# Zasada działania PLC.

# Struktura programu i organizacja pamięci.

**Dr inż. Patrycja Paduchowicz**

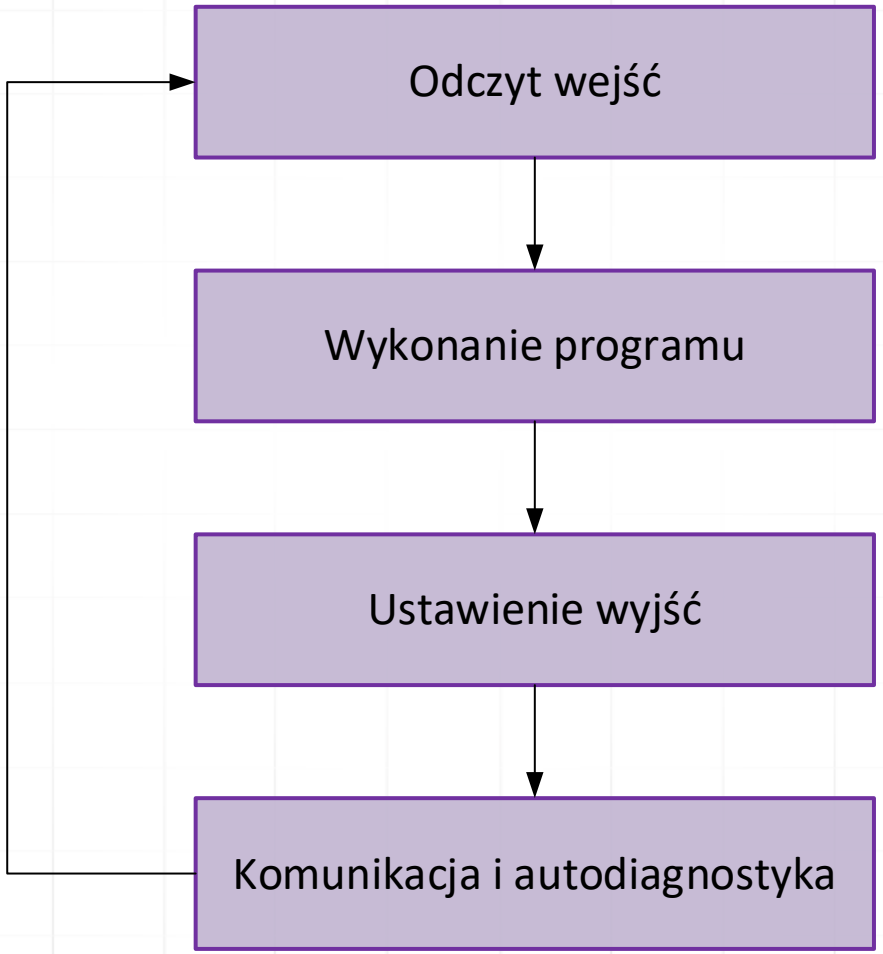
HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

# Zasada działania sterownika PLC

Sterownik PLC działa na zasadzie zero-jedynkowej, to znaczy włącza lub wyłącza poszczególne elementy wykonawcze maszyny czy urządzenia przy użyciu impulsów elektrycznych. Wyposażony jest w odpowiednią liczbę układów wejściowych i wyjściowych. Informacje o stanie obiektu oraz zadanej czynności, dostarczane są poprzez moduły wejściowe. Mikroprocesor, zaprogramowany w odpowiednim języku programowania, analizuje i zapisuje informacje (obrazy) uzyskane z układów wejściowych w sposób cykliczny. Sterownik weryfikuje polecenia w takiej kolejności, w jakiej zostały one zapisane w programie. Następnie tworzy obraz dla układów wyjściowych, a system operacyjny wyjścia steruje elementami wykonawczymi maszyny.

# Cykl pracy sterownika



# Cykl pracy sterownika

Inicjalizacja sterownika – przy każdym uruchomieniu sterownika następuje sprawdzenie poprawności działania.

Odczyt sygnałów wejściowych sterownika – pierwsza pętla programu w sterowniku analizuje i wykonuje odczyt stanów urządzeń wejściowych.

W trakcie wykonywania programu, na podstawie sygnałów wejściowych, zostają wygenerowane sygnały wyjściowe.

Obsługa komunikacji – odbiór informacji w przypadku podłączenia z innymi sterownikami oraz aktualizacja programu.

Autodiagnostyka – zbieranie informacji o błędach np. stanie baterii, zasilaniu itp. W przypadku wykrycia błędu sterownik przerywa działanie.

# Cykl pracy sterownika

**Przykład błędnego rozumienia:**

Zadanie:

1. Nalanie cieczy do zbiornika do poziomu max,
2. Mieszanie przez 20 sec.,
3. Opróżnienie zbiornika do poziomu min.,
4. Powrót do pkt.1 .

# Cykl pracy sterownika

**Przykład błędnego rozumienia:**

Program:

1. Instrukcja otwarcia zaworu nalewającego do momentu osiągnięcia poziomu max.,
2. Instrukcja włączenia mieszadełka na zadany czas,
3. Instrukcja otwarcia zaworu opróżniającego do osiągnięcia poziomu min.,
4. Powrót do pkt. 1.

# Cykl pracy sterownika

**Błąd:**

Wszystkie instrukcje programu wykonywane są „milion razy na sekundę”, a nie tak, jak niektórym się wydaje, po kolei (krok 2 realizowany po zakończeniu kroku 1).

**Poprawne rozwiązanie:**

W instrukcji nr 2 trzeba wprowadzić warunek, który blokuje jej wykonanie do momentu zakończenia wykonywania kroku nr 1. Analogicznie w kolejnych krokach.

# Zasady programowania

W celu zaprogramowania sterownika PLC koniecznym jest posiadanie informacji na temat elementów podłączanych na jego wejścia (przyciski, czujniki) oraz elementach na wyjściu (cewki przekaźników, styczników, lampki itp.).Dzięki tym informacjom możliwe jest:

- stworzenie tzw. listy przyporządkowania,
- napisanie programu sterowniczego,
- załadowanie programu sterowniczego do sterownika poprzez programator/ komputer,
- przetestowanie programu,
- uruchomienie i sprawdzenia układu rzeczywistego.

# Lista przyporządkowania

Lista umożliwia określenie:

- Czujników połączonych z wejściami oraz ich roli w sterowanym urządzeniu,
- Urządzeń połączonych z wyjściami oraz ich roli w sterowanym urządzeniu,
- Jakie funkcje wewnętrzne sterownika (markery, liczniki, moduły czasowe itp.) wykorzystano w programie sterowniczym.

Lista przyporządkowania może zawierać także krótki opis lub komentarz dotyczący warunków sterowania urządzenia, który ułatwi zrozumienie działania układu sterowniczego.

Przy tworzeniu listy przyporządkowań stosuje się tzw. operandy:

- Operandy absolutne (adresowanie absolutne) – są związane z adresami odpowiednio wejść/ wyjść sterownika, do których przyłączony jest dany element (czujnik, cewka elektrozaworu) np. I1, Q2 itd.
- Operandy symboliczne – są przypisywane adresom absolutnym elementów użytych w programie. Ich nazwy nadaje użytkownik np. S1 – oznaczenie przycisku, Y1 – oznaczenie cewki.

# Pisanie i testowanie programu

Aby napisać program sterujący, trzeba znać chociaż jeden z języków programowania (IL, FBD, LD..). Specjalne edytory lub oprogramowanie dedykowane przez producenta danego sterownika umożliwiają nie tylko wprowadzanie samego programu, ale także inne użyteczne funkcje, jak choćby sprawdzenie kodu pod względem składni, czy przesłanie go do sterownika PLC. Schematy stykowe zapisane w języku LAD, FBD pokazują jedynie stany logiczne na poszczególnych wejściach i wyjściach. Na ich podstawie nie można wnioskować jakie zestyki: zwierne czy rozwierne są przyłączone na wejścia, ponieważ sygnał 1 może wystąpić zarówno przy zamkniętym łączniku zwiernym jak i otwartym łączniku rozwiernym. Dlatego też, należy jednoznacznie określić rodzaje zestyków w liście przyporządkowań.

# Typy danych

| Binarne | | |
|---|---|---|
| Bool (bit) | False (0) | True (1) |

| Rejestry bitowe | | |
|---|---|---|
| Byte | 8 bitów | 0…255 |
| Word | 16 bitów | 0…65 535 |
| DWord | 32 bity | 0…4 294 967 295 |
| LWord* | 64 bity | 0…18 446 744 073 709 551 615 |

* Dostępne tylko w sterownikach Siemens serii S7-1500

# Typy danych

| Całkowite | | |
|---|---|---|
| Sint | 8 bitów | -128…+127 |
| INT | 16 bitów | -32 768 …+32 767 |
| DINT | 32 bity | -2 147 483 648…+2 147 483 647 |
| Usint | 8 bitów | 0…255 |
| UINT | 16 bitów | 0…65 535 |
| UDINT | 32 bity | 0…4 294 967 295 |
| LINT* | 64 bity | -9 223 372 036 854 775 808…<br>+9 223 372 036 854 775 807 |
| ULINT* | 64 bity | 0…18 446 744 073 709 551 615 |

* Dostępne tylko w sterownikach Siemens serii S7-1500

# Typy danych

| Rzeczywiste, zmiennoprzecinkowe | | |
|---|---|---|
| Real | 32 bity | $-3.402823e^{+38}$ ... $-1.175495e^{-38}$<br>$+1.175495e^{-38}$ ... $+3.402823e^{+38}$ |
| Lreal* | 64 bity | $-1.7976931348623158e^{+308}$...<br>$-2.2250738585072014e^{-308}$<br>$+2.2250738585072014e^{-308}$...<br>$+1.7976931348623158e^{+308}$ |

| Timery | | | |
|---|---|---|---|
| Time | 32 bity | ≈T#-24d...≈T#24d | P:T#1 h_10 min |
| Ltime* | 64 bity | ≈LT#-106 751d...≈LT#106 751d | P:LT#5s_12ns |

* Dostępne tylko w sterownikach Siemens serii S7-1500

# Typy danych

| Data i czas | | |
|---|---|---|
| Date | 2 bajty | D#1990-01-01…D#2169-06-06 |
| Time_of_Day (TOD) | 4 bajty | TOD#00:00:000 … TOD#23:59:59.999 |
| LTime_of_Day (LTOD)* | 8 bajtów | LTOD#00:00:00.000000000…<br>LTOD#23:59:59.999999999<br>P: LTOD#10:20:30.400 365 215 |
| Date_and_time (DT)* | 8 bajtów | DT#1990-01-01-00:00:00.000…<br>DT#2089-12-31-23:59:59.999<br>P: DT#2022-01-27-08:14:59.999 |
| DTL | 12 bajtów | DTL#1970-01-01-00:00:00.000000000…<br>DTL#2262-04-11-23:47:16.854775807<br>P: DTL#2019-03-09-20:30:20.250 |

* Dostępne tylko w sterownikach Siemens serii S7-1500

# Struktura DTL

| Bajt | Składnik | Typ danych | Zakres |
|---|---|---|---|
| 0 | Year | UINT | Od 1970 do 2262 |
| 1 | | | |
| 2 | Month | USINT | Od 1 do 12 |
| 3 | Day | USINT | Od 1 do 31 |
| 4 | Weekday | USINT | Od 1 (niedziela) do 7 (sobota) |
| 5 | Hour | USINT | Od 0 do 23 |
| 6 | Minute | USINT | Od 0 do 59 |
| 7 | Second | USINT | Od 0 do 59 |
| 8 | Nanosecond | UDINT | Od 0 do 999 999 999 |
| 9 | | | |
| 10 | | | |
| 11 | | | |

# Zmienne

Przez zmienne rozumie się takie dane, których wartości mogą zmieniać się w trakcie pracy sterownika, np. dane skojarzone z wejściami, wyjściami lub pamięcią sterownika. W deklaracji zmiennej podaje się typ do jakiego należy.

# Zmienne skalarne

Inaczej jednoelementowe, reprezentują pojedynczą daną. Zmienna taka może być reprezentowana symbolicznie (przez nazwę) lub w sposób bezpośrednio określający powiazanie ze strukturą wejść, wyjść lub pamięci sterownika. W tym drugim sposobie zmienna reprezentowana jest przez znak %, przedrostek określający jej lokalizację i ewentualnie rozmiar oraz jedną lub więcej liczb całkowitych bez znaku oddzielonych kropkami, określających adres zmiennej.

Przykłady:

| | |
|---|---|
| %QX75 lub %Q75 | bit 75 na wyjściu |
| %IW215 | słowo 215 na wejściu |
| %QB7 | bajt 7 na wyjściu |
| %MD48 | słowo podwójne pod adresem 48 w pamięci |

# Organizacja pamięci

Podstawowe typy pamięci:

M – pamięć ogólnego przeznaczenia,

I – pamięć wejść,

Q – pamięć wyjść.

Rozmiar rejestrów pamięci:

X lub brak - bit

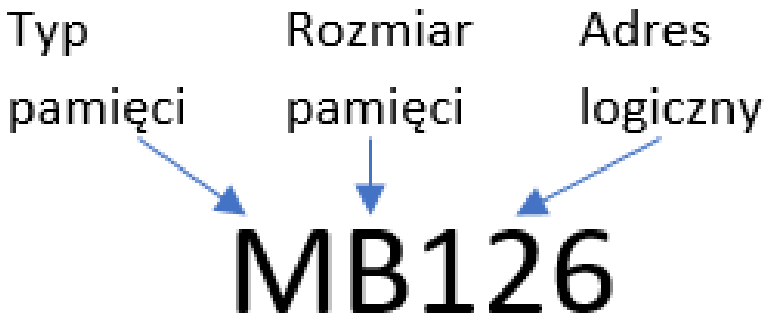
B – bajt (8 bitów),

W – word (słowo) – 16 bitów,

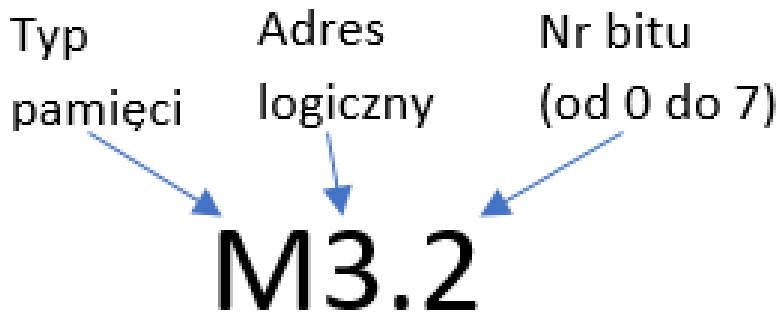
D – duble word (podwójne słowo) – 32 bity,

L – słowo poczwórne – 64 bity.

# Adresowanie logiczne




Adresowanie bitowe (w bajtach):



# Adresowanie logiczne

1. Pamięć o rozmiarze rejestru 8 bitów:

- Kolejne liczby naturalne: 0, 1, 2, 3 …

Np. MB0, MB1, MB2, MB3 …

2. Pamięć o rozmiarze rejestru 16 bitów:

- Kolejne liczby naturalne parzyste: 0, 2, 4, 6 …

Np. MW0, MW2, MW4, MW6 …

3. Pamięć o rozmiarze rejestru 32 bity:

- Kolejne liczby naturalne podzielne przez 4: 0, 4, 8, 12 …

Np. MD0, MD4, MD8, MD12 …

# Wzajemne zależności pomiędzy adresami rejestrów:

<table>
<tr><td>MB0</td><td rowspan="2">MW0</td><td rowspan="4">MD0</td></tr>
<tr><td>MB1</td></tr>
<tr><td>MB2</td><td rowspan="2">MW2</td></tr>
<tr><td>MB3</td></tr>
<tr><td>MB4</td><td rowspan="2">MW4</td><td rowspan="4">MD4</td></tr>
<tr><td>MB5</td></tr>
<tr><td>MB6</td><td rowspan="2">MW6</td></tr>
<tr><td>MB7</td></tr>
</table>

# Zmienne wieloelementowe

Czyli tablice i struktury. Tablica jest zbiorem elementów tego samego typu, do którego dostęp uzyskuje się za pomocą jednego lub więcej indeksów umieszczonych w nawiasach kwadratowych i oddzielonych przecinkami. Indeks musi być wyrażeniem, którego wartość odpowiada jednemu z typów całkowitych. Poniżej przedstawiono przykład adresowania elementu z siódmego wiersza i kolumny określonej przez wartość zmiennej %MB6 dla dwuwymiarowej tablicy INPUT_TAB

**INPUT_TAB[7,%MB6]**

# Zmienna strukturalna

Jest to zmienna deklarowana jako struktura zawierająca elementy różnych typów. Kolejne elementy reprezentowane są przez identyfikatory oddzielone kropkami. Poniżej przedstawiono przykład przyporządkowania wartości SINGLE_ENDED do elementu SIGNAL_TYPE oraz wartości BIPOLAR_10V do RANGE stanowiącego element piątego składnika elementu CHANNEL zmiennej o nazwie MODULE_5_CONFIG.

**MODULE_5_CONFIG.SIGNAL_TYPE:=SINGLE_ENDED;**

**MODULE_5_CONFIG.CHANNEL[5].RANGE:=BIPOLAR_10V.**

# Deklaracja zmiennych

Każdy element oprogramowania powinien na początku zawierać deklaracje używanych zmiennych zawierającą określenie typów do jakich należą i ewentualnie ich fizyczną lub logiczną lokalizację. Zakres działania takiej deklaracji jest lokalny w obrębie tego elementu oprogramowania, w którym się pojawia, tzn. że inne elementy oprogramowania nie mają dostępu do tych zmiennych. Co najwyżej mogą być one przekazywane jako argumenty wejściowe lub wyjściowe przy wywoływaniu tego elementu oprogramowania. Jedyny wyjątek stanowią tu zdefiniowane w konfiguracji zmienne globalne, do których dostęp w elemencie oprogramowania uzyskuje się za pomocą deklaracji VAR_EXTERNAL.

# Słowa kluczowe do deklaracji zmiennych

| Skrót | Składnik |
|---|---|
| VAR | Deklaracja zmiennych wewnętrznych w elemencie |
| VAR_INPUT | Deklaracja zmiennych dostarczanych z zewnątrz, nie mogą być zmieniane w elemencie oprogramowania |
| VAR_OUTPUT | Deklaracja zmiennych wyprowadzanych na zewnątrz do innych jednostek oprogramowania |
| VAR_IN_OUT | Deklaracja zmiennych dostarczanych z zewnątrz, mogą być zmieniane |
| VAR_EXTERNAL | Deklaracja użycia zmiennych definiowanych w konfiguracji jako VAR_GLOBAL, mogą być zmieniane |
| VAR_GLOBAL | Deklaracja zmiennych globalnych |
| VAR_ACCESS | Deklaracja ścieżek dostępu |
| RETAIN | Deklaracja zmiennej zachowywanej |
| CONSTANT | Deklaracja stałej |
| AT | Określenie lokalizacji zmiennej |

# Elementy oprogramowania

W normie IEC 61131-3 wyróżnia się następujące elementy organizacyjne oprogramowania:

- Funkcje,
- Bloki funkcyjne,
- Programy.

Elementy te mogą być dostarczane przez producentów lub programowane przez użytkownika.

# Funkcje

Do celów programowania sterowników PLC funkcję definiuje się jako element organizacyjny programu, który z chwilą wykonania dostarcza jeden element danych, chociaż może składać się on z wielu wartości, np. tablica. Funkcje nie zawierają wewnętrznej informacji o stanie (jest to więc „element statyczny”), tzn. wywołanie funkcji z tymi samymi argumentami (parametrami wejściowymi) zawsze daje tę samą wartość na wyjściu, w przeciwieństwie do bloku funkcyjnego.

Przykładami funkcji są : funkcje trygonometryczne, funkcje matematyczne (dodawanie, mnożenie, pierwiastek…)…

# Blok funkcyjny

W przeciwieństwie do funkcji z chwilą wykonania może dostarczać na wyjściu jedną lub wiele wartości. Blok posiada wewnętrzne zmienne zawierające pewną informację o stanie bloku. Wszystkie wartości zmiennych wyjściowych oraz koniecznie wartości zmiennych wewnętrznych są przechowywane pomiędzy kolejnymi chwilami wykonania bloku funkcyjnego, stąd wywołanie bloku z tymi samymi argumentami wejściowymi niekoniecznie musi prowadzić do tych samych wartości wyjściowych. W programie bloki funkcyjne mogą być wykorzystywane wielokrotnie, a każda taka „kopia” bloku musi zawierać swoją nazwę oraz strukturę danych zawierających wyjście i wewnętrzne zmienne bloku.

Przykładami bloków funkcyjnych mogą być : Przerzutniki RS i SR, Elementy detekcji zbocza, Liczniki itd…

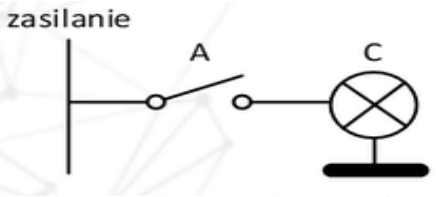
# Programy

Norma IEC61131 definiuje program jako „zbiór logicznie powiązanych elementów języka programowania koniecznych do zamierzonego przetwarzania sygnałów w celu sterowania urządzeniem lub procesem za pomocą systemu sterowników programowalnych”.

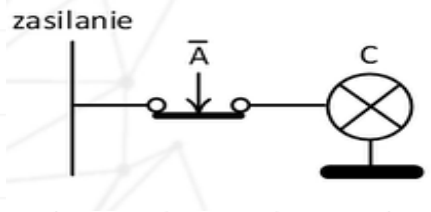
# Układy logiczne – elementy stykowe

styk normalnie otwarty

styk normalnie zamknięty

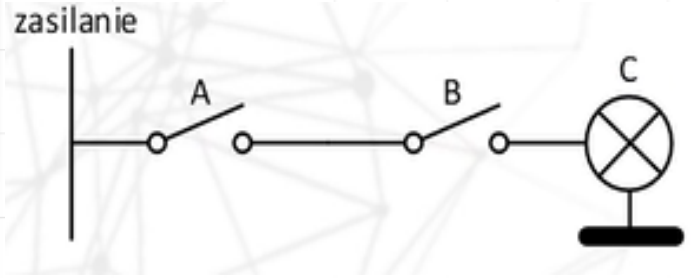



Tożsamość logiczna $C = A$




Negacja logiczna $C = \bar{A}$

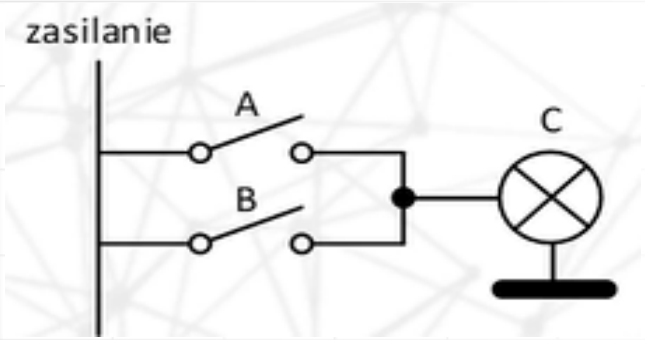
# Układy logiczne




Iloczyn logiczny

$C = A * B$

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

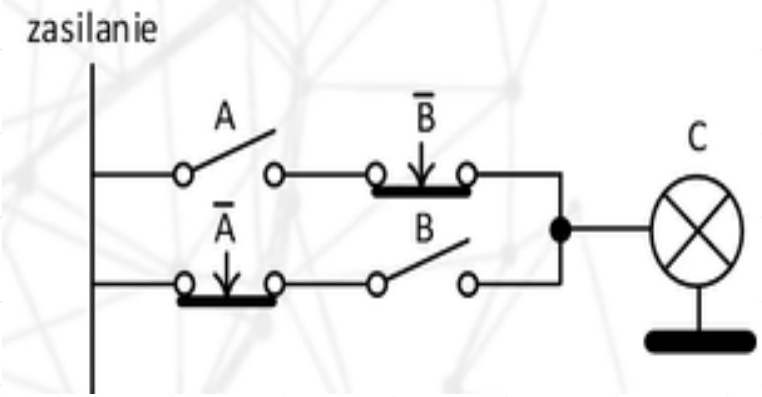



Suma logiczna

$C = A + B$

| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

# Układy logiczne

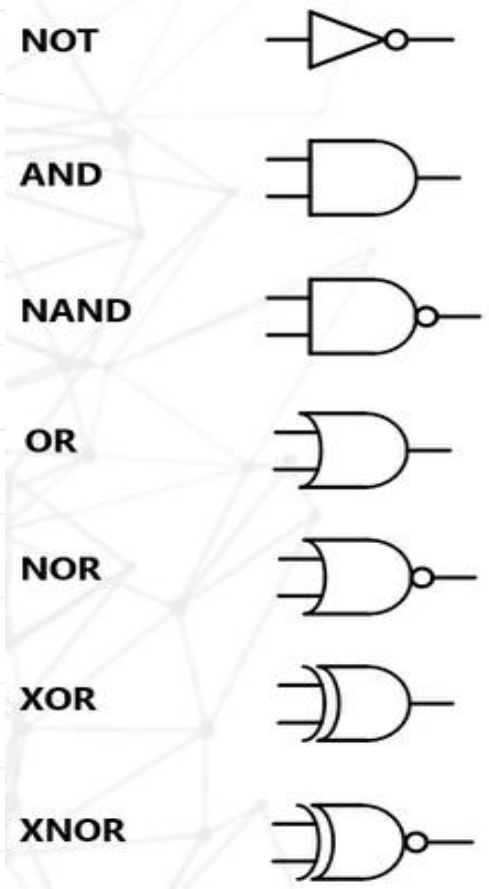



Suma logiczna specjalna

$C = A \oplus B = A * \bar{B} + \bar{A} * B$

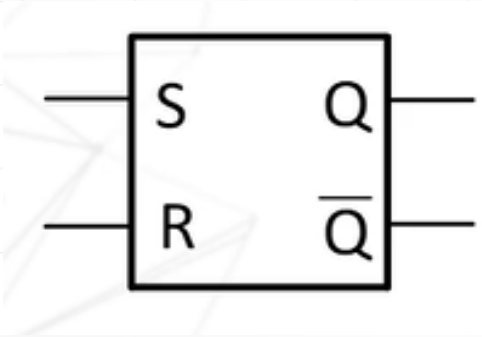
| A | B | C |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

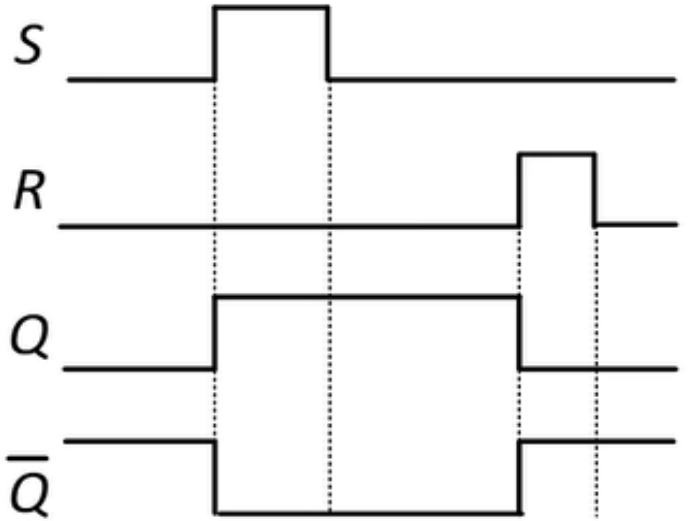
# Bramki logiczne




| A | B | C | | | | | |
|---|---|---|---|---|---|---|---|
| | | AND | NAND | OR | NOR | XOR | XNOR |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

# Przerzutnik




| $Q^t \rightarrow$ | $Q^{t+1}$ | S | R |
|---|---|---|---|
| 0 | 0 | 0 | - |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | - | 0 |

SR = 11 stan zabroniony

# Elementy konfiguracji

Służą do wspomagania instalowania i uruchamiania programów w systemach sterownikowych. Zalicza się do nich:

- Konfiguracje (configurations),
- Zasoby (Resources),
- Zadania (Tasks),
- Zmienne globalne (Global variables),
- Ścieżki dostępu (Access paths).

# Konfiguracja

Jest to element języka, który odpowiada systemowi sterowników programowalnych, rozumianemu jako całość obejmującą wszystkie pozostałe elementy oprogramowania.

Zasób z kolei jest odpowiednikiem funkcji przetwarzania sygnałów, łącznie z funkcjami określonymi przez podłączone czujniki i elementy wykonawcze oraz urządzenia operatorskie HMI.

Konfiguracja zawiera jeden lub więcej zasobów, z których każdy zawiera jeden lub więcej programów wykonywanych pod kontrolą zero lub więcej zadań. Program może zawierać zero lub więcej bloków funkcyjnych lub innych elementów języka zdefiniowanych w IEC 61131-3.

# Konfiguracja

Konfiguracje i zasoby mogą być uruchamiane i zatrzymywane za pomocą urządzeń operatorskich, funkcji systemu operacyjnego lub funkcji umożliwiających programowanie, testowanie i kontrolowanie. Uruchomienie konfiguracji powinno spowodować zainicjowanie zmiennych globalnych według zasad ustalonych dla fazy inicjacji, a następnie uruchomienie wszystkich należących do niej zasobów. Uruchomienie zasobu powinno z kolei spowodować inicjację wszystkich zmiennych, a następnie wszystkich zadań w zasobie.

# Konfiguracja

Faza inicjacji zmiennych może polegać na podtrzymaniu dotychczasowych wartości zmiennych, na nadaniu im wartości początkowych ustalonych przez programistę lub wartości domyślnych ustalonych dla poszczególnych typów danych.

Zatrzymanie zasobu powinno spowodować unieruchomienie wszystkich zadań w zasobie, a zatrzymanie konfiguracji powinno spowodować zatrzymanie wszystkich jej zasobów.

# Bibliografia


1. Tadeusz Legierski i inni, „Programowanie sterowników PLC”. Wydawnictwo Pracowni Komputerowej Jacka SKALMIERSKIEGO 2008.
2. https://www.ebmia.pl/wiedza/porady/automatyka-porady/sterownik-plc-budowa-rodzaje-zastosowanie/
3. http://www.zspgrot.edu.pl/www_plc/zasady-programowania.html
4. Tomasz Gilewski; „Szkoła programisty PLC Język LAD w programowaniu sterowników przemysłowych”. Helion 2018.
5. Tomasz Gilewski; „Szkoła programisty PLC Sterowniki przemysłowe”. Helion 2017.
6. Janusz Kwaśniewski; „Inteligentny dom i inne systemy sterowania w 100 przykładach”. Wydawnictwo BTC 2011.
7. Janusz Kwaśniewski; „Programowalny sterownik SIMATIC S7-300 w praktyce inżynierskiej”. Wydawnictwo BTC 2009.
8. http://plc-blog.pl

# Dziękuję za uwagę

Politechnika Wrocławska

Wydział Mechaniczny