KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Politechnika Wrocławska

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Sterowniki Programowalne

Marcin Pawlak

„Wzrost liczby absolwentów w Politechnice Wrocławskiej na kierunkach o kluczowym znaczeniu dla gospodarki opartej na wiedzy" nr UDA-POKL.04.01.02-00-065/09-01

**Projekt współfinansowany ze środków Unii Europejskiej
w ramach Europejskiego Funduszu Społecznego**

Marcin PAWLAK

STEROWNIKI PROGRAMOWALNE

Sterowniki programowalne stanowią dzisiaj podstawowy element sterujący zautomatyzowanych procesów technologicznych, obecnych niemalże w każdej gałęzi przemysłu. Znajomość ich budowy, zasady działania oraz podstaw programowania jest niezbędna wśród osób zajmujących się zawodowo projektowaniem i obsługą techniczną nowoczesnych systemów sterowania i automatyki przemysłowej. W ostatnich latach, na rynku pracy zauważyć można wzrost zapotrzebowania na inżynierów elektryków, specjalizujących się w szeroko pojętej dziedzinie automatyki przemysłowej. Niedobór wykwalifikowanej kadry technicznej powoduje konieczność odpowiedniego, kierunkowego kształcenia na wyższych uczelniach technicznych, których absolwenci oprócz ugruntowanej wiedzy teoretycznej powinni posiadać również przygotowanie praktyczne.

Niniejszy skrypt jest przeznaczony dla studentów wyższych uczelni technicznych, odbywających studia na kierunku Automatyka i Robotyka oraz kierunkach pokrewnych. Zawarte w nim informacje teoretyczne mogą być wykorzystane jako materiał uzupełniający do wykładów i ćwiczeń laboratoryjnych z różnorodnych kursów, związanych z programowaniem sterowników PLC.

Część teoretyczna skryptu przedstawia genezę i rozwój pierwszych sterowników programowalnych oraz ich budowę i ogólną zasadę działania. Na podstawie normy IEC 61131 omówiono języki programowania sterowników oraz ich elementy wspólne, takie jak: typy danych, zmienne i bloki funkcyjne. W dalszej części skryptu przedstawiono rodzinę sterowników PLC firmy OMRON, ze szczególnym uwzględnieniem sterowników modułowych serii CJ1M, stanowiących wyposażenie Laboratorium Sterowania Urządzeń i Napędów Przemysłowych, w Instytucie Maszyn Napędów i Pomiarów Elektrycznych w Politechnice Wrocławskiej.

W drugiej części skryptu, obejmującej praktyczne aspekty programowania sterowników, zostały omówione programy narzędziowe wchodzące w skład pakietu CX-One, który stanowi podstawowe narzędzie do konfiguracji i programowania wszystkich obecnie produkowanych sterowników PLC firmy OMRON. Przedstawiono instrukcję obsługi oraz możliwości funkcjonalne środowiska CX-Programmer, w zakresie konfiguracji projektu, edycji programów w języku drabinkowym oraz uruchamiania i testowania aplikacji użytkownika. Na zakończenie przedstawiono wybrane instrukcje i funkcje języka drabinkowego sterowników OMRON oraz zasady ich użycia i parametryzacji.

Spis treści

1. Wprowadzenie

Dynamiczny rozwój przemysłu obserwowany w ciągu ostatnich lat spowodował wzrost złożoności procesów technologicznych oraz zaostrzenie wymagań w stosunku do jakości i wydajności produkcji. Ogromna konkurencja, panująca na światowym rynku wśród największych potentatów przemysłowych wymusza stosowanie coraz to nowocześniejszych, bezpieczniejszych i bardziej niezawodnych systemów sterowania i zarządzania produkcją. Powszechnie dąży się do automatyzacji przedsiębiorstw przemysłowych, polegającej na zastąpieniu lub ograniczeniu ludzkiej pracy fizycznej i umysłowej przez maszyny, wykonujące pewne powtarzalne czynności automatycznie. Realizacja tej strategii możliwa jest tylko przy zastosowaniu nowoczesnych rozwiązań z szeroko pojętej dziedziny automatyki przemysłowej, która nieustannie się rozwija.

W ubiegłym stuleciu opanowanie technologii wytwarzania półprzewodników sprawiło, że zdecydowanie najszybciej rozwijającą się dziedziną techniki stała się elektronika. W szczególności rozwój techniki cyfrowej spowodował niesamowity skok technologiczny w zakresie wytwarzania złożonych urządzeń technicznych, charakteryzujących się swego rodzaju inteligencją. Bez wątpienia największym sukcesem rozwoju tej dziedziny stało się wynalezienie mikroprocesora, który bardzo szybko stał się „sercem" wszelkich urządzeń elektronicznych powszechnego użytku, dzięki czemu z końcem ubiegłego stulecia trafił pod strzechy przeciętnego użytkownika.

Gwałtowny rozwój techniki mikroprocesorowej miał również kolosalne znaczenie dla rozwoju automatyki przemysłowej, powodując wprowadzenie programowalnej techniki cyfrowej do kombinacyjnych i sekwencyjnych układów sterowania ówczesnych procesów przemysłowych. Naturalnym następstwem tego było wynalezienie i upowszechnienie programowalnego sterownika logicznego PLC (*ang. programmable logic controller*), który dzisiaj jest podstawowym i najczęściej występującym elementem sterującym prawie każdego, nowoczesnego procesu technologicznego.

Sterownik programowalny jest uniwersalnym urządzeniem mikroprocesorowym przystosowanym do pracy w trudnych warunkach przemysłowych, gdzie jest wykorzystywany do sterowania pracą maszyn, urządzeń oraz całych ciągów technologicznych. Głównym zadaniem sterowników PLC jest realizacja zaprogramowanych algorytmów sterowania w czasie rzeczywistym, na podstawie analizy sygnałów wejściowych, pochodzących od sterowanego procesu. W wyniku reakcji na zmiany sygnałów wejściowych sterownik generuje odpowiednie sygnały

wyjściowe i przekazuje je do zewnętrznych elementów wykonawczych, tak aby uzyskać pożądane działanie sterowanego procesu.

To co wyróżnia współczesne sterowniki programowalne od pozostałych wbudowanych systemów mikroprocesorowych jest ich uniwersalność. Duże możliwości funkcjonalne jednostek centralnych pozwalają na realizację praktycznie dowolnych, złożonych algorytmów sterujących. Ponadto budowa modułowa sterowników daje możliwość elastycznej konfiguracji sprzętowej, co pozwala na dopasowanie ich do współpracy z zewnętrznymi sygnałami o określonych parametrach fizycznych oraz na wykorzystanie ich do pracy w różnych warunkach środowiskowych.

1.1. Sterownik programowalny – rys historyczny

Historia wynalezienia pierwszego sterownika programowalnego sięga roku 1968, kiedy to grupa inżynierów z firmy General Motors podjęła się realizacji projektu, którego celem było opracowanie konstrukcji uniwersalnego sterownika nowej generacji, przystosowanego do pracy w warunkach przemysłowych. Przyjęto następujące założenia projektowe:

- łatwe programowanie i przeprogramowanie w zależności od zmieniających się warunków przemysłowych,
- łatwe utrzymanie w ruchu produkcyjnym,
- budowa modułowa, zapewniająca prostą naprawę poprzez wymianę uszkodzonych modułów,
- mniejsze gabaryty i większa niezawodność w stosunku do instalacji przekaźnikowych,
- niższe koszty instalacji w stosunku do przekaźnikowych szaf sterowniczych [5,6].

Prace nad pierwszym sterownikiem programowalnym prowadzone były w Stanach Zjednoczonych równolegle przez pięć firm: Bedford Associates, General Motors, International Instruments, Digital Equipment Corporation i Struthers-Dunn Systems Division. Efektem tych prac było powstanie w 1969r. pierwszego w historii sterownika programowalnego, nazwanego Modicon 084. Posiadał on imponującą jak na owe czasy wielkość pamięci programu - 4kB, a jego konstrukcja ważyła 46kg! Oficjalnym twórcą tego sterownika był Richard Morley, założyciel firmy MODICON, której nazwa pochodzi od słów: *Modular Digital Control*.

Początkowo głównym odbiorcą sterowników programowalnych był przemysł samochodowy, który dynamicznie rozwijał się w tym okresie. Jednak już na początku lat 70-tych sterowniki PLC zdobyły ogromną popularność w przemyśle, zastępując

stopniowo przekaźnikowe układy sterowania oraz urządzenia sterowania sekwencyjnego, wykorzystujące mechaniczne układy bębnowe i krzywkowe.

W połowie lat siedemdziesiątych wprowadzono do produkcji rozproszone moduły wejść-wyjść, które umożliwiały zdalne sterowanie na odległość kilkuset metrów od głównej stacji sterownika, wykorzystując przewodowe sieci komunikacyjne. W latach osiemdziesiątych zaczęto stosować tzw. inteligentne moduły rozszerzeń, które posiadały własne procesory i umożliwiały realizację złożonych funkcji obliczeniowych. Moda na sterowniki programowalne opanowała szybko cały świat, co zaowocowało zwiększeniem konkurencji wśród producentów, prowadząc pośrednio do obniżenia ich cen. Do wzrostu popularności sterowników przyczyniły się szczególnie firmy japońskie, które wprowadziły do oferty małe, kompaktowe sterowniki programowalne o dużych możliwościach funkcjonalnych, a przy tym były znacznie tańsze od sterowników innych producentów.

W latach 90-tych sterowniki PLC były już powszechnie stosowane praktycznie we wszystkich gałęziach przemysłu, zastępując niemalże całkowicie układy sterowania przekaźnikowego czy analogowe układy regulatorów przemysłowych. Wzrastająca popularność komputerów PC oraz dynamiczny rozwój oprogramowania pozwoliły na rozwój możliwości komunikacyjnych sterowników oraz ich integrację z komputerowymi systemami wymiany i analizy danych. Powstały pierwsze systemy sterowania nadrzędnego i gromadzenia danych SCADA (ang. *Supervisory Control and Data Acquisition*), które znacznie rozszerzyły możliwości ówczesnych systemów sterowania, zapewniając kontrolowany przepływ danych procesowych pomiędzy poszczególnymi warstwami systemu sterowania [5, 6]. Nowoczesne oprogramowanie typu SCADA oferuje następujące funkcje:

- wizualizacja stanu i przebiegu procesu przemysłowego,
- zdalne sterowanie przebiegiem procesu i jego parametryzacja,
- diagnostyka procesu - wyświetlanie ostrzeżeń i sygnalizacja stanów alarmowych,
- wsparcie działań operatora procesu (proponowanie działania, system podpowiedzi),
- obserwacja i analiza krytycznych zmiennych procesowych,
- archiwizacja danych procesowych,
- opracowanie raportów, zestawień, wydruków.

W ostatnim dziesięcioleciu, szczególnie w branży automatyki procesowej (przemysł chemiczny, spożywczy, farmaceutyczny...) coraz częściej obserwuje się trend zastępowania tradycyjnych, skupionych systemów sterowania, systemami rozproszonymi DCS (ang. *Distributed Control System*). Systemy DCS zawierają podstawowe elementy automatyki (sterowniki programowalne, przemysłowe sieci komunikacyjne, systemy wizualizacji) i jednocześnie stanowią integralną całość, gdyż posiadają wspólną bazę zmiennych procesowych, wykorzystywanych zarówno do sterowania jak i wizualizacji. W systemach DCS powszechnie stosuje się szybkie,

rozbudowane sterowniki PLC pracujące pod kontrolą systemu operacyjnego czasu rzeczywistego (ang. *RTOS – Real Time Operating System*), które nadzorują deterministyczne wykonanie wszystkich zadań procesowych oraz potrafią obsłużyć nawet kilka tysięcy zmiennych procesowych. Oprócz tego, w systemach tych szczególnie duży nacisk kładzie się na niezawodność, poprzez stosowanie redundancji sprzętowej i komunikacyjnej (np. dwa równolegle pracujące sterowniki, wielokrotne moduły I/O, podwójny ring światłowodowy, itp.). Ogólnie zasada jest prosta – system sterowania musi pracować niezawodnie, nie ma prawa „zawiesić się", jeżeli awarii ulegnie sterownik, sieć komunikacyjna lub nawet jeśli pojawią się błędy programowe.

Rozproszone systemy sterowania DCS coraz częściej stanowią pomost pomiędzy procesem technologicznym a aplikacjami z branży IT (Technologia Informacyjna - ang. *Information Technology*). Integracja IT obejmuje aplikacje z zakresu:

- **ERP** (ang. *Enterprice Resource Planning*) – zarządzanie zamówieniami, logistyka, finanse;
- **MES** (ang. *Manufacturing Execution System*) – zarządzanie produkcją z analizą i optymalizacją przepływu materiałów, utrzymanie ruchu, zarządzanie przepływem dokumentów.

Reasumując, można stwierdzić, że integracja systemów sterowania z nadrzędnymi komputerowymi systemami zarządzania produkcją spowodowała znaczne rozszerzenie funkcjonalne współczesnych systemów automatyki, do zadań których można zaliczyć:

- sterowanie procesem technologicznym,
- diagnostyka procesu i jego ocena technologiczna,
- analiza ekonomiczna przedsiębiorstwa,
- planowanie zasobów produkcyjnych,
- optymalizacja produkcji.

1.2. Podział sterowników programowalnych

Aktualnie w branży automatyki przemysłowej spotkać można kilkudziesięciu czołowych producentów tych urządzeń, oferujących sterowniki programowalne o różnych możliwościach funkcjonalnych. W celu zapewnienia kompatybilności pomiędzy różnymi modelami sterowników pochodzących od jednego producenta, przyjęto koncepcję rodzin sterowników. Polega ona na podobnym projektowaniu wszystkich modeli sterowników z danej rodziny, co pozwala na lepszy dobór sprzętu, stosownie do wielkości projektowanego systemu sterowania. Poszczególne modele sterowników wchodzące w skład rodziny charakteryzują się następującymi, wspólnymi cechami:

- mogą być programowane w tym samym pakiecie programowym używając tego samego języka programowania,
- posiadają takie same zmienne programowe oraz podobną strukturę modułów rozszerzeń,
- istnieje możliwość bezproblemowego przenoszenia programów pomiędzy modelami.

Jednym z kryteriów podziału sterowników w obrębie rodziny jest ich wielkość, często utożsamiana z ilością dostępnej pamięci oraz liczbą obsługiwanych wejść/wyjść. Mimo, że nie istnieją formalne definicje, które jednoznacznie kwalifikują dany sterownik do określonej grupy, biorąc pod uwagę kryterium wielkości, według [3] sterowniki PLC można podzielić na:

- **małe** – obsługujące do 128 wejść/wyjść I/O (ang. *Input/Output*) i posiadające pamięć programu do 2kB;
- **średnie** – obsługujące do 512 I/O i posiadające pamięć programu do 16kB;
- **duże** – obsługujące do 4096 I/O i posiadające pamięć do 96kB.

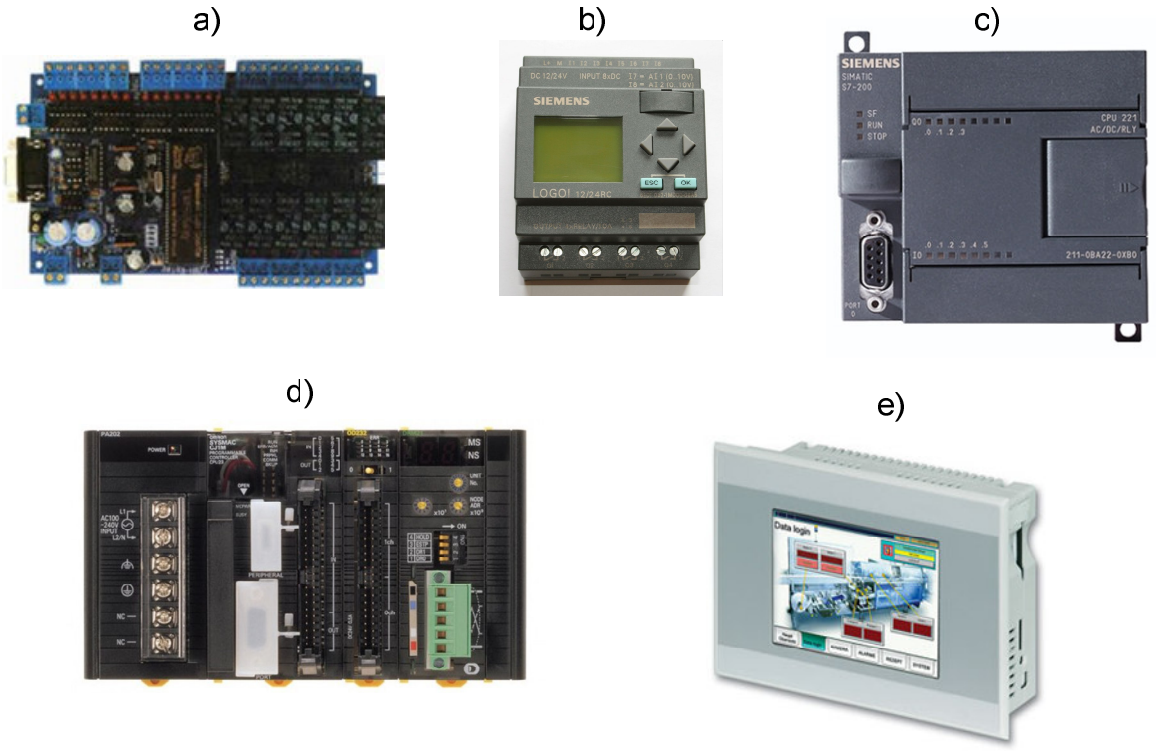
Przedstawione przedziały wielkości sterowników obowiązywały w latach 90-tych. Jest rzeczą oczywistą, że biorąc pod uwagę dynamiczny rozwój elektroniki oraz postęp technologiczny w dziedzinie wytwarzania półprzewodników, współczesne sterowniki programowalne będą posiadały ilość pamięci znacznie przekraczającą przedstawione wyżej wartości.

Sterowniki PLC można podzielić również ze względu na ich cechy konstrukcyjne, takie jak rodzaj obudowy czy możliwości instalacji dodatkowych modułów rozszerzeń. Według tych kryteriów spotkać można sterowniki:

- **bez obudowy** (ang. *Open Frame*) – stosowane jako tzw. wbudowane systemy sterowania (ang. *Embedded Control System*), przeznaczone do instalacji wewnątrz maszyn i urządzeń. Przykład: sterownik Triangle Research T28H (Rys.1.1a).
- **kompaktowe** – o prostej budowie i zazwyczaj małych wymiarach. Konstrukcja sterownika integruje w jednej obudowie: zasilacz, jednostkę centralną oraz moduły wejść i wyjść. Przykład: sterownik SIMATIC LOGO (Rys.1.1b).
- **kompaktowe rozbudowywalne** – o budowie kompaktowej, z możliwością instalowania dodatkowych modułów rozszerzeń. Przykład: SIMATIC S7-200 (Rys.1.1c).
- **modułowe** – występujące najczęściej w formie średnich i dużych sterowników. Charakteryzują się elastyczną konstrukcją, w której własności funkcjonalne użytkownik sam konfiguruje poprzez dobór odpowiednich modułów, takich jak: jednostka centralna, moduły wejść-wyjść, moduły komunikacyjne, moduły specjalne. Poszczególne moduły rozszerzeń instaluje się w specjalnych kasetach-panelach (ang. *rack*). Przykład: sterownik OMRON CJ1M (Rys.1.1d).

- **zintegrowane z panelem operatorskim** – w obudowie panelu operatorskiego znajduje się jednostka centralna z interfejsami komunikacyjnymi oraz moduły wejść-wyjść. Zaciski podłączeniowe znajdują się zazwyczaj z tyłu sterownika, bądź dołączane są specjalne kasety rozszerzeń, w których instaluje się moduły I/O. Przykład: sterownik Moeller XV200 (Rys.1.1e).

Na rysunku 1.1 przedstawiono fotografie przykładowych sterowników programowalnych.



Rys.1.1. Fotografie przykładowych sterowników PLC: a) – Triangle Research T28H, b) – SIMATIC LOGO 12/24RC, c) – SIMATIC S7-200, d) – OMRON CJ1M, e) – Moeller XV200.

Ostatnio, oprócz wyżej wymienionych typów sterowników programowalnych, w ofertach handlowych wielu producentów urządzeń automatyki coraz częściej można spotkać małe sterowniki logiczne, zaliczane do grupy tzw. przekaźników programowalnych, niekiedy zwanych przekaźnikami inteligentnymi. Są to kompaktowe urządzenia sterujące, o ograniczonych możliwościach programowych, obsługujące zazwyczaj niewielką liczbę wejść/wyjść, które są przeznaczone do montażu w typowych instalacjach automatyki domowej. Niewątpliwie największym atutem tych urządzeń jest bardzo niska cena (nawet poniżej 200zł!), która zawiera koszt samego sterownika i oprogramowania projektowego. Zakres zastosowań tych sterowników najczęściej obejmuje typowe aplikacje sterowania układów napędowych oraz aplikacje sterujące urządzeń automatyki domowej:

- automatyczny rozruch silnika indukcyjnego w układzie gwiazda-trójkąt,
- sterowanie pracą silnika asynchronicznego w układzie nawrotnym,
- sterowanie oświetlenia klatki schodowej,
- sterowanie napędu bramy wjazdowej/garażowej,
- sterowanie czasowe napędów rolet i żaluzji,
- automatyka ogrodowa (zmierzchowe sterowniki oświetlenia, podlewanie grządek, sterowanie fontannami ogrodowymi, systemy alarmowe, itp.).

1.3. Dobór sterownika PLC do systemu sterowania

Ze względu na ogromną liczbę dostępnych sterowników programowalnych wybór odpowiedniego modelu do nowoprojektowanego systemu sterowania nie jest zadaniem łatwym. Rozwiązanie tego problemu należy rozpocząć od szczegółowej analizy zagadnienia, aby dokładnie określić wymagania procesu technologicznego.

Pierwszym etapem jest sformułowanie zadania sterowania. Jednym z elementów procesu automatyzacji powinno być możliwie dokładne opracowanie opisu przebiegu sterowanego procesu technologicznego. Automatyk powinien poznać dokładnie zasady działania poszczególnych maszyn i urządzeń i orientować się w przebiegu tego procesu. Na tym etapie niezbędna jest ścisła współpraca z technologiem produkcji. Spośród wielu możliwych wariantów realizacji zadania sterowania należy wybrać rozwiązanie optymalne, wynikające między innymi z analizy ekonomicznej oraz możliwości praktycznej realizacji w danym środowisku. W przypadku skomplikowanych procesów technologicznych należy spróbować rozłożyć problem na mniejsze, niezależne od siebie zadania sterowania. Dla poszczególnych zadań należy sformułować algorytmy sterowania i na ich podstawie dobrać rodzaj sterownika oraz innych elementów automatyki. Należy przy tym pamiętać o konieczności zachowania integralności procesu sterowania, co wymaga zapewnienia komunikacji i synchronizacji poszczególnych urządzeń sterujących [9].

Jednym z ważniejszych etapów projektowania systemu automatyzacji jest opracowanie algorytmu sterowania całego procesu technologicznego. Dobry algorytm sterowania zapewnia skuteczne i bezpieczne działanie układu sterowania w każdych warunkach, nawet w stanach awaryjnych. Znajomość algorytmu sterowania pozwala wstępnie określić wymagania dla jednostki centralnej sterownika, od której zależy między innym szybkość wykonania programu użytkownika. Do realizacji prostych algorytmów dyskretnych zazwyczaj wystarczy zastosowanie sterowników o budowie kompaktowej, dobranych tak, aby spełnić wymagania w zakresie obsługi odpowiedniej liczby sygnałów I/O. W przypadku złożonych procesów technologicznych niezbędne jest stosowanie sterowników o większych możliwościach, wyposażonych w wydajne mikroprocesory oraz w odpowiednio dużą pamięć. To właśnie parametry zastosowanego mikroprocesora w danym modelu

sterownika decydują o szybkości działania sterownika, liczbie obsługiwanych modułów rozszerzeń, wielkości dostępnej pamięci dla programu sterującego oraz możliwościach współpracy z systemami komunikacyjnymi. Dlatego też wybór odpowiedniej jednostki CPU sterownika jest bardzo ważny i powinien być przeprowadzony rozważnie z uwzględnieniem wielu czynników.

Kolejnym etapem jest opracowanie listy wszystkich zmiennych procesowych, która powinna zawierać spis i specyfikację wszystkich sygnałów wejściowych i wyjściowych z podziałem na sygnały analogowe, dyskretne i specjalne. Na podstawie tej listy należy dobrać wymaganą liczbę modułów I/O dla sterownika PLC.

Ostatecznie, przy doborze sterownika programowalnego do procesu technologicznego należy wziąć dodatkowo pod uwagę następujące czynniki:

- **interfejsy komunikacyjne** – należy określić typ, parametry i topologię zastosowanych sieci komunikacyjnych, które integrują poszczególne elementy systemu sterowania i wymieniają dane z innymi urządzeniami,
- **wizualizacja procesu** – jeżeli jest wymagana, należy wybrać interfejs człowiek-maszyna (ang. HMI – Human - Machine Interface) i dopasować urządzenia sprzętowe (panele operatorskie, tablice synoptyczne) oraz oprogramowanie typu SCADA,
- **rozbudowa systemu** – jeżeli w przyszłości jest planowana rozbudowa systemu sterowania, rozszerzenie jego możliwości funkcjonalnych lub integracja z innymi nadrzędnymi systemami sterowania, warto na etapie projektowym to przewidzieć i zastosować sprzęt, który tą rozbudowę umożliwi.

2. Budowa i zasada działania sterowników PLC

2.1. Ogólna zasada działania sterowników programowalnych

Sterownik programowalny pracuje według programu użytkownika zapisanego w jego pamięci, zawierającego ciąg rozkazów logicznych, sterujących pracą urządzenia. Program sterujący stanowi zapis algorytmu sterowania w postaci listy rozkazów, wykonywanych kolejno jeden po drugim. Pojedynczy rozkaz jest najmniejszą częścią programu sterującego i składa się z operacji, określającej rodzaj wykonywanej funkcji oraz argumentów, określających sygnały wejściowe i wyjściowe sterownika powiązane ze sobą funkcjami logicznymi.

Program sterujący opracowany przez użytkownika przy pomocy komputera lub programatora jest przesyłany i zapisany w jego pamięci w postaci zakodowanych liczbowo rozkazów oraz ich argumentów. Rozkazy te są następnie przetwarzane przez system mikroprocesorowy sterownika, który na ich podstawie wykonuje odpowiednie działanie, takie jak: operacje logiczne, operacje arytmetyczne, odczyt i zapis danych z/do pamięci [9].

2.1.1. CYKL PRACY STEROWNIKA

W odróżnieniu od klasycznego systemu mikroprocesorowego, sterowniki programowalne pracują według określonego cyklu, którego schemat ogólny przedstawiono na rysunku 2.1. W cyklu pracy sterownika PLC można wyróżnić kilka charakterystycznych etapów, które są powtarzane w nieskończonej pętli. Zasada pracy cyklicznej dotyczy wszystkich sterowników programowalnych, niezależnie od ich budowy i producenta. Poznanie i zrozumienie tej zasady jest niezbędne do dalszej nauki programowania sterowników programowalnych. Poniżej zostaną omówione poszczególne fazy cyklu pracy sterownika [9].

a) Inicjalizacja sterownika

W fazie inicjalizacji sterownik testuje poprawność działania swoich obwodów wewnętrznych i sprawdza poprawność konfiguracji sprzętowej wszystkich podzespołów i modułów rozszerzeń.

b) Odczyt sygnałów wejściowych sterownika

W tej fazie następuje odczytanie stanu wszystkich fizycznych wejść sterownika i zapisanie ich w specjalnym obszarze pamięci, zwanym obrazem wejść procesu *PII*

(ang. *Process-Image Input*). Należy zwrócić uwagę, że odczyt wejść sterownika jest wykonywany jednorazowo w każdym cyklu pracy sterownika. Oznacza to, że jeżeli stan sygnału wejściowego zmieni się w fazie wykonywania programu, to zmiany te będą uwzględnione dopiero w następnym cyklu pracy sterownika.

c) Wykonanie programu użytkownika

W fazie wykonania programu użytkownika przetwarzane są kolejno instrukcje programu sterującego, na podstawie których mikroprocesor sterownika wykonuje odpowiednie działania i zapisuje wyniki obliczeń w pamięci danych. Po wykonaniu wszystkich instrukcji programu zostaje wygenerowany nowy stan wyjść, który zostaje zapisany w specjalnym obszarze pamięci, nazywanym obrazem wyjść procesu *PIO* (ang. *Process-Image Output*).

Podstawową zasadą pracy sterowników PLC jest działanie sekwencyjne, które polega na tym, że poszczególne rozkazy wykonywane są kolejno po sobie. W związku z tym nie ma możliwości wykonania dwóch rozkazów jednocześnie. Dlatego też pisząc program należy uwzględnić odpowiednią kolejność obróbki sygnałów. Należy pamiętać także o zasadzie, że przypisanie wartości do danej zmiennej wyjściowej powinno wystąpić **tylko w jednym miejscu programu**, gdyż ostateczna wartość tej zmiennej będzie zależała tylko i wyłącznie od ostatnio wykonanej instrukcji przypisania.

d) Zapis sygnałów wyjściowych sterownika

W fazie tej następuje pobranie wartości stanu wyjść z obszaru pamięci obrazu wyjść procesu (PIO) i zapisanie ich do fizycznych portów wyjściowych sterownika. W wyniku tej operacji wyjścia sterownika zostają uaktualnione i przyjmują wartości obliczone na podstawie algorytmu programu sterującego, wykonanego w danym cyklu pracy sterownika. Należy pamiętać, że aktualizacja sygnałów wyjściowych jest wykonywano tylko jeden raz, w każdym cyklu.

e) Obsługa komunikacji

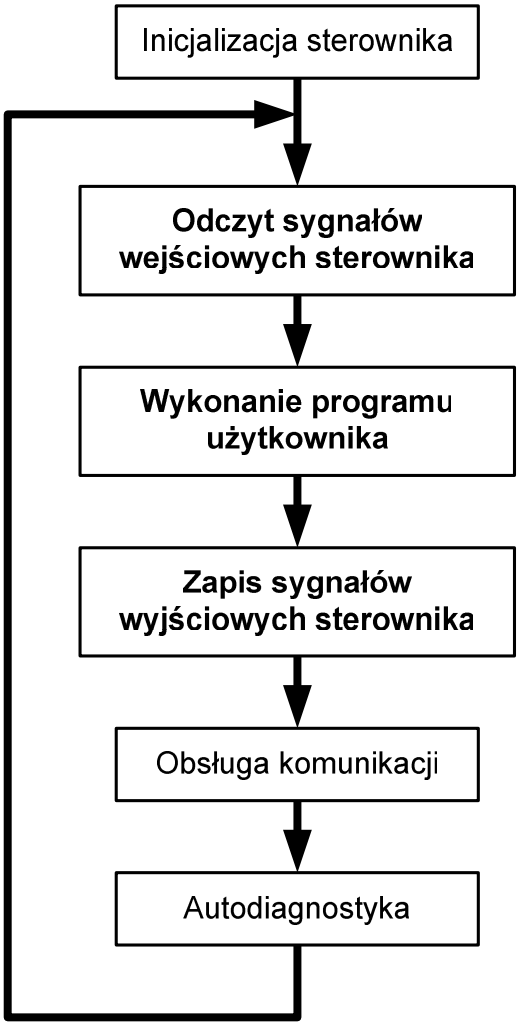
W przypadku, gdy sterownik programowalny wykorzystuje komunikację sieciową pomiędzy zewnętrznymi elementami systemu sterowania, takimi jak inne sterowniki PLC czy panele operatorskie, w fazie tej następuje obsługa dwukierunkowej transmisji danych. W zależności od zastosowanych rozwiązań sprzętowych, procesor sterownika wymienia dane z modułami komunikacyjnymi, które przejmują kontrolę nad prawidłowym przebiegiem transmisji.

W fazie tej odbywa się również obsługa programatora. Sterownik sprawdza, czy do systemu jest dołączone urządzenie programujące (np. komputer) i w razie konieczności odczytuje przychodzące komendy sterujące. Do najczęściej występujących komend sterujących można zaliczyć komendy przełączenia trybu pracy sterownika (ang. *RUN / STOP*), czy też komendy odczytu (ang. *Upload*) i zapisu (ang. *Download*) programu użytkownika. Załadowanie nowej zawartości programu

użytkownika do sterownika odbywa się zazwyczaj po uprzednim zatrzymaniu jego pracy (przełączenie w tryb STOP), choć niekiedy możliwe jest również podczas normalnej jego pracy (*on-line*).

f) Autodiagnostyka

W fazie autodiagnostyki sterownik przeprowadza kontrolę poprawności działania podstawowych jego podzespołów. Sprawdza między innymi: wartość napięcia zasilania systemu, stan baterii podtrzymującej zawartość pamięci czy też pojawienie się błędów systemowych podczas realizacji programu. Na podstawie wyników tych testów zostaje uaktualniony status sterownika, który jest wyświetlany w postaci diod świecących umieszczonych zwykle na panelu czołowym sterownika. Jeżeli podczas pracy sterownika pojawią się krytyczne błędy systemowe, program sterownika zostaje niezwłocznie zatrzymany, a wszystkie wyjścia zostają automatycznie wyłączone, co zapobiega niekontrolowanemu załączeniu zewnętrznych urządzeń wykonawczych.



Rys.2.1. Cykl pracy sterownika PLC.

2.1.2. TRYBY PRACY STEROWNIKA PLC

Sterownik programowalny może znajdować się w różnych trybach pracy, które jednoznacznie określają jego działanie:

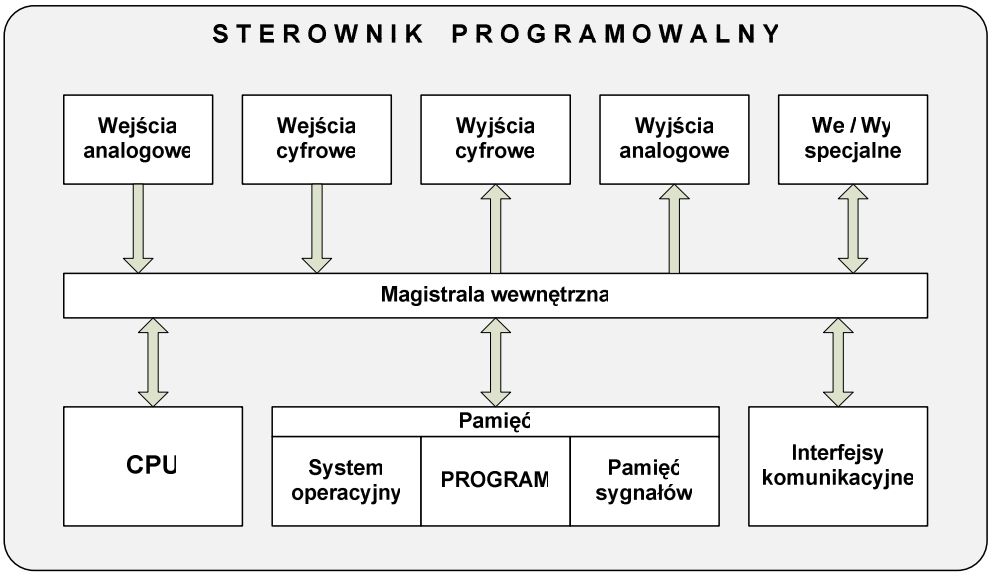
- **Tryb wykonywania programu (RUN)** – jest to właściwy tryb pracy sterownika, który odbywa się w trakcie normalnej realizacji programu aplikacyjnego użytkownika. W tym trybie realizowane są wszystkie fazy cyklu pracy sterownika i jednocześnie zablokowane są możliwości modyfikacji zawartości programu.
- **Tryb zatrzymania sterownika (STOP)** – jest to tryb, w którym sterownik nie wykonuje programu sterowania, ale jest gotowy do przyjęcia wszelkich komend sterujących od programatora. W tym trybie możliwa jest zmiana zawartości programu użytkownika (programowanie), jak i możliwe jest sterowanie wyjściami sterownika, poprzez wymuszanie odpowiednich stanów logicznych.
- **Tryb monitorowania (MONITOR)** – tryb ten umożliwia wykonywanie programu aplikacyjnego użytkownika i pozwala jednocześnie na dostęp do pamięci sterownika. Dzięki temu użytkownik może testować poprawność działania swojej aplikacji, poprzez podgląd wartości charakterystycznych zmiennych procesowych oraz możliwość wymuszania stanów logicznych na wyjściach sterownika. W trybie tym możliwa jest również edycja większości obszarów pamięci sterownika (zmiana wartości timerów, liczników, pamięci I/O. itp.). Tryb MONITOR jest szczególnie przydatny w fazie uruchamiania systemu sterowania.

2.2. Budowa sterowników PLC

Sterowniki programowalne można podzielić ze względu na ich architekturę na dwie podstawowe grupy [5]:

- **sterowniki kompaktowe**, należące do klasy małych sterowników o sztywnej architekturze, których cechą konstrukcyjną jest integracja wszystkich niezbędnych do działania podzespołów funkcjonalnych w jednej obudowie;
- **sterowniki modułowe**, należące do klasy średnich i dużych sterowników o elastycznej architekturze, w których własności funkcjonalne użytkownik sam konfiguruje poprzez dobór odpowiednich modułów funkcjonalnych, instalowanych w specjalnych kasetach lub na szynie montażowej.

Na rysunku 2.2 przedstawiono ogólny schemat budowy typowego sterownika programowalnego o budowie kompaktowej.



Rys.2.2. Schemat budowy typowego kompaktowego sterownika PLC.

Podstawowa konfiguracja sterownika modułowego zawiera następujące elementy (moduły) funkcjonalne:

- zasilacz,
- jednostkę centralną,
- moduł wejść cyfrowych,
- moduł wyjść cyfrowych,
- moduł wejść analogowych,
- moduł wyjść analogowych,
- moduły komunikacyjne,
- moduły specjalne.

2.2.1. ZASILACZ

Sterowniki programowalne na ogół przystosowane są do zasilania napięciem stałym o wartości 24V. Spotyka się również konstrukcje, przystosowane do zasilania bezpośrednio z sieci energetycznej o napięciu 110VAC lub 230VAC. W zależności od konstrukcji sterownika zasilacz PS (ang. *Power Supply*) może być zainstalowany wewnątrz obudowy sterownika (sterowniki kompaktowe, przekaźniki programowalne) lub stanowić odrębny element w sterownikach o budowie modułowej.

Do zasilania sterowników programowalnych wykorzystuje się zazwyczaj wysokiej jakości zasilacze impulsowe, które charakteryzują się następującymi cechami:

- wysoka sprawność przetwornicy AC/DC,
- stosunkowo duża moc (>100W) przy małych gabarytach,

- dobra stabilizacja wartości napięcia wyjściowego w szerokim zakresie zmian napięcia sieci zasilającej,
- wbudowane zabezpieczenia przeciążeniowe oraz przepięciowe,
- możliwość pracy w trudnych przemysłowych warunkach,
- spełniają rygorystyczne normy w zakresie kompatybilności elektromagnetycznej.

Zasilacze impulsowe spełniające powyższe parametry są urządzeniami dość kosztownymi. Jednak konstrukcje sterowników modułowych bardzo często wymagają stosowania zasilaczy dedykowanych, co praktycznie uniemożliwia zastosowanie tańszych zamienników, pochodzących od innych producentów.

2.2.2. JEDNOSTKA CENTRALNA I PAMIĘĆ STEROWNIKA

Jednostka centralna – CPU (ang. *Central Processing Unit*) jest głównym podzespołem konstrukcji każdego sterownika PLC. Do zadań jednostki centralnej należy:

- realizacja programu sterującego użytkownika,
- zarządzanie pracą całego sterownika,
- obsługa modułów rozszerzeń,
- komunikacja z programatorem oraz opcjonalnie z innymi elementami systemu sterowania.

Podstawowym elementem budowy jednostki centralnej jest mikroprocesor. Współcześnie produkowane sterowniki PLC wykorzystują mikroprocesory 8-, 16-, a nawet 32-bitowe. To właśnie rodzaj i parametry zastosowanego mikroprocesora bezpośrednio decydują o możliwościach funkcyjnych i parametrach sterownika, spośród których najważniejsze to:

- szybkość działania sterownika, od której zależy szybkość przetwarzania programu użytkownika. Parametrem charakteryzującym szybkość jednostki centralnej jest czas cyklu programowego, wyrażający np. liczbę wykonanych instrukcji przypadająca na jednostkę czasu (np. 1000 instrukcji bitowych wykonanych w ciągu 1ms);
- rozmiar pamięci przeznaczonej dla programu i danych użytkownika (np. 16kB, 512kB, 2MB);
- maksymalna liczba obsługiwanych zmiennych I/O;
- maksymalna liczba dołączanych modułów rozszerzeń;
- możliwości komunikacyjne (rodzaje obsługiwanych sieci, np. Profibus DP, DeviceNet);
- możliwość korzystania z funkcji specjalnych, np. operacji zmiennoprzecinkowych [5, 10].

Od typu zastosowanej jednostki centralnej CPU zależy również rodzaj pamięci, wykorzystywanej do przechowywania programu sterującego. Spośród najczęściej stosowanych pamięci półprzewodnikowych wyróżnić można:

- **RAM** (ang. *Random Access Memory*) – pamięć o swobodnym dostępie, która charakteryzuje się dużą szybkością wykonywanych operacji zapisu i odczytu. Stosowana jest w roli pamięci operacyjnej, do przechowywania danych i programu użytkownika. Z uwagę na utratę zawartości zapisanych danych po wyłączeniu zasilania, pamięć ta wymaga podtrzymania bateryjnego;
- **EEPROM** (ang. *Electrically Erasable Programmable Read-Only Memory*) – pamięć stała tylko do odczytu, programowalna i kasowalna elektrycznie. Właściwości fizyczne tej pamięci pozwalają na przechowywanie programu i ważnych danych procesu po zaniku zasilania, bez potrzeby stosowania podtrzymania bateryjnego. Czasami jest wykorzystywana jako pamięć typu backup, do tworzenia kopii zapasowej aplikacji użytkownika;
- **FLASH EPROM** (ang. *Flash Erasable Programmable Read-Only Memory*) – szybka pamięć stała programowalna i kasowalna elektrycznie. Stosowana w nowszych sterownikach do przechowywania programu sterującego oraz krytycznych danych procesowych, wymagających zachowania także po wyłączeniu zasilania sterownika. Oprócz tego wykorzystywana jest jako pamięć typu backup oraz jako nośnik do przenoszenia/kopiowania danych z jednego sterownika na drugi (popularne karty Compact Flash, SD, MMC itp.).

2.2.3. MODUŁY WEJŚĆ CYFROWYCH

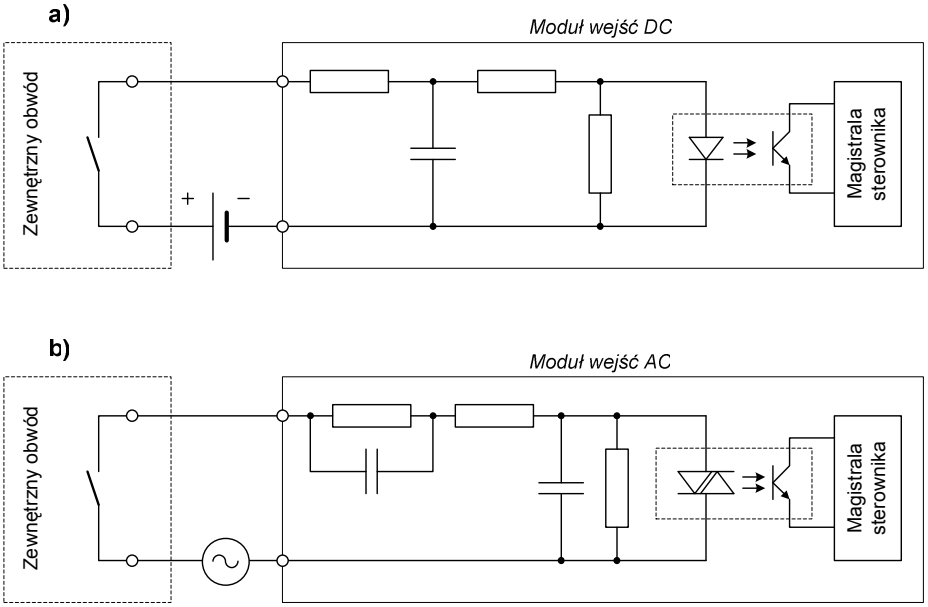
Moduły wejść cyfrowych (nazywanych sygnałami dyskretnymi, czy też logicznymi) pełnią rolę podstawowych obwodów wejściowych sterownika i stanowią interfejs pomiędzy dwustanowymi elementami zewnętrznego systemu sterowania a jednostką centralną sterownika. Moduły te zmieniają sygnały prądu stałego lub przemiennego pochodzące od zewnętrznych urządzeń na sygnały cyfrowe o poziomach napięciowych dopasowanych do elektronicznych obwodów wejściowych jednostki centralnej sterownika.

Obwody elektryczne modułów wejść cyfrowych występują w dwóch podstawowych konfiguracjach:

- **zasilane prądem stałym** – najczęściej akceptują sygnały wejściowe o napięciach od 0V (logiczne „0”) do +24V (logiczna „1”)
- **zasilane prądem przemiennym** – dostosowane do poziomów napięć sieci energetycznych 120/240V AC.

Zdecydowana większość rozwiązań sprzętowych sterowników stosuje separację galwaniczną pomiędzy wszystkimi obwodami zewnętrznymi a wewnętrzną magistralą

sterownika. Dlatego też najczęściej stosowanym elementami w obwodach wejść cyfrowych są transoptory (dla prądu stałego) i optoizolatory z wejściem bipolarnym (dla prądu przemiennego). Elementy te pełną funkcję izolowanych przekaźników sygnałów logicznych. Na rysunku 2.3 przedstawiono schematy ideowe dla pojedynczego wejścia cyfrowego, zasilanego prądem stałym (a) i przemiennym (b) [1]. Na schemacie (a) styk zewnętrznego obwodu (np. czujnik krańcowy) zasilany ze źródła prądu stałego dostarcza napięcie do modułu wejść DC, które po przejściu przez filtr dolnoprzepustowy RC zasila diodę świecącą transoptora, co powoduje wysterowanie fototranzystora i przekazanie sygnału logicznego na magistralę sterownika. Analogicznie, na schemacie (b), styk zewnętrznego obwodu zasilany ze źródła napięcia przemiennego, podaje napięcie na wejście optoizolatora, powodując wysterowanie fototranzystora i przekazanie sygnału logicznego na magistralę sterownika.



Rys.2.3. Schematy ideowe pojedynczego wejścia cyfrowego dla obwodów prądu stałego (a) i przemiennego (b).

Przy doborze modułu wejść dyskretnych do systemu sterowania należy wziąć pod uwagę następujące jego parametry:

- liczba wejść cyfrowych modułu (np. 8, 16, 32),
- napięcie znamionowe (np. 24V DC, 230V AC),
- napięcie przebicia izolacji,
- poziomy napięć wejściowych dla stanów załączenia (ON) i wyłączenia (OFF),
- charakterystyki czasowe wejść,
- prąd wejściowy (niezbędny do wysterowania wewnętrznego optoizolatora),
- środowiskowe warunki pracy [5, 9].

2.2.4. MODUŁY WYJŚĆ CYFROWYCH

Moduły wyjść cyfrowych służą do wyprowadzenia dyskretnych sygnałów sterownika do zewnętrznych obwodów wykonawczych sterowanego obiektu. Do wyjść cyfrowych sterownika najczęściej podłącza się cewki przekaźników i styczników, które sterują pracą maszyn i urządzeń obsługiwanego procesu technologicznego. Ze względu na rodzaj zastosowanego elementu sterującego wyjścia cyfrowe sterowników programowalnych można podzielić na dwie główne grupy:

- wyjścia przekaźnikowe (stykowe),
- wyjścia półprzewodnikowe.

Wyjścia przekaźnikowe posiadają wyprowadzone zaciski dla styków zwiernych NO (ang. *Normally Open*) i rozwiernych NC (ang. *Normally Closed*), które służą do załączania i rozłączania zewnętrznych obwodów elektrycznych prądu stałego i przemiennego. Ze względu na dość dobrą izolację napięciową (>3kV), obwody te mogą znajdować się na innym potencjale niż obwody wyjściowe sterownika.

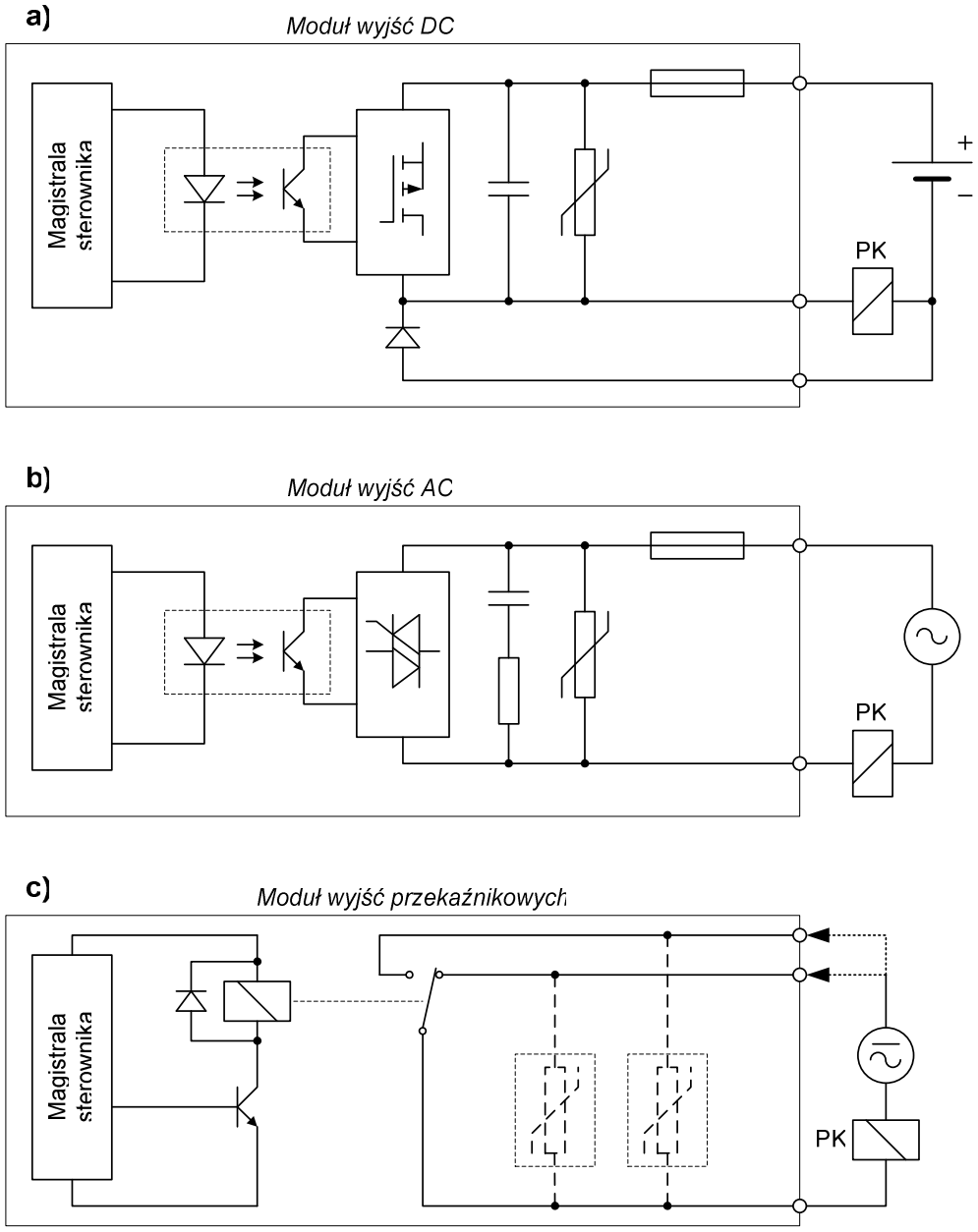
W wyjściach półprzewodnikowych w roli dwustanowych kluczy elektronicznych zwykle stosuje się tranzystory w obwodach wyjściowych prądu stałego oraz tyrystory i triaki w obwodach wyjściowych prądu przemiennego. Elementy te w porównaniu z przekaźnikami charakteryzują się dużo większą szybkością działania oraz posiadają znacznie większą trwałość. Oprócz tego pracują bezszelestnie, co może być istotne przy wyborze miejsca instalacji sterownika.

Podobnie jak w przypadku wejść sterownika, w modułach wyjść dyskretnych stosowana jest separacja galwaniczna pomiędzy magistralą sterownika PLC a zewnętrznymi obwodami wyjściowymi. Na rysunku 2.4 przedstawiono schematy ideowe przykładowych rozwiązań obwodów wyjściowych prądu stałego, przemiennego i z wyjściem przekaźnikowym [1]. W przypadku modułu z wyjściami DC spotyka się dwie odmienne konfiguracje połączeń tranzystora sterującego, od których zależy sposób dołączenia zewnętrznego źródła napięcia i odbiornika:

- **konfiguracja o logice dodatniej** (ang. *Positive Logic*), w której w stanie aktywnym wyjścia cyfrowego (ON) prąd płynie w kierunku od wyjścia modułu do odbiornika, a na zaciskach wyjściowych modułu pojawia się polaryzacja dodatnia (jak na schemacie z rysunku 2.4a).
- **konfiguracja o logice ujemnej** (ang. *Negative Logic*), w której w stanie aktywnym wyjścia prąd płynie w kierunku od odbiornika do modułu, a na zaciskach wyjściowych pojawia sie polaryzacja ujemna.

Wyjścia dyskretne sterowników PLC zazwyczaj sterują obciążeniami o charakterze indukcyjnym (cewki przekaźników, styczników, elektrozawory). Odbiorniki te powodują powstawanie przepięć łączeniowych podczas rozwierania ich obwodów prądowych, dlatego też moduły wyjść cyfrowych sterownika powinny być

wyposażone w odpowiednie zabezpieczenia przeciwprzepięciowe. W modułach półprzewodnikowych zabezpieczenia takie są zazwyczaj wbudowane, a do ich realizacji stosuje się powszechnie szybkie diody zabezpieczające, diody Zenera, warystory i układy RC. W przypadku wyjść przekaźnikowych, instalator musi na ogół sam zadbać o dołączenie odpowiednich zewnętrznych elementów zabezpieczających styki przekaźników przed skutkami przepięć, które dobiera się stosownie do zasilania (prąd stały, przemienny), polaryzację oraz parametrów obciążenia [5].



Rys.2.4. Schematy ideowe pojedynczego wyjścia cyfrowego dla obwodów prądu stałego (a), prądu przemiennego (b) oraz wyjść przekaźnikowych (c).

Dobierając moduł wyjść dyskretnych do systemu sterowania należy najpierw dokładnie przeanalizować algorytm sterujący oraz wziąć pod uwagę parametry sterowanego obiektu. Należy pamiętać, że z uwagi na ograniczoną trwałość mechaniczną przekaźników, w obwodach, gdzie występuje duża częstotliwość przełączeń powinny być stosowane wyłącznie wyjścia półprzewodnikowe. Oprócz tego, przy doborze modułu wyjść cyfrowych należy wziąć pod uwagę następujące jego parametry:

- liczba wyjść cyfrowych modułu (np. 8, 16, 32),
- rodzaj obwodu wyjściowego (przekaźnik, tranzystor, triak),
- napięcie znamionowe (np. 24V DC, 230V AC),
- napięcie przebicia izolacji,
- poziomy napięć wyjściowych dla stanów załączenia (ON) i wyłączenia (OFF),
- charakterystyki czasowe odpowiedzi wyjść,
- maksymalny prąd pojedynczego wyjścia półprzewodnikowych oraz sumaryczny prąd wyjściowy dla całego modułu,
- obciążalność i izolacja napięciowa styków przekaźników,
- trwałość mechaniczna przekaźników (np. 1.000.000 cykli ON/OFF),
- środowiskowe warunki pracy [5, 9].

2.2.5. MODUŁY WEJŚĆ ANALOGOWYCH

Moduły wejść analogowych przetwarzają ciągłe (analogowe) sygnały wejściowe na postać cyfrową, akceptowalną przez procesor sterownika. Głównym elementem budowy modułów wejściowych jest przetwornik analogowo-cyfrowy (ang. *ADC – Analog to Digital Converter*), który dokonuje zamiany ciągłego sygnału wejściowego na postać dyskretną (skwantowaną), reprezentowaną przez wartość liczbową, najczęściej zakodowaną na 16 bitach (tzw. słowo – ang. *Word*). Rozdzielczość typowych przetworników A/C spotykanych w torach analogowych sterowników PLC waha się od 12 do 16 bitów.

Spośród najczęściej występujących sygnałów analogowych w automatyce wymienić można:

- sygnały napięciowe, których typowe wartości znamionowe to:
  - -10 ÷ +10V;
  - 0 ÷ +10V;
  - 1 ÷ 5V;
- sygnały prądowe, występujące najczęściej w przedziałach:
  - 0 ÷ 20mA;
  - 4 ÷ 20mA;

- rezystancję, stanowiącą wyjściowy sygnał pomiarowy czujników wielkości nieelektrycznych (np. temperatury – PT100).

Obwody wejściowe modułów analogowych z wejściami napięciowymi zawierają najczęściej wzmacniacze sygnałowe, występujące w dwóch podstawowych konfiguracjach:

- **z wejściami pojedynczymi** (jednokońcówkowymi – ang. *single-ended*), w których wartość napięcia wejściowego mierzona jest względem potencjału odniesienia (tzw. masy), wspólnego dla wszystkich wejść analogowych.
- **z wejściami różnicowymi** – w tym układzie połączeń sygnał pomiarowy doprowadzony jest do dwóch niezależnych wejść wzmacniacza (IN+ i IN-), a wynikiem przetwarzania jest różnica potencjałów na tych wejściach.

W modułach z wejściami pojedynczymi połączenia sygnałów analogowych muszą być starannie ekranowane, ze względu na dużą wrażliwość na zakłócenia pochodzące od elektrycznych obwodów zewnętrznych. Wejścia różnicowe charakteryzują się większą odpornością na zakłócenia, a symetryczny sygnał pomiarowy może być doprowadzony za pomocą pary skręconych przewodów (najlepiej umieszczonych w ekranie).

Wzmacniacze wejściowe sygnałów napięciowych posiadają najczęściej dużą impedancję wejściową, co dodatkowo zwiększa podatność tych obwodów na zakłócenia elektromagnetyczne i elektrostatyczne, szczególnie o charakterze impulsowym. W celu eliminacji tych zakłóceń stosuje się powszechnie dolnoprzepustowe filtry wejściowe. Obecność tych filtrów w torach sygnałowych może niekiedy powodować pogorszenie dynamiki układu pomiarowego, dlatego też przy doborze modułu wejść analogowych należy wziąć pod uwagę ich parametry częstotliwościowe, podawane przez producentów w karcie katalogowej.

Do najważniejszych parametrów modułów wejść analogowych należą:

- liczba kanałów wejściowych (np. 1, 2, 4, 8),
- możliwość konfiguracji trybu pracy (jako wejścia napięciowe, prądowe lub do pomiaru rezystancji),
- zakresy napięć wejściowych lub prądów wejściowych,
- parametry przetwornika A/C (rozdzielczość, dokładność bezwzględna, liniowość, itp.),
- napięcie przebicia izolacji,
- tłumienie napięcia wspólnego dla wejść różnicowych (CMRR),
- tłumienie zakłóceń międzykanałowych (CCRR),
- czas akwizycji (pomiaru) podanej liczby kanałów wejściowych,
- środowiskowe warunki pracy [5].

2.2.6. MODUŁY WYJŚĆ ANALOGOWYCH

Moduły wyjść analogowych sterowników programowalnych służą do zamiany wartości liczbowych, zapisanych w pewnych obszarach pamięci sterownika, na sygnały ciągłe, które stanowią sprzęg pomiędzy wykonywanym programem użytkownika a sterowanym obiektem (maszyną, urządzeniem, procesem technologicznym). Najważniejszym elementem modułów wyjściowych jest przetwornik cyfrowo-analogowy (ang. *DAC – Digital to Analog Converter*), który dokonuje konwersji sygnału dyskretnego na równoważną mu postać ciągłą, reprezentowaną w postaci sygnałów napięciowych lub prądowych. Podobnie jak w przypadku przetworników A/C, rozdzielczość typowych przetworników C/A spotykanych w torach analogowych sterowników PLC mieści się w przedziale od 12 do 16 bitów.

Najczęściej spotykanymi analogowymi sygnałami wyjściowymi są:

- sygnały napięciowe, dla których typowe wartości znamionowe to:
  - -10 ÷ +10V;
  - 0 ÷ +10V;
  - 1 ÷ 5V;
- sygnały prądowe, zawierające się w zakresach:
  - 0 ÷ 20mA;
  - 4 ÷ 20mA;

Przy doborze modułu wyjść analogowych należy wziąć pod uwagę następujące jego parametry:

- liczba wyjść modułu (np. 1, 2, 4, 8),
- możliwość konfiguracji trybu pracy (jako wyjścia napięciowe lub prądowe),
- zakresy napięć wyjściowych lub prądów wyjściowych,
- parametry przetwornika C/A (rozdzielczość, dokładność, liniowość, itp.),
- napięcie przebicia izolacji,
- czas ustalania napięć wyjściowych,
- dopuszczalny prąd obciążenia wyjścia,
- parametry elektryczne obciążenia dla wyjść prądowych i napięciowych (rezystancja, indukcyjność, pojemność)
- środowiskowe warunki pracy [5].

2.2.7. MODUŁY KOMUNIKACYJNE

Moduły komunikacyjne pełnią funkcję interfejsu wymiany danych pomiędzy sterownikiem programowalnym a innymi urządzeniami, które połączone ze sobą za

pomocą przemysłowych sieci komunikacyjnych tworzą integralny system sterowania. Do urządzeń tych zaliczyć można:

- inne sterowniki programowalne, realizujące powiązane lub niezależne funkcje sterujące;
- sterowniki pracujące w systemie redundantnym;
- stacje (wyspy) rozproszonych wejść/wyjść;
- urządzenia wykonawcze wyposażone w interfejsy komunikacyjne (falowniki, styczniki inteligentne, układy łagodnego rozruchu silników, itp.);
- cyfrowe czujniki i przetworniki pomiarowe (np. przetworniki tlenu, temperatury i pH z interfejsem RS485);
- czujniki inteligentne (wyposażone w chipy komunikacyjne, np. czujniki indukcyjne z interfejsem ASI);
- urządzenia HMI: panele operatorskie, pulpity sterownicze, wyświetlacze, itp.;
- systemy sterowania nadrzędnego i komputery z oprogramowaniem SCADA.

Nowoczesne moduły komunikacyjne najczęściej posiadają swój własny procesor sterujący, który odpowiedzialny jest za prawidłowy przebieg transmisji danych. Do sterownika PLC może być dołączony jeden lub kilka modułów komunikacyjnych, obsługujących różne sieci przemysłowe. Do najczęściej występujących sieci komunikacyjnych w automatyce przemysłowej zaliczyć można:

- PROFIBUS (DP, FMS, PA);
- Industrial Ethernet (PROFINET);
- DeviceNet;
- CANopen;
- ASi (Actuator-Sensor Interface).

Oprócz wyżej wymienionych sieci przemysłowych, do wymiany danych pomiędzy sterownikami PLC a innymi urządzeniami powszechnie wykorzystywane są popularne standardy transmisji, do których należą interfejsy RS-232, RS-422, RS-485 oraz protokół Modbus. Ponadto wielu producentów urządzeń automatyki wprowadza swoje własne rozwiązania komunikacyjne, stosowane do niezawodnej wymiany danych pomiędzy sterownikiem a specjalizowanymi modułami rozszerzeń oraz kontrolerami ruchu (ang. *Motion Control*). Przykładem takiej sieci może być szybka sieć CC-link, opracowana przez firmę Mitsubishi Electric. Inną znaną siecią, przeznaczoną do bardzo szybkiej transmisji danych jest sieć Mechatrolink, która jest stosowana przez czołowych producentów (OMRON, Yaskawa, Mycom) w urządzeniach napędowych (serwonapędy, falowniki), kontrolerach ruchu, maszynach CNC oraz w systemach sterowania czasu rzeczywistego (INtime Micronet Co.).

Oprócz przewodowych sieci komunikacyjnych, w automatyce przemysłowej stosowane są również moduły do bezprzewodowej transmisji danych. Moduły te najczęściej wykonane są w postaci zewnętrznych urządzeń-konwerterów, które łączą

się ze sterownikiem programowalnym za pomocą sieci przewodowych (np. RS-232, DeviceNet), a następnie wysyłają i odbierają dane drogą radiową. W zależności od pożądanego zasięgu działania, systemy radiowej transmisji danych stosują najczęściej standardy:

- ZeegBee (IEEE 802.15.4) - odległość pomiędzy węzłami do 100m,
- WLAN (IEEE 802.11) – odległości do kilku kilometrów (z antenami kierunkowymi),
- GSM/GPRS – zasięg ograniczony infrastrukturą GSM.

2.2.8. MODUŁY SPECJALNE

Moduły specjalne są to urządzenia, które rozszerzają możliwości sterowników programowalnych o dodatkowe funkcje programowe, sprzętowe i komunikacyjne. Spośród wielu różnorodnych specjalnych modułów rozszerzeń najczęściej spotykane to:

- **moduły do pomiaru temperatury** – stanowią dodatkowy interfejs analogowy, umożliwiający podłączenie specjalnych czujników pomiarowych, takich jak termopary, czujniki półprzewodnikowe czy czujniki rezystancyjne (np. Pt 100, Pt 1000, Ni 100);
- **moduły szybkich liczników** (ang. HSC – *High Speed Counter*) – umożliwiają zliczanie zewnętrznych impulsów o wysokiej częstotliwości (100kHz, 1MHz). Wykorzystywane są najczęściej w układach sterowania napędów elektrycznych, w roli interfejsów do podłączenia obrotowo-impulsowych przetworników do pomiaru położenia i prędkości (tzw. enkoderów inkrementalnych). Oprócz tego umożliwiają pomiar częstotliwości i okresu sygnałów prostokątnych, pochodzących od zewnętrznych urządzeń, np. przetworników pomiarowych z wyjściem częstotliwościowym.
- **moduły generatorów impulsów i sygnałów PWM** – Umożliwiają wytworzenie sygnałów prostokątnych o zadanej częstotliwości i współczynniku wypełnienia (tzw. modulacja PWM – ang. *Pulse Width Modulation*). Są powszechnie stosowane do sterowania pracą silników krokowych i prądu stałego.
- **moduły pozycjonowania osi** (ang. APM – *Axis Positioning Module*) – stosowane do sterowania wieloosiowych napędów pozycjonujących.
- **moduły precyzyjnych wejść analogowych** – moduły wejść analogowych o specjalnie dobranych zakresach i zwiększonej rozdzielczości. Wykorzystywane są do precyzyjnych pomiarów sygnałów o małych wartościach, pochodzących z czujników specjalnych, mostków tensometrycznych i urządzeń automatyki wagowej.

3. Programowanie sterowników PLC

3.1. Norma IEC 61131

Niewątpliwie największą zaletą sterowników PLC jest możliwość ich zaprogramowania, co czyni je najbardziej uniwersalnymi elementami wszelkich systemów sterowania. Pierwsze sterowniki były programowane przy użyciu specjalnych urządzeń, wyposażonych w klawiaturę i prosty, cyfrowy wyświetlacz. Zapis algorytmu sterowania polegał na rozłożeniu zadania sterującego na elementarne instrukcje, które były kolejno wprowadzane do pamięci sterownika przy pomocy dołączonego programatora. Z czasem upowszechniły się komputery osobiste, które niemalże całkowicie zastąpiły sprzętowe programatory. Obecnie, do konfiguracji, parametryzacji i programowania sterowników PLC wykorzystuje się najczęściej komputery PC ze specjalistycznym oprogramowaniem. Komputery te muszą być wyposażone w odpowiedni interfejs komunikacyjny (np. port szeregowy, kartę sieci ethernet), umożliwiający połączenie i wymianę danych ze sterownikiem.

Od chwili pojawienia się pierwszych sterowników na rynku, ich producenci wprowadzali różne metody programowania sekwencji ich działania. Początkowo nie istniały żadne normy i wytyczne w zakresie używanych języków programowania, dlatego też każdy produkowany sterownik posiadał własną listę instrukcji, za pomocą których kodowano algorytm sterowania.

Z technicznego punktu widzenia, takie podejście było mało praktyczne, gdyż wymagało od inżynierów-automatyków znajomości wielu technik programowania, w zależności od zastosowanego sprzętu. W związku z tym powstała konieczność standaryzacji sterowników PLC, a w szczególności ujednolicenia języków ich programowania. W odpowiedzi na te potrzeby w roku 1993 Międzynarodowa Komisja Elektrotechniki (ang. *IEC – International Electrotechnical Commission*) wydała normę IEC 1131 pod tytułem „Programmable Controllers", która od 1998r. przyjęła oznaczenie IEC 61131.

Przedmiotem normy jest:

- wprowadzenie definicji i ustalenie ogólnych właściwości sterowników programowalnych oraz ich urządzeń peryferyjnych,
- określenie wymagań konstrukcyjnych i parametrów sterowników programowalnych,

- określenie zasad bezpieczeństwa, warunków serwisowania i testów sterowników PLC,
- specyfikacja reguł i składni ogólnie stosowanych języków programowania [5].

Obecnie ciągle trwają prace nad udoskonaleniem i rozszerzeniem normy. Aktualnie składa się ona z ośmiu części, przy czym część szósta została zarezerwowana do wykorzystania w przyszłości. Poniżej przedstawiono ogólnie zawartość poszczególnych części normy [5]:

Część 1. Postanowienia ogólne.

W części pierwszej zawarto ogólne definicje, nazewnictwo i własności sterowników PLC, pozwalające odróżnić je od innych systemów sterowania. Przedstawiono standardowe właściwości sterowników, które należy wziąć pod uwagę przy ich wyborze do zastosowania w konkretnej aplikacji. Omówiono fazy cyklu pracy sterownika i zdefiniowano zasady współpracy z programatorem oraz urządzeniami obsługi operatora.

Część 2. Wymagania i badania części sprzętowej.

W części tej określono wymagania sprzętowe i funkcjonalne sterowników. Zdefiniowano środowiskowe warunki ich pracy, przechowywania i transportu. Przedstawiono klasyfikację sterowników oraz narzędzi programowania. Część ta opisuje również metodykę badań i procedury testowe sterowników oraz kryteria oceny jakości.

Część 3. Języki programowania.

Jest to najbardziej rozbudowana i jednocześnie najlepiej dopracowana część normy. Określa ona podział oraz pojęcia podstawowe dotyczące języków programowania. Oprócz tego definiuje zasady tworzenia programu sterującego w poszczególnych językach.

Część 4. Wskazówki dla użytkownika

Przedstawiono tu praktyczne wytyczne dla użytkownika, dotyczące projektowania systemu sterowania z wykorzystaniem sterowników PLC. Określono metody analizy sterowanego procesu, zasady doboru urządzeń. Zostały opisane także metody syntezy algorytmów, testów końcowych i zasad konserwacji.

Część 5. Komunikacja

W części piątej normy zdefiniowano zasady komunikacji pomiędzy sterownikami programowalnymi z różnych rodzin oraz urządzeniami zewnętrznymi. Przedstawiono między innymi: zasady wymiany danych pomiędzy urządzeniami, metody adresowania poszczególnych urządzeń, metody obsługi alarmów oraz zasady dostępu do sieci i jej administrowania.

Część 6. Zarezerwowana do wykorzystania w przyszłości

Część 7. Programowanie zbiorów rozmytych (ang. Fuzzy Logic)

W tej części przedstawiono możliwości zastosowania typowych języków programowania zdefiniowanych w części trzeciej, do implementacji algorytmów sterowania wykorzystujących teorię zbiorów rozmytych.

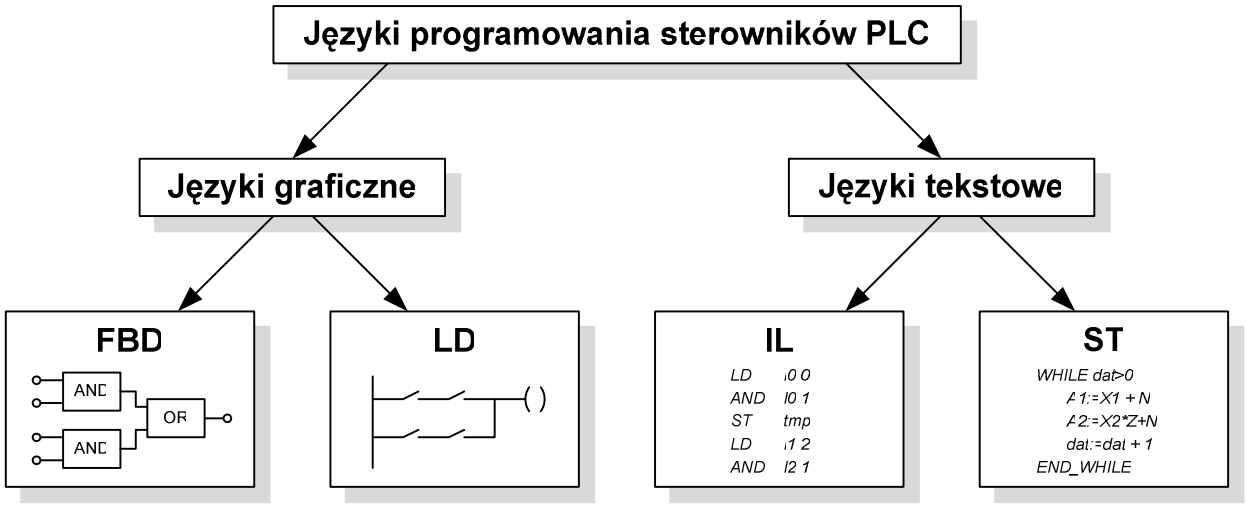
Część 8. Wskazówki do implementacji języków programowania.

Dodatkowa część normy, będąca uzupełnieniem części 3, przedstawia informacje i praktyczne wskazówki dotyczące zastosowania języków programowania. Określa również ogólne wymagania dotyczące sprzętu i oprogramowania, które są niezbędne do rozwijania i konserwacji programów użytkownika.

3.2. Języki programowania sterowników PLC

3.2.1. PODZIAŁ JĘZYKÓW PROGRAMOWANIA

Język programowania jest to uporządkowany ciąg symboli, instrukcji i wyrażeń, które w sposób jednoznaczny zapisują sposób działania sterownika programowalnego. Na rysunku 3.1 przedstawiono ogólny podział języków programowania sterowników PLC.



Rys.3.1. Ogólny podział języków programowania sterowników PLC.

Według normy IEC 61131 języki programowania sterowników PLC można podzielić na dwie główne grupy: języki tekstowe oraz języki graficzne [5].

Do grupy języków tekstowych zalicza się:

- **IL - Lista instrukcji** (ang. *Instruction List*) – jest to język programowania niskiego poziomu, składający się z zestawu instrukcji mnemotechnicznych, do których zaliczyć można operacje logiczne i arytmetyczne, funkcje czasowe i licznikowe, operacje porównania i transferu danych. Struktura tego języka jest podobna do asemblera, a nazwy i sposób wywołania poszczególnych instrukcji zależą od typu sterownika. Ze względu na małą przejrzystość kodu, język IL jest rzadko stosowany do zapisu całego algorytmu sterowania. Niemniej jednak, idealnie nadaje się do kodowania pewnych funkcji programowych, zwłaszcza złożonych algorytmów obliczeniowych, które w zapisie IL są zwarte i proste oraz są szybciej wykonywane przez procesor sterownika.
- **ST – Tekst Strukturalny** (ang. *Structured Text*) – jest to język zaliczany do grupy języków wyższego poziomu, gdyż nie używa operatorów zorientowanych maszynowo. Struktura języka ST przypomina nieco strukturę znanych języków algorytmicznych, takich jak Pascal czy Basic, powszechnie wykorzystywanych do tworzenia oprogramowania dla komputerów PC. Do podstawowych elementów języka ST należą wyrażenia i instrukcje. W odróżnieniu od języka IL, powstały kod jest bardzo czytelny i zwięzły, gdyż zbudowany jest z typowych struktur funkcyjnych, takich jak: *IF, THEN, ELSE, FOR, WHILE, REPEAT*, itp. Niestety, programy napisane w języku ST po skompilowaniu działają wolniej (od tych zakodowanych bezpośrednio w IL) oraz wykorzystują więcej zasobów pamięci operacyjnej.

Do grupy języków graficznych zalicza się:

- **FBD – Funkcjonalny Schemat Blokowy** (ang. *Functional Block Diagram*) – język wzorowany na schematach ideowych stosowanych w elektronice, opisujących przepływ sygnałów topologię połączeń układów scalonych wykonanych w technice cyfrowej. Głównymi elementami wykorzystywanymi w języku FBD są bloki i elementy sterujące (np. łączniki, instrukcje skoków), połączone między sobą liniami. Realizacja programu w FBD opiera się na przepływie sygnałów, których działanie definiuje topologia obwodu. Przepływ sygnału następuje z wyjścia funkcji lub bloku funkcyjnego do przyłączonego wejścia następnej funkcji lub bloku funkcyjnego.
- **LD – Schemat Drabinkowy** (ang. *Ladder Diagram*) – opiera się na symbolach schematów elektrycznych układów sterowania wykonanych w technice stykowo-przekaźnikowej. Podstawowymi symbolami języka drabinkowego są styki, przedstawiające wartości logiczne sygnałów wejściowych i zmiennych boolowskich oraz dwustanowe wyjścia, będące odzwierciedleniem cewek przekaźnika, które służą do wysterowania wyjść dyskretnych oraz przypisania wartości logicznych do zmiennych boolowskich. Oprócz tego w schematach drabinkowych wykorzystuje się bloki funkcyjne, używane do opisu bardziej złożonych funkcji (liczniki, timery, komparatory, operacje arytmetyczne, itp.).

Wszystkie wyżej wymienione języki programowania posiadają swoje cechy osobliwe, które predysponują je do realizacji odmiennych zadań. Ogólnie, najchętniej przez programistów wybierane są języki graficzne, które przedstawiają algorytm działania sterownika w sposób intuicyjny i łatwy do analizy. Dlatego też, będą one najczęściej wykorzystywane do realizacji wszelkich operacji logicznych. Z kolei operacje pamięciowe wykonywane na dużych blokach danych, złożone funkcje obliczeniowe oraz procedury iteracyjne będą łatwiejsze do zakodowania przy użyciu języków tekstowych. Ponadto każdy programista ma swoje preferencje, które wynikają z przyzwyczajeń i doświadczenia zawodowego. Tak więc elektryk z pewnością skorzysta z języka drabinkowego, który do złudzenia przypomina schematy elektryczne stosowane w automatyce przekaźnikowej. Elektronik chętniej wybierze język FBD, z którego korzysta na co dzień przy tworzeniu schematów elektronicznych urządzeń cyfrowych. Natomiast informatyk pewnie skieruje się w stronę języków tekstowych, a zwłaszcza ST, który najlepiej, w sposób dla niego zrozumiały przedstawia kolejność wykonywanych działań.

Oprócz czterech, wyżej wymienionych języków programowania sterowników PLC, do zapisu kolejności wykonywanych działań w dużych, skomplikowanych sekwencyjnych układach sterowania często wykorzystuje się tzw. **sekwencyjny schemat funkcjonalny** (ang. ***SFC*** *– Sequential Function Chart*). Narzędzie to opisuje poszczególne zadania sterowania w sposób graficzny, za pomocą sieci składającej się z kroków i warunków przejść pomiędzy tymi krokami. Za pomocą SFC buduje się jedynie nadrzędną strukturę programu, natomiast poszczególne zadania sterowania koduje się za pomocą wyżej wymienionych języków programowania.

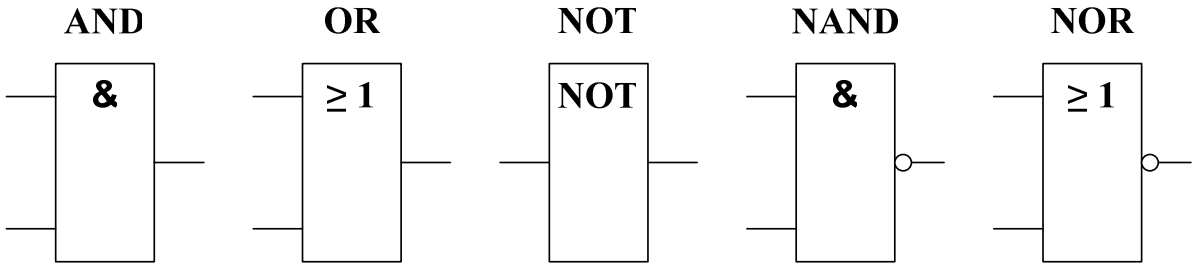
3.2.2. ZASADY TWORZENIA PROGRAMÓW W JĘZYKU FBD

Opracowanie programu w języku FBD polega na odpowiednim połączeniu bloków funkcyjnych, tak aby przepływający sygnał wykonywał odpowiednie działania na zmiennych oraz sygnałach wejściowych i wyjściowych sterownika. Do podstawowych bloków języka FBD zaliczyć można:

- funkcje logiczne (AND, OR, NOT, XOR ...)
- elementy bistabilne – (przerzutniki RS i SR)
- czasomierze (timery TON, TOF, TP, TONR ...)
- liczniki (CTU, CTD, CTUD)
- komparatory (EQ, GE, LT ...)
- detektory zbocza (R-TRIG, F-TRIG)
- funkcje arytmetyczne i logiczne (ADD, DIV, SQRT, AND, OR ...)

Na rysunku 3.2 przedstawiono symbole podstawowych funkcji logicznych, występujących w języku FBD. Przy tworzeniu schematów często korzysta się z

operacji negacji sygnału, która polega na umieszczenia symbolu małego okręgu na wejściu lub wyjściu bloku. Pomimo, że operacja negacji należy do podstawowych funkcji logicznych, niektóre edytory nie posiadają w swojej bibliotece funktora NOT w postaci bloku, a oferują jedynie operator negacji. Dlatego też funktory logiczne NAND i NOR (bramki AND i OR z negacją na wyjściu) najczęściej nie występują w postaci osobnych bloków, a ich realizacja wymaga zastosowania formy podstawowej bloku (AND, OR) w połączeniu z operatorem negacji na wyjściu.

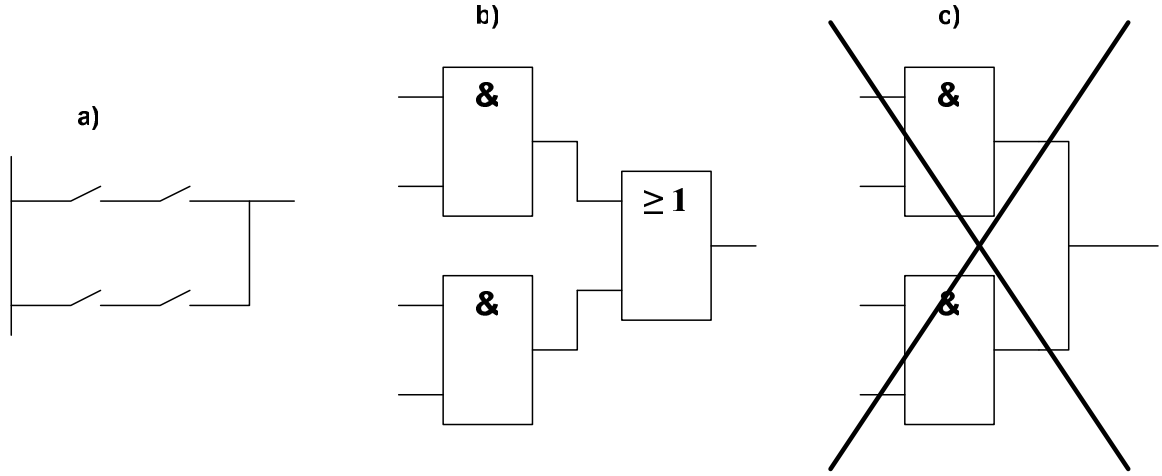


Rys.3.2. Podstawowe funkcje logiczne w języku FBD.

Pomimo, że język FBD pozwala na dość swobodne tworzenie obwodów logicznych, przy opracowywaniu programu sterowania należy przestrzegać kilku podstawowych zasad:

- bezpośrednie połączenia pomiędzy blokami są dozwolone dla sygnałów przesyłających dane tego samego typu (np. BOOL→BOOL, Integer→Integer, Real→Real, ...);
- połączenia pomiędzy blokami dla sygnałów różnych typów danych są możliwe poprzez zastosowanie pośrednich funkcji konwersji typów (np. Byte_to_Word, INT_to_REAL, ...);
- przepływ sygnałów sterujących realizuje się poprzez połączenie wyjścia jednego bloku z wejściem innego bloku;
- z jednego wyjścia można doprowadzić sygnał do kilku wejść;
- wejścia bloków mogą pozostać niepodłączone (zazwyczaj); w takim przypadku domyślną wartością na tych wejściach jest zero;
- niedopuszczalne jest połączenie dwóch wyjść ze sobą;

Szczególnie często popełnianym błędem wśród początkujących programistów jest próba połączenia wyjść dwóch różnych bloków ze sobą, tzw. suma galwaniczna OR (ang. *wired-OR*). Błędy te najczęściej biorą się z przyzwyczajeń do języka drabinkowego, w którym suma logiczna sygnałów przedstawiana jest jako ich równoległe połączenie. Problem ten zilustrowano na rysunku 3.3, gdzie pewna funkcja logiczna opracowana w języku drabinkowym (a) jest prawidłowo przedstawiona w języku FBD (b). Z kolei rysunek (c) przedstawia nieprawidłowy obwód logiczny, w którym błędem jest połączenie ze sobą dwóch wyjść bloków.



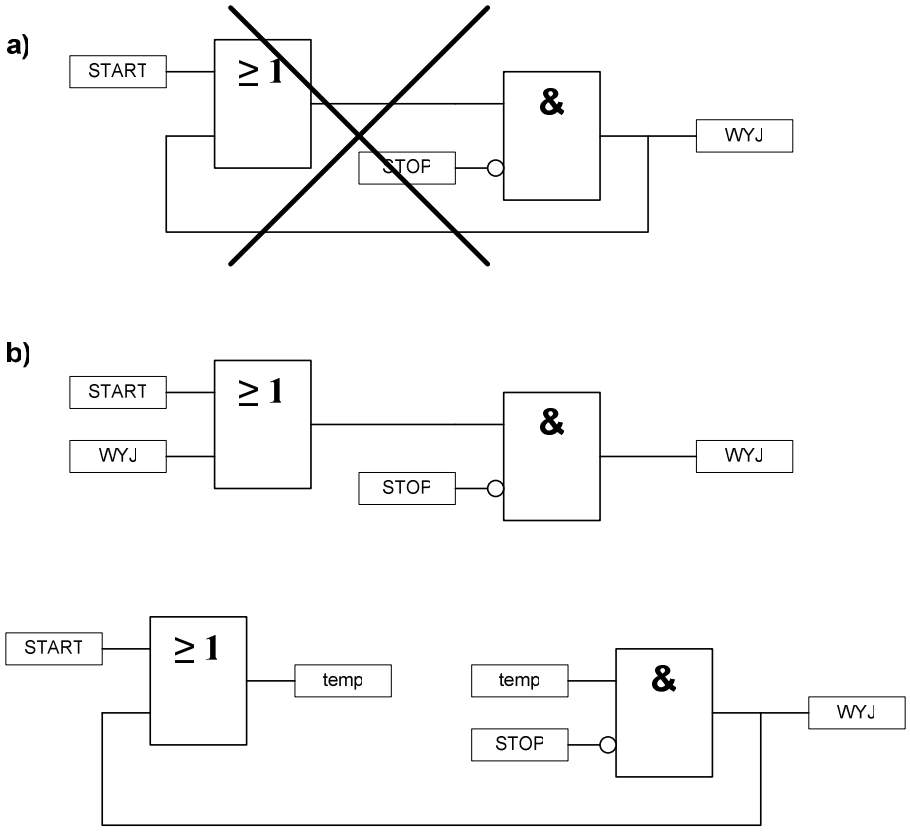
Rys.3.3. Zapis pewnej funkcji logicznej: a)- w języku drabinkowym, b)- w języku FBD, c)- próba nieprawidłowej realizacji funkcji w języku FBD.

Niektóre programy logiczne, realizujące funkcje pamięciowe (zatrzaskowe) wymagają zastosowania obwodów ze sprzężeniami zwrotnymi, tzn. takich, które tworzą zamknięte pętle przepływu sygnałów. Najprostszym przykładem może być klasyczny obwód sterowania stycznika z funkcją podtrzymania.

Trzy sposoby realizacji tego programu przedstawiono na rysunku 3.4. Działanie układu jest następujące: po przyciśnięciu przycisku START następuje załączenie wyjścia WYJ. Zwolnienie przycisku START nie powoduje żadnych zmian w układzie, gdyż w obwodzie zastosowano funkcję podtrzymania (zmienna WYJ została podłączona do bramki OR, razem z sygnałem wyzwalającym START). Stan układu może się zmienić, po przyciśnięciu przycisku STOP (poprzez podanie na wejście logicznej „jedynki"). Na schemacie z rysunku 3.4a zastosowano tzw. jawną pętlę sprzężenia zwrotnego, która w większości przypadków jest traktowana jako błąd. Pozostałe dwa schematy przedstawiają prawidłowe programy, w których pętla sprzężenia zwrotnego została rozdzielona poprzez zmienne (WYJ oraz temp).

W przypadku zastosowania pętli jawnej nie ma możliwości określenia kolejności wykonywania działań logicznych sterownika, dlatego też, taka struktura w większości przypadków jest niedopuszczalna. W programie zrealizowanym wg schematu 3.4b kolejność działań jest następująca: najpierw zostaje obliczona wartość na wyjściu bramki OR, a następnie zrealizowana jest funkcja AND, której wynik steruje sygnałem wyjściowym WYJ. Z kolei program ze schematu c) najpierw oblicza wartość funkcji AND, a na końcu realizuje funkcję OR, której wynik przepisywany jest do zmiennej temp.

Analizując powyższy przypadek można stwierdzić, że kolejność opracowywania działań przez sterownik zależy od sposobu zapisu programu użytkownika. Niekiedy kolejność wykonywanych operacji może mieć duże znaczenie, gdyż może doprowadzić do nieprawidłowego działania programu sterującego.



Rys.3.4. Program sterowania wyjścia z funkcją podtrzymania: a)- zastosowanie jawnej pętli sprzężenia zwrotnego (najczęściej nieprawidłowe), b-c)- przerwanie pętli sprzężenia zwrotnego poprzez zastosowanie zmiennych.

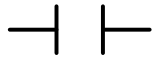
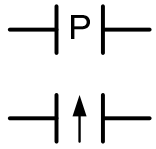
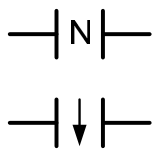
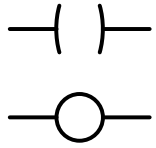
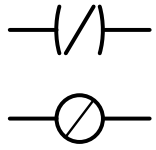
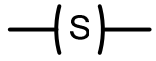
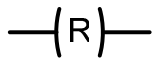
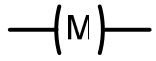
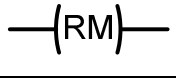
3.2.3. EDYCJA PROGRAMÓW W JĘZYKU DRABINKOWYM

Edycja programu sterującego w języku drabinkowym przypomina projektowanie schematu elektrycznego dawnego układu sterowania, w którym wykorzystywano elementy stykowe -przekaźniki i styczniki. W języku drabinkowym sposób działania programu określają instrukcje, reprezentowane przez standaryzowane symbole graficzne, umieszczone poziomo pomiędzy umownymi liniami zasilania. Z wyglądu przypomina to drabinę z wieloma szczeblami – stąd nazwa „język drabinkowy".

W języku drabinowym rozróżnia się dwa rodzaje instrukcji: instrukcje stykowe oraz instrukcje bloków funkcyjnych. Instrukcje stykowe służą do wykonywania operacji na pojedynczych bitach. Stanowią również programowy interfejs dla wejściowych i wyjściowych sygnałów dyskretnych sterownika. Podstawowe symbole instrukcji stykowych przedstawiono w tabeli 3.1 [5].

Instrukcje bloków funkcyjnych sprowadzają bardziej skomplikowane operacje do postaci pojedynczego bloku, którego warunki wykonania definiuje się za pomocą instrukcji stykowych. Do przykładowych bloków funkcyjnych zaliczyć można: liczniki, timery, regulatory itp.

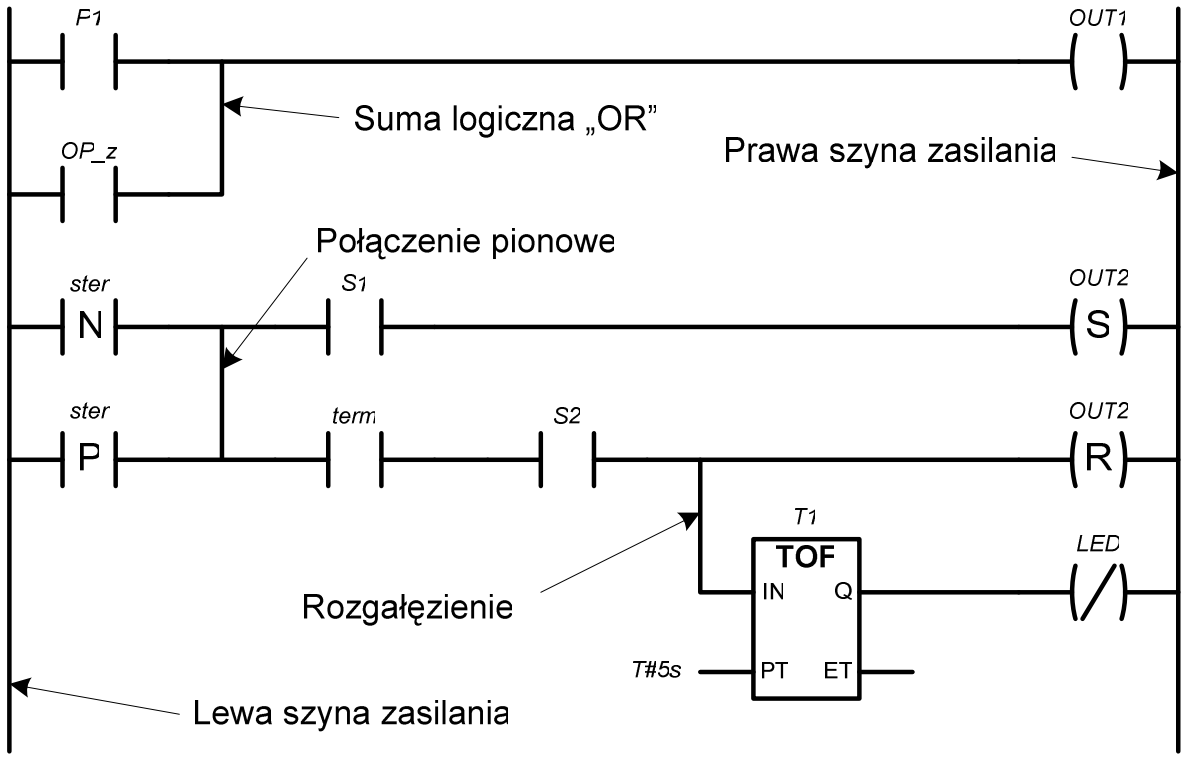
Tab.3.1. Podstawowe symbole instrukcji stykowych.

Symbol	Opis działania
	Styk zwierny – normalnie otwarty (NO). Stan połączenia z lewej strony styku przenoszony jest na stronę prawa, jeżeli zmienna logiczna przypisana do styku ma wartość 1.
	Styk rozwierny – normalnie zamknięty (NC). Stan połączenia z lewej strony styku przenoszony jest na stronę prawa, jeżeli zmienna logiczna przypisana do styku ma wartość 0.
	Styk detekcji zbocza narastającego. Jeżeli skojarzona ze stykiem zmienna logiczna zmieniła stan z 0 na 1 i jednocześnie stan logiczny z lewej strony ma wartość 1, wówczas po prawej stronie styku wystąpi wartość 1 w czasie trwania jednego cyklu przetwarzania.
	Styk detekcji zbocza opadającego. Jeżeli skojarzona ze stykiem zmienna logiczna zmieniła stan z 1 na 0 i jednocześnie stan logiczny z lewej strony ma wartość 1, wówczas po prawej stronie styku wystąpi wartość 1 w czasie trwania jednego cyklu przetwarzania.
	Cewka normalna. Stan połączenia z lewej strony cewki jest przenoszony na prawą stronę i jednocześnie jest zapisywany do skojarzonej z nią zmiennej boolowskiej.
	Cewka z negacją. Stan połączenia z lewej strony cewki jest przenoszony na prawą stronę, a do skojarzonej z nią zmiennej boolowskiej jest zapisywana wartość odwrotna tego stanu (negacja).
	Cewka ustawiająca. Jeżeli stan połączenia z lewej strony ma wartość 1, to skojarzona zmienna boolowska zostanie trwale ustawiona na wartość 1. Stan ten będzie się utrzymywał do chwili wyzerowania przez cewkę kasującą (R).
	Cewka kasująca. Jeżeli stan połączenia z lewej strony ma wartość 1, to skojarzona zmienna boolowska zostanie trwale ustawiona na wartość 0. Stan ten będzie się utrzymywał do chwili ponownego ustawienia przez cewkę ustawiającą (S).
	Cewka z zapamiętaniem stanu. Działa tak jak zwykła cewka, z tą różnicą, że skojarzona z nią zmienna ma atrybut podtrzymania (retentive variable), co oznacza, że jej stan będzie zapamiętany po zatrzymaniu programu.
	Cewka ustawiająca z zapamiętaniem stanu. (opis j.w.)
	Cewka kasująca z zapamiętaniem stanu. (opis j.w.)

—(P)—	Cewka wrażliwa na zbocze narastające. Jeżeli stan logiczny z lewej strony zmieni wartość z 0 na 1, wówczas skojarzona zmienna przyjmie wartość 1 na czas trwania jednego cyklu przetwarzania.
—(N)—	Cewka wrażliwa na zbocze opadające. Jeżeli stan logiczny z lewej strony zmieni wartość z 1 na 0, wówczas skojarzona zmienna przyjmie wartość 1 na czas trwania jednego cyklu przetwarzania.

Przy tworzeniu schematu drabinkowego najczęściej używa się połączeń poziomych pomiędzy poszczególnymi elementami. Oprócz tego stosowane są połączenia pionowe pomiędzy sąsiednimi obwodami (szczeblami) oraz połączenia rozgałęzione. W odróżnieniu od języka FBD, w języku drabinkowym można łączyć dwa lub więcej wyjść ze sobą – w ten sposób realizuje się sumę logiczną OR. Na rysunku 3.5 przedstawiono przykładowy program w języku drabinkowym, na którym zaznaczono typowe elementy łączące.

Sekwencja poszczególnych obwodów w języku drabinkowym jest wykonywana w kolejności od góry do dołu. W niektórych edytorach dopuszcza się stosowanie instrukcji skoków, które pozwalają na zmianę tej kolejności. Należy jednak pamiętać, że nieostrożne posługiwanie się instrukcjami skoku może doprowadzić w skrajnych przypadkach do zapętlenia programu, co skutkuje przekroczeniem maksymalnego czasu wykonania cyklu i zatrzymaniem krytycznym sterownika.



Rys.3.5. Przykładowy program opracowany w języku drabinkowym.

3.2.4. SEKWENCYJNY SCHEMAT FUNKCJONALNY

W trzeciej części normy IEC 61131, oprócz podstawowych języków programowania sterowników – *LD, FBD, IL* oraz *ST*, przedstawiono formalizm sekwencyjnego schematu funkcjonalnego (ang. SFC – *Sequential Function Chart*), który jest stosowany do modelowania złożonych algorytmów sterowania procesów, w których występują pewne sekwencje działań. Metoda SFC wywodzi się bezpośrednio z metody *Grafcet*, która jest stosowana do modelowania algorytmów procesu i algorytmów sterowania. Teoretyczne podstawy metody *Grafcet* stanowi formalizm sieci Petriego typu P/T (Pozycja/Tranzycja) [2, 7, 8].

Strukturyzacja programu za pomocą SFC polega na opracowaniu sieci złożonej z kroków (ang. *steps*) i warunków przejść pomiędzy tymi krokami – tzw. tranzycji (ang. *transitions*). Natomiast działanie wewnętrzne poszczególnych kroków opisuje się za pomocą zdefiniowanych w normie języków programowania.

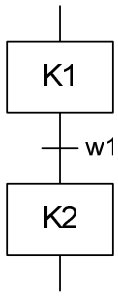
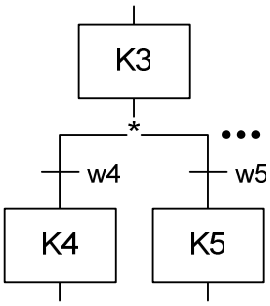
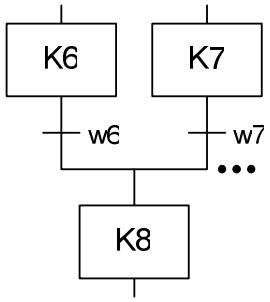
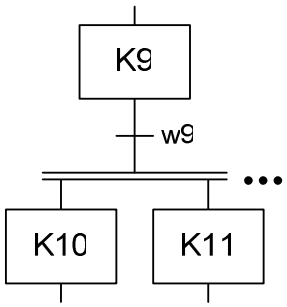
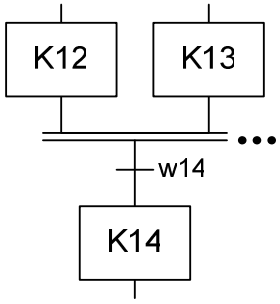
Symbolem graficznym kroku jest prostokąt. Krok może być aktywny albo nieaktywny. Aktywność danego kroku określa tzw. flaga kroku, która jest reprezentowana przez zmienną logiczną (wartość 1 dla kroku aktywnego, 0 dla nieaktywnego). Stan początkowy procesu określają zmienne wewnętrzne i wyjściowe oraz kroki początkowe. W każdej strukturze SFC może znajdować się tylko jeden krok początkowy, który na schemacie występuje jako prostokąt rysowany podwójną linią.

Tranzycje przedstawiają warunki logiczne, które muszą zostać spełnione, żeby nastąpiło przejście z jednego etapu procesu do następnego. Przejście bedzie dozwolone tylko w przypadku, kiedy wszystkie poprzedzające je kroki będą aktywne. Tranzycja może posiadać skojarzoną z nią zmienną boolowską lub być wynikiem rozwiązania pojedynczego wyrażenia logicznego. Spotyka się również tranzycje „zawsze spełnione", tzn. takie, do których jest przypisana na stałe wartość logiczna „1". W takim przypadku następuje bezwarunkowe przejście z jednego kroku do drugiego.

Kolejne kroki i przejścia muszą zawsze występować naprzemiennie. Niedopuszczalne jest bezpośrednie połączenie dwóch kroków, podobnie jak nie można umieszczać pod rząd dwóch tranzycji. Poprzez zastosowanie różnych kombinacji kroków i przejść można utworzyć rozmaite sekwencje, których podstawowe przykłady zestawiono w tabeli 3.2.

Z każdym krokiem może być skojarzony pewien zestaw instrukcji – tzw. akcji, które będą wykonywane w czasie aktywności danego kroku. Poszczególne instrukcje są skojarzone z krokiem za pomocą tzw. bloków akcji, a warunki wykonania tych instrukcji specyfikują tzw. kwalifikatory akcji. Kwalifikatory akcji oddziałują na flagę kroku i określają w jaki sposób akcja ma być uruchomiona i z jakimi warunkami czasowymi. W tabeli 3.3 przedstawiono listę kwalifikatorów akcji, które zostały zdefiniowane w normie IEC 61131 [5].

Tab.3.2. Podstawowe sekwencje kroków i przejść w SFC.

Przykład	Opis sekwencji
	**Sekwencja pojedyncza – prosta** Kroki i przejścia występują kolejno po sobie. Przejście z kroku K1 do K2 nastąpi, jeżeli krok K1 będzie aktywny i jednocześnie zostanie spełniony warunek przejścia w1 (w1=TRUE)
	**Sekwencja rozbieżna** Wybór pomiędzy różnymi sekwencjami odbywa się na zasadzie sprawdzenia warunków przejścia, w kolejności od lewej do prawej. Spotyka się również struktury z numeracją kolejności odpytywania. Przejście z K3 do K5 nastąpi, jeśli K3 będzie aktywny oraz warunek przejścia w4 nie będzie spełniony, a będzie spełniony warunek w5.
	**Sekwencja zbieżna** Wszystkie sekwencje rozbieżne kończą się symbolami przejścia. Przejście do kroku K8 nastąpi, jeśli K6 będzie aktywny oraz warunek przejścia w6 będzie spełniony, albo K7 będzie aktywny i jednocześnie będzie spełniony warunek w7.
	**Sekwencja równoczesna rozbieżna** W tym przypadku następuje rozdzielenie procesu na dwie lub więcej równoczesnych sekwencji. Przejście z kroku K9 do kroków K10 i K11 nastąpi, jeśli K9 będzie aktywny oraz warunek przejścia będzie spełniony.
	**Sekwencja równoczesna zbieżna** Wszystkie sekwencje rozbieżne równoczesne kończą się symbolem podwójnej linii. Przejście do kroku K14 nastąpi, jeśli wszystkie kroki przed podwójną linią będą aktywne (K12 i K13) oraz będzie spełniony warunek przejścia w14.

Tab.3.3. Lista kwalifikatorów akcji stosowanych w SFC.

Kwalifikator	Opis
Brak	Nie zapamiętywany. Akcja jest wykonywana w czasie aktywności kroku
N	Nie zapamiętywany – działanie j.w.
R	Kasowanie nadrzędne (Reset). Akcja ustawiona w innym kroku zostaje skasowana
S	Ustawienie (Set). Akcja będzie wykonywana nawet gdy krok będzie nieaktywny, do momentu aż zostanie skasowana w innym kroku za pomocą kwalifikatora R.
L	Ograniczony w czasie. Akcja jest wykonywana przez zadany czas, ale nie dłużej niż trwa aktywność kroku.
D	Opóźniony w czasie. Wykonanie akcji rozpocznie się z opóźnieniem, pod warunkiem, że krok pozostanie aktywny.
P	Impuls. Akcja jest wykonana tylko przez jeden cykl pracy sterownika.
SD	Zapamiętany i opóźniony. Kombinacja kwalifikatorów S i D. Akcja będzie wykonana z opóźnieniem i zostanie zapamiętana (trwale ustawiona).
DS	Opóźniony i zapamiętywany. Ustawienie i zapamiętanie akcji nastąpi z opóźnieniem, po zadanym czasie.
SL	Zapamiętany i ograniczony w czasie. Akcja jest wykonywana przez zadany czas, niezależnie czy krok z nią skojarzony będzie aktywny.

3.3. Zmienne i typy danych

3.3.1. ZMIENNE

Zmienne (ang. *Variables*) są wykorzystywane do przetwarzania i przechowywania danych użytkownika. Umieszczone są w pamięci operacyjnej sterownika, przy czym ich bezwzględna lokalizacja w pamięci nie musi być zdefiniowana przez programistę. Deklaracja zmiennych polega jedynie na nadaniu jej unikatowej nazwy oraz określenia typu danych, reprezentowanego przez zmienną. Oprócz tego, użytkownik może zdefiniować pewne właściwości zmiennej, jak na przykład atrybut *RETAIN*, który oznacza, że wartość zmiennej będzie podtrzymywana także po zaniku zasilania sterownika.

Zmienne reprezentują dane aplikacji, których wartości w czasie wykonywania programu mogą się zmieniać. Wyjątek stanowią zmienne z atrybutem *CONSTANT* (stałe), które podczas wykonywania programu pozostaje niezmienione. W różnych językach programowania wartości danych są przedstawiane w postaci tzw. literałów, które ze względu na typ danych podzielić można na:

- literały liczbowe,
- literały w postaci czasowej,
- ciągi znaków.

W tabeli 3.4 przedstawiono przykładowe sposoby przedstawiania danych

Tab.3.4. Przykładowe sposoby reprezentacji danych liczbowych, łańcuchowych i czasowych.

Typ danej	**Przykład**
Liczba całkowita	*123 ; -95 ; 100_000*
Liczba rzeczywista	*10.0 ; -123.456 ; 0.0 ; -1.5E-3*
Liczba binarna	*2#11110000 ; 2#0101_0101*
Liczba szesnastkowa	*16#A8 ; 16#FFFF*
Liczby i wyrażenia boolowskie	*0 ; 1 ; TRUE ; FALSE*
Ciąg znaków	*'ABCD' ; 'TEXTRL'*
Czas trwania	*T#10s ; T#3h_15m ; TIME#30ms*
Data	*DATE#2010-12-31 ; D#1970-01-01*
Godzina dnia	*TIME_OF_DAY#14:15:59.25*

Rozróżnia się zmienne proste i wieloelementowe. Zmienna prosta (skalarna, jednoelementowa) reprezentuje pojedynczą daną, należącą do typu elementarnego, wyliczeniowego lub okrojonego. Do zmiennych wieloelementowych zalicza się tablice i struktury. Tablica reprezentuje zbiór elementów tego samego typu, zajmujących jednolitą przestrzeń adresową. Tablice mogą być jednowymiarowe lub wielowymiarowe. Dostęp do elementów tablicy jest możliwy poprzez adresowanie bezpośrednie pamięci, lub poprzez adresowanie pośrednie – indeksowe (np. TAB[10,25]). Z kolei struktury (rekordy) zawierają zbiór zmiennych należących do różnych typów, które są reprezentowane przez identyfikatory (pola) oddzielone kropkami (np. *Silnik3.Parametry.PRAD*) [5].

3.3.2. TYPY DANYCH

Typy danych definiują wielkość obszaru pamięci potrzebną do zapisania danych (zmiennych, stałych, literałów) oraz określają ich strukturę, zakresy wartości oraz rodzaj operacji, jakie mogą być z ich udziałem wykonywane. W normie IEC 61131-3 zdefiniowano elementarne typy danych, które są najczęściej wykorzystywane do programowania sterowników PLC. W tabeli 3.5 przedstawiono listę wszystkich elementarnych typów danych oraz podano wielkość pamięci niezbędnej do ich zapisania (liczbę zajmowanych bitów) [5, 6]. W przypadku typów związanych z datą i czasem zajmowana wielkość pamięci jest niezdefiniowana, gdyż może być zmienna i zależna od typu sterownika.

Tab.3.5. Elementarne typy danych.

Nazwa typu	Typ danych	Liczba bitów
BOOL	Boolowski (logiczny)	1
SINT (short integer)	Liczba całkowita krótka	8
INT (integer)	Liczba całkowita	16
DINT (double integer)	Liczba całkowita podwójna	32
LINT (long integer)	Liczba całkowita długa	64
USINT (unsigned short integer)	Liczba całkowita krótka bez znaku	8
UINT (unsigned integer)	Liczba całkowita bez znaku	16
UDINT (unsigned double integer)	Liczba całkowita podwójna bez znaku	32
ULINT (unsigned long integer)	Liczba całkowita długa bez znaku	64
REAL	Liczba rzeczywista	32
LREAL (long real)	Liczba rzeczywista długa	64
TIME	Czas trwania	
DATE	Data	
TIME_OF_DAY lub TOD	Godzina dnia	
DATE_AND_TIME lub DT	Data i czas	
STRING	Ciąg znaków	
BYTE	Bajt	8
WORD	Słowo	16
DWORD (double word)	Podwójne słowo	32
LWORD (long word)	Słowo długie	64

Na podstawie elementarnych typów danych programista może tworzyć typy własne (pochodne), które mogą być wykorzystane do deklaracji stałych i zmiennych. Spośród często spotykanych pochodnych typów danych wymienić można:

- **typ wyliczeniowy**, który zawiera skończony zbiór wartości, jakie może przyjmować zmienna (np. CYFRA={*0,1,2,3,4,5,6,7,8,9*});
- **typ okrojony**, który polega na ograniczeniu zakresu wartości jakie może przyjmować zmienna (np. NAPIĘCIE = -1024 ... +1024);
- **typ tablicowy**, w którym zmienne reprezentowane są poprzez zbiór elementów tego samego typu, a dostęp do tych elementów możliwy jest za pomocą indeksów (wskaźników);
- **typ strukturalny**, definiujący złożone struktury danych, zawierających pola ze zmiennymi o różnych typach [5, 6].

3.4. Synteza układów logicznych

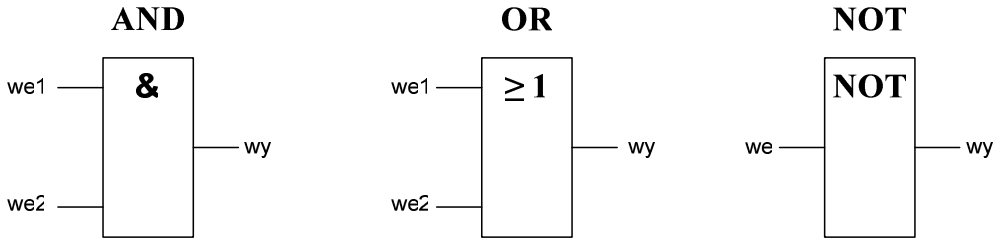
3.4.1. PODSTAWOWE OPERACJE LOGICZNE

Większość elementarnych operacji w układach sterowania procesów dyskretnych wykonywana jest na sygnałach cyfrowych (dwustanowych), których stan w dowolnej chwili można opisać za pomocą wartości logicznych (0 lub 1). Podstawowym nośnikiem informacji logicznej jest bit, stanowiący najmniejszą, niepodzielną komórkę pamięci. W układach sterowania do opisu sygnałów wejściowych i wyjściowych procesu przeważnie stosuje się dyskretną postać, która jednoznacznie opisuje stan różnych urządzeń zewnętrznych, takich jak: przyciski, czujniki, lampki sygnalizacyjne, przekaźniki, elektrozawory itp.

Fundamentem teoretycznym dla cyfrowych układów logicznych jest algebra Boole'a, której podstawowe twierdzenia przedstawił angielski matematyk George Boole już w 1854 roku. Obecnie teoria układów logicznych jest powszechnie wykorzystywana do syntezy i analizy wszelkich układów przełączających.

Funkcja logiczna to taka funkcja, dla której wszystkie sygnały wejściowe i wyjściowe należą do zbioru dwuelementowego, przyjmującego wartości 0 i 1. Podstawowe funkcje logiczne stosowane w technice cyfrowej to:

- iloczyn logiczny – funkcja AND,
- suma logiczna – funkcja OR,
- negacja – funkcja NOT.



we1	we2	wy
0	0	0
0	1	0
1	0	0
1	1	1

we1	we2	wy
0	0	0
0	1	1
1	0	1
1	1	1

we	wy
0	1
1	0

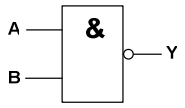
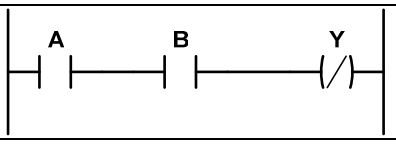
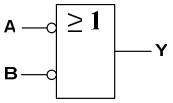
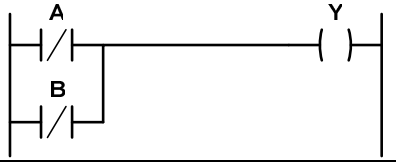
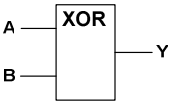
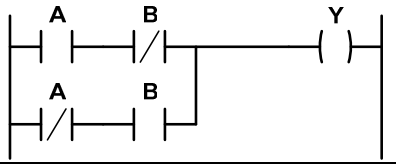
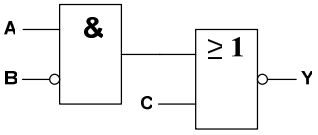
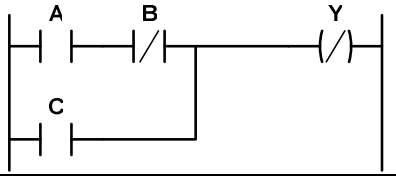
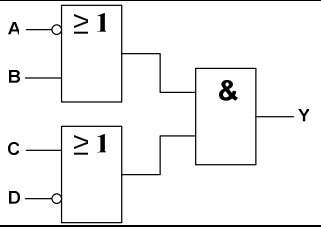
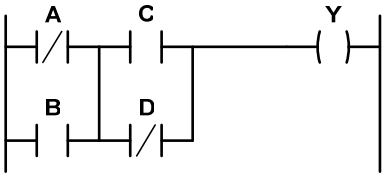
Rys.3.6. Elementarne funkcje logiczne – symbole graficzne i tablice prawdy.

Do opisu działania różnych funkcji logicznych najczęściej stosuje się tzw. tablice prawdy, które przedstawiają wartości wyjściowe funkcji dla wszystkich możliwych kombinacji sygnałów wejściowych. Dla funkcji logicznej o *N*-wejściach, liczba wszystkich możliwych stanów wynosi 2^N. Na rysunku 3.6 przedstawiono symbole graficzne podstawowych funkcji logicznych, oraz odpowiadające im tabele prawdy.

Za pomocą powyższych funkcji logicznych można zbudować dowolnie skomplikowaną relację logiczną, składającą się z wielu sygnałów wejściowych i wyjściowych. W języku FBD dowolne funkcje logiczne przedstawia się w postaci schematów funkcyjnych, zawierających odpowiednio połączone bloki, realizujące podstawowe operacje logiczne: AND i OR. Operację negacji sygnałów wejściowych i wyjściowych bloków przeprowadza się najczęściej za pomocą operatora negacji, oznaczanego symbolem okręgu na wejściu lub wyjściu bloku (Tab.3.6).

W języku drabinkowym, operacja iloczynu logicznego (AND) realizowana jest poprzez szeregowe połączenie „styków" reprezentujących stany logiczne argumentów funkcji. Z kolei połączenie równoległe dwóch lub więcej gałęzi obwodu, traktowane jest jako operacja sumy logicznej (OR). W tabeli 3.6 przedstawiono sposób realizacji kilku wybranych funkcji logicznych w językach LD i FBD.

Tab.3.6. Przykłady realizacji funkcji logicznych w językach FBD i LD.

Wyrażenie logiczne	Realizacja w języku FBD	Realizacja w języku LD
$Y = \overline{A \cdot B}$		
$Y = \overline{A} + \overline{B}$		
$Y = A \oplus B$		
$Y = \overline{A + \overline{B} + C}$		
$Y = (\overline{A} + B) \cdot (C + \overline{D})$		

3.4.2. ZASTOSOWANIE ALGEBRY BOOLE'A DO MINIMALIZACJI FUNKCJI LOGICZNYCH

Algebra Boole'a jest strukturą algebraiczną, w której przedmiotem jest zbiór dwuelementowy **B** = {0, 1} oraz trzy działania z tego zbioru: negacja, suma logiczna i iloczyn logiczny. Reguły algebry boolowskiej stosuje się powszechnie do przekształceń różnych wyrażeń logicznych. Do podstawowych twierdzeń należą:

- Prawa łączności – operacje sumy i iloczynu logicznego są łączne:

$$(a + b) + c = a + (b + c) \tag{3.1}$$

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \tag{3.2}$$

- Prawa przemienności – operacje sumy i iloczynu logicznego są przemienne:

$$a + b = b + a \tag{3.3}$$

$$a \cdot b = b \cdot a \tag{3.4}$$

- Prawa rozdzielności – dodawanie jest rozdzielne względem mnożenia, a mnożenie względem dodawania:

$$(a + b) \cdot c = a \cdot c + a \cdot b \tag{3.5}$$

$$a \cdot b + c = (a + c) \cdot (b + c) \tag{3.6}$$

- Prawa de Morgana:

$$\overline{a + b} = \bar{a} \cdot \bar{b} \tag{3.7}$$

$$\overline{a \cdot b} = \bar{a} + \bar{b} \tag{3.8}$$

- Pozostałe zależności:

$$a + \bar{a} = 1 \tag{3.9}$$

$$a \cdot \bar{a} = 0 \tag{3.10}$$

$$a + 0 = a \tag{3.11}$$

$$a \cdot 1 = a \tag{3.12}$$

$$a \cdot 0 = 0 \tag{3.13}$$

$$a + 1 = 1 \tag{3.14}$$

$$a + a = a \tag{3.15}$$

$$a \cdot a = a \tag{3.16}$$

W teorii układów cyfrowych ważnym zagadnieniem jest minimalizacja funkcji logicznych, która polega na zastosowaniu takiego przekształcenia wyrażenia boolowskiego, aby funkcja kosztów posiadała wartość minimalną. Najczęściej zadania logiczne optymalizuje się pod kątem minimalizacji wielkości programu wynikowego, zatem funkcję kosztów definiuje się tak, aby do realizacji wyrażenia logicznego zastosować jak najmniej zasobów – zmiennych i operatorów. Do minimalizacji funkcji boolowskich powszechnie stosuje się:

- metody algebraiczne wykorzystujące twierdzenia algebry Boole'a,
- metodę tablic Karnaugha,
- metodę Quine'a-McCluskeya,
- metodę Espresso,
- metody iteracyjne.

Korzyści płynące z zastosowania minimalizacji funkcji można zilustrować na prostym przykładzie. Rozważmy pewną trójargumentową funkcję logiczną postaci:

$$Y = A \cdot \left(\overline{A} + AB + BC\right)$$

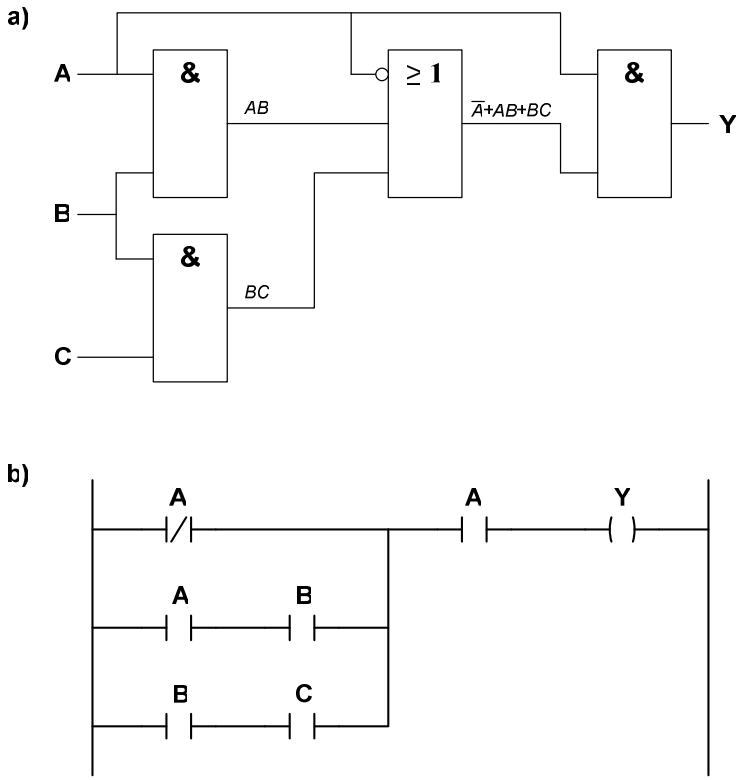
Korzystając z języków FBD oraz LD należy opracować program dla sterownika PLC, obliczający podaną formułę. Na pierwszy rzut oka widać, że do realizacji tej funkcji w języku FBD wymagane jest użycie trzech 2-wejściowych bramek AND oraz jednej 3-wejściowej bramki OR. Analogicznie, w języku drabinkowym zastosujemy odpowiednią topologię połączeń szeregowych i równoległych. Ostatecznie otrzymujemy dwa równoważne programy, których schematy przedstawiono na rysunku 3.7. Analizując otrzymane schematy można stwierdzić, że oba przedstawione programy działają prawidłowo. Pozostaje jednak pytanie: czy przedstawione rozwiązania są optymalne? Czy nie dałoby się czegoś uprościć?

Korzystając z twierdzeń algebry Boole'a spróbujmy teraz uprościć pierwotną funkcję stosując metodę dekompozycji algebraicznej:

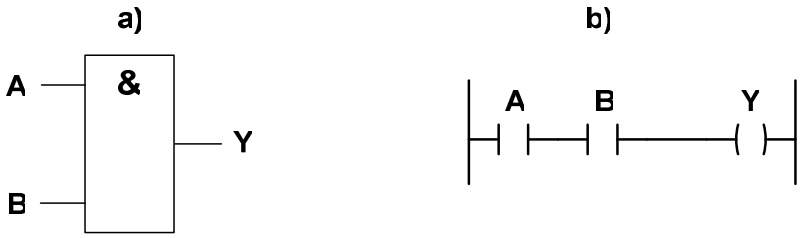
$$A \cdot \left(\overline{A} + AB + BC\right) = A \cdot \overline{A} + A \cdot A \cdot B + A \cdot B \cdot C = \ldots$$

$$\ldots = 0 + A \cdot B + A \cdot B \cdot C = A \cdot B \cdot \left(C + 1\right) = A \cdot B$$

Otrzymany wynik jest zaskakujący. Przedstawione wyrażenie udało uprościć się do postaci jednej dwuwejściowej bramki AND! Poza tym okazało się, że zmienna wejściowa C pierwotnej funkcji nie miała żadnego wpływu na wynik operacji logicznej. Ostatecznie, na rysunku 3.8 przedstawiono programy realizujące zadaną funkcję logiczną po zastosowaniu minimalizacji.



Rys.3.7. Realizacja funkcji logicznej z przykładu: a) – w języku FBD, b) – w języku LD.



Rys.3.8. Programowa realizacja zadanej funkcji logicznej po zastosowaniu minimalizacji: a) – w języku FBD, b) – w języku LD.

Na podstawie powyższego przykładu można wyciągnąć wniosek, że zastosowanie minimalizacji funkcji logicznych umożliwia niekiedy znaczącą redukcję stopnia złożoności programu wynikowego dla sterownika programowalnego, co w rezultacie prowadzi do skrócenia czasu cyklu i zwolnienia zasobów sterownika.

3.5. Standardowe funkcje i bloki funkcjonalne

3.5.1. PODZIAŁ FUNKCJI I BLOKÓW FUNKCJONALNYCH

W części trzeciej normy IEC 61131 określono pewną grupę standardowych funkcji i bloków funkcjonalnych, które mogą występować we wszystkich zdefiniowanych językach programowania sterowników PLC. Przyjęto, że funkcjami będą nazywane elementy realizujące podstawowe operacje logiczne i arytmetyczne, natomiast do bloków funkcjonalnych zaliczać się będą elementy, które charakteryzują się pamięcią stanu (np. przerzutniki, czasomierze, liczniki).

Według normy, funkcje standardowe zostały podzielone na 7 grup:

- Funkcje konwersji typów,
- Funkcje liczbowe,
- Funkcje na ciągach bitów,
- Funkcje wyboru i porównania,
- Funkcje na ciągach znaków,
- Funkcje na typach czasowych,
- Funkcje na typach wyliczeniowych.

Spośród standardowych bloków funkcjonalnych wyróżnić można następujące grupy:

- Elementy bistabilne (dwustanowe),
- Detektory zbocza,
- Liczniki,
- Czasomierze.

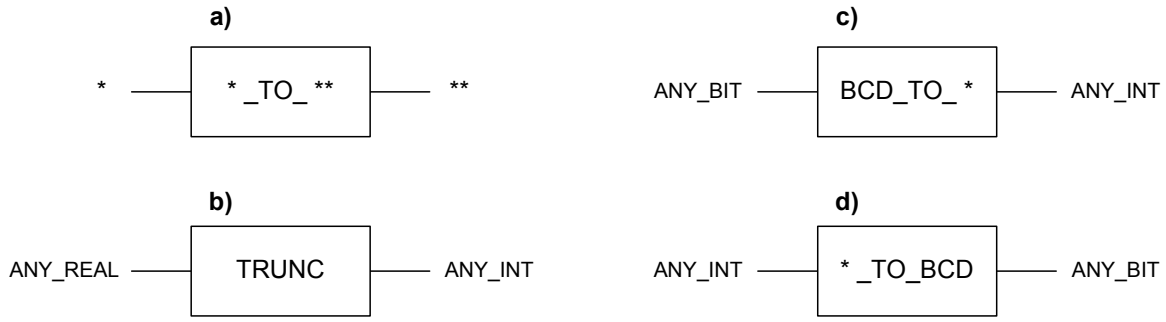
Dostępność wyżej wymienionych elementów funkcyjnych zależy od producenta oraz typu sterownika programowalnego. Może się tak zdarzyć, że oprogramowanie inżynierskie do edycji programów sterujących nie będzie posiadać w swojej bibliotece wszystkich przedstawionych funkcji i bloków standardowych. W tej sytuacji użytkownik musi zastąpić brakujące bloki innymi funkcjami (programami), które realizują tą samą lub podobną procedurę sterującą. Najczęściej jednak jest tak, że oprogramowanie narzędziowe nowoczesnych sterowników PLC wyposażone jest w bogate biblioteki programowe, które zawierają wiele gotowych bloków funkcyjnych, realizujących złożone funkcje, takie jak: funkcje skalowania sygnałów analogowych, procedury dwukierunkowej komunikacji sieciowej, regulatory PID, zaawansowane operacje matematyczne, algorytmy sterowania napędów.

W kolejnych rozdziałach niniejszego skryptu omówiono działanie wybranych funkcji standardowych oraz bloków funkcjonalnych.

3.5.2. FUNKCJE KONWERSJI TYPÓW

Funkcje konwersji typów przetwarzają typy danych, które reprezentują wartości liczbowe, stałe, zmienne oraz sygnały wejściowe i wyjściowe. W wyniku konwersji typu zmienia się sposób interpretacji zapisanych danych, a czasem także wielkość zajmowanego przez nie obszaru w pamięci sterownika.

Ogólna postać funkcji konwersji typu przedstawiona została na rysunku 3.9a. Symbolem (*) zaznaczono typ zmiennej wejściowej, zaś symbol (**) przedstawia wyjściowy typ zmiennej, po przetworzeniu. Przykładowe funkcje konwersji typów to: *INT_TO_REAL*, *BYTE_TO_WORD, BCD_TO_INT,* itp. Do funkcji konwertującej typ danej należy także funkcja *TRUNC* (rys.3.9b), która obcina część ułamkową liczby typu rzeczywistego, przekształcając ją do postaci liczby całkowitej.



Rys.3.9. Przykładowe funkcje konwersji typów.

Wśród funkcji standardowych, zdefiniowanych przez normę, spotyka się również takie funkcje, które operują na zmiennych o dowolnych typach (tzw. typ *ANY*). Funkcje takie nazywa się funkcjami przeciążonymi (ang. *overloaded functions*). Na rysunku 3.9c–d przedstawiono funkcje dwustronnej konwersji pomiędzy typami całkowitymi i BCD. W tym wypadku zmienna typu *ANY_BIT* może być reprezentowana przez dowolny typ binarny (*BYTE, WORD, DWORD, LWORD*), zaś zmienna *ANY_INT* może wyrażać dowolną liczbę całkowitą w formacie *INT, SINT, DINT* czy *LINT* [5].

3.5.3. FUNKCJE LICZBOWE

Funkcje liczbowe wykonują operacje na liczbach o różnych typach danych. Ogólnie dzielą się na dwie grupy:

- funkcje jednej zmiennej (np. pierwiastek kwadratowy, funkcje logarytmiczne, trygonometryczne),
- funkcje wielu zmiennych (funkcje arytmetyczne takie jak: dodawanie, odejmowanie, mnożenie, dzielenie itp.).

Funkcje jednej zmiennej występują jako funkcje przeciążone w zakresie typów uniwersalnych (np. *ANY_REAL, ANY_INT*), lub jako funkcje o określonym typie danych. Funkcje wielu zmiennych mogą wykonywać operacje arytmetyczne na danych liczbowych należących do tego samego typu. W grupie tych funkcji można wyróżnić:

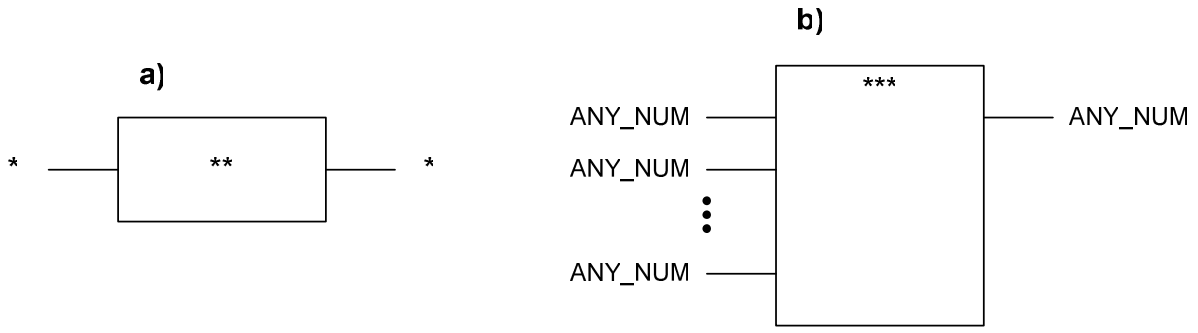
- funkcje o rozszerzalnej liczbie wejść (np. dodawanie, mnożenie);
- funkcje o stałej liczbie wejść (np. dzielenie, odejmowanie).

Na rysunku 3.10a przedstawiono ogólną postać funkcji liczbowej jednej zmiennej. Symbol (*) oznacza typ danych wejściowych i wyjściowych, natomiast (**) oznacza nazwę funkcji. Z kolei rysunek 3.10b przedstawia ogólną postać funkcji wielu zmiennych, gdzie symbol (***) oznacza nazwę funkcji, a uniwersalny typ danych *ANY_NUM* może reprezentować dowolne liczby całkowite (*INT, SINT,* ...) lub rzeczywiste (*REAL, LREAL*).

W tabeli 3.7 przedstawiono wybrane standardowe funkcje liczbowe jednej i wielu zmiennych.

Tab.3.7. Standardowe funkcje liczbowe.

Nazwa funkcji	**Opis**
ABS	Wartość bezwzględna
SQRT	Pierwiastek kwadratowy
LN	Logarytm naturalny
LOG	Logarytm dziesiętny
EXP	Funkcja wykładnicza ex
SIN	Sinus [rad]
COS	Cosinus [rad]
TAN	Tangens [rad]
ASIN	Arcus sinus
ACOS	Arcus cosinus
ATAN	Arcus tangens
ADD	Dodawanie
MUL	Mnożenie
SUB	Odejmowanie
DIV	Dzielenie
MOD	Reszta dzielenia całkowitego



Rys.3.10. Ogólna postać funkcji liczbowych: a) – jednej zmiennej, b) – wielu zmiennych.

3.5.4. FUNKCJE NA CIĄGACH BITÓW

Funkcje standardowe operujące na ciągach bitów przeprowadzają operacje bitowe na danych należących do dowolnego typu binarnego *ANY_BIT* (np. *BYTE, WORD, DWORD*). Funkcje te dzielą się na:

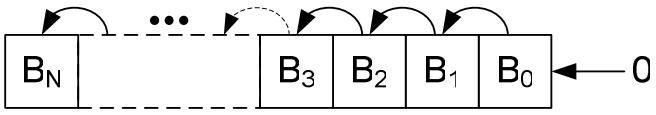
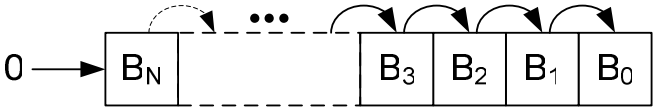
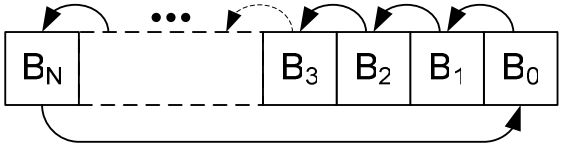
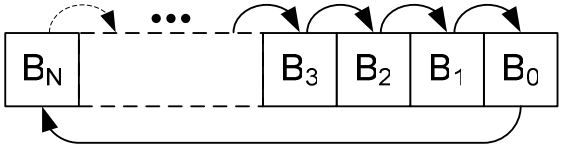
- funkcje przesuwania bitów,
- funkcje logiczne.

Na rysunku 3.11 przedstawiono ogólną postać funkcji przesuwania bitów (a) oraz standardowej funkcji logicznej (b). Symbol (***) określa nazwę funkcji.

Rys.3.11. Ogólna postać funkcji na ciągach bitów:
a) – funkcja przesuwania bitów, b) – funkcja logiczna.

Funkcje przesuwania bitów są zazwyczaj dwuargumentowe: pierwsze wejście jest przeznaczone dla danej wejściowej typu uniwersalnego *ANY_BIT*, a na drugie wejście podaje się liczbę rzeczywistą typu *ANY_INT*, określającą krotność przesunięcia bitowego. Standardowe funkcje logiczne, za wyjątkiem funkcji NOT, przeważnie występują jako dwuargumentowe lub rozszerzalne i wykonują operacje na danych typu *ANY_BIT*. W tabeli 3.8 przedstawiono standardowe funkcje na ciągach bitów.

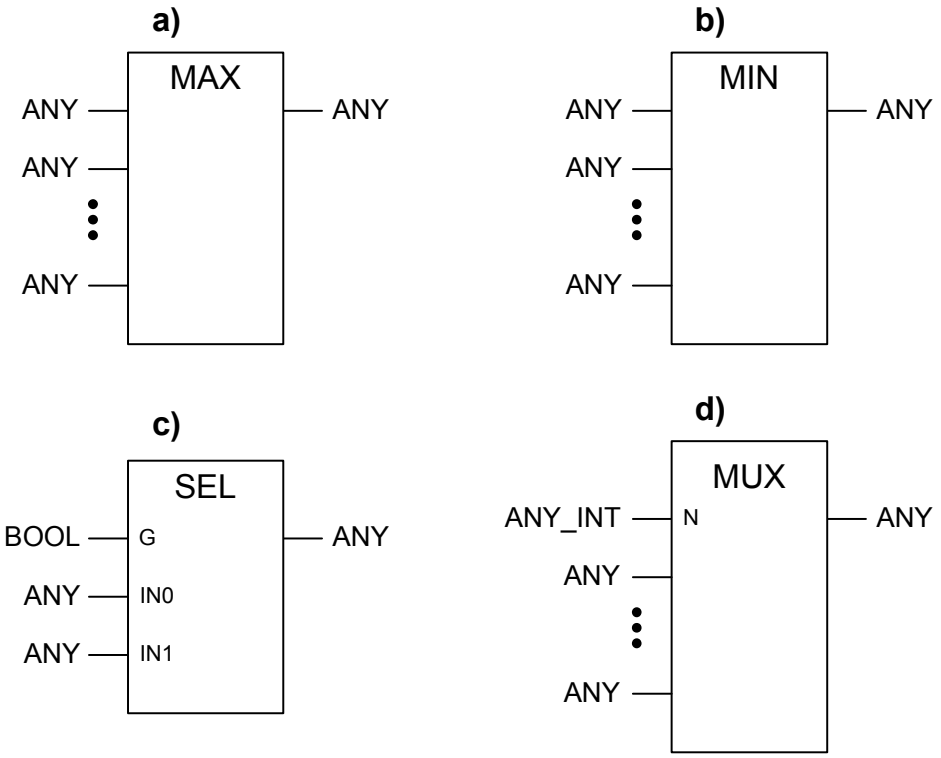
Tab.3.8. Standardowe funkcje na ciągach bitów.

Nazwa funkcji	Opis działania
Standardowe funkcje przesuwania bitów	
SHL	Przesunięcie logiczne bitów w lewo o N pozycji 
SHR	Przesunięcie logiczne bitów w prawo o N pozycji 
ROL	Przesunięcie cykliczne (rotacja) bitów w lewo o N pozycji 
ROR	Przesunięcie cykliczne (rotacja) bitów w prawo o N pozycji 
Standardowe funkcje logiczne	
AND	Iloczyn logiczny
OR	Suma logiczna
XOR	Alternatywa wykluczająca (*Exclusive OR*)
NOT	Negacja

3.5.5. FUNKCJE WYBORU I PORÓWNANIA

Na rysunku 3.12 przedstawiono standardowe funkcje wyboru, które operują na argumentach typu uniwersalnego *ANY*, przy założeniu, że dane wejściowe i wyjściowe są tego samego typu. Funkcja MAX znajduje wartość maksymalną spośród

wszystkich sygnałów wejściowych i wyprowadza ją na wyjście funkcji. Analogicznie działa funkcja MIN, z tą różnicą, że wyszukuje wartość najmniejszą. Funkcja SEL jest funkcją wyboru jednego z dwóch wejść (przełącznik). Jeżeli na wejściu G panuje niski stan logiczny, to na wyjściu pojawia się taka sama wartość jak na wejściu IN0. Jeżeli wejście G przyjmie wysoki stan logiczny, to do wyjścia zostaje podłączony sygnał z wejścia IN1. Funkcja MUX jest to multiplekser *N*-wejściowy (przełącznik 1 z N), który przenosi na wyjście sygnał z wejścia o numerze określonym przez parametr K. Sygnał sterujący K musi być liczbą całkowitą z przedziału (0 ... N-1), w przeciwnym wypadku zostanie wygenerowany błąd systemu [5].



Rys.3.12. Standardowe funkcje wyboru: a) – maksimum, b) – minimum, c) – wybór 1 z 2, d) – multiplekser.

W tabeli 3.9 zestawiono standardowe funkcje porównania – tzw. komparatory. Funkcje te operują na danych wejściowych dowolnego typu. Wyjściem komparatorów jest sygnał logiczny, który przyjmuje wartość logiczną 1, wtedy gdy warunek porównania jest spełniony. Funkcje te występują przeważnie jako dwuargumentowe, chociaż spotyka się czasem komparatory wielowejściowe, które realizują sekwencję operacji relatywistycznych. W tym przypadku funkcja przyjmuje na wyjściu wartość logiczną „1", jeżeli poszczególne warunki porównania są spełnione w ustalonej kolejności – od wejścia pierwszego do ostatniego (np. dla komparatora GT musi zachodzić relacja: $WE_1 > WE_2 > WE_3 > ... > WE_N$).

Tab.3.9. Standardowe funkcje porównania.

Nazwa	Symbol	Opis działania
GT	>	Komparator „większy niż” (ang. *Greater Than*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 > WE_2 > ... > WE_N$
GE	>=	Komparator „większy lub równy” (ang. *Greater or Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 \geq WE_2 \geq ... \geq WE_N$
EQ	=	Komparator „równy” (ang. *Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 = WE_2 = ... = WE_N$
LE	<=	Komparator „mniejszy lub równy” (ang. *Less or Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 \leq WE_2 \leq ... \leq WE_N$
LT	<	Komparator „mniejszy niż” (ang. *Less Than*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 < WE_2, < ... < WE_N$
NE	<>	Komparator „nie równy” (ang. *Not Equal*) Wyjście przyjmuje stan 1, gdy zachodzi relacja: $WE_1 \neq WE_2 \neq ... \neq WE_N$

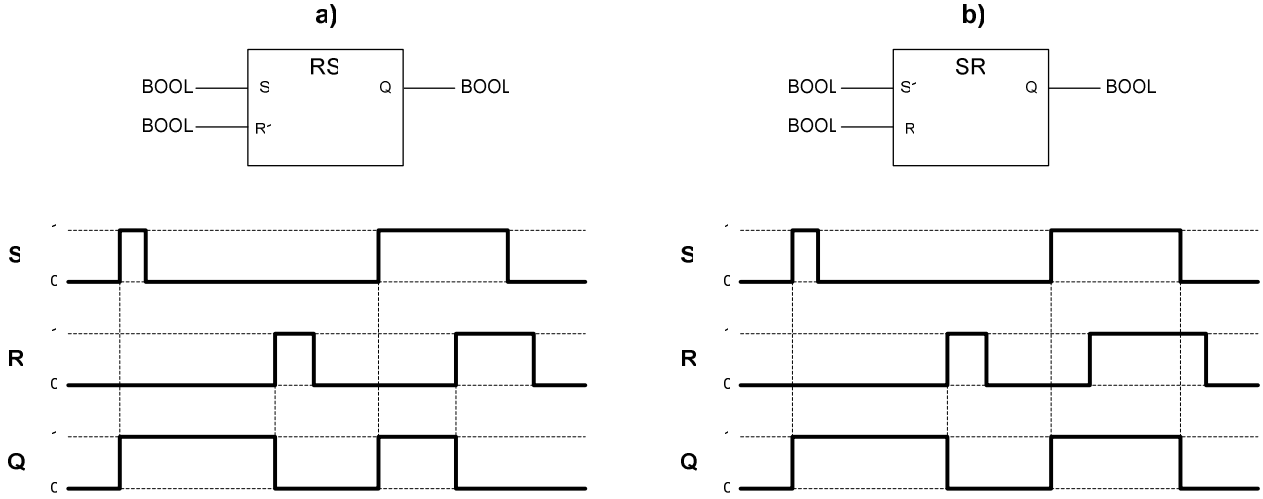
3.5.6. ELEMENTY BISTABILNE

Elementy bistabilne (dwustanowe) są zaliczane do grupy standardowych bloków funkcjonalnych, które występują w każdym sterowniku PLC. Do podstawowych przedstawicieli tej grupy należą dwa rodzaje przerzutników bistabilnych: RS i SR. Przerzutniki są to elementy dwuwejściowe, które charakteryzują się pamięcią stanu. Oznacza to, że stan logiczny wyjścia przerzutnika zależy nie tylko od aktualnych stanów logicznych na jego wejściach, ale także od zmian wartości sygnałów wejściowych występujących w przeszłości.

Na rysunku 3.13 przedstawiono symbole graficzne oraz wykresy czasowe wyjaśniające działanie przerzutników RS i SR. Załączenie wyjścia przerzutników następuje z chwilą pojawienia się stanu wysokiego na wejściu ustawiającym S (Set), natomiast wyłączenie wyjścia występuje po podaniu stanu wysokiego na wejście kasujące R (Reset). Różnica pomiędzy tymi elementami dotyczy specyfiki działania, przy jednoczesnym podaniu wartości logicznej „1” na oba wejścia (R i S). Otóż w przerzutniku RS dominującym wejściem jest wejście kasujące R, zaś w przerzutniku SR priorytetowym jest sygnał ustawiający S.

W języku drabinkowym często do realizacji funkcji przerzutników stosuje się rozdzielone pary sygnałów wyjściowych (cewek) S i R. W tym przypadku określenie dominującej funkcji przerzutnika zależy od kolejności występowania cewek w programie. Jeśli obwód z cewką R występuje w programie jako ostatni, wówczas

realizowana jest funkcja przerzutnika RS, a jeśli ostatnią użytą cewką jest cewka ustawiająca S, to jest to przerzutnik SR.



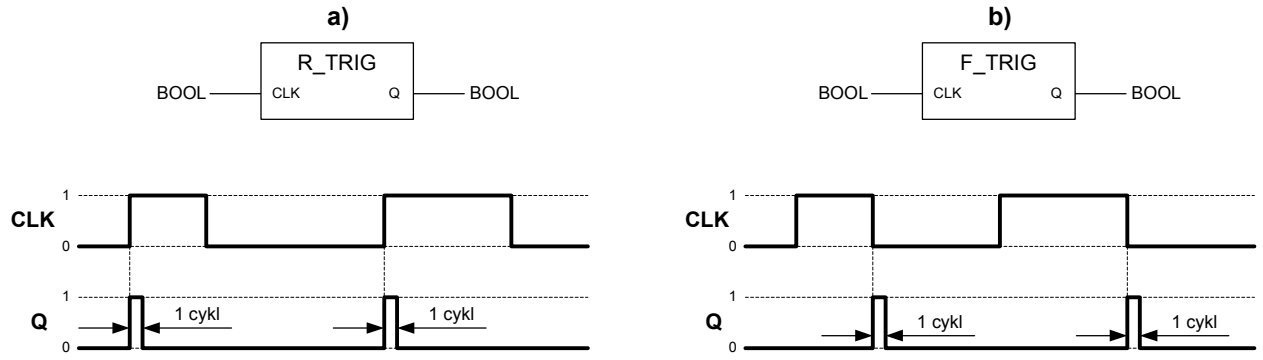
Rys.3.13. Symbole graficzne i zasada działania przerzutników RS (a) i SR (b).

3.5.7. DETEKTORY ZBOCZA

Do standardowych bloków funkcjonalnych, umożliwiających detekcję zbocza sygnałów cyfrowych należą:

- detektor zbocza narastającego R_TRIG (ang. *Rising Trigger*),
- detektor zbocza opadającego F_TRIG (ang. *Falling Trigger*).

Elementy te należą do bloków dynamicznych, gdyż reagują na zmianę wartości sygnałów wejściowych. Na rysunku 3.14 przedstawiono symbole graficzne oraz przebiegi czasowe obrazujące zasadę działania tych elementów.



Rys.3.14. Symbole graficzne i zasada działania detektorów zbocza: R_TRIG (a) i F_TRIG (b).

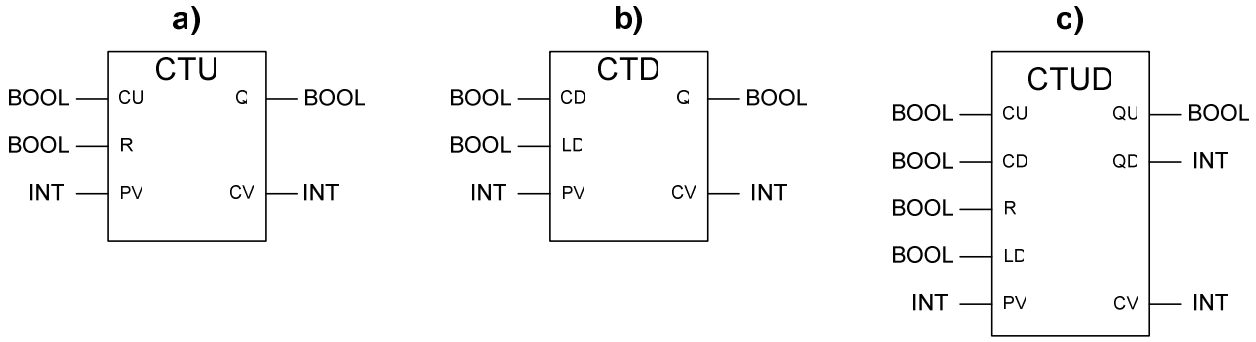
Detektor R_TRIG wykrywa zmianę stanu logicznego na wejściu CLK z wartości 0 na 1, powodując ustawienie na wyjściu wysokiego stanu logicznego na czas jednego cyklu przetwarzania sterownika. Detektor F_TRIG działa podobnie, z tą różnicą, że reaguje na zmianę stanu logicznego z 1 na 0. Pomimo, że czas trwania wysokiego stanu logicznego na wyjściach tych bloków jest bardzo krótki, detektory zbocza umożliwiają ustawienie lub skasowanie przerzutników, taktowanie liczników czy też wyzwolenie czasomierzy.

3.5.8. LICZNIKI

Liczniki należą do standardowych bloków funkcjonalnych, umożliwiających zliczanie impulsów. W grupie tych elementów wyróżnia się:

- licznik zliczający w górę CTU (ang. Counter Up),
- licznik odliczający w dół CTD (ang. Counter Down),
- licznik dwukierunkowy CTUD (ang. Counter Up-Down).

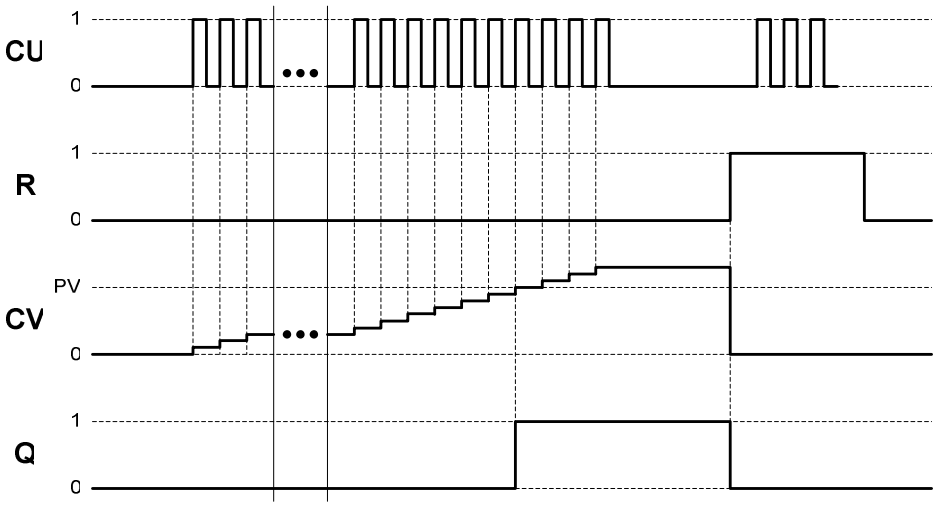
Na rysunku 3.15 przedstawiono symbole graficzne podstawowych liczników.



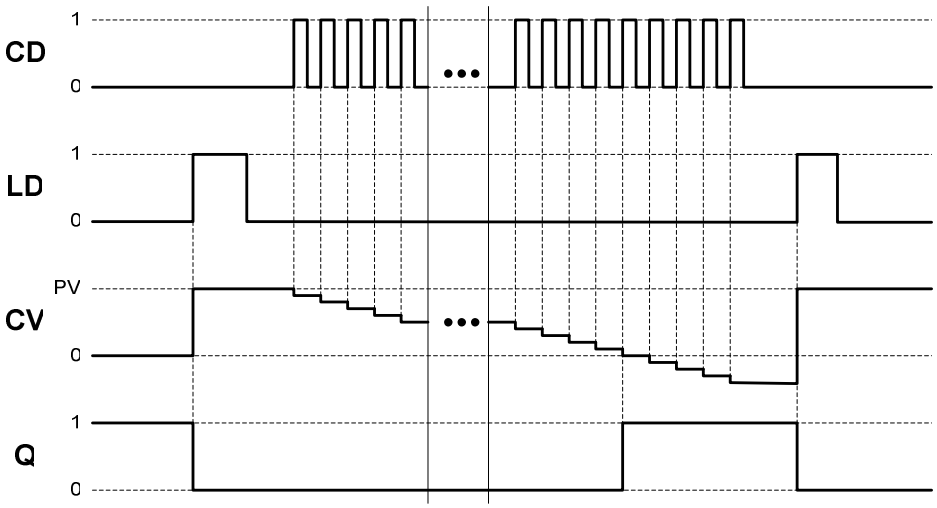
Rys.3.15. Symbole graficzne liczników: a) – CTU, b) – CTD, c) – CTUD.

Na rysunku 3.16 przedstawiono wykres czasowy wyjaśniający działanie licznika CTU. Każdy impuls wejściowy, rozumiany jako zmiana stanu logicznego na wejściu CU z 0 na 1, powoduje inkrementację licznika, którego aktualna wartość dostępna jest na wyjściu CV (ang. *Current Value*). Wyjście Q zostanie ustawione, jeżeli wartość licznika CV osiągnie lub przekroczy wartość zadaną na wejściu PV (ang. *Preset Value*). Podanie wysokiego stanu logicznego na wejście R (*Reset*) powoduje wyzerowanie licznika.

Z kolei na rysunku 3.17 przedstawiono wykres czasowy objaśniający działanie licznika CTD. Zmiana stanu na wejściu CD z wysokiego na niski powoduje dekrementację wartości licznika CV. Wyjście Q przyjmuje wysoki stan logiczny, jeśli wartość licznika jest równa lub mniejsza od zera. Wejście LD (*Load*) powoduje wpisanie do licznika wartości zadanej, podanej na wejściu PV.



Rys.3.16. Wykres czasowy objaśniający działanie licznika CTU.



Rys.3.17. Wykres czasowy objaśniający działanie licznika CTD.

Licznik CTUD jest połączeniem liczników CTU i CTD i umożliwia dwukierunkowe zliczanie impulsów. Posiada dwa wyjścia cyfrowe, na których sygnalizuje następujące stany:

- wyjście QU zostaje ustawione wtedy, gdy CV ≥ PV
- wyjście QD zostaje ustawione wtedy, gdy CV ≤ 0.

Licznik ten posiada cztery wejścia cyfrowe o następujących funkcjach:

- CU – zbocze narastające na tym wejściu powoduje inkrementację wartości CV;
- CD – zbocze narastające na tym wejściu powoduje dekrementację wartości CV;
- R – wejście kasujące, stan wysoki na tym wejściu wpisuje wartość 0 do CV;

- LD – wejście ładowania wartości początkowej – stan wysoki powoduje wpisanie do CV wartości z wejścia PV.

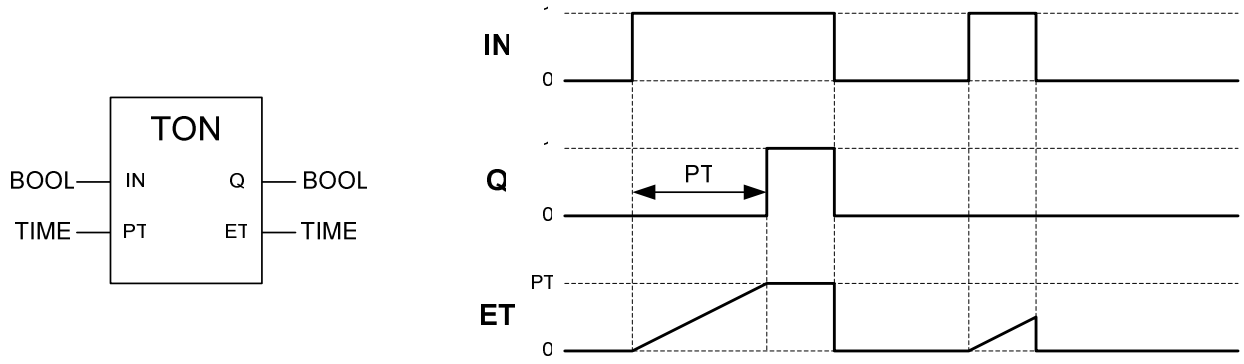
Podczas pracy z licznikami należy pamiętać, że elementy te w żaden sposób nie ograniczają wartości osiąganych na wyjściu CV. Oznacza to, że liczniki CTU i CTUD po osiągnięciu wartości zadanej PV, będą nadal zliczać impulsy pojawiające się na wejściu CU, powodując inkrementację wartości CV. Analogicznie, liczniki CTD i CTUD po osiągnięciu wartości zerowej będą w dalszym ciągu zliczać impulsy na wejściu CD, co spowoduje, że wyjście CV osiągnie wartości ujemne.

3.5.9. CZASOMIERZE

Czasomierze (ang. *Timers*), potocznie nazywane *tajmerami*, należą do standardowych bloków funkcjonalnych, które realizują funkcje sterowania z uwzględnieniem uwarunkowań czasowych. Norma IEC 61131 w grupie czasomierzy definiuje następujące elementy podstawowe [5]:

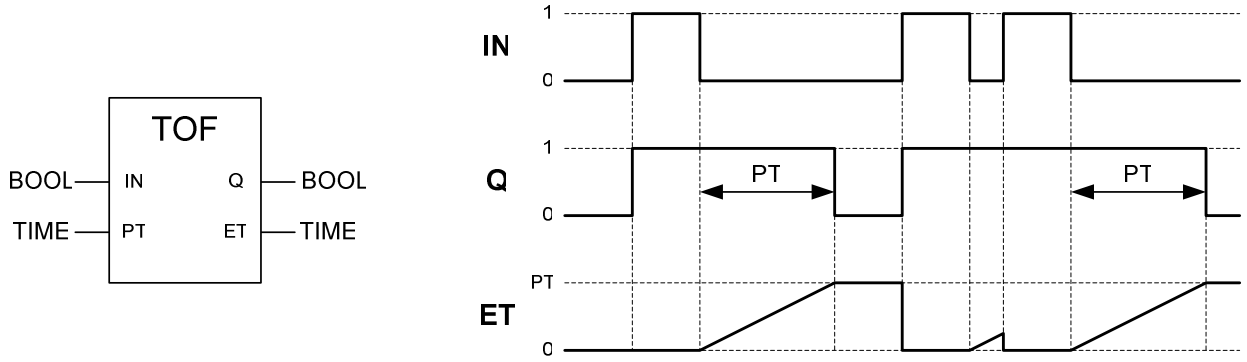
- czasomierz TON (*ON-delay*), realizujący funkcję opóźnionego załączenia;
- czasomierz TOF (*Off-delay*), realizujący funkcję opóźnionego wyłączenia;
- czasomierz TP (*Pulse*), realizujący funkcję generatora monostabilnego;

Na rysunku 3.18 przedstawiono symbol graficzny i wykres czasowy ilustrujący zasadę działania czasomierza TON. Blok ten steruje załączeniem wyjścia Q z opóźnieniem w stosunku do sygnału wejściowego. Podanie wysokiego stanu logicznego na wejście IN powoduje rozpoczęcie zliczania czasu ET (ang. *Elapsed Time*), którego wartość jest wyprowadzona na wyjściu ET. Jeżeli czas ET osiągnie wartość podaną na wejściu PT (ang. *Preset Time*), nastąpi zatrzymanie odliczania czasu i wysterowanie wyjścia Q. Stan ten utrzymuje się dopóty, dopóki na wejściu IN panuje wysoki stan logiczny. Z chwilą, gdy na wejściu IN pojawi się wartość logiczna 0, następuje wyzerowanie czasomierza i natychmiastowe wyłączenie wyjścia Q.



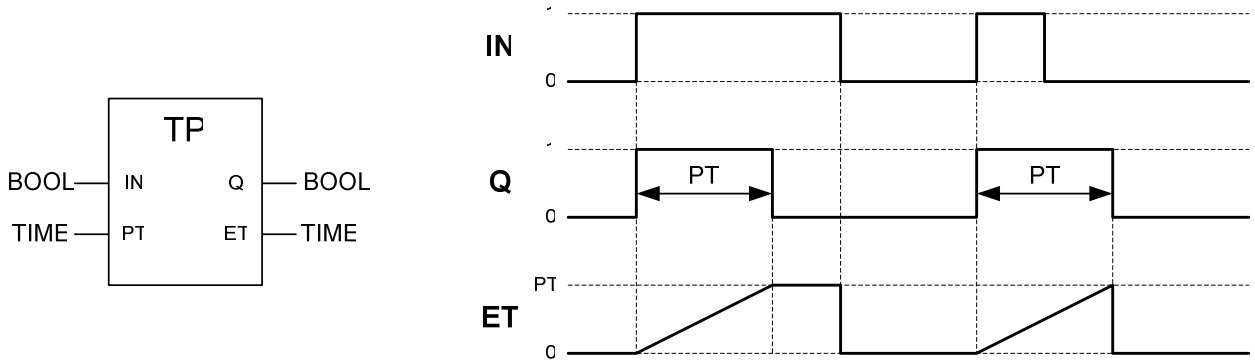
Rys.3.18. Symbol graficzny i wykres czasowy ilustrujący działanie czasomierza TON.

Czasomierz TOF realizuje funkcję przekaźnika czasowego, wyłączającego wyjście Q z opóźnieniem w stosunku do wejścia sterującego IN. Symbol graficzny i wykres czasowy ilustrujący zasadę działania czasomierza TOF zostały przedstawione na rysunku 3.19. Wartość opóźnienia zadawana jest na wejściu PV. Podanie wysokiego stanu logicznego na wejście IN powoduje natychmiastowe wysterowanie wyjścia Q. Z chwilą zaniku sygnału wyzwalającego IN rozpoczyna się odliczanie czasu ET. Jeżeli czas ET osiągnie wartość podaną na wejściu PV, nastąpi wyłączenie wyjścia Q. Warunek ten będzie spełniony, jeżeli w międzyczasie nie zostanie ponownie ustawione wejście sterujące IN, powodując wyzerowanie czasu ET.



Rys.3.19. Symbol graficzny i wykres czasowy ilustrujący działanie czasomierza TOF.

Czasomierz TP generuje na wyjściu Q impuls o ustalonym czasie trwania, którego wartość jest zadawana na wejściu PT. Warunkiem wyzwolenia czasomierza jest pojawienie się zbocza narastającego sygnału na wejściu sterującym IN. Czas trwania stanu wysokiego na wyjściu Q jest stały i nie zależy od czasu trwania sygnału wyzwalającego. Symbol graficzny i wykres czasowy czasomierza TP przedstawiono na rysunku 3.20.

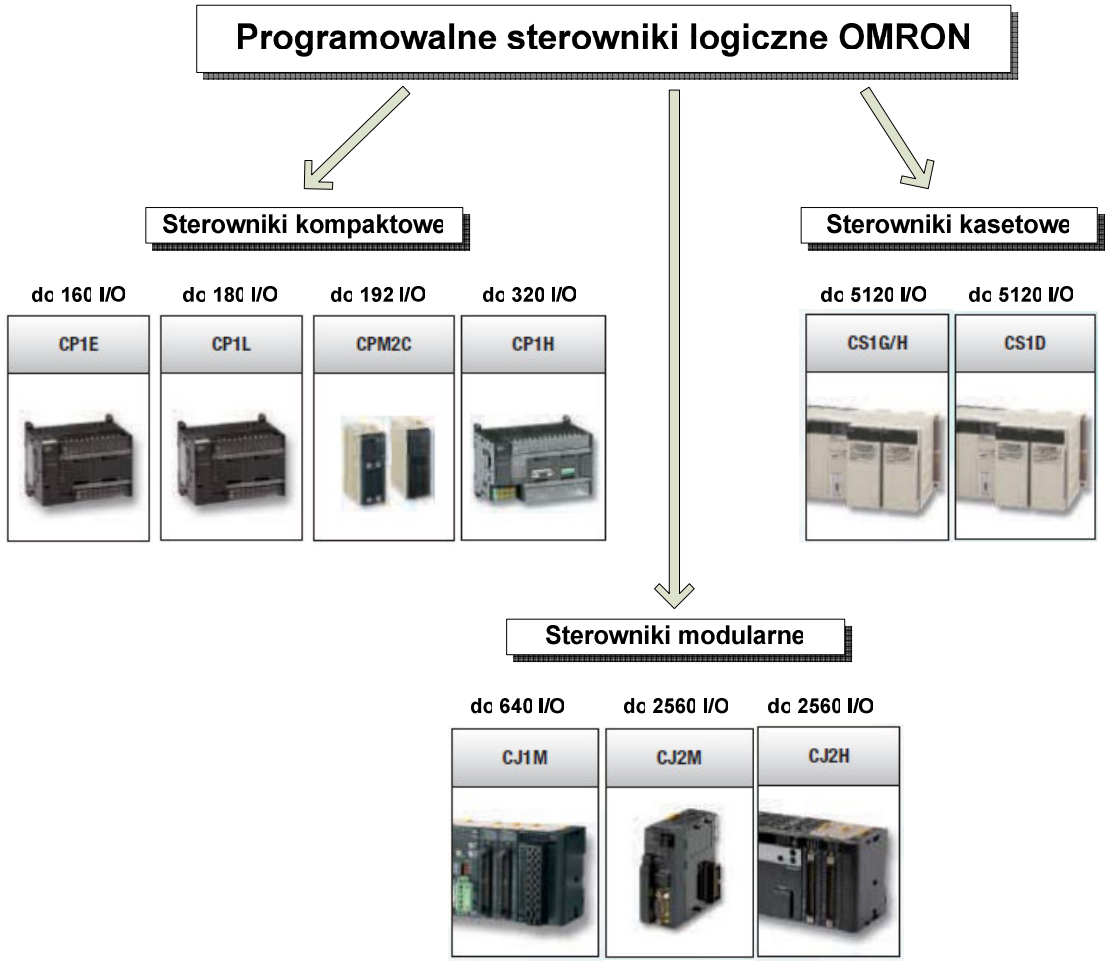


Rys.3.20. Symbol graficzny i wykres czasowy ilustrujący działanie czasomierza TP.

4. Sterowniki programowalne OMRON z rodziny CJ

4.1. Rodzina sterowników OMRON

Na rysunku 4.1 przedstawiono klasyfikację aktualnie produkowanych sterowników OMRON [17].



Rys.4.1. Klasyfikacja sterowników PLC firmy OMRON.

Firma OMRON obecnie jest jednym z największych producentów systemów i urządzeń automatyki przemysłowej na świecie. W swojej ofercie posiada kilkanaście modeli programowalnych sterowników logicznych, uszeregowanych w trzech grupach, w zależności od ich wielkości i możliwości funkcjonalnych:

- **sterowniki kompaktowe**, które posiadają zintegrowany zasilacz oraz wbudowane wejścia i wyjścia. Do grupy tej zalicza się zarówno proste, nierozszerzalne sterowniki 10-punktowe typu CPM1A, jak i zawansowane sterowniki CP1H.
- **sterowniki modułowe serii CJ,** obsługujące do 2500 punktów I/O – seria sterowników elastycznie konfigurowalnych, posiadających szeroką gamę wymiennych modułów CPU, zestawów wejść-wyjść oraz modułów komunikacyjnych.
- **zaawansowane sterowniki serii CS1**, instalowane na płytach montażowych (kasetach), obsługujące do 5000 zmiennych procesowych, umożliwiające redundancję jednostek centralnych, znajdujące zastosowanie w przemysłowych systemach sterowania automatyki procesowej.

4.2. Sterownik CJ1M

4.2.1. BUDOWA I PARAMETRY STEROWNIKA

Sterownik programowalny CJ1M jest przedstawicielem grupy sterowników modularnych, należących do wysokowydajnej rodziny CJ1. W grupie tej oferowanych jest kilka modeli jednostek centralnych CPU, różniących się między sobą parametrami i możliwościami funkcyjnymi. Jednostki CPU2x wyposażone są w zestawy szybkich wejść i wyjść cyfrowych, dzięki czemu mogą być wykorzystane w aplikacjach sterujących napędami elektrycznymi. Z kolei jednostki centralne typu CPUxx-ETN posiadają zintegrowany moduł komunikacyjny sieci Ethernet, który umożliwia wymianę danych pomiędzy innymi sterownikami, panelami operatorskimi oraz komputerami z oprogramowaniem typu SCADA. W tabeli 4.1 przedstawiono wybrane parametry, którymi różnią się poszczególne jednostki centralne sterowników serii CJ1M [11].

Możliwości funkcjonalne i zasoby sprzętowe sterowników serii CJ1M zależą od doboru modułów rozszerzeń, do których należą:

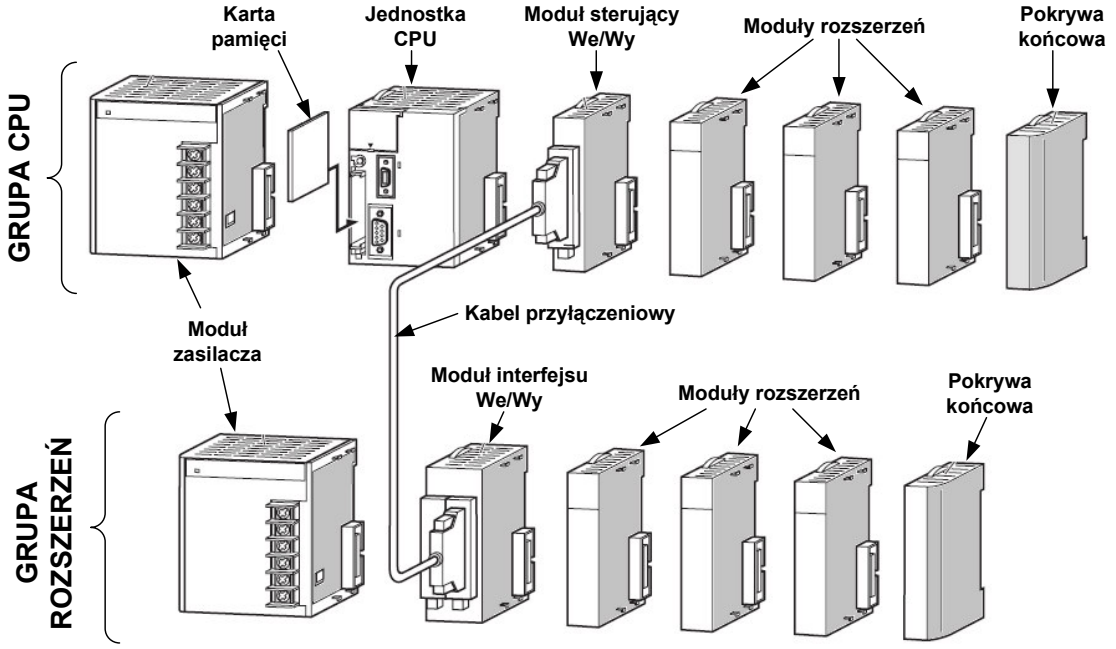
- moduł zasilacza – dostarcza odpowiednich napięć do zasilania jednostki centralnej oraz poszczególnych modułów rozszerzeń;
- podstawowe moduły wejść/wyjść – stanowią interfejs cyfrowy sterownika i zawierają: moduły wejść dyskretnych (16, 32, 64) oraz moduły wyjść dyskretnych (8, 16, 32, 64);
- specjalne moduły wejść/wyjść, do których zalicza się: moduły z wejściami/wyjściami analogowymi, regulatory temperatury, moduły z szybkimi licznikami oraz moduły pozycjonujące;

- moduły komunikacyjne, wśród których występują m.in.: moduły komunikacji szeregowej, moduły sieci Ethernet, moduły sieci DeviceNet, moduły sieci Profibus DP;
- moduły sterujące, które umożliwiają dołączenie dodatkowych grup rozszerzeń do sterownika.

Tab.4.1. Wybrane parametry jednostek centralnych sterowników rodziny CJ1M.

Jednostka centralna	Maksymalna liczba cyfrowych wejść/wyjść	Pojemność programu [kroków]	Maksymalna liczba modułów rozszerzeń	Funkcje wbudowane
CPU11	160	5.000	10	
CPU11-ETN			9	Port 100 base-Tx Ethernet
CPU12	320	10.000	10	
CPU12-ETN			9	Port 100 base-Tx Ethernet
CPU13	640	20.000	20	
CPU13-ETN			19	Port 100 base-Tx Ethernet
CPU21	160	5.000	10	2 wejścia enkodera (100kHz) 2 wyjścia impulsowe (100kHz) 4 wejścia przerwaniowe / wejścia licznika
CPU22	320	10.000	10	
CPU23	640	20.000	20	

Na rysunku 4.2 przedstawiono przykładową konfigurację sprzętową sterownika z rodziny CJ1M.



Rys.4.2. Przykładowa konfiguracja sprzętowa sterownika CJ1M.

Podstawowe jednostki centralne CPUx1 oraz CPUx2 umożliwiają dołączenie maksymalnie 10 modułów rozszerzeń. Minimalna konfiguracja sprzętowa sterownika wymaga obecności zasilacza, jednostki centralnej oraz pokrywy krańcowej, która niezbędna jest do prawidłowego działania magistrali rozszerzeń sterownika. Jednostki centralne typu CPUx3 umożliwiają dołączenie dodatkowej grupy modułów rozszerzeń, za pomocą specjalnego modułu sterującego We/Wy. Wówczas maksymalna liczba modułów obsługiwanych przez sterownik wzrasta do 20.

4.2.2. Jednostka centralna CJ1M CPU12-ETN

Jednostka centralna CJ1M CPU12-ETN jest odmianą bardzo popularnej, wysokowydajnej jednostki CPU sterowników rodziny CJ, zawierającej w jednej obudowie zintegrowany moduł komunikacyjny sieci Ethernet. Oprócz możliwości komunikacyjnych, rozwiązanie to umożliwia zaprogramowanie sterownika za pomocą dowolnego komputera PC wyposażonego w standardową kartę sieciową. W tabeli 4.2 zestawiono podstawowe parametry jednostki sterownika CJ1M z jednostką centralną CPU12-ETN [11].

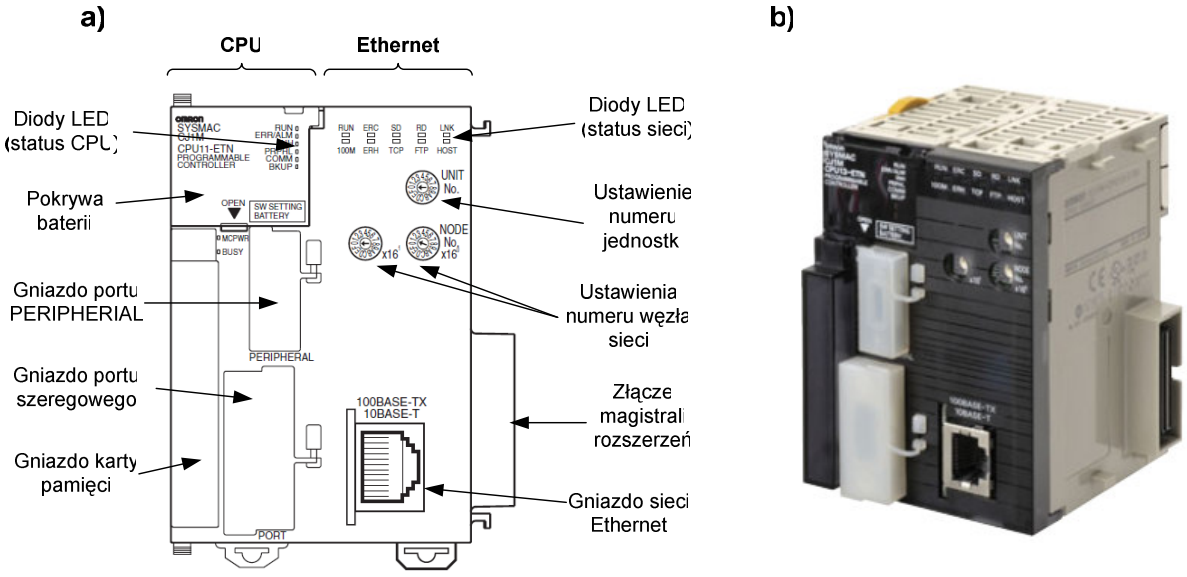
Tab.4.2. Wybrane parametry sterownika OMRON CJ1M CPU12-ETN.

Obsługiwana liczba wejść/wyjść (*I/O points*)		320
Pamięć programu użytkownika		10 tys. kroków
Maksymalna liczba modułów rozszerzeń		9
Pamięć danych		32 tys. słów
Liczba wejść przerwaniowych		1
Maksymalna liczba podprogramów		256
Maksymalna liczba instrukcji skoku (JMP)		256
Bloki funkcyjne (FB)	maksymalna liczba definicji	128
	maksymalna liczba wystąpień	256
Pamięć Flash	Pamięć dla bloków funkcyjnych	256kB
	Obszar komentarzy	64kB
	Obszar *Program Index File*	64kB
	Symbole zmiennych	64kB
Czas wykonania instrukcji	Instrukcje podstawowe	100ns
	Instrukcje specjalne	300ns
Czas przetwarzania		0,5ms
Maksymalna liczba zadań (*Tasks*)	Zadania przerwaniowe	256
	Zadania cykliczne	32
Zakres ustawień minimalnego czasu trwania cyklu		1–32ms

Oprócz wbudowanego modułu sieci Ethernet, sterownik ten wyposażony jest w dwa złącza komunikacji szeregowej: zwykły port szeregowy zgodny ze standardem RS-232C oraz dodatkowy port „Peripherial", który jest głównie wykorzystywany do

komunikacji sieciowej między sterownikami OMRON. Główny port szeregowy umożliwia podłączenie dowolnego urządzenia zewnętrznego obsługującego standard RS-232C, przy czym użytkownik może zdefiniować własny protokół wymiany danych. Ponadto oba porty obsługują komunikację szeregową przy wykorzystaniu protokołów *Host Link*, *NT Links* czy *Serial Gateway (CompoWay/F master)*. Przy podłączeniu zewnętrznego urządzenia należy pamiętać o zastosowaniu specjalnego kabla do transmisji szeregowej, ponieważ rozmieszczenie sygnałów w złączu DSUB9 jest niestandardowe.

Na rysunku 4.3 przedstawiono rozmieszczenie elementów funkcjonalnych na panelu sterownika oraz jego fotografię. W górnej części obudowy znajdują się diody LED, które sygnalizują status sterownika (napięcie zasilania, stan pracy CPU, błędy sprzętowe i programowe). Pod pokrywą znajduje się bateria podtrzymująca napięcie zasilania oraz zestaw przełączników typu *DIP-switch* służących do zaawansowanych ustawień sprzętowych sterownika (ochrona zapisu do pamięci, transfer programu do/z karty pamięci, wybór nastaw domyślnych portów szeregowych) [11, 17].



Rys.4.3. Jednostka centralna CJ1M CPU12-ETN: a) – rozmieszczenie elementów funkcjonalnych , b) – fotografia modułu.

Poniżej, po lewej stronie gniazd portów szeregowych znajduje się gniazdo karty pamięci typu Compact Flash. Karta ta jest opcjonalna i może być wykorzystana do przechowywania kopii zapasowej programu oraz do przenoszenia programu pomiędzy sterownikami. W przypadku wykonania kopii zapasowej programu (*ang. program backup*), na kartę flash zostanie skopiowany cały program użytkownika, bloki funkcyjne, tabele symboli (zmiennych) globalnych i lokalnych oraz wszystkie komentarze umieszczone w programie. Należy przy tym pamiętać, że sterownik obsługuje tylko karty CF o pojemnościach 15, 30 i 60MB.

Prawa część panelu sterownika dotyczy ustawień modułu sieci Ethernet. W górnej części obudowy umieszczone są diody LED informujące o statusie połączenia sieciowego. Poniżej znajdują się nastawniki numeru jednostki CPU (Unit Number) oraz nastawniki numeru węzła sieci (Node Number). Wszystkie moduły komunikacyjne pracujące w jednej sieci powinny posiadać swój własny, unikatowy numer węzła. W przypadku modułów ethernetowych zaleca się, żeby numer węzła był zgodny z ostatnim oktetem adresu IP (np. dla modułu o adresie IP: 192.168.115.5 należy wprowadzić numer węzła 5).

4.2.3. MAPA PAMIĘCI STEROWNIKA CJ1M

Pamięć danych sterownika CJ1M posiada strukturę 16-bitową, w związku z tym adres komórki pamięci określa numer słowa, a nie bajtu, jak to ma miejsce w innych sterownikach (np. rodziny SIMATIC S7). Pamięć ta została podzielona na kilka odrębnych obszarów, w których przechowane są dane pełniące określone funkcje. Dostęp do poszczególnych komórek pamięci uzyskuje się poprzez podanie prefiksu obszaru pamięci (np. W, D, T...) oraz określenie adresu słowa i ewentualnie numeru bitu (np. W100.15). Wyjątkiem od tej reguły jest dostęp do obszaru wejść/wyjść sterownika (ang. *CIO – Common Input-Output area*), gdzie podaje się jedynie adres, bez prefiksu (np. 500.05).

W adresowaniu obszarów pamięci, które pozwalają na bezpośredni dostęp do bitów stosuje się dwie metody notacji [4]:

- **z kropką** – w której podaje się adres słowa, a po kropce (separatorze) umieszcza sie numer bitu (np. W15.01 lub W15.1 ; 100.15);
- **bez kropki** – gdzie nie podaje się separatora bitowego, a dwie ostatnie cyfry określają numer bitu (np. W10010 odpowiada zapisowi W100.10). W takim przypadku podanie adresu w postaci „10” będzie odnosiło się do 10-tego bitu komórki z obszaru CIO o adresie 0 (czyli 0.10).

Jeżeli adresowanie dotyczy całego słowa (16 bitów), wówczas adres składa się tylko z sekcji określającej numer komórki (np. 10, W30, D1000).

W tabeli 4.3 przedstawiono podział pamięci sterownika serii CJ1 na poszczególne obszary funkcyjne.

W obszarze ***CIO*** przechowywane są dane określające stany wejść i wyjść cyfrowych sterownika, dane modułów komunikacyjnych, modułów specjalnych oraz bity robocze ogólnego przeznaczenia. Wewnątrz pamięci *CIO* wydzielono kilka specyficznych obszarów, pełniących określone funkcje, dla których na stałe przypisano odpowiednie zakresy adresowe.

Do przechowywania danych użytkownika sterujących przebiegiem programu (w szczególności bitów roboczych – tzw. markerów) służy obszar ***W*** (ang. *Working area*) o pojemności 512 słów. Dostęp do tego obszaru może odbywać się zarówno w formie

adresowania bitów, jak i całych słów. Zawartość pamięci w tym obszarze nie jest podtrzymywana po zaniku napięcia zasilającego, ani po zresetowaniu sterownika. W przypadku, gdy wielkość pamięci typu *W* okaże się niewystarczająca, użytkownik może umieszczać dane (bity robocze) w wolnym obszarze *CIO*.

Obszar danych ***D*** (ang. *Data area*) posiada pojemność 32 kilosłów i może służyć do przechowywania ogólnych danych liczbowych użytkownika. W tym obszarze nie ma możliwości zaadresowania konkretnego bitu, możliwy jest tylko dostęp do całych komórek 16-bitowych. Zawartość całej pamięci obszaru *D* jest podtrzymywana po zaniku zasilania i restarcie sterownika. W obszarze *D* znajdują się wydzielone komórki, które służą do wymiany danych pomiędzy sterownikiem a specjalnymi modułami rozszerzeń (D20000-D29599) oraz komórki, które przechowują dane transferowane przez moduły komunikacyjne (D30000-D31599).

Obszar pamięci ***H*** (ang. *Holding area*) służy do przechowywania różnych danych w postaci słów albo bitów. Dane umieszczone w tym obszarze są podtrzymywane po zaniku zasilania sterownika. Oprócz danych ogólnego przeznaczenia, w obszarze H umieszczane są dane generowane przez program, w przypadku użycia bloków funkcyjnych oraz innych instrukcji wymagających zapamiętania stanów.

Obszar pomocniczy ***A*** (ang. *Auxiliary area*) zawiera flagi i bity sterujące, które mogą służyć do monitorowania i sterowania pracą sterownika PLC. Ogólnie obszar ten dzieli się na dwie części: komórki tylko do odczytu (A000-A447) oraz komórki do zapisu i odczytu (A448-A959). W obszarze pomocniczym przechowywane są miedzy innymi: kody błędów sterownika, flagi systemowe, bieżący czas i data, czas działania sterownika, znaczniki czasowe ostatnich wyłączeń i zaników zasilania sterownika, dane i parametry związane z obsługą szybkich liczników i wiele innych pożytecznych informacji.

Obszar czasomierzy ***T*** (ang. *Timer area*) służy do przechowywania bieżących wartości PV (ang. *Preset Value*) oraz flag wypełnienia (ang. *Completion Flag*) poszczególnych czasomierzy, o numerach z zakresu 0–4095. W zależności od typu zastosowanej instrukcji czasomierza (TIM, TIMX), wartości PV mogą posiadać format binarny albo BCD. Z kolei flagi wypełnienia czasomierzy mają format bitowy i w programie występują w postaci styków normalnie otwartych (NO) lub zamkniętych (NC). Po włączeniu zasilania sterownika wszystkie wartości PV i znaczniki wypełnienia zostają wyzerowane.

W Obszarze liczników ***C*** (ang. *Counter area*) przechowywane są wartości bieżące PV oraz znaczniki wypełnienia (flagi) poszczególnych liczników, o numerach z zakresu 0–4095. Podobnie jak w przypadku czasomierzy, format wartości bieżących liczników PV zależy od typu użytych instrukcji licznikowych i może przyjąć postać binarną albo BCD. Po wyłączeniu zasilania i powtórnym uruchomieniu sterownika, wartości PV i znaczniki wypełnienia dla wszystkich liczników zostają zachowane [4].

Tab.4.3. Obszary pamięci sterownika serii CJ1.

Obszar		Zakres	Opis
Obszar Wejść – Wyjść (CIO)	Obszar We/Wy	CIO (0000 – 0159)	Obszar przydzielany automatycznie do podstawowych modułów We/Wy. Ustawienia domyślne dla pierwszej grupy modułów (0000) można zmieniać w zakresie (0000 – 0999)
	Obszar wbudowanych We/Wy	CIO (2960 – 2961)	Dla wbudowanych wejść w sterownikach CJ1M-CPU22/23
	Obszar wspólny	CIO (1000 – 1199)	Bity wspólne są używane jako dane wspólne przydzielane do modułów Controller Link
	Obszar modułów komunikacyjnych	CIO (1500 – 1899)	W obszarze tym przechowywany jest stan pracy modułów komunikacyjnych (25 słów na moduł, maksymalnie 16 modułów)
	Obszar specjalnych modułów We/Wy	CIO (2000 – 2959)	Obszar przydzielany do specjalnych modułów rozszerzeń, np. analogowych We/Wy (10 słów na moduł, maksymalnie 96 modułów)
	Obszar połączenia szeregowego sterowników	CIO (3100 – 3189)	Do przekazywania danych przy połączeniach szeregowych sterowników PLC – tylko dla CJ1M.
	Obszar sieci DeviceNet	CIO (3200 – 3799)	Obszar przydzielany dla urządzeń sieci DeviceNet realizujących komunikację ze zdalnymi stacjami We/Wy dla stałych alokacji w module typu master.
	Obszar We/Wy wewnętrznych (bity robocze)	CIO (1200 – 1499) (3800 – 6143)	Bity używane jako standardowe bity robocze. Nie można ich wykorzystywać do adresowania We/Wy zewnętrznych
Obszar roboczy		W (000 – 511)	Obszar przeznaczony na dane użytkownika. Komórki pamięci w tym obszarze mogą być adresowane bitowo (np. W100.15).
Obszar podtrzymywania		H (000 – 511)	Dane umieszczone w tym obszarze zostają zachowane po wyłączeniu zasilania oraz po zmianie trybu działania sterownika.
Obszar pomocniczy		A (000 – 959)	Przechowywane są tu bity sterujące, flagi systemowe, znaczniki błędów oraz bity pomocnicze, pełniące funkcje specjalne.
Obszar tymczasowy		TR (00 – 15)	16 bitów tymczasowych służy do przechowywania informacji o punktach rozgałęzienia programów.
Obszar czasomierzy		T (0000 – 4095)	Przechowuje wartości bieżące czasomierzy oraz ich znaczniki wypełnienia.
Obszar liczników		C (0000 – 4095)	Przechowuje wartości bieżące liczników oraz ich znaczniki wypełnienia.

Obszar danych	D (00000 – 32767)	Obszar przeznaczony dla danych użytkownika. Odczyt i zapis danych z tego obszaru możliwy jest tylko dla całych słów. Nie ma możliwości bezpośredniego dostępu do pojedynczych bitów.
Rejestry indeksowe	IR (00 – 15)	W obszarze tym przechowywane są adresy komórek pamięci, przy adresowaniu pośrednim – indeksowym. Każdy rejestr indeksowy ma pojemność 32-bitów (dwóch słów).
Znaczniki zadań	TK (00 – 31)	Znaczniki zadań są to bity, które przyjmują wartość logiczną 1, wtedy, gdy odpowiadające im zadanie cykliczne jest wykonywalne.

4.2.4. ALOKACJA MODUŁÓW ROZSZERZEŃ

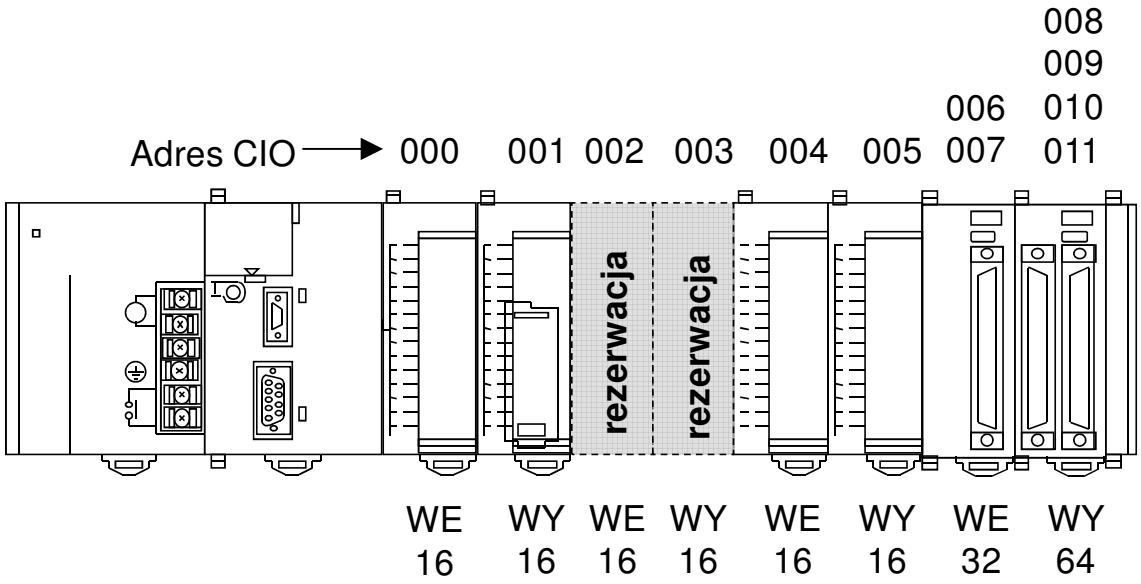
Sterowniki programowalne serii CJ umożliwiają wykonanie automatycznej alokacji adresów wszystkich dołączonych podstawowych modułów I/O, zaraz po włączeniu zasilania sterownika. Proces ten polega na zarezerwowaniu dla każdego modułu odpowiedniej przestrzeni pamięci oraz przypisanie mu unikalnego adresu w obszarze CIO.

Dla modułów podstawowych (*Basic I/O Units*) adresy przypisywane są po kolei, rozpoczynając od modułu znajdującego się najbliżej jednostki centralnej CPU. W konfiguracji domyślnej alokacja rozpoczyna się od adresu CIO 0000, jednak użytkownik za pomocą oprogramowania *CX-Programmer* może początkowy adres ustawić w zakresie CIO 0000–0999. Dla każdego modułu alokowane jest tyle słów pamięci, ile jest wymagane. Przykładowo, dla modułów obsługujących 16 wejść cyfrowych będzie przydzielone tylko jedno słowo pamięci (16 bitów), zaś moduły 64-wejściowe otrzymają 4 słowa (4x16 bitów).

Jeżeli przewidywane są w przyszłości zmiany w konfiguracji sprzętowej sterownika, np. planowane jest zainstalowanie dodatkowych modułów I/O, wówczas użytkownik może zawczasu zarezerwować pewien obszar pamięci dla tych modułów, tak aby później nie trzeba było zmieniać adresów w programie sterującym. Na rysunku 4.4 przedstawiono przykładową alokację adresową dla zestawu podstawowych modułów I/O. W tej konfiguracji, zarezerwowano 2 słowa pamięci (002 i 003) przeznaczone dla przyszłych modułów, które będą umieszczone w 3 i 4 slocie.

Alokacja specjalnych modułów wejść/wyjść (ang. *SIOU – Specjal Input-Output Unit*) polega na przypisaniu dla każdego modułu 10 kolejnych słów pamięci w obszarze CIO 2000–2959. Adres początkowy zależy od numeru modułu (ang. *Unit Number*), który jest ustawiany za pomocą odpowiednich nastawników znajdujących się na obudowie modułu. Każdy moduł SIOU musi posiadać swój niepowtarzalny

numer w zakresie 0–95. W tabeli 4.4 przedstawiono zakresy adresowe modułów specjalnych, w zależności od ich numerów [11].



Rys.4.4. Przykładowa alokacja pamięci modułów podstawowych I/O.

Tab.4.4. Alokacja pamięci specjalnych modułów wejść/wyjść.

Numer modułu	**Alokacja pamięci**
0	CIO 2000 ÷ 2009
1	CIO 2010 ÷ 2019
2	CIO 2020 ÷ 2029
50	CIO 2050 ÷ 2059
N	CIO (2000+*N**10) ÷ (2009+*N**10)
95	CIO 2950 ÷ 2959

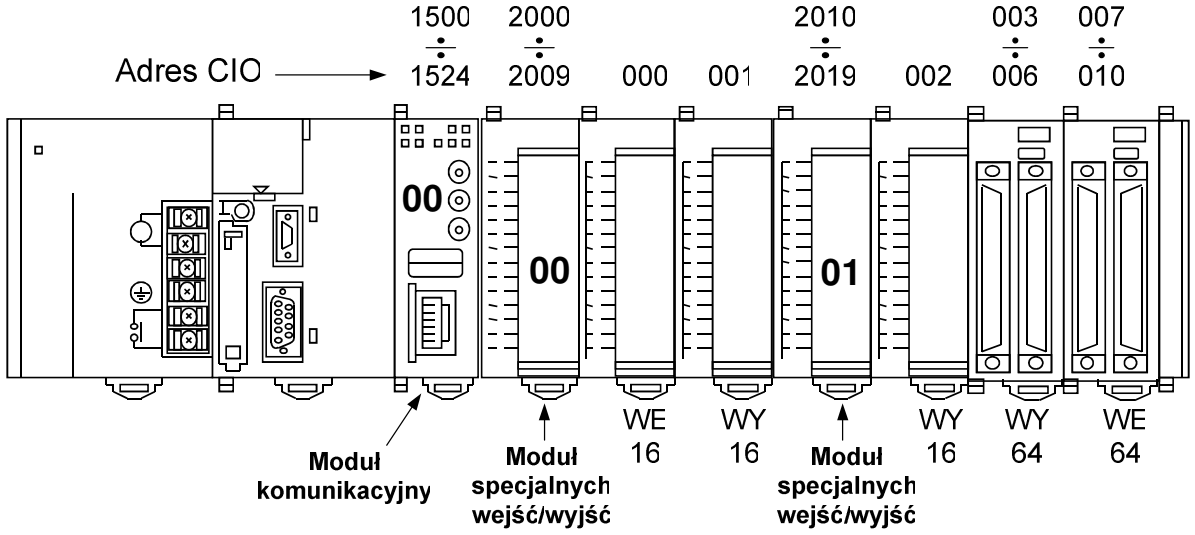
Dla każdego modułu komunikacyjnego zostaje alokowanych 25 słów pamięci w obszarze CIO 1500–1899. Podobnie jak w przypadku modułów SIOU, adres początkowy modułu komunikacyjnego zależy od jego numeru, ustawionego za pomocą nastawników obrotowych znajdujących sie na jego obudowie. Każdy moduł komunikacyjny musi posiadać unikalny numer w zakresie 0–15. W tabeli 4.5 przedstawiono zasady alokacji pamięci dla modułów komunikacyjnych.

Tab.4.5. Alokacja pamięci modułów komunikacyjnych.

Numer modułu	Alokacja pamięci
0	CIO 1500 ÷ 1524
1	CIO 1525 ÷ 1549
2	CIO 1550 ÷ 1574
N	CIO (1500+*N**25) ÷ (1524+*N**25)
15	CIO 1875 ÷ 1899

Zasada unikalności modułów dotyczy tylko urządzeń jednego typu. To znaczy, że w jednej konfiguracji sprzętowej może znajdować się moduł z grupy SIOU o tym samym numerze, co moduł komunikacyjny czy też moduł z grupy podstawowych wejść/wyjść.

Na rysunku 4.5 przedstawiono przykładową alokację pamięci modułów w konfiguracji mieszanej, w której jednocześnie występują moduły podstawowe, specjalne i komunikacyjne.



Rys.4.5. Przykładowa alokacja pamięci modułów mieszanych.

4.3. Wybrane moduły rozszerzeń sterowników serii CJ

4.3.1. MODUŁ WEJŚĆ CYFROWYCH ID211

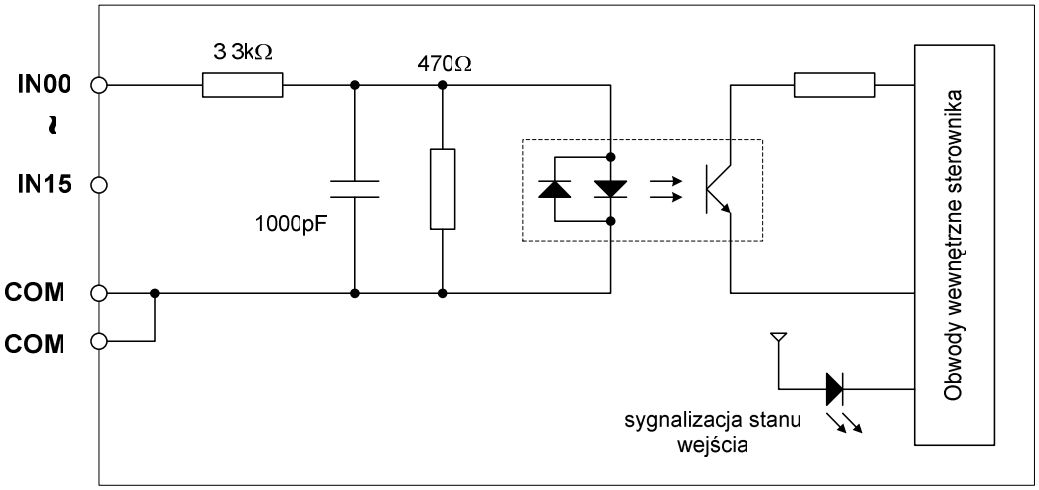
Moduł CJ1W–ID211 jest 16-kanałowym modułem wejść dyskretnych o napięciu znamionowym $24V_{DC}$. W przestrzeni adresowej pamięci CIO zajmuje jedno słowo, w którym kolejne bity odpowiadają wartościom logicznym występującym na poszczególnych wejściach. Przy kodowaniu sygnałów wejściowych przyjęto logikę dodatnią, co oznacza, że obecność napięcia +24V na wejściu fizycznym modułu będzie powodować ustawienie odpowiadającego mu bitu na wartość logiczną „1". W tabeli 4.7 zestawiono podstawowe parametry modułu ID211 [11].

Tab.4.7. Podstawowe parametry modułu ID211.

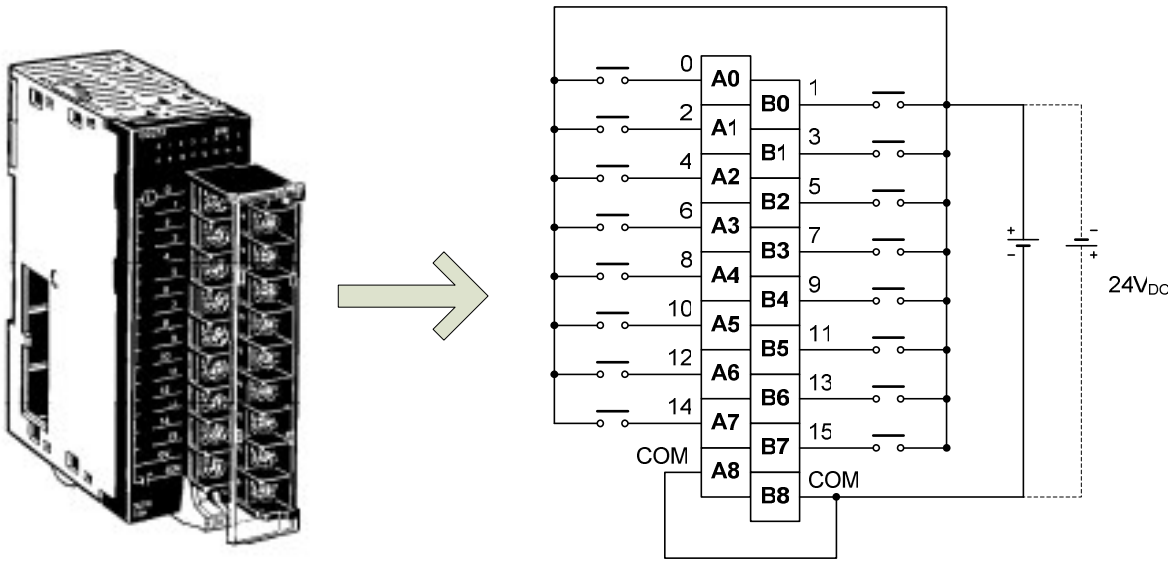
Napięcie znamionowe	24 V DC
Zakres napięcia wejściowego	20,4–26,4 V DC
Impedancja wejściowa	3,3 kΩ
Prąd wejścia	7mA (przy 24V)
Napięcie załączenia (ON)	14,4 V DC min./3mA min.
Napięcie wyłączenia (OFF)	5 V DC max./1mA max.
Czas załączenia (ON)	8ms max. (możliwość ustawienia od 0 do 32ms w PLC Setup)
Czas wyłączenia (OFF)	8ms max. (możliwość ustawienia od 0 do 32ms w PLC Setup)
Rezystancja izolacji	20 MΩ (pomiędzy zaciskami zewnętrznymi a zaciskiem GR)
izolacja napięciowa	1000 V AC
Pobór prądu modułu	5 V DC: 80mA max.

Na rysunku 4.6 przedstawiono schemat ideowy obwodu wejściowego dla pojedynczego wejścia modułu ID211. Wejściowe elementy pasywne R i C tworzą dzielnik napięciowy oraz prosty układ filtru dolnoprzepustowego, którego celem jest tłumienie zakłóceń wejściowych o charakterze impulsowym. Sygnał napięcia wejściowego zostaje doprowadzony do diody LED wewnętrznego transoptora, którego zadaniem jest przekazanie sygnału sterującego do wewnętrznych obwodów sterownika PLC przy jednoczesnym zachowaniu izolacji galwanicznej. Dla takiej konfiguracji obwodów wejściowych, wejściowy prąd sterujący osiąga wartość ok. 7mA przy napięciu 24V.

W module ID211 zastosowano transoptory bipolarne z wejściem AC, które mogą pracować przy dowolnej polaryzacji napięcia sterującego, w tym również przy napięciach przemiennych. Dzięki temu użytkownik może dowolnie skonfigurować połączenia zewnętrznych sygnałów sterujących, tak aby podawać na wejścia modułu poziomy wysokie lub niskie napięcia (układ z logiką dodatnią lub ujemną). Sposób dołączenia sygnałów wejściowych do zacisków modułu przedstawiono na rysunku 4.7 [11].



Rys.4.6. Schemat ideowy obwodów wejściowych modułu ID211.



Rys.4.7. Sposób dołączenia sygnałów sterujących do modułu ID211.

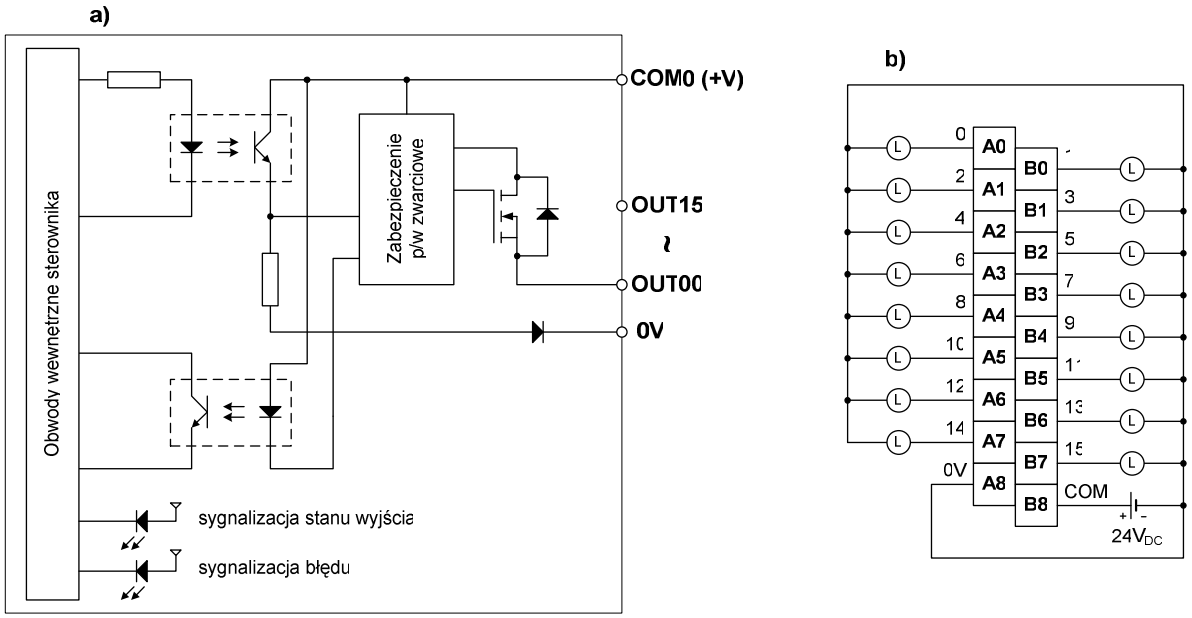
4.3.2. MODUŁ WYJŚĆ CYFROWYCH OD212

Moduł CJ1W–OD212 jest 16-kanałowym modułem półprzewodnikowych wyjść dyskretnych o napięciu znamionowym $24V_{DC}$. Podobnie jak w module wejściowym, stan poszczególnych wyjść cyfrowych jest reprezentowany w przestrzeni adresowej pamięci CIO przez jedno słowo (16 bitów). W module zastosowano wyjścia tranzystorowe o logice dodatniej, co oznacza, że w stanie aktywnym (ON) na zaciskach wyjściowych pojawia się napięcie +24V DC. W tabeli 4.8 przedstawiono podstawowe parametry modułu OD212 [11].

Tab.4.8. Podstawowe parametry modułu OD212.

Napięcie znamionowe	24 V DC
Zakres zmian napięcia wyjściowego	20,4–26,4 V DC
Maksymalny prąd obciążenia	0,5 A / kanał, 5 A / moduł
Zabezpieczenie przeciążeniowe	Sygnalizacja przeciążenia w zakresie prądów 0,7–2,5A
Czas załączenia (ON)	0,5 ms max.
Czas wyłączenia (OFF)	1,0 ms max.
Rezystancja izolacji	20 MΩ (pomiędzy zaciskami zewnętrznymi a zaciskiem GR)
izolacja napięciowa	1000 V AC
Zewnętrzne napięcie zasilające	20,4–26,4 V DC, 40 mA min.
Pobór prądu modułu	5 V DC: 100 mA max.

Na rysunku 4.8a przedstawiono schemat ideowy wewnętrznego obwodu sterowania wyjścia cyfrowego.



Rys.4.8. Schemat obwodów wyjściowych modułu OD212 (a) oraz schemat podłączenia odbiorników zewnętrznych do zacisków wyjściowych (b).

W roli wyjściowych kluczy półprzewodnikowych zostały zastosowane tranzystory MOSFET wyposażone w dodatkowy układ zabezpieczenia nadprądowego, który wykrywa nadmierny wzrost prądu obciążenia i sygnalizuje go za pomocą czerwonej diody LED umieszczonej na panelu modułu. Dodatkowo, stan przeciążenia powoduje

uaktywnienie flagi błędu w obszarze pamięci pomocniczej, w odpowiednich komórkach z zakresu A050–A069. Rysunek 4.8b przedstawia sposób podłączenia odbiorników i zewnętrznego źródła zasilania do zacisków modułu OD212. Należy zwrócić uwagę na właściwą polaryzację źródła zasilania, gdyż odwrotne podłączenie może spowodować załączenie odbiorników wskutek przepływu prądu przez diody zwrotne wbudowane w wyjściowe tranzystory MOSFET.

4.3.3. MODUŁ WEJŚĆ ANALOGOWYCH AD041

Moduł specjalny AD041 posiada 4 wejścia analogowe, które mogą być skonfigurowane jako wejścia napięciowe lub jako wejścia prądowe. Oprócz standardowych funkcji pomiarowych, moduł analogowy oferuje dodatkowe funkcje:

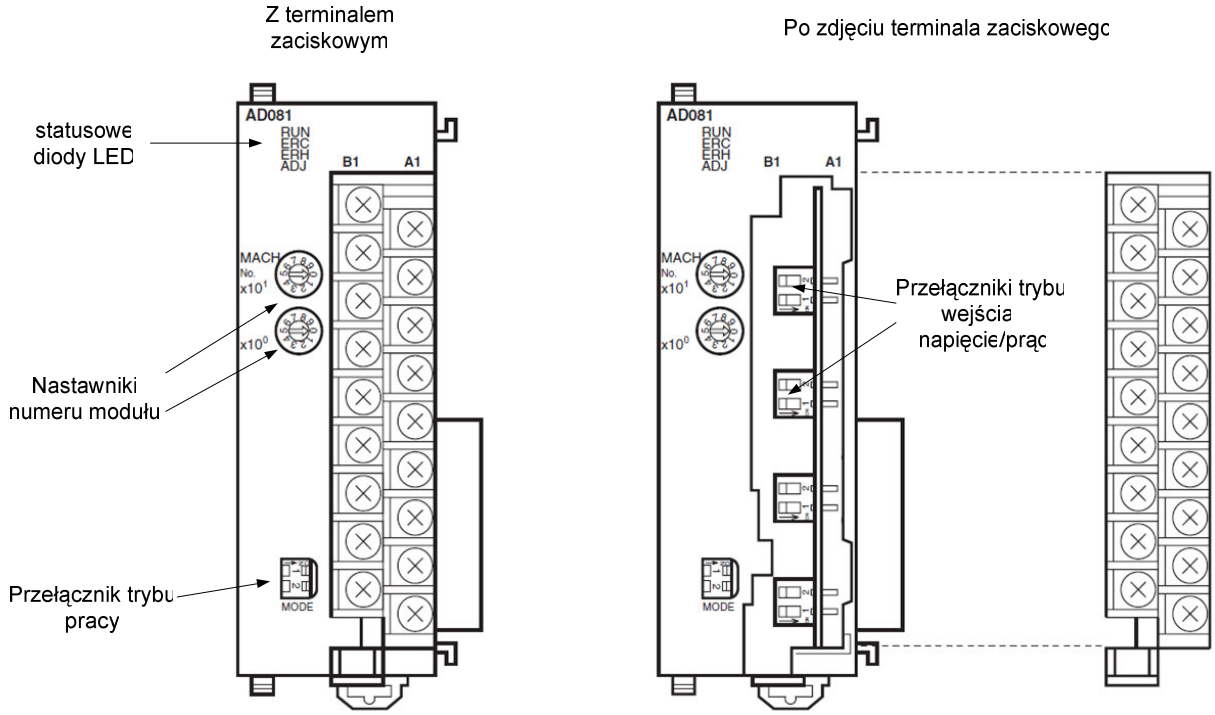
- możliwość wykrycia stanu rozłączenia obwodu pomiarowego (tylko przy konfiguracji zakresów wejściowych: 1–5V lub 4–20mA),
- obliczanie wartości średniej sygnału wejściowego, na podstawie konfigurowalnej liczby buforowanych próbek (2, 4, 8, 16, 32 i 64),
- detekcja wartości szczytowych mierzonych sygnałów,
- możliwość kalibracji (regulacja wzmocnienia i składowej stałej dla wzmacniaczy wejściowych).

W tabeli 4.9 przedstawiono podstawowe parametry modułu AD041 [16].

Tab.4.9. Wybrane parametry modułu wejść analogowych AD041.

Liczba wejść analogowych		4
Zakresy sygnałów wejściowych		1–5V 0–5V 0–10V -10–10V 4–20mA
Maksymalne wartości napięć i prądów wejściowych		Wejście napięciowe: ±15V Wejście prądowe: ±30mA
Impedancja wejściowa		Wejście napięciowe: 1MΩ min. Wejście prądowe: 250Ω
Rozdzielczość		4.000/8.000
Format danych wyjściowych		16-bitowy binarny
Dokładność przetwarzania	23±2ºC	Wejście napięciowe: ±0,2% zakresu Wejście prądowe: ±0,4% zakresu
	0–50ºC	Wejście napięciowe: ±0,4% zakresu Wejście prądowe: ±0,6% zakresu
Czas konwersji		1ms/250μs

Na rysunku 4.9 przedstawiono rozmieszczenie elementów funkcyjnych na panelu modułu AD041. Przełączniki trybu pracy wejścia (napięcie/prąd) znajdują się pod terminalem zaciskowym. W górnej części panelu znajdują się diody LED sygnalizujące stan pracy modułu. W tabeli 4.10 przedstawiono znaczenie poszczególnych sygnałów diagnostycznych diod LED [16].

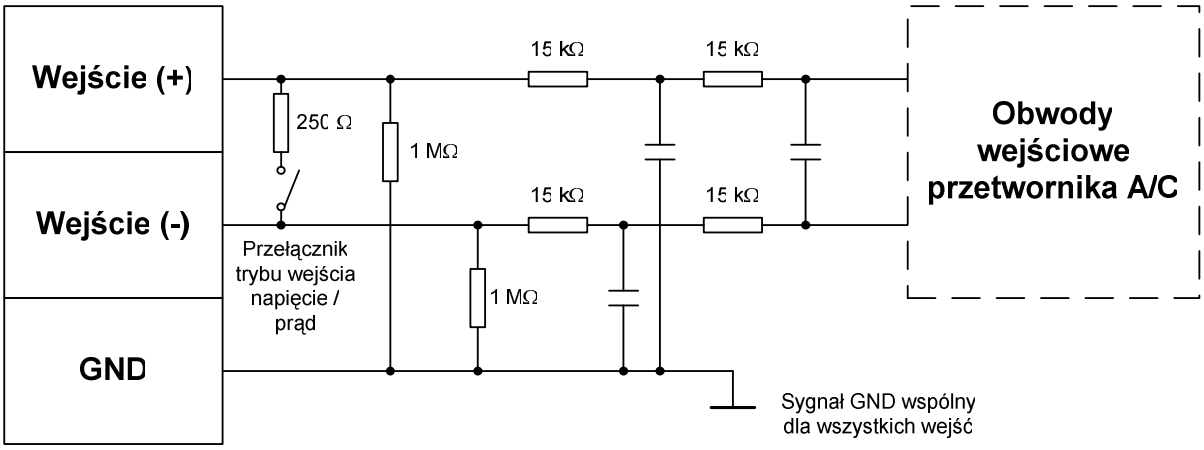


Rys.4.9. Rozmieszczenie elementów funkcyjnych na przednim panelu modułu AD041/AD081.

Tab.4.10. Znaczenie sygnałów diagnostycznych wyświetlanych przez diody LED.

LED	funkcja	Stan	Opis
RUN (zielona)	Tryb pracy	świeci	Działanie normalne
		nie świeci	Moduł został zatrzymany podczas wymiany danych z CPU
ERC (czerwona)	Błąd wykryty przez moduł	świeci	Pojawił się stan alarmowy lub wprowadzono nieprawidłowe ustawienia początkowe
		nie świeci	Działanie normalne
ERH (czerwona)	Błąd w module CPU	świeci	Wystąpił błąd podczas wymiany danych z CPU
		nie świeci	Działanie normalne
ADJ (żółta)	Tryb ustawień	świeci	Działa w trybie kalibracji wzmocnienia lub offsetu
		nie świeci	żaden z powyższych

Rysunek 4.10 przedstawia schemat ideowy obwodu wejściowego dla jednego kanału pomiarowego. Na wejściu znajduje się przełącznik rodzaju pracy, który przy wyborze trybu pomiaru prądu dołącza obwód z rezystorem wejściowym o wartości 250Ω. W dalszej części obwodu znajdują się filtry RC, które redukują zakłócenia wysokoczęstotliwościowego. Poszczególne wejścia analogowe znajdują się na tym samym potencjale – posiadają wspólny punkt masy sygnałowej GND. Wewnętrzne obwody wejściowe modułu są odseparowane galwanicznie od magistrali sterownika PLC za pomocą transoptorów [16].



Rys.4.10. Schemat ideowy obwodu wejściowego dla jednego kanału modułu AD041.

Wymiana danych pomiędzy modułem analogowym a sterownikiem odbywa się poprzez komórki pamięci w obszarze CIO (10 słów na każdy moduł) w zakresie adresowym 2000–2959. Oprócz tego, ustawienia i parametry modułu przechowywane są w obszarze pamięci D zakresie adresowym 20000–29599 (100 słów dla każdego modułu).

4.3.4. MODUŁ WYJŚĆ ANALOGOWYCH DA041

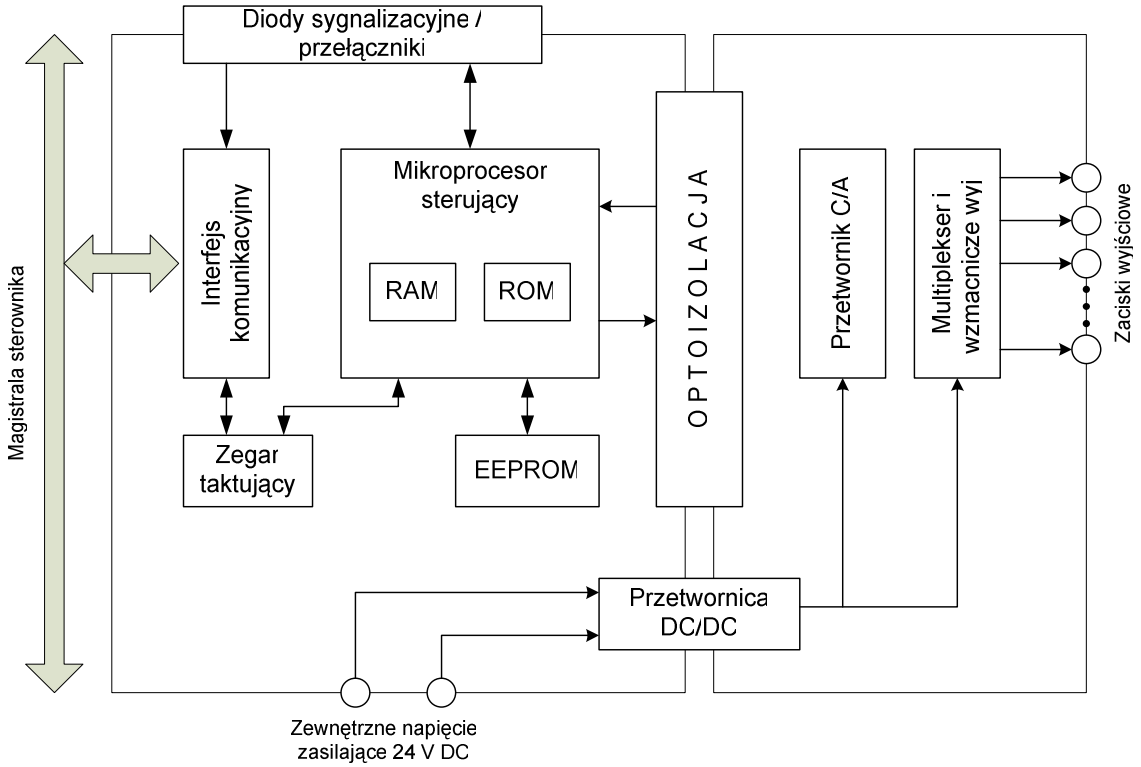
Moduł wyjść analogowych CJ1W-DA041 służy do wyprowadzenia ze sterownika sygnałów analogowych, w postaci napięcia lub prądu. Dla każdego z czterech wyjść analogowych można wprowadzić niezależne ustawienia parametrów sygnałów wyjściowych, wybierając tryb pracy (napięcie/prąd) oraz dobierając zakres przetwarzania w zależności od wymagań aplikacji użytkownika. W tabeli 4.11 przedstawiono podstawowe parametry modułu DA041 [16].

Rozmieszczenie elementów funkcyjnych na panelu modułu DA041 oraz znaczenie sygnałów diagnostycznych diod LED jest identyczne jak dla modułu wejść analogowych AD041.

Tab.4.11. Wybrane parametry modułu wejść analogowych DA041.

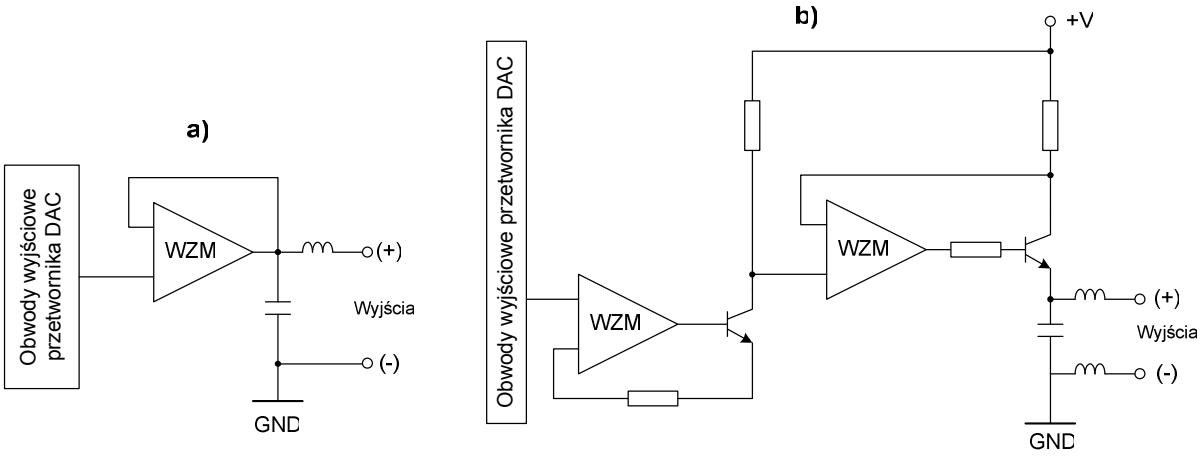
Liczba wyjść analogowych		4
Zakresy sygnałów wyjściowych		1–5V 0–5V 0–10V -10–10V 4–20mA
Impedancja wyjściowa		0,5 Ω max. (dla wyjścia napięciowego)
Maksymalny prąd wyjścia		12 mA (dla wyjścia napięciowego)
Maksymalna rezystancja obciążenia		600 Ω max. (dla wyjścia prądowego)
Rozdzielczość		4.000
Format danych wejściowych		16-bitowy binarny
Dokładność przetwarzania	23±2ºC	Wyjście napięciowe: ±0,3% zakresu Wejście prądowe: ±0,5% zakresu
	0–50ºC	Wejście napięciowe: ±0,5% zakresu Wejście prądowe: ±0,8% zakresu
Czas konwersji		1ms max. / wyjście

Na rysunku 4.11 przedstawiono schemat blokowy wewnętrznych obwodów modułu DA041 [16].



Rys.4.11. Schemat blokowy modułu wyjść analogowych DA041.

Głównym elementem jest mikroprocesor sterujący, który zarządza konfiguracją sprzętową i parametryczną modułu, wymienia dane z jednostką centralną sterownika oraz nadzoruje poprawność jego pracy. Wyjściowa sekcja analogowa modułu jest odseparowana galwanicznie od obwodów wewnętrznych sterownika PLC, dzięki czemu możliwe jest dołączenie do obwodów zewnętrznych, znajdujących się na innym potencjale niż sterownik. Obwody wyjściowe modułu DA041 zawierają multiplekser analogowy oraz cztery niezależne sekcje wzmacniaczy wyjściowych, po jednej dla każdego kanału wyjściowego. Wzmacniacze wyjściowe pracują w jednej z dwóch konfiguracji, przełączanych za pomocą przełączników trybu napięcie/prąd. Na rysunku 4.12 przedstawiono schematy ideowe obwodów wyjściowych wzmacniaczy, dla trybu napięciowego (a) i prądowego (b).



Rys.4.12. Schematy ideowe obwodów wyjściowych modułu DA041
w konfiguracji wyjścia napięciowego (a) oraz prądowego (b).

5. Oprogramowanie narzędziowe dla sterowników OMRON

5.1. Pakiet programowy CX-One

Wszystkie obecnie produkowane sterowniki programowalne firmy OMRON oraz inne produkty z grupy automatyki przemysłowej mogą być programowane, konfigurowane i uruchamiane za pomocą jednego, wspólnego pakietu programowego o nazwie CX-ONE. Rozwiązanie to znacznie upraszcza procedury projektowe, gdyż użytkownik korzysta tylko z jednego oprogramowania, z jedną instalacją i pojedynczym numerem licencji [13, 17].

Pakiet CX-One zapewnia obsługę następujących urządzeń:

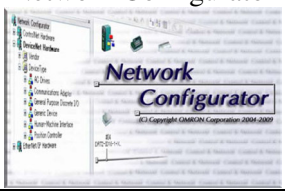
- wszystkie sterowniki PLC serii CP, CJ i CS;
- wszystkie terminale operatorskie serii NS;
- sterowniki Trajexia;
- serwonapędy i falowniki OMRON Yaskawa;
- systemy i urządzenia komunikacyjne sieci Profibus, ProfiNet i CompoNet;
- regulatory temperatury;
- czujniki.

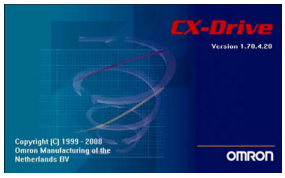
Programy narzędziowe pakietu CX-One zostały pogrupowane w zależności od pełnionej funkcji:

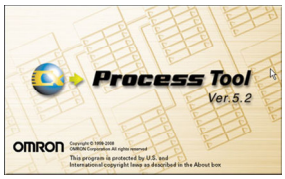
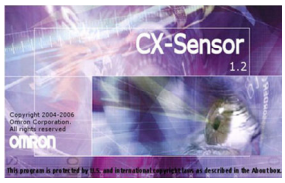
- narzędzia do programowania;
- narzędzia obsługi i konfiguracji sieci;
- programy do obsługi napędów;
- narzędzi do regulacji i przełączania;
- programy obsługi czujników.

W tabeli 5.1 przedstawiono podział programów narzędziowych należących do pakietu CX-One [17].

Tab.5.1. Programy narzędziowe pakietu CX-One.

Program	Opis
Programowanie	
CX Programmer 	CX-Programmer zapewnia wspólną platformę oprogramowania sterowników PLC dla wszystkich urządzeń tego typu produkowanych przez firmę Omron. Umożliwia łatwą konwersję i ponowne wykorzystanie kodu w sterownikach PLC różnych typów, a także korzystanie (w pełnym zakresie) z programów kontrolnych utworzonych za pomocą aplikacji do programowania sterowników PLC starszych generacji.
CX Simulator 	CX-Simulator to zintegrowane środowisko debugowania odpowiadające rzeczywistemu sterownikowi PLC, które umożliwia ocenę działania programu, sprawdzenie długości cyklu i skrócenie czasu debugowania przed oddaniem urządzenia do użytku.
CX Designer 	Program CX-Designer służy do tworzenia danych ekranowych dla potrzeb programowalnych terminali z serii NS. Za jego pomocą można również sprawdzać na komputerze działanie utworzonych ekranów. CX-Designer zawiera około tysiąca standardowych obiektów funkcyjnych wraz z grafikami i zaawansowanymi funkcjami.
Obsługa sieci	
CX-Integrator 	CX-Integrator to główny składnik konfiguracyjny pakietu CX-One. Umożliwia on łatwe wykonywanie wielu operacji, takich jak monitorowanie stanu połączeń różnych sieci, ustawianie parametrów i diagnozowanie sieci.
CX-ConfiguratorFDT 	Oparty na technologii FDT/DTM program CX-ConfiguratorFDT służy do konfiguracji urządzeń dowolnego producenta podłączonych do sieci PROFIBUS. Koncepcja ta zostanie w przyszłości rozwinięta w celu obsługi innych sieci z użyciem technologii FDT/DTM.
CX-Protocol 	Służy do tworzenia procedur przesyłania danych (makro protokołów) w celu wymiany danych między standardowymi urządzeniami szeregowymi a sterownikami PLC za pośrednictwem modułu lub karty do obsługi transmisji szeregowej.
Network Configurator 	Network Configurator for EtherNet/IP umożliwia konfigurację jednostek EtherNet/IP oraz łączy danych „sterownik-sterownik”.

Sterowniki i Napędy	
CX-Motion 	Program CX-Motion służy do tworzenia, edycji i drukowania rozmaitych parametrów, danych pozycji oraz programów sterowania ruchem (w języku G code) niezbędnych do działania sterowników ruchu, przesyłania danych do modułów sterowników ruchu oraz działania monitorów tychże modułów.
CX-Motion Pro 	Dzięki CX-Motion Pro programowanie sterowania ruchem jest proste, przejrzyste i szybkie.
CX-Drive 	Program CX-Drive umożliwia pełen dostęp do parametrów wszystkich produkowanych obecnie falowników i serwonapędów Omron Yaskawa na jednym z trzech wybranych przez operatora poziomów. Program pozwala również w prosty sposób przeglądać parametry, filtrować wartości różne od domyślnych, różne od wartości falownika oraz nieprawidłowe ustawienia. Prezentacje graficzne służą pomocą przy konfiguracji bardziej szczegółowych parametrów, takich jak częstotliwość przeskoków, profile U/f i parametry nastawników analogowych.
CX-Position 	Program CX-Position upraszcza wszystkie aspekty kontroli pozycjonowania, od tworzenia i edycji danych wykorzystywanych przez moduły pozycjonujące (moduły NC) po komunikację online i monitoring. Program zawiera funkcje poprawiające wydajność, takie jak automatyczne tworzenie danych projektu i korzystanie z istniejących danych.
CX-Motion NCF 	Ten zintegrowany program konfiguracyjny umożliwia szybkie tworzenie rozwiązań w zakresie ruchu z punktu do punktu dla systemów opartych na sieci MECHATROLINK-II.
CX-Motion MCH 	Umożliwia programowanie zaawansowanych aplikacji sterowania ruchem za pomocą modułów sterowników ruchu CS1W-MCH71 i CJ1W-MCH71, ustawianie parametrów systemowych, danych pozycji, programów zadań ruchu i danych procesu CAM.
Regulacja i przełączanie	
CX-Thermo 	CX-Thermo to program do konfiguracji i monitorowania regulatorów temperatury z serii E5CN i E5ZN. Umożliwia łatwą konfigurację, rejestrowanie danych online i monitorowanie w czasie rzeczywistym. Użytkownicy mogą w prosty sposób tworzyć i edytować parametry oraz pobierać je w plikach z komputera PC, co zmniejsza nakład pracy potrzebnej do ich ustawienia. Można monitorować dane nawet z 31 regulatorów temperatury jednocześnie.

CX-Process 	Program CX-Process upraszcza sterowanie pętli, od tworzenia i przenoszenia bloków funkcyjnych po uruchamianie funkcji kart i modułów oraz debugowanie (tuning parametrów PID itp.). Istnieje możliwość łatwego tworzenia programów z bloków funkcyjnych przez wklejanie bloków do okna i tworzenie połączeń programowych za pomocą myszy.
C z u j n i k i	
CX-Sensor 	CX-Sensor umożliwia konfigurację i monitorowanie czujników firmy Omron z serii ZX za pomocą łatwych w użyciu wskaźników. Graficzne okno dialogowe pozwala na wyświetlanie i porównywanie danych z kilkunastu czujników jednocześnie, dzięki czemu można konfigurować złożone procesy. Oprogramowanie obejmuje również sterownik umożliwiający dostęp do danych czujników za pośrednictwem modułów szeregowych SCU firmy Omron i innych aplikacji firmy, takich jak CX-Supervisor. W połączeniu z programem CX-Server OPC można nawet monitorować dane czujników w czasie rzeczywistym, korzystając w tym celu z programu Microsoft Excel.

5.2. CX-Programmer

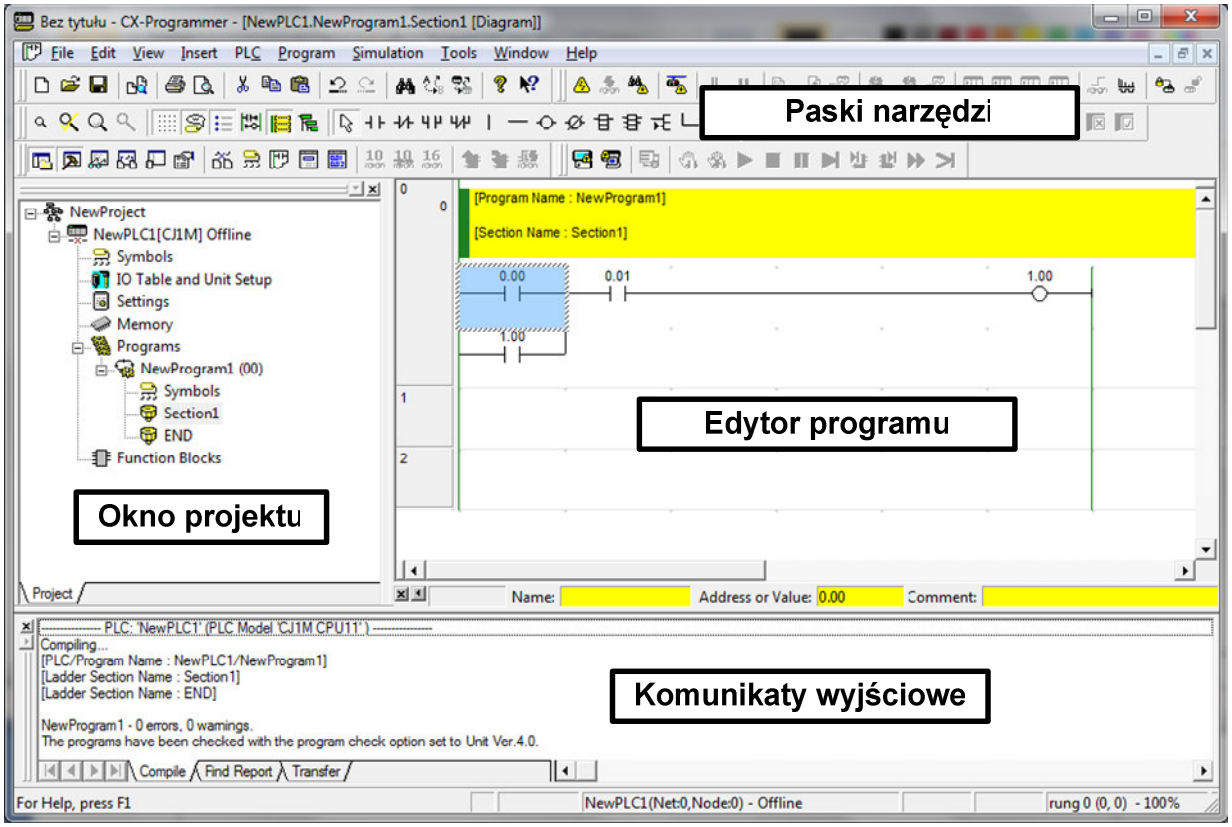
5.2.1. GŁÓWNE FUNKCJE PROGRAMU

CX-Programmer jest podstawowym oprogramowaniem narzędziowym do tworzenia, testowania i uruchamiania programów dla sterowników PLC OMRON z rodziny CP/CJ/CS. Edycja programu użytkownika możliwa jest w języku drabinkowym (LD), w języku ST oraz za pomocą grafu sekwencji SFC. CX-Programmer udostępnia bogatą bibliotekę skompilowanych bloków funkcyjnych, w których użytkownik znajdzie m.in.: gotowe funkcje do obsługi specjalnych modułów rozszerzeń, procedury komunikacyjne czy też zaawansowane funkcje do obsługi urządzeń peryferyjnych. Oprócz tego, oprogramowanie narzędziowe pozwala na tworzenie własnych bloków funkcyjnych, które mogą być umieszczone w bibliotece użytkownika, w celu ich wielokrotnego wykorzystania [14, 15].

Na rysunku 5.1 przedstawiono wygląd głównego okna środowiska CX-Programmer. Główne okno programu zawiera kilka charakterystycznych obszarów funkcyjnych, z których najważniejsze to:

- **okno projektu** – przedstawia drzewo plików projektu i umożliwia wyświetlenie i wprowadzenie ustawień sprzętowych i parametrów sterownika, podgląd i edycję tabeli symboli globalnych i lokalnych oraz otwieranie i przełączanie się pomiędzy programami (sekcjami) aplikacji użytkownika;

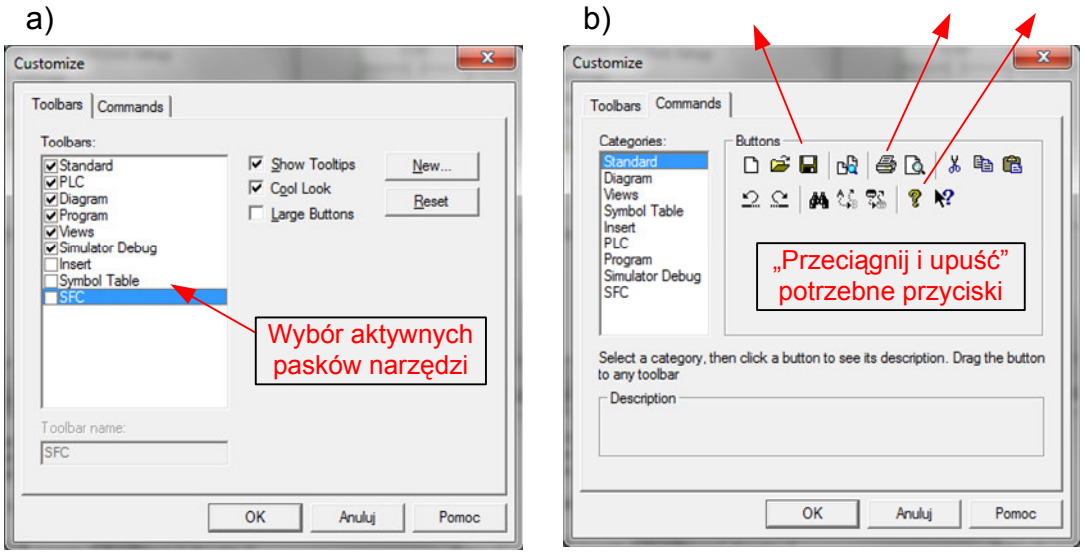
- **okno edycji programu** – przedstawia w postaci tekstowej lub graficznej program użytkownika i umożliwia jego edycję oraz analizę podczas fazy debugowania;
- **okno komunikatów wyjściowych** – wyświetla komunikaty i błędy pojawiające się podczas kompilacji ładowania programu do sterownika;
- **paski narzędzi** – zawierają przyciski, za pomocą których programista uzyskuje szybki dostęp do większości funkcji programu CX-Programmer. Zawartość palety przycisków może być dowolnie konfigurowana przez użytkownika.



Rys.5.1. Główne okno programu CX-Programmer.

Po zainstalowaniu pakietu CX-One, środowisko projektowe CX-Programmer uruchamia się z domyślnymi ustawieniami. Efektem tego jest pojawienie się okna jak na rysunku 5.1, w którym większość z dostępnych pasków narzędzi jest od razu aktywna, a oczom użytkownika ukazuje się paleta z zawartością blisko stu przycisków funkcyjnych. Przyciski te na poszczególnych paskach narzędzi zostały pogrupowane tematycznie, co pozwala dość łatwo odnaleźć żądaną funkcję spośród „gąszcza" dostępnych opcji. Jednak konfiguracja taka nie każdemu musi odpowiadać. Po pierwsze nadmiar eksponowanych przycisków pogarsza czytelność menu oraz zajmuje sporo miejsca na ekranie, zwłaszcza na małych monitorach o ograniczonej rozdzielczości. Po drugie, większość z dostępnych funkcji jest uruchamiana

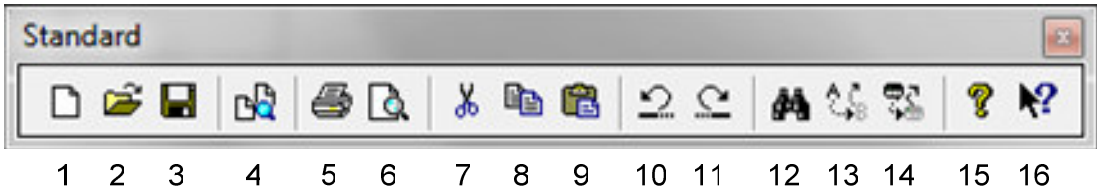
sporadycznie, a jeśli już zajdzie potrzeba ich uruchomienia, to użytkownik i tak bez problemu znajdzie ją w menu głównym programu. Z tego względu, przed przystąpieniem do pracy z programem CX-Programmer zaleca się wykonanie „personalizacji" pasków narzędzi, tak aby usunąć zbędne przyciski i dostosować wygląd programu do własnych preferencji. W tym celu należy uruchomić polecenie z menu View/Toolbars, czego efektem będzie pojawienie się okna „Customize" – jak na rysunku 5.2a. W oknie tym wyszczególnione są wszystkie aktywne i nieaktywne standardowe paski narzędzi, które użytkownik może dowolnie włączać i wyłączać. Po wybraniu zakładki „Commands" (Rys.5.2b) możliwa jest dalsza personalizacja palety narzędzi, polegająca na dodaniu nowych lub usunięciu niepotrzebnych przycisków metodą „przeciągnij i upuść".



Rys.5.2. Personalizacja palety przycisków w programie CX-Programmer.

Poniżej przedstawiono znaczenie poszczególnych przycisków na wybranych paskach narzędzi.

Pasek narzędzi „Standard"



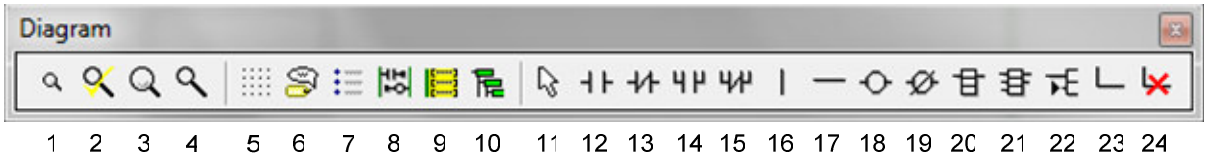
1 – Tworzy nowy projekt
2 – Otwiera zapisany projekt

3 – Zapisuje projekt
4 – Porównuje aktualny program z innym programem, zapisanym na dysku
5 – Drukuje bieżący projekt
6 – Otwiera podgląd wydruku
7 – Wytnij
8 – Kopiuj
9 – Wklej
10 – Cofnij ostatnią operację
11 – Ponów cofniętą operację
12 – Znajdź
13 – Zamień
14 – Zamień wszystko
15 – Wyświetla okno „o programie” (About...)
16 – Pomoc kontekstowa

Pasek narzędzi „Widok”

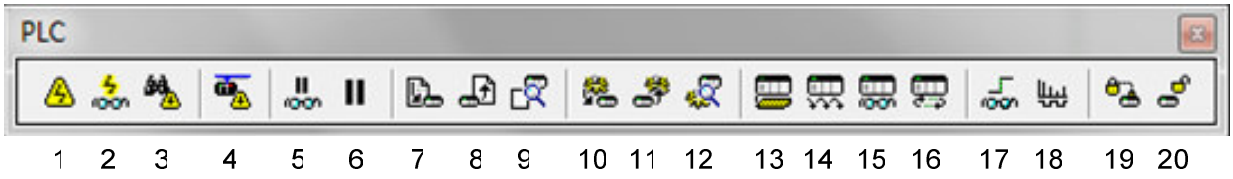
1 – Otwórz/zamknij okno projektu
2 – Otwórz/zamknij okno komunikatów wyjściowych
3 – Otwórz/zamknij okno śledzenia zmiennych (*Watch window*)
4 – Otwórz/zamknij okno adresów źródłowych
5 – Otwórz/zamknij okno podglądu bloku funkcyjnego
6 – Właściwości obiektu
7 – Okno użycia zmiennych „*Cross Reference*”
8 – Otwórz tabelę zmiennych lokalnych
9 – Widok schematu drabinkowego
10 – Reprezentacja programu w postaci listy instrukcji
11 – Okno komentarzy użytych zmiennych
12 – Wyświetla monitorowane wartości w postaci dziesiętnej (*Decimal*)
13 – Wyświetla monitorowane wartości w postaci dziesiętnej ze znakiem (*Signed Decimal*)
14 – Wyświetla monitorowane wartości w postaci szesnastkowej
15 – Pokaż wyższy poziom (program wywołujący blok funkcyjny)
16 – Pokaż niższy poziom (zawartość bloku funkcyjnego)
17 – Włącz monitorowanie bloku funkcyjnego

Pasek narzędzi „Diagram”



1 – Pomniejsz
2 – Dopasuj powiększenie do okna
3 – Powiększ
4 – Resetuj powiększenie
5 – Włącz siatkę
6 – Włącz/wyłącz opis zmiennych
7 – Włącz/wyłącz komentarze nagłówkowe dla instrukcji
8 – Zawijaj drabinki schematu (jeśli są za długie)
9 – Włącz/wyłącz komentarze sekcji
10 – Pokaż strukturę zagnieżdżeń
11 – Kursor normalny (strzałka)
12 – Wstawia styk NO
13 – Wstawia styk NC
14 – Połączenie równoległe ze stykiem NO
15 – Połączenie równoległe ze stykiem NC
16 – Połączenie pionowe
17 – Połączenie poziome
18 – Cewka (wyjście) normalna
19 – Cewka negująca
20 – Instrukcja języka drabinkowego (funkcja)
21 – Wywołanie bloku funkcyjnego
22 – Edycja parametru (pola) bloku funkcyjnego
23 – Włącz/wyłącz tryb rysowania połączeń
24 – Włącz/wyłącz tryb wymazywania połączeń („gumka”)

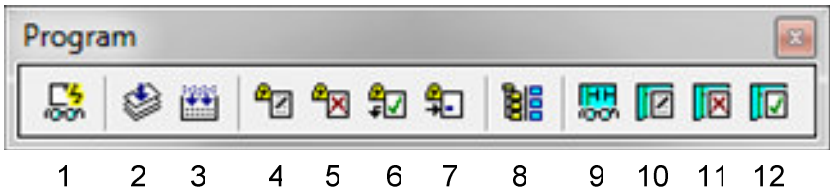
Pasek narzędzi „PLC”



1 – Praca w trybie *On-line*
2 – Przełączenie trybu monitorowania

3 – Bezpośrednie połączenie ze sterownikiem
4 – Tryb *On-line* poprzez sieć Ethernet
5 – Wstrzymuje monitorowanie po wystąpieniu warunku wyzwalającego
6 – Wstrzymuje monitorowanie
7 – Ładowanie programu do PLC
8 – Odczyt programu z PLC
9 – Porównanie bieżącego programu z programem zawartym w PLC
10 – Ładowanie zadania do PLC
11 – Odczyt zadania z PLC
12 – Porównanie bieżącego zadania z zadaniem zawartym w PLC
13 – Tryb programowania
14 – Tryb debugowania
15 – Tryb Monitor
16 – Tryb RUN
17 – Monitorowanie zbocza sygnału
18 – Narzędzie Data Trace
19 – Ustawienie hasła zabezpieczającego program
20 – Usunięcie hasła zabezpieczającego

Pasek narzędzi „Program”



1 – Przełącza okno monitorowania
2 – kompiluje bieżący program
3 – kompiluje wszystkie programy projektu
4 – Edycja w trybie On-line
5 – Koniec edycji w trybie On-line
6 – Akceptacja zmian w trybie On-line
7 – Przejście do edytowanej linii
8 – Uruchomienie menadżera sekcji/linii
9 – Włączenie trybu tylko do odczytu
10 – Rozpoczyna edycję w trybie tylko do odczytu
11 – Rezygnacja z wprowadzonych zmian
12 – Akceptacja wprowadzonych zmian

Pasek narzędzi „Simulator Debug”

1 – Uruchamia symulator zintegrowany z symulatorem paneli operatorskich
2 – Uruchamia symulator programowy
3 – Uruchamia symulator błędów sterownika
4 – Ustawia/usuwa punkt wstrzymania
5 – Usuwa wszystkie punkty wstrzymania
6 – Uruchamia program (w trybie Monitor)
7 – Zatrzymuje i resetuje program (tryb programowania)
8 – Wstrzymuje działanie programu (pauza)
9 – Wykonuje jeden krok programu
10 – Wykonuje jeden krok programu (wchodzi w głąb obiektu zagnieżdżonego, np. FB)
11 – Opuszcza obiekt zagnieżdżony (FB)
12 – Wykonuje program krok po kroku
13 – Wykonuje program przez jeden cykl przetwarzania sterownika

5.2.2. ZAKŁADANIE I KONFIGURACJA NOWEGO PROJEKTU

W niniejszym rozdziale zostaną przedstawione instrukcje dotyczące zakładania i konfiguracji nowego projektu, na przykładzie sterowników CJ1M znajdujących się na stanowiskach ćwiczeniowych w Laboratorium Sterowania Urządzeń i Napędów Przemysłowych, w Instytucie Maszyn Napędów i Pomiarów Elektrycznych w Politechnice Wrocławskiej.

Na rysunku 5.3 przedstawiono drzewo typowego projektu. Każdy nowotworzony projekt posiada charakterystyczną strukturę, w której można wyróżnić następujące elementy składowe:

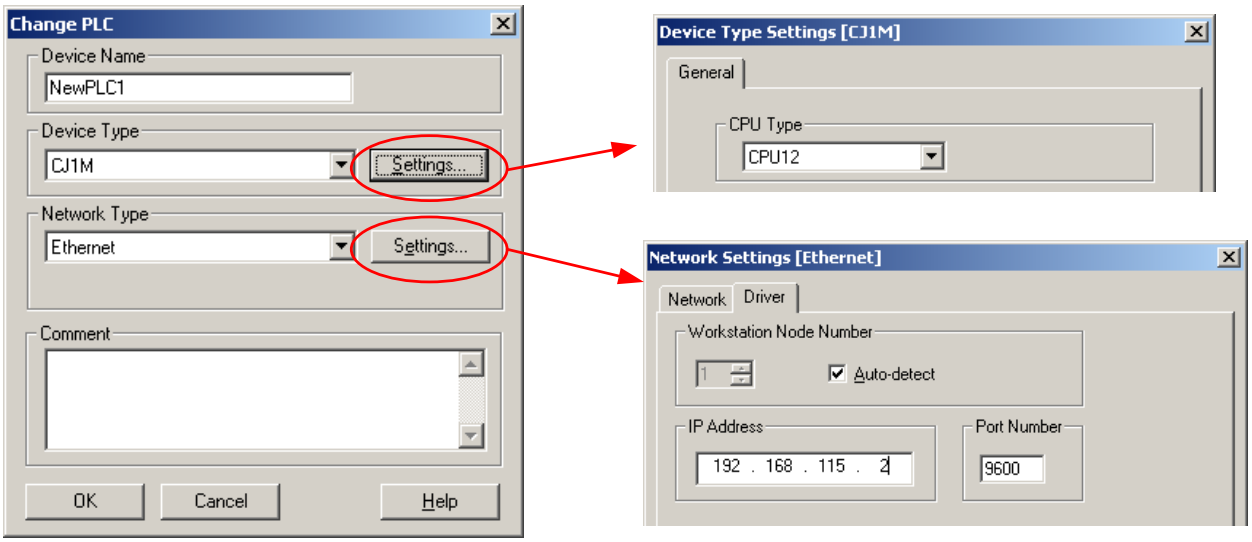
- *Nazwa projektu* – dowolny identyfikator projektu, który powinien kojarzyć się z realizowanym zadaniem;
- *Nazwa sterownika* – w przypadku, gdy projekt składa się z kilku sterowników, warto im nadawać odpowiednio dobrane nazwy, które ułatwią ich identyfikację;
- *Symbols* – tablica zawierająca nazwy wszystkich symboli i zmiennych globalnych, dostępnych w obrębie całego projektu;
- *IO Table and Unit Setup* – tabela zawierająca listę wszystkich dołączonych modułów rozszerzeń oraz ich ustawienia;

- *Memory card* – narzędzie dostępu do karty pamięci *Compact flash* (dostęp w trybie *online*);
- *Error log* – dziennik błędów sterownika (dostęp w trybie *online*);
- *PLC Clock* – narzędzie ustawień bieżącej daty i czasu sterownika (dostęp w trybie *online*);
- *Settings* – przechowuje ustawienia parametrów sterownika;
- *Memory* – narzędzie dostępu do wszystkich obszarów pamięci sterownika, które mogą być odczytywane i modyfikowane w trybie *online*;
- *Programs* – zawiera programy użytkownika, które są przypisane do zadań (*Tasks*). Poszczególne zadania mogą być wykonywane cyklicznie, lub zdarzeniowo – w postaci przerwań sprzętowych lub programowych. Do programów mogą być dołączone tabele zmiennych lokalnych, które nie są widoczne dla innych programów;
- *Program główny* – wykonywany jest w każdym cyklu przetwarzania sterownika;
- *Sekcje* – poszczególne programy mogą być podzielone na sekcje, które są wykonywane jedna po drugiej, w kolejności ich występowania;
- *Function Blocks* – bloki funkcyjne, które mogą być utworzone przez użytkownika lub wczytane z biblioteki.

Rys.5.3. Elementy składowe projektu.

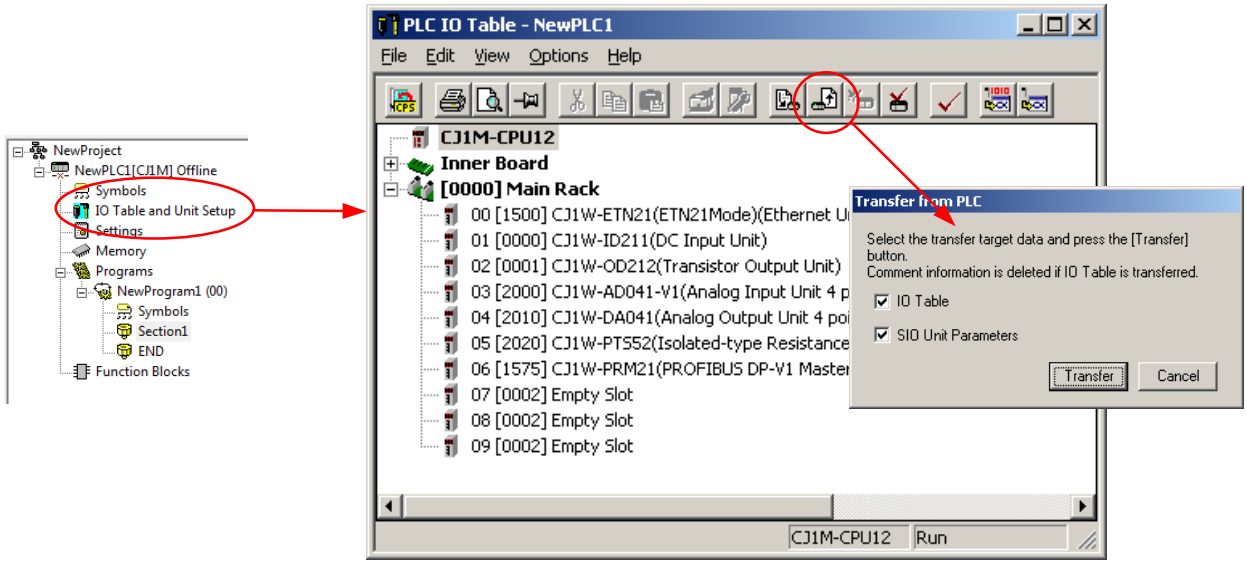
W celu założenia nowego projektu w programie CX-Programmer należy wybrać polecenie *New*, znajdujące się w menu *File*. W wyniku tego pojawi się okno dialogowe „Change PLC", dotyczące ustawień sterownika PLC (Rys. 5.4). W polu *Device Name* należy nadać dowolną nazwę dla sterownika a następnie wybrać jego

typ, rozwijając listę *Device Type*. Dla wybranego typu sterownika należy określić rodzaj posiadanej jednostki centralnej (w tym przypadku CPU12), w oknie dialogowym, wywoływanym po naciśnięciu przycisku *Settings*. W następnej kolejności, w polu *Network Type* należy określić rodzaj interfejsu komunikacyjnego, używanego do zaprogramowania sterownika. Dla sterowników znajdujących się w laboratorium będzie to sieć Ethernet, której parametry trzeba zdefiniować w oknie *Network Settings* (Rys. 5.4), dostępnym po naciśnięciu przycisku *Settings*. W zakładce *Network* nie trzeba wprowadzać żadnych ustawień, natomiast w zakładce *Driver* należy wprowadzić adres IP sterownika programowalnego, w zakresie podsieci 192.168.115.xxx. Po zatwierdzeniu wprowadzonych ustawień zostanie otwarte główne okno projektowe, takie jak na rysunku 5.1.



Rys.5.4. Wybór i ustawienia sterownika w programie CX-Programmer.

Po założeniu nowego projektu należy utworzyć tabelę wejść/wyjść sterownika (*IO Table*), przedstawiającą konfigurację adresową i ustawienia wszystkich modułów rozszerzeń. Można to zrobić na dwa sposoby. Pierwszy sposób polega na ręcznej edycji tabeli I/O, w której dodaje się kolejne moduły rozszerzeń i wprowadza ich ustawienia, zgodnie z ustawieniami fizycznymi sterownika. Drugi sposób, znacznie prostszy, polega na wczytaniu istniejącej już tabeli I/O, bezpośrednio z pamięci sterownika. W tym celu trzeba połączyć się ze sterownikiem, wybierając polecenie *Work Online* dostępne w menu *PLC* lub na pasku narzędzi. W dalszej kolejności należy otworzyć tabelę I/O, klikając dwukrotnie na pozycji *IO Table and Unit Setup*, znajdującej się w oknie projektu po lewej stronie. Następnie trzeba przetransferować ze sterownika zawartość tej tabeli oraz parametry ustawień modułów specjalnych. Szczegóły tej operacji przedstawiono na rysunku 5.5. Po tych czynnościach można przystąpić do edycji programu użytkownika.



Rys.5.5. Konfiguracja tabeli wejść/wyjść sterownika.

5.2.3. EDYCJA PROGRAMU W JĘZYKU DRABINKOWYM

W środowisku CX-Programmer programy mogą być opracowywane w języku drabinkowym, ST oraz SFC. Z racji, że język drabinkowy jest najpopularniejszy, poniżej zostanie omówiony sposób edycji programu użytkownika właśnie w tym języku. Struktura projektu umożliwia podział programu na mniejsze fragmenty, zwane sekcjami. Poszczególne sekcje są wykonywane w każdym cyklu, kolejno jedna po drugiej. Kolejność wykonywania sekcji zależy od ich umiejscowienia w drzewie projektu. Obowiązuje tu reguła, że sekcja położona najwyżej jest wykonywana jako pierwsza.

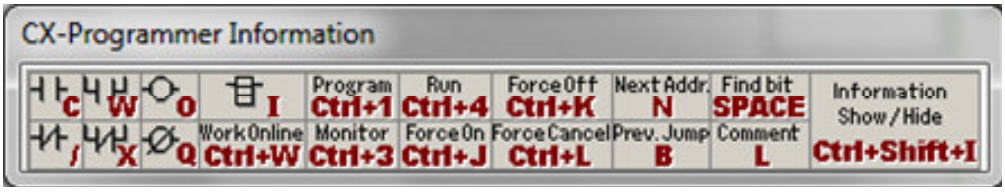
Po założeniu nowego projektu, automatycznie zostają utworzone dwie sekcje, o nazwach domyślnych *Section1* oraz *END*. W sekcji *END* znajduje się tylko jedna instrukcja – *END*, która jest instrukcją kończącą każdy program i jest wymagana przez kompilator. Użytkownik może umieścić instrukcję END w dowolnym miejscu programu, jednak należy pamiętać, że wszystkie instrukcje programu znajdujące się poza nią nie będą wykonywane przez sterownik, a przy próbie kompilacji zostanie wyświetlone ostrzeżenie „*Unreachable code at rung...*”.

Główny program użytkownika jest tworzony w sekcji *Section1*, której nazwę można dowolnie zmienić. W przypadku bardzo rozbudowanych programów, zaleca się podzielenie ich na mniejsze fragmenty, które będą umieszczone w kilku kolejnych sekcjach. Zabieg ten spowoduje zwiększenie czytelności programu oraz ułatwi jego analizę pod kątem znalezienia ewentualnych błędów. Aby dodać nową sekcję do programu należy kliknąć prawym przyciskiem na nazwie programu w drzewie projektu (domyślnie *NewProgram1*) i wybrać polecenie „*Insert new section*”. W

wyniku tej operacji, w strukturze projektu pojawi się dodatkowa sekcja o nazwie domyślnej *Section2*.

Edycja programu w języku drabinkowym polega na umieszczeniu w oknie edytora schematów, odpowiednich instrukcji stykowych i funkcji specjalnych. W trakcie opracowywania programu, CX-Programmer na bieżąco kontroluje poprawność jego składni i podświetla w kolorze czerwonym wszelkie nieprawidłowości, takie jak: niewłaściwe użycie instrukcji, błędna struktura schematu czy nieprawidłowy typ danych. Pełna kompilacja programu możliwa jest po usunięciu wszystkich błędów semantycznych.

Początkujący programiści zazwyczaj tworzą schematy drabinkowe przy użyciu myszy oraz przycisków dostępnych na pasku narzędzi „*Diagram*". Z kolei bardziej doświadczeni użytkownicy najczęściej edytują swoje programy bez użycia myszki, wyłącznie za pomocą klawiatury komputera. Metoda ta przy odrobinie wprawy przynosi najlepsze efekty, ponieważ jest wygodniejsza i zdecydowanie szybsza. W środowisku CX-Programmer można włączyć okno informacji (*Ctrl+Shift+I*), które jest umieszczone zawsze na wierzchu i przedstawia najważniejsze predefiniowane skróty klawiszowe, wykorzystywane do szybkiej edycji i uruchamiania programu. Rysunek 5.6 przedstawia wygląd okienka informacji. Oprócz tego, użytkownik może zdefiniować własne skróty klawiszowe dla większości poleceń CX-Programmera, wybierając w menu *Tools* polecenie *Keyboard Mapping*.

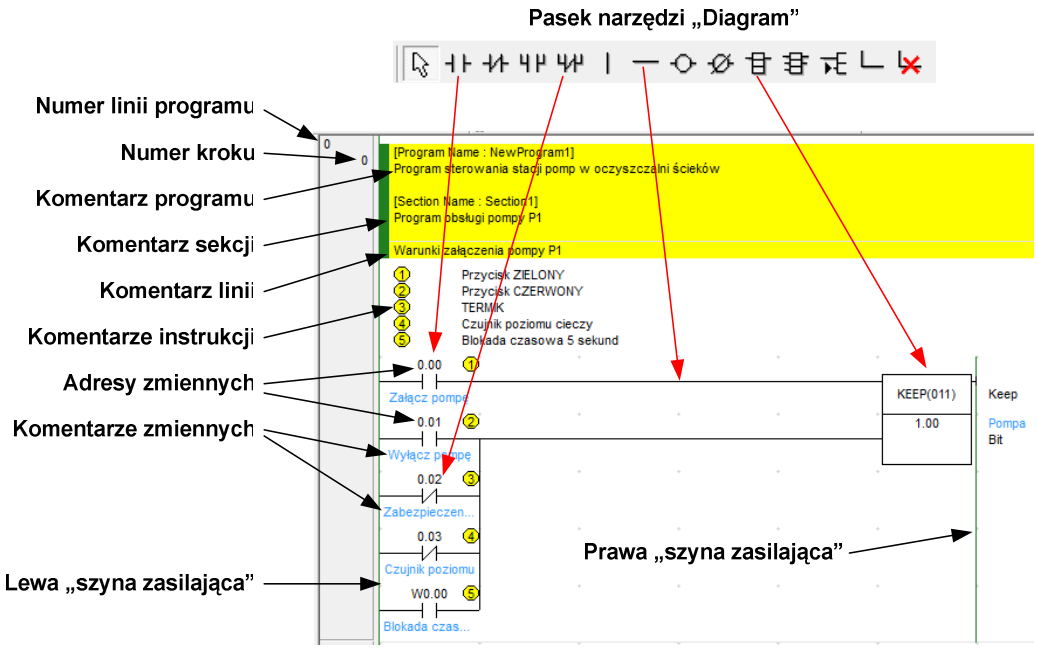


Rys.5.6. Okno informacji o skrótach klawiszowych programu CX-Programmer.

Na rysunku 5.7 przedstawiono fragment okna edytora schematów, na którym widać jedną linię przykładowego programu do sterowania pompy. W programie tym wyróżnić można kilka charakterystycznych elementów, do których należą:

- instrukcje stykowe – elementy reprezentujące stan zmiennych boolowskich w postaci styków normalnie otwartych i normalnie zamkniętych. Możliwe są połączenia równoległe i szeregowe styków oraz kombinacje mieszane tych połączeń;
- instrukcje funkcyjne – oznaczane na schemacie w postaci prostokąta. Większość spotykanych instrukcji funkcyjnych to instrukcje kończące linię programu. Oznacza to, że z prawej strony instrukcji nie można podłączyć już żadnego elementu. Na schemacie przykładowym znajduje się instrukcja KEEP, realizująca funkcję przerzutnika RS;

- komentarze programu, sekcji, linii, zmiennych oraz instrukcji – przedstawiają opis tekstowy tych elementów, znacznie ułatwiający późniejszą analizę programu i identyfikację jego fragmentów.



Rys.5.7. Okno edytora schematów – podstawowe elementy programu.

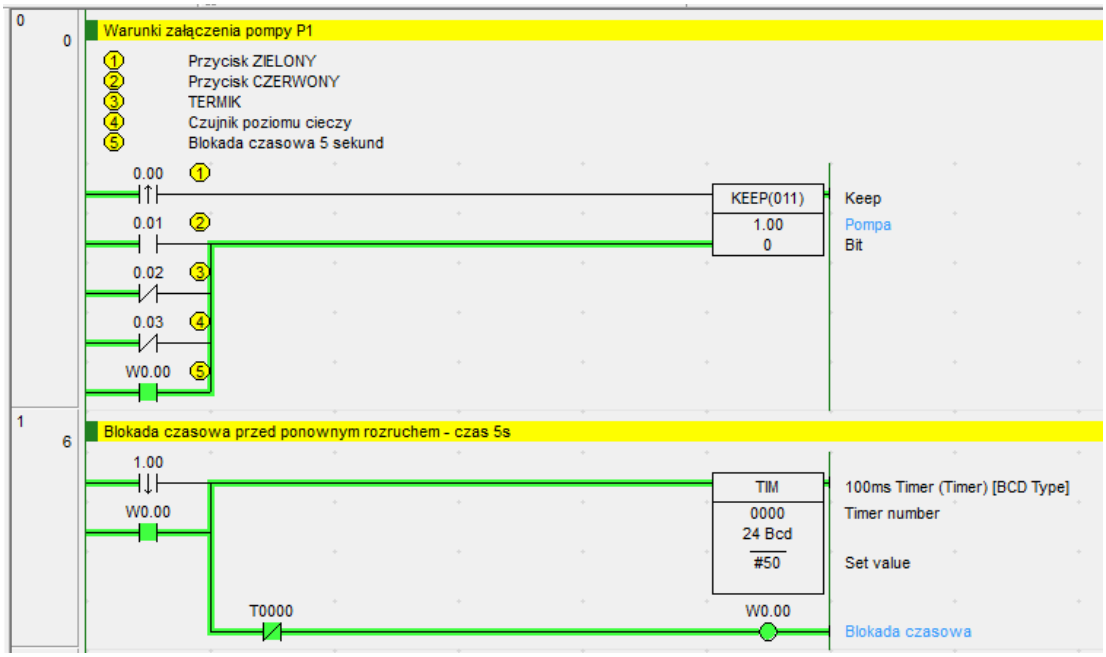
Chociaż umieszczanie komentarzy w programie jest dobrowolne, dobrym zwyczajem jest stosowanie ich możliwie często. Dobrze napisany program, oprócz poprawnego zapisu algorytmu działania, powinien posiadać przejrzystą strukturę i właściwie dobrane komentarze, objaśniające działanie zastosowanych funkcji i algorytmów oraz opisujące znaczenie użytych zmiennych. Z funkcji komentarzy warto często korzystać szczególnie w sterownikach firmy OMRON, ponieważ są one zapisywane w pamięci sterownika równolegle z programem i listą zmiennych. Jest to bardzo istotne z punktu widzenia konieczności wprowadzenia późniejszych zmian w programie, zwłaszcza przez osoby inne niż autor oprogramowania. Wyświetlanie komentarzy w oknie programu można w dowolnej chwili wyłączyć, używając przycisków na pasku narzędzi „*Diagram*”, skrótów klawiszowych lub poleceń w menu *View*.

5.2.4. URUCHAMIANIE I TESTOWANIE PROGRAMÓW

Przed uruchomieniem programu warto sprawdzić jego poprawność pod kątem występowania wszelkich błędów składniowych i strukturalnych. W tym celu należy użyć polecenia kompilacji, które jest dostępne na pasku narzędzi *„Program”*, w menu głównym (*PLC / Compile All PLC Programs*) lub poprzez naciśnięcie klawisza F7. W

przypadku, gdy w programie znajdą się jakieś nieprawidłowości, kompilator umieści odpowiednie komunikaty o błędach i ostrzeżeniach w oknie wyjściowym (*Output Window*). W celu odnalezienia zgłaszanego błędu w programie wystarczy dwukrotnie kliknąć na wiersz, w którym znajduje się komunikat o błędzie (*ERROR ...*), wówczas w programie zostanie podświetlona nieprawidłowa instrukcja.

Po usunięciu wszystkich błędów w programie można przystąpić do załadowania programu do sterownika. W tym celu należy połączyć się ze sterownikiem, wybierając polecenie *Work OnLine*. W trybie połączenia *OnLine* tło okna edytora schematu zmienia się z koloru białego na kolor szary. Następnie należy załadować program(y) do sterownika, używając polecenia *Transfer To PLC*. W pojawiającym się oknie dialogowym należy zaznaczyć te składniki, które powinny zostać wysłane do sterownika (*Programs, Settings, Symbols, Comments...*). Po wykonaniu tych czynności sterownik jest gotowy do pracy i może zostać ustawiony w trybie wykonywania programu (*RUN Mode*) lub w trybie monitorowania (*Monitor mode*). W fazie uruchamiania programów prototypowych szczególnie przydatny jest tryb monitorowania, gdyż użytkownik uzyskuje dostęp do wszystkich zasobów pamięciowych sterownika, co daje mu szerokie możliwości testowania działającego programu.

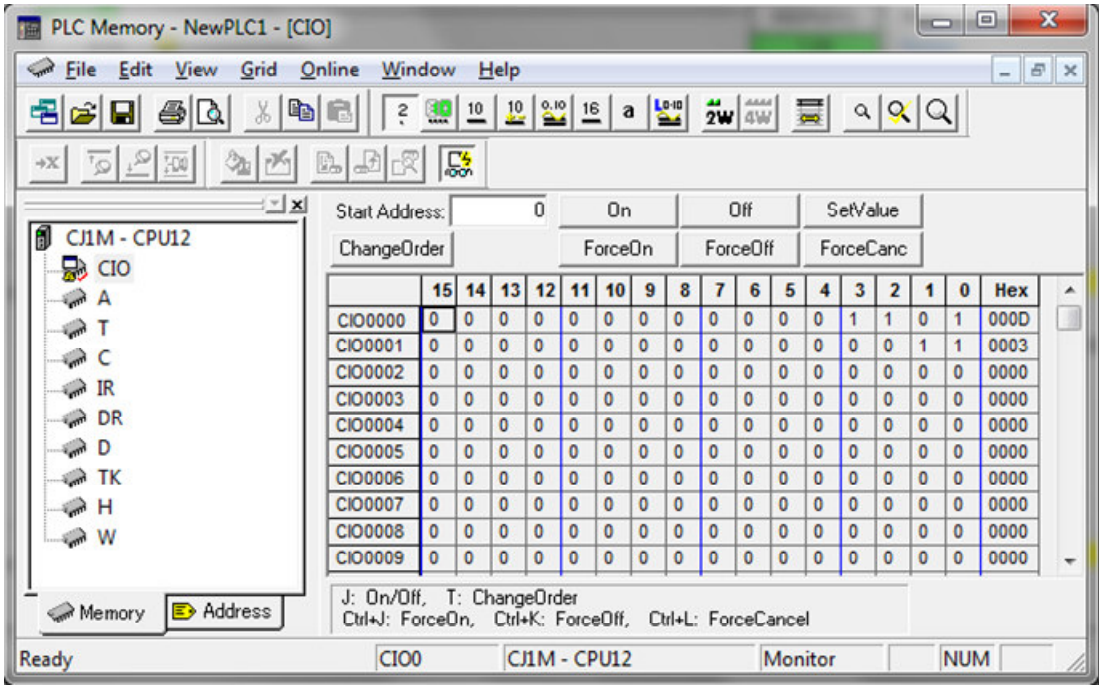


Rys.5.8. Przykładowy program w trybie monitorowania.

Na rysunku 5.8 przedstawiono okno podglądu działania przykładowego programu, uruchomionego w sterowniku w trybie monitorowania. Użytkownik na bieżąco może obserwować występowanie wysokich stanów logicznych na poszczególnych elementach i liniach sygnałowych, które są zaznaczone kolorem zielonym. Oprócz

tego, tryb monitorowania daje możliwość zmiany wartości większości zmiennych logicznych i modyfikacji zawartości innych komórek pamięci (np. zmiana wartości liczników, parametrów wejściowych czasomierzy, itp.).

Na rysunku 5.9 przedstawiono okno narzędzia „*PLC Memory*", które pozwala na dostęp do wszystkich obszarów pamięci sterownika i umożliwia podgląd komórek pamięci oraz edycję ich wartości. Po lewej stronie znajduje się okno wyboru typu pamięci. W prawej części z kolei znajduje się wyświetlana zawartość wybranego obszaru. Po włączeniu trybu monitorowania można na bieżąco podglądać i modyfikować zawartość poszczególnych komórek. Wszystkie wprowadzane zmiany są na bieżąco aktualizowane w fizycznej pamięci sterownika, dzięki czemu można w czasie rzeczywistym zmieniać parametry i bezpośrednio oddziaływać na przebieg sterowanego procesu.



Rys.5.9. Okno programu PLC Memory w trybie monitorowania pamięci CIO.

Oprócz przedstawionych podstawowych możliwości testowania programów użytkownika w czasie rzeczywistym, środowisko CX-Programmer oferuje znacznie więcej funkcji ułatwiających debugowanie i śledzenie przebiegu działania programów. Do najciekawszych narzędzi zaliczyć można [15]:

- **Differential Monitor** – służy do wykrywania i zliczania krótkotrwałych stanów logicznych (zboczy) i sygnalizacji ich wystąpienia w postaci graficznej i akustycznej;
- **Pause Upon Trigger** – funkcja, która zatrzymuje tryb monitorowania po wystąpieniu określonego zdarzenia wyzwalającego;

- **Data Trace** oraz **Time Chart Monitor** – programy do monitorowania i wyświetlania zmian sygnałów w funkcji czasu. Ich działanie przypomina pracę wielokanałowego oscyloskopu, który rejestruje sygnały w takt zadanej podstawy czasu, po wystąpieniu określonych warunków wyzwalania;
- **Switch Box Utility** – narzędzie do wygodnej zmiany wartości bitów i słów wybranych obszarów pamięci.

Przedstawione wyżej narzędzia współpracują również z symulatorem programowym, zintegrowanym w środowisku CX-Programmer. Symulator ten umożliwia przetestowanie działania programów użytkownika wirtualnie, bez konieczności posiadania sterownika rzeczywistego. Narzędzie to jest niezwykle przydatne dla początkujących programistów, którzy mają możliwość przetestowania większości instrukcji i funkcji języka drabinkowego, dostępnych dla różnych sterowników firmy OMRON.

Aby uruchomić symulator należy wybrać polecenie *Work Online Simulator* znajdujące się w menu *Simulation* lub nacisnąć odpowiedni przycisk na pasku narzędzi *Simulator Debug*. W tym momencie testowany program zostanie uruchomiony w wirtualnym sterowniku, w trybie monitorowania, dając użytkownikowi możliwość podglądu wartości poszczególnych zmiennych i stanów logicznych na połączeniach sygnałowych. W trybie monitorowania można również modyfikować wartości różnych zmiennych oraz zmieniać stany logiczne instrukcji stykowych. W celu zmiany wartości wybranej komórki lub styku, należy umieścić na niej kursor i nacisnąć klawisz *Enter*. W pojawiającym się oknie dialogowym należy wprowadzić nową wartość w polu *Value* i zatwierdzić jej zmianę klawiszem *OK*. Wprowadzone zmiany dają efekt natychmiastowy, powodując odpowiednią reakcję programu użytkownika.

6. Wybrane instrukcje i funkcje języka drabinkowego dla sterowników OMRON

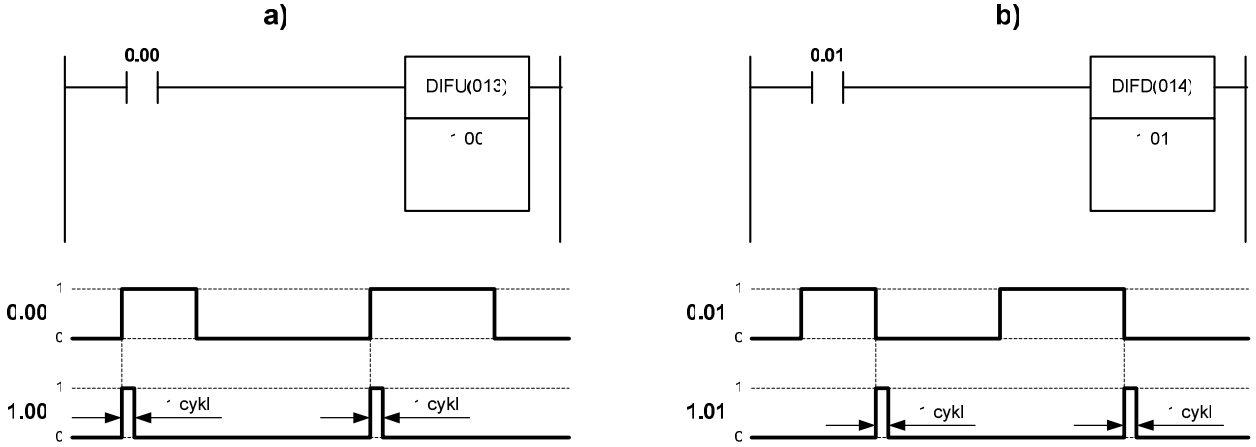
6.1. Instrukcje detekcji zboczy

Detektory zboczy należą do podstawowych instrukcji bitowych, które wykrywają dynamiczne zmiany sygnałów cyfrowych. Wyróżnia się dwa rodzaje zboczy:

- zbocze narastające – zmiana wartości logicznej ze stanu niskiego na wysoki (0→1);
- zbocze opadające – zmiana wartości logicznej ze stanu wysokiego na niski (1→0);

6.1.1. INSTRUKCJE DIFU I DIFD

Do wykrywania zbocza narastającego stosuje się instrukcję **DIFU** (*Differentate Up*). Instrukcja ta posiada jedno wejście sygnałowe oraz jeden operand wyjściowy – dowolny adres bitowy z obszarów pamięci CIO, W, H, A, IR. Jeżeli wartość na wejściu sygnałowym zmieni się ze stanu 0 na 1, operand wyjściowy zostanie ustawiony na wartość logiczną 1 na czas trwania jednego cyklu przetwarzania sterownika [4, 12].



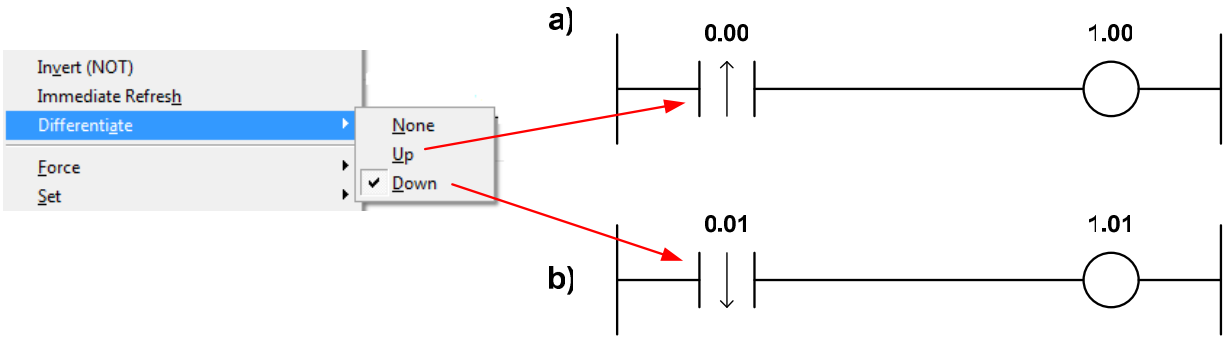
Rys.6.1. Działanie instrukcji DIFU (a) i DIFD (b).

Do wykrywania zbocza opadającego służy instrukcja **DIFD** (*Differentiate Down*), która działa analogicznie do **DIFU**, z tą różnicą, że reaguje na zmianę stanu sygnału wejściowego z wartości 1 na 0.

Na rysunku 6.1 przedstawiono przykładowe programy wykorzystujące instrukcje DIFU i DIFD oraz wykresy czasowe, objaśniające działanie tych instrukcji.

6.1.2. INSTRUKCJE STYKOWE RÓŻNICZKUJĄCE

W programie CX-Programmer dostępne są również różniczkujące instrukcje stykowe, reagujące na zmiany stanów logicznych skojarzonych z nimi zmiennych. W celu wprowadzenia warunku różniczkowania do instrukcji stykowej, należy kliknąć na niej prawym przyciskiem myszy i wybrać opcję *Differentiate>Up* lub *Differentiate>Down*. Wówczas na symbolu instrukcji stykowej pojawi się strzałka skierowana do góry (dla detektora zbocza narastającego) lub w dół (dla detektora zbocza opadającego). Przykład wykorzystania tych instrukcji przedstawiono na rysunku 6.2.



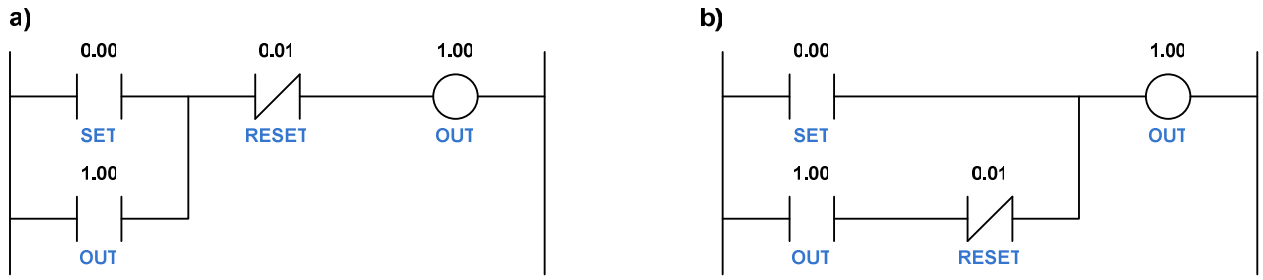
Rys.6.2. Zastosowanie styków wykrywających zbocze narastające (a) i opadające (b).

6.2. Elementy bistabilne

Podstawowymi elementami bistabilnymi są przerzutniki RS i SR, zaliczane do najprostszych elementów pamięciowych. Elementy te posiadają dwa wejścia i jedno wyjście. Wejście ustawiające (SET) powoduje trwałe ustawienie stanu wysokiego na wyjściu przerzutnika, natomiast wejście kasujące (RESET) powoduje zapamiętanie stanu niskiego na wyjściu. Różnica pomiędzy przerzutnikami RS i SR polega na odmiennym priorytecie sygnałów wejściowych. W przypadku, gdy na oba wejścia zostanie jednocześnie podany stan wysoki, wyjście przerzutnika RS przyjmie niski, zaś wyjście przerzutnika SR ustawi się w stan wysoki.

6.2.1. ZASTOSOWANIE INSTRUKCJI STYKOWYCH DO REALIZACJI FUNKCJI PRZERZUTNIKÓW

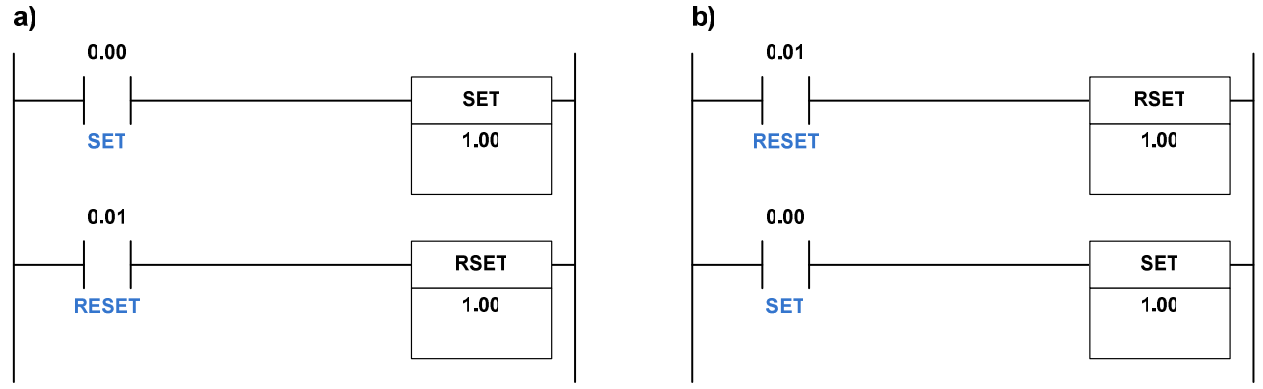
W programie CX-Programmer, w języku drabinkowym istnieje kilka możliwości realizacji funkcji przerzutników RS i SR. Jedną z nich jest wykorzystanie zwykłych instrukcji stykowych, połączonych w układzie sprzężenia zwrotnego od wyjścia (tzw. układy z podtrzymaniem). Przykład wykorzystania instrukcji stykowych do realizacji obu typów przerzutników przedstawiono na rysunku 6.3.



Rys.6.3. Zastosowanie instrukcji stykowych do realizacji funkcji przerzutników RS (a) i SR (b).

6.2.2. INSTRUKCJE SET I RSET

Inną metodą realizacji przerzutników RS i SR jest wykorzystanie instrukcji **SET** i **RSET**, które posiadają po jednym wejściu i jednym operandzie wyjściowym. Instrukcje te zazwyczaj występują w parze (dla tego samego operandu wyjściowego), choć nie koniecznie muszą być użyte w jednym miejscu programu. Podanie stanu wysokiego na wejście instrukcji SET powoduje trwałe przypisanie jedynki logicznej do operandu wyjściowego. Konsekwentnie, w instrukcji RSET, podanie na wejście stanu wysokiego spowoduje wyzerowanie operandu wyjściowego.

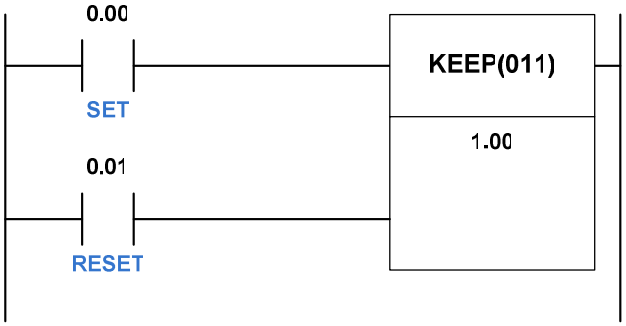


Rys.6.4. Zastosowanie instrukcji SET i RSET do realizacji przerzutników RS (a) i SR (b).

Ponieważ instrukcje SET i RSET występują w programie osobno, kolejność ich występowania na schemacie drabinkowym będzie jednoznacznie określać priorytet działania tak powstałego przerzutnika, przy jednoczesnym podaniu stanów wysokich na wejścia tych instrukcji. Dominującą instrukcją w tym przypadku będzie ta, która w programie zostanie wykonana jako ostatnia. Na rysunku 6.4 przedstawiono przykład realizacji przerzutników RS i SR przy wykorzystaniu instrukcji SET i RSET.

6.2.3. INSTRUKCJA KEEP

Oprócz przedstawionych wyżej metod realizacji funkcji przerzutników RS i SR, w programie CX-Programmer dostępna jest instrukcja **KEEP**, będąca gotowym odpowiednikiem klasycznego przerzutnika RS. Instrukcja ta posiada dwa wejścia (SET i RESET) oraz jeden operand wyjściowy, typu boolowskiego. Sposób wykorzystania tej instrukcji przedstawiono na rysunku 6.5.



Rys.6.5. Zastosowanie instrukcji KEEP realizującej funkcję przerzutnika RS.

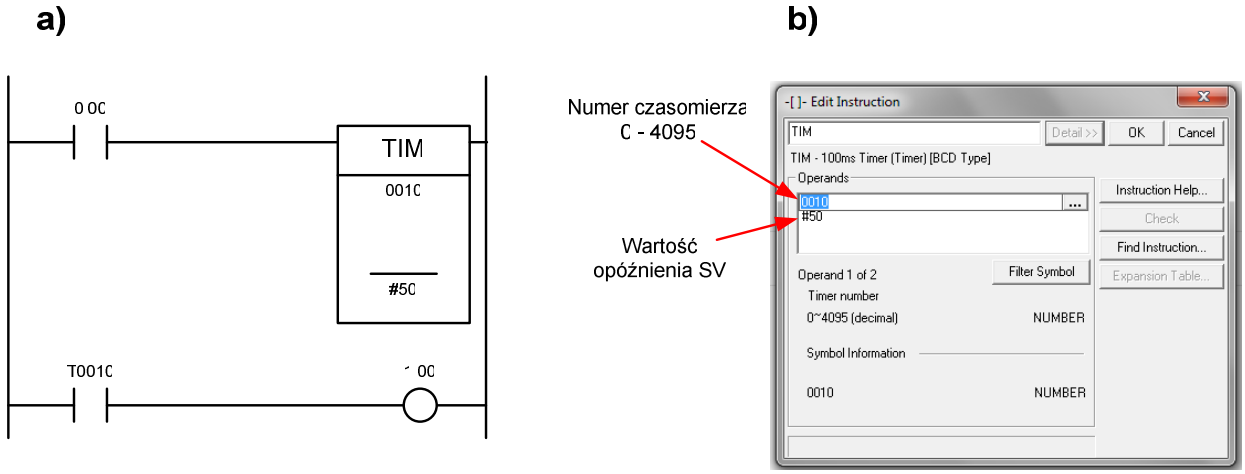
6.3. Czasomierze

W sterownikach programowalnych OMRON rodziny CJ występuje kilka instrukcji realizujących funkcje czasowe. Wszystkie czasomierze wykorzystują wspólny obszar pamięci typu T, do przechowywania aktualnych wartości opóźnienia oraz stanów logicznych ich flag wypełnienia. W sterownikach serii CJ można wykorzystać maksymalnie 4096 czasomierzy [12].

6.3.1. INSTRUKCJE TIM, TIMH I TMHH

Podstawową instrukcją realizującą funkcję czasomierza jest instrukcja **TIM**. Czasomierz ten umożliwia odmierzanie czasu w zakresie 0-999,9s z rozdzielczością 100ms. Wartość zadaną opóźnienia wprowadza się w postaci operandu SV (ang. *Set Value*), reprezentującego liczbę całkowitą w kodzie BCD, która wyraża krotność

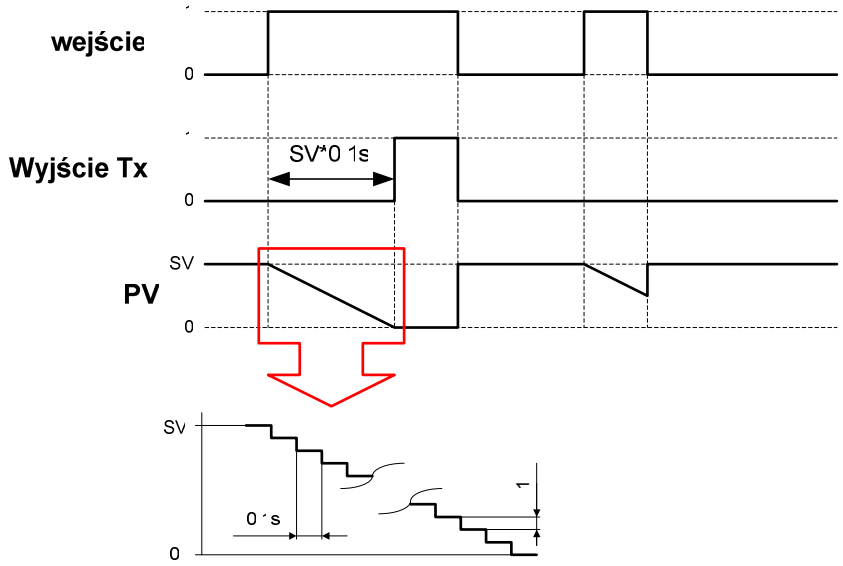
jednostkowego opóźnienia o wartości 0,1s. Przykładowo, chcąc uzyskać czas opóźnienia równy 5s, należy wprowadzić wartość wejściową #50 (50*0,1s=5s). Każdy użyty czasomierz TIM musi posiadać swój unikalny numer z zakresu 0-4095, który jednoznacznie identyfikuje go w obszarze pamięci T. Ponadto, numer ten związany jest bezpośrednio z tzw. flagą wypełnienia czasomierza, która zostaje ustawiona po odmierzeniu ustalonego czasu. Podobnie jak większość instrukcji w języku drabinkowym sterowników OMRON, instrukcja TIM jest instrukcją kończącą linię programu. Na rysunku 6.6 przedstawiono przykład zastosowania czasomierza oraz sposób parametryzacji instrukcji TIM.



Rys.6.6. Zastosowanie instrukcji TIM – program przykładowy (a) i edycja parametrów czasomierza (b).

W programie przykładowym przedstawionym na rysunku 6.6a, czasomierz TIM (o numerze T0010) działa w układzie opóźniającym załączenie wyjścia sterownika. Po ustawieniu stanu wysokiego na wejściu 0.00, czasomierz rozpoczyna odliczanie czasu opóźnienia, odejmując co 100ms wartość 1, od wartości początkowej 50 do zera. Z chwilą wyzerowania czasomierza (po upływie 5s) następuje ustawienie znacznika wypełnienia T10, który powoduje ustawienie stanu wysokiego na wyjściu 1.00 sterownika. Stan taki trwa do momentu, kiedy na wejściu 0.00 pojawi się niski stan logiczny. Wykres czasowy objaśniający działanie czasomierza TIM przedstawiono na rysunku 6.7.

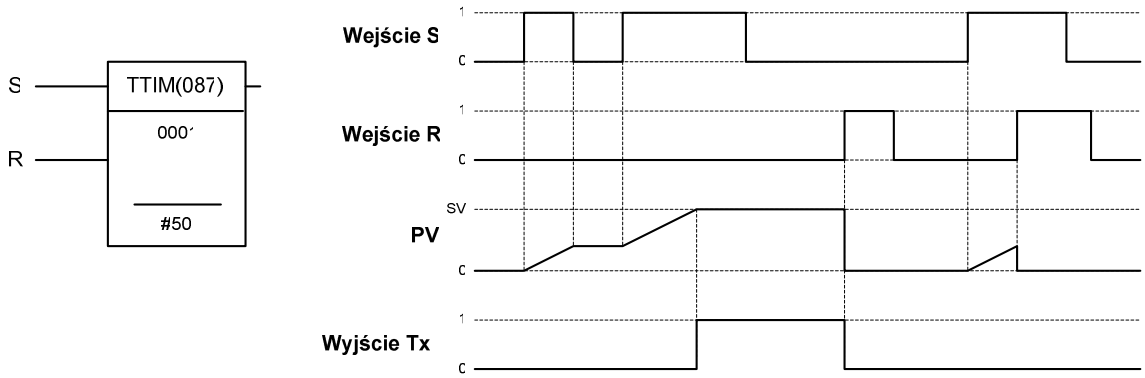
W sterownikach CJ1 czasomierze TIM występują dodatkowo w dwóch „szybszych" wersjach: **TIMH** (*High-Speed Timer*) oraz **TMHH** (*Ultra-High-Speed Timer*). Zasada działania tych czasomierzy jest taka sama jak czasomierza TIM, z tą różnicą, że wartość zadana SV jest dekrementowana z krokiem 1/100 s w czasomierzu TIMH oraz z krokiem 1/1000 s w czasomierzu TTMH. Przy stosowaniu ultraszybkiego czasomierza typu TTMH należy pamiętać, aby jego numer był wybierany z przedziału 0–15. W przeciwnym wypadku czasomierz może działać niedokładnie [4, 12].



Rys.6.7. Wykres czasowy czasomierza TIM.

6.3.2. CZASOMIERZ AKUMULUJĄCY TTIM

Oprócz standardowego czasomierza, realizującego funkcję opóźnionego załączenia, w programie CX-Programmer dostępny jest tzw. czasomierz akumulujący, uruchamiany za pomocą instrukcji **TTIM** (ang. *Totalising Timer*). Czasomierz ten posiada dwa wejścia: **S** (*SET*) – wejście wyzwalające oraz **R** (*RESET*) – wejście kasujące.



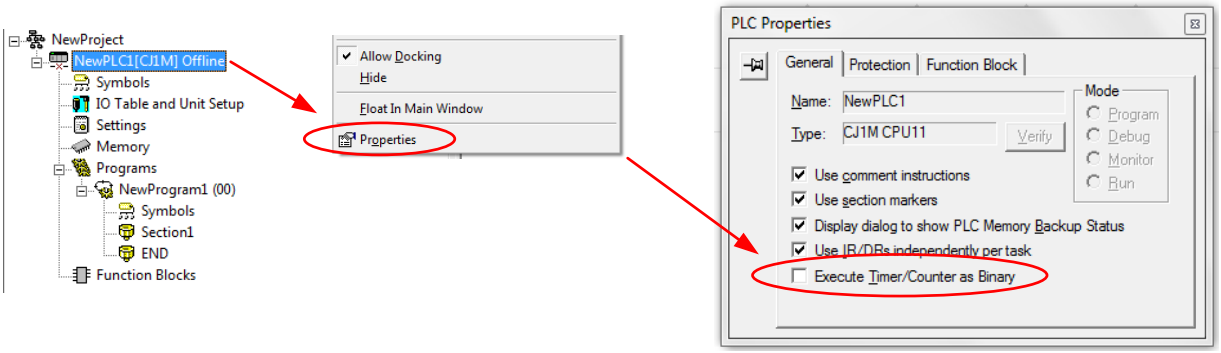
Rys.6.8. Zasada działania czasomierza TTIM.

Podanie na wejściu S wysokiego stanu logicznego powoduje inkrementację wartości PV czasomierza z krokiem 100ms. Po wyłączeniu sygnału wyzwalającego, czasomierz przestaje zwiększać wartość PV, a jej ostatnia wartość zostaje zapamiętana. Ponowne podanie stanu wysokiego na wejściu S powoduje dalszą inkrementację wartości PV (akumulacja). Flaga wypełnienia czasomierza zostaje ustawiona z chwilą, gdy wartość PV osiągnie wartość zadaną, wprowadzoną jako

operand SV. Wejście R służy do wyzerowania wartości PV czasomierza i skasowania flagi wypełnienia. Na rysunku 6.8 przedstawiono wykres czasowy obrazujący działanie czasomierza TTIM.

6.3.3. ZMIANA FORMATU DANYCH CZASOMIERZY

Wszystkie przedstawione czasomierze (TIM, TIMH, TTIM, TTIMH) domyślnie operują na liczbach całkowitych w formacie BCD. W sterownikach serii CJ istnieje możliwość zamiany typu danych w czasomierzach i licznikach, z formatu BCD na format binarny. Po tej zamianie, wartości opóźnienia SV mogą być wprowadzane jako liczby dziesiętne z zakresu 0–65535, lub w zapisie szesnastkowym w przedziale 0–FFFF. Po zamianie typu na binarny, wszystkie mnemoniki instrukcji czasomierzy i liczników otrzymują w nazwie przyrostek X (*TIM→TIMX, TTIM→TTIMX,* ...). Należy jednak pamiętać, że w jednym projekcie można używać tylko jednego formatu danych – niedopuszczalne jest jednoczesne użycie czasomierzy TIM i TIMX. Dlatego też wyboru typu danych należy dokonać na etapie zakładania projektu. Ustawienia te można zmienić klikając prawym klawiszem myszy na nazwie sterownika znajdującej się w drzewie projektu (*NewPLC1*), a następnie wybraniu opcji *Properties* i zaznaczeniu pola „*Execute Timer/Counter as Binary*” [4]. Szczegóły tej zmiany przedstawia rysunek 6.9.



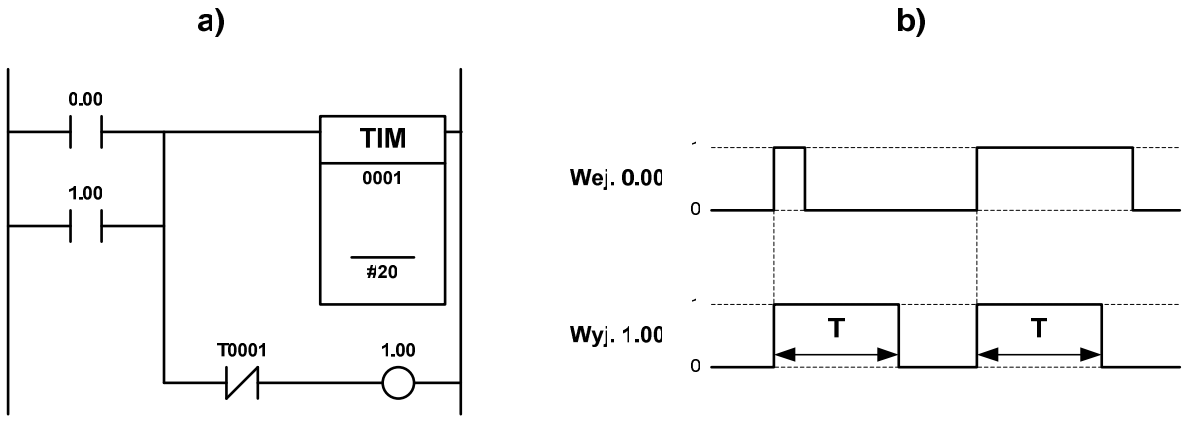
Rys.6.9. Wybór typu danych czasomierzy i liczników.

6.3.4. PROGRAMOWA REALIZACJA FUNKCJI CZASOWYCH TYPU TP I TOF

W normie IEC 61131 zdefiniowano trzy typy standardowych czasomierzy: TON, TOF i TP. Analizując działanie czasomierza TIM można stwierdzić, że jest to dokładna implementacja funkcji czasomierza **TON** (*On-Delayed Timer*). Jednak w środowisku CX-Programmer użytkownik nie znajdzie gotowych instrukcji, które realizowałyby funkcje czasomierza impulsowego **TP** (*Pulse Timer*) i czasomierza z opóźnionym wyłączeniem **TOF** (*Off-Delayed Timer*). Ponieważ są to dość często

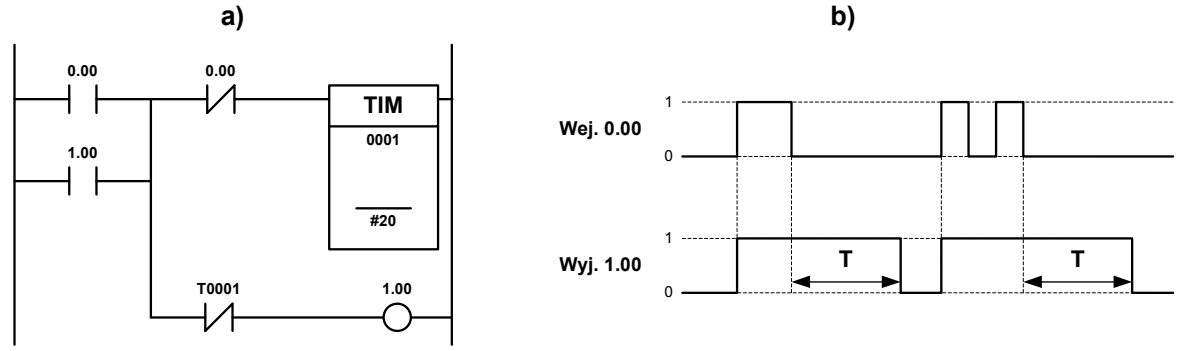
stosowane struktury czasowe, poniżej zostaną przedstawione możliwości programowej realizacji funkcji tych czasomierzy.

Na rysunku 6.10 przedstawiono prosty program realizujący funkcję czasomierza impulsowego TP oraz jego wykres czasowy. Podanie stanu wysokiego na wejście *0.00* powoduje wyzwolenie czasomierza TIM. W tej samej chwili, zanegowana flaga *T0001* powoduje ustawienie stanu wysokiego na wyjściu *1.00*. Wyjście to zostało użyte w roli bitu podtrzymania, który równolegle połączony z wejściem sterującym 0.00, zapewnia ciągłość działania czasomierza TIM po ustawienia stanu niskiego na wejściu *0.00*.



Rys.6.10. Realizacja funkcji czasomierza TP: a) – program w języku drabinkowym, b) – wykres czasowy.

Rysunek 6.11 przedstawia przykładową implementację funkcji czasomierza TOF. Załączenie wejścia *0.00* powoduje przepływ sygnału przez szeregowo połączoną, zanegowaną flagę *T0001* i ustawienie stanu wysokiego wyjściu *1.00*. W chwili wyłączenia wejścia *0.00* następuje wyzwolenie czasomierza TIM, który po upływie nastawionego czasu spowoduje ustawienie flagi *T0001* i jednoczesne wyłączenie wyjścia *1.00*. Jeżeli przed upływem nastawionego czasu wejście *0.00* ponownie zostanie przełączone w stan wysoki, czasomierz TIM zostanie zresetowany, a cała akcja powtórzy się po zaniku sygnału wyzwalającego na wejściu *0.00*.



Rys.6.11. Realizacja czasomierza TOF: a) – program w języku drabinkowym, b) – wykres czasowy.

6.4. Liczniki

W sterownikach serii CJ1 wszystkie liczniki wykorzystują wspólny obszar pamięci typu C, o zakresie adresowym 0–4095. W jednym projekcie można zastosować maksymalnie 4096 liczników, z których każdy zajmuje w pamięci 1 słowo i musi posiadać swój unikalny numer, odpowiadający adresowi komórki z obszaru C [12].

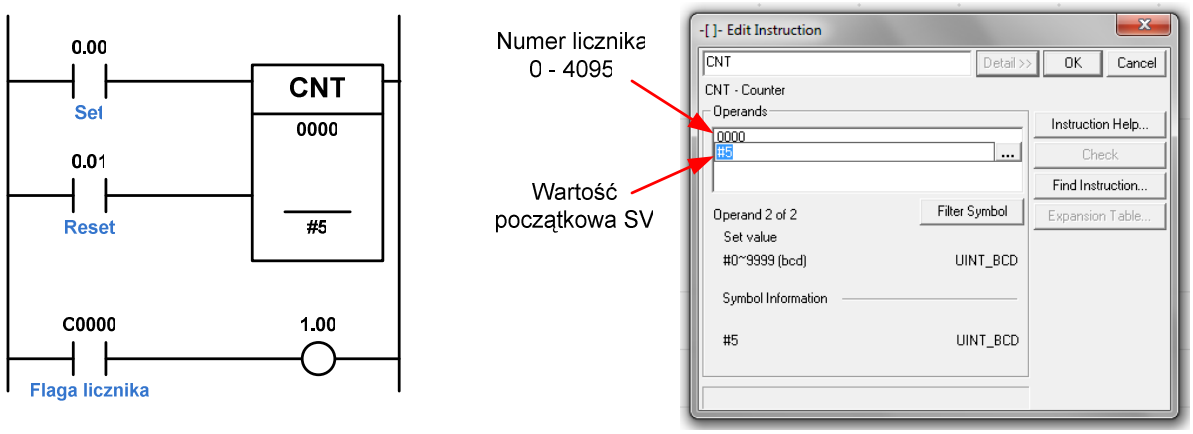
6.4.1. LICZNIK CNT

Instrukcja **CNT** (*Counter*) pełni funkcję podstawowego licznika, którego zadaniem jest zliczanie impulsów pojawiających się na jego wejściu **S**. W tym przypadku, impuls rozumiany jest jako zmiana stanu logicznego z wartości 0 na wartość 1 (zbocze narastające). Licznik CNT jest licznikiem zliczającym w dół, co oznacza, że z każdym impulsem wejściowym następuje dekrementacja wartości PV licznika, począwszy od wartości początkowej, zadeklarowanej jako parametr SV (*Set Value*). Z chwilą osiągnięcia wartości zerowej zostaje ustawiona flaga wypełnienia licznika – bit *Cx*, gdzie indeks *x* odpowiada numerowi licznika. Drugie wejście licznika R służy do skasowaniu flagi wypełnienia i wyzerowania licznika, polegającego na przypisaniu do PV wartości początkowej SV. Na rysunku 6.12 przedstawiono przykład zastosowania instrukcji CNT. Z kolei rysunek 6.13 przedstawia wykres czasowy obrazujący działanie licznika.

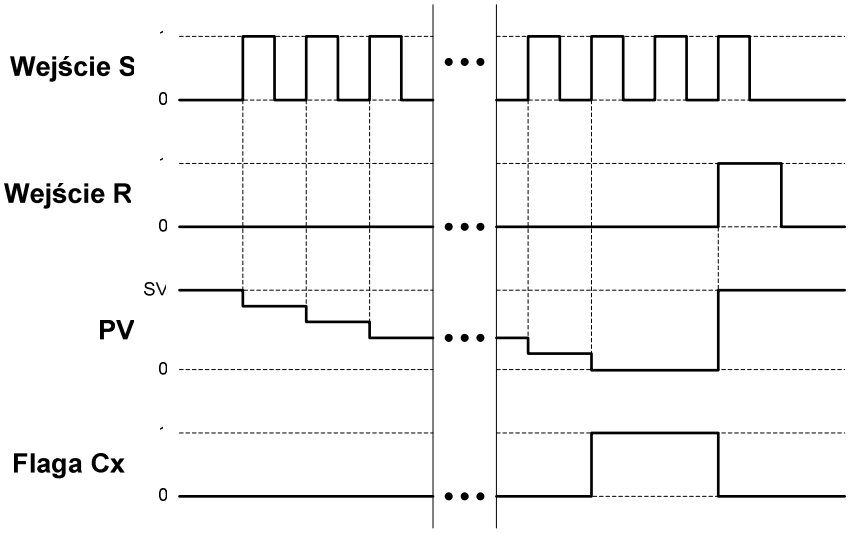
6.4.2. LICZNIK REWERSYJNY CNTR

Instrukcja **CNTR** realizuje funkcję dwukierunkowego licznika, posiadającego dwa wejścia liczące: UP (w górę) i DN (w dół) oraz wejście kasujące R. Licznik CNTR jest licznikiem cyklicznym (okrężnym), co oznacza, że po osiągnięciu wartości maksymalnej proces zliczania w górę kontynuowany jest od wartości zero. Podobna zasada obowiązuje w kierunku zliczania w dół. Maksymalna wartość, do której licznik zlicza impulsy jest określona przez użytkownika jako parametr SV. Po osiągnięciu wartości maksymalnej, kolejny impuls pojawiający się na wejściu UP powoduje wyzerowanie licznika (PV=0) i ustawienie znacznika wypełnienia Cx. Analogicznie, w chwili gdy licznik posiada wartość zerową a na wejściu DN pojawi się impuls, wartość licznika zostanie ustawiona na maksymalną (PV=SV) i również w tym przypadku nastąpi ustawienie flagi wypełnienia Cx. Każdy impuls na wejściach UP lub DN niepowodujący przepełnienia licznika kasuje znacznik wypełnienia. Wejście R służy do natychmiastowego wyzerowania wartości licznika i skasowania flagi wypełnienia.

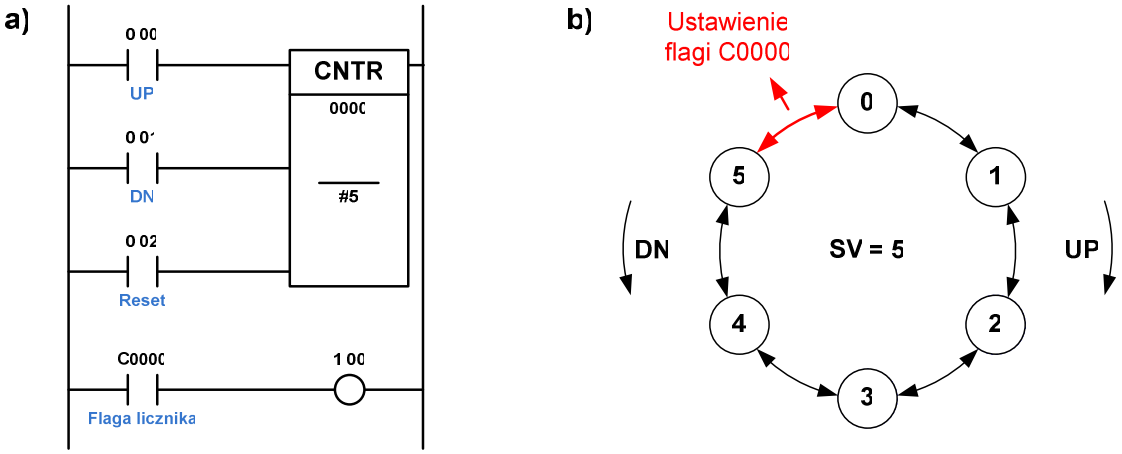
Na rysunku 6.14 przedstawiono przykład wykorzystania licznika CNTR oraz zasadę pracy cyklicznej, a na rysunku 6.15 jego wykres czasowy.



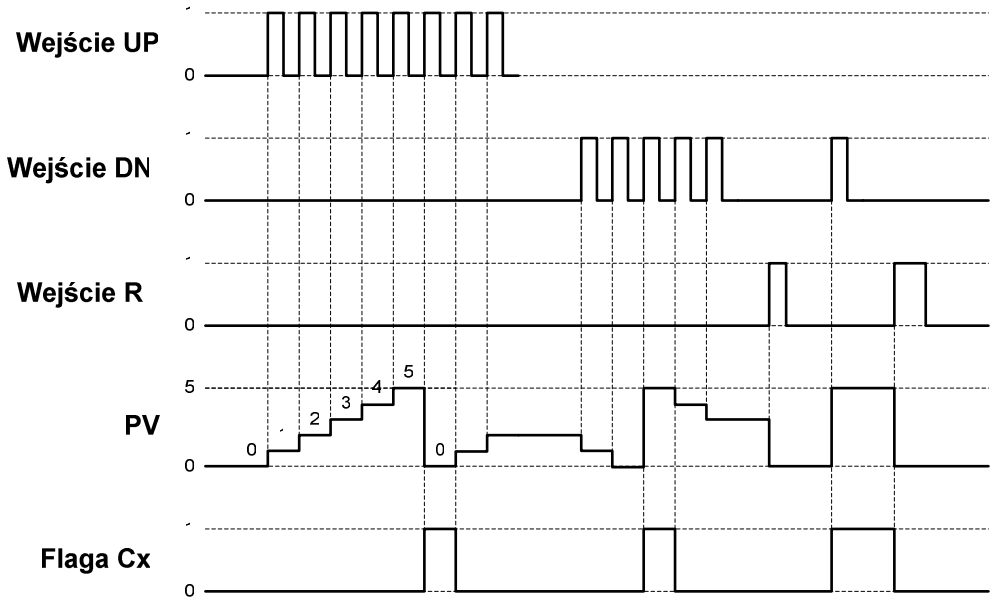
Rys.6.12. Przykład zastosowania instrukcji CNT.



Rys.6.13. Wykres czasowy obrazujący zasadę działania licznika CNT.



Rys.6.14. Przykład użycia licznika CNTR (a) oraz zasada jego pracy cyklicznej (b).



Rys.6.15. Wykres czasowy obrazujący zasadę działania licznika rewersyjnego CNTR.

6.5. Komparatory

Komparatory są to elementy funkcyjne, które wykonują operację porównania ze sobą dwóch wartości liczbowych, najczęściej typu całkowitego lub rzeczywistego. Jeżeli zadany warunek porównania jest spełniony, zostają ustawione odpowiednie flagi.

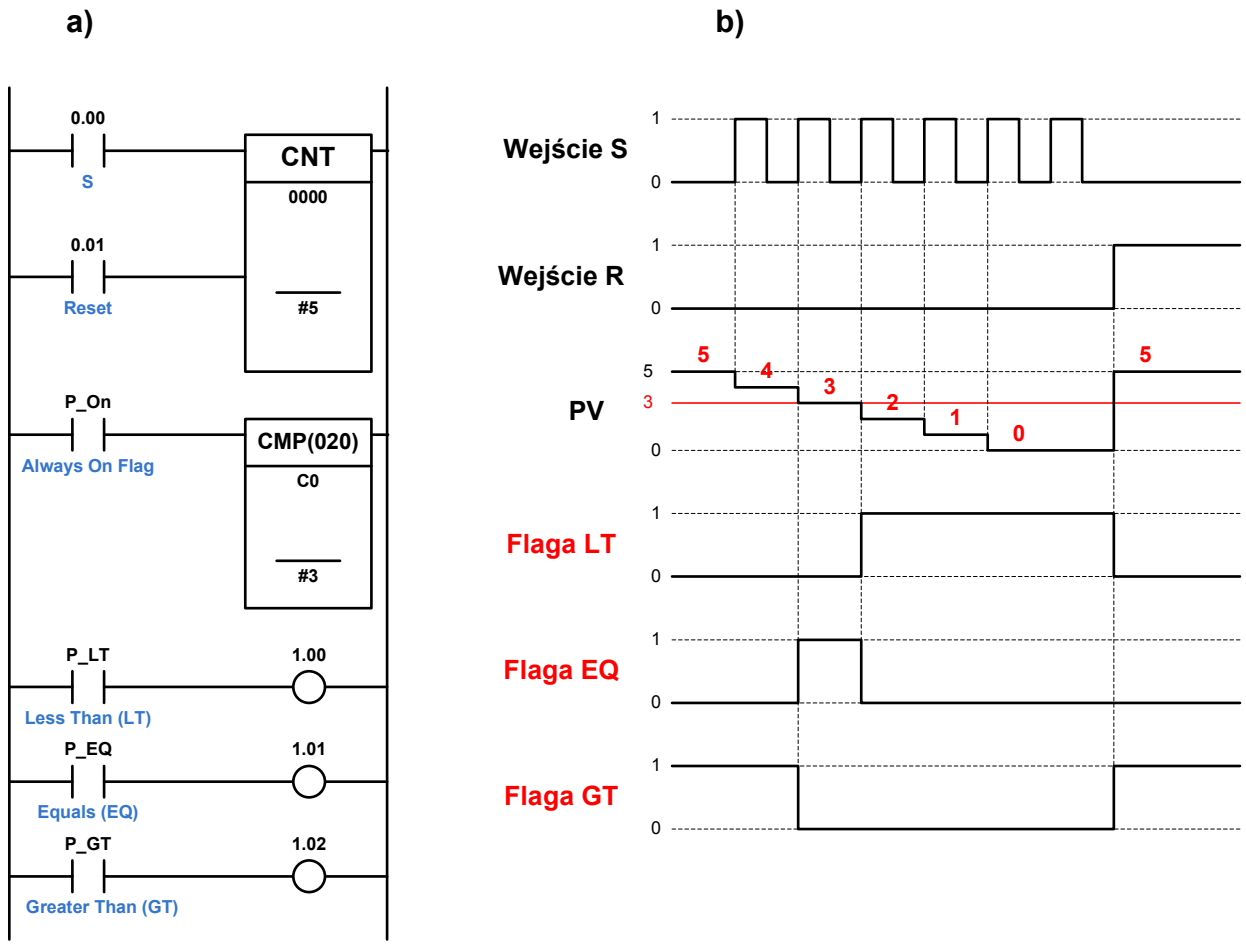
6.5.1. KOMPARATORY CMP I CPS

Instrukcja **CMP** realizuje funkcję komparatora, którego argumentami wejściowymi są liczby całkowite bez znaku, o długości jednego słowa (UINT). W wyniku porównania zostają ustawione systemowe flagi porównania, które określają relację pomiędzy porównywanymi wartościami. Instrukcją pokrewną, działającą na liczbach całkowitych ze znakiem (INT) jest instrukcja **CPS**. W tabeli 6.1 zestawiono wszystkie systemowe flagi porównania oraz odpowiadające im operatory relacji [4].

Na rysunku 6.16a przedstawiono przykładowy program zawierający licznik CNT i komparator CMP, który porównuje bieżącą wartość licznika z wartością #3. W zależności od wyniku tego porównania zostają odpowiednio ustawione flagi *P_LT, P_EQ* i *P_GT*, które dołączone są do kolejnych wyjść sterownika (1.00–1.02). Przedstawiony wykres czasowy (rys.6.16b) objaśnia zasadę działania komparatora CMP, w połączeniu z systemowymi flagami porównania.

Tab.6.1. Systemowe flagi porównania.

Symbol	Operator relacji	Nazwa flagi
P_LT	<	Less Than Flag
P_LE	<=	Less Than or Equal Flag
P_EQ	=	Equal Flag
P_GE	>=	Greather Than or Equal Flag
P_GT	>	Greather Than Flag
P_NE	<>	Not Equal Flag

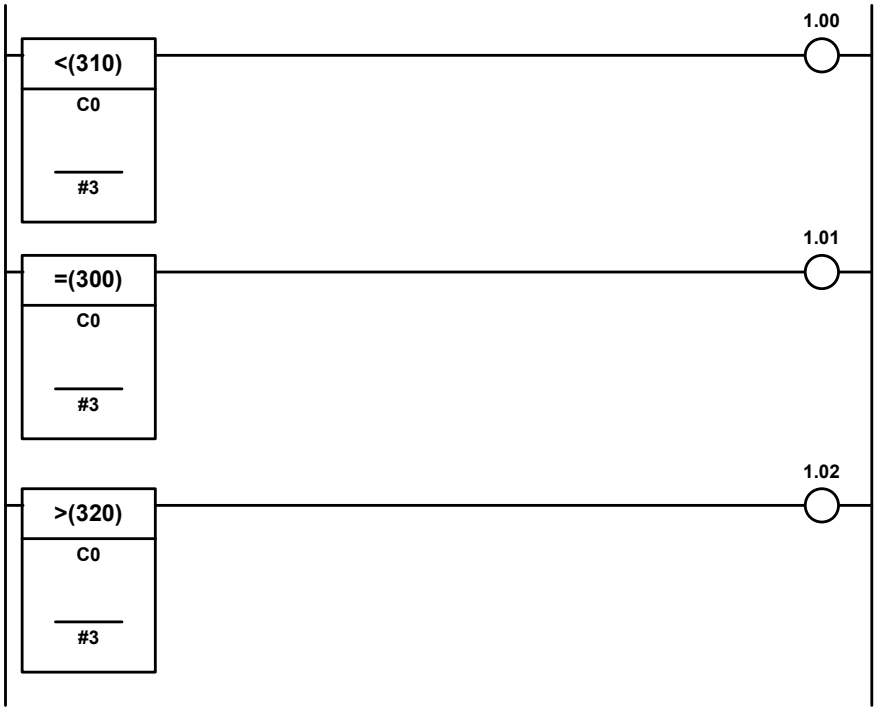


Rys.6.16. Przykładowy program z użyciem komparatora CMP (a) oraz wykres czasowy obrazujący jego działanie (b).

6.5.2. BEZPOŚREDNIE INSTRUKCJE PORÓWNANIA

W sterownikach serii CJ możliwe jest zastosowanie bezpośrednich instrukcji porównania, które mogą być wstawione w dowolnym miejscu schematu drabinkowego. Instrukcje te posiadają jedno wejście boolowskie (warunek wykonywalności) oraz jedno wyjście, na którym pojawia wysoki stan logiczny, jeżeli warunek porównania jest spełniony. Nazwa instrukcji (mnemonik) wskazuje bezpośrednio na wykonywaną przez nią operację porównania (np. =, <, >=). Na rysunku 6.17 przedstawiono fragment przykładowego programu, w którym wykorzystano bezpośrednie instrukcje porównania. W programie tym instrukcje te porównują aktualną wartość licznika *C0* z wartością liczbową #3 i bezpośrednio sterują wyjściami sterownika, bez potrzeby użycia flag porównania. Na wyjściu *1.00* pojawi się wysoki stan logiczny, wówczas, kiedy wartość licznika *C0* będzie mniejsza od 3. Analogicznie, wyjście *1.01* zostanie ustawione w stan wysoki, gdy licznik *C0* osiągnie wartość 3, zaś po jej przekroczeniu zostanie ustawione wyjście *1.02*.

Bezpośrednie instrukcje porównania w podstawowej formie wykonują działania na liczbach całkowitych bez znaku, o długości jednego słowa (*UINT*). Możliwe jest stosowanie dodatkowych modyfikatorów tych instrukcji, które pozwalają na działanie na liczbach o długości 32 bitów (*Double Length*) i/lub na liczbach całkowitych ze znakiem (*Signed*). W tabeli 6.2 przedstawiono wykaz tych modyfikatorów oraz opis ich działania.



Rys.6.17. Przykładowy program z użyciem bezpośrednich instrukcji porównania.

Tab.6.2. Modyfikatory bezpośrednich instrukcji porównania.

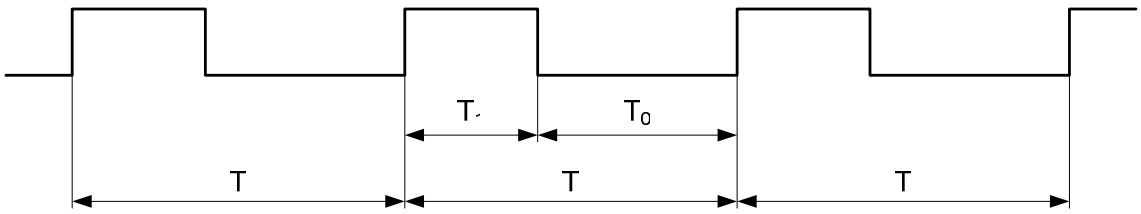
Modyfikator	Typ danych	Przykład zastosowania
brak	liczby całkowite bez znaku w formacie binarnym lub BCD (*Unsigned Data*)	=, >, <, >=, <>
*L	liczby całkowite bez znaku podwójnej długości (*Unsigned Double Length*)	=L, <L, >=L
*S	liczby całkowite ze znakiem w formacie binarnym (*Signed Data*)	=S, >S, <>S
*SL	liczby całkowite ze znakiem podwójnej długości (*Signed Double Length*)	=SL, >=SL, <SL

6.6. Generatory taktujące

Generator taktujący jest elementem, który wytwarza sygnał prostokątny o zadanych parametrach czasowych. Do podstawowych parametrów charakteryzujących przebiegi prostokątne zalicza się:

- częstotliwość **f** ,
- okres sygnału **T** (odwrotność częstotliwości),
- czas trwania wysokiego stanu logicznego $\mathbf{T_1}$,
- czas trwania niskiego stanu logicznego $\mathbf{T_0}$,
- współczynnik wypełnienia $\mathbf{d=T_1/T}$.

Na rysunku 6.18 przedstawiono przykładowy przebieg sygnału prostokątnego, na którym zaznaczono jego podstawowe parametry czasowe.



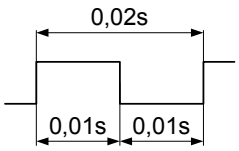
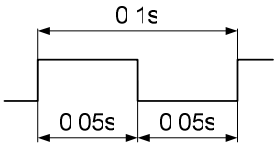
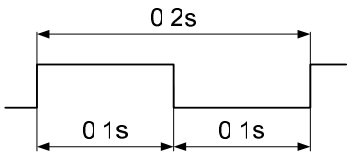
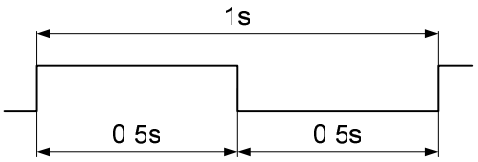
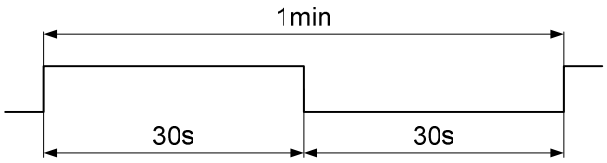
Rys.6.18. Podstawowe parametry czasowe przebiegu prostokątnego.

6.6.1. SYSTEMOWE GENERATORY TAKTUJĄCE

W sterownikach programowalnych OMRON dostępne są specjalne flagi systemowe, reprezentujące wyjścia wewnętrznych generatorów sygnałów prostokątnych, o różnych częstotliwościach i stałym współczynniku wypełnienia, wynoszącym 0,5 (50%). Zastosowanie tych sygnałów w programie użytkownika

polega na przypisaniu odpowiedniej flagi generatora do instrukcji stykowej, typu boolowskiego. W tabeli 6.3 przedstawiono wykaz dostępnych flag generatorów taktujących oraz ich parametry czasowe.

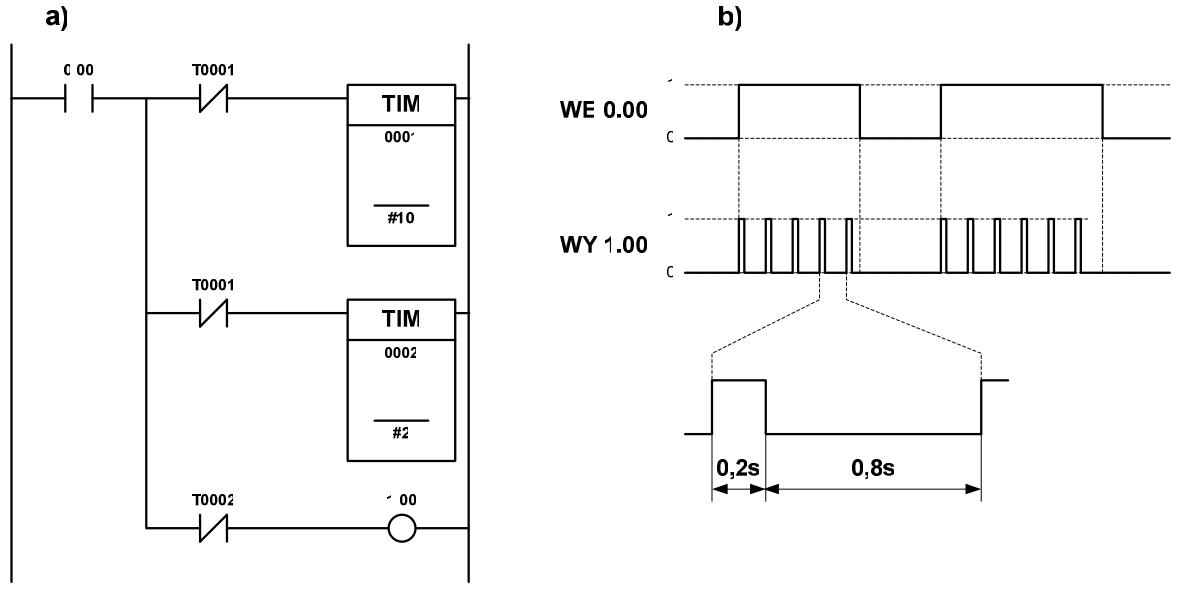
Tab.6.3. Flagi generatorów sygnałów prostokątnych.

Flaga generatora	Przebieg czasowy	Częstotliwość
P_0_02s		f = 50 Hz
P_0_1s		f = 10 Hz
P_0_2s		f = 5 Hz
P_1s		f = 1 Hz
P_1min		f = 1/60 Hz

6.6.2. PROGRAMOWA REALIZACJA GENERATORA TAKTUJĄCEGO

W przypadku, gdy program użytkownika wymaga użycia generatora taktującego o nietypowych parametrach czasowych, wówczas niezbędna jest jego programowa realizacja. Generator taki można wykonać na wiele sposobów. Jedną z częściej stosowanych struktur programowych jest wykorzystanie dwóch czasomierzy TIM, które umożliwiają wytworzenie przebiegów prostokątnych praktycznie o dowolnej częstotliwości i dowolnym współczynniku wypełnienia. Na rysunku 6.19a przedstawiono przykładowy program, realizujący funkcję sterowanego generatora

sygnałów prostokątnych, o częstotliwości 1Hz i współczynniku wypełnienia 20%. W programie tym wykorzystano dwa czasomierze TIM, z których pierwszy (T1) pracuje w układzie generatora podstawy czasu (1Hz), a drugi (T2) odmierza czas trwania wysokiego stanu logicznego (200ms) na wyjściu *1.00*. Generator ten posiada wejście sterujące *0.00*, które musi być ustawione w stan wysoki, aby na wyjściu układu pojawił się sygnał prostokątny. Na rysunku 6.19b przedstawiono wykres czasowy obrazujący działanie generatora.

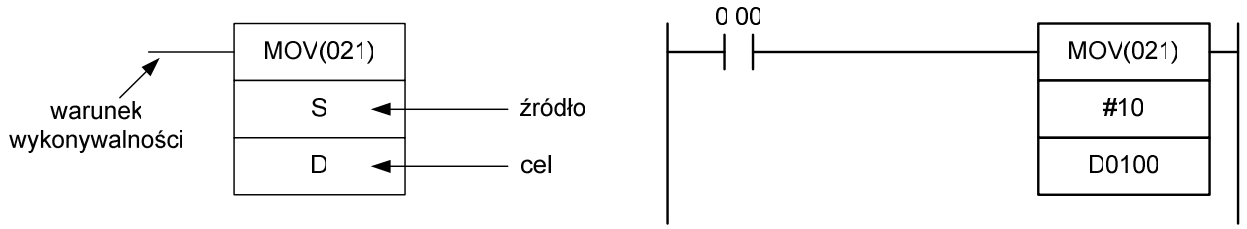


Rys.6.19. Przykład programowej realizacji generatora przebiegu prostokątnego (a) i wykres czasowy obrazujący jego działanie (b).

6.7. Operacje na danych

6.7.1. INSTRUKCJE TRANSFERU DANYCH

Podstawową instrukcją używaną do przenoszenia (kopiowania) danych jest instrukcja **MOV**. Instrukcja ta posiada jedno wejście zezwalające, na które wprowadza się logiczny warunek wykonywalności. Zastosowanie instrukcji MOV wymaga wprowadzenia dwóch operandów, z których pierwszy określa komórkę źródłową albo wartość liczbową, zaś drugi określa adres komórki docelowej, do której ma być zapisana dana. Na rysunku 6.20 przedstawiono sposób użycia instrukcji MOV, która w programie przykładowym wpisuje wartość #10 do komórki pamięci z obszaru D o adresie 100. W tym przypadku, warunkiem wykonywalności tej instrukcji jest wysoki stan logiczny na wejściu *0.00*.

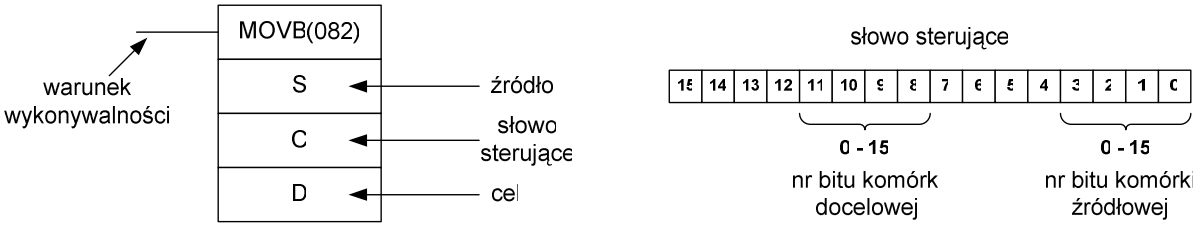


Rys.6.20. Przykład zastosowania instrukcji MOV do wprowadzania danych liczbowych.

Instrukcja MOV występuje również w postaci różniczkowej, która polega na jednokrotnym wykonaniu tej instrukcji z chwilą, gdy warunek wykonywalności zmieni się z wartości 0 na wartość 1. Różniczkową postać instrukcji, która reaguje na zbocze narastające oznacza się dodatkowym prefiksem „@", dodawanym przed mnemonikiem instrukcji (@MOV) [4, 12].

Instrukcja **MVN** (*Move Not*) jest odmianą instrukcji MOV, która pobiera daną z komórki źródłowej, dokonuje inwersji wszystkich jej bitów (negacja słowa) a następnie umieszcza ją w komórce docelowej. Instrukcja MVN podlega takim samym regułom wykonania, jak instrukcja MOV.

Instrukcja **MOVB** służy do skopiowania wartości jednego określonego bitu z komórki źródłowej do wybranego bitu komórki docelowej. Instrukcja ta posiada trzy operandy, które kolejno określają: adres komórki źródłowej, słowo sterujące i adres komórki docelowej. Słowo sterujące zawiera informacje, który bit z komórki źródłowej ma być skopiowany do którego bitu komórki docelowej. Na rysunku 6.21 przedstawiono sposób użycia instrukcji MOVB oraz znaczenie bitów słowa sterującego.

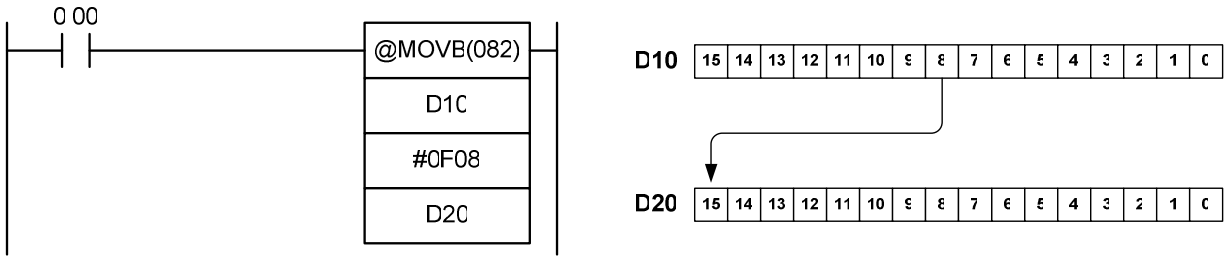


Rys.6.21. Sposób parametryzacji instrukcji MOVB.

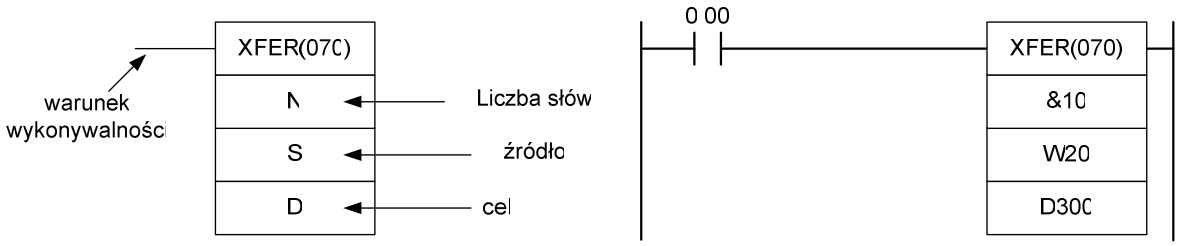
Na rysunku 6.22 przedstawiono przykładowy program wykorzystujący instrukcję MOVB. W programie tym, kopiowana jest wartość ósmego bitu z komórki pamięci D10 do piętnastego bitu komórki D20. Obecność prefiksu „@" przed mnemonikiem instrukcji oznacza, że instrukcja zostanie wykonana jednokrotnie, w chwili zmiany stanu bitu 0.00 z wartości 0 na wartość 1.

Instrukcja **XFER** służy do kopiowania bloków danych. Posiada 3 operandy i jedno wejście zezwolenia wykonania. Pierwszy operand określa liczbę kopiowanych

komórek, drugi wskazuje na adres początkowy źródłowego obszaru pamięci, natomiast trzeci operand określa adres docelowego obszaru pamięci. Na rysunku 6.23 przedstawiono sposób użycia instrukcji XFER, która w programie przykładowym kopiuje 10 kolejnych słów z obszaru pamięci W o adresie początkowym 20, do pamięci z obszaru D o adresie początkowym 300.

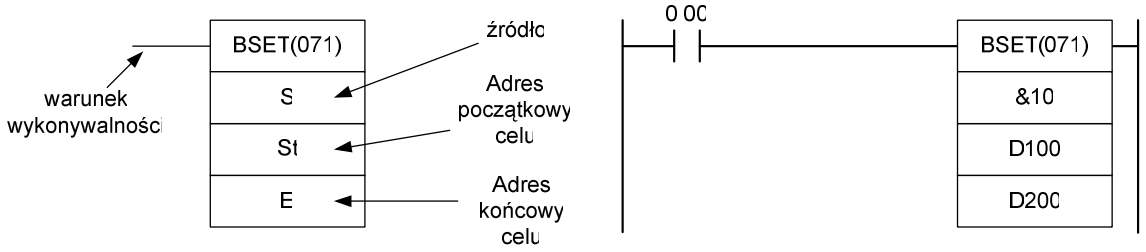


Rys.6.22. Przykładowy program wykorzystujący instrukcję MOVB.



Rys.6.23. Przykład zastosowania instrukcji XFER do kopiowania bloków danych.

Instrukcja **BSET** umożliwia wpisanie tej samej danej do kilku kolejnych komórek pamięci naraz. Instrukcja posiada 3 operandy i wejście zezwolenia. Pierwszy operand wskazuje na adres źródła kopiowanej danej lub określa kopiowaną wartość liczbową. Pozostałe dwa operandy określają adres początkowy i końcowy docelowego obszaru pamięci. Instrukcja ta najczęściej jest stosowana do zerowania lub ustawiania wartości początkowej jednocześnie w wielu komórkach pamięci. Na rysunku 6.24 przedstawiono sposób użycia instrukcji BSET, która w programie przykładowym wpisuje wartość 10 do wszystkich komórek pamięci D, mieszczących się w zakresie adresowym D100–D200.



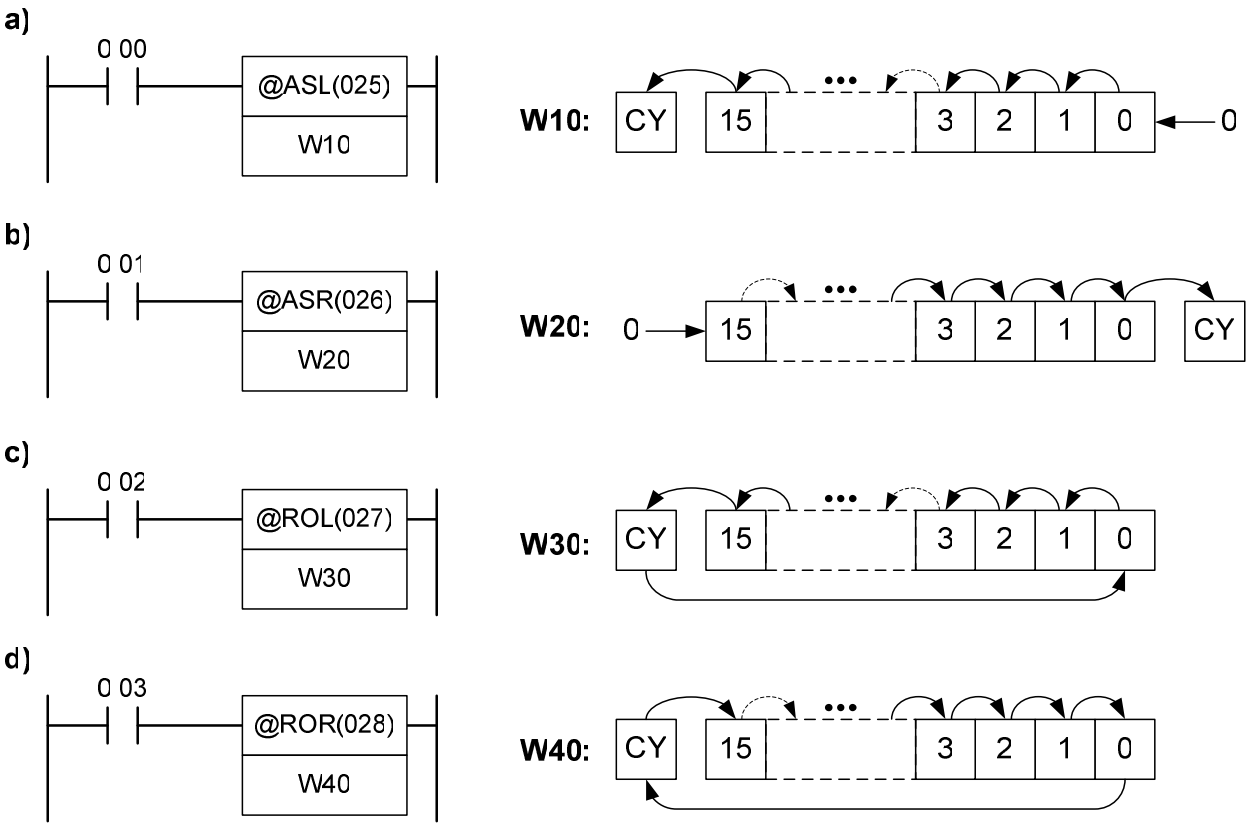
Rys.6.24. Przykład zastosowania instrukcji BSET do wpisywania stałej wartości do bloku danych.

6.7.2. OPERACJE PRZESUWANIA BITÓW

Instrukcje **ASL** i **ASR** realizują funkcję arytmetycznego przesunięcia bitów w słowie, odpowiednio w lewo lub w prawo. Instrukcje te posiadają wejście wyzwalające i jeden operand, który wskazuje na adres przesuwanej danej. W wyniku wykonania operacji przesunięcia danej w lewo, najmłodszy jej bit otrzymuje wartość zerową, zaś wartość najstarszego bitu zostaje wpisana do znacznika CY. Analogicznie, po przesunięciu w prawo najstarszy bit słowa zostaje wyzerowany, a do znacznika CY zostaje wpisana wartość najmłodszego bitu [4].

Instrukcje **ROL** i **ROR** realizują funkcję rotacji – cyklicznego przesunięcia bitów. W wyniku wykonania operacji ROL, poszczególne bity słowa zostają przesunięte o jedną pozycję w lewo, przy czym w miejsce najmłodszego bitu zostaje podstawiona wartość znacznika CY, natomiast wartość najstarszego bitu zostaje wpisana do znacznika CY. Instrukcja ROR działa analogicznie, z tą różnicą, że przesuwa bity w prawo, również z uwzględnieniem flagi przeniesienia CY [4].

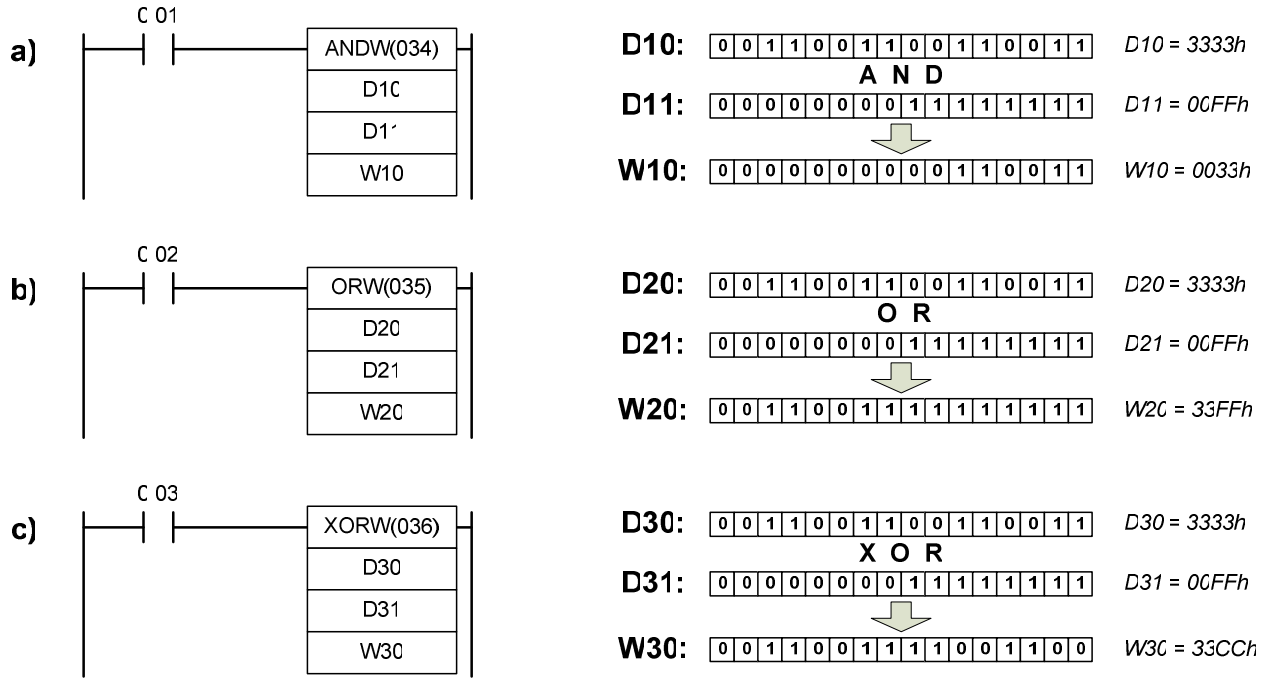
Na rysunku 6.25 przedstawiono zasadę działania poszczególnych instrukcji przesuwania bitów.



Rys.6.25. Zasada działania instrukcji ASL, ASR, ROL i ROR.

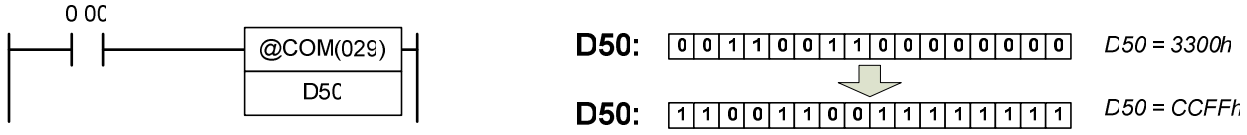
6.7.3. OPERACJE LOGICZNE

Instrukcja **ANDW** realizuje funkcję iloczynu logicznego (AND) na poszczególnych bitach dwóch argumentów wejściowych. Posiada jedno wejście zezwolenia oraz trzy operandy. Pierwsze dwa operandy zawierają dane wejściowe o szerokości jednego słowa. Trzeci operand wskazuje na adres komórki, w której umieszczony będzie wynik operacji logicznej. Identyczną strukturę posiadają instrukcje logiczne **ORW** i **XORW,** realizujące funkcje sumy logicznej (OR) oraz alternatywy rozłącznej (XOR), pomiędzy poszczególnymi bitami dwóch słów wyjściowych. Na rysunku 6.26 przedstawiono zasadę działania wymienionych funkcji logicznych na przykładowych wartościach liczbowych.



Rys.6.26. Zasada działania instrukcji logicznych ANDW, ORW i XORW.

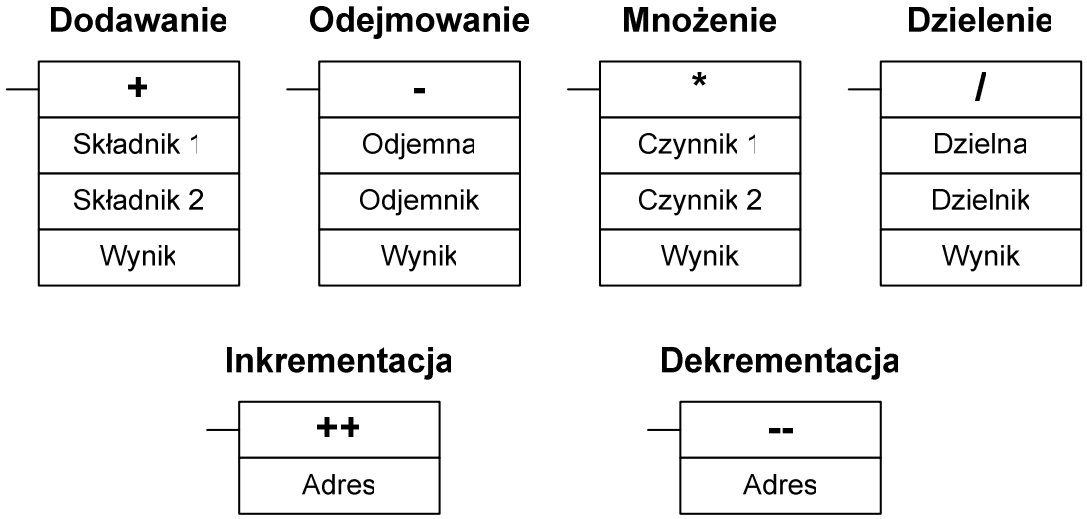
Instrukcja **COM** realizuje funkcję inwersji (negacji) wszystkich bitów argumentu wejściowego o długości 1 słowa. Na rysunku 6.27 przedstawiono zasadę działania tej instrukcji.



Rys.6.27. Zasada działania instrukcji COM.

6.7.4. OPERACJE ARYTMETYCZNE

W sterownikach programowalnych OMRON instrukcje arytmetyczne występują w postaci operatorów, które jednoznacznie określają wykonywaną operację. Instrukcje te mogą wykonywać działania na liczbach wprowadzonych w formacie binarnym lub BCD. Na rysunku 6.28 przedstawiono ogólną postać podstawowych instrukcji arytmetycznych.



Rys.6.28. Ogólna postać podstawowych instrukcji arytmetycznych.

Mnemonik instrukcji arytmetycznej, oprócz operatora, może posiadać dodatkowe oznaczenia literowe, które specyfikują typ danych i szczegółowo określają działanie instrukcji. Wśród tych oznaczeń literowych można wyróżnić [12]:

- **C** – instrukcja wykonuje operację arytmetyczną z uwzględnieniem bitu przeniesienia CY, np. +C, -C;
- **B** – obliczenia wykonywane są na liczbach w formacie BCD, np. +B, *B, ++B;
- **L** – operandy mają długość podwójnego słowa – 32 bity, np. ++L. –L, /L;
- **U** – operacje dzielenia i mnożenia wykonywane są na danych bez znaku (*Unsigned*), np. *U, /U;
- **F** – operacje na liczbach zmiennoprzecinkowych 32-bitowych (*Float*), np. +F, *F, /F;
- **D** – operacje na liczbach zmiennoprzecinkowych 64-bitowych (*Double*), np. +D, /D.

W tabeli 6.4 przedstawiono symbole oraz opis działania wybranych instrukcji arytmetycznych.

Tab.6.4. Wybrane instrukcje arytmetyczne.

Symbol instrukcji	Opis działania
+	Dodawanie liczb całkowitych ze znakiem bez uwzględnienia flagi CY
+C	Dodawanie liczb całkowitych ze znakiem z uwzględnieniem flagi CY
+L	Dodawanie liczb całkowitych ze znakiem podwójnej długości, bez uwzględnienia flagi CY
+CL	Dodawanie liczb całkowitych ze znakiem podwójnej długości, z uwzględnieniem flagi CY
+F	Dodawanie liczb zmiennoprzecinkowych o długości 32 bitów
+B	Dodawanie liczb całkowitych w formacie BCD bez uwzględnienia flagi CY
-	Odejmowanie liczb całkowitych ze znakiem bez uwzględnienia flagi CY
-F	Odejmowanie liczb zmiennoprzecinkowych
-C	Odejmowanie liczb całkowitych ze znakiem z uwzględnieniem flagi CY
*	Mnożenie liczb całkowitych ze znakiem
*U	Mnożenie liczb całkowitych bez znaku
*BL	Mnożenie liczb całkowitych w formacie BCD o podwójnej długości
/	Dzielenie liczb całkowitych ze znakiem
/D	Dzielenie liczb zmiennoprzecinkowych o podwójnej długości
/B	Dzielenie liczb całkowitych w formacie BCD
++	Inkrementacja liczby binarnej
++B	Inkrementacja liczby w formacie BCD
--	Dekrementacja liczby binarnej
--L	Dekrementacja liczby binarnej o podwójnej długości

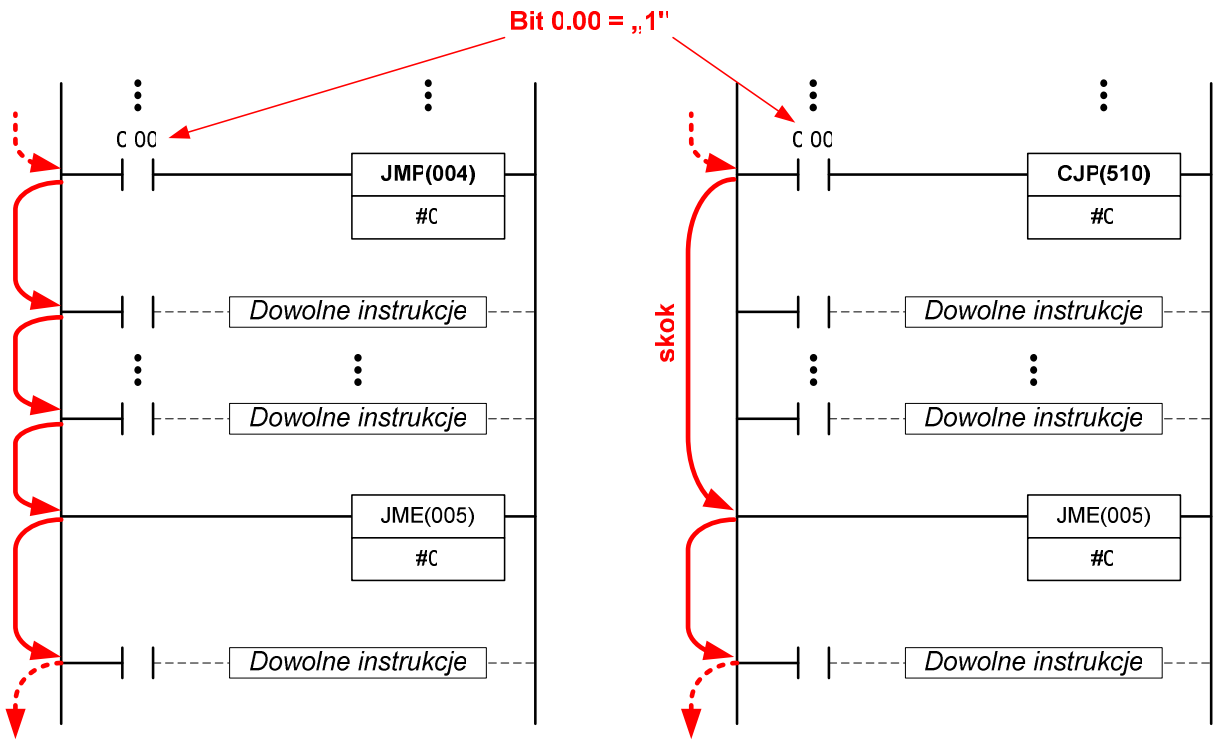
6.8. Instrukcje sterujące przebiegiem wykonania programu

6.8.1. INSTRUKCJE SKOKÓW

Instrukcja **JMP** występuje w parze zawsze z instrukcją **JME** i realizuje funkcję skoku bezwarunkowego do miejsca w programie, w którym jest umieszczona skojarzona z nią instrukcja JME. Obie instrukcje posiadają jeden operand, na który wprowadza się numer instrukcji skoku (dla obu ten sam) w zakresie 0–1023. Dodatkowo, instrukcja JMP posiada jedno wejście zezwolenia, które steruje wykonaniem skoku. W tym miejscu należy zaznaczyć, że działanie to jest nieco mylące, gdyż instrukcja JMP wykona skok jeśli warunek wykonywalności **nie będzie spełniony**. W przeciwnym wypadku, jeśli na wejściu sterującym pojawi się wysoki stan logiczny, instrukcja skoku będzie ignorowana i program wykona po kolei wszystkie instrukcje występujące po instrukcji JMP [4].

Instrukcja **CJP** realizuje funkcję skoku warunkowego i posiada identyczną postać wywołania jak instrukcja JMP. Różnica w działaniu tych instrukcji polega na tym, że w instrukcji CJP skok zostanie wykonany, jeżeli warunek wykonywalności będzie spełniony.

Na rysunku 6.29 przedstawiono sposób użycia oraz zobrazowano zasadę działania instrukcji JMP i CJP na programach przykładowych.



Rys.6.29. Zasada działania instrukcji JMP i CJP.

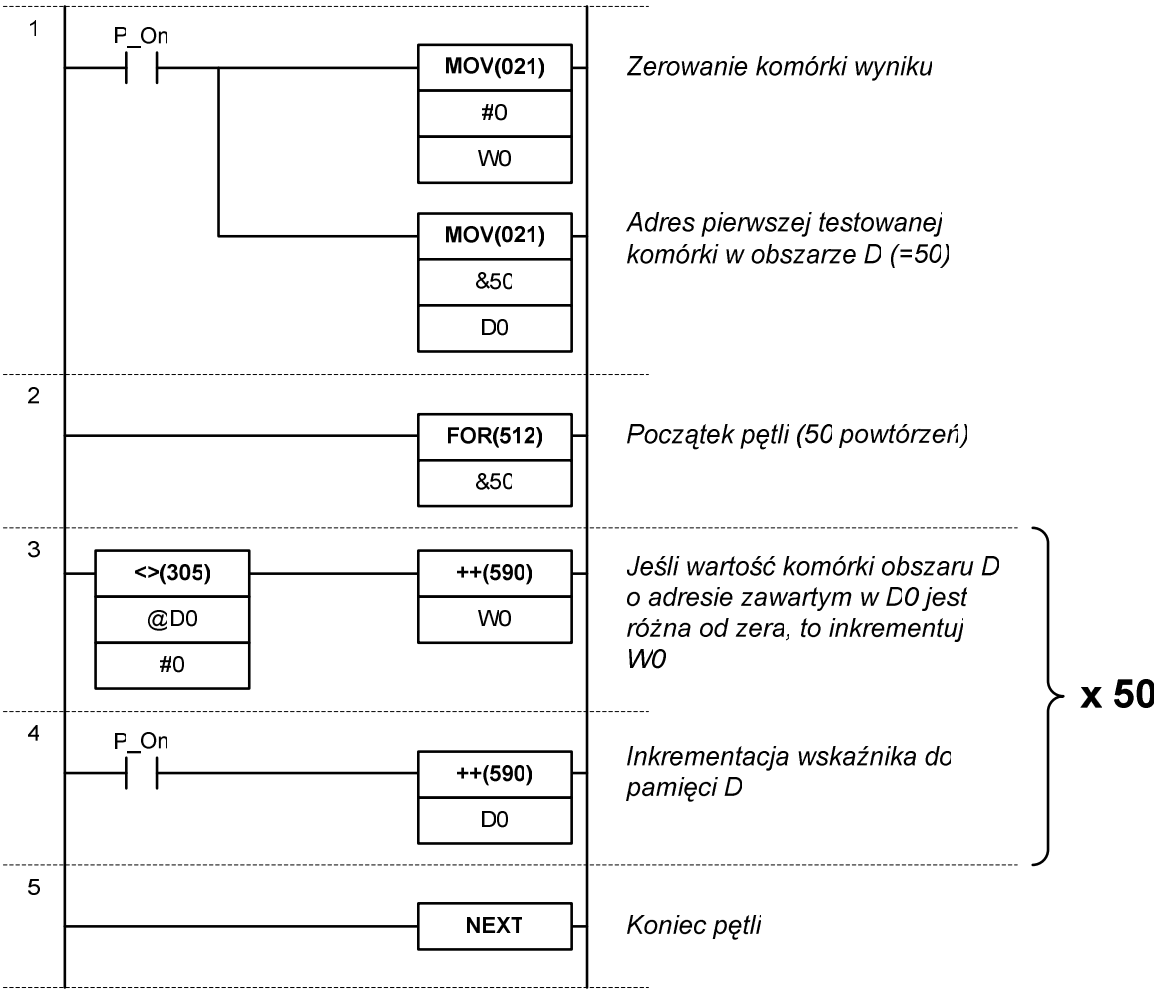
Jeżeli bit zezwolenia *0.00* będzie znajdował się w stanie wysokim, to w programie z lewej strony skok nie zostanie wykonany, w związku z tym wszystkie instrukcje występujące pomiędzy instrukcjami JMP i JME zostaną kolejno wykonane. Z kolei w programie znajdującym się po prawej stronie, skok warunkowy wywoływany instrukcją CJP zostanie wykonany, a instrukcje znajdujące się pomiędzy instrukcjami CJP i JME zostaną ominięte.

6.8.2. PĘTLA FOR-NEXT

Instrukcja **FOR** w połączeniu z instrukcją **NEXT** umożliwia realizację pętli programowych, które powtarzają wykonanie fragmentu programu zadeklarowaną liczbę razy pod rząd. Instrukcja FOR posiada jeden operand wejściowy, na który wprowadza się wartość liczbową z przedziału 0–65535, określającą liczbę powtórzeń pętli. Należy zauważyć, że instrukcja FOR jest wykonywana bezwarunkowo, co oznacza, że nie posiada ona wejścia zezwolenia i jest pierwszą a zarazem ostatnią instrukcją w linii programu.

Instrukcja **NEXT** jest bezparametrową instrukcją zamykającą powtarzany fragment programu. W przypadku, gdy występuje konieczność wcześniejszego zakończenia wykonywania pętli, można zastosować instrukcję **BREAK**, która spowoduje, że zamiast instrukcji następujących po niej zostaną wykonane instrukcje puste (NOP) i pętla zostanie zakończona.

Na rysunku 6.30 przedstawiono przykładowy program, który sprawdza ile komórek pamięci z obszaru D o adresach od 50 do 99 posiada niezerową wartość. W pierwszej linii programu następuje wyzerowanie komórki W0, która będzie zawierała wynik działania. W tej samej linii zostaje wpisana wartość początkowa 50 do komórki D0, będącej wskaźnikiem do przeszukiwanego obszaru pamięci (D50–D99). W drugiej linii programu znajduje się instrukcja FOR, która definiuje liczbę iteracji (50 cykli) i jednocześnie stanowi początek powtarzanego fragmentu programu. W trzeciej linii programu znajduje się instrukcja komparatora (<>), która sprawdza czy wartość komórki o adresie wskazywanym przez D0 jest różna od zera. Jeśli tak, wówczas zostaje wykonana instrukcja inkrementacji (++), która powiększa wartość komórki W0, stanowiącej końcowy wynik działania programu. Czwarta linia programu inkrementuje wartość wskaźnika D0, tak aby w kolejnej iteracji program sprawdzał następną komórkę z przeszukiwanego obszaru. Instrukcja NEXT powoduje powtórne wykonanie trzeciej i czwartej linii programu, i tak do momentu, aż zostanie wyzerowany licznik powtórzeń. Po wykonaniu pięćdziesięciu iteracji, program przechodzi do realizacji kolejnych linii programu, a w komórce W0 znajduje się wartość, która określa liczbę niezerowych komórek pamięci obszaru D w zakresie adresowym 50-99.



Rys.6.30. Przykładowy program wykorzystujący strukturę iteracyjną FOR-NEXT.

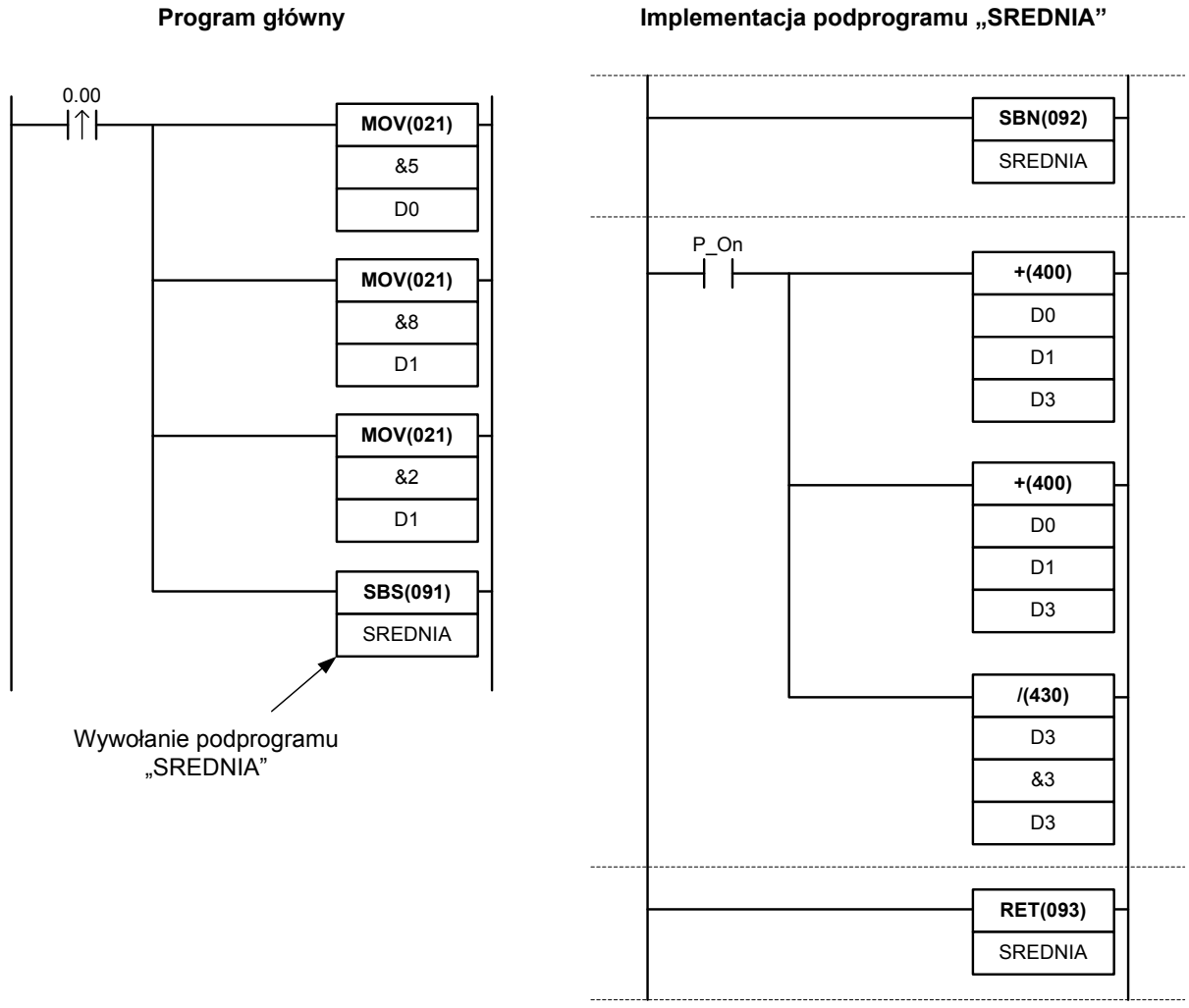
6.8.3. REALIZACJA PODPROGRAMÓW

Podprogramy są wydzielonymi fragmentami programu głównego, które zawierają zestaw instrukcji realizujących pewną funkcję, która może być wywoływana wielokrotnie, w obrębie działania programu głównego. W sterownikach OMRON serii CJ istnieje możliwość definiowania tzw. podprogramów globalnych, które są dostępne dla wszystkich programów, należących do jednego projektu.

Definicja podprogramu rozpoczyna się bezwarunkową instrukcją **SBN**, która posiada jeden operand wejściowy, oznaczający numer podprogramu (0–1023). W kolejnych liniach umieszcza się instrukcje realizujące zamierzoną funkcję podprogramu. Obszar podprogramu musi kończyć się bezargumentową instrukcję **RET**, która oznacza powrót do punktu, z którego wywołany był podprogram. W celu zachowania przejrzystości w programie głównym, implementacja podprogramów może być zrealizowana w osobnych sekcjach.

Wywołanie podprogramu polega na wykonaniu instrukcji **SBS**, której parametrem jest numer podprogramu. Po wykonaniu wszystkich instrukcji zawartych w podprogramie, następuje powrót do programu głównego i realizacja kolejnych jego instrukcji.

Na rysunku 6.31 przedstawiono przykładowy program, który wywołuje podprogram o nazwie *SREDNIA*. Podprogram ten oblicza wartość średnią z trzech komórek: D0, D1 i D2 i umieszcza wynik w komórce D3. Nazwa podprogramu *SREDNIA* została zdefiniowana jako symbol lokalny typu liczbowego (*Number*), do którego przypisano wartość 0 (numer podprogramu). Używanie nazw symbolicznych w podprogramach, znacznie ułatwia ich identyfikację i poprawia czytelność całego programu.



Rys.6.31. Przykładowy program wywołujący podprogram „SREDNIA” i jego implementacja.

Literatura

[1] BROCK S., MUSZYŃSKI R., URBAŃSKI K., ZAWIRSKI K., *Sterowniki Programowalne*, Wydawnictwo Politechniki Poznańskiej, Poznań 2000;
[2] BROEL-PLATER B., *Układy wykorzystujące sterowniki PLC. Projektowanie algorytmów sterowania*, PWN, Warszawa 2008;
[3] COLLINS D.A., LANE E.J., *Programmable controllers. A practical guide*, Woodlands. Church Hill, Glanmire, Co. Cork 1992;
[4] FLAGA S., *Programowanie sterowników PLC w języku drabinkowym*, BTC, Legionowo 2010;
[5] KASPRZYK J., *Programowanie sterowników przemysłowych*, WNT, Warszawa 2006;
[6] LEGIERSKI T., KASPRZYK J., WYRWAŁ J., HAJDA J., *Programowanie sterowników PLC*, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice 1998;
[7] MIKULCZYŃSKI T., SAMSONOWICZ Z., *Automatyzacja dyskretnych procesów produkcyjnych*, WNT, Warszawa 1997;
[8] MIKULCZYŃSKI T., *Automatyzacja procesów produkcyjnych*, WNT, Warszawa 2006;
[9] SAŁAT R., KORPYSZ K., OBSTAWSKI P., *Wstęp do programowania sterowników PLC*, WKŁ, Warszawa 2010;
[10] SETA Z., *Wprowadzenie do zagadnień sterowania. Wykorzystanie programowalnych sterowników logicznych*, MIKOM, Warszawa 2002;

Dokumentacje techniczne i materiały reklamowe:

[11] W393-E1-13 – *Sysmac CJ Series Programmable Controllers. Operational Manual*, OMRON 2008;
[12] W340-E1-15 – *Programmable Controllers. Instructions Reference Manual*, OMRON 2008;
[13] R135-E1-02 – *CX-ONE – Introduction Guide*, OMRON 2008;
[14] R132-E1-04 – *CX-Programmer – Introduction Guide*, OMRON 2008;
[15] W446-E1-06 – *CX-Programmer Ver. 8.1. Operational Manual*, OMRON 2008
[16] W345-E1-10 – *Analog I/O Units. Operational Manual*, OMRON 2008
[17] *http://industrial.omron.pl*;