2019

Podstawy sterowników programowalnych PLC

WYKORZYSTANIE PRZEMYSŁOWYCH URZĄDZEŃ MIKROPROCESOROWYCH W STEROWANIU PROSTYMI I ZŁOŻONYMI PROCESAMI MECHATRONICZNYMI

DR INŻ. ZBIGNIEW SETA

UKŁAD CYFROWY, MIKROPROCESOR, STEROWNIK CYFROWY, STEROWNIK PROGRAMOWALNY PLC, SYSTEM MECHATRONICZNY, PROGRAMOWANIE URZĄDZEŃ MIKROPROCESOROWYCH, MODUŁY STEROWNIKA PLC, KONFIGURACJA STEROWNIKA PLC, NORMA IEC 1131, METODY PROGRAMOWANIA STEROWNIKÓW PLC, APLIKACJA STEROWNIKA PLC W SYSTEMACH MECHAT.

W niniejszej publikacji podjęto się zadania przedstawienia zagadnień stosowalności w sterowaniu mechatronicznym nowoczesnych mikroprocesorowych urządzeń – sterowników programowalnych PLC (ang. Programmable Logic Controllers). Procesy sterowania, na których skupiono się przy wyjaśnieniu zasad wykorzystania tych urządzeń obejmują przypadki prostych oraz złożonych układów mechatronicznych. Do tych pierwszych zaliczymy sterowanie siłownikami pneumatycznymi, zaś do tych drugich sterowanie napędami elektrycznymi z wykorzystaniem przetwornic częstotliwości. Za wszelkie uwagi dotyczące prezentowanego materiału Autor niniejszej publikacji będzie bardzo wdzięczny. Ewentualne uwagi Czytelnik może kierować pod adres e-mail: seta@mchtr.pw.edu.pl.

Spis treści

PRZEDMOWA

W niniejszym opracowaniu autor podjął się zadania przedstawienia stosowalności w sterowaniu systemami mechatronicznymi nowoczesnych mikroprocesorowych urządzeń procesu produkcyjnego – **Sterowników Programowalnych PLC** (ang. *Programmable Logic Controllers* - w skrócie: sterowników PLC). Ponieważ z racji swoich parametrów technicznych oraz funkcjonalnych sterownik PLC wykorzystywany jest w przeważającej większości przypadków w sterowaniu systemami mechatronicznymi, w których układ sterowania operuje na sygnałach wejścia/wyjścia, będących sygnałami dyskretnymi (dwustanowymi, pochodzącymi ze zbioru {0,1}), przy omawianiu zagadnień wykorzystania sterownika PLC w takim sterowaniu skupiono się przede wszystkim na omówieniu przykładów sterowania procesami dyskretnymi.

Przykłady sterowania systemami mechatronicznymi, które posłużyły autorowi do wyjaśnienia roli oraz znaczenia wykorzystania w takim systemie sterownika PLC, objęły przypadki prostych jak i złożonych systemów mechatronicznych. Do tych pierwszych układów zaliczono np. sterowanie silnikiem elektrycznym w różnych konfiguracjach. Do tych drugich układów zaliczono np. sterowanie pracą szybowej windy towarowej oraz procesem mieszania materiałów sypkich.

Przy prezentowaniu materiału w zakresie aplikacji fizycznej sterowników PLC do sterowania przykładowymi systemami mechatronicznymi skoncentrowano się na zilustrowaniu Czytelnikowi najważniejszych aspektów wykorzystania tych urządzeń w takim sterowaniu, czyli:

- wkomponowaniu sterownika PLC w układ sterowania przykładowym systemem mechatronicznym;
- syntezie algorytmu procesu metodą modelowania określaną jako GRAFCET;
- syntezie algorytmu sterowania metodą modelowania określaną jako SFC;
- tworzeniu zapisu programu wykonywalnego (użytkowego) dla sterownika PLC przy użyciu metod programowania, określonych w normie IEC 131-3;
- podaniu uwarunkowań związanych z bezpieczeństwem funkcjonowania sterownika PLC w układzie sterowania przykładowymi systemami mechatronicznymi.

W związku z faktem, iż w systemach mechatronicznych funkcjonują nadal układy, określane mianem układów sterowania stykowego, (czyli układy, w których dany algorytm sterowania realizowany jest za pomocą sterowania przekaźnikami, stycznikami, przyciskami, itp. i ich łącznikami stykowymi - zestykami), wszędzie tam, gdzie było to konieczne i wskazane, sterowanie za pomocą sterownika PLC odniesiono do takiego właśnie układu stykowego. Uczyniono tak dla lepszego zrozumienia przez Czytelnika prezentowanego materiału zakładając, że w niektórych przypadkach pokazanie analogii dwóch rodzajów sterowań, które spotykane są w mechatronice, odniesionych do tego samego zadania sterowania systemem mechatronicznym może być korzystne dla Czytelnika. Założono przy tym, że Czytelnik dysponuje podstawową wiedzą na temat działania oraz konstrukcji układów sterowania stykowego.

Autor zaznacza, że zakres materiału, zilustrowany w niniejszej publikacji podzielono na cztery następujące moduły:

⇒ ***Wykorzystanie mikroprocesorów w sterowaniu. Podstawy sterowników programowalnych PLC*** – treść tego modułu ilustruje najważniejsze cechy mikroprocesora, który jest „sercem" każdego sterownika PLC, bez względu na jego rodzaj czy typ, oraz ilustruje sposób włączenia (wkomponowania) sterownika PLC do systemu sterowania mechatronicznego wraz z przedstawieniem podstaw syntezy algorytmu sterowania, która powinna być dokonana przed utworzeniem programu sterującego dla sterownika PLC;

⇒ ***Budowa sterowników programowalnych PLC. Podstawowe moduły sterownika PLC*** – treść tego modułu omawia rodzaje sterowników PLC z podziałem na urządzenia typu Compact oraz typu modułowego, oraz prezentuje rodzaje oraz parametry techniczne modułów, które wchodzą w skład konfiguracji każdego sterownika PLC;

⇒ ***Norma przemysłowa IEC 1131-3. Metody programowania sterowników PLC*** – treść tego modułu omawia trzecią część normy (europejska sygnatura to EN 61131-3), w której zawarte jest omówienie metod programowania sterowników PLC;

⇒ ***Uruchamianie oraz testowanie programu w sterowniku PLC*** – treść tego modułu omawia włączenie sterownika PLC jako serca układu sterowania do przykładowych prostych oraz złożonych systemów mechatronicznych, włączając zilustrowanie uruchomienia sterownika PLC oraz późniejsze testowanie programu użytkowego na sterowanym obiekcie.

Autor zaznacza również, że niniejsza publikacja uzupełniona została odpowiednio dobranymi materiałami multimedialnymi, w dużej ich liczbie, które są swoistego rodzaju mini-wykładami autora opracowania, i które będą dostępne dla Czytelnika.

Dodatkowo autor opracowania przygotował dla każdego modułu zestaw pytań kontrolnych oraz testów sprawdzających wraz z odpowiedziami. Powyższe powinno dać Czytelnikowi odpowiedź, w jakim stopniu przyswoił on sobie materiał niniejszej publikacji.

Na zakończenie autor podkreśli, że niniejszą publikację opracowano na podstawie materiału, będącego treścią licznych wykładów, jakie autor prowadził dla studentów uczelni technicznych. Odbiorcami niniejszej publikacji powinni być Czytelnicy zajmujący się zawodowo projektowaniem aplikacji sterujących w systemach mechatronicznych, które zawierają m.in. urządzenia mikroprocesorowe typu sterowniki PLC. Publikacja będzie również przydatna dla studentów wydziałów elektrycznych, informatycznych, mechatronicznych, itp., uczelni technicznych, czyli wszędzie tam, gdzie występuje kształcenie studentów o specjalnościach pokrewnych automatyce, mechatronice oraz sterowaniu procesami technologicznymi.

Za wszelkie uwagi dotyczące prezentowanego materiału autor będzie bardzo wdzięczny. Czytelnicy mogą je kierować pod adres e-mail: seta@mchtr.pw.edu.pl lub zbigniew.seta@pw.edu.pl.

1. Mikroprocesor jako serce cyfrowego układu sterowania

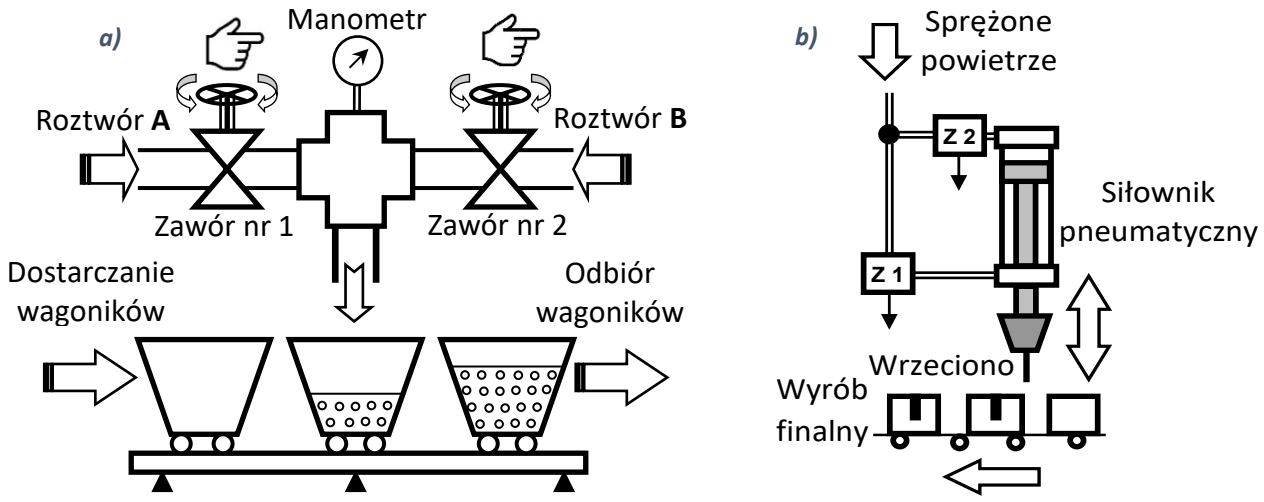
DEFINICJA

*[**System mechatroniczny** – struktura sterowania procesem produkcyjnym, złożona z podsystemu sensoryki, (zbierającego dane z procesu produkcyjnego), podsystemu sterowania, (realizującego algorytm sterowania procesem produkcyjnym), podsystemu wykonawczego, (realizującego sterowanie układami wykonawczymi procesu produkcyjnego) oraz podsystemu sygnałowego, (realizującego przesyłanie danych w obrębie wyżej wymienionych podsystemów)]*

DEFINICJA

*[**Proces produkcyjny** – zespół działań, których zadaniem jest przekształcenie półproduktów, surowców, itp. w wyroby gotowe, przeznaczone zazwyczaj na rynek wyrobów użytkowych]*

Nieustanny rozwój systemów mechatronicznych do sterowania procesami produkcyjnymi, podyktowany wprowadzaniem do procesu produkcyjnego coraz to nowocześniejszych wysokospecjalizowanych urządzeń procesowych typu wtryskarki, obrabiarki sterowane numerycznie czy urządzenia do montażu powierzchniowego elementów elektronicznych, itp., doprowadził do stopniowego wypierania starych układów sterowania, w których rola człowieka jako operatora procesu była dominująca, i wprowadzania nowszych rozwiązań układów sterowania, w których człowiek zaczął pełnić rolę obserwacyjną lub/i sprawozdawczą.
Rysunek 1 pokazuje przykład procesu produkcyjnego, w którym rola człowieka jako operatora była dominująca oraz dla kontrastu przykład procesu produkcyjnego, w którym człowiek pełni tylko rolę obserwatora.



Rysunek 1: Przykłady procesów produkcyjnych (wytwórczych): a) proces o dominującej roli człowieka jako operatora; b) proces, w którym człowiek pełni tylko rolę obserwatora

Wskazana tendencja modernizacji układów sterowania procesami produkcyjnymi zapoczątkowana została pojawieniem się w połowie XX wieku struktur, określanych jako cyfrowe układy sterowania – sterowników

cyfrowych, które bazowały na scalonych bipolarnych układach cyfrowych małej skali integracji **SSI** (ang. ***S**mall **S**cale of **I**ntegration*). Takie cyfrowe układy sterowania procesami mogły wykonywać podstawowe operacje arytmetyczne na danych wejściowych w postaci binarnej (wtedy najczęściej o długości nie przekraczającej 4 czy 8 bitów), które pochodziły z kontrolowanego procesu produkcyjnego.

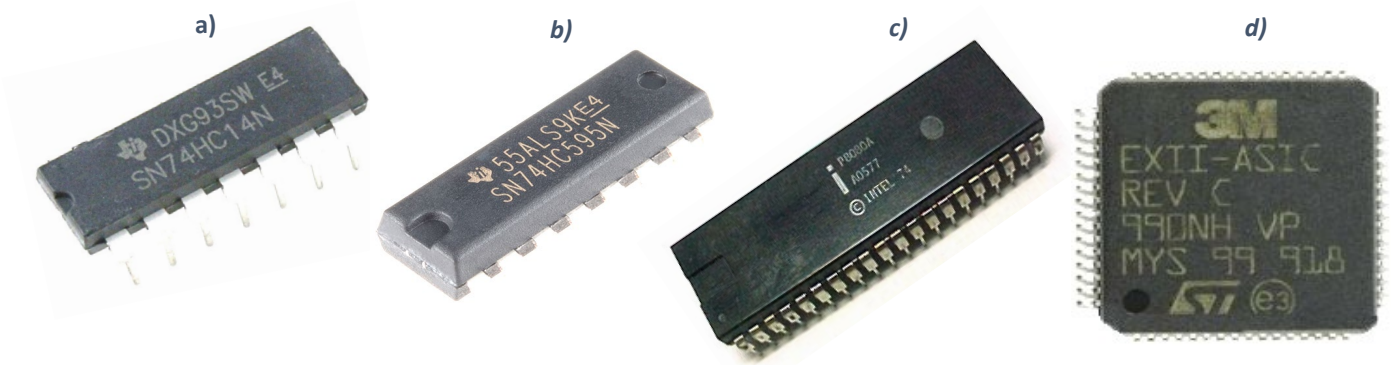
Przypomnijmy dla porządku, że ideą wprowadzania sterowników cyfrowych do układów sterowania różnymi procesami przemysłowymi było zastąpienie konwencjonalnego sterowania stykowego nowoczesną techniką cyfrową, z algorytmem sterowania zawartym w pamięci programu odpowiedniego układu cyfrowego, a nie „zapisanym" w postaci łączeniowej poprzez połączenie ze sobą w odpowiedniej konfiguracji cewek przekaźników (styczników), ich zestyków oraz przycisków sterujących. Czyli, w miejscu sieci połączeń elektrycznych różnych zestyków, łączników, cewek przekaźników i styczników, itp., wprowadzono pojedyncze urządzenie – sterownik cyfrowy, w którym cała taka struktura połączeń zawierała się w postaci odpowiedniego programu sterującego. Przez taką modyfikację układu sterowania dokonywała się w sposób stopniowy całkowita eliminacja fizycznego (czyli z użyciem przewodów, kabli, itp.) tworzenia programu sterowania w szafie sterowniczej na rzecz utworzenia algorytmu, który należało „załadować" do pamięci sterownika cyfrowego.

Takie nowe podejście w zastąpieniu konwencjonalnego sterowania stykowego sterowaniem cyfrowym musiało wywołać szereg problemów technicznych, które należało na bieżąco starać się rozwiązać. Jednym z nich był fakt, że sterownik cyfrowy jako urządzenie elektroniczne pracował „bezszumowo" w analogii do układów typu przekaźniki czy styczniki, które na skutek załączania się zestyków generowały dźwięki określane czasem przez praktyków terminem „szycie przekaźnika". Należało, zatem, zapoznać personel techniczny, obsługujący daną maszynę czy linię technologiczną z nowego typu diagnostyką takiego sterownika cyfrowego, która dokonywała się niejako samoistnie wewnątrz struktury sterownika na skutek odpowiednich wbudowanych w program sterujący procedur diagnostycznych. Dla takich załóg było to coś nowego, bowiem przy konwencjonalnym sterowaniu stykowym diagnostyka działania układu polegała najczęściej (oprócz obserwacji samego procesu) na kontroli załączania odpowiednich układów wykonawczych, które miały zadziałać w „odpowiedniej chwili".

Należy zaznaczyć, że mała skala integracji scalonych układów cyfrowych, które zaczęły znajdować zastosowanie w sterownikach cyfrowych, stanowiła poważne ograniczenie w rozwoju cyfrowych układów sterowania. Nie można było bowiem mówić o skomplikowanym algorytmie, który mógłby być realizowany przez taki układ sterowania czy mówić o obsłudze dużej liczby sygnałów wejścia/wyjścia, pochodzących z kontrolowanego procesu produkcyjnego. W pewnym sensie powyższe stało się możliwe dopiero w momencie pojawienia się scalonych układów cyfrowych o większej skali integracji, czyli:

- układów średniej skali integracji **MSI** (ang. ***M**edium **S**cale of **I**ntegration*);
- układów wielkiej skali integracji **LSI** (ang. ***L**arge **S**cale of **I**ntegration*);
- układów specjalizowanych do określonych zastosowań **ASIC** (ang. ***A**pplication in **S**pecific **I**ntegrated **C**ircuit*).

Rysunek 2 pokazuje przykłady cyfrowych układów scalonych, które należą do wyżej wymienionych grup różnej skali integracji wykonania tych układów.



Rysunek 2: a) układ małej skali integracji SSI – funktory logiczne; b) układ średniej skali integracji MSI – rejestr przesuwny; c) układ wielkiej skali integracji LSI – mikroprocesor 8-mio bitowy 8080 firmy INTEL; d) układ specjalizowany ASIC

DEFINICJA

[Skala integracji układu scalonego - liczba „upakowanych" bramek logicznych na 1cm^2 powierzchni tego układu]

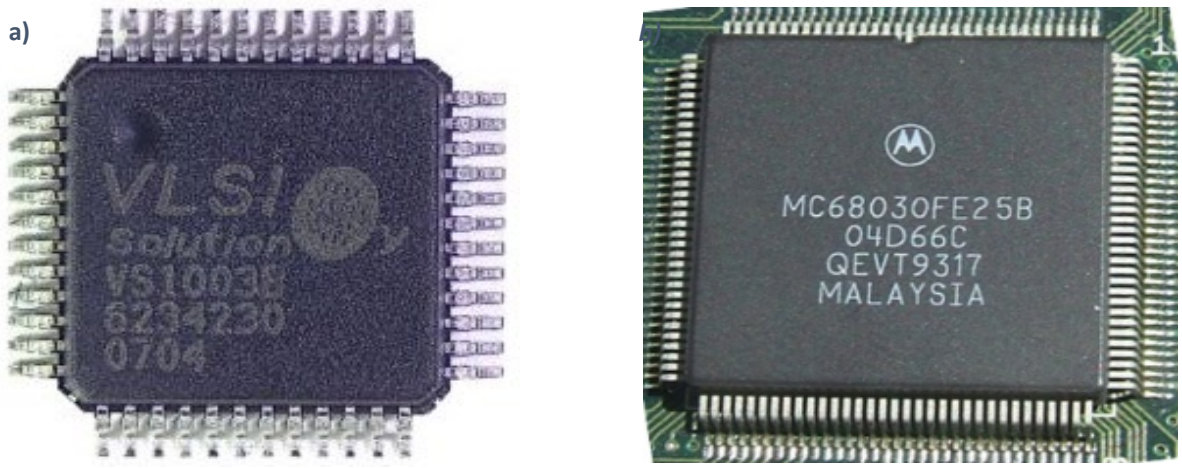
Pomimo zwiększających się odpowiadająco w miarę rozwoju technologii cech funkcjonalnych układów cyfrowych różnej skali integracji, zilustrowanych na rysunku 2, takich cech jak wykonywanie operacji logicznych na coraz większej liczbie danych cyfrowych oraz coraz większa szybkość przetwarzania pojedynczej informacji cyfrowej, które to cechy spowodowały jak wiadomo użycie tych układów jako „serca" sterowników cyfrowych, te ostatnie posiadały jednak szereg wad, które musiały być znacząco wyeliminowane w przyszłości. Do tych wad zaliczono m.in.:

- „sztywną" strukturę fizyczną sterownika cyfrowego bez możliwości szybkiej jego sprzętowej rekonfiguracji w zależności od potrzeb, zwłaszcza przy modernizacji układu sterowania, wymuszonej przez unowocześnianiu linii technologicznych;
- stały, wbudowany algorytm pracy sterownika cyfrowego jako serca układu sterowania, algorytm ustalony przez producenta (projektanta) urządzenia przed jego zamontowaniem w cyfrowym układzie sterowania, bez łatwej możliwości zmiany tego algorytmu lub jego modernizacji w trakcie pracy sterownika cyfrowego;
- brak komunikacji między sterownikami cyfrowymi w systemie sterowania określanym jako Nadrzędny – Podrzędny **MS** (ang. ***M**aster - **S**lave*), uniemożliwiający łączenie pojedynczych cyfrowych układów sterowania w większe, warstwowe rozproszone systemy sterowania procesami.

Dalszy rozwój w 70 – tych XX. wieku mikroprocesorów zlikwidował wreszcie bariery, które blokowały rozwój dużych i nowoczesnych cyfrowych systemów sterowania, opartych na możliwości wzajemnej komunikacji między sterownikami cyfrowymi. Pojawiły się na rynku scalonych układów cyfrowych nowe następujące grupy wykonania mikroprocesorów w oparciu, o które zaczęto budować już sterowniki cyfrowe:

- układy wielkiej skali integracji **VLSI** (ang. ***V**ery **L**arge **S**cale of **I**ntegration*)
- układy ultra-wielkiej skali integracji **ULSI** (ang. ***U**ltra **L**arge **S**cale of **I**ntegration*).

Rysunek 3 pokazuje przykłady mikroprocesorów, które należą do dwóch ww. grup technologii wykonania cyfrowych układów scalonych.



Rysunek 3: a) układ wielkiej skali integracji VLSI - mikroprocesor;
b) układ ultra-wielkiej skali integracji ULSI - mikroprocesor

Mikroprocesor z rodziny scalonych układów cyfrowych, zilustrowanych na rysunku 3, użyty jako serce sterownika cyfrowego, (ale jeszcze nie sterownika PLC), umożliwiał już obsługę dużej liczby sygnałów wejścia/wyjścia procesu produkcyjnego, co było niezbędne przy propagowaniu układów sterowania rozproszonego. Z kolei znaczne zwiększenie szybkości realizacji w takim mikroprocesorze pojedynczej operacji logicznej na określonych danych, umożliwiło pojawienie się tzw. systemów czasu rzeczywistego **RTS** (ang. ***R**eal **T**ime **S**ystem*), które mogły zagwarantować zwiększenie reakcji cyfrowego układu sterowania danym procesem produkcyjnym na zaistniałe zdarzenie procesowe i następnie szybko dostosować do tego zdarzenia określone sterowanie, wypracowane przez sterownik cyfrowy.

2. Architektura mikroprocesora jako bazy dla sterownika cyfrowego

DEFINICJA

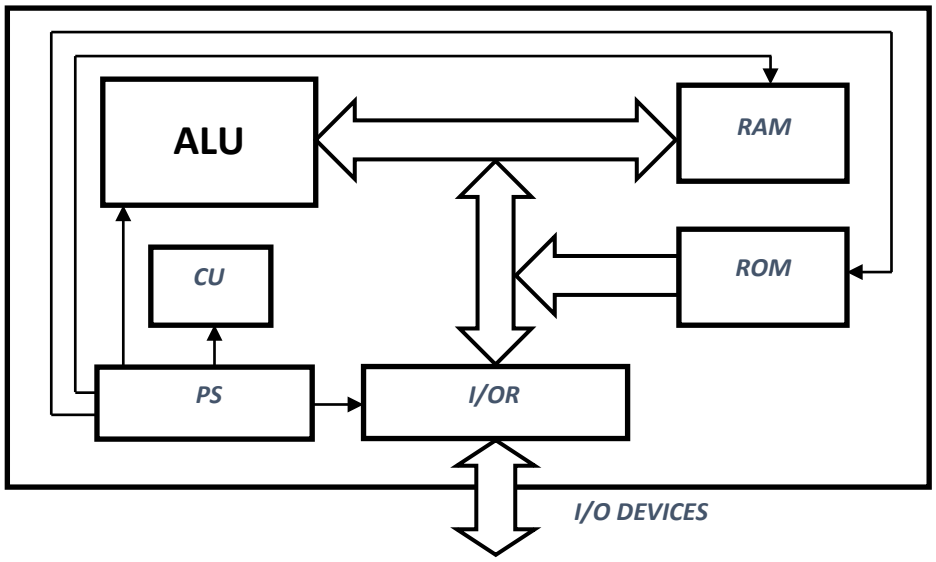
[Mikroprocesor - to zaawansowany technologicznie oraz konstrukcyjnie układ cyfrowy, który zdolny jest do wykonywania operacji logicznych na danych cyfrowych według dostarczonego mu ciągu instrukcji zapisanych w kodzie binarnym]

Z powyższej definicji wynika w związku z nakreślonymi zadaniami dla mikroprocesora, że struktura pojedynczego mikroprocesora, (czyli cyfrowego układu scalonego, zawartego w pojedynczej obudowie – rys.3), jak również struktura układu mikroprocesora, (czyli zespołu układów cyfrowych, rozmieszczonych na jednej lub kilku tzw. płytkach drukowanych, złożonego m.in. z pojedynczego mikroprocesora i układów z nim „współpracujących”), muszą posiadać następujące elementy funkcjonalne m.in.:

- jednostkę arytmetyczno-logiczną **ALU** (ang. ***A**rithmetic **L**ogic **U**nit*), która na dostarczonych danych wykonuje operacje logiczne, takie jak dodawanie, operacje logiczne typu AND, XOR, OR i NOT, oraz przesunięcia bitów w lewo i w prawo;

- układ sterowania **CU** (ang. ***C**ontrol **U**nit*), który odpowiedzialny jest za dekodowanie ciągu instrukcji, dostarczonych mikroprocesorowi w postaci kodu binarnego i odpowiednie sterowanie pozostałymi blokami mikroprocesora;
- pamięć stałą **ROM** (EPROM, E^2PROM) (ang. ***R**ead **O**nly **M**emory*), z której zapisane dane można tylko odczytywać lub zapisywać, ale wyłącznie przy użyciu specjalnych urządzeń programujących; w tego rodzaju pamięci stałej przechowywane są dane, które nie powinny być utracone po odłączeniu zasilania mikroprocesora;
- pamięć swobodnie programowalną **RAM** (ang. ***R**andom **A**cces **M**emory*), w której możliwy jest wielokrotny i łatwy zapis danych, które stanowią m.in. wyniki realizowanych przez ALU obliczeń i dane tymczasowe, które mogą być wykorzystywane wielokrotnie; w tym rodzaju pamięci przechowywane są utracone po odłączeniu zasilania mikroprocesora, chyba że zastosowano podtrzymanie bateryjne zasilania pamięci;
- rejestry wejścia/wyjścia **I/OR** (ang. ***I**nput **O**utput **R**egisters*), czyli komórki pamięci o organizacji Bajtu, słowa lub dwu-słowa, (czyli odpowiednio ośmiu, szesnastu i trzydziestu dwu bitów), które umieszczone są wewnątrz struktury mikroprocesora; zadaniem tych rejestrów jest przechowywanie danych wsadowych oraz wyników obliczeń, które mogą być zamienione na działania mikroprocesora w stosunku do układów zewnętrznych;
- zasilacza **PS** (ang. ***P**ower **S**ource*), którego zadaniem jest dostarczenie dla infrastruktury mikroprocesora napięć zasilających poszczególne moduły tego układu.

Rysunek 4 ilustruje schemat blokowy struktury układu mikroprocesora, zawierającej kluczowe jego elementy.



Rysunek 4: Schemat blokowy struktury układu mikroprocesora: ALU – jednostka arytmetyczno-logiczna; CU – układ sterowania; RAM – pamięć ulotna; ROM – pamięć nieulotna; PS – zasilacz układu mikroprocesora; I/OR – rejestry we/wy

Rozwój technologiczny układów cyfrowych wielkiej skali integracji o strukturze pokazanej na rysunku 4 doprowadził w konsekwencji do pojawienia się w końcu XX w. maszyny cyfrowej, czyli komputera rozumianego w sensie współczesnym, czyli komputera typu **PC** (ang. ***P**ersonal **C**omputer*).

DEFINICJA

[***Komputer*** *- maszyna elektroniczna zawierająca mikroprocesor lub kilka mikroprocesorów, która realizując odpowiedni algorytm, przetwarza informacje, wprowadzane zazwyczaj za pomocą tzw. klawiatury komputerowej, które można zapisać w formie ciągu cyfr a efekt pracy komputera obserwować na monitorze ekranowym*]

Pomijając naznaczoną pierwotnie rolę dla komputera PC, czyli używanie go do przetwarzania informacji wprowadzanych za pomocą tzw. klawiatury komputerowej lub wcześniej, za pomocą innych urządzeń wejściowych komputera, np. czytników dla taśmy perforowanej, by następnie wyniki obserwować na ekranie monitora, zaczęto rozważać użycie takiego komputera do sterowania procesami produkcyjnymi. Oczywiście komputera typu PC zbudowanego od podstaw lub istniejącego, ale poddanego gruntownej modernizacji, aby parametry techniczne tegoż odpowiadały trudnym warunkom, które istnieją w środowisku przemysłowym. Przykładem takich trudnych warunków mogą być np. występujące tam wibracje czy wysoka temperatura, znacznie wyższa od temperatury otoczenia, czyli temperatury pracy przeciętnego komputera. Przeciętny komputer typu PC mógłby nie „sprawdzić" się w takim środowisku przemysłowym. O tego momentu możemy zatem już mówić o pojawieniu się tzw. sterownika cyfrowego, ale jeszcze nie sterownika PLC.

DEFINICJA

[***Sterownik cyfrowy*** *– urządzenie elektroniczne zawierające mikroprocesor lub kilka mikroprocesorów, które przeznaczone jest do realizacji algorytmu sterowania procesem produkcyjnym, i które operuje na danych wejścia/wyjścia pochodzących i przeznaczonych dla tego procesu produkcyjnego*]

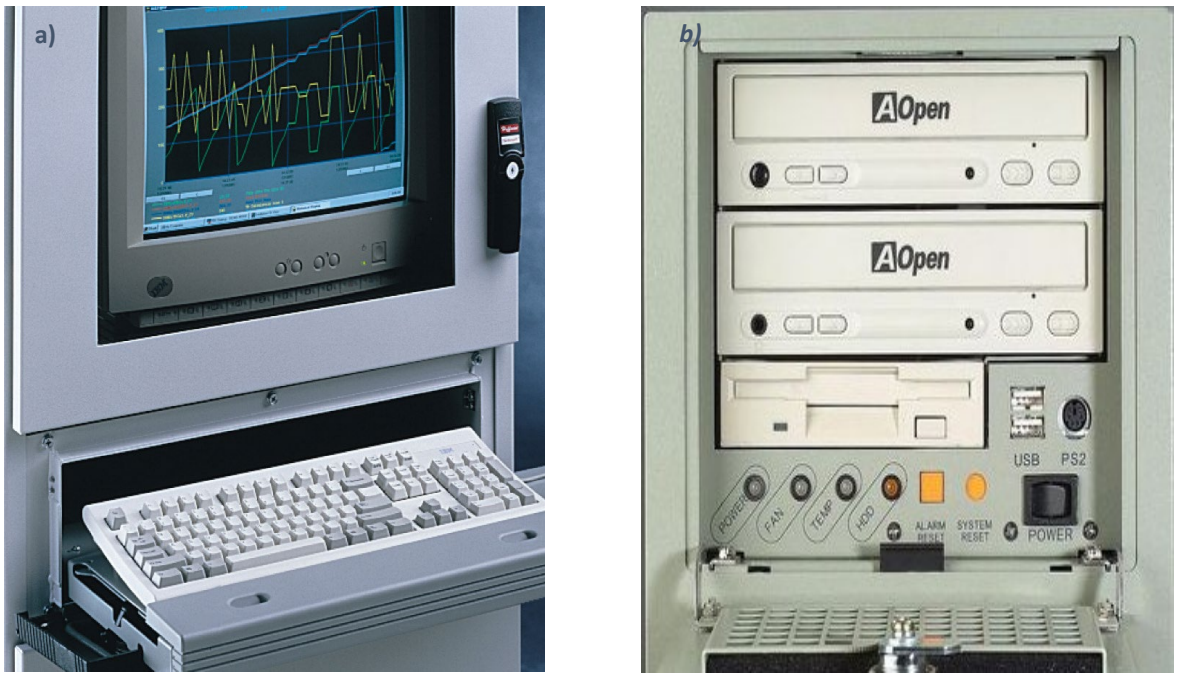
Budowa sterownika cyfrowego od podstaw, który dedykowany był dla konkretnego procesu przemysłowego polegała m.in. na „zamknięciu" części elektronicznej o architekturze komputera PC w odpowiedniej obudowie przemysłowej, która mogła sprostać wysokiemu reżimowi jej zastosowania w dedykowanym procesie przemysłowym, oraz na użyciu podzespołów elektronicznych odpornych na wyższe temperatury. Oprócz tego celem współpracy takiego sterownika cyfrowego z kontrolowanym procesem oraz operatorem tegoż, obudowa urządzenia posiadała odpowiednie gniazda elektryczne, które służyły do podłączenia wiązek przewodów sygnałowych, które obsługiwały elementy wejścia/wyjścia procesu, oraz opcjonalnie mogła posiadać tzw. klawiaturę przemysłową, odporną na działanie warunków przemysłowych.

Odmienne podejście projektantów urządzeń obowiązywało przy dostosowywaniu istniejących na rynku komputerów typu PC do warunków procesu przemysłowego, w którym to procesie zazwyczaj taki komputer typu PC nie znajdował się w pobliżu np. kontrolowanej pracującej maszyny lub linii technologicznej, tak jak to

robił sterownik cyfrowy wyżej opisany, tylko znajdował się w pewnym oddaleniu od takiej maszyny czy linii, czyli np. znajdował się w pomieszczeniu maszynowni, itp., czyli w warunkach zbliżonych do warunków otoczenia. Przy takim podejściu skupiano się przede wszystkim na wyposażeniu istniejącego komputera PC w odpowiednie karty wejścia/wyjścia, umieszczane w tzw. slotach komputera, które umożliwiały podłączenie do tegoż przewodów sygnałowych o roli i znaczeniu jak w poprzednio omówionym sterowniku cyfrowym. Do współpracy takiego komputera z operatorem służyła zazwyczaj tradycyjna klawiatura oraz monitor ekranowy.

Rysunek 5 ilustruje dwa omówione wyżej typy sterowników cyfrowych, dedykowane do kontrolowania procesów przemysłowych, powstałe na bazie opracowanej w końcu XX w. architektury komputera PC i które to rozwiązania przetrwały do dnia dzisiejszego i są w niektórych aplikacjach przemysłowych dalej stosowane.

Rysunek 5a) ilustruje konstrukcję sterownika cyfrowego do sterowania procesem przemysłowym na przykładzie zabudowy w szafie sterowniczej poszczególnych komponentów komputera typu PC (widocznych), takich jak monitor ekranowy oraz klawiatura, która umieszczona została na odpowiedniego rodzaju wysuwanych prowadnicach. Tego typu obudowy dla sterownika cyfrowego charakteryzują się odpornością m.in. na duże wibracje, które mogą pochodzić od pracującej maszyny lub linii, w pobliżu której sterownik jest umieszczony. Dodatkowo operator procesu posiada możliwość kontroli optycznej na monitorze ekranowym wybranych parametrów procesu oraz dokonywania zmiany nastaw przy użyciu widocznej klawiatury komputerowej.



Rysunek 5: Dwa typy sterowników cyfrowych opracowane na bazie architektury komputera typu PC: a) komputer typu PC w szafie sterowniczej firmy Schroff; b) komputer typu PC firmy Advantech

Rysunek 5b) ilustruje z kolei drugi przypadek użycia architektury komputera typu PC do budowy sterownika cyfrowego dla procesów przemysłowych. Zaprezentowana obudowa jest obudową typu Tower, czyli rodzajem obudowy, w której montowane są konwencjonalne komputery typu PC. Różnica sprowadza się do jej wzmocnienia oraz doposażenia zestawu od dodatkowe elementy do komunikacji z procesem przemysłowym.

3. Powstanie sterownika programowalnego PLC

DEFINICJA

[***Sterownik programowalny PLC*** *(ang.* ***P****rogrammable* ***L****ogic* ***C****ontroller) – to urządzenie mikroprocesorowe, które w swojej architekturze logicznej oprócz mikroprocesora zawiera moduły wejścia/wyjścia dla sygnałów procesowych, i który po zaprogramowaniu go przy użyciu metod objętych normą IEC 1131-3 realizuje program sterujący procesem mechatronicznym w sposób cykliczny*]

Wykorzystywanie sterowników cyfrowych, opracowanych na bazie rozwiązań z rysunku 5 do sterowania procesami produkcyjnymi było naturalną konsekwencją wypierania z układów sterowania procesami konwencjonalnego sterowania stykowego oraz konsekwencją przenikania nowych technologii rozwoju urządzeń cyfrowych, które najpierw opracowane zostały do działań użytkowych, zwykle biurowych, statystycznych, itp., by potem zostać zaadaptowane do szeroko rozumianego przemysłu. Tak jest do chwili obecnej. Opracowana nowa rodzina mikroprocesorów znajduje najpierw zastosowanie w aplikacjach, w których należy przetwarzać olbrzymią ilość danych, zazwyczaj o postaci sygnału analogowego, a potem znajduje zastosowanie w innych aplikacjach, np. w sterownikach cyfrowych, a na jej miejsce pojawiają się nowsze już odmiany mikroprocesorów.

Burzliwy rozwój rodzin mikroprocesorów różnych firm, notowany w drugiej połowie XX wieku spowodował, że powstałe dzięki temu sterowniki cyfrowe (jeszcze nie sterowniki PLC) nie można było objąć określonym wspólnym standardem ich wykonania oraz programowania. Wytworzone sterowniki cyfrowe dla sterowania procesami przemysłowymi (już wtedy zaczęło pojawiać się pojęcie systemu mechatronicznego) były skonstruowane w sposób indywidualny pod określoną aplikację według zamówienia właściciela zakładu przemysłowego lub według własnego, indywidulanego spojrzenia producenta sterownika na rozwiązanie problemu sterownia linią produkcyjną, maszyną przemysłową, procesem, itp. Do uruchomienia, serwisowania, konserwacji, itp. takich urządzeń przewidziany był zazwyczaj wyłącznie producent tychże, czyli firma z branży automatyki, która była producentem sterowników cyfrowych. Personel techniczny, nawet wyższego szczebla danego zakładu przemysłowego, w którym określony sterownik znalazł zastosowanie, nie mógł dokonywać napraw czy modernizacji takiego sterownika, gdyż po prostu nie posiadał ku temu zaawansowanej wiedzy, która musiała obejmować informatykę czy konstrukcję urządzeń elektronicznych. Z kolei brak modułowości takich sterowników cyfrowych, rozumiany jako urządzenia zbudowane z wymiennych, współpracujących ze sobą modułów urządzenia, prawie wykluczał wymianę jakiegoś uszkodzonego pojedynczego elementu, tylko należało wymieniać całe urządzenie sterujące, co było nie tylko pracochłonne, ale także i kosztowne. Pojawiająca się potrzeba wykorzystania sterowników cyfrowych w coraz większej liczbie aplikacji, co było związane z rozwojem przemysłu różnych branż spędzała sen z powiek producentom sterowników cyfrowych, którzy przy istniejącym wtedy podejściu w budowie sterowników cyfrowych, czyli budowie tych urządzeń bez standaryzacji, (o tym była

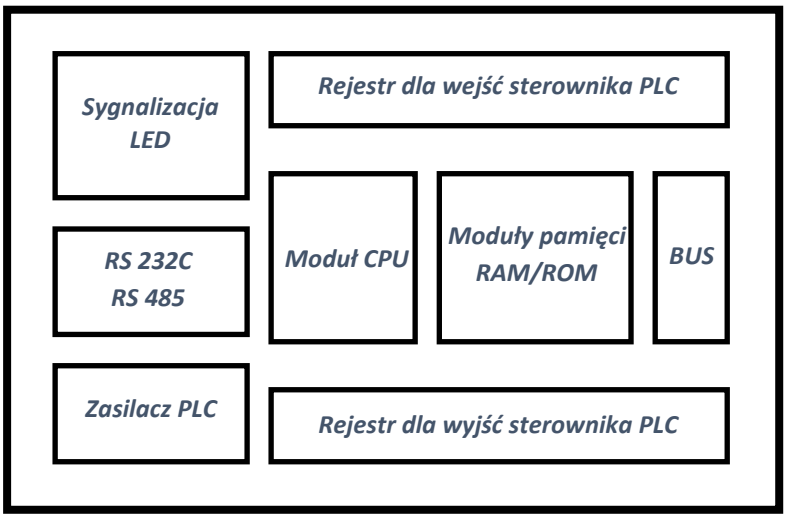
mowa wcześniej), nie byli w stanie zaoferować kompleksowych usług, czyli budowa sterownika cyfrowego plus jego uruchomienie plus konserwacja plus uaktualnienie algorytmu sterowania, itp., na najwyższym poziomie, gdyż ich zespoły uruchomieniowe nie były tak liczne aby sprostać wszystkim tego typu zamówieniem. Powyższe zaowocowało pojawieniem się trendu do ujednolicenia budowy sterownika cyfrowego tak, aby sposób jego aplikacji oraz obsługi mógł być dokonywany przez personel techniczny niższego szczebla, czyli taki, który występuje w większości zakładów produkcyjnych. Poza tym, w zamyśle standaryzacja oraz modułowość wykonania sterowników cyfrowych powodowałaby szybszą niż do tej pory naprawę lub modernizację pracującego układu sterowania pod kontrolą sterownika cyfrowego. Dodatkowo założono, że taki standaryzowany sterownik powinien posiadać taki rodzaj pamięci półprzewodnikowej, aby była możliwa szybka wymiana programu sterującego, najlepiej przy braku wyłączania z działania sterownika cyfrowego lub przy jego ograniczonym w czasie wyłączaniu. Koncepcja sterownika, o którym mowa pojawiała się stopniowo w zamyśle wielu firm producenckich sterowników cyfrowych takich jak Siemens czy ABB, a wykrystalizowała się ostatecznie pod koniec lat 70-tych XX wieku. Od tego momentu możemy zatem mówić o pojawieniu się na rynku automatyki i sterownia sterowników cyfrowych PLC.

4. Architektura sterownika cyfrowego PLC

Wykrystalizowana w latach 70-tych XX wieku architektura sterownika PLC, (jeszcze nie objęta międzynarodową normą IEC 1131, która to norma został opublikowana w 1993 roku), musiała sprostać wymaganiom, jakie nałożono na zadania procesowe dla tego urządzenia. Były to następujące wymagania m. in.:

- prosta do przeprowadzenia zmiana koncepcji sterowania danym procesem przemysłowym, polegająca na łatwej do zrealizowania modyfikacji lub wymianie programu sterującego w pamięci programu sterownika PLC przy użyciu dostępnego protokołu komunikacyjnego (np. PROFIBUS DP, MODBUS, Ethernet, MPI, PPI) oraz łącza szeregowego (np. RS 485 czy RS 232C);
- możliwe do przeprowadzenia na ekranie monitora (programatora) obserwacje, umożliwiające testowanie sterownika PLC oraz diagnostykę poprawności działania jego programu sterującego w tzw. trybie bezpośredniego połączenia ze sterownikiem (ang. ***ON-Line***), często bez zatrzymania realizowanego programu sterującego;
- odpowiednia dla kontrolowanego procesu szybkość realizacji tzw. pojedynczej pętli programu sterującego (ang. ***S**can **C**ycle*) mniejsza niż 1ms, gwarantująca odpowiednią jakość sterowania danym procesem produkcyjnym;
- duża niezawodność samego sterownika PLC wyrażana tzw. średnim czasem międzyawaryjnym **MTBF** (ang. ***M**ean **T**ime **B**etween **F**ailure*) o wartości kilku milionów godzin bezawaryjnej pracy;
- możliwość zaprogramowania i obsługi sterownika PLC przez personel techniczny średniego szczebla zakładu produkcyjnego, w którym określony sterownik PLC funkcjonuje.

Na podstawie powyższych wymagań opracowano uniwersalną architekturę sterownika PLC, która z niewielkimi zmianami przetrwała do dnia dzisiejszego.



Rysunek 6: Uniwersalna architektura sterownika PLC

Przeznaczenie modułów sterownika PLC, wyszczególnionych na rysunku 6 jest następujące:

- ✓ **Moduł CPU** (ang. ***C**entral **P**rocessing **U**nit*) – najważniejszy blok sterownika PLC, który zawiera mikroprocesor o odpowiednich parametrach technicznych. Bardzo często numer (liczba), który przyporządkowany jest do fabrycznego oznaczenia CPU, oznacza możliwości funkcjonalne (techniczne) sterownika PLC w zakresie m.in. czasu realizacji pojedynczego cyklu programowego *Scan Cycle* lub liczby obsługiwanych wejść/wyjść sterownika (np. CPU 221 w przypadku sterownika rodziny SIMATIC S7 200 firmy Siemens oznacza m.in. liczbę sześciu wejść oraz czterech wyjść sterownika);
- ✓ **Moduł pamięci RAM/ROM** (ang. *Memory Module RAM/ROM*) – jest blokiem sterownika PLC, który wyposażony jest w odpowiedni rodzaj pamięci zarówno dla danych ulotnych jak i nieulotnych. W starszych rodzinach sterowników PLC pamięć typu ROM zastąpiono pamięcią półprzewodnikową typu EPROM, czyli układem, który programuje się elektrycznie za pośrednictwem programatora, zaś kasuje zawartość tej pamięci za pośrednictwem promieniowania ultrafioletowego (w odpowiednim „kasowniku") po uprzednim wyjęciu układu ze sterownika PLC. W nowszych odmianach sterowników PLC funkcjonuje już pamięć półprzewodnikowa typu EEPROM (E^2PROM), czyli układ elektroniczny programowalny oraz kasowalny elektrycznie bez wymontowywania go ze sterownika PLC. Do tego modułu pamięci RAM/ROM możemy wliczyć również pamięć opcjonalną, która może być montowana w zależności od potrzeb do sterownika PLC oraz stanowić nadrzędną pamięć programu sterownika (np. sterowniki firmy SCHIELE);
- ✓ **Rejestr dla wejść sterownika PLC** (ang. *Input Module*) – jest blokiem sterownika PLC, który jest logicznie powiązany z tzw. modułem wejść sterownika **SM** (ang. ***S**ignal **M**odule*). Rejestr ten o organizacji będącej wielokrotnością 1-go Bajtu (1 Bajt = 8 bitów), przechowuje dla CPU informacje o stanie wejść sterownika PLC, która została wpisana do tego rejestru w czasie wykonywania pojedynczego cyklu programowego *Scan Cycle*. Jeżeli na poszczególnym wejściu sterownika PLC wystąpi informacja bitowa ze zbioru {0,1}, czyli *fałsz* lub *prawda* (np. sygnał z czujnika obecności materiału), to informacja ta po wpisaniu do rejestru dla wejść sterownika PLC zajmie „organizacyjnie" jeden bit. Jeżeli zaś na poszczególnym wejściu sterownika PLC

wystąpi informacja analogowa, (np. doprowadzono sygnał analogowy z czujnika ciśnienia powietrza), to informacja ta (po procesach próbkowania, kwantowania oraz kodowania) po wpisaniu jej do tego rejestru będzie zajmować organizacyjnie 1 Bajt lub więcej, czyli osiem lub więcej bitów, w zależności od rozdzielczości przetwornika AC, który został wbudowany w rejestr dla wejść sterownika PLC. W przeważającej większości przypadków o stanie sygnału na poszczególnym wejściu modułu SM, który jest „przedsionkiem" dla rejestru wejść sterownika PLC informuje zorientowana z tym wejściem dioda LED. W przypadku stanu sygnału typu *fałsz*, dioda jest wygaszona, zaś w przypadku sygnału typu *prawda* zapala się.

- ✓ **Rejestr dla wyjść sterownika PLC** (ang. *Output Module*) - jest blokiem sterownika PLC, który jest logicznie powiązany z tzw. modułem wyjść sterownika **OM** (ang. ***O**utput **M**odule*). Rejestr ten analogicznie do rejestru wcześniej opisanego jest również o organizacji, będącej wielokrotnością 1-go Bajtu (1 Bajt = 8 bitów). Rejestr ten przechowuje dla CPU informacje o stanie wyjść sterownika PLC, która została wpisana do tego rejestru w czasie wykonywania pojedynczego cyklu programowego *Scan Cycle*. Jeżeli poszczególne wyjście sterownika PLC „zamierza" sterować urządzeniem za pośrednictwem sygnału ze zbioru {0,1}, czyli *nie pracuje* lub *pracuje* (np. sygnał sterowania przekaźnikiem elektromagnetycznym), to informacja ta po wpisaniu do rejestru dla wyjść sterownika PLC w danym momencie wykonywania programu zajmie „organizacyjnie" jeden bit. Jeżeli zaś na poszczególnym wyjściu sterownika PLC ma wystąpić sygnał analogowy z przedziału 0 ÷ 10V DC, (np. sygnał doprowadzony do silnika elektrycznego prądu stałego), to informacja ta po wpisaniu jej do rejestru wyjść sterownika PLC będzie zajmować organizacyjnie 1 Bajt lub więcej, czyli osiem lub więcej bitów, (też w zależności od rozdzielczości przetwornika, ale tym razem CA, który został wbudowany w rejestr dla wyjść sterownika PLC), by następnie w postaci żądanego sygnału analogowego być dostępna na odpowiednim wyjściu w module wyjść sterownika PLC. W przeważającej większości przypadków o stanie sygnału na poszczególnym wyjściu modułu OM, który znajduje się „za" rejestrem wyjść sterownika PLC, informuje zorientowana z tym wyjściem dioda LED. W przypadku stanu sygnału typu *nie pracuje*, dioda jest wygaszona, zaś w przypadku sygnału typu *pracuje* zapala się. Jak widać sygnalizacja LED występuje tylko dla sygnałów binarnych, czyli ze zbioru {0,1};
- ✓ **Sygnalizacja LED** (ang. *Status LED*) – jest zespołem sterownika PLC, który odpowiada za optyczne informowanie operatora procesu, serwisanta, itp., o stanie pracy samego urządzenia i ewentualnych błędach w pracy CPU. Wśród najczęściej stosowanych informacji za pomocą tych diod LED występują:
  - ⇒ **RUN** – dioda informująca o uruchomieniu sterownika PLC;
  - ⇒ **STOP** – dioda informująca o zatrzymaniu sterownika PLC;
  - ⇒ **DC24V** – dioda informująca o prawidłowym napięciu zasilającym układ sterownika PLC;
  - ⇒ **BATF** – dioda informująca o uszkodzeniu lub wyczerpaniu baterii podtrzymującej CPU;
  - ⇒ **SF** – dioda informująca o możliwym wystąpieniu błędów w zakresie programowym, np. użycie adresu, który w danej konfiguracji CPU nie występuje lub sprzętowym, np. użycie moduły, który nie może współpracować z pozostałymi modułami sterownika PLC.

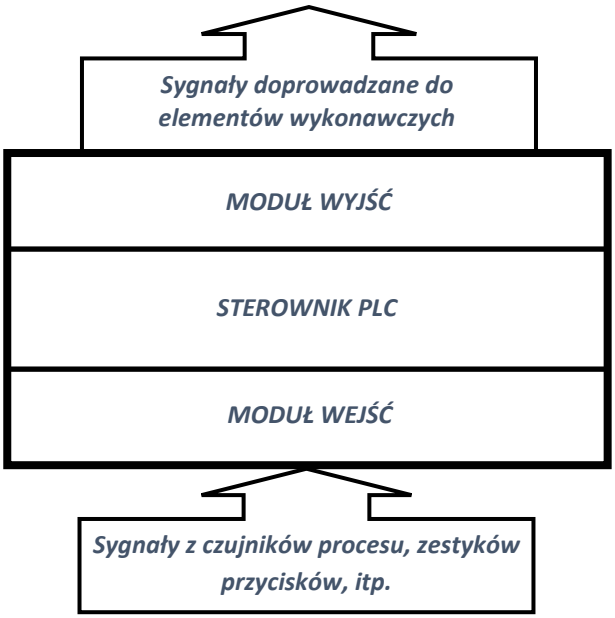
- ✓ **RS 232C, RS 485** – jest blokiem sterownika PLC, który umożliwia komunikację z programatorem urządzenia (starsze wersje sterowników PLC) lub komputerem typu PC z zainstalowanym oprogramowaniem narzędziowym do tworzenia programu sterującego celem przesłania go do sterownika PLC, jak również odpowiada za pracę sterownika PLC w odpowiednich topologiach sieci przemysłowych;
- ✓ **Zasilacz PLC** – jest blokiem sterownika PLC, który odpowiada za wytworzenie oraz dostarczenie napięć zasilających poszczególne bloki funkcjonalne sterownika PLC. Typowym napięciem zasilającym, które doprowadzane jest z zewnątrz do bloku zasilacza PLC jest napięcie 230V AC lub 24V DC. W dużej liczbie przypadków napięcie zasilające sam sterownik PLC jest napięciem zasilającym układy wejścia/wyjścia sterownika PLC;
- ✓ **BUS** – jest blokiem sterownika PLC, odpowiedzialnym za fizyczną rozbudowę sterownika o dodatkowe moduły wejścia/wyjścia. W pewnych rozwiązaniach konstrukcyjnych sterowników PLC fizyczne połączenie bazowego sterownika z dodatkowym modułem wejścia/wyjścia następuje poprzez użycie wbudowanych zatrzasków elektrycznych, zaś w innych rozwiązaniach należy wykorzystać dodatkowe elementy łącznikowe typu *Bus Connector*.

5. Sposób włączenia sterownika PLC w układ sterowania systemem mechatronicznym

Wkomponowanie sterownika PLC do układu sterowania systemem mechatronicznym (będzie o tym mowa bardziej szczegółowo w rozdziale czwartym na konkretnych przykładach sterowania wybranymi systemami mechatronicznymi), musi być poprzedzone określeniem rodzaju oraz liczby sygnałów wejścia oraz wyjścia kontrolowanego systemu mechatronicznego, które muszą być przyjęte oraz wyprowadzone z układu sterowania, opartym na sterowniku PLC. Informacje te są niezbędne dla poprawnego dobrania lub skonfigurowania sterownika. Przy czym sygnały, które pochodzą ze wszelkiego rodzaju czujników, zestyków przycisków sterujących, krańcówek położenia, zestyki pomocnicze przekaźników czy styczników, itp., doprowadzane są do modułu (-ów) wejść sterownika PLC, natomiast sygnały wyprowadzone na zaciskach modułu (-ów) wyjść sterownika PLC doprowadzane są do elementów wykonawczych systemu mechatronicznego (np. cewek przekaźników lub styczników, zaworów elektropneumatycznych lub elektrohydraulicznych, itp.), czyli tych, które pośrednio załączają inne urządzenia wykonawcze lub są same urządzeniami wykonawczymi. Rysunek 7 schematycznie pokazuje ideę włączenia sterownika PLC w układ sterowania systemem mechatronicznym.

DEFINICJA

[***Sygnał wejściowy systemu mechatronicznego*** *– sygnał binarny lub analogowy pochodzący z modułu wyjść sterownika PLC i dostarczany do elementu wykonawczego systemu typu przekaźnik, stycznik, zawór elektromagnetyczny czy silnik elektryczny.*]

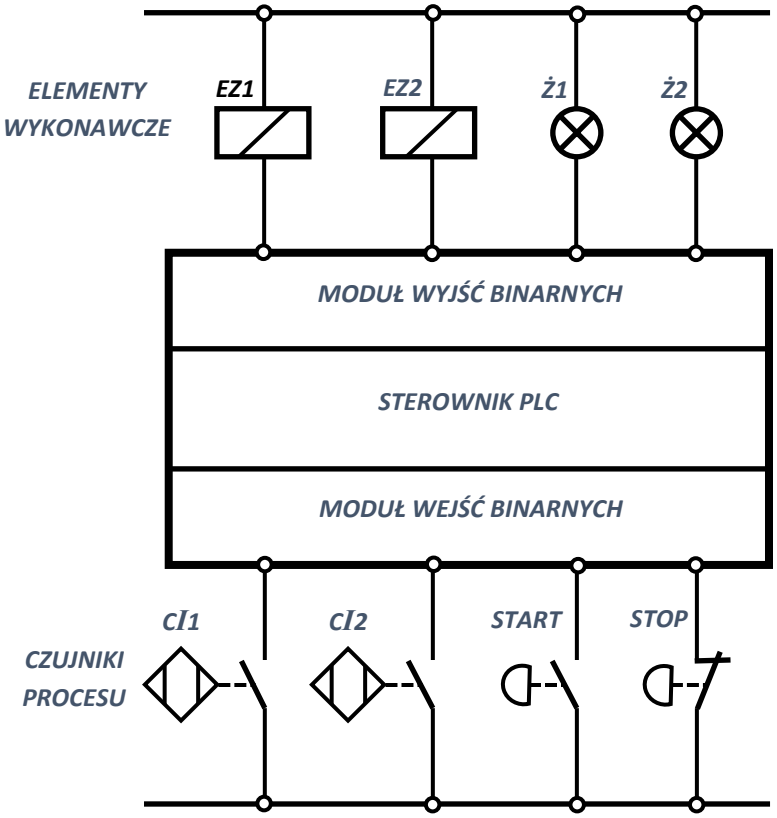


Rysunek 7: Idea włączenia sterownika PLC w układ sterowania

DEFINICJA

[***Sygnał wyjściowy systemu mechatronicznego*** *– sygnał pochodzący z elementu systemu typu czujnik położenia materiału, przycisk sterujący pracą maszyny, czujnik ciśnienia, itp., który jest doprowadzony do modułu wejść sterownika PLC w postaci sygnału binarnego lub analogowego*]

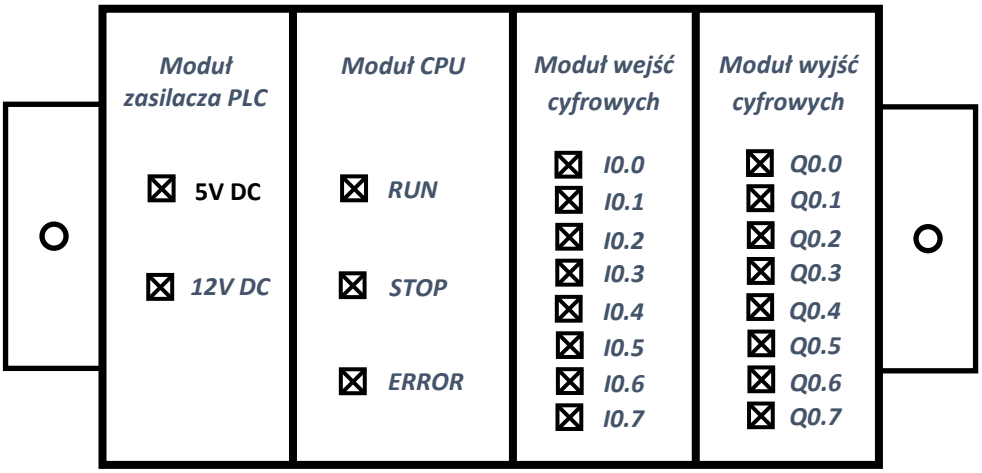
Przyjmując, że dla sterowania systemem mechatronicznym najpierw określono sygnały wejścia/wyjścia systemu i potem dobrano sterownik PLC, schemat sterowania poglądowo wygląda jak na rysunku 8.



Rysunek 8: Poglądowy schemat sterowania z wykorzystaniem sterownika PLC

Poglądowy schemat sterowania z wykorzystaniem sterownika PLC, zilustrowany na rysunku 8 jest pewnym standardem w rysowaniu schematów elektrycznych w systemach mechatronicznych. Widoczny szkic sterownika PLC wyszczególnia moduł wejść oraz moduł wyjść. Do odpowiednich oraz z odpowiednich zacisków tych dwóch rozdzielnych modułów sterownika PLC, (zasygnalizowanych na rysunku 8 za pomocą niewielkich „oczek"), doprowadzono oraz wyprowadzono sygnały, pochodzące od oraz do elementów wejścia/wyjścia systemu mechatronicznego. Na rysunku 8 elementami, które dostarczają dla sterownika PLC sygnałów wyjściowych są dwa czujniki indukcyjne **CI1** oraz **CI2** oraz przyciski **START** oraz **STOP**, zaś elementami, które przyjmują sygnał wejściowy ze sterownika PLC są dwa elektrozawory **EZ1** i **EZ2** oraz dwa sygnalizatory optyczne **Ż1** i **Ż2**.

Należy zaznaczyć, że przy bardziej skomplikowanych schematach elektrycznych, a tak jest prawie zawsze, schemat elektryczny włączenia sterownika PLC do systemu mechatronicznego ulega znacznemu rozbudowaniu. Poszczególne karty dokumentacji elektrycznej (nie zaś tylko jedna karta, np. z rysunkiem 8) włączenia sterownika PLC do systemu mechatronicznego zawierają rozgraniczenie m.in. na schematy elektryczne podłączenia sygnałów wyjściowych do modułów wejść sterownika, (których może być wiele), podłączenia urządzeń wykonawczych do modułów wyjść sterownika PLC, (których również może być wiele), podłączenia elektrycznego jednostki CPU, itp. Dopiero tak rozbudowana dokumentacja elektryczna, nierzadko zawierająca nawet kilkadziesiąt kart, pokazuje sposób aplikacji sterownika PLC do sterowania systemem mechatronicznym. Przyjmując, że rysunek 6 pokazuje postać schematyczną sterownika typu kompaktowego (ang. *Compact PLC*), w którym w jednej obudowie umieszczono wszystkie moduły funkcjonalne sterownika PLC, to rysunek 9 pokazuje schemat funkcjonalny sterownika PLC o budowie modułowej (ang. *Module PLC*).



Rysunek 9: Przykładowa konfiguracja modułowego sterownika PLC

Widoczna na rysunku 9 sprzętowa konfiguracja sterownika PLC o strukturze modułowej „składana" jest najczęściej na szynach typu DIN o szerokości 35 lub 115 mm. Należy zaznaczyć, że pomimo iż prawie każdy moduł takiego sterownika umożliwia podłączenie innego z jednej lub drugiej strony, to zazwyczaj obowiązuje tutaj pewna prawidłowość narzucona konfiguracją programową sterownika modułowego PLC. Z reguły pierwszym z lewej jest moduł zasilacza o odpowiednich parametrach wydajnościowych, dalej w konfiguracji występuje moduł CPU, którego parametry funkcjonalne muszą odpowiadać kontrolowanemu systemowi mechatronicznemu. Konfigurację „zamykają" odpowiednio dobrane moduły wejść i moduły wyjść sterownika.

6. Synteza algorytmu dla systemu mechatronicznego

Przyjęcie założenia, że do sterowania systemem mechatronicznym zostanie wykorzystany sterownik PLC, narzuca podejście informatyczne do rozwiązania problemu utworzenia poprawnego programu dla tego urządzenia. Zazwyczaj jest to związane z dokonaniem tzw. algorytmizacji zadania sterowania.

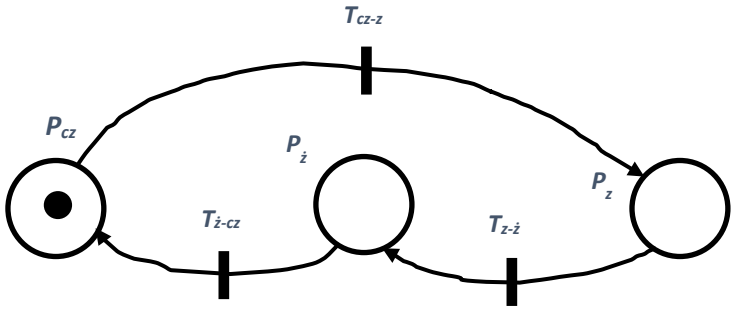
> **DEFINICJA**
>
> [***Algorytm*** *– to skończony ciąg zdefiniowanych czynności, koniecznych do wykonania pewnego rodzaju zadań lub określony sposób postępowania, który doprowadzi do rozwiązania określonego problemu. Najczęściej algorytm prezentowany jest za pomocą odpowiedniego formalizmu graficznego, odpowiedniego dla specyfiki jego wykorzystania*]

Utworzenie algorytmu oprócz rozłożenia określonego zadania na czynniki pierwsze, co dla złożonych systemów mechatronicznych jest nieodzowne, posiada również walor dokumentacyjny, a co za tym idzie i praktyczny. Analiza, bowiem, algorytmu, który będzie przecież częścią dokumentacji technicznej, pokaże krok po kroku sposób funkcjonowania systemu mechatronicznego, który będzie kontrolowany przez sterownik PLC. Można natychmiast zasugerować niecelowość tworzenia algorytmu dla prostego systemu mechatronicznego, np. dla sterowania LEWO/PRAWO siłownikiem pneumatycznym, którego zrozumienie działania może być nietrudne oraz zajmie tylko chwilę. Jednak dobrą praktyką jest, aby przed utworzeniem programu sterującego dla sterownika PLC dokonać syntezy algorytmu dla docelowego systemu mechatronicznego.

> **DEFINICJA**
>
> [***Synteza algorytmu*** *– to proces poszukiwania dla realizacji określonego zadania optymalnej konfiguracji komórek algorytmu, zrealizowanej za pomocą tzw. sieci połączeń*]

6.1. Sieci Petri'ego

Każda sieć połączeń komórek algorytmu, bez względu na jego postać graficzną pochodzi o znanego formalizmu, zwanego sieciami Peri'ego **PN** (ang. ***P****etri* ***N****ets*), której zasady zostały opublikowane w latach 60-tych XX wieku. Sieci PN umożliwiają graficzne modelowanie działań sekwencyjnych w tzw. postaci grafu zorientowanego. Miejsce w sieci PN odpowiada jednej składowej stanu. Zmiana wartości tylko jednej składowej stanu wymusza zmianę całego stanu, co prowadzi do wzrostu w sposób wykładniczy liczby stanów wraz z liczbą miejsc w sieci. Zdarzenia zachodzące w procesach i modelowane za pomocą sieci PN przedstawia się jako: ***P*** *- węzły (kółka)*, natomiast tzw. warunki przejścia (ang. *Transition*) jako ***T*** *- węzły (kreski)*. Do każdego z ***P*** *- węzłów* może przychodzić i odchodzić dowolna liczba ***T*** *- węzłów*. Sterowanie, czyli stan zadziałania symbolizują znaczniki, np. czarne punkty, które umieszczane są w niektórych z ***P*** *– węzłów*. Postać graficzną modelowania sieci PN na przykładzie algorytmu dla sterowania prostą sygnalizacją świetlną na skrzyżowaniu dróg ilustruje rysunek 10.



Rysunek 10: Przykład sieci Peri'ego dla sygnalizacji świetlnej

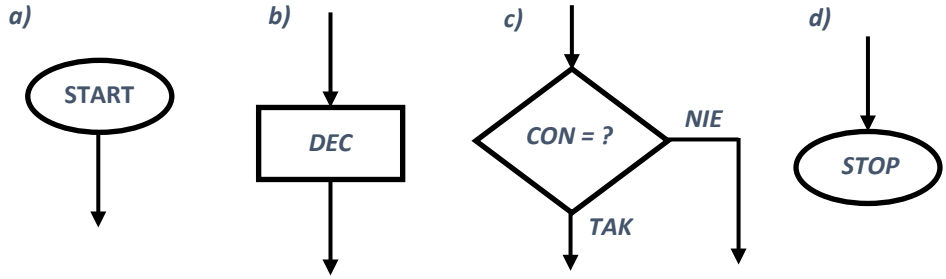
Opiszmy sposób realizacji algorytmu z rysunku 10: zdarzenie procesu określone stanem P_{cz} (zapalone światło koloru czerwonego – czarny punkt wewnątrz okręgu) oczekuje na pojawienie się tranzycji T_{cz-z}, czyli warunku przejścia z tego stanu do oczywistego następnego w kolejności zdarzenia, czyli zapalenia światła koloru zielonego P_z. Po wystąpieniu tego stanu oraz spełnieniu tranzycji $T_{z-ż}$, czyli warunku przejścia dalej, wystąpi stan $P_ż$, czyli zapalenie światła koloru żółtego. Wystąpienie kolejno tranzycji $T_{ż-cz}$ spowoduje pojawienie się stanu początkowego P_{cz}, czyli przejście do zapalenia światła koloru czerwonego. Cykl taki się może nieskończenie powtarzać, gdyż warunkiem prawidłowości każdego algorytmu jest tzw. zupełność, czyli prawidłowe zapętlenie sieci połączeń komórek algorytmu. Dodajmy, co wynika z logiki sieci Petri'ego z rysunku 10, że wszystkie tranzycie są typu Timer, czyli pojawiają się po spełnieniu określonego warunku upływu czasu między pojedynczymi zdarzeniami sieci PN, oraz że rysując kolejne „przejścia" algorytmu czarny punkt „wędrowałby" kolejno do dwóch pozostałych okręgów, oznaczających stany procesu. Dodajmy również jako niezmiernie istotną kwestię, że jeżeli nie założono inaczej, (np. tak jak na rysunku 10 wszystkie trzy tranzycje są typu Timer), to poszczególne tranzycie ***T – węzły***, które w każdej sieci PN przyczyniają się do przejścia od jednego stanu procesu do następnego powinny trwać tylko chwilę, którą można określić jako impuls tranzycji. Gdyby było inaczej, czyli pojawiająca się tranzycja, która wyzwoliła przejście z jednego stanu procesu do drugiego nie „zgasłaby", tylko istniałaby dalej, (w analogii do jej wcześniejszego „zapalenia"), to mogłoby dojść do niekontrolowanego przejścia z tego jednego stanu procesu w drugi przy realizacji sieci Petri'ego od początku.

6.2. Algorytm blokowy

Już takie ogólne omówienie zasad modelowania graficznego, które obowiązują w sieciach Petri'ego daje pogląd na algorytmizację procesu, który ma zostać poddany syntezie. Przypomnijmy, że modelowanie to rozwinęło się w latach 60-tych XX wieku. Na bazie sieci PN rozwinęła się tzw. metoda blokowa ilustrowania algorytmu, którą zaczęto stosować przy modelowaniu zadań dla systemów mikroprocesorowych, a jak wiemy stąd już niedaleko do systemów cyfrowych, i dalej do sterowników PLC. Podstawowymi elementami tego grafu blokowego, który jest w dalszym ciągu używany zwłaszcza w modelowaniu procesów w informatyce są m.in.:

- komórki **START** oraz **STOP**, które odpowiednio rozpoczynają oraz kończą dany obraz algorytmu;
- komórka decyzyjna **DEC**, w której pokazuje się jakie elementy procesu podlegają sterowaniu i jak;
- komórka warunkowa **CON**, w której umieszcza się pojedynczy warunek lub zespół warunków, powiązanych operatorami logicznym, i które *de facto* są tranzycją w rozumieniu sieci typu PN;
- węzły oraz strzałkowanie, które jest niezbędne do zrealizowania sieci połączeń algorytmu.

Podstawowe komórki algorytmu blokowego ilustruje rysunek 11.



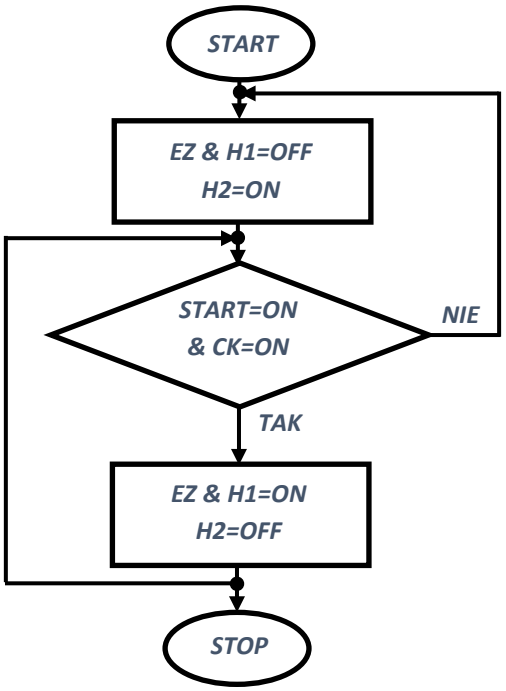
Rysunek 11: Podstawowe komór algorytmu blokowego: a) komórka START; B) komórka decyzyjna; c) komórka warunkowa; d) komórka STOP

Dla wyjaśnienia zasad tworzenia algorytmu metodą blokową posłużmy się następującym przykładem zadania, opisującego działanie układu mechatronicznego:

ZADANIE 1:

[Zrealizować algorytm blokowy dla układu sterowania siłownikiem pneumatycznym jednostronnego działania (powrót sprężyną) za pomocą elektrozaworu EZ. Sterowanie siłownikiem odbywa się za pomocą przycisku dłoniowego START, który umieszczony jest na pulpicie operatora procesu; pobudzenie mono-stabilne tego przycisku przy stanie spoczynkowym tłoczyska (czujnik CK=ON), powoduje wyzwolenie tłoczyska siłownika do ruchu roboczego w kierunku: pchaj. Po wykonaniu pojedynczego cyklu roboczego, tłoczysko wraca samoczynnie do położenia początkowego. Cykl może się powtarzać przy każdorazowym pobudzaniu przycisku START. Należy przewidzieć kontrolę optyczną zarówno ruchu (H1) jak i stopu (H2) tłoczyska siłownika]

Strukturę algorytmu blokowego dla ZADANIA 1 zilustrowano na rysunku 12.



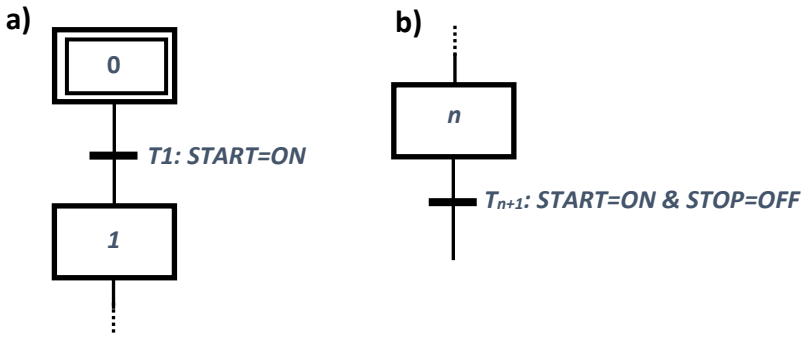
Rysunek 12: Algorytm blokowy do ZADANIA 1

Jądro algorytmu blokowego z rysunku 12 składa się z dwóch komórek decyzyjnych oraz jednej komórki warunkowej. Ponieważ przyjęte jest, iż analizę algorytmu wygodniej jest opisywać poczynając od zatrzymania procesu, to pierwsza komórka decyzyjna opisuje taki właśnie stan procesu: wyłączenie elektrozaworu **EZ** oraz sygnalizatora **H1**, zapalenie zaś sygnalizatora **H2**. Druga komórka decyzyjna opisuje stan przeciwny procesu: załączenie elektrozaworu **EZ** oraz sygnalizatora **H1**, wyłączenie zaś sygnalizatora **H2**. Komórka warunkowa zawiera zaś dwa warunki, które dla zadziałania tłoczyska siłownika muszą wystąpić jednocześnie: **START&CK=ON**. W przeciwnym razie, czyli innej kombinacji stanów logicznych tych warunków, nie wystąpi zadziałanie tłoczyska siłownika, tylko oczekiwanie na spełnienie warunku START&CK=ON.

6.3. Algorytm GRAFCET

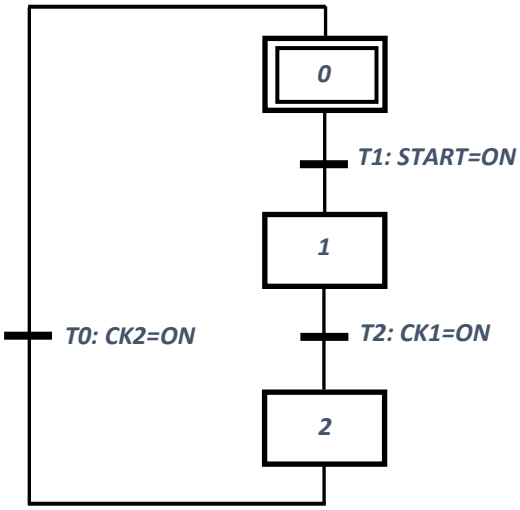
Zaprezentowany wyżej formalizm dla tworzenia algorytmu blokowego jest nadal używany przy syntezie algorytmów dla systemów mechatronicznych. Jednak wprowadzanie sterowania cyfrowego do sterowania procesów produkcyjnych, co jak pamiętamy zaczęło się już odbywać w drugiej połowie XX wieku, wymusiło odrębne podejście do konstrukcji algorytmów dla procesów produkcyjnych. Takim formalizmem był formalizm **GRAFCET**, opracowany przez inżynierów praktyków z branży automatyki oraz naukowców francuskich w końcu lat 70-tych ub. wieku. Formalizm **GRAFCET**, tak jak algorytm blokowy opierał się również na poznanych już zasadach, które obowiązywały w sieciach Petri'ego.

Istotną cechą metody **GRAFCET** jest przedstawianie etapu procesu (tzw. kroku) i warunków przejścia (tzw. tranzycji) za pomocą odpowiednich symboli graficznych, zarezerwowanych dla tego formalizmu. Ten krok algorytmu definiuje się jako sytuację, w której jednostka organizacyjna programu sterującego (np. funkcja) zachowuje się względem swoich wejść i wyjść według reguł zdefiniowanych przez tzw. akcje, czyli zmienne boolowskie związane z tym krokiem. Tranzycję natomiast definiuje się jako warunek przechodzenia sterowania od jednego lub więcej kroków, poprzedzających tę tranzycie, do jednego lub więcej kroków następnych wzdłuż ścieżki programowej. Symbol graficzny kroku, czyli danego etapu procesu symbolizuje czworokąt z podaniem wewnątrz jego pola nazwy kroku, zaś tranzycja, czyli warunek przejścia reprezentowana jest przez pogrubiony odcinek linii poziomej, krzyżującej się ze ścieżką programową wraz z podaniem przy tej linii numeru tranzycji oraz warunku przejścia. Warunek przejścia może być pojedynczy lub być zespołem warunków, połączonych ze sobą operatorami logicznymi. Rysunek 13 ilustruje te dwa podstawowe elementy formalizmu **GRAFCET**.



Rysunek 13: Podstawowe elementy formalizmu GRAFCET: a) sieć algorytmu, która zawiera krok zerowy „0"; b) sieć algorytmu ze złożoną tranzycją T_{n+1}

Na rysunku 13a) symbol kroku o oznaczeniu „0” skonstruowany jest w ten sposób, że w czworokąt o większym wymiarze wpisany jest drugi o mniejszym wymiarze. Jest to tak zwany krok zerowy (początkowy), który w danym algorytmie **GRAFCET** może wystąpić tylko raz. Ważną informacją jest to, że formalizm **GRAFCET** przewiduje również tzw. zapętlenie algorytmu poprzez taki krok początkowy. Wtedy krok o oznaczeniu „0” jest początkowym tylko przy pierwszym rozwoju algorytmu. Później, przy kolejnych zapętleniach jest krokiem, jak każde inne kroki algorytmu. Dogodne jest również numerowanie tranzycji tak, aby kolejno następujące warunki przejścia odnosiły się do kroków, które te tranzycie wywołują. Oba powyższe zagadnienia wyjaśnia rysunek 14.



Rysunek 14: Krok zerowy jako kolejny krok algorytmu

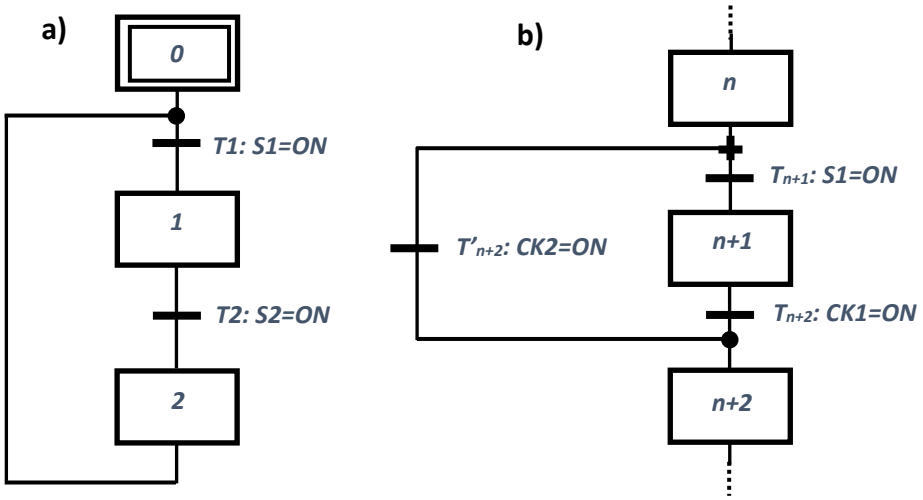
Narzucono, aby przy tworzeniu algorytmu **GRAFCET** przestrzegać następujących uwarunkowań:

⇒ algorytm może zawierać tylko jeden krok początkowy (zerowy), odpowiednio oznaczony;

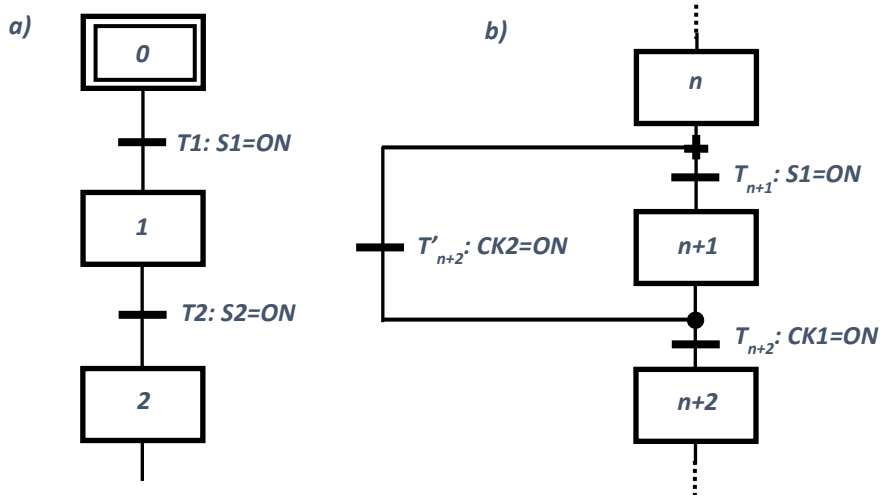
⇒ pomiędzy kolejnymi etapami (krokami) może znajdować się tylko jedna tranzycja, czyli warunek przejścia;

⇒ algorytm powinien być zupełny (skończony), czyli z ostatniego kroku analizowanego wzdłuż ścieżki programowej powinno być zrealizowane zapętlenie do innego kroku algorytmu uzupełnione o tranzycję.

Rysunek 15 ilustruje inne przykłady prawidłowo skonstruowanych algorytmów GRAFCET, zaś rysunek 16 przedstawia typowe błędy, które są popełniane przy tworzeniu algorytmu.



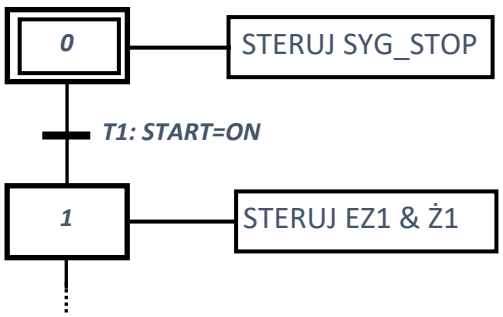
Rysunek 15: Prawidłowe grafy algorytmu GRAFCET: a) realizacja zapętlenia algorytmu z pominięciem kroku zerowego: b) realizacja możliwości ominięcia kroku „n+1”



Rysunek 16: Nieprawidłowe grafy algorytmu GRAFCET: a) niezupełność algorytmu; b) pomiędzy krokiem „n" i „n+2" dwie tranzycie: T'_{n+2} oraz T_{n+2}

Analizując rysunek 15, który pokazuje prawidłowe postacie algorytmu **GRAFCET**, (nazywane często rozwojami algorytmu), można zwrócić uwagę, że w obu algorytmach spełniono warunki prawidłowości konstrukcji algorytmu **GRAFCET**. W pierwszym algorytmie – rysunek 15a) zastosowano zupełność algorytmu oraz spełniono obowiązek wprowadzenia pomiędzy poszczególnymi krokami tylko jednej tranzycji. W drugim algorytmie – rysunek 15b) widać, że jest on fragmentem większej całości. Należy więc tylko podkreślić, że zastosowano prawidłowy rozwój algorytmu z kroku „**n**" do: albo kroku „**n+1**" po spełnieniu tranzycji „$\mathbf{T_{n+1}}$", albo do kroku „**n+2**" poprzez ominięcie kroku „**n+1**" i spełnienie tranzycji „$\mathbf{T'_{n+2}}$". Przy obu tych alternatywnych ścieżkach programowych, wybieranych przez tranzycję „$\mathbf{T_{n+1}}$" (i dalej) lub tranzycję „$\mathbf{T'_{n+2}}$" zachowany jest warunek, iż pomiędzy krokiem poprzednim a następnym jest tylko jedna tranzycja.

W algorytmie z rysunku 16, pokazującym przypadki nieprawidłowych rozwojów grafu **GRAFCET** zauważamy, że w algorytmie z rysunku 16a) tranzycja „**T2: S2=ON**" nie „przyczyni się" do realizacji kolejnego etapu procesu, gdyż algorytm ten nie jest zupełny (skończony). Nie występuje zapętlenie algorytmu. W algorytmie z rysunku 16b) zauważamy, że przy sekwencji kroków o numerach: „**n**" – „**n+1**" – „**n+2**" algorytm jest prawidłowy jak na rysunku 15b). Błąd wystąpił przy umieszczaniu tranzycji „$\mathbf{T_{n+2}}$", która powinna być umieszczona „nad" punktem połączenia linii, oznaczających rozwój algorytmu, nie zaś „pod" tym punktem, jak to uczyniono na rysunku 16b). Elementami, o których musi być jeszcze mowa przy opisie **GRAFECT** są bloki działania, w których umieszcza się działania algorytmu, wykonywane w skojarzonych z blokami działania krokach. Ilustruje to rysunek 17.



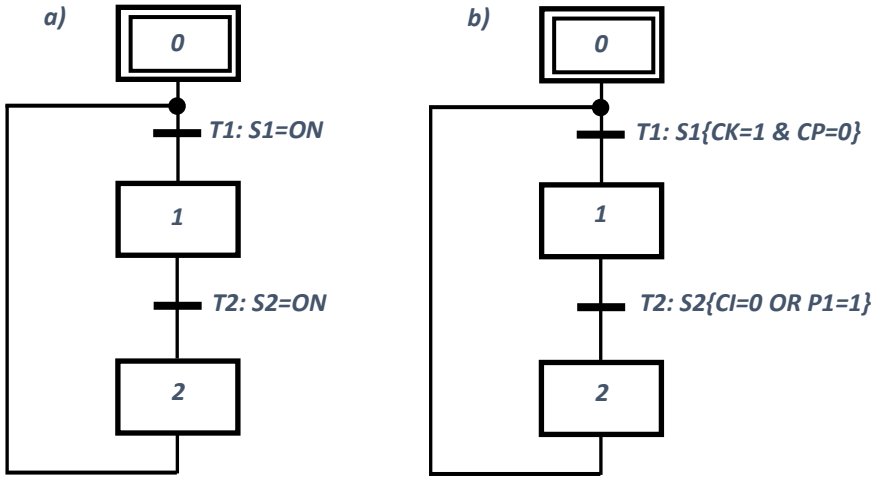
Rysunek 17: Bloki działania w GRAFCET

Z rysunku 17 widać, że blokami działania są „ramki” graficzne z wpisanym tekstem, które są połączone (skojarzone) z danymi krokami algorytmu **GRAFCET** za pomocą poziomych linii. Należy zaznaczyć, że może wystąpić taka postać algorytmu **GRAFCET**, w którym nie wszystkie kroki będą posiadały „swoje” bloki działania. Często taka sytuacja występuje na samym początku algorytmu, przy kroku zerowym, z którym żaden blok działania może nie być skojarzony, gdyż zadaniem kroku zerowego jest inicjacja ścieżki programowej algorytmu, co musi być w żaden sposób zasygnalizowane.

Według rysunku 17 w korku zerowym „**0**” wystąpi działanie, określone opisem: „**STERUJ SYG_STOP**” (steruj sygnalizacją stopu procesu), zaś w kroku oznaczonym jako „**1**”, wystąpi działanie, określone opisem: „**STERUJ EZ1 & Ż1**”, (steruj urządzeniami wykonawczymi **EZ1** oraz **Ż1**). Oczywiście należy zaznaczyć, że nie istnieje sformalizowany standard opisu działań w blokach działania algorytmu **GRAFCET**. Wystarczy, że użyte będą opisy, które są zrozumiałe dla zespołu tworzącego dokumentację procesu.

6.4. Algorytm SFC

Dążenie do ujednolicenia w budowie sterowników PLC, które jak wiadomo począwszy od ich powstania były produkowane przez wiele firm z branży automatyki, oraz dążenie do ujednolicenia zasad programowania tych urządzeń znalazło swoje potwierdzenie w opublikowanej w 1993 roku normie IEC 1131. (O normie tej będzie mowa obszerniej w rozdziale trzecim). W trzecim jej tomie norma ta (w wydaniu europejskim EN 61131) oprócz języków programowania wprowadziła również formalizm dla algorytmu o nazwie **SFC** (ang. ***S**equential **F**unction **C**hart*), opartym na poznanym już formalizmie **GRAFCET**. W tym miejscu podajmy jedynie tę podstawową różnicę, która sprowadza się do tego, iż w przeciwieństwie do algorytmu **GRAFCET** formalizm **SFC** w tworzeniu tranzycji pokazuje dokładne warunki i zależności logiczne między nimi zamiast ogólnych warunków tranzycji. Jest to powodem, iż algorytm SFC jest nazywany algorytmem sterowania, w odniesieniu do algorytmu GRAFCET, który jest nazywany algorytmem procesu. Wyjaśnijmy to na przykładzie na rysunku 18.



Rysunek 18: Algorytmy typu PN: a) algorytm GRAFCET: algorytm SFC

W algorytmie **GRAFCET** (rysunek 18a) w tranzycjach **T1** i **T2** pod postacią warunków **S1** i **S2** kryją się uwarunkowania, które w tym algorytmie nie są pokazywane, pomimo tego, że podczas tworzenia programu sterującego dla sterownika PLC w oparciu o ten algorytm są znane. Do analizy sieci programowej tego algorytmu

wystarczająca jest wiedza, że przejście od jednego kroku algorytmu do drugiego odbędzie się wtedy, gdy raz spełniony zostanie warunek **S1** (co oznaczono jako ON) i następnie **S2** (co oznaczono identycznie). Odwrotna sytuacja występuje w grafie algorytmu SFC (rysunek 18b). Tutaj w tranzycjach **T1** i **T2** pod postacią warunków **S1** i **S2** kryją się uwarunkowania, które w tym algorytmie są pokazane. Warunek **S1** to jednoczesne zadziałanie czujnika **CK** (CK=1) i niezadziałanie czujnika **CP** (CP=0), a warunek **S2** to alternatywa: albo niezadziałanie czujnika **CI** (CI=0) albo zadziałanie czujnika **P1** (P1=1). Zatem, podczas analizy sieci programowej algorytmu jest wiadomo, jakie czujniki oraz w jaki sposób powinny zadziałać, aby następowało przejście od jednego kroku algorytmu do drugiego. (Dokładne omówienie zasad tworzenia algorytmu SFC zawarto w rozdziale trzecim).

7. Metody programowania sterowników PLC

Wprowadzenie cyfrowych urządzeń programowalnych do sterowania procesami produkcyjnymi, co jak pamiętamy, odbywało się stopniowo w miarę rozwoju technologii wytwarzania układów cyfrowych (mikroprocesorowych) mogło być możliwe tylko dzięki temu, że można było najpierw utworzyć program dla najważniejszego układu takiego sterownika, czyli np. mikroprocesora i później umieszczeniu tego programu (zaprogramowanie) w pamięci programu tego urządzenia. Pomijając opis historycznie archaicznych już metod tworzenia kodu cyfrowego dla układu cyfrowego, dopiero pojawienie się komputerów z monitorem ekranowym oraz systemów typu DOS wpłynęło na zwiększenie szybkości oraz efektywności tworzenia kodu cyfrowego, rozumianego jako program dla sterownika cyfrowego. Uproszczonym wydaniem takiego urządzenia programującego do tworzenia programu dla sterownika cyfrowego było urządzenie nazywane po prostu programatorem. Programator taki wyposażony był w uproszczoną wersję systemu operacyjnego oraz przyciski, które możemy określić rodzajem klawiatury. Klawiatura programatora umożliwiała wprowadzanie znaku alfanumerycznych do urządzenia i ich obserwację na ekranie typu LED. Po wprowadzeniu sekwencji kodu, która była programem dla sterownika, całość kodu była przesyłana do pamięci programu sterownika cyfrowego.

Należy zaznaczyć, że podobne podejście obowiązywało dla sposobu tworzenia programu sterującego dla pierwszych sterowników PLC. Wtedy jeszcze nie można było raczej mówić o metodach tworzenia programu PLC w wydaniu obecnym ze względu na możliwości sprzętu programującego. (Metody programowania sterowników PLC objęte normą IEC 1131-3 omówiono szczegółowo w rozdziale trzecim).

Historycznie rzecz ujmując pierwsze metody programowania sterowników PLC były metodami (językami) tekstowymi. Do nich zaliczamy metodę **STL** (ang. ***S**tatement **L**ist*) niekiedy nazywaną **IL** (ang. ***I**nstruction **L**ist*), oraz metodę **ST** (ang. ***S**trutural **T**ext*).

Pojawienie się najpierw metod tekstowych do tworzenia programu sterującego dla sterownika PLC związane było z takimi a nie innymi parametrami programatorów, które w tamtym czasie umożliwiały tylko wprowadzanie prostych instrukcji programowych za pomocą tzw. mnemoników rozkazów. Określony mnemonik rozkazu, zapisany za pomocą prostych znaków alfanumerycznych, po jego wprowadzeniu za pomocą prostej klawiatury, był widoczny na wyświetlaczu programatora. Po jego zatwierdzeniu klawiszem typu <ENTER> wpisywano kolejny mnemonik rozkazu. Pozostałe czynności powtarzano. W ten sposób, wiersz po wierszu tworzono

kompletny program dla sterownika PLC. Następnie za pośrednictwem odpowiedniego interfejsu szeregowego ładowano ten program do pamięci programu sterownika PLC, który mógł być potem uruchomiony w tym urządzeniu. Tworzenie i późniejsze wprowadzenie takiego programu do sterownika PLC było czasochłonne.

W późniejszym czasie, gdy siłą rzeczy do programowania sterowników PLC postanowiono użyć komputera klasy PC (bo już istniały na rynku), który jeszcze wtedy pracował pod kontrolą systemów operacyjnych typu DOS, nastąpiło dokoptowanie do metody **STL** oraz **ST** (które były i są nadal używane), nowo opracowanej graficznej metody **LAD** (ang. ***L**adder **D**iagram*), nazywanej często metodą drabinkową. Metoda ta polegała na wprowadzaniu na ekran komputera typu PC za pomocą klawiatury komputerowej odpowiednich symboli graficznych i ich wirtualne łączenie, co pozwalało na utworzenie struktury programu dla sterownika PLC. Po jego zakończeniu cały program, odpowiednio skompilowany był przesyłany za pośrednictwem odpowiedniego łącza szeregowego do pamięci programu sterownika PLC i mógł być uruchomiony. Już wtedy programowanie sterownika PLC może nazwać efektywnym w porównaniu do sposobu z użyciem prymitywnego programatora.

W momencie pojawienia się systemów operacyjnych typu okienkowego typu Windows 3.1, zainstalowanych na komputerach klasy PC, pojawiła się trzecia metoda tworzenia programu dla sterownika PLC, metoda **FBD** (ang. ***F**unction **B**lock **D**iagram*). Metoda ta polegała na umieszczaniu (rozmieszczaniu) najczęściej już za pomocą myszy komputerowej funktorów logicznych i innych bloków programowych na ekranie komputera, tworząc program sterujący dla sterownika PLC. Czynności załadowania programu do sterownika PLC i jego uruchomienie były podobne do metod omówionych wyżej.

8. Podsumowanie

W module pierwszym zawarto informacje, niezbędne do zrozumienia działania oraz istoty włączenia sterownika PLC do układu sterowania systemem mechatronicznym. W kolejności chronologicznej przypomniano moment pojawienia się mikroprocesora, który zastąpił w budowie urządzeń cyfrowych układy cyfrowe małej oraz średniej skali integracji i zapoczątkował erę najpierw programowalnych sterowników cyfrowych, które zaczęły być używane do kontrolowania procesów produkcyjnych, by później stać się rdzeniem opracowanych w drugiej połowie XX wieku sterowników cyfrowych PLC. Wskazano na sposób włączenia sterownika PLC do układu sterowania systemem mechatronicznycm, wskazano na sposoby tworzenia algorytmów dla systemów mechatronicznych oraz przedstawiono zarys powstania i cechy metod tworzenia programu sterującego dla sterownika PLC. Całość materiału uzupełniają niezbędne definicje oraz rysunki, które autor zaproponował dla lepszego zrozumienia wykładanego materiału.

BIBLIOGRAFIA

1. Borelbach K.H., i inni: *Steuerungstechnik mit speicherprogrammierten steuerrungen SPS*. Munchen 1992.
2. Czemplik A., Jabłoński A.: *Stacje operatorskie w systemach automatyki - zadania i oprogramowanie*. IX Krajowa Konferencja Naukowo-Techniczna nt. Zadania mikroprocesorów w automatyce i pomiarach, Warszawa, Październik 1994.
3. Hajda J., Kasprzyk J., Wyrwał J.: *Programowanie sterowników PLC*. Gliwice 1998.
4. Jelonek K., Trawiński A., Zakrzewski D.: *Popularne standardy transmisji szeregowej. Przegląd interfejsów i protokołów komunikacyjnych*. Elektronizacja nr 6-8 1997.
5. Markiewicz H.: *Instalacje elektryczne*. Warszawa WNT 1996.
6. Mikulczyński T., Samsonowicz Z.: *Automatyzacja dyskretnych procesów produkcyjnych*. Warszawa WNT 1997.
7. Norma IEC 1131 Programmable Controllers. 1993.
8. Norma PN-89/M-42007/01 Automatyka i pomiary przemysłowe. Oznaczenia na schematach.
9. Norma PN-90/M-42007/02 Automatyka i pomiary przemysłowe. Oznaczenia funkcji systemów komputerowych.
10. OMRON, *Programmable Controllers*. Katalog 1995.
11. Sacha K., *Projektowanie oprogramowania systemów sterujących*. Wydawnictwo Politechniki Warszawskiej 1996.
12. Sacha K.: *Systemy czasu rzeczywistego*. Wydawnictwo Politechniki Warszawskiej 1997.
13. SCHIELE, *Programmable Controllers*. Katalog 1995.
14. Seta Z., *Wprowadzenie do zagadnień sterowania. Wykorzystanie programowalnych sterowników logicznych PLC.* Wydawnictwo MIKON, Warszawa 2002.
15. SIMATIC S5 - Step 5 Ladder 90. Manual. Siemens.
16. SIMATIC, *LAD/STL/FBD Programming Manual*. Siemens 1998.
17. SIMATIC - S7 300 *Programmable Controller, Instalation and Hardware. Manual*. Siemens 1998.
18. SIMATIC - S7 300 and M7 300, *Programmable Controllers, Module Specyfications, Reference Manual*. Siemens 1998.
19. SIMATIC - S7 200, *Programmable Controllers, Hardware and Instalation, Manual*. Siemens 1997.
20. Traczyk W.: *Układy cyfrowe automatyki*. Warszawa WNT 1974.
21. Winiecki W.: *Organizacja komputerowych systemów pomiarowych*. Wydawnictwo Politechniki Warszawskiej, Warszawa 1997.