENT* na dysku *D*. Nazwa powinna precyzyjnie określać grupę.

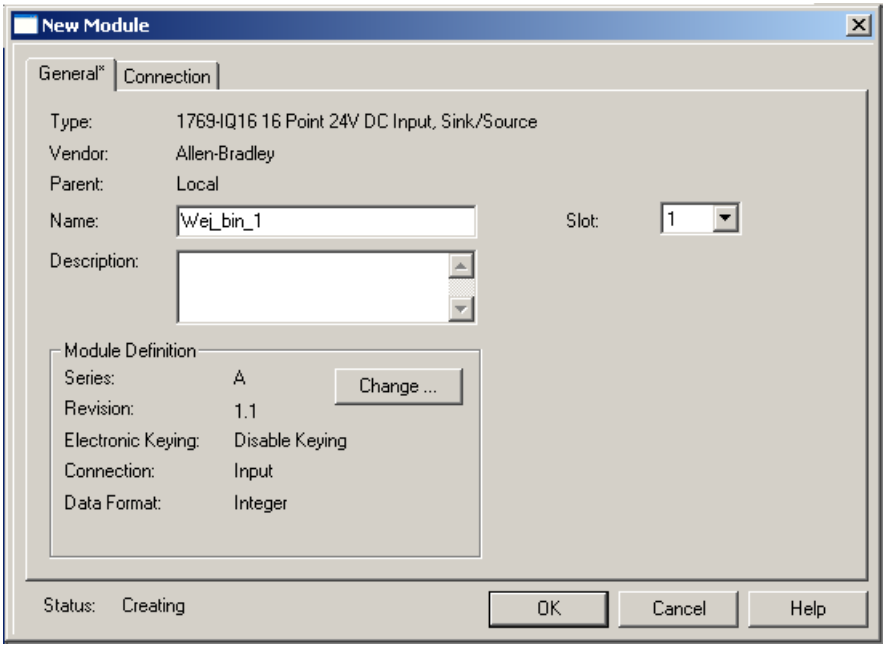
Definicja sprzętu – modułów
W oknie projektu (po lewej stronie) rozwiń *I/O Configuration/Backplane, CompactLogix, System/*, następnie prawym klawiszem myszy zaznaczając *ComapctBus Local* wybierz opcję *New module*.



Rysunek 3. Menu *Select Module*

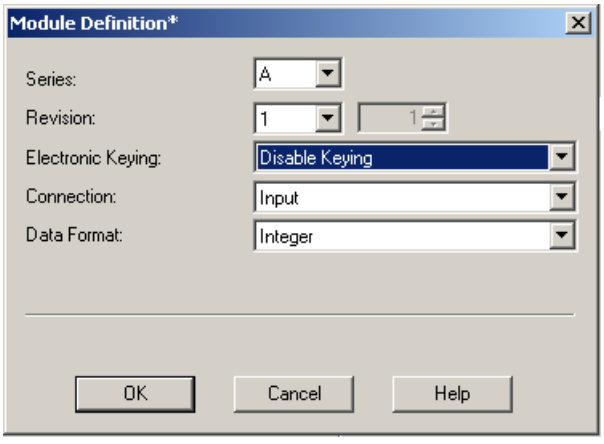
O okienku *Select Module* (rys. 3) wybierz moduł zgodnie z rzeczywistym podłączeniem np. *Module/Digital/1769-IQ16 Description 16 Point 24V DV Input, Sink/Source.*
Opis modułu znajduje się na sterowniku w przypadku Allen-Bradley'a najczęściej na wewnętrznej stronie drzwiczek.

Następnie w okienku *New Module* (rys. 4) należy zdefiniować, mazwę moduł (*Name*) oraz określić położenie (*Slot*) – jednostka centralna usytuowana jest domyślnie w slocie 0, natomiast zasilacz w numeracji jest pomijany, dlatego nadajemy kolejny numer, licząc moduły od lewej do prawej, czyli 1, a kolejny moduł 2 itd.



Rysunek 4. Menu *New Module*

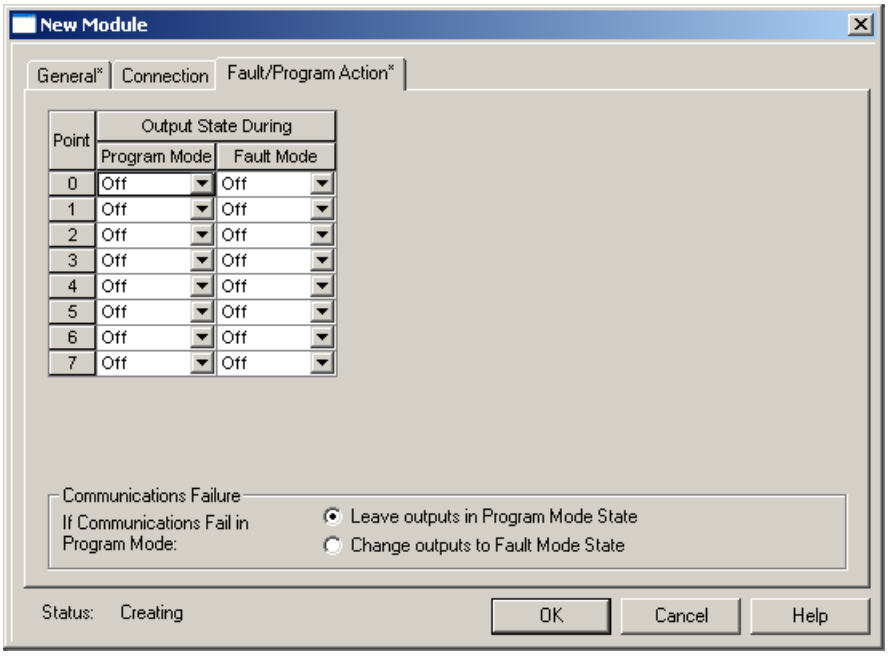
W obszarze *Module Definition* po naciśnięciu *Change ...*, pojawi się okno *Module Definition* (rys. 5) którym należy zaznaczyć *Electronic Keying: Disable Keying*. Powoduje to, że moduł nie będzie podlegał weryfikacji serii i wersji oprogramowania, dzięki temu nie trzeba tych parametrów definiować. W praktyce przemysłowej warto opcję wykorzystywać, w celu automatycznej weryfikacji podmiany danego modułu.



Rysunek 5. Menu *Module Definition*

W podobny sposób należy zdefiniować pozostałe moduły.

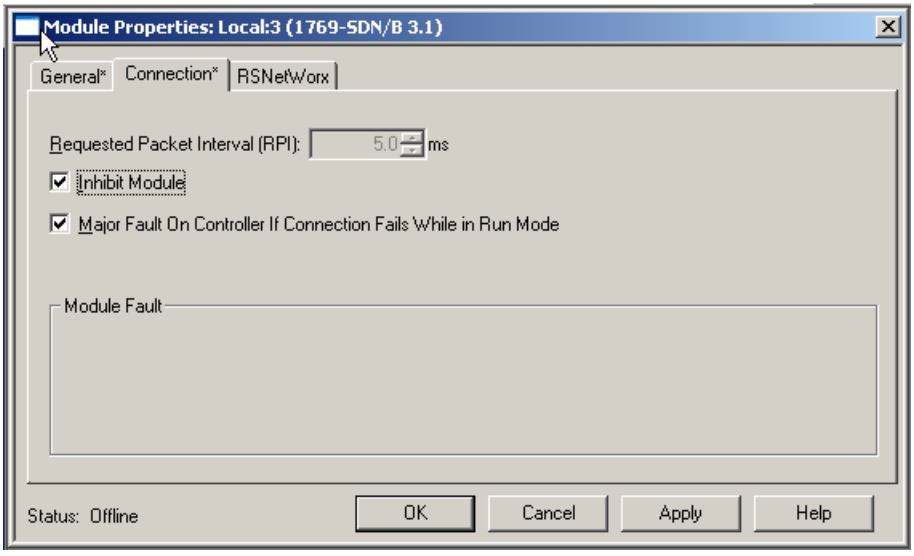
Przy definicji modułu wyjściowego w okienku *New Module*. Pojawia się także zakładka *Fault/Program Action**, porównaj rysunek 6.



Rysunek 6. Menu *New Module* z zakładką *Fault/Program Action** dla moduły wyjściowego

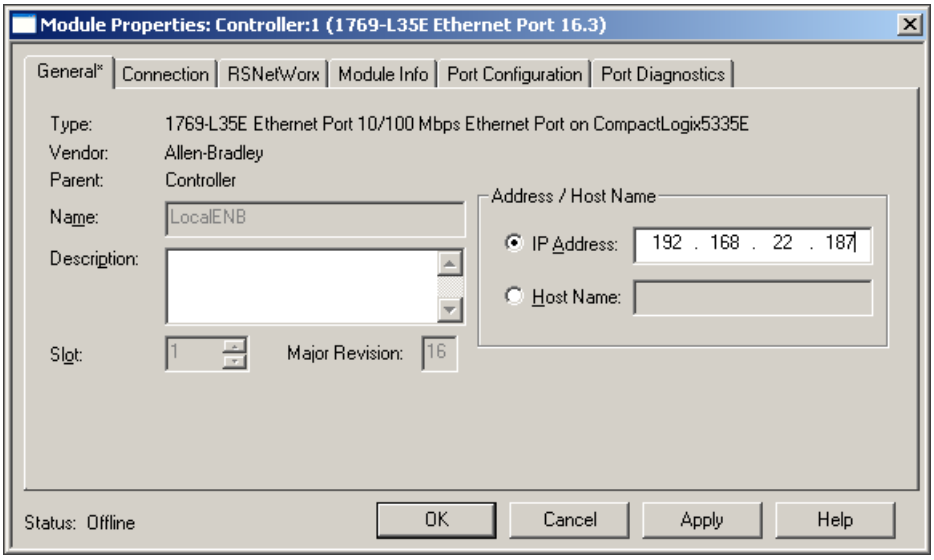
Warto zwrócić uwagę, że w praktyce projektant musi zastanowić się także, jak powinny zachować się stany wyjść w przypadku wyłączenia/wstrzymania/czy błędu programu. Czy zostaną w stanie w jakim były, czy też istnieje zdefiniowane położenie bazowe do którego należy wrócić (np. czy po awarii dane drzwi powinny zostać otwarte, czy zamknięte)?

Należy skonfigurować wszystkie moduły rzeczywiście podłączone do sterownika. Jeżeli na stanowisku znajduje się moduł sieciowy np. DeviceNet, mimo iż nie wykorzystywany, również należy go skonfigurować (nadać odpowiedni numer slotu), natomiast jeżeli projektant nie zna w danej chwili wszystkich parametrów, można moduł oznaczyć w zakładce *Connection* jako *Inhibit Module* (rys. 7), co spowoduje wstrzymanie komunikacji – innymi słowy niepoprawna konfiguracja, czy brak działania nie będzie sygnalizowany przez zieloną diodę *I/O* na jednostce *CPU*.



Rysunek 7. Menu *Module Properties Local* z otwartą zakładką *Connection*

Nadawanie adresu IP jednostce centralnej (opcjonalne – jeżeli nie ustawiony)
Większość sterowników w laboratorium to urządzenia widoczne w sieci, posiadające przypisany adres IP (opisany na sterowniku), w związku z powyższym nawet jeżeli wykorzystuje się połączenie bezpośrednie poprzez port szeregowy warto ustawić adres w sterowniku, co umożliwia późniejsze połączenie ze sterownikiem poprzez innych użytkowników. W tym celu należy się z panelu bocznym (*I/O Configuration*) wybrać port ethernetowy (nazwa zależna od jednostki) np. *1769-L35E Ethernet Port LocalENB*, następnie po kliknięciu prawym przyciskiem myszy wybrać *Properties* i w zakładce w zakładce*General* (rys. 8) wpisać właściwy adres IP.

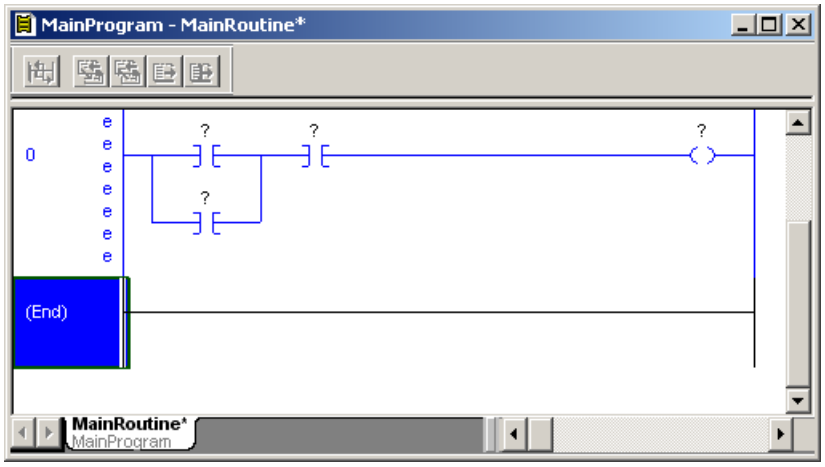


Rysunek 8. Menu *Module Properties: Controller* z otwartą zakładką *General*

Programowanie

Main Routine
W celu rozpoczęcia programowania sterownika należy w menu projektu rozwinąć *Task/MainTask/MainProgram* i uruchomić *MainRoutine*. W *RSLogix* przyjęto konwencje, że do każdego programu przypisana jest jedna główna procedura (*MainRoutine*), z której można wywoływać inne procedury, poprzez wykorzystanie polecenia *Skocz do podprocedury* (*JSR – Jump To Subroutine*).
Główna procedura domyślnie wykorzystuje schemat drabinkowy (Ladder) jako język programowania.

Tworzenie struktury programu

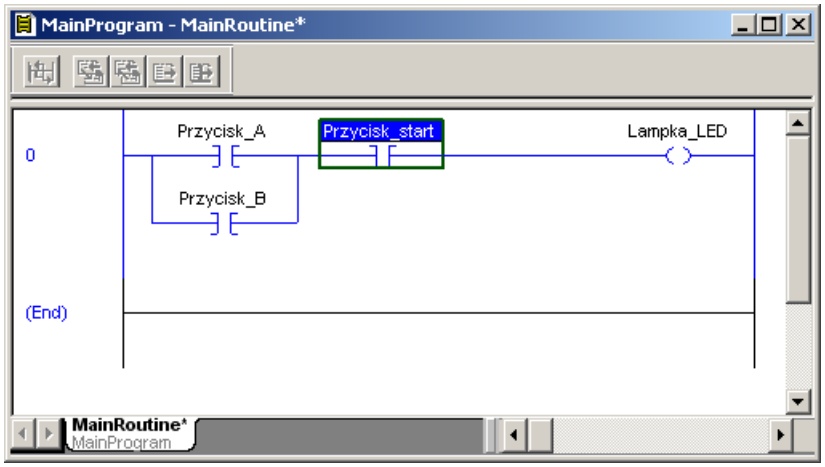


Rysunek 9. Obszar programu głównego – przykładowy program w języku Ladder

Aby rozpocząć programowanie w języku Ladder przeciągnij *Rung* (szczebel) do obszaru tworzonego *MainRoutine*. Następnie przesuń styki normalnie otwarte (*XIC – Examin If Colsed*) oraz cewki zwykłe (*OTE – Outut Energize*) do obszaru i stwórz przykładowy program jak na rysunku 9.

Tagi (zmienne)
W sterownikach wykorzystuje się podobne struktury danych jak informatyce, zmienne binarne, integer, real itp. Przed rozpoczęciem pracy przy nowych sterowniku warto zapoznać się z dostępnymi strukturami danych oraz sposobem adresowania, gdyż często zdarza się, że pewne obszary pamięci na siebie nachodzą, a stosowanie systemu liczbowego dziesiątkowego nie zawsze jest przyjętym standardem.
Aby przypisać zmienne do programu w sterownikach Allen Bradley wykorzystuje się tagi. Idea polega na tym, aby nadrzędnie wykorzystywać nazwy zmiennych odnoszących się do rzeczywistych urządzeń (np. przycisk start), a nie fizycznych adresów wejść wyjść na sterowniku, które i tak w pewnym momencie trzeba zdefiniować. Takie podejście pozwala na zaprogramowanie sterownika w sposób przejrzysty, a nawet w sytuacji gdy jeszcze nie wiadomo do którego wejścia sterownika będzie podłączone dane urządzenie wykonawcze.

Tag bazowy
Poniżej został pokazany program z rysunku 9, lecz z przypisanymi tagami bazowymi.



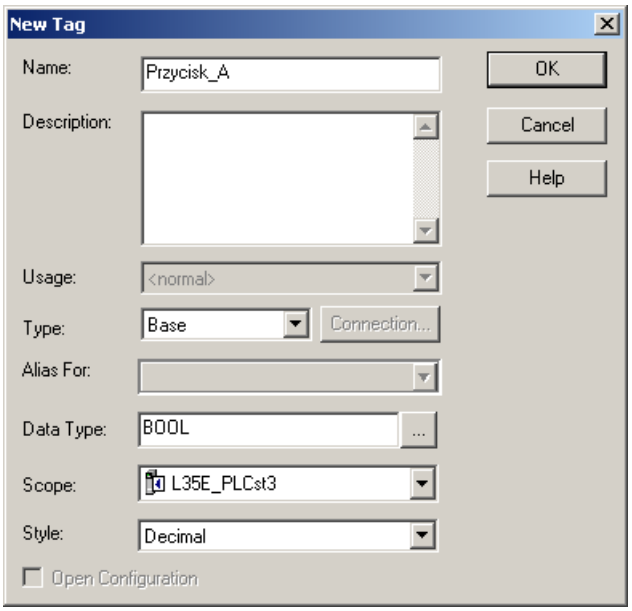
Rysunek 10. Przykładowy program z przypisanym tagami bazowymi

Program realizuje następujące zadanie: jeżeli (*Przycisk_A or Przycisk_B*) *and Przycisk_start* zostanie naciśniety to zostanie zapalona *lampa_LED*. Przykład obrazuje jak tworzyć w Ladderze proste funkcje logiczne.

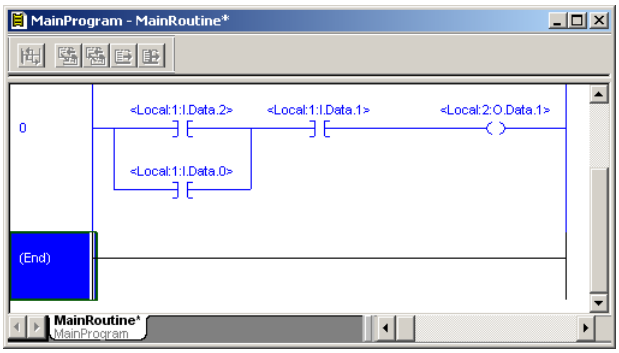
Zwróćmy uwagę, że w tym wypadku nie podano odniesienia do rzeczywistych zmiennych: obszarów pamięci, czy danego wejścia w sterowniku, natomiast przypisano jedynie nazwę. Do tego celu został użyte tagi bazowe.

Aby stworzyć tag bazowy umieść kursor na styku bądź cewce, naciśnisnąć prawy przycisk myszy i wywołać pierwszą opcję *New tag...*, (jeżeli nazwa tagu już istnieje na pierwszej pozycji pojawi się *Edit 'Nazwa' Properties*) gdy otworzy się okienko rys. 11, Należy umieścić nazwę oraz wybrać *type* (*typ*) jako *Base* (*bazowy*).

Przypisanie tagu do cewki, bądź styku możliwe jest również poprzez wybranie już istniejącego tagu z listy. Poprzez podwójne kliknięcie w miejscu nazwy tagu pojawia się lista rozwijana z już zapamiętanymi tagami. Formalnie użytkownik w ten sposób może uzyskać dostęp także do adresów wejść czy wyjść bezpośrednio, zatem można przypisać do danej funkcji żądany adres tak jak na rysunku 12, jednakże ze względu na czytelność jest to nie zalecane.



Rysunek 11. Menu *New Tag Base*



Rysunek 12. Przykładowy program wyłącznie z adresami fizycznymi

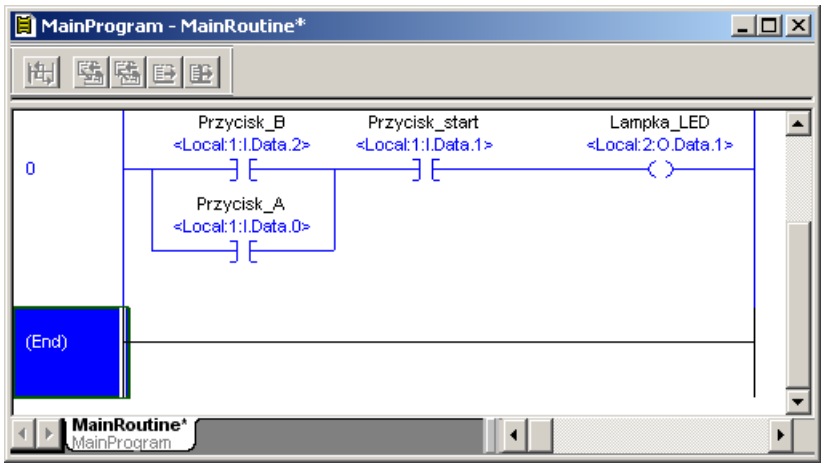
Adresowanie w Allen-Bradley

Format adresowania jest następujący: *Lokalizacja:Slot:Typ.Member.SubMeber.Bit*
gdzie:

- *Lokalizacja* – miejsce podłączenia (sieci), może przyjmować wartości:
  - *Local* – oznacza że w dany adres znajduje się w kasecie z jednostką centralną.
  - *Adapter_Name* – wskazuje na adapter komunikacyjny, moduł itp.
- *Slot* – numer slotu modułu w kasecie (na szynie DIN).
- *Typ* – typ danych:
  - *I* – sygnał wejściowy,
  - *O* – sygnał wyjściowy,
  - *C* – konfiguracja,
  - *S* – status.
- *Member* – specyfikacja rodzaju danych jakie moduł może przechowywać:
  - *Data* – zazwyczaj wartości binarne (moduł cyfrowy),
  - *Channel (CH#)* – zazwyczaj dane dla danego kanału (moduł analogowy).
- *SubMember* – opcjonalne uszczegółowienie pola Member.
- *Bit* – konkretny punkt w cyfrowym module (np. przyjmuje wartości 0-15 w module 16 wejściowym) [2].

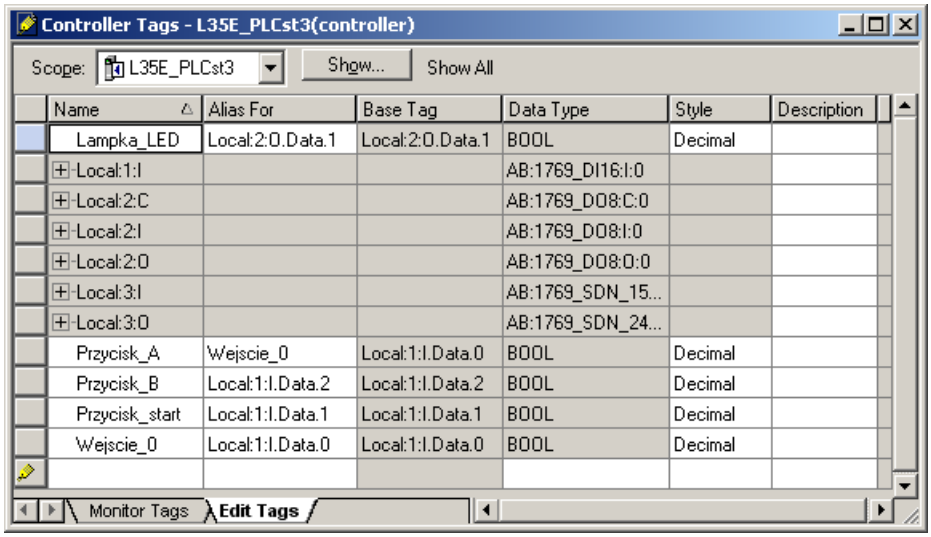
Stąd pierwsze wejście w module cyfrowym znajdującym się w slocie pierwszym przyjmie adres: *Local:1:I:Data.0*, a wyjście 7 w module wyjść cyfrowych umieszczonym w slocie 2 to *Local:2:O:Data.6*.

Tag Alias
Poprzez aliasy przypisuje się danej nazwie adres fizyczny. Alias pozwala także odnieść się do innego już istniejącego tagu. Tworzy się go analogicznie jak tag Bazowy, jednakże w okienku *New Tag* w polu *Type* wybiera się *Alias* i przypisuje adres fizyczny (bądź inny tag) – *Alias for*.
Rezultat przykładowego programu wraz z tagami aliasami został pokazany na rysunku 13.



Rysunek 13. Przykładowy program z tagami aliasami

Dodatkowo na rysunku 14 można zobaczyć przegląd tagów utworzonych w przedstawionym zadaniu.

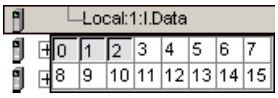


Name	Alias For	Base Tag	Data Type	Style	Description
Lampka_LED	Local:2:O.Data.1	Local:2:O.Data.1	BOOL	Decimal	
Local:1:I			AB:1769_DI16:I:0		
Local:2:C			AB:1769_DO8:C:0		
Local:2:I			AB:1769_DO8:I:0		
Local:2:O			AB:1769_DO8:O:0		
Local:3:I			AB:1769_SDN_15...		
Local:3:O			AB:1769_SDN_24...		
Przycisk_A	Wejscie_0	Local:1:I.Data.0	BOOL	Decimal	
Przycisk_B	Local:1:I.Data.2	Local:1:I.Data.2	BOOL	Decimal	
Przycisk_start	Local:1:I.Data.1	Local:1:I.Data.1	BOOL	Decimal	
Wejscie_0	Local:1:I.Data.0	Local:1:I.Data.0	BOOL	Decimal	

Rysunek 14. Menu *Controller Tags* z listą utworzonych tagówi

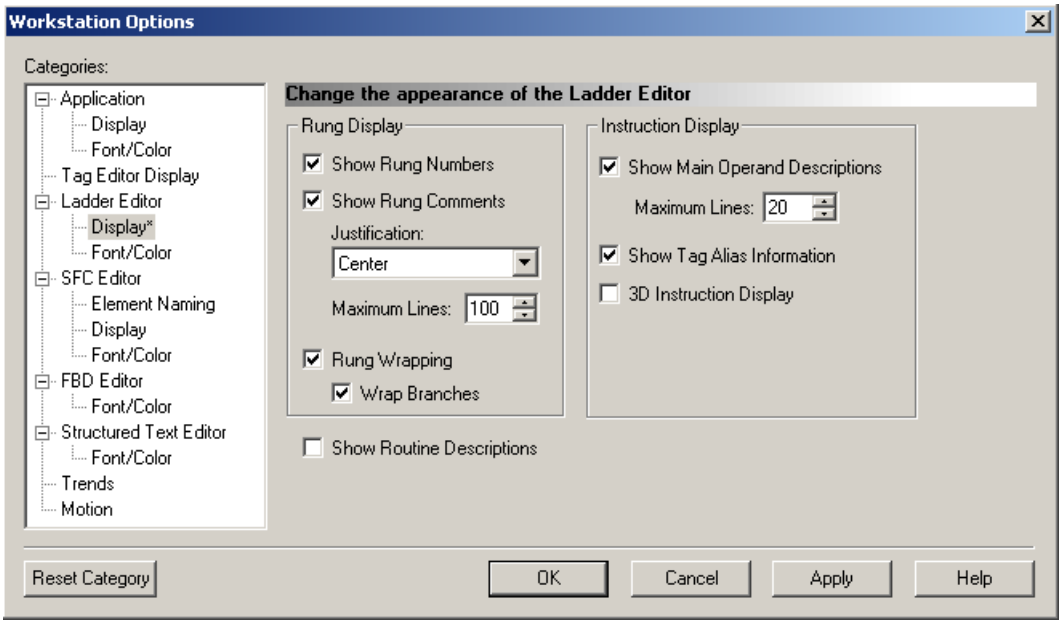
Warto zwrócić uwagę, że tag *Przycisk_A* wskazuje na tag *Wejscie_0*, natomiast tag *Wejscie_0* na konkretne adres w sterowniku *Local:1:I:Data.0*. Mimo takiej konwencji w programie drabinkowym i tak widać tylko: nazwę tagu aktualnie wybranego i odniesienie do adresu fizycznego, bez względu na to przez ile aliasów prowadzi ścieżka do finalnego adresu.
Wykorzystanie tagów powoduje stosowanie pewnej specyficznej konwencji programowania języka drabinkowego. Poprzez odpowiednie wykorzystanie tagów, użytkownik nie musi znać fizycznych adresów raz już przypisanych (gdzie łatwo popełnić błąd) tylko w prosty sposób przypisać na przykład przycisk *B* do Wejścia 0 (z którym to dopiero związany jest fizyczny adres).

W przypadku tworzenia nowego tagu można dodatkowo podejrzeć (oznaczenie szarym kolorem), które zmienne wejścia posiadają już przypisany tag (por. rys. 15).



Rysunek 15. Okno wyboru bitu wejsciowego

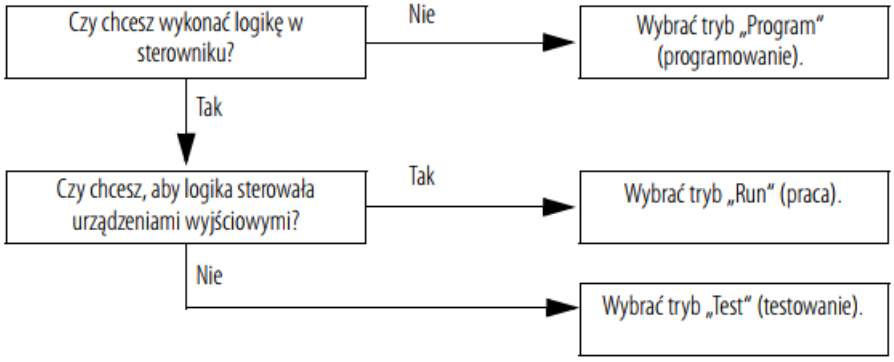
Zależnie od konfiguracji może się zdarzyć, że pod tagami nie widać informacji o fizycznych adresach (mimo, iż skorzystano z Aliasów). Wtedy należy upewnić się, iż ich wyświetlanie jest poprawnie ustawione w opcjach. Należy wejść w *Tools/Options/Ladder Editor/Display* i zaznaczyć opcję *Show Tag Alias Information* (rys. 16).



Rysunek 16. Menu *Workstation Options*

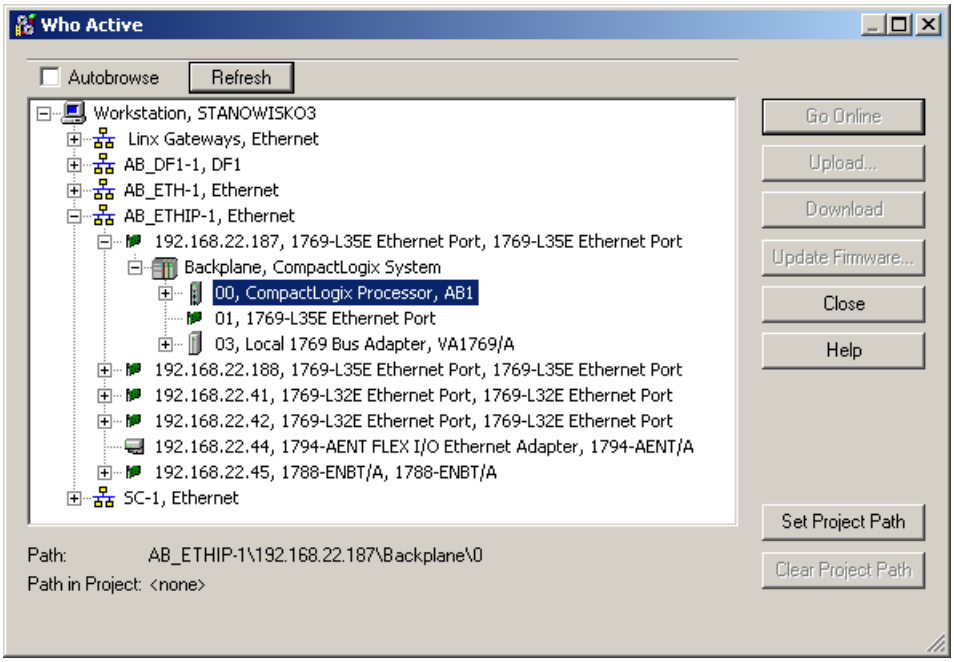
Uruchamianie programu

Sterowniki Allen Bradley z serii CompactLogix umożliwiają działanie sterownika w jednym z trzech trybów. Wzajemne zależności pomiędzy poszczególnymi opcjami podsumowuje rysunek 17.



Rysunek 17. Trzy tryby pracy sterowników Allen Bradley (źródło: [3] s.31)

Przestaw kluczyk w tryb Program, następnie wybierz zakładkę *Communications/Who Active*, odnajdź jednostkę centralną z którą chcesz się połączyć i naciśnij *Download* jak pokazano na rys. 18.

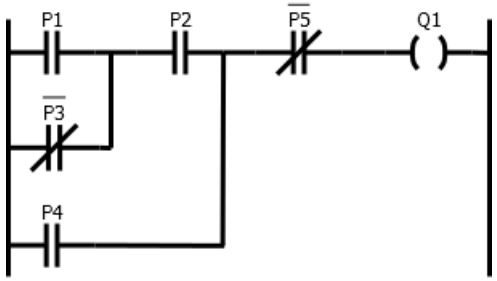


Rysunek 18. Menu *Who Active*

W celu testowania programu należy pamiętać o przełączeniu się w tryb *Run* bądź *Test*, w zależności od preferencji wykorzystania sterownika (por. rys. 17). Aby znów przejść do trybu edycji programu niezbędne jest rozłączenie się z urządzeniem – tryb *offline*. W menu należy wybrać *Communications/Go Offline*.
Uwaga, jeżeli występują problemy z łącznością warto wyłączyć lub zmienić ustawienia zapory sieciowej.

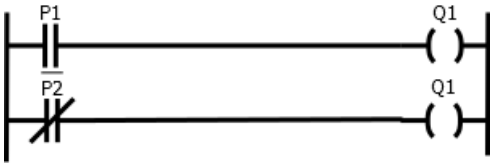
Zadania do wykonania

Logika języka Ladder
Zad. 1 Wprowadzanie do podstawowych struktur logicznych języka Ladder
a) Przepisz 1 wejście binarne na 3 wyjście binarne sterownika.
b) Na wyjście 2 wyślij negacje wejścia 1 ( ) .
c) Zrealizuj program pokazany na rysunku 19, czy można zapalić styk *Q1* jednym przyciskiem? Zapisz funkcję logiczną realizowaną przez pokazany program.

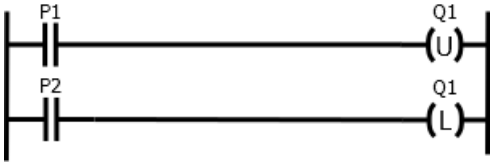


Rysunek 19. Zadanie 1c

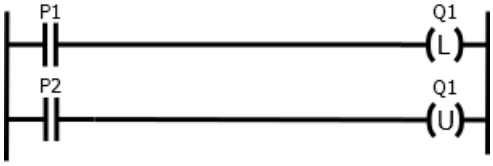
Zad. 2 Kolejność wykonywania instrukcji
Zrealizuj i przetestuj trzy poniższe programy (rys. 20, 21 oraz 22). Jaka jest kolejność wykonywania zadań w języku Ladder, czym różni się od programowania w językach tekstowych (np. *C++*)? Narysuj diagramy czasowe.
Na rysunkach wykorzystano nomenklaurę zaproponowaną przez Allena-Bradley'a do prezentacji sytków Latch ( ) oraz Unlatch ( ). Należy zwrócić uwagę, iż nie jest to przyjęty standard, w zdecydowanej większości sterowników styki nazywają się *Set* oraz *Reset* i są onzaczane odpowiednio (S) i (R).



Rysunek 20. Zadanie 2a

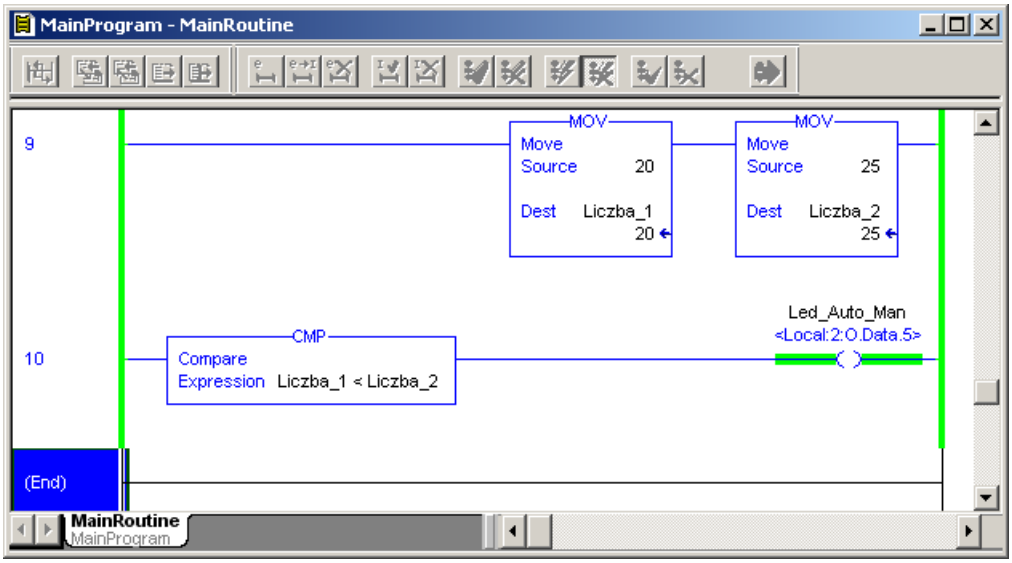


Rysunek 21. Zadanie 2b



Rysunek 22. Zadanie 2c

Podstawowe funkcje



Rysunek 23. Przykład wykorzystania funkcji Move i komparatora

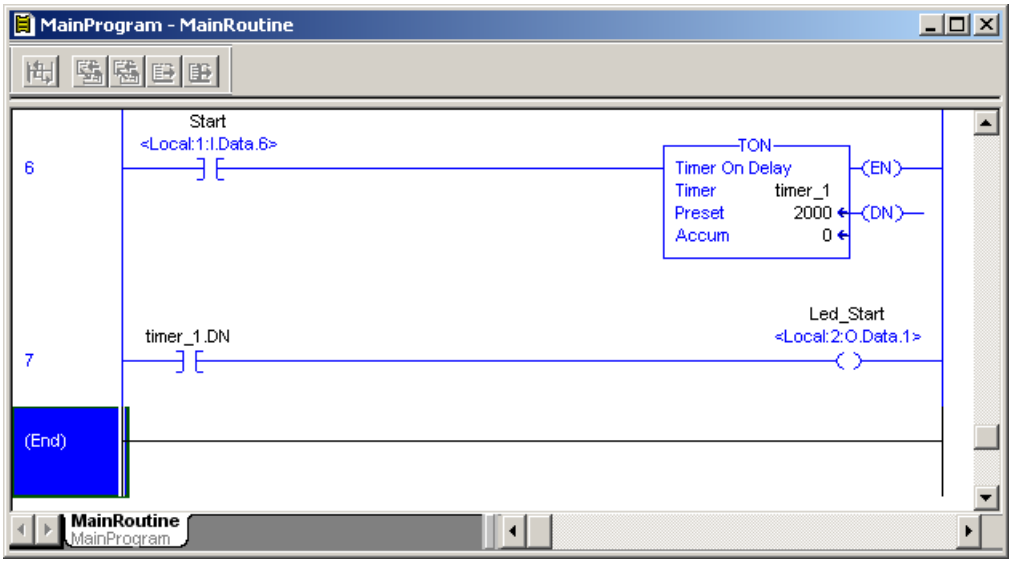
Styki, cewki wykorzystywane powyżej pozwalają na wykorzystanie zmiennych binarnych, natomiast zmienne analogowe (typu całkowitego, rzeczywistego) są przetwarzane w blokach funkcyjnych. Na rysunku 23 został pokazany program realizujący przypisanie do dwóch zmiennych konkretnych liczb *Liczba_1=20*, *Liczba_2=25*. Następnie następuje porównanie *Liczba_1<Liczba_2*, jeżeli warunek jest spełniony to tag *Led_Auto_Man* zostaje załączony. Wstawianie funkcji odbywa się przez przeciągnięcie w obszar roboczy wybranego bloku (na przykład funkcję *CMP* – komparator odnajduje się w zakładce *Compare*).

Typy zmiennych w Allen-Bradley

- *BOOL*, 1-bitowy wartość binarna {0, 1}
- *SINT*, (*Short Integer*),1-bajtowa liczba całkowita $[-128, 127]$
- *INT*, (*Integer*), 2-bajtowa liczba całkowita $[-32768, 32767]$
- *DINT*, (*Double Integer*), 4-bajtowa liczba całkowita $[-2147483648, 2147483647]$
- *REAL*, 4-bajtowa liczba zmiennoprzecinkowa
$[-3,402823E^{38}, -1,1754944E^{-38}] \cup 0 \cup [1,1754944E^{-38}, 3,402823E^{38}]$ [2]

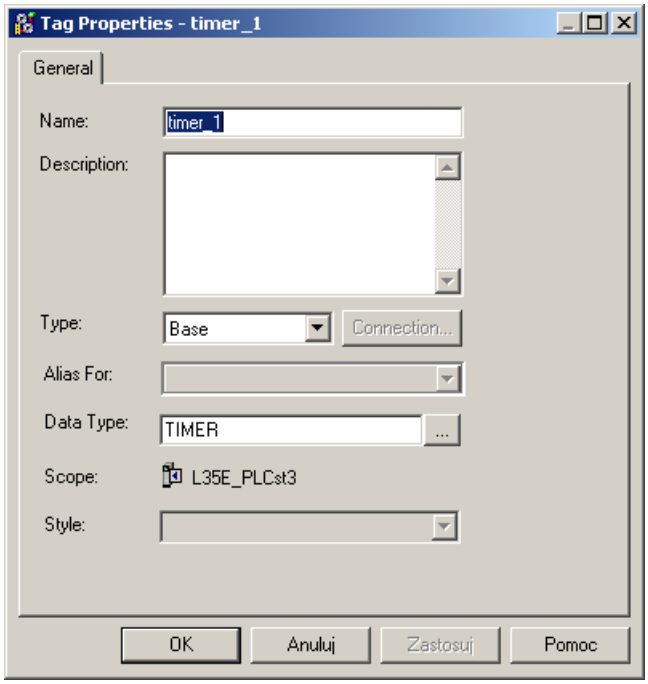
Zad. 3 Operacje na zmiennych całkowitych.
Niech zmienna *Pomiar* typu całkowitego (*int* – wartości od -32767 do 32768) przypisana jest do rzeczywistego czujnika mierzącego wysokość pewnego elementu w zakresie od 10 do $20cm$. Formalnie sygnał odzwierciedlający wysokość, przetwarzany jest na standardowy sygnał prądowy w zakresie $4..20mA$, a następnie odczytywany jest przez wejście analogowe sterownika – przyjmij błąd $\pm 0,01mA$.
Napisz funkcję która będzie zapalała *Led_yellow*, jeżeli wysokość obiektu jest większa niż 13cm.
Napisz funkcję która będzie zapalała *Led_red*, jeżeli wysokość obiektu większa niż $13cm$ – przy założeniu najgorszej ewentualności (uwzględnij błąd).

Timery i liczniki
Przykład wykorzystania funkcji Timera został pokazany na rysunku 24.



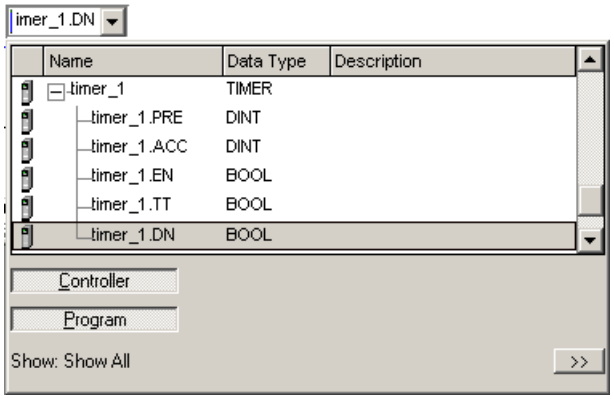
Rysunek 24. Przykład wykorzystania funkcji Timera

Aby wstawić blok *TON* należy, odnaleźć go w menu funkcyjnym i przeciągnąć do wybranego szczebla programu. W bloku *TON* (On Delay) – włączanie z opóźnieniem, widoczne są trzy parametry: *Timer*, *Preset*, *Accum*. Pierwszy z nich wskazuje na tag, związany z danym Timerem, należy zatem utworzyć tag bazowy, typu *TIMER* (por. rys. 25).



Rysunek 25. Przykład tworzenia tagu Timera

Wszystkie zmienne ukryte w strukturze danych typu *TIMER* pokazano na rys. 26.

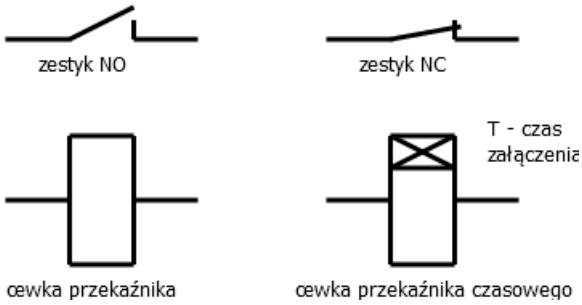


Rysunek 26. Struktura tagu Timera

Zatem w *TON* wykorzystane są następujące zmienne:
Timer_1.PRE – preset value – wartość czasu po, której timer zostanie załączony (On Delay), zmienna typu *double int* (*DINT*) wyrażona w milisekundach (ms). Zmienną należy wpisać w Polu *Preset* w bloku timera.
Timer_1.ACC – *accumulated value* (*DINT*) – Pole *Accum* w bloku timera – bieżąca wartość licznika.
Timer_1.EN – *enable bit* – wskazuje, że blok timera jest załączony.
Timer_1.TT – *timing bit* – pokazuje kiedy następuje inkrementacja licznika.
Timer_1.DN – *done bit* – załącza się gdy wartość licznika *Accum*, przekroczy *Preset*.
Zad. 4 Timery
a) Uruchom i przetestuj przykładowy program funkcję Timera *TON* (rys. 24).
Narysuj diagramy czasowe.
b) Wykorzystując dokumentacje przetestuj *Timer Off Delay* (*TOF*).
c) Zbadaj *Retentive Timer On* (*RTO*).
Zad. 5 Liczniki
Wykorzystując dokumentacje zbadaj i zaprezentuj działanie licznika *Count Up* (*CTU*).

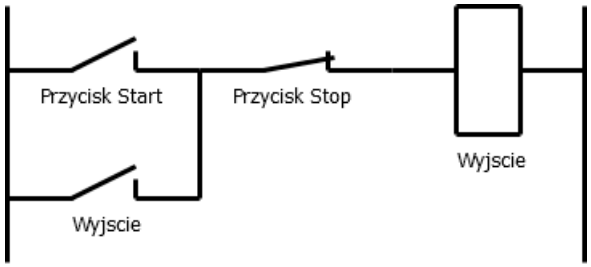
Zad. 6 Wykorzystanie poznanych funkcji
Korzystając z timerów, liczników, funkcji move i komparatorów napisz program, który będzie pokazywał jak długo był naciśnięty wybrany przycisk (wejście sterownika).
Zdefiniuj zmienne minuta, sekunda, milisekunda. pozwól także uruchamiać i resetować timer przez wybrane wejścia.

Modelowanie układów automatyki
Na rysunku 27 zaprezentowano legendę symboli wykorzystanych w dalszej części instrukcji. Są to: styk NO – normalnie otwarty (zwierny), styk NC – normalnie zamknięty, prostokąty – uzwojenie przekaźnika oraz przekaźnika czasowego.



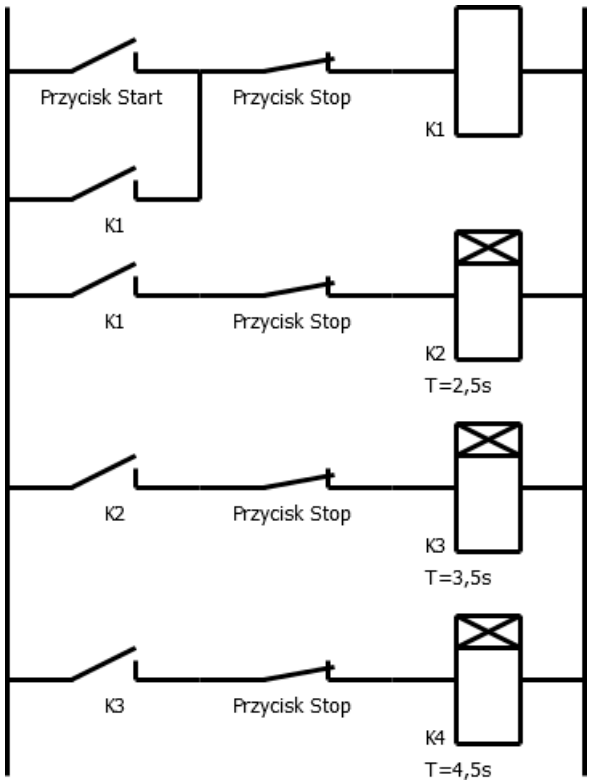
Rysunek 27. Legenda – symbolika wykorzystywana w instrukcji

Wykonaj następujące programy realizujące rzeczywiste układy przekaźnikowe.
Zad. 7 Stop awaryjny
Zrealizuj układ start ze stopem awaryjnym w języku drabinkowym (rys. 28). Dlaczego przy stopie używa się zestyków normalnie zamkniętych?



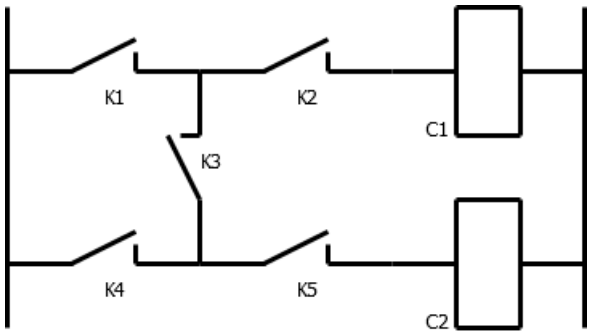
Rysunek 28. Układ start ze stopem awaryjnym

Zad. 8 Układ włączania sekwencyjnego
W przypadku większych linii produkcyjnych często zdarza się, że urządzenia napędzane są przez szereg silników, których jednoczesne uruchomienie jest nie zalecane, ze względu na przeciążenie sieci. Do takich celów stosuje się układy włączania sekwencyjnego. Zrealizuj układ z rysunku 29).
Zad. 9 Wyłączanie z opóźnieniem
Zaprojektuj układ w którym po naciśnięciu sygnału *stop* urządzenie podłączone do *K1* wyłączy się od razu, pompa *K2* po 2 *s*, a pompa *K3* po 5*s* od naciśnięcia przycisku.
Zad. 10 Układ uruchamia urządzenia z dodatkowym zabezpieczeniem
Urządzenie *X*, które będzie uruchamiane stanowi istotne zagrożenie dla środowiska. Użytkownik musi być pewien swojej decyzji uruchomienia. Zaprojektuj układ wymagający wykonania przez użytkownika specjalnej sekwencji ruchów. Na przykład aby uruchomić *X* należy przytrzymać przycisk *Start_1* przynajmniej przez 10 *s*, następnie nacisnąć 5 razy przycisk *Start_2* (podtrzymując *Start_1*), by w końcu puścić *Start_1* (podtrzymując *Start_2*), aby uruchomić urządzenie *X*.
Zad. 11 Układ wzajemnej blokady
Zrealizuj układ zasilania trzech lampek *L1*, *L2* i *L3*. Każda z nich jest zapalana jest przez włącznik odpowiednio *W1*, *W2* oraz *W3*. Przy czym układzie może palić się tylko jedna lampka, po uruchomieniu jednej z nich dwie pozostałe muszą zostać wyłączone.



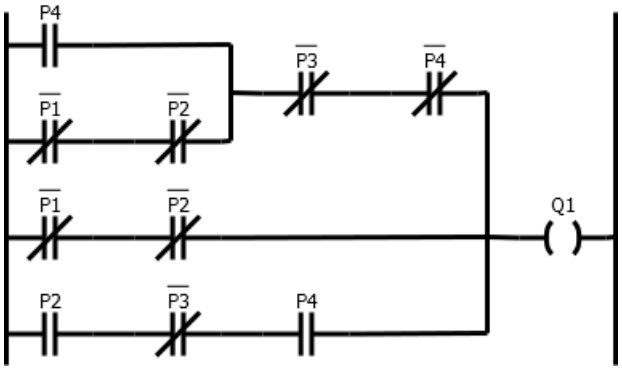
Rysunek 29. Układ włączania sekwencyjnego

Zad. 12 Układy nietrywialne – zestyk międzygałęziowy
Niektóre układy nie mogą być w sposób bezpośrednio przetransformowane do układów w języku drabinkowym. Przykładem takiego rozwiązania jest układ z zestykiem międzygałęziowym przedstawiony na rys. 30 – rozwiązania takie stosuje się w celu zaoszczędzenia styków (*K3* może zasilać dolny bądź górny szczebel układu). Zaproponuj realizację układu w języku ladder, który będzie odpowiednikiem logicznym przedstawionego układu.



Rysunek 30. Układ z zestykiem międzygałęziowym

Zad. 13 Minimalizacja funkcji logicznych
Przeanalizuj i uprość program na rysunku 31.
Najpopularniejszymi metoda analizy i minimalizacji funkcji logicznych jest *metoda tablic Karnaugha* oraz *metoda Quine-McCluskey'a*.



Rysunek 31. Układ styków do zadania 13

Literatura

[1] Allen-Bradley, *1769 CompactLogix Controllers User Manual Catalog Numbers 1769-L31, 1769-L32C, 1769-L32E, 1769-L35CR, 1769-L35E* Rockwell Automation Publication, 1769-UM011I-EN-P, February 2013.

[2] Allen-Bradley, *LOGIX 5000 Controllers General Instructions Reference Manual, 1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, Emulate 5570*, Rockwell Automation Publication, 1756-RM003S-EN-P, February 2018.

[3] Allen-Bradley, *Sterowniki Logix5000, Platformy sprzętowe 1756 ControlLogix, 1756 GuardLogix, 1768 CompactLogix, 1768 Compact GuardLogix, 1769 CompactLogix, 1789 SoftLogix, PowerFlex DriveLogix*, Publikacja 1756-QS001E-PL-P, Październik 2009.

[4] *Informator Rynkowy Automatyki, Edycja 2018* Numer 9, Wydawnictwo AVT Korporacja sp. z o. o., Warszawa, 2018

[5] Klimesz W., *nr 21, 22 Sterowniki PLC. Konfiguracja, podstawowe funckje logiczne, układy czasowe, liczniki i inne, Materiały pomocnicze dotyczące sterownika TSX37 i TSX57.* Instrukja Laboratorium Systemów Automatyki i Mechatroniki, Politechnika Wrocławska, Wrocław, 2016.

[6] *Rockwell Automation Celebrates the 100th Anniversary of the Allen-Bradley Brand* http://phx.corporate-ir.net/ dostęp 30.09.2018.

[7] Solnik W., *Tworzenie projektu w RSLogix 5000 (CompactLogix – 1769-L32E, 1769-L32C, FlexLogix5434 – 1794-L34).* Instrukja Laboratorium Systemów Automatyki i Mechatroniki, Politechnika Wrocławska.

[8] *Strona Rockwell Automation*, www.rockwellautomation.com, dostęp 30.09.2018.

[9] Tadeusiewicz R., Piwniak G.G., Tkaczow W.W., Szaruda W.G., Oprzędkiewicz K., *Modelowanie komputerowe i obliczenia współczesnych układów automatyzacji.* Wyd. AGH, 2004.

[10] Trybus L., *Regulatory wielofunkcyjne.* Wydawnictwo Naukowo-Techniczne, Warszawa, 1992.

Kontakt:
Politechnika Wrocławska
Wydział Elektroniki
Katedra Automatyki, Mechatroniki i Systemów Sterowania
Pracownia Automatyki, Modelowania i Mechatroniki

Laboratorium urządzeń i układów automatyki
tel. (+48 71) 320 39 98
bud. C-3, sala 22
ul. Janiszewskiego 11/17
50-370, Wrocław

Laboratorium Systemów Automatyki i Mechatroniki
tel. (+48 71) 320 39 95
bud. C-3, sala 21
ul. Janiszewskiego 11/17
50-370, Wrocław

Back to Contents