KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Politechnika Gdańska
Wydział Elektrotechniki i Automatyki

STEROWNIKI PROGRAMOWALNE

Część 2

Ireneusz Mosoń

Gdańsk, 2010

Publikacja jest dystrybuowana bezpłatnie

Materiał został przygotowany w związku z realizacją projektu pt. „Zamawianie kształcenia na kierunkach technicznych, matematycznych i przyrodniczych – pilotaż" współfinansowanego ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego
Nr umowy: 46/DSW/4.1.2/2008 – zadanie 018240 w okresie od 21.08.2008 – 15.03.2012

**Materiały pomocnicze do przedmiotu „Sterowniki programowalne”
prowadzonego w języku angielskim w semestrze zimowym w roku
akademickim 2010/2011 dla V semestru specjalności zamawianej
„Technologie informatyczne w elektrotechnice”
na studiach pierwszego stopnia na kierunku Elektrotechnika**

Przedmowa

Projektowanie i budowę nowoczesnych systemów sterowania trudno sobie obecnie wyobrazić bez szerokiego stosowania sterowników programowalnych (*ang. Programmable Controllers*). Od czasu, gdy zastosowano pierwsze proste sterowniki programowalne w przemyśle samochodowym minęło już ponad czterdzieści lat, w ciągu których sterowniki programowalne potwierdziły swoją przydatność w systemach sterowania w różnych zastosowaniach. Przede wszystkim pracują jednak w przedsiębiorstwach i zakładach produkcyjnych w różnych gałęziach przemysłu. Sterują w nich pojedynczymi maszynami i urządzeniami, zautomatyzowanymi liniami produkcyjnymi oraz procesami technologicznymi. W czasie od pierwszych zastosowań sterowników do obecnych czasów nastąpił ogromny rozwój w konstrukcjach i możliwościach funkcjonalnych sterowników programowalnych - od prostych sterowników wykonujących w sposób programowy funkcje sterowania logicznego (realizowane wcześniej za pomocą układów sterowania stykowo-przekaźnikowych), do nowoczesnych sterowników mogących pracować w sieci i realizujących nawet bardzo złożone zadania sterowania.

Ponieważ sterowniki programowalne są obecnie podstawowymi urządzeniami w oparciu o które projektuje się i realizuje sterowanie w systemach automatyki, od absolwentów uczelni technicznych – w tym i kierunku Elektrotechnika – oczekuje się, że będą posiadali niezbędną wiedzę o sterownikach programowalnych i ich zastosowaniach, oraz umiejętność tworzenia i uruchamiania programów sterowania [35,38]. Tym bardziej taką wiedzę i umiejętności powinni posiadać przyszli absolwenci specjalności zamawianej „Technologie informatyczne w elektrotechnice”.

Podstawowym źródłem informacji dotyczących sterowników programowalnych jest norma międzynarodowa IEC 61131 (przyjęta również jako norma europejska EN 61131, a także jako polska norma uznaniowa PN-EN 61131). W normie zawarte są pojęcia i zagadnienia wspólne (ogólne i szczegółowe) dotyczące sprzętu i programowania sterowników programowalnych. Informacje o sterownikach programowalnych można znaleźć także w artykułach w specjalistycznych czasopismach i w coraz liczniejszych pozycjach książkowych. W artykułach najczęściej omawiane są nowe rozwiązania sprzętowe i programowe różnych sterowników oraz nowe możliwości ich zastosowań, które z nich wynikają, a także opisywane są przykłady konkretnych zastosowań sterowników programowalnych w różnych dziedzinach. W książkach zazwyczaj przedstawiane są podstawy budowy, zasada działania i programowanie sterowników - najczęściej na przykładach wybranych sterowników programowalnych różnych producentów. W książkach często zawarte są również zagadnienia

ogólne i szczegółowe dotyczące sterowania automatycznego, a związane ze stosowaniem sterowników programowalnych. Jednak podstawowym, a także szczegółowym i najbardziej aktualnym źródłem informacji dotyczących konkretnych sterowników programowalnych są dokumentacje techniczne. Jest to też źródło najbardziej aktualne – szczególnie w przypadku dokumentacji w wersji elektronicznej.

Przygotowane materiały pomocnicze (Część 2) w dużej części zawierają przedstawienie treści programowych przedmiotu „Sterowniki programowalne" (wykłady, ćwiczenia i laboratorium) prowadzonego w roku akademickim 2010/2011 dla V semestru specjalności zamawianej „Technologie informatyczne w elektrotechnice". W materiałach tych, oprócz omówienia wybranych zagadnień dotyczących pracy sterowników programowalnych w sieci, dokonano również próby usystematyzowania materiału dotyczącego strukturyzacji programów sterowania i sterowania procesami sekwencyjnymi.

Część 1 tych materiałów pomocniczych napisana jest w języku angielskim, gdyż zajęcia dla studentów specjalności zamawianej prowadzone są po angielsku. Dzięki zajęciom prowadzonym w języku angielskim, tym materiałom przygotowanym w dwóch językach, oraz pozycjom ze spisu literatury, studenci będą mogli opanować terminologię angielską i polską dotyczącą sterowników oraz doskonalić swoje umiejętności w posługiwaniu się językiem angielskim w technice.

W spisie literatury (wspólnym dla obu części) zamieszczono zarówno te pozycje literatury do których znajdują się odwołania w tych materiałach, jak również i inne wybrane pozycje przydatne do zrozumienia i opanowania przez studentów materiału z wykładów, ćwiczeń i zajęć laboratoryjnych.

Spis treści

Treści i efekty kształcenia przedmiotu „Sterowniki programowalne”

Treści kształcenia

WYKŁAD

Sterowniki programowalne w systemach sterowania. Rodzaje, budowa i zasada działania. Wykonywanie programu sterowania. Pamięć obrazu procesu. Charakterystyka sprzętowa sterownika. Powiązanie ze sterowanym procesem. Układy wejść/wyjść binarnych, analogowych i specjalnych.

Podstawy programowania. Norma PN-EN 61131-3. Model oprogramowania. Języki programowania. Typy danych i deklarowanie zmiennych. Adresowanie. Elementy organizacyjne oprogramowania - programy, funkcje i bloki funkcyjne. Tworzenie funkcji i bloków funkcyjnych użytkownika. Strukturyzacja programów sterowania. Czynniki jakości oprogramowania.

Praca sterowników programowalnych w sieci. Struktury sieci. Sprzęgi komunikacyjne i media transmisji. Sposoby sterowania dostępem do łącza. Protokoły komunikacyjne (Suconet K, Modbus RTU, Profibus DP, AS-i). Ethernet przemysłowy (protokoły: Modbus TCP, Powerlink, Profinet).

Projektowanie układów i systemów sterowania ze sterownikami programowalnymi. Dobór sterownika do konkretnego zastosowania. Realizacja dialogu człowiek - maszyna (HMI). Programy SCADA.

ĆWICZENIA

Systemy liczbowe stosowane w sterownikach programowalnych. Typy danych i funkcje ich konwersji. Tworzenie algorytmów sterowania; elementy graficzne algorytmów. Oprogramowanie narzędziowe Easy Soft CoDeSys. Pisanie programów sterowania (z wykorzystaniem języków: IL, LD, FBD, ST, CFC) i ich uruchamianie z wykorzystaniem symulatora programowego (sterownik wirtualny). Tworzenie ekranów wizualizacji. Programowanie sterowania procesami sekwencyjnymi w języku SFC.

LABORATORIUM

Oprogramowanie narzędziowe Sucosoft S40 (struktura oprogramowania; konfigurowanie układu sterowania; edycja programu, uruchamianie testowanie i dokumentowanie programów sterowania). Program sterowania przenośnikiem - I i II. Funkcje konwersji danych i operacje arytmetyczne. Liczenie zdarzeń i opcje kompilatora. Tworzenie bloku funkcyjnego użytkownika. Modyfikacja programu i zmiana wartości zmiennych w trybie On-line. Wejście szybkiego licznika i jego wykorzystanie. Programowanie sterowników serii PS4-200 i PS4150 pracujących w sieci (master - active slave).

Efekty kształcenia

Student opisuje typy i budowę sterowników programowalnych. Wyjaśnia zasadę działania sterownika i wykonywania programu użytkownika. Student dobiera sterownik do konkretnego zastosowania. Analizuje wymagania zadań sterowania i opracowuje algorytmy sterowania. Pisze, uruchamia i testuje programy o małej i średniej złożoności do sterowania różnymi obiektami sterowania. Tworzy funkcje i bloki funkcyjne użytkownika. Tworzy proste aplikacje wizualizacyjne.

1. Strukturyzacja programów sterowania

1.1. Wprowadzenie

Strukturyzacja jest i zawsze była zalecaną formą tworzenia oprogramowania. Stwierdzenie to odnosi się także do programowania sterowników programowalnych. Na przestrzeni lat zmieniały się możliwości strukturyzacji programów sterowania [26,36].

Biorąc pod uwagę narzędzia do pisania programów sterowania na sterowniki programowalne oraz możliwości strukturyzacji programów sterowania można wyróżnić trzy podstawowe etapy w procesie zmian sposobów tworzenia oprogramowania.

Pierwszy etap obejmuje okres od 1968 roku, czyli od wyprodukowania pierwszego sterownika programowalnego, do początku lat osiemdziesiątych ubiegłego wieku, czyli do pojawienia się pierwszych komputerów osobistych typu IBM PC.

Drugi etap obejmuje okres od początku lat osiemdziesiątych do roku 1993, czyli do ustanowienia normy międzynarodowej IEC 1131 (od roku 1996 IEC 61131), a w szczególności części trzeciej tej normy dotyczącej języków programowania.

Trzeci etap obejmuje okres od roku 1993 do dnia dzisiejszego.

1.2. Etap pierwszy strukturyzacji programów sterowania

Etap pierwszy charakteryzował się tym, że do programowania sterowników powszechnie używane były programatory ręczne.[1] Program napisany z wykorzystaniem programatora ręcznego stanowił jeden blok programowy bez możliwości stosowania zmiennych symbolicznych, a adresy wszystkich skoków były definiowane jako konkretne liczby będące numerami linii programu, do których dany skok był wykonywany.

Z tych powodów możliwości strukturyzacji programów sterowania były bardzo ograniczone. Sprowadzały się głównie do odpowiedniego uporządkowania sekwencji programowych tak, aby w jak największym stopniu odzwierciedlały one algorytm sterowania.

[1] W tym okresie, oprócz programatorów ręcznych, istniały również programy narzędziowe do programowania sterowników instalowane na minikomputerach. Jednakże ze względu na cenę minikomputerów i oprogramowania narzędziowego mogły sobie na nie pozwolić tylko duże firmy zajmujące się automatyką przemysłową. Względy ekonomiczne powodowały więc, że dla większości inżynierów zajmujących się programowaniem sterowników, a pracujących w małych i średniej wielkości firmach, minikomputery z programami narzędziowymi były niedostępne; pozostawało im programowanie za pomocą programatorów ręcznych.

Najczęściej otrzymywano wówczas taką kolejność sekwencji programowych, która odpowiadała fazom pracy sterowanej maszyny, urządzenia lub procesu technologicznego, albo też etapom pracy sterowanej linii technologicznej.

Poszczególne fragmenty programu, podczas wykonywania przez sterownik cykli programu, mogły być pomijane jako nieistotne na danym etapie sterowania. Służyły do tego celu instrukcje skoków warunkowych lub bezwarunkowych oraz specjalne instrukcje włączające/wyłączające warunkowo fragmenty programu do analizy w bieżącym cyklu programu. Przykładami takich instrukcji są MCS (*ang. Master Control Set*) i MCR (*ang. Master Control Reset*) stosowane np. w sterownikach firmy Hitachi [12], oraz instrukcje MLS (*ang. Master Line Set*) i MLR (*ang. Master Line Reset*) w sterownikach firmy Koyo [8]. Wykorzystanie wyżej wymienionych instrukcji nie zmniejszało pracochłonności przy tworzeniu programów sterowania, było natomiast bardzo ważne z punktu widzenia strukturyzacji oprogramowania, gdyż prowadziło do większej przejrzystości oprogramowania i dzięki temu ułatwiona była jego analiza i uruchamianie.

Odrębnym problemem, pośrednio związanym ze strukturyzacją oprogramowania a zwiększającym pracochłonność, była konieczność uaktualniania adresów w programie po każdorazowym wprowadzeniu zmiany w programie sterowania. Sytuacje takie były stosunkowo częste, gdyż występowały zarówno podczas modyfikacji programu na etapie jego tworzenia jak i później, podczas wprowadzania zmian do programu na etapie uruchamiania. Częściowym rozwiązaniem pozwalającym wyeliminować każdorazowe uaktualnianie adresów po wprowadzeniu zmiany w programie sterowania było wykorzystywanie pustych instrukcji NOP (*ang. No OPeration*). Za pomocą tych instrukcji były rezerwowane - pod ewentualne przyszłe zmiany - niewielkie fragmenty pamięci programu. Dzięki temu, jeśli nie następowała zmiana długości programu, to nie następowały również zmiany adresów w tych częściach programu, które nie były modyfikowane.

Jako przykład na rysunku 1.1 przedstawiono fragment (przed i po modyfikacji) programu sterowania napisany w języku IL (lista instrukcji) dla sterowników serii PS4-100 firmy Moeller Electric w oprogramowaniu Sucosoft S30-S3. Z lewej strony zamieszczono fragment programu przed, a z prawej po modyfikacji. Po dokonaniu modyfikacji sekwencji programowej zapisanej w liniach programu od adresu 058 do adresu 061, a polegającej na dodaniu instrukcji A I0.7 w 060 linii programu usuwając równocześnie jedną instrukcję NOP, adresy linii programu po pozostałych instrukcjach NOP nie ulegają zmianie. Dzięki temu adres skoku warunkowego w 066 linii programu, jak również wszystkie inne adresy skoków nie pokazane w tym fragmencie programu, również nie ulegają zmianie.

...	...	...	...	...	...
058	L	I 0.2	058	L	I 0.2
059	O	I 0.3	059	O	I 0.3
060	AN	M 2.5	060	A	I 0.7
061	=	Q 0.1	061	AN	M 2.5
062	NOP		062	=	Q 0.1
063	NOP		063	NOP	
064	NOP		064	NOP	
065	L	I 0.1	065	L	I 0.1
066	JC	095	066	JC	095
067	L	M 0.7	067	L	M 0.7
...	...	...	...	...	...

Rys.1.1. Przykład rezerwowania pamięci z wykorzystaniem instrukcji NOP

Wszystkie wymienione powyżej wady zostały wyeliminowane na kolejnych etapach procesu zmian sposobów tworzenia oprogramowania. Wyraźnie zwiększyły się też możliwości strukturyzacji programów sterowania. Było to możliwe dzięki rozwojowi i upowszechnieniu komputerów osobistych typu IBM PC i opracowaniu przez producentów sterowników programów narzędziowych uruchamianych na tego typu komputerach.

1.3. Możliwości strukturyzacji programów sterowania na drugim etapie

Na drugim etapie w procesie zmian sposobów tworzenia oprogramowania, dzięki programom narzędziowym, tworzenie programów sterowania na sterowniki programowalne stało się wygodniejsze, gdyż:

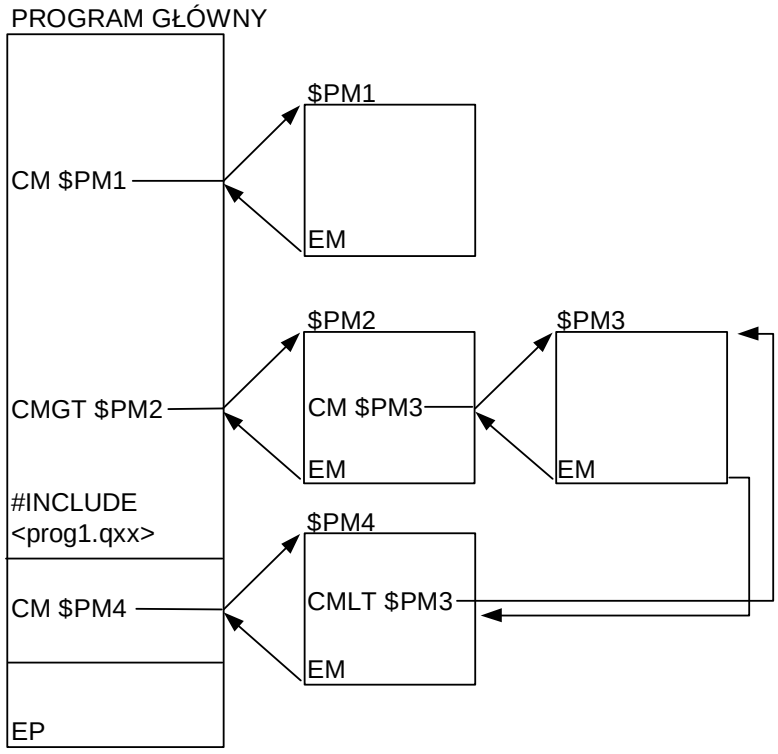
- zmiennym można było przyporządkować nazwy symboliczne;
- adresy skoków były definiowane jako etykiety bloków programowych do których skoki były wykonywane;
- zmiennym i blokom programowym można było przyporządkować komentarze;
- możliwa była pełniejsza dokumentacja oprogramowania;
- łatwiejsze stało się wprowadzanie zmian w programach sterowania;
- zwiększyły się możliwości strukturyzacji programów sterowania.

Ponieważ w programach narzędziowych producentów sterowników występowały różnice w instrukcjach i językach programowania, w różny sposób rozwiązywane było również zagadnienie strukturyzacji programów sterowania. Poniżej omówiono kilka rozwiązań.

W sterownikach serii PS306, PS316 i PS4 firmy Moeller (do roku 1999 Klockner-Moeller) można było tworzyć programy sterowania i podprogramy, przy czym podprogramy mogły być kompilowane oddzielnie lub łącznie z programem głównym [9,26]. W pierwszym

przypadku, gdy podprogram był kompilowany oddzielnie a nie łącznie z programem głównym, był on umieszczany w pamięci sterownika za programem głównym i mógł być przywoływany w programie głównym wielokrotnie. Wywoływanie podprogramu następuje poprzez instrukcję CM (*ang. Call Module*); jest to wywołanie bezwarunkowe. W drugim przypadku kod podprogramu był przywoływany i dołączany do kodu programu głównego na etapie kompilacji i kompilowany łącznie z programem głównym. Z tego powodu taki podprogram mógł być przywołany w programie głównym tylko raz. Dołączenie tego typu podprogramu następuje poprzez instrukcję INCLUDE. Zarówno w pierwszym jak i drugim przypadku możliwe jest zagnieżdżanie podprogramów (do 8 lub 64 poziomów – w zależności od rodzaju sterownika).

Na rysunku 1.2 przedstawiono przykładową strukturę programu sterowania składającą się z programu głównego, czterech podprogramów i jednego podprogramu o nazwie prog1.qxx dołączonego za pomocą instrukcji INCLUDE.

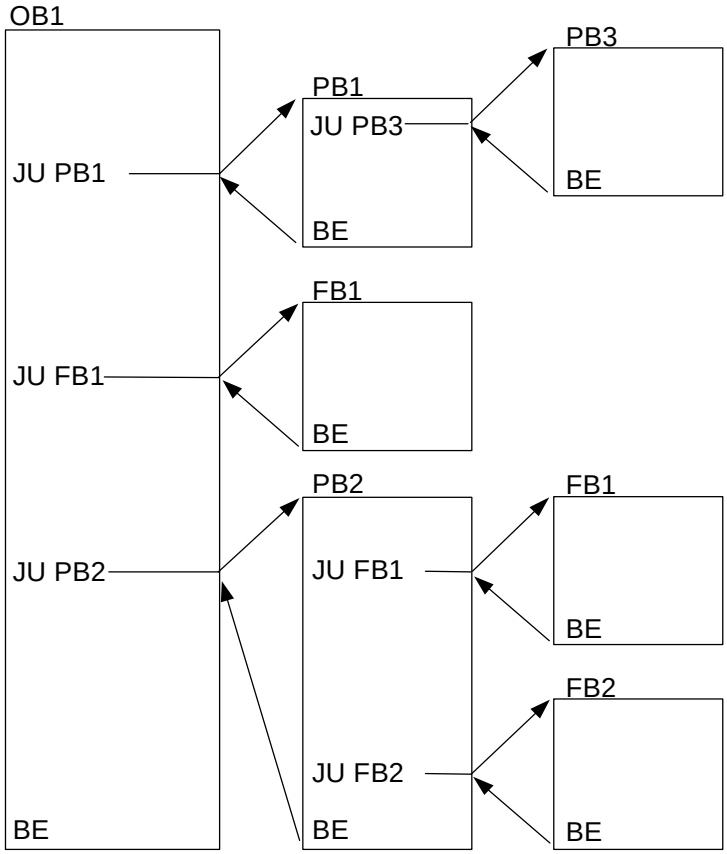


Rys.1.2. Przykład struktury programu dla sterowników PS4 w oprogramowaniu S30-S4

Na rysunku 1.2 pokazano bezwarunkowe wywołania podprogramów $PM1 (*ang. Program Module, $ oznacza podprogram*) i $PM3, warunkowe wywołanie CMGT (*ang. Call Module on Greater Than*) podprogramu $PM2 i warunkowe wywołanie CMLT (*ang. Call Module on Less Than*) podprogramu $PM3. Bezwarunkowe wywołanie podprogramu $PM4 jest

wykonywane z podprogramu który został dołączony do programu głównego instrukcją INCLUDE. Z żadnego z podprogramów przedstawionych na rysunku 1.2 nie następuje wcześniejszy (w wyniku instrukcji powrotu warunkowego lub bezwarunkowego) powrót do wywołującego go programu lub podprogramu. Ostatnią instrukcją programu jest EP (*ang. End of Program*) a ostatnią instrukcją podprogramu jest EM (*ang. End of Module*). Podprogram prog1.qxx nie posiada instrukcji końca, gdyż jest kompilowany łącznie z programem głównym.

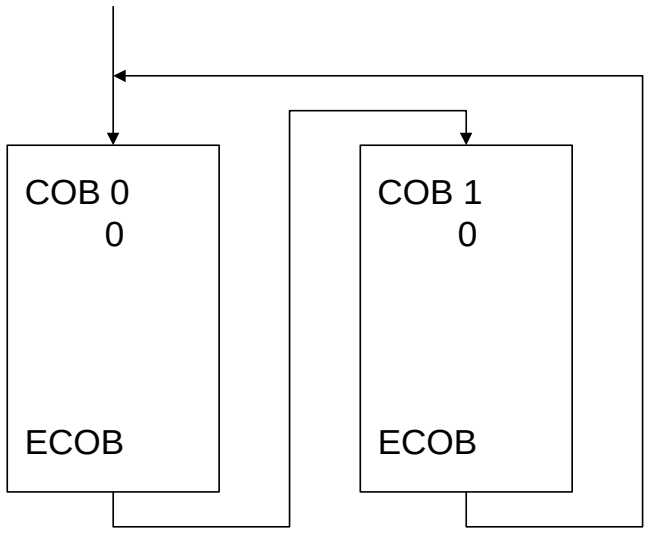
W sterownikach serii Simatic S5 (również i Simatic S7) firmy Siemens program dla sterownika składa się z mniejszych elementów oprogramowania zwanych blokami [4,48]. Bloki mogą być pięciu typów: bloki organizacyjne OB (*ang. Organisation Block*), bloki programowe PB (*ang. Program Block*), bloki funkcyjne FB (*ang. Function Block*), bloki sekwencyjne SB (*ang. Sequential Block*) i bloki danych DB (*ang. Data Block*). Na rysunku 1.3 przedstawiono przykładową strukturę programu sterowania składającą się z programu głównego OB1, trzech podprogramów (PB1, PB2, PB3) i dwóch bloków funkcyjnych (FB1 i FB2). W przedstawionym przykładzie wszystkie wywołania podprogramów i bloków funkcyjnych są bezwarunkowe (JU); instrukcja BE jest instrukcją końca bloku.



Rys.1.3. Przykład struktury programu dla sterowników Simatic S5 w oprogramowaniu Step 5

Każdy program musi posiadać przynajmniej blok organizacyjny OB1 który jest cyklicznie wywoływany przez system operacyjny. Pozostałe bloki organizacyjne są albo wywoływane przez system operacyjny po wystąpieniu określonego zdarzenia, np. OB22 – po załączeniu automatycznym przy powrocie napięcia, lub za ich pomocą wywoływane są pewne standardowe funkcje zintegrowane z systemem operacyjnym, np. OB251 – algorytm regulacji PID. Bloki PB i SB spełniają rolę podprogramów i normalnie są przywoływane w oprogramowaniu jednokrotnie. Bloki funkcyjne FB też spełniają rolę podprogramów ale mogą być wywoływane wielokrotnie i z różnymi argumentami [4].

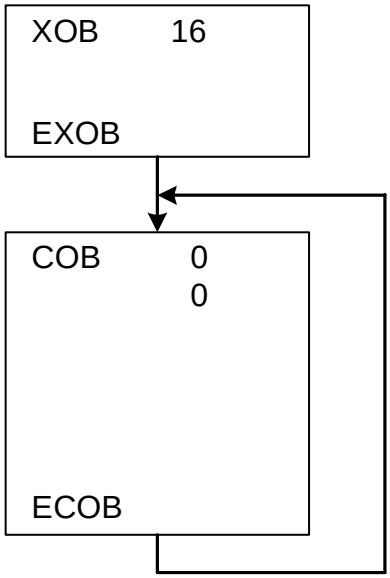
W sterownikach serii PCD firmy SAIA [51] program składa się z bardzo podobnych elementów oprogramowania jak w sterownikach Simatic S5. Są to: cykliczny blok organizacyjny COB (*ang. Cyclic Organization Block*), blok organizacyjny wyjątku XOB (*ang. Exception Organization Block*), blok programowy PB (*ang. Program Block*) i blok funkcyjny FB (*ang. Function Block*). Można zaprogramować więcej niż jeden blok COB (maksymalnie do 16); wszystkie te bloki są wykonywane w ustalonej kolejności w każdym cyklu programowym. Strukturę programu sterowania, gdy oprogramowane są dwa bloki COB przedstawiono na rysunku 1.4.



Rys.1.4. Dwa bloki COB (program Bloctek – SAIA)

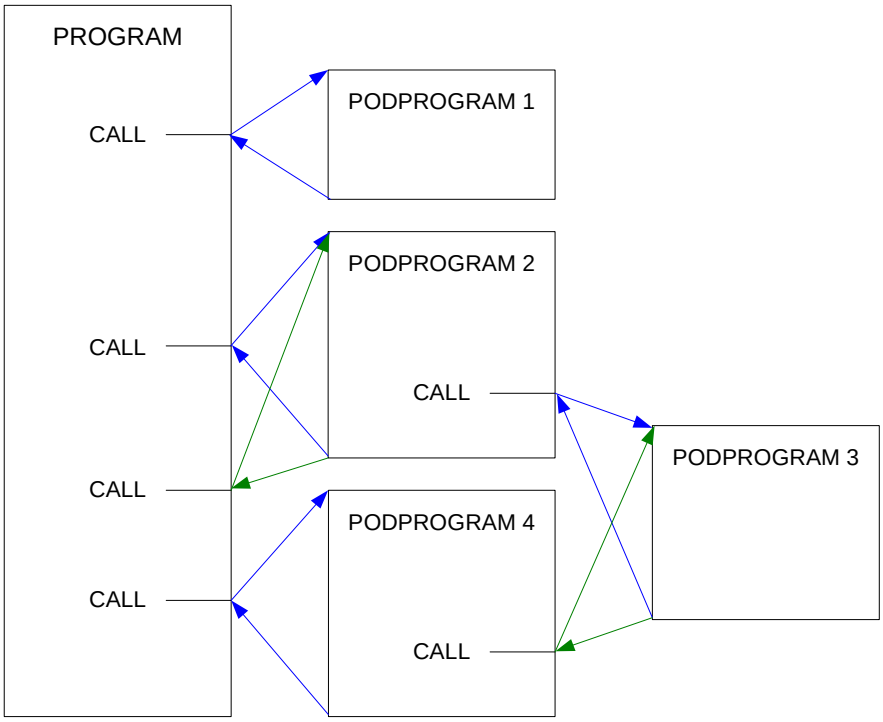
Bloki XOB są wywoływane w przypadku wystąpienia określonego zdarzenia (błędu sprzętowego lub programowego), np. XOB1 – gdy brak jest zasilania kasety rozszerzenia. Inne przeznaczenie ma tylko blok XOB16, który jest wykonywany jednokrotnie po uruchomieniu sterownika, a dopiero po nim wykonywane są cyklicznie bloki COB. Strukturę programu, gdy oprogramowany jest blok XOB16 i jeden blok COB, przedstawiono na

rysunku 1.5. Bloki PB i FB są podprogramami, które mogą być wywoływane warunkowo lub bezwarunkowo, a różnica pomiędzy nimi polega tylko na tym, że FB może być wywoływany z różnymi parametrami.



Rys.1.5. Struktura programu z blokiem XOB16 i jednym blokiem COB

W sterownikach GE Fanuc serii 90-30 (również i VersaMax) oprogramowanie użytkownika może składać się z programu głównego i podprogramów [50]. W odróżnieniu od wcześniej omówionych przykładów dla innych sterowników, podprogramy w sterownikach GE Fanuc nie dzielą się na różne rodzaje o ściśle określonych funkcjach. Podprogramy mogą być wywoływane (bezwarunkowo lub warunkowo) z programu głównego lub też z innych podprogramów. Podprogram może być wywoływany dowolną liczbę razy podczas wykonywania programu i dopuszczalne są także wywołania rekurencyjne. Przy wywołaniach podprogramu z programu głównego i podprogramów dopuszczalnych jest osiem poziomów zagnieżdżeń (a właściwie siedem, gdyż poziom programu głównego jest liczony jako poziom pierwszy). Na rysunku 1.6 pokazano przykładową strukturę programu sterowania składającą się z programu głównego i czterech podprogramów (zagnieżdżenia do poziomu trzeciego). Dwa podprogramy są wywoływane dwukrotnie: podprogram drugi dwa razy z programu głównego, a podprogram trzeci raz z podprogramu drugiego, a raz z podprogramu czwartego. Jeśli wszystkie wywołania są bezwarunkowe, to w każdym cyklu pracy sterownika podprogram trzeci jest w rzeczywistości wywoływany trzy razy – ponieważ podprogram drugi jest wywoływany dwukrotnie.



Rys.1.6. Przykładowa struktura programu w sterownikach GE Fanuc

Innym sposobem strukturyzacji oprogramowania w sterownikach programowalnych, oprócz wykorzystywania podprogramów, jest programowanie w językach graficznych takich jak np. język GRAFCET, najbardziej popularny w sterownikach TSX firmy Schneider Electric [42], czy też programowanie w bardzo podobnym do GRAFCET-u języku GRAFTEC stosowanym w sterownikach PCD firmy SAIA [51]. Języki te pozwalają na opisanie systemu odnoszącego się do procesu technologicznego lub urządzenia pracującego sekwencyjnie tj. krok po kroku. Diagram GAFCET (lub GRAFTEC) wykorzystuje ograniczoną liczbę prostych symboli (kroków zawierających działania, przejść zawierających warunki oraz połączeń pomiędzy krokami i przejściami) i kieruje się zbiorem określonych reguł. Ponieważ język GRAFCET (i GRAFTEC) steruje tylko sekwencją sterowań, to w celu opisania działań zawartych w krokach i warunków przejść należy użyć innego języka programowania (najczęściej IL, LD lub FBD). Przykłady sterowań sekwencyjnych i odpowiadających im diagramów języka GRAFCET omówiono w rozdziale 2.

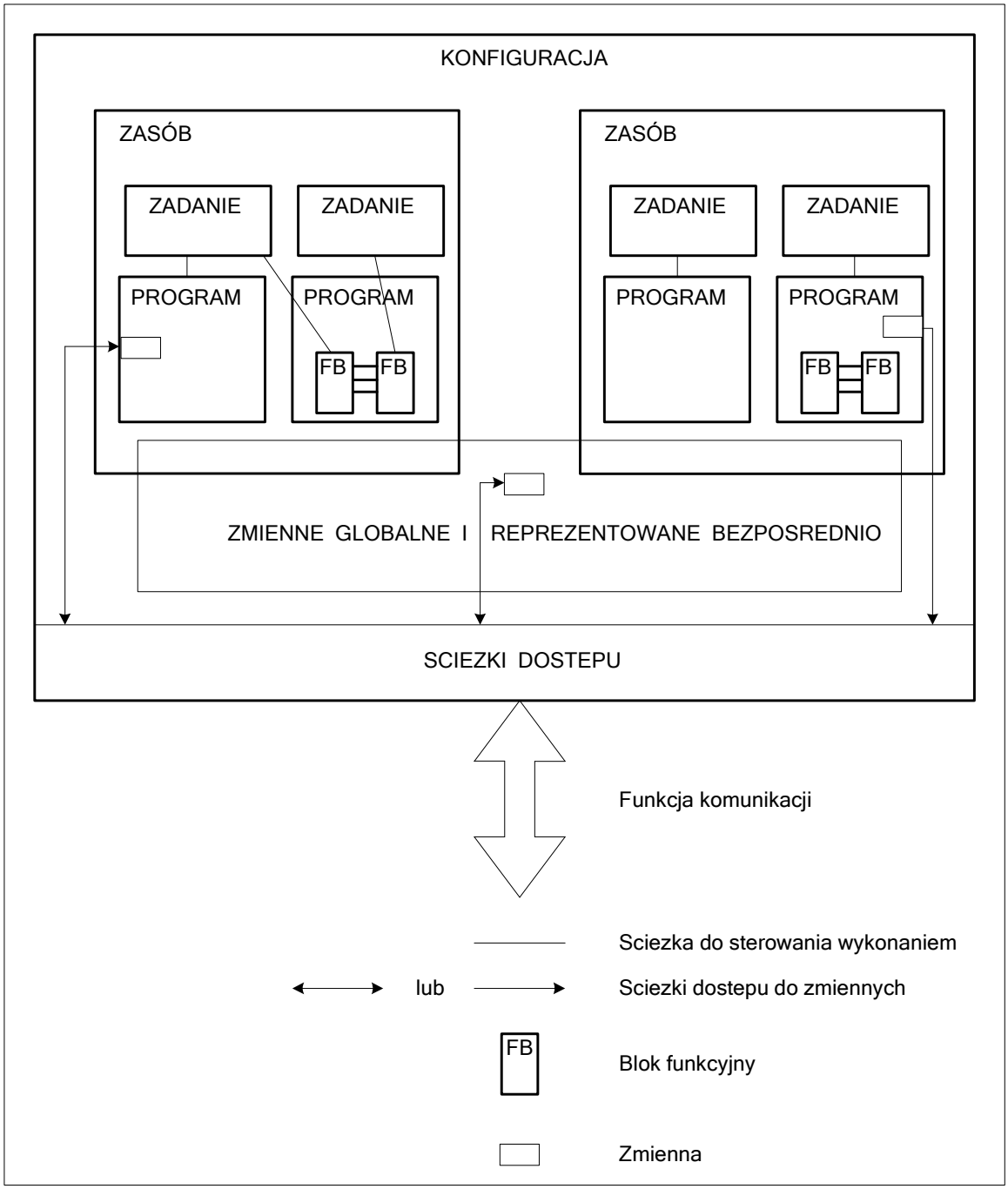
Cechą wspólną przedstawionych powyżej jak również i innych powstałych na drugim etapie programów narzędziowych jest konieczność przyporządkowywania zmiennym adresów w pamięci sterownika. Przekazywanie danych pomiędzy programem głównym a podprogramami lub pomiędzy podprogramami, czy też ogólnie elementami oprogramowania, jest realizowane poprzez wspólne obszary znacznikowe (obszary pamięci).

1.4. Strukturyzacja zgodnie z normą IEC 61131-3 – etap trzeci

Trzeci etap w procesie zmian sposobów tworzenia oprogramowania jest ściśle związany z utworzeniem organizacji PLCopen zrzeszającej producentów sterowników programowalnych i programów narzędziowych do nich, oraz opracowaniem i wprowadzeniem w 1993 roku międzynarodowej normy IEC 1131 (po aktualizacji 61131) [40] której część 3 dotyczy języków programowania [41]. Norma ta definiuje pojęcia podstawowe, zasady ogólne, model programowy i model komunikacyjny oraz podstawowe typy i struktury danych. Określono w niej dwie grupy języków programowania: tekstowe i graficzne. Do języków tekstowych należą: lista instrukcji IL (*ang. Instruction List*) i język strukturalny ST (*ang. Structured Text*). Do języków graficznych należą: schemat drabinkowy LD (*Ladder Diagram*), język schematów blokowych FBD (*ang. Function Block Diagram*) i graf sekwencji SFC (*ang. Sequential Function Chart*). W normie wyróżniono następujące elementy organizacyjne oprogramowania POU (*ang. Program Organization Units*): funkcje F (*ang. Functions*), bloki funkcyjne FB (*ang. Function Blocks*) i programy P (*ang. Programs*). Zdefiniowano również model oprogramowania, na który, oprócz elementów organizacyjnych oprogramowania składają się elementy konfiguracji; są to: konfiguracje (*ang. Configurations*), zasoby (*ang. Resources*), zadania (*ang. Tasks*), zmienne globalne (*ang. Global Variables*) i ścieżki dostępu (*ang. Access Paths*). Model oprogramowania według normy międzynarodowej przedstawiono na rysunku 1.7. Obecnie jest to również norma europejska (EN 61131-3) i polska (PN-EN 61131-3:2004) [41].

Tak więc strukturyzacja oprogramowania jest ściśle określona przez model oprogramowania i model komunikacyjny, a na ograniczenie złożoności i poprawę przejrzystości struktury i zasady działania programu wpływa stosowanie funkcji i bloków funkcyjnych. Ważna jest również wprowadzona możliwość definiowania zmiennych reprezentowanych symbolicznie, a nie tylko zmiennych reprezentowanych bezpośrednio. Przekazywanie danych pomiędzy poszczególnymi elementami oprogramowania, oprócz wykorzystywania wspólnych obszarów znacznikowych, może odbywać się także za pomocą zmiennych globalnych, funkcji komunikacyjnych i ścieżek dostępu. Równie ważna, z punktu widzenia wygody programisty i strukturyzacji programów sterowania, jest możliwość prostego tworzenia bloków funkcyjnych użytkownika. Bloki takie, w odróżnieniu od gotowych bloków funkcyjnych dostarczanych przez producentów oprogramowania, mogą realizować nie tylko przetwarzanie sygnałów związane z algorytmem sterowania, ale również mogą realizować funkcje związane z obsługą urządzeń współpracujących ze sterownikiem w

układzie sterowania. Przykład takiego urządzenia i obsługującego go bloku funkcyjnego użytkownika przedstawiono w następnym podrozdziale.



Rys.1.7. Model oprogramowania według normy IEC 61131-3

1.5. Tworzenie funkcji i bloków funkcyjnych użytkownika

Kompaktowe sterowniki programowalne oprócz wejść i wyjść binarnych często posiadają również niewielką liczbę wejść i wyjść analogowych. Przykładowo, sterowniki PS4-141-MM1 i PS4-201-MM1 (pokazany na rysunku 1.8) firmy Moeller [1,2] posiadają po

dwa wejścia analogowe napięciowe (0÷10V) i jedno wyjście analogowe napięciowe (0÷10V). Nawet w niezbyt rozbudowanych układach sterowania ta liczba wejść analogowych sterownika często okazuje się niewystarczająca. Można wówczas sterownik rozbudować o moduły rozszerzenia lokalnego LE4 (np. LE4-206-AA1 z czterema wejściami analogowymi napięciowymi, LE4-206-AA2 z czterema wejściami analogowymi prądowymi) lub o moduły rozszerzenia zewnętrznego EM4 (np. EM4-101-AA2 z czterema wejściami analogowymi napięciowymi i czterema prądowymi, EM4-101-TX1 z sześcioma wejściami do czujników Pt100/Ni1000 i dwoma wejściami analogowymi napięciowymi).

Rys.1.8. Sterownik programowalny PS4-201-MM1

W przypadku, gdy sygnały analogowe w układzie sterowania są sygnałami wolnozmiennymi (np. pomiary temperatury, odczytywanie nastaw parametrów z zadajników potencjometrycznych) można nie rozbudowywać układu sterowania o dodatkowe moduły EM4 i/lub LE4, a zastosować rozwiązanie tańsze polegające na multipleksowaniu sygnałów analogowych [36,38].

Przedstawiony na rysunku 1.9 multiplekser sygnałów analogowych MSA-08 posiada osiem kanałów wejściowych i jest przewidziany do współpracy ze sterownikiem programowalnym. Wszystkie kanały wejściowe multipleksera MSA-08 posiadają separację galwaniczną, są konfigurowane za pomocą mikroprzełączników na różne rodzaje i zakresy sygnałów wejściowych (napięciowe 0÷10V i ±10V oraz prądowe 0-20mA i 4-20mA) i są przetwarzane na sygnał wyjściowy napięciowy 0÷10V – taki sam jak zakres na wejściu analogowym sterownika. Wybór adresu multipleksowanego kanału jest realizowany poprzez podanie na wejścia adresowe (A0,A1,A2) multipleksera sekwencji sygnałów napięciowych 24V DC –

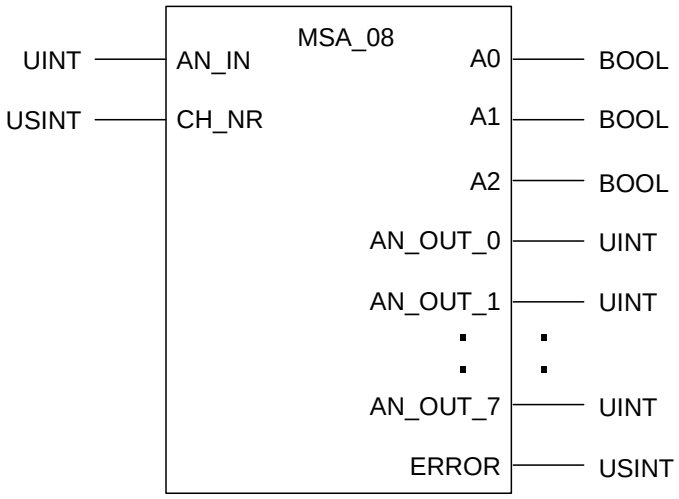
czyli bezpośrednio z tranzystorowych wyjść binarnych 24V DC sterownika. Sygnały z wejścia analogowego po przetworzeniu A/C muszą być w sterowniku demultipleksowane programowo na taką samą liczbę kanałów wyjściowych jak liczba wykorzystywanych w układzie sterowania kanałów wejściowych multipleksera.

Rys.1.9. Multiplekser sygnałów analogowych MSA-08

Do obsługi multipleksera i demultipleksowania sygnałów w sterowniku stworzono, w zgodnym z normą IEC 61131-3 programie narzędziowym Sucosoft S40, przedstawiony na rysunku 1.10 blok funkcyjny użytkownika o nazwie MSA_08 z następującymi zmiennymi wejściowymi i wyjściowymi:

AN_IN	Wejście analogowe sterownika na które podawany jest sygnał z wyjścia multipleksera MSA-08;
CH_NR	Liczba multipleksowanych kanałów MSA-08 (obsługiwane są kolejne kanały licząc od kanału 0);
A0	Bit 0 adresu wyboru kanału MSA-08;
A1	Bit 1 adresu wyboru kanału MSA-08;
A2	Bit 2 adresu wyboru kanału MSA-08;
AN_OUT_0	Wartość cyfrowa sygnału analogowego z kanału 0 multipleksera MSA-08;
AN_OUT_1	Wartość cyfrowa sygnału analogowego z kanału 1 multipleksera MSA-08;
.....	
AN_OUT_7	Wartość cyfrowa sygnału analogowego z kanału 7 multipleksera MSA-08;
ERROR	Kod błędu (0 - nie ma błędu, 1 - niepoprawnie zadana liczba kanałów).

Dzięki temu obsługa programowa multipleksera i demultipleksowanie sygnałów w sterowniku są realizowane w programie sterowania przez sprawdzony, poprawnie działający, blok funkcyjny użytkownika.



Rys.1.10. Blok funkcyjny użytkownika MSA-08

Przez cały czas gdy sterownik jest w trybie RUN obsługiwana jest zadana liczba kanałów; czas potrzebny na obsługę 1 kanału jest równy czasowi dwóch cykli programu.

Ponieważ rozpatrywane sterowniki posiadają dwa wejścia analogowe i wystarczającą liczbę wyjść binarnych, to można do nich podłączyć dwa multipleksery MSA-08. W celu wykorzystania drugiego multipleksera wymagane jest zadeklarowanie drugiego bloku typu MSA_08 (drugie ukonkretnienie bloku).

1.6. Podsumowanie

Przedstawione w tym rozdziale trzy podstawowe etapy w procesie zmian sposobów tworzenia oprogramowania pokazują, że zmiany te prowadzą w kierunku unifikacji i zwiększenia możliwości strukturyzacji programów sterowania, oraz ujednolicenia programów narzędziowych. Poniżej przedstawiono zestawienie najważniejszych cech charakterystycznych poszczególnych etapów, a także możliwości strukturyzacji programów sterowania na każdym z trzech wyróżnionych etapów.

Etap pierwszy

Cechy charakterystyczne:

- zmienne reprezentowane bezpośrednio;
- adresy skoków do konkretnych numerów linii programu;

- cały program stanowił jeden element oprogramowania.

Możliwości strukturyzacji programów sterowania:

- uporządkowanie sekwencji programowych zgodnie z algorytmem sterowania;
- wykorzystywanie instrukcji skoków warunkowych i bezwarunkowych;
- rezerwowanie, pod ewentualne zmiany, fragmentów pamięci programu za pomocą instrukcji NOP;
- stosowanie specjalnych instrukcji włączających warunkowo fragmenty programu do analizy w bieżącym cyklu (np. MLS i MLR).

Etap drugi

Cechy charakterystyczne:

- symboliczne nazwy zmiennych;
- etykiety sekwencji lub bloków programowych jako adresy skoków;
- bezpośrednia reprezentacja zmiennych w pamięci sterownika;
- wymiana danych pomiędzy elementami oprogramowania poprzez wspólne obszary znacznikowe.

Możliwości strukturyzacji programów sterowania:

- możliwość tworzenia podprogramów;
- podział programu sterowania na bloki (podprogramy) różnych typów;
- wprowadzenie języków graficznych do programowania.

Etap trzeci

Cechy charakterystyczne:

- wprowadzenie normy międzynarodowej IEC 1131-3 (61131-3);
- zdefiniowanie modelu programowego i komunikacyjnego, podstawowych typów i struktur danych oraz języków programowania;
- możliwość bezpośredniej i symbolicznej reprezentacji zmiennych w pamięci sterownika;
- wymiana danych poprzez zmienne globalne, funkcje komunikacyjne i ścieżki dostępu;
- postępująca unifikacja programów narzędziowych.

Możliwości strukturyzacji programów sterowania:

- strukturyzacja oprogramowania wynikająca z normy;
- łatwość tworzenia funkcji (F) i bloków funkcyjnych użytkownika (FB).

Podsumowując można stwierdzić, że zagadnienie strukturyzacji programów sterowania na sterowniki programowalne jest jednym z ważniejszych zagadnień ich programowania, gdyż od tego w głównej mierze zależy, czy stworzony program będzie zrozumiały, sprawdzalny,

łatwy w uruchomieniu, modyfikowalny, i czy stworzone przy tym poszczególne elementy oprogramowania (głównie bloki funkcyjne użytkownika) będą nadawały się do wielokrotnego użycia.

Dzięki stosowaniu bloków funkcyjnych użytkownika tworzenie programów jest szybsze i łatwiejsze. Bloki funkcyjne redukują złożoność programu (dzięki ukryciu złożoności wewnętrznej algorytmu bloku funkcyjnego), a raz utworzone mogą być używane w programie wielokrotnie (wielokrotne ukonkretnienie).

W rozdziale tym pominięto omawianie elementów oprogramowania wywoływanych w wyniku przerwań, gdyż powinny być one omawiane łącznie z zagadnieniami czasowymi (czas jednego cyklu programu i jego podział, czas reakcji układu sterowania), a nie zmieniają one zasadniczo ani struktury programu sterowania, ani metod strukturyzacji programów sterowania.

2. Sterowanie procesami sekwencyjnymi

2.1. Układy logiczne i ich rodzaje

Układ logiczny (często nazywany również dyskretnym) jest to układ w którym zarówno sygnały wejściowe jak i wyjściowe są zmiennymi binarnymi (przyjmują tylko dwie wartości: 0 i 1). Układy logiczne dzielą się na kombinacyjne i sekwencyjne.

W układach kombinacyjnych aktualny stan ich wyjść zależy jedynie od aktualnego stanu wejść.

W układzie kombinacyjnym wyróżnia się:

- skończony zbiór sygnałów wejściowych $X = \{ X_1 , X_2 , ... , X_n \}$;
- skończony zbiór sygnałów wyjściowych $Y = \{ Y_1 , Y_2 , ... , Y_m \}$.

Działanie układu kombinacyjnego opisywane jest przez funkcję λ wyjść układu

$$Y^t = \lambda(X^t) \ ,$$

gdzie indeks górny t odnosi się do pewnej, tej samej chwili czasu t.

W układach sekwencyjnych aktualny stan ich wyjść zależy nie tylko od aktualnego stanu wejść ale również od poprzednich stanów wejść. Zależność między sygnałami wyjściowymi i sygnałami wejściowymi nie jest więc jednoznaczna. Tym samym sygnałom wejściowym mogą odpowiadać różne sygnały wyjściowe, zależnie od stanu w jakim układ znajdował się poprzednio. Zależność aktualnego stanu układu sekwencyjnego od jego stanu w chwili poprzedniej jest realizowana za pomocą elementów pamięci. Stan elementów pamięci jest nazywany stanem wewnętrznym układu.

W układzie sekwencyjnym wyróżnia się więc:

- skończony zbiór sygnałów wejściowych $X = \{ X_1 , X_2 , ... , X_n \}$;
- skończony zbiór sygnałów wyjściowych $Y = \{ Y_1 , Y_2 , ... , Y_m \}$;
- skończony zbiór stanów wewnętrznych $Q = \{ Q_1 , Q_2 , ... , Q_k \}$.

Działanie układu sekwencyjnego opisywane jest przez funkcję δ przejść układu

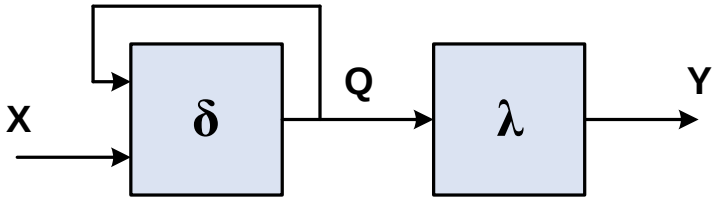
$$Q^t = \delta(Q^{t-1}, X^t) \ ,$$

gdzie indeks t-1 odnosi się do poprzedniej chwili czasu w której nastąpiła zmiana stanu wewnętrznego układu, oraz, w zależności czy jest to układ sekwencyjny Moore'a czy Mealy'ego, przez odpowiednią funkcję λ wyjść układu:

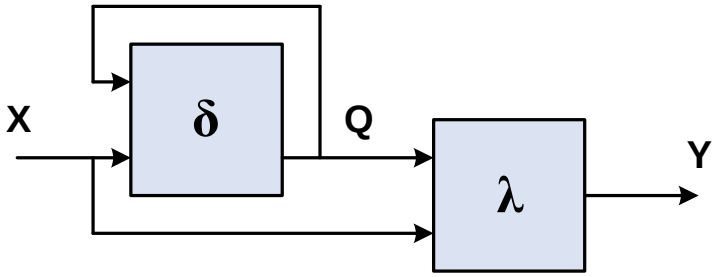
$$Y^t = \lambda(Q^t) \qquad \text{dla układu Moore'e ,}$$

lub $Y^t = \lambda(Q^t, X^t)$ dla układu Mealy'ego .

Schematy blokowe układów sekwencyjnych Moore'a i Mealy'ego składające się z układu pamięci opisanego funkcją przejść δ i układu kombinacyjnego opisanego funkcją wyjść λ przedstawiono na rysunkach 2.1 i 2.2.



Rys.2.1. Schemat blokowy układu sekwencyjnego Moore'a



Rys.2.2. Schemat blokowy układu sekwencyjnego Mealy'ego

Wszystkie układy sekwencyjne dzielą się także na asynchroniczne i synchroniczne.

Układy sekwencyjne asynchroniczne to takie układy sekwencyjne w których zmiany stanów wewnętrznych układu mogą występować w dowolnych chwilach czasu określonych przez zmiany sygnałów wejściowych.

Układy sekwencyjne synchroniczne to takie układy sekwencyjne w których zmiany stanów wewnętrznych mogą występować tylko w ściśle określonych, dyskretnych chwilach czasu wyznaczonych zmianami dodatkowego sygnału taktującego.

W przemyśle zdecydowana większość układów sterowania pojedynczych urządzeń, linii produkcyjnych i procesów technologicznych może być zakwalifikowana jako układy sterowania sekwencyjnego asynchronicznego [30]. Z tego względu dalsza część tego rozdziału poświęcona jest sposobom opisu, modelowania i programowania układów sterowania procesów sekwencyjnych.

2.2. Metody opisu i realizacji algorytmów sterowania sekwencyjnego

Do opisu dyskretnych procesów sekwencyjnych najczęściej wykorzystuje się metody graficzne przedstawienia algorytmów sterowania sekwencyjnego. Cechą charakterystyczną tych metod jest to, że wymagają one wcześniejszego zamodelowania działania urządzenia lub realizacji procesu technologicznego w procesie sterowania. Poprzez modelowanie rozumiane jest modelowanie działania układu, a nie budowa modelu komputerowego programowanego systemu sekwencyjnego.

Zgodnie z normą międzynarodową IEC 61131-3 [41] dotyczącą programowania sterowników jednym z języków programowania jest język graficzny SFC (*ang. Sequential Function Chart*). Jest to język używający kroków (etapów) i przejść (warunków przejść) w celu przedstawienia algorytmu programu sterowania procesem sekwencyjnym. Język graficzny SFC został oparty na języku GRAFCET (*fr. Graphe de Commande Etape – Transition*) [11,42] opisanym we francuskiej normie NF-C-3-190 z 1978 roku. W 1988 roku organizacja IEC (*ang. International Electrical Commission*) adaptowała metodę GRAFCET jako międzynarodowy standardowy język do programowania procesów sekwencyjnych; oznaczono go symbolem IEC 848 – „Preparation of function charts for control systems". Norma ta została następnie powtórzona w normie IEC 61131-3 właśnie jako język SFC.

Język GRAFCET został opisany w podrozdziale 2.3, a w podrozdziale 2.4 zamieszczono i omówiono przykład wykorzystania tego języka do przedstawienia algorytmu sterowania prasą hydrauliczną, będącą jednym z urządzeń linii produkcyjnej.

Metoda GRAFCET posiada wiele elementów wspólnych, stosowanych do opisu sterowania procesami sekwencyjnymi, z niemiecką normą DIN 40719. Główna różnica pomiędzy metodą GRAFCET a metodą DIN 40719 polega na tym, że w normie niemieckiej warunki przejścia nie są przyporządkowane do konkretnego połączenia pomiędzy etapami, lecz do każdego z etapów. Warunki przejścia, zwane w normie DIN 40719 warunkami początkowymi, stanowią zatem integralną część każdego z etapów.

Należy zauważyć, że w oprogramowaniach narzędziowych różnych producentów sterowników programowalnych, języki graficzne do programowania procesów sekwencyjnych (bazujące na językach SFC i GRAFCET) mogą mieć różne nazwy, jak np.: GRAFTEC dla sterowników firmy SAIA, MEGRAF dla sterowników firmy Mitsubishi, czy GRAPH 5/II dla sterowników firmy Siemens.

Przykładami innych metod opisu dyskretnych procesów sekwencyjnych i sterowania nimi są też: sieć operacyjna ZPT (zautomatyzowanych procesów technologicznych) [28] i sieć logiczna GRAFPOL [29].

Do programowania sterowania procesami sekwencyjnymi niekoniecznie trzeba wykorzystywać graficzne języki programowania takie jak SFC i GRAFCET. Niekiedy wystarczające jest wykorzystanie możliwości, jakie dają trzy podstawowe języki programowania sterowników, czyli: lista instrukcji (*ang. Instruction List - IL*), schemat drabinkowy (*ang. Ladder Diagram – LD*) i schemat bloków funkcji (*ang. Function Block Diagram – FBD*). A możliwości te, to np. wykorzystanie instrukcji S i R (Set i Reset), wykorzystanie bloków funkcyjnych rejestrów przesuwnych (dla zmiennych binarnych) jako elementów przyporządkowujących i pamiętających. Niekiedy można też wykorzystać specjalne bloki funkcyjne sterowania sekwencyjnego, jak np. SK w sterownikach firmy Moeller czy SB w sterownikach firmy Siemens.

W podrozdziale 2.5 pokazano, w jaki sposób proces sterowania sekwencyjnego, zamodelowany za pomocą grafu skierowanego, może być oprogramowany wykorzystując do tego celu instrukcje SET i RESET.

2.3. Język GRAFCET i jego charakterystyka

Język GRAFCET jest jednym z języków pozwalających na opisanie systemu odnoszącego się do urządzenia lub procesu technologicznego pracującego sekwencyjnie tj. krok po kroku. Może być stosowany w przemysłowych systemach sterowania niezależnie od wykorzystywanej technologii: elektryczna, elektromechaniczna, pneumatyczna itp., gdyż norma francuska opisuje tylko funkcjonalne przedstawienie systemu sterowania nie opisując programowania.

Celem języka GRAFCET jest standaryzacja funkcjonalna i graficzna przedstawienia systemu sterowania. W swoich zastosowaniach diagram GRAFCET wykorzystuje ograniczoną liczbę prostych symboli i kieruje się zbiorem określonych reguł. Na rysunku 2.3 przedstawiono elementy języka GRAFCET z których tworzony jest diagram GRAFCET.

Diagram GRAFCET składa się z:

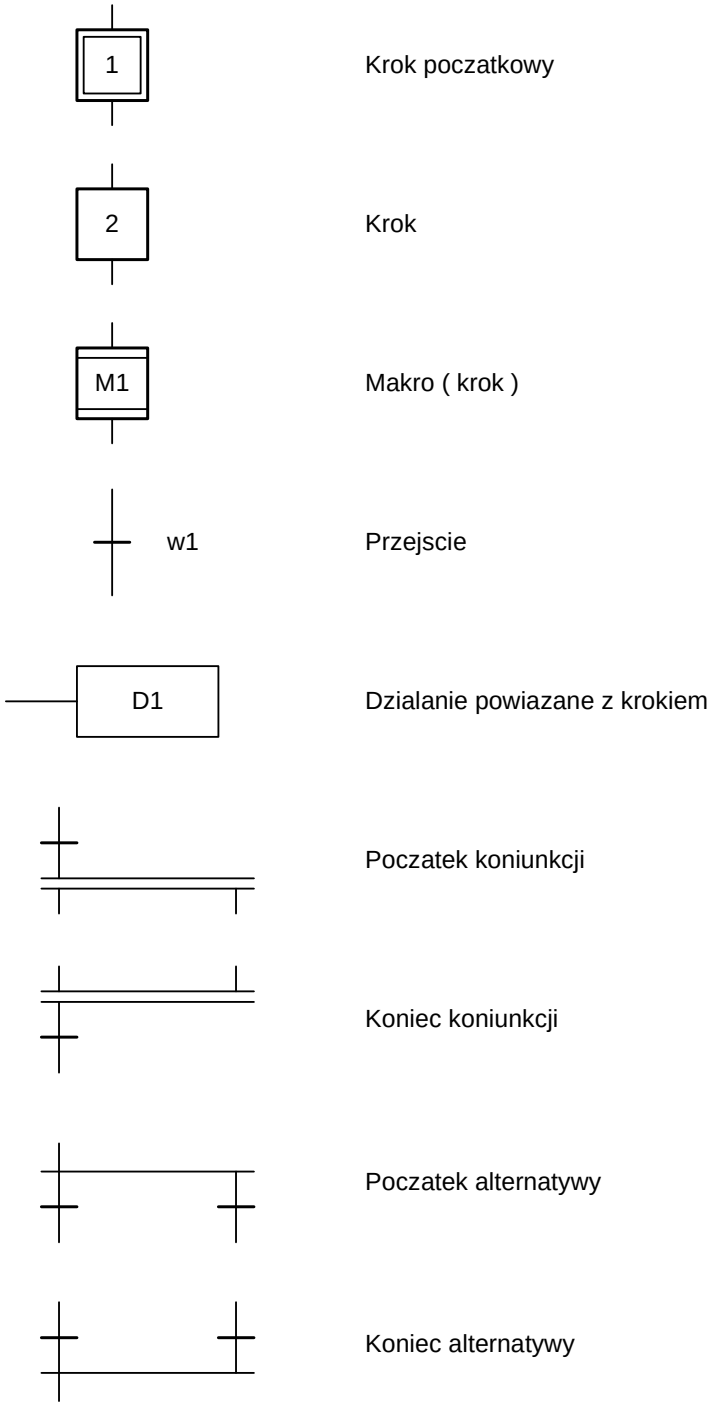
- kroków - które zawierają działania;
- przejść - które zawierają warunki;
- połączeń pomiędzy krokami i przejściami .

Przejście jest realizowane tylko wówczas, gdy:

- krok bezpośrednio przed nim jest aktywny;
- warunek przejścia jest spełniony (prawdziwy).

Realizacja przejścia powoduje dezaktywację kroku (kroków), który wcześniej był aktywny, i aktywację kroku (kroków), do którego prowadzi przejście, którego warunek przejścia został

spełniony. Następstwo kroków i przejść, które muszą występować naprzemiennie, tworzy sekwencję diagramu.

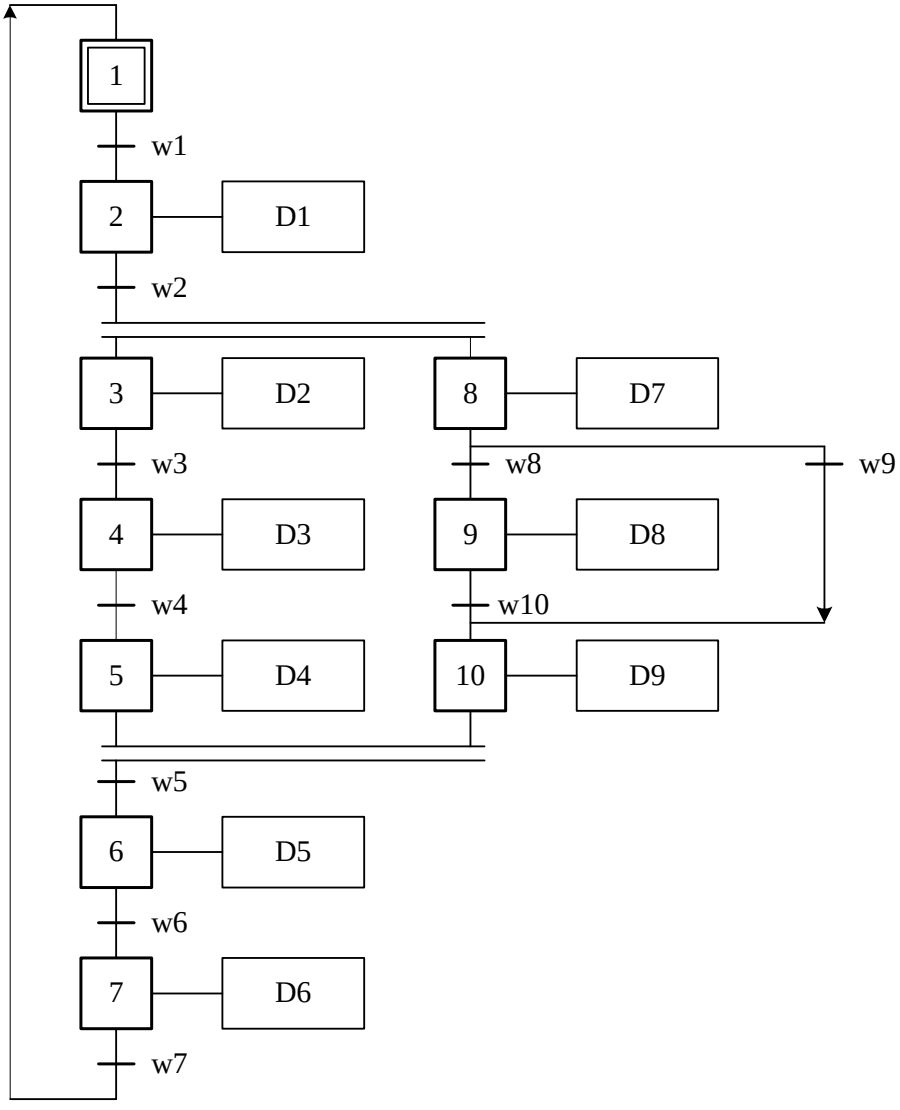


Rys.2.3. Symbole języka GRAFCET

Jeśli wykonanie przejścia prowadzi do uaktywnienia kilku sekwencji mówi się o równoczesnym lub równoległym rozgałęzieniu. Po równoczesnym uruchomieniu sekwencje te są wykonywane równolegle, jedna niezależnie od drugiej.

Jeśli wykonanie kroku prowadzi do rozgałęzienia na kilka przejść mówi się o alternatywnym rozgałęzieniu. Wówczas wykonywana jest tylko jedna gałąź, ta mianowicie, dla której warunek początkowego przejścia został spełniony jako pierwszy, analizując warunki przejścia od lewej strony diagramu do prawej.

Powyższe, skrótowo przedstawione reguły tworzenia diagramu GRAFCET i reguły ewolucyjne, określające przechodzenie w diagramie pomiędzy krokami, związane z ich aktywacją i dezaktywacją, można zilustrować na przykładzie diagramu GRAFCET przedstawionego na rysunku 2.4.



Rys.2.4. Przykładowy diagram języka GRAFCET (oznaczenia: 1÷10 – kroki, w1÷w10 – przejścia, D1÷D9 – działania powiązane z krokami)

W tym diagramie jest w sumie 10 kroków (w tym jeden krok początkowy oznaczony numerem 1) oraz 10 przejść zawierających warunki przejść oznaczone na rysunku 2.4 od w1

do w10. Z założenia połączenia pomiędzy krokami przebiegają z góry do dołu diagramu; w przypadkach nieoczywistych lub innych połączeniach, kierunek połączenia może być oznaczony strzałką. Na rysunku 2.4 dwa takie połączenia oznaczono strzałkami.

Z krokami, za wyjątkiem kroku początkowego, który jest zazwyczaj krokiem spoczynkowym, są powiązane określone działania – na rysunku oznaczone od D1 do D9. Działania te są wykonywane wówczas, gdy krok z którym są powiązane jest aktywny, i przestają być wykonywane wówczas, gdy krok przestaje być aktywny.

Możliwość aktywacji kroków zależy od przejść je poprzedzających i aktualnej aktywności kroków w diagramie. Przykładowo, jeśli aktywny jest krok 1 i spełniony jest warunek przejścia w1 to nastąpi przejście do kroku 2, czyli dezaktywacja kroku 1 i aktywacja kroku 2. Po spełnieniu warunku przejścia realizacja przejścia jest obligatoryjna i natychmiastowa.

W diagramie warunki przejść mogą się powtarzać, ale nie mogą być identyczne dla dwóch kolejnych przejść, gdyż prowadziłoby to do pominięcia zawartego pomiędzy nimi kroku. Przykładowo, gdyby warunki w3 i w4 były identyczne, to krok 4 byłby zawsze pomijany i nigdy nie byłyby realizowane działania D3 powiązane z tym krokiem; taka sytuacja byłaby błędna i nie może mieć miejsca.

W diagramie oprócz połączeń prostych (typu: krok – przejście – krok –przejście itd.) są również połączenia wielokrotne.

Gdy aktywny jest krok 2 i spełniony jest warunek w2 znajdującego się za nim przejścia, to nastąpi dezaktywacja kroku 2 i równoczesna aktywacja kroków 3 i 8 (synchronizacja sterowania) co na diagramie pokazane jest równoległymi liniami po warunku w2. Wyjście z tej sekwencji równoległej nastąpi wówczas, gdy będą aktywne kroki 5 i 10 oraz spełniony będzie warunek przejścia w5. Nastąpi wówczas dezaktywacja kroków 5 i 10 oraz aktywacja kroku 6.

Gdy aktywny jest krok 8, to kolejnym krokiem aktywnym może stać się albo krok 9 albo krok 10. Gdy spełniony będzie warunek przejścia w8 to nastąpi dezaktywacja kroku 8 i aktywacja kroku 9. Natomiast, gdy spełniony będzie warunek przejścia w9, to nastąpi dezaktywacja kroku 8 i aktywacja kroku 10 (krok 9 i działania z nim powiązane są omijane). Ponieważ jest to wybór jednego kroku z dwóch możliwych, to, aby sterowanie było określone, warunek przejścia w9 powinien być negacją warunku w8.

Jak widać z powyższego, w diagramie GRAFCET w danej chwili może być aktywny więcej niż jeden krok. W przykładzie z rysunku 2.4 aktywne mogą być równocześnie dwa kroki (jeden spośród kroków 3, 4, 5 i jeden spośród kroków 8, 9, 10).

W diagramie GRAFCET może być także więcej niż jeden krok początkowy, ale taki przypadek nie został na rysunku 2.4 pokazany.

Kilka kroków i przejść może być połączonych tworząc tzw. makro krok. Takie połączone kroki i przejścia powinny być funkcjonalnie powiązane, tworząc w sumie złożone zadanie (funkcję) sterowania. Makro krok zawsze rozpoczyna się i kończy krokiem, i dzięki temu może być umieszczony w diagramie GRAFCET tak samo jak zwykły pojedynczy krok z poprzedzającym go przejściem i z następującym po nim przejściem. Dzięki wykorzystaniu makro kroków można osiągnąć większą przejrzystość diagramu. Diagram taki odzwierciedla ogólną strukturę sterowania, podczas gdy szczegóły ukryte są w makro krokach (możliwe jest zagnieżdżanie – tj. umieszczanie makro kroków w makro krokach).

Język GRAFCET steruje tylko sekwencją sterowań. W celu opisania działań zawartych w krokach i warunków przejść należy użyć innego języka programowania np.: języka drabinkowego (LD) lub listy instrukcji (IL).

Organizacja sekcji programu w języku GRAFCET jest ściśle określona. Program (lub fragment programu) zapisany w języku GRAFCET można podzielić na trzy części:

- przetwarzanie wstępne;
- przetwarzanie sekwencyjne;
- przetwarzanie końcowe.

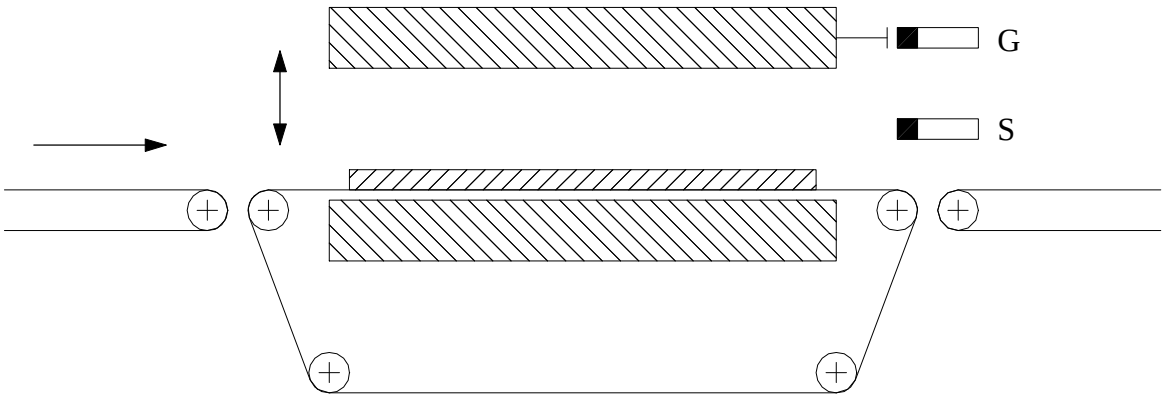
Przetwarzanie wstępne umożliwia przetwarzanie wszystkich zdarzeń związanych z zarządzaniem systemem w razie zaniku zasilania i podczas kolejnych inicjalizacji, oraz z kasowaniem lub wstępnym zadawaniem parametrów diagramu GRAFCET.

Przetwarzanie sekwencyjne jest główną częścią sekcji GRAFCET i służy do przetwarzania struktury sekwencyjnej programu sterowania urządzeniem lub procesem technologicznym. W czasie przetwarzania sekwencyjnego analizowane są warunki przejść i warunki powodujące dezaktywację lub uaktywnienie kroków, w dalszej kolejności aktualizowany jest stan diagramu GRAFCET zgodnie z aktualnie spełnionymi warunkami, a następnie wykonywane są akcje przyporządkowane aktywnym krokom.

Przetwarzanie końcowe obejmuje wykonanie akcji związanych bezpośrednio z aktualizacją stanów wyjść sterownika przypisanych do akcji dla poszczególnych kroków, aktualizację stanów wyjść nie związanych z przetwarzaniem sekwencyjnym programu oraz monitorowanie wykonywania diagramu GRAFCET poprzez kontrolę czasu aktywności wybranych kroków. W przetwarzaniu końcowym, podczas aktualizowania stanów wyjść, uwzględniane są również wszystkie zaprogramowane zabezpieczenia.

2.4. Przykład wykorzystania języka GRAFCET

Jako przykład rozpatrywane jest sterowanie fragmentem linii technologicznej z prasą hydrauliczną [34] przedstawioną na rysunku 2.5. Należy skonstruować diagram GRAFCET opisujący działanie prasy hydraulicznej w trybie pracy automatycznej linii technologicznej.

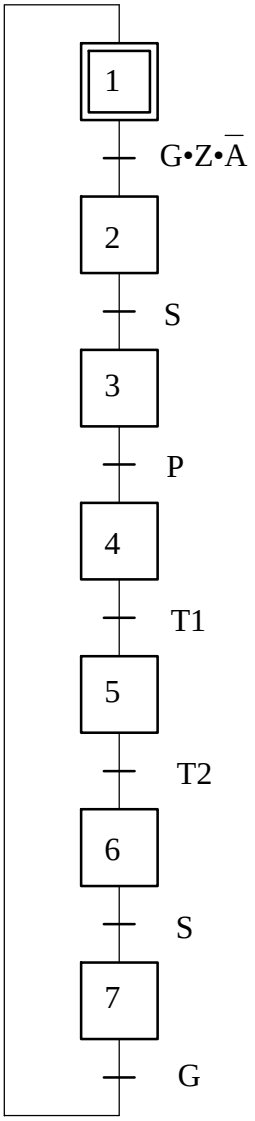


Rys.2.5. Fragment linii technologicznej z prasą hydrauliczną

Prasa jest otwarta, gdy jest sygnał z czujnika indukcyjnego G , tj. gdy ruchoma płyta prasy jest w górnym położeniu. Sygnał z czujnika indukcyjnego S określa pośrednie położenie górnej płyty prasy. Wartość ciśnienia jest odczytywana z analogowego czujnika ciśnienia z przetwornikiem pomiarowym. Pozostałe (nieistotne z punktu widzenia tworzonego diagramu) czujniki pomiarowe linii zostały pominięte.

Skonstruowany diagram GRAFCET przedstawiono na rysunku 2.6. Warunkiem wykonania cyklu prasowania w trybie pracy automatycznej jest górne położenie prasy (czujnik G), obecność płyty do prasowania i brak awarii linii. Cykl prasowania przedstawia się następująco: prasa otwarta (krok początkowy) - szybkie zamykanie prasy (do sygnału z czujnika S) - wolne zamykanie i sprężanie (do osiągnięcia ciśnienia prasowania P) - prasowanie (T1 - zadany czas prasowania) - dekompresja (T2 - zadany czas dekompresji) - wolne podnoszenie prasy (do sygnału z czujnika S) - szybkie podnoszenie (do sygnału z czujnika G) - prasa otwarta (oczekiwanie na wyładunek sprasowanej i załadunek nowej płyty). Jest to więc prosty diagram GRAFCET dla sterowanego urządzenia uwzględniający tylko normalne (poprawne) stany pracy prasy.

Rozpatrywany jest tylko proces prasowania. Nie rozpatruje się procesu załadunku elementów (płyt) do prasowania i procesu rozładunku elementów sprasowanych.



1 - prasa otwarta

2 - szybkie zamykanie

3 - wolne zamykanie i sprężanie

4 - prasowanie

5 - dekompresja

6 - wolne otwieranie

7 - szybkie otwieranie

(G•Z•$\overline{A}$) - prasa w górnym położeniu, załadowana płyta do prasowania i brak awarii

S - czujnik spowolnienia ruchu prasy

P - osiągnięto ciśnienie prasowania

T1 - upłynął czas prasowania

T2 - upłynął czas dekompresji

G - czujnik górnego położenia prasy

Rys.2.6. Diagram GRAFCET cyklu prasowania w trybie pracy automatycznej

Do przedstawienia procesu rozładunku / załadunku musiałby być stworzony oddzielny diagram GRAFCET. Załadunek i rozładunek mogą odbywać się równocześnie lub też kolejno: rozładunek – załadunek. Z punktu widzenia wydajności linii najbardziej pożądany jest równoczesny rozładunek i załadunek elementów. W przypadku, gdy odbyło się prasowanie i brak jest, lub nie jest gotowy do załadunku nowy element do prasowania, odbędzie się tylko rozładunek bez załadunku. Następnie, gdy jest już gotowy element do prasowania, a przenośnik w prasie jest pusty, odbywa się tylko załadunek. Przy opracowywaniu diagramu należałoby pamiętać o konieczności dokonywania rozładunku i załadunku przy takich samych prędkościach (ustalonych) przenośników przed prasą, w prasie i za prasą, oraz o załączaniu przenośnika przed prasą z opóźnieniem w stosunku do przenośnika w prasie. Opóźnienie to wynika z konieczności centrycznego załadunku elementu

do prasowania pod płytę prasy, i jest określane na podstawie różnicy w długościach przenośników w prasie i przed prasą.

Sygnałami dzięki którym procesy prasowania i załadunku / rozładunku są realizowane naprzemiennie są: sygnał Z – obecności załadowanego elementu do prasowania w diagramie GRAFCET opisującym prasowanie i analogiczny sygnał (element sprasowany), który byłby w warunku przejścia po kroku początkowym w diagramie GRAFCET opisującym załadunek / rozładunek. Dzięki tym sygnałom możliwe byłoby połączenie tych dwóch diagramów w jeden diagram GRAFCET dla procesu z dwoma krokami początkowymi: dla prasy i dla przenośników.

W diagramie przedstawionym na rysunku 2.6 nie pokazano działań (operacji) powiązanych z krokami. Należy jednakże mieć zawsze na uwadze, że z każdym krokiem – za wyjątkiem kroku początkowego, który odpowiada stanowi spoczynkowemu – są powiązane określone działania. Dla przedstawionego przykładu będą to odpowiednie sekwencje załączeń elektrozaworów w układzie hydraulicznym, dzięki którym realizowany jest cykl pracy prasy. Przykładowe sekwencje załączeń elektrozaworów dla prasy hydraulicznej z czterema elektrozaworami [33] przedstawiono w tabeli 2.1.

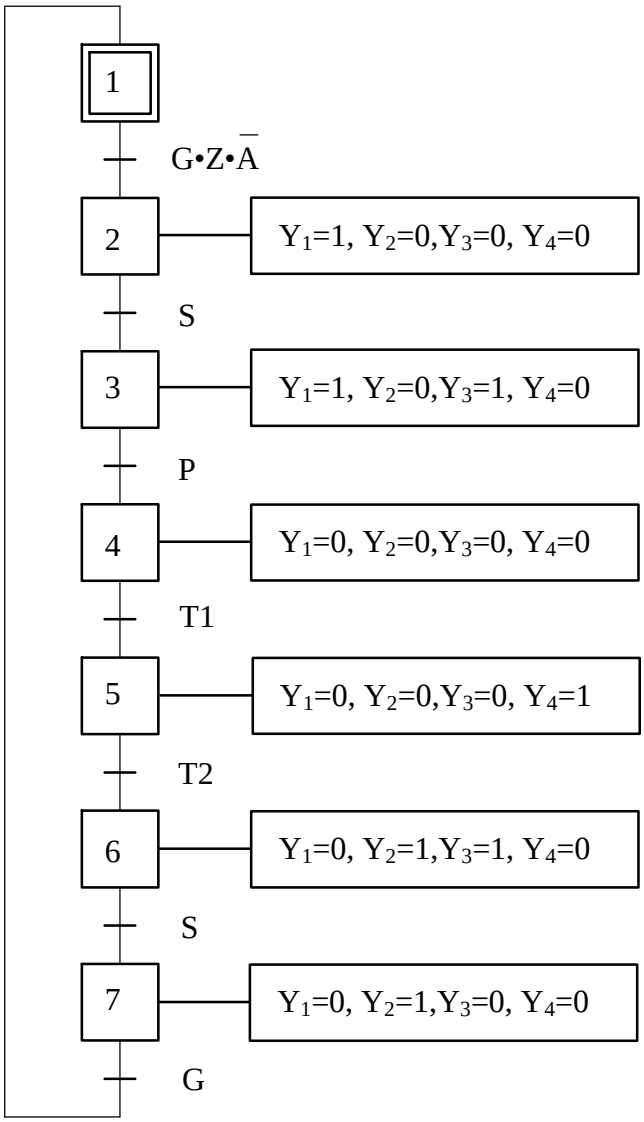
Tabela 2.1. Sterowanie elektrozaworami (Y_1, Y_2, Y_3, Y_4)

Etap / **Sterowanie**	**Y_1**	**Y_2**	**Y_3**	**Y_4**
Prasa otwarta	0	0	0	0
Szybkie zamykanie	1	0	0	0
Wolne zamykanie i sprężanie	1	0	1	0
Prasowanie	0	0	0	0
Dekompresja	0	0	0	1
Wolne otwieranie	0	1	1	0
Szybkie otwieranie	0	1	0	0

Informacja przedstawiona w formie tabeli może być również przedstawiona w diagramie GRAFCET. Na rysunku 2.7 przedstawiono diagram GRAFCET z rysunku 2.6, ale uzupełniony o działania powiązane z krokami diagramu.

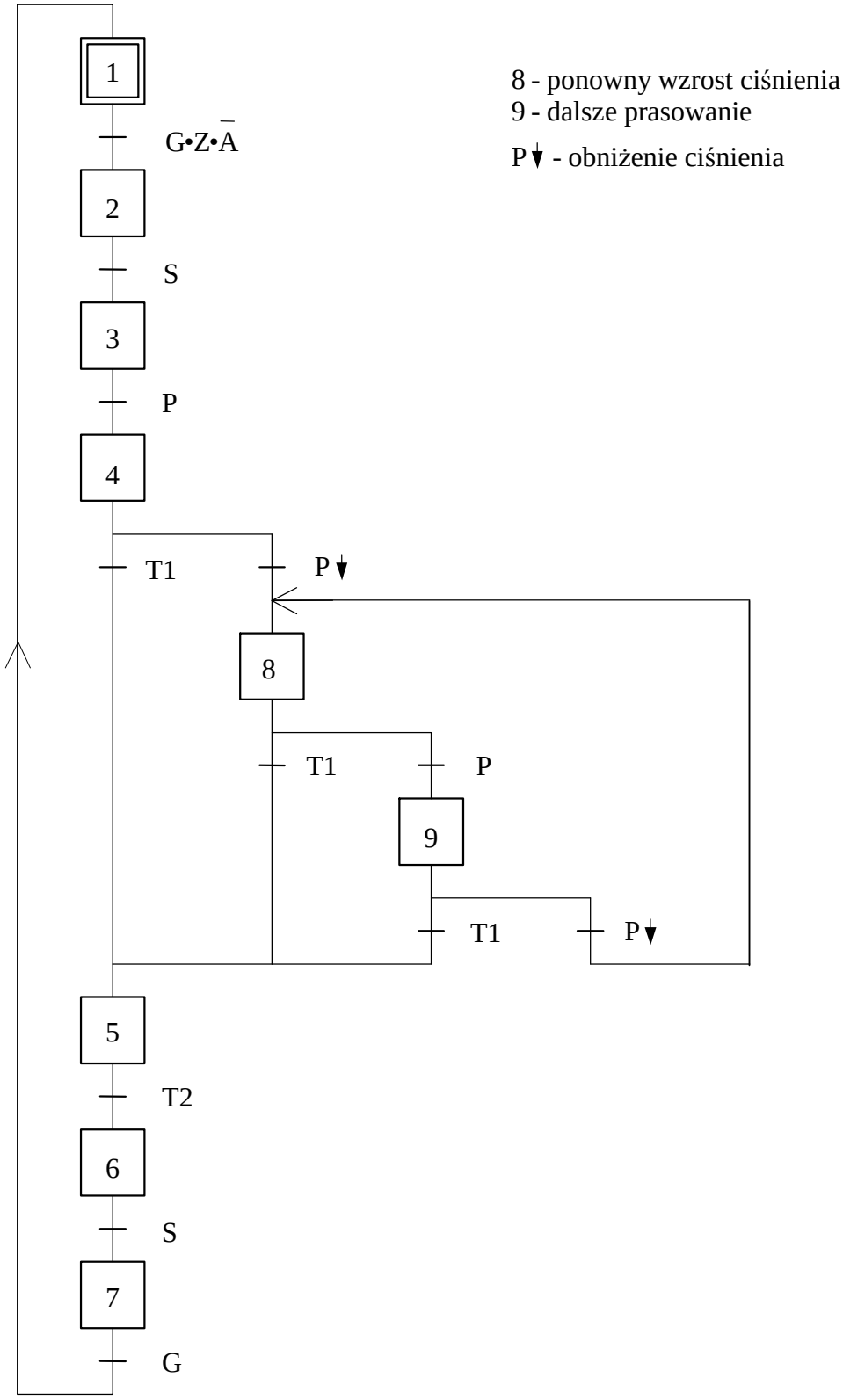
Często spotykanym wymaganiem dla czynności prasowania w procesach technologicznych jest utrzymywanie z dużą dokładnością stałej wartości ciśnienia prasowania. Ponieważ podczas prasowania ciśnienie może wolno się obniżać, co jest związane z nieidealnością układu hydraulicznego, w przypadku długich czasów prasowania

niezbędne jest wprowadzenie funkcji ponownego wzrostu ciśnienia i funkcji dalszego prasowania. Jeśli ciśnienie zmaleje poniżej dopuszczalnej, granicznej wartości (określanej zazwyczaj w % wartości zadanej), to musi być zrealizowana funkcja sterowania powodująca ponowny wzrost ciśnienia do wartości zadanej. Następnie prasowanie jest kontynuowane.



Rys.2.7. Diagram GRAFCET cyklu prasowania w trybie pracy automatycznej z uwzględnieniem działań powiązanych z krokami

Diagram GRAFCET cyklu prasowania w trybie pracy automatycznej uzupełniony o funkcje ponownego wzrostu ciśnienia i dalszego prasowania przedstawiono na rys.2.8. Z punktu widzenia przedstawienia algorytmu sterowania najważniejsze są kroki, połączenia pomiędzy krokami i przejścia (warunki przejść) na tych połączeniach, a mniej istotne są działania. Dlatego więc na tym i kolejnym rysunku, aby nie pogarszać czytelności rysunków, pominięto graficzne przedstawienie działań (operacji) powiązanych z krokami diagramu.



Rys.2.8. Diagram GRAFCET cyklu prasowania w trybie pracy automatycznej uzupełniony o funkcję ponownego wzrostu ciśnienia i funkcję dalszego prasowania

Warunkiem przejścia do kroku ponownego wzrostu ciśnienia jest obniżenie ciśnienia poniżej wartości dopuszczalnej (P↓). Czas prasowania T1 jest liczony łącznie dla kroków: prasowania, ponownego wzrostu ciśnienia i dalszego prasowania. Sekwencje ponownego wzrostu ciśnienia i dalszego prasowania mogą być wykonywane wielokrotnie.

Podobnie jak w przypadku funkcji dodatkowych (rozpatrzona powyżej funkcja ponownego wzrostu ciśnienia) diagram GRAFCET może być uzupełniony również o kroki odpowiadające awaryjnym stanom pracy.

Rozpatrzone zostaną dwa przypadki:

- uszkodzenie czujnika spowolnienia S ;
- nie nastąpił wzrost ciśnienia prasowania do wartości zadanej w założonym czasie.

Uszkodzenie czujnika spowolnienia ruchu prasy jest wykrywane wówczas, gdy:

- podczas opuszczania prasy brak będzie sygnału z czujnika S przed osiągnięciem zadanej wartości ciśnienia prasowania P ;
- podczas podnoszenia prasy brak będzie sygnału z czujnika S przed osiągnięciem górnego położenia prasy (sygnał z czujnika G).

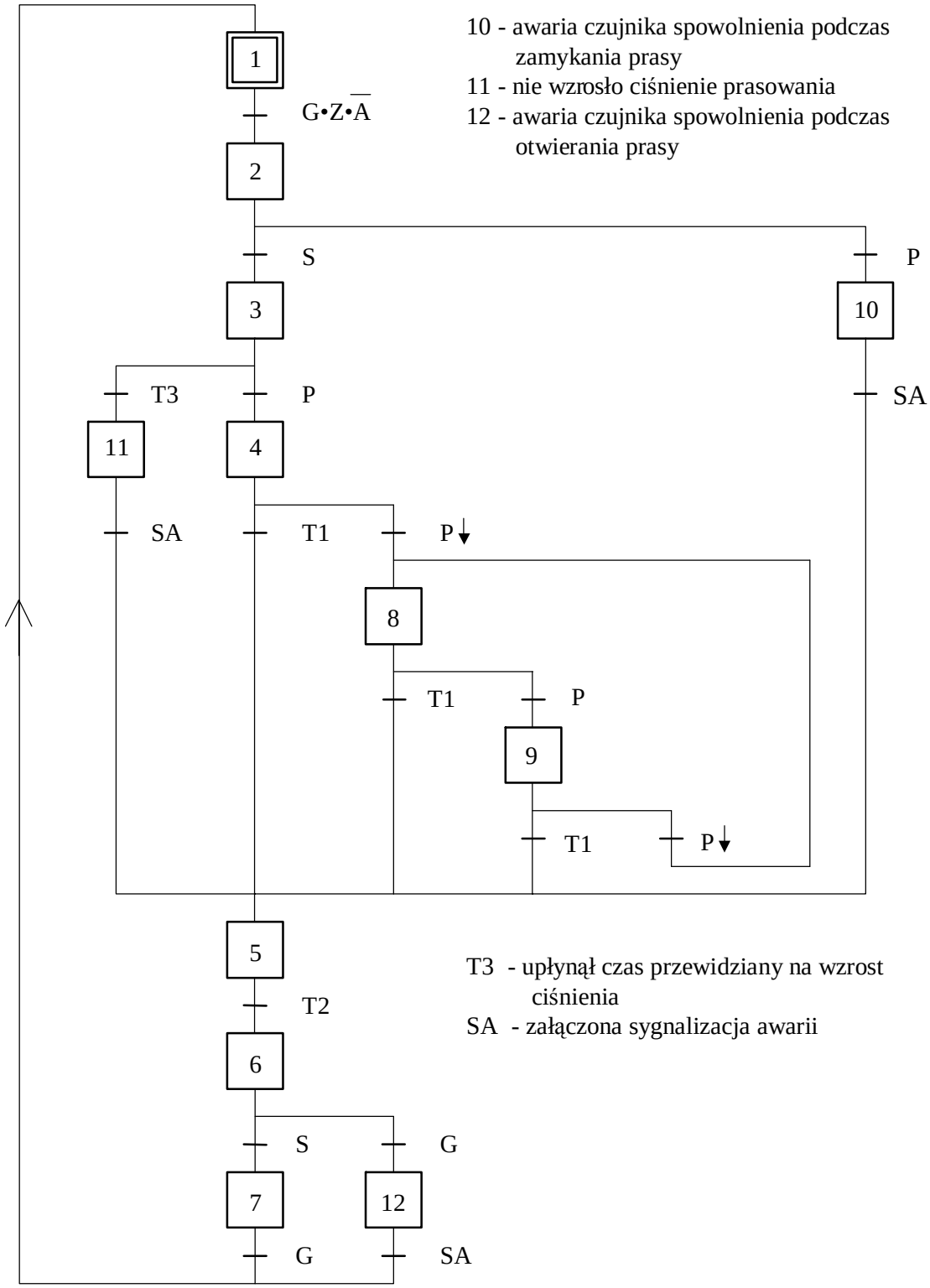
Przyczyną, że w założonym czasie nie nastąpił wzrost ciśnienia do wartości zadanej (T3 - maksymalny czas narastania ciśnienia) może być niesprawność układu hydraulicznego podczas cyklu prasowania, spowodowana np. wyciekiem oleju z układu hydraulicznego, uszkodzeniem zaworu realizującego sprężanie itp. Charakterystyczne dla tego typu uszkodzeń jest to, że program sterowania nie może stwierdzić ich zaistnienia bezpośrednio na podstawie sygnałów wejściowych, tylko w sposób pośredni.

Diagram GRAFCET, analogiczny do przedstawionego na rysunku 2.8, uzupełniony trzema krokami odpowiadającymi rozpatrywanym awaryjnym stanom pracy przedstawiono na rysunku 2.9. Ponieważ krokom dla tych trzech awaryjnych stanów pracy nie są przyporządkowane działania (za wyjątkiem sygnalizacji awarii) możliwe jest umieszczenie w diagramie po tych krokach a przed dekompresją warunków przejść które będą spełnione od razu po zasygnalizowaniu awarii (SA).

Rozpatrzone trzy stany awaryjne nie obejmują wszystkich możliwych stanów awaryjnych, które mogą wystąpić w trybie pracy automatycznej prasy w linii; zostały one wybrane i przedstawione tylko dla zilustrowania przyjętego sposobu tworzenia algorytmu.

Przedstawiony sposób opisu algorytmów w języku GRAFCET może być dużym ułatwieniem podczas weryfikacji opracowywanych algorytmów. Dzięki takiemu opisowi algorytmu sterowania istnieje również możliwość szybkiego sprawdzenia, jakie uszkodzenia (czy wszystkie istotne i najczęściej występujące), zostały uwzględnione na etapach

projektowania systemu i pisania programu sterowania. Analiza samego programu sterowania nie dałaby odpowiedzi tak szybko.



Rys.2.9. Diagram GRAFCET z rysunku 2.8 uzupełniony trzema krokami odpowiadającymi rozpatrywanym awaryjnym stanom pracy cyklu prasowania w trybie pracy automatycznej

Do niewątpliwych zalet języka GRAFCET należy więc duża przejrzystość i uporządkowanie algorytmów sterowania oraz możliwość przedstawienia za pomocą diagramu GRAFCET całego, złożonego zadania automatyzacji (jak i wydzielonych, poszczególnych zadań sterowania), co może być ułatwieniem przy zespołowej pracy nad oprogramowaniem.

2.5. Modelowanie procesów sekwencyjnych za pomocą grafów skierowanych

W tym podrozdziale opisana jest metoda wykorzystująca do opisu procesów sekwencyjnych grafy skierowane, oraz sposób jej realizacji w programie sterownika programowalnego [33]. Do najważniejszych zalet tej metody (zbliżonej do metody GRAFCET) należą:

- prostota metody wynikająca z intuicyjnego sposobu tworzenia algorytmu sterowania procesem sekwencyjnym;
- możliwość uwzględnienia w opisie algorytmu sterowania sekwencyjnego stanów normalnej pracy jak i stanów awaryjnych;
- możliwość programowej realizacji algorytmu sterowania sekwencyjnego za pomocą prostych instrukcji SET i RESET.

Etap podstawowy odpowiada stanowi normalnej pracy, a stanowi awaryjnemu odpowiada etap zabezpieczeń. Wszystkie etapy są wierzchołkami grafu skierowanego. Etapy podstawowe są to etapy, które przy poprawnym działaniu programu wystarczają do prawidłowego (poprawnego z punktu widzenia założeń, czyli zgodnego z wymaganiami użytkownika) sterowania urządzeniem lub procesem technologicznym. Etapy zabezpieczeń (inaczej nazywane etapami awaryjnymi) są to etapy, które realizują funkcje bezpieczeństwa w przypadku wystąpienia defektu.

Algorytmy sterowania w tej metodzie są przedstawiane za pomocą grafów skierowanych. Każdy graf skierowany algorytmu sterowania składa się (jest złożeniem) dwóch grafów skierowanych - jednego dla etapów podstawowych i drugiego obejmującego wszystkie etapy zabezpieczeń. Pełny opis algorytmu sterowania, oprócz zbudowania grafów skierowanych, wymaga jeszcze utworzenia tablic etapów i warunków przejść pomiędzy etapami.

Tak więc metoda ta jest opisywana przez następujące zbiory:

E = { E1 , E2 , ... , En }	skończony zbiór etapów podstawowych, gdzie n - liczba etapów podstawowych;
A = { A1 , A2 , ... , Ak }	skończony zbiór etapów zabezpieczeń, gdzie k - liczba etapów zabezpieczeń (awaryjnych);

X = { X1 , X2 , ... , Xp } skończony zbiór połączeń skierowanych (gałęzi grafu) wskazujących kierunki przejścia z jednego etapu do drugiego, gdzie p - liczba połączeń skierowanych;

W = { W1 , W2 , ... , Wm } skończony zbiór warunków przejść pomiędzy etapami, gdzie m = p (warunki przejścia są przyporządkowane do połączeń skierowanych) lub m = n + k (gdy warunki przejścia są tzw. warunkami brzegowymi i nie są powiązane z etapami.

Dowolna liczba etapów może zostać połączona tworząc w ten sposób fragment programu modelujący złożoną funkcję realizowaną przez program, lub też może zostać połączona tworząc podprogram przywoływany z programu głównego.

Tworzony jest wówczas zbiór dodatkowy:

M = { M1 , M2 , ... , Mi } skończony zbiór funkcji złożonych (podprogramów), gdzie i - liczba funkcji złożonych, i < n .

Do funkcji złożonej może prowadzić tylko jedno połączenie skierowane (wejściowe), ze związanym z nim warunkiem przejścia, oraz wychodzić tylko jedno połączenie skierowane (wyjściowe), ze związanym z nim warunkiem przejścia. Dzięki temu funkcja złożona może być uwzględniana w grafie skierowanym modelującym proces sekwencyjny tak samo jak pojedynczy etap.

Funkcje złożone (podprogramy) nie posiadają warunków przejść nawet wówczas, gdy są to tzw. warunki brzegowe powiązane z etapami. Warunki te posiadają etapy z których składa się funkcja złożona, w szczególności istotne są warunki przejścia dla pierwszego i ostatniego etapu składającego się na funkcję złożoną.

Do grafu pierwszego przynależą tylko etapy podstawowe i odpowiednie warunki przejść pomiędzy nimi. Graf ten zawiera te etapy, które reprezentują wypełnienie wszystkich zadań postawionych systemowi, gdy nie występują defekty. Tak więc pierwszy graf przedstawia poprawne działanie układu.

Graf drugi zawiera zarówno etapy podstawowe jak i etapy awaryjne. Graf ten reprezentuje reakcje programowanego systemu sterowania na wystąpienie defektu w systemie.

Etapy podstawowe spełniają w nim dwie funkcje:

- przedstawiają etapy ochraniane; z etapu tego, po wykryciu defektu, układ przechodzi do etapu awaryjnego;
- przedstawiają etapy do których układ przechodzi z etapów awaryjnych.

Złożenie obu tak zbudowanych grafów skierowanych daje w wyniku graf opisujący działanie programowanego sekwencyjnego układu sterowania dla wszystkich przewidzianych przez projektanta zdarzeń. Takie przedstawienie programu czyni jego strukturę bardziej przejrzystą i upraszcza jego analizę.

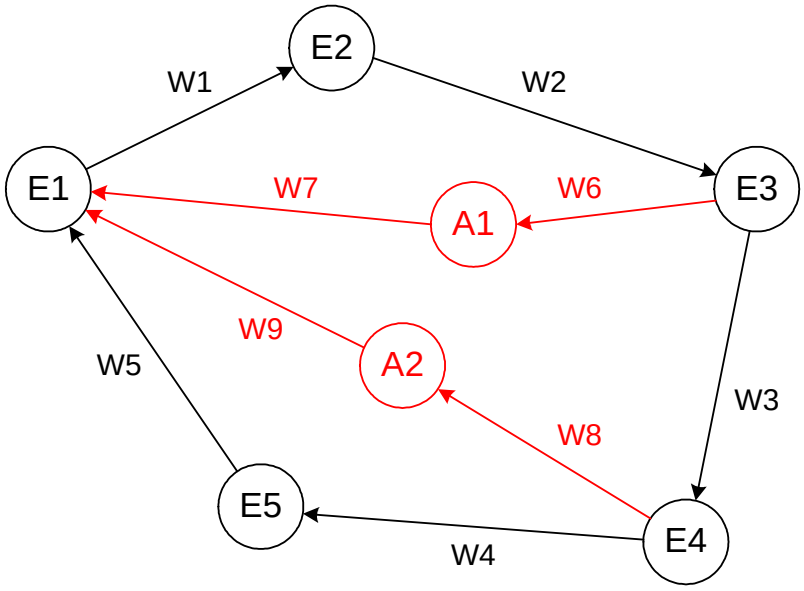
Podstawowe reguły tworzenia grafu skierowanego modelującego proces sekwencyjny są następujące:

- jeden z etapów powinien być wyróżniony jako etap początkowy;
- etapy podstawowe tworzą obwód (drogę zamkniętą z etapem początkowym);
- z etapu podstawowego można przejść do kolejnego etapu podstawowego lub do etapu zabezpieczeń;
- z etapu zabezpieczeń można przejść tylko do etapu podstawowego;
- w grafie co najmniej jeden etap jest etapem aktywnym;
- przejście jest realizowane wówczas, gdy aktywny jest etap je poprzedzający i spełniony jest warunek przejścia;
- realizacja przejścia jest obligatoryjna i natychmiastowa;
- realizacja przejścia jest związana z dezaktywacją etapu poprzedzającego i aktywacją etapu następnego;
- z etapem powiązane są działania realizowane wówczas, gdy etap jest aktywny;
- synchronizacja sterowania (koniunkcja) jak i wybór wariantu sterowania (alternatywa) są realizowane poprzez odpowiednie zdefiniowanie warunków przejść.

Dla przykładu, na rysunku 2.10 pokazano prosty diagram skierowany modelujący proces sekwencyjny. Diagram posiada etap początkowy oznaczony jako E1, cztery inne etapy podstawowe (E2, E3, E4, E5), dwa etapy zabezpieczeń (A1 i A2) oraz dziewięć połączeń skierowanych zawierających odpowiednie warunki przejść (W1÷W9).

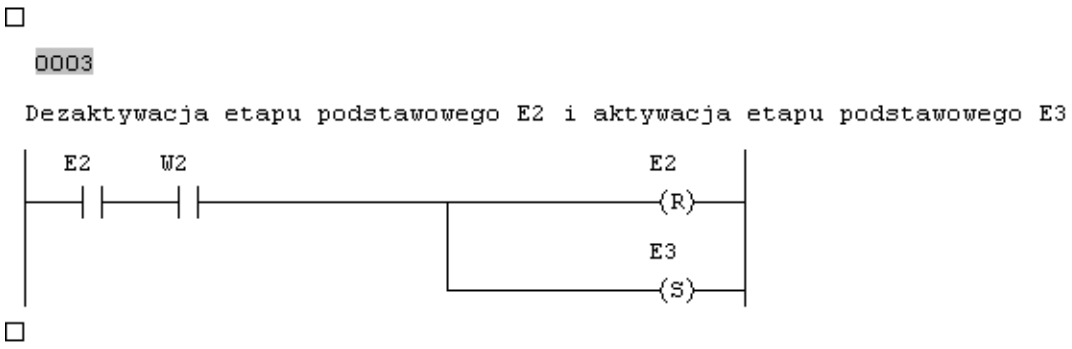
Przedstawiony na rysunku graf jest złożeniem dwóch grafów – pierwszego składającego się z etapów podstawowych (E1÷E5) i drugiego składającego się z dwóch etapów zabezpieczeń (A1 i A2) oraz trzech etapów podstawowych (E1, E3 i E4).

Ponieważ reguła budowy grafu skierowanego mówi o tym, że co najmniej jeden etap powinien być etapem aktywnym, to w chwili pierwszego uruchomienia programu jedna zmienna odpowiadająca etapowi (zazwyczaj etapowi początkowemu) powinna przyjąć wartość logiczną TRUE (poprzez zadeklarowanie wartości początkowej lub poprzez iloczyn negacji zmiennych dla wszystkich etapów). Kolejne uruchomienia programu już tego nie wymagają, pod warunkiem oczywiście, że zmienne odpowiadające etapom zadeklarowano jako pamiętane.



Rys.2.10. Przykładowy graf algorytmu sterowania z etapami podstawowymi i z etapami zabezpieczeń

Na rysunku 2.11 przedstawiono przykładową realizację programową przejścia pomiędzy etapami podstawowymi.

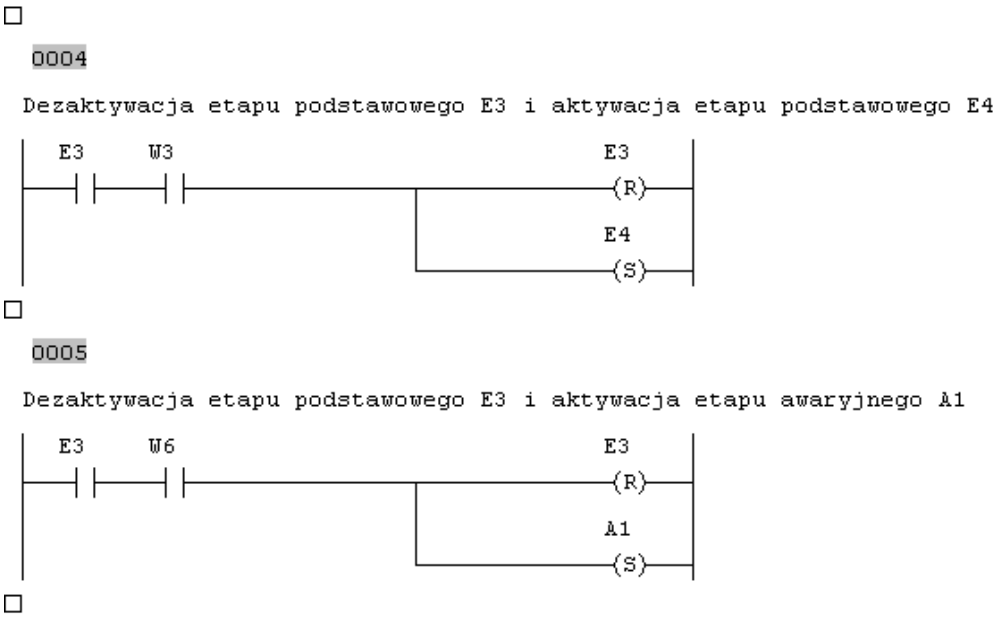


Rys.2.11. Przykład przejścia w grafie z rysunku 2.10 pomiędzy etapami podstawowymi

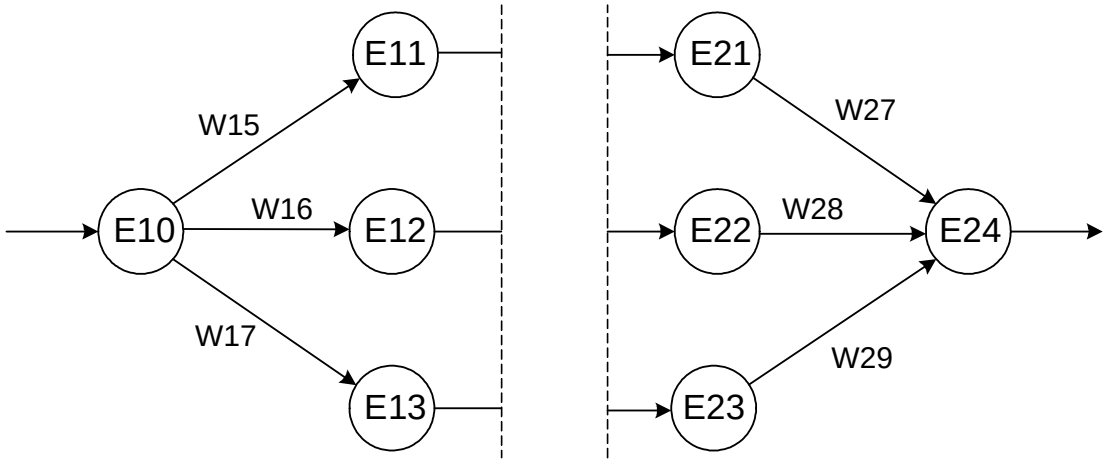
Na rysunku 2.12 przedstawiono natomiast realizację programową przejść pomiędzy etapem podstawowym E3 a etapem podstawowym E4, oraz pomiędzy etapem podstawowym E3 a etapem zabezpieczeń A1.

W podobny sposób wygląda realizacja programowa w przypadku wyboru jednego wariantu spośród kilku wariantów sterowania (realizacji procesu sekwencyjnego). Na rysunku 2.13 przedstawiono przykładowy fragment grafu z wyborem (alternatywą) wariantu sterowania (w tym przypadku jeden spośród trzech). Realizowany będzie ten wariant sterowania, dla którego warunek przejścia będzie spełniony jako pierwszy. Sprawdzenie

odbywa się w ustalonej kolejności przejść (warunków przejść) czyli najpierw W15, następnie W16 i na końcu W17. Aby sterowanie było określone, jeden z warunków (najczęściej ostatni) powinien być dopełnieniem logicznym wszystkich poprzednich, czyli W17 powinien być negacją sumy logicznej W15 i W16. Dla poszczególnych wariantów sterowania graf może zawierać również etapy zabezpieczeń.



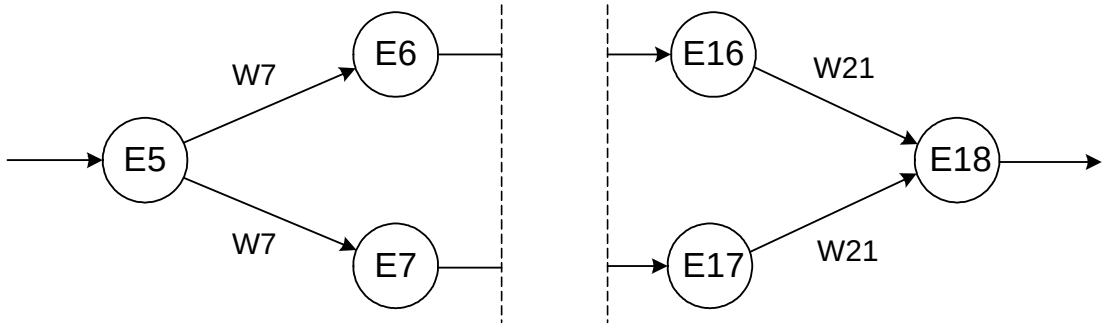
Rys.2.12. Przykłady możliwości przejść w grafie z rysunku 2.10 pomiędzy etapami podstawowymi oraz pomiędzy etapem podstawowym a etapem awaryjnym



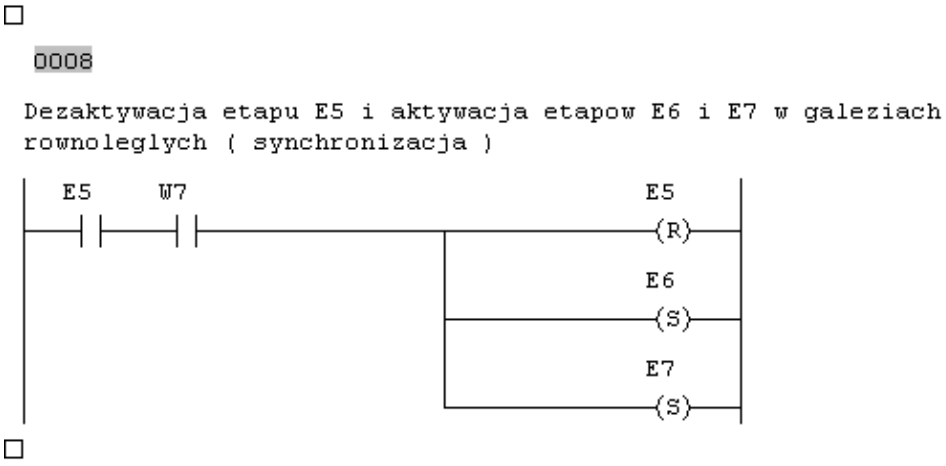
Rys.2.13. Przykładowy fragment grafu dla wyboru (alternatywy) sterowania

W przypadku synchronizacji (koniunkcji) sterowania realizacja programowa musi uwzględniać wszystkie etapy, które muszą stać się aktywne równocześnie, jak również wszystkie etapy które równocześnie muszą przestać być aktywne.

Na rysunku 2.14 przedstawiono przykładowy fragment grafu z synchronizacją (koniunkcją) sterowania (w tym przypadku dla dwóch etapów: E6 i E7). Synchronizacja jest realizowana dzięki zdefiniowaniu takich samych warunków przejść pomiędzy etapami E5 i E6 oraz pomiędzy E5 i E7. Realizację programową rozpoczęcia synchronizacji sterowań przedstawiono na rysunku 2.15.



Rys.2.14. Przykładowy fragment grafu dla synchronizacji (koniunkcji) sterowania

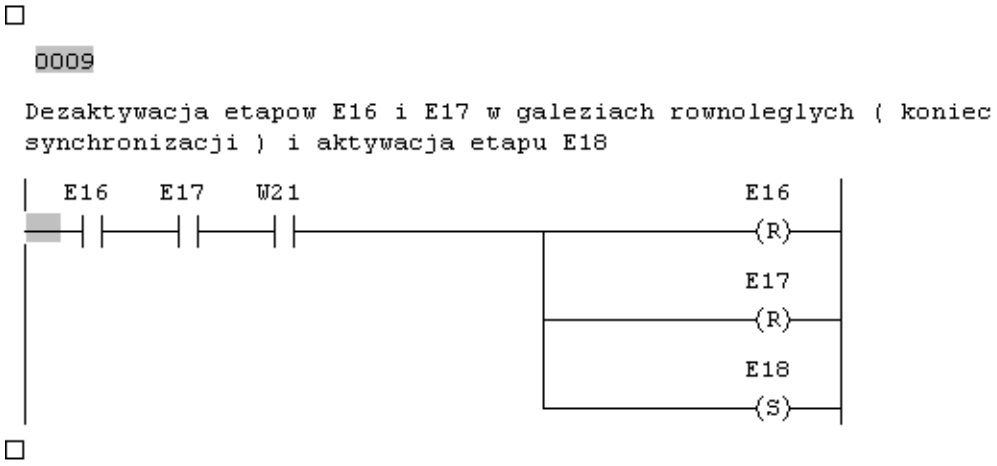


Rys.2.15. Przykład przejścia w grafie z rysunku 2.14 do etapów w gałęziach równoległych z synchronizacją sterowania

Realizację programową zakończenia synchronizacji sterowań przedstawiono na rysunku 2.16. Zakończenie synchronizacji może być zrealizowane wówczas, gdy wszystkie etapy kończące gałęzi równoległe staną się aktywne. Dla rozpatrywanego przykładu są to etapy E16 i E17.

Przy zakończeniu synchronizacji również musi być zdefiniowany, dla każdej gałęzi sterowania, ten sam warunek przejścia. Tak więc aktywacja etapu E18 nastąpi tylko wówczas, gdy aktywne są etapy E16 i E17 oraz spełniony jest warunek przejścia W21.

Z każdym z etapów są powiązane odpowiednie działania, które wykonywane są tylko wówczas, gdy dany etap jest aktywny, czyli gdy zmienna dla danego etapu określająca jego aktywność ma wartość logiczną TRUE.



Rys.2.16. Przykład przejścia w grafie z rysunku 2.14 z gałęzi równoległych z synchronizacją sterowania do jednego wspólnego etapu podstawowego

Przedstawione w tym podrozdziale modelowanie procesu sekwencyjnego z wykorzystaniem grafu skierowanego oraz jego realizacja programowa mogą być przydatne wówczas, gdy wykorzystuje się małe sterowniki programowalne nie pozwalające na programowanie w języku GRAFCET (SFC – zgodnie z normą IEC 61131-3).

3. Praca sterowników programowalnych w sieci

3.1. Wprowadzenie

Do realizacji systemów komputerowo zintegrowanego wytwarzania, nazywanego skrótowo CIM (*ang. Computer Integrated Manufacturing*), niezbędny jest efektywny i niezawodny system przesyłania informacji pomiędzy wszystkimi poziomami sterowania i zarządzania (a także coraz częściej, i w coraz większym stopniu, również poziomem modelowania działalności przedsiębiorstwa). Na rysunku 3.1 przedstawiono poziomy sterowania i zarządzania wyróżniane w systemach CIM. W przedstawionej piramidzie im niższy poziom tym ostrzejsze są ograniczenia czasowe na przesyłanie danych, a im wyższy poziom, tym większe są wymagania dotyczące ilości przesyłanej informacji.

Rys.3.1. Struktura sterowania w zakładach przemysłowych posiadających systemy komputerowo zintegrowanego wytwarzania

Pierwsze dwa poziomy odnoszą się do układów pomiarowych, wykonawczych i sterowania. Ilość przesyłanych danych pomiędzy urządzeniami na tych poziomach jest niewielka (od pojedynczych bitów do kilkudziesięciu, kilkuset bajtów), ale wymagana jest duża szybkość uaktualniania informacji (przyjmuje się, że dla większości zastosowań czas uaktualniania danych powinien być nie dłuższy niż 5÷10 ms, a maksymalnie 20 ms). Wiąże się to także z koniecznością zapewnienia takiego czasu reakcji systemu sterowania, który

byłby nie dłuższy od dopuszczalnego maksymalnego czasu reakcji. Dla wyższych niż drugi poziomów sterowania i zarządzania ilość przesyłanych danych wzrasta (od Kbajtów do Mbajtów i więcej), ale równocześnie dopuszczalne są pewne opóźnienia w przesyłaniu informacji (od ułamków sekundy do pojedynczych sekund dla poziomu trzeciego, oraz do pojedynczych minut i dłużej dla poziomów czwartego i piątego).

Urządzenia i elementy automatyki wykorzystywane i przesyłające między sobą dane na pierwszym i drugim poziomie sterowania są połączone tzw. sieciami miejscowymi, nazywanymi również sieciami przemysłowymi lub sieciami polowymi (*ang. fieldbus*). Często spotyka się także określenie „magistrale miejscowe" zamiast „sieci miejscowe", co wynika bezpośrednio z tłumaczenia terminów (*ang. bus* to magistrala, a *ang. net* lub *network* to sieć).

Rozwój i coraz powszechniejsze wykorzystywanie sieci miejscowych są ściśle związane z ogólną tendencją do decentralizacji sterowania. Szczególnie wyraźnie tendencja ta jest widoczna w systemach sterowania ze sterownikami programowalnymi.

Dzięki stosowaniu sieci miejscowych w systemach sterowania uzyskuje się wiele wymiernych korzyści:

- niższe są koszty okablowania i skraca się czas potrzebny na okablowanie systemu. Duża część tradycyjnego okablowania systemów sterowania (nawet na poziomie czujników i elementów wykonawczych) jest zastępowana przez sieci miejscowe;
- zwiększa się niezawodność systemu sterowania jako całości, gdyż szacuje się, że zdecydowana większość uszkodzeń w systemach sterowania lokalizowana jest na urządzeniach we/wy (w tym są również uszkodzenia w okablowaniu związanym z doprowadzeniem i wyprowadzeniem sygnałów we/wy) oraz innych urządzeniach peryferyjnych. Dodatkowo zwiększają się możliwości diagnostyczne, gdyż możliwa jest bieżąca diagnostyka wszystkich urządzeń i elementów pracujących w sieci;
- uzyskuje się możliwość łatwej i szybkiej modyfikacji sieci, a przez to łatwość rozbudowy systemów sterowania w przyszłości;
- wykorzystywanie sieci miejscowych umożliwia budowę zdecentralizowanych systemów sterowania, a w konsekwencji daje możliwość podziału złożonych zadań sterowania na prostsze zadania wykonywane np. nie przez jeden, a przez kilka sterowników programowalnych pracujących w sieci.

Urządzenia pracujące i przesyłające między sobą informacje na trzecim, czwartym i piątym poziomie sterowania i zarządzania są połączone sieciami lokalnymi LAN (*ang. Local Area Network*), gdy ich długości nie przekraczają kilkuset metrów lub kilku kilometrów. W przypadku rozległych zakładów przemysłowych, gdy długości sieci są rzędu kilkunastu lub

nawet kilkudziesięciu kilometrów są to już sieci metropolitalne MAN (*ang. Metropolitan Area Network*). Zdarza się także, że siecią (zazwyczaj na poziomie czwartym lub piątym) połączone są wszystkie zakłady produkcyjne i biura jakiejś firmy, i wówczas będzie to już sieć rozległa WAN (*ang. Wide Area Network*).

Aby budowa takich sieci była możliwa, oraz aby urządzenia różnych producentów mogły pracować w jednej sieci, niezbędne było opracowanie odpowiednich standardów dotyczących modelu komunikacji.

3.2. Wzorcowy model łączenia systemów otwartych

Do opisu procesu przesyłania danych w sieciach komputerowych, w tym także w systemach sterowania ze sterownikami programowalnymi, wykorzystywany jest tzw. wzorcowy model łączenia systemów otwartych [5,49]. Jest to model opracowany w 1978 roku przez Międzynarodową Organizację d/s Standaryzacji - ISO (*ang. International Organization for Standardization*; ten akronim pochodzi od pierwszych liter nazwy w języku francuskim) i nazywany modelem OSI (*ang. Open System Interconnection*). Model ISO/OSI służy do uporządkowanego opisu realizowanych funkcji oraz wzajemnych zależności pomiędzy sprzętem i oprogramowaniem sieciowym, których spełnienie umożliwia pracę urządzeń sieciowych w jednej sieci.

Aby uporządkować opis zadań realizowanych przez urządzenie sieciowe, w modelu ISO/OSI wyszczególniono siedem warstw opisujących poszczególne funkcje. Warstwy modelu ISO/OSI dla dwóch urządzeń (stacji) pracujących w jednej sieci przedstawiono na rysunku 3.2. Każda warstwa realizuje i udostępnia wyższej warstwie określone usługi. Korzystając z usług udostępnianych przez warstwę *n*-1, warstwa *n* jednej stacji może się komunikować z warstwą *n* innej stacji. Reguły współpracy warstw *n* w różnych stacjach są nazywane protokołem warstwy *n*.

Funkcje realizowane przez poszczególne warstwy są następujące.

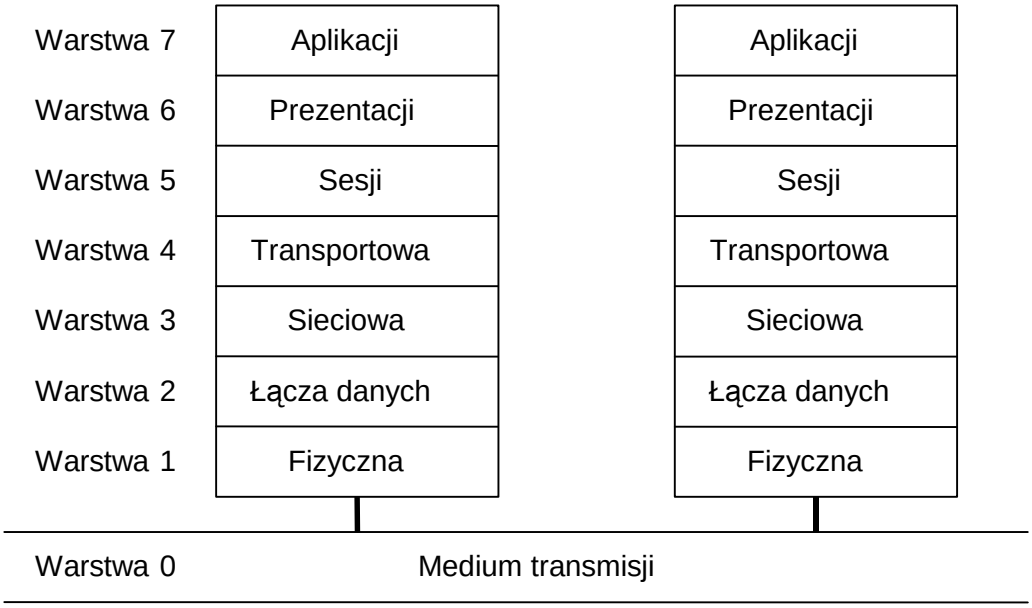
Warstwa 0 – medium transmisji

Określa rodzaj medium transmisji wykorzystywanego do przesyłania sygnałów.

Warstwa 1 - fizyczna

Opisuje parametry mechaniczne, elektryczne i funkcjonalne łączy danych związane z wykorzystywanym medium transmisji (szybkość transmisji, poziom napięcia itp.). Określa także topologię sieci. W tej warstwie następuje przesyłanie strumienia bitów nadsyłanych

przez warstwę łącza danych pomiędzy bezpośrednio połączonymi stacjami sieci, ale bez określania jakiejś struktury przesyłanego strumienia bitów.



Rys.3.2. Warstwowy model ISO/OSI przesyłania danych w sieci

Warstwa 2 - łącza danych (liniowa)

Określa komunikację pomiędzy dwiema stacjami odbywającą się w jednym łączu. Jest to warstwa której zadaniem jest nadawanie i odbiór przesyłanych danych (w postaci ramek). Funkcje warstwy obejmują sterowanie dostępem do łącza sieciowego oraz realizację łącza logicznego. W warstwie tej realizowana jest także kontrola poprawności odbieranych danych i ewentualnie powtarzanie nadawania ramek w przypadku wykrycia błędu.

Warstwa 3 - sieciowa

Określa komunikację pomiędzy dwiema stacjami odbywającą się w sieci złożonej z wielu łączy danych; odnosi się więc do sieci o rozgałęzionej strukturze. Warstwa ta uniezależnia wszystkie wyższe warstwy od pierwszych dwóch warstw modelu, ponieważ realizuje funkcje i procedury wysokiego poziomu obsługujące wymianę danych między stacjami połączonymi siecią złożoną.

Warstwa 4 - transportowa

Zapewnia niezawodne mechanizmy przesyłania danych, korekcje błędów występujących w połączeniu między stacjami oraz mechanizm kontroli przepływu zawarty w oprogramowaniu sieciowym wyższego poziomu.

Warstwa 5 - sesji

Zapewnia mechanizmy nawiązywania niezawodnego połączenia pomiędzy współpracującymi aplikacjami.

Warstwa 6 - prezentacji

Zapewnia mechanizmy obsługi procesu prezentowania danych w aplikacjach. Warstwa ta może także szyfrować i deszyfrować dane.

Warstwa 7 - aplikacji

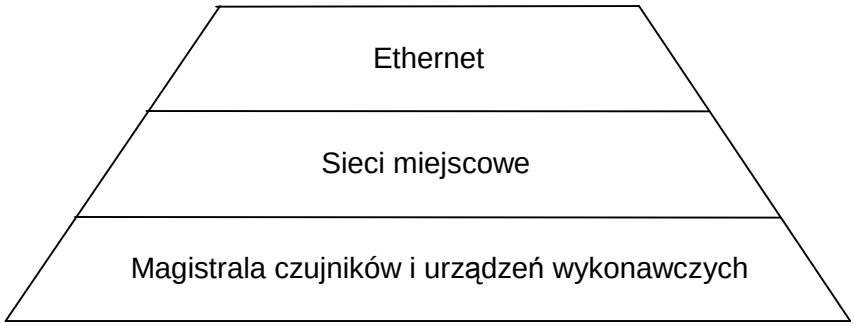
Zapewnia mechanizmy obsługi aplikacji użytkownika.

W modelu przedstawionym na rysunku dodatkowo wyróżniono warstwę 0 czyli medium transmisji, chociaż bardzo często spotyka się, że jest ono kojarzone z warstwą 1.

Z punktu widzenia realizowanych funkcji warstwy modelu można podzielić na dwie grupy. Warstwy od 0 do 4 odnoszą się do funkcji przesyłania, a warstwy od 5 do 7 do funkcji zastosowania (aplikacji).

3.3. Charakterystyka sieci miejscowych

Sterowniki programowalne mogą być i są wykorzystywane na trzech pierwszych poziomach struktury sterowania przedstawionej na rysunku 3.1. Są to poziomy sterowania związane z automatyzacją produkcji. Z każdym poziomem sterowania związany jest określony rodzaj sieci, a z nimi z kolei wykorzystywane protokoły komunikacyjne. Na rysunku 3.3 przedstawiono sieci sterowania stosowane na poziomach automatyzacji produkcji (ze sterowaniem nadrzędnym procesami produkcji).



Rys.3.3. Sieci sterowania na poziomach produkcji

Na poziomie pierwszym – układów pomiarowych i wykonawczych – wykorzystywana jest magistrala czujników i urządzeń wykonawczych. Najczęściej

wykorzystywaną w przemyśle magistralą tego typu jest sieć AS-i (*ang. Actuator-Sensor Interface*).

Na poziomie drugim – systemów sterowania – wykorzystywane są sieci miejscowe. Jest to podstawowy poziom na którym pracują sterowniki programowalne. Do przesyłania danych w sieciach miejscowych stosuje się wiele różnych protokołów np. Profibus-DP, Modbus, CAN/CANopen, Interbus, Suconet K.

Na poziomie trzecim – sterowania nadrzędnego – najczęściej stosowaną siecią jest Ethernet. Sterowniki programowalne (zazwyczaj duże sterowniki modułowe) realizują na tym poziomie funkcje sterowania nadrzędnego, aczkolwiek częściej te funkcje są realizowane przez komputery typu IBM/PC. Wykorzystywane są różne protokoły pracujące w sieci Ethernet. Wcześniejsze rozwiązania na poziomie sterowania nadrzędnego wykorzystywały różne inne sieci i protokoły (np. Profibus-FMS).

Systemy sterowania ze sterownikami programowalnymi najczęściej wykorzystują sieci miejscowe. Dlatego też w tym podrozdziale przedstawiono ogólną charakterystykę sieci miejscowych. Jako sieć miejscowa może także pracować tzw. Ethernet przemysłowy, ale zagadnienia z nim związane omówiono w podrozdziale 3.7.

Sieci miejscowe wykorzystują warstwę 0, 1 i 2, a w przypadku, gdy siecią miejscową jest Ethernet przemysłowy, warstwy 0, 1, 2, 3, 4 i 7 (przy pracy bez kanału czasu rzeczywistego i warstwy 0, 1, 2 i 7 (przy pracy z kanałem czasu rzeczywistego).

Projektując lub analizując systemy sterowania [5,18,49], w tym i systemy sterowania ze sterownikami programowalnymi, należy rozpatrzyć więc następujące zagadnienia wynikające z funkcji realizowanych przez warstwy 0, 1 i 2 wzorcowego modelu łączenia systemów otwartych:

- rodzaj medium transmisji i wykorzystywany sprzęg komunikacyjny;
- topologię sieci;
- metodę dostępu do łącza sieciowego;
- wykorzystywany protokół komunikacyjny.

Medium transmisji

Wyróżnić można trzy rodzaje medium transmisji: przewód elektryczny, włókno światłowodowe i fale elektromagnetyczne.

Spośród rozwiązań wykorzystujących przewód elektryczny (nieekranowana skrętka, foliowana skrętka, ekranowana skrętka, kabel koncentryczny) w systemach sterowania ze sterownikami programowalnymi najczęściej stosuje się ekranowaną skrętkę. W sieciach o długościach rzędu kilkudziesięciu, kilkuset metrów i szybkościach transmisji do 12 Mbodów

zastosowanie ekranowanej skrętki jest rozwiązaniem najbardziej uzasadnionym ekonomicznie. Podstawową wadą ekranowanej skrętki jest jej wrażliwość na zakłócenia elektromagnetyczne przy dużych odległościach i przy dużych szybkościach transmisji.

W przypadkach gdy poziom zakłóceń elektromagnetycznych jest bardzo wysoki, lub gdy transmisja musi być realizowana na duże odległości (rzędu kilku, kilkunastu kilometrów), stosowane są łącza światłowodowe. Kolejną zaletą połączenia światłowodowego jest możliwość osiągnięcia większych szybkości transmisji (w porównaniu z połączeniem ekranowaną skrętką). Podstawową wadą połączeń światłowodowych jest ich wysoki koszt.

Za pomocą fal elektromagnetycznych realizowane są połączenia bezprzewodowe. Spośród obecnie stosowanych połączeń bezprzewodowych – optycznego (w zakresie podczerwieni) i radiowego (w odpowiednich pasmach MHz) – w systemach sterowania zdecydowanie częściej wykorzystywane jest połączenie radiowe. Połączenia bezprzewodowe są stosowane w przypadkach, gdy poprowadzenie sieci przewodowej jest niemożliwe lub bardzo drogie, lub gdy w sieci wymagane są mobilne stacje.

Sprzęg komunikacyjny

Istnieje kilka standardów połączeń przewodami elektrycznymi w komunikacji szeregowej (RS-232, RS-422, RS-423, RS-485, TTY). Ich charakterystykę przedstawiono np. w [52]. W systemach sterowania ze sterownikami programowalnymi najczęściej wykorzystuje się następujące sprzęgi komunikacyjne (szeregowe interfejsy cyfrowe): RS-232 i RS-485 (lub jego odmianę RS-422). W ostatnich latach w nowych modelach sterowników programowalnych bardzo często jest wykorzystywany sprzęg USB oraz Ethernet. Sprzęg RS-232 jest wykorzystywany najczęściej do podłączenia urządzenia programującego (komputer IBM/PC z programem narzędziowym) lub rzadziej do podłączenia jakiegoś jednego urządzenia którym może być np. panel operatorski lub drukarka. Sprzęg RS-485 (RS-422) służy do podłączenia sterownika do sieci miejscowej.

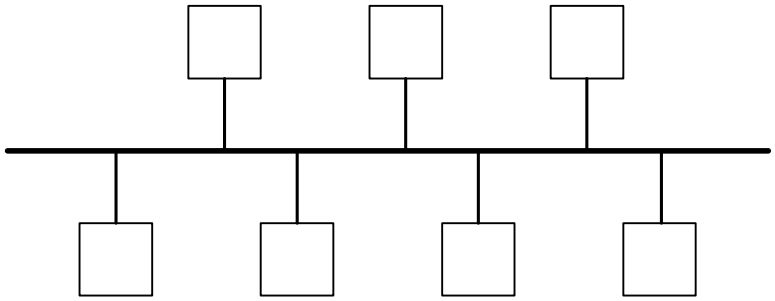
Topologia sieci

Spośród topologii sieci stosowanych w sieciach lokalnych (topologie: gwiazdy, pierścieniowa, magistralowa, drzewiasta, osiowa, nieregularna) w systemach sterowania ze sterownikami programowalnymi najczęściej stosowane są trzy topologie: magistralowa, drzewiasta i pierścieniowa.

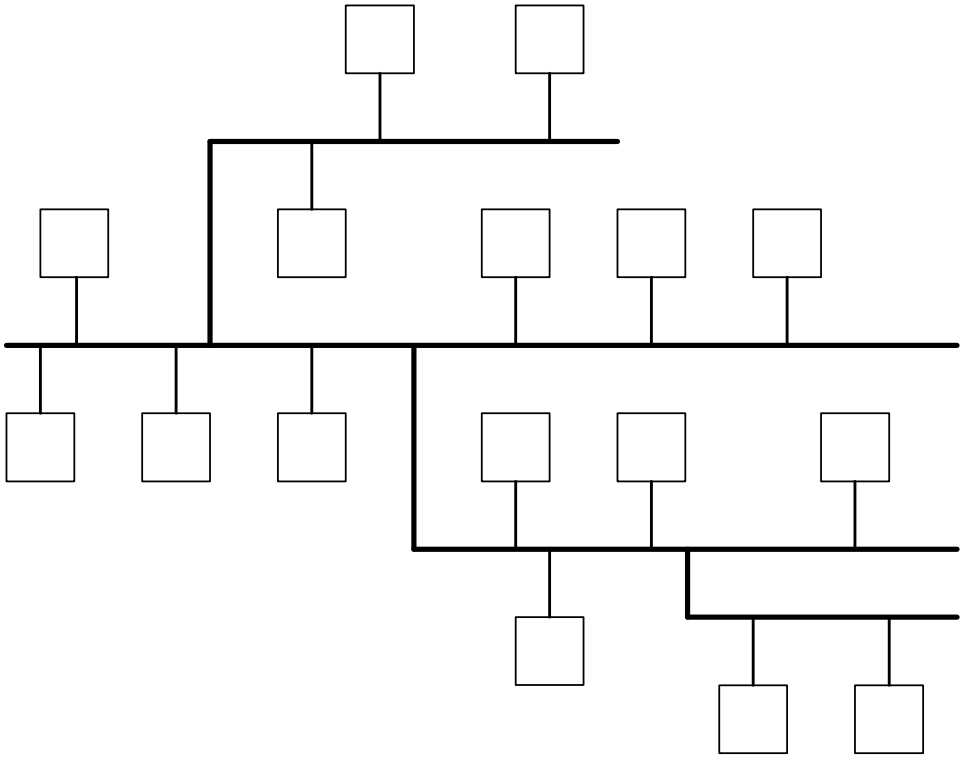
Przykład topologii magistralowej pokazano na rysunku 3.4. Topologia ta charakteryzuje się tym, że główna linia komunikacyjna ma formę magistrali. Stacje są przyłączone do magistrali przez krótki przewód. Wszystkie stacje przyłączone do magistrali

mogą się ze sobą komunikować. Ponieważ wszystkie stacje mają równy dostęp do magistrali, to wymagana jest regulacja przepływu danych w sieci poprzez zastosowanie odpowiedniej metody dostępu do łącza.

Przykład topologii drzewiastej pokazano na rysunku 3.5. Jest ona często stosowaną i najbardziej uniwersalną topologią. Taką topologię mają rozbudowane systemy sterowania. Ze względu na łatwą rozbudowę i możliwość różnorodnej konfiguracji jest ona najlepszą topologią do adaptacji sieci miejscowych pod specyficzne wymagania.



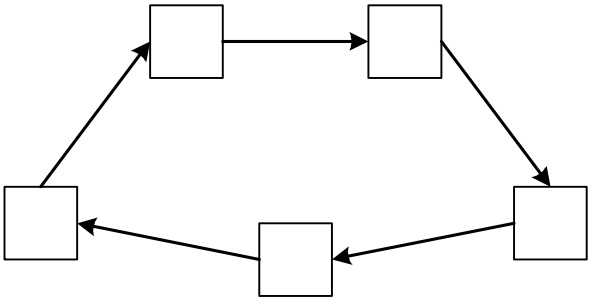
Rys.3.4. Topologia magistralowa



Rys.3.5. Topologia drzewiasta

Przykład topologii pierścieniowej pokazano na rysunku 3.6. W tej topologii wszystkie stacje są połączone jednym przewodem transmisyjnym i tworzą zamknięty pierścień. Każda z

tych stacji jest integralnym elementem sieci. Dane wykonują dokładnie jedno przejście dookoła pierścienia. Każda ze stacji sprawdza, czy dane są zaadresowane do niej i jeśli tak się zdarzy, kopiuje dane do swojej pamięci.



Rys.3.6. Topologia pierścieniowa

Metoda dostępu do łącza sieciowego

Wszystkie metody dostępu do łącza sieciowego można podzielić na dwie grupy [18]: metody stochastyczne i metody deterministyczne.

Spośród metod stochastycznych w sieciach lokalnych i rozległych najczęściej wykorzystywane są dwie metody CSMA – metody dostępu rywalizacyjnego z nasłuchem:

- CSMA/CD (ang. Carrier Sense Multiple Access with Collision Detection);
- CSMA/CA (ang. Carrier Sense Multiple Access with Collision Avoidance).

Spośród metod deterministycznych w sieciach lokalnych najczęściej wykorzystywane są:

- metoda master-slave;
- metoda przekazywania uprawnienia (żetonu).

Z kolei w metodzie przekazywania żetonu (ang. token-passing) można wyróżnić dwie jej odmiany:

- pierścień fizyczny w strukturze pierścieniowej (ang. token ring);
- pierścień logiczny w strukturze magistralowej (ang. token bus).

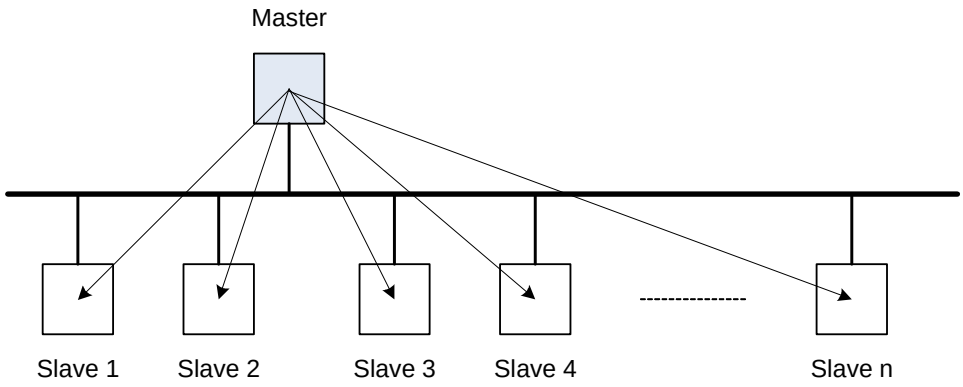
W systemach sterowania ze sterownikami programowalnymi wykorzystywane są prawie wyłącznie tylko metody deterministyczne, ze względu na to, że sterowanie w systemach czasu rzeczywistego wymaga determinizmu czasowego.

Wyjątkiem jest Ethernet przemysłowy wykorzystujący metodę stochastyczną CSMA/CD. W Ethernecie przemysłowym zapewnienie determinizmu czasowego (maksymalny czas opóźnienia przy przesyłaniu informacji rzędu 10÷20 ms) jest osiągane poprzez odpowiedni przydział priorytetów i ograniczanie domen.

W metodzie CSMA/CD każdy węzeł sieci prowadzi ciągły nasłuch swojego segmentu łącza i może rozpocząć nadawanie wówczas, gdy żadna inna stacja nie nadaje. Po stwierdzeniu kolizji, w wyniku równoczesnego rozpoczęcia nadawania przez kilka węzłów, nadawanie jest przerywane i ponawiane po pewnym czasie (czas określany losowo, różny dla każdej stacji).

Z metod deterministycznych w systemach sterowania ze sterownikami programowalnymi częściej wykorzystywana jest metoda master-slave, ale metoda z przekazywaniem uprawnienia jest również stosowana.

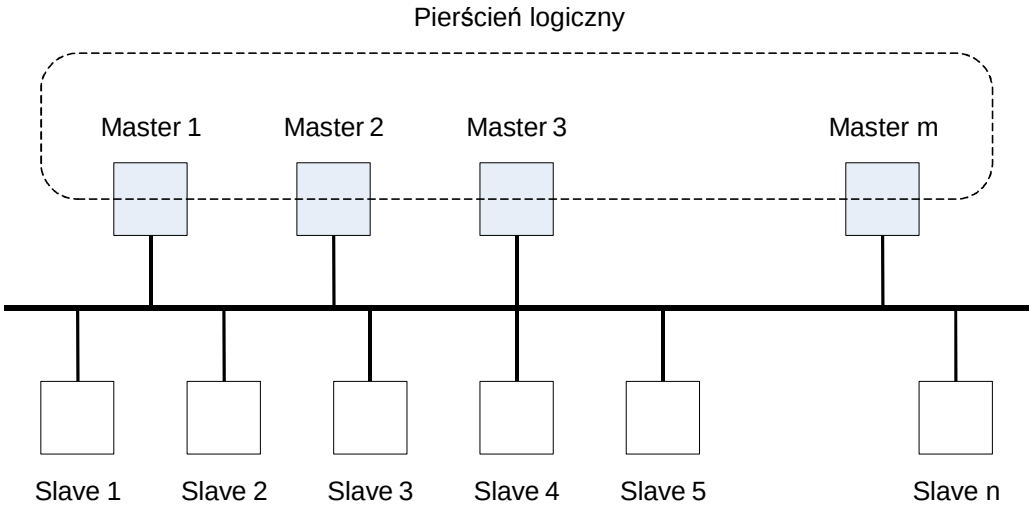
Metoda master-slave dostępu do łącza sieciowego polega na tym, że tylko jedno urządzenie (stacja) może inicjować transmisję (jest to urządzenie nadrzędne – master), a pozostałe urządzenia (podporządkowane – slaves) odpowiadają jedynie na zapytania mastera. Ilustrację metody master-slave przedstawiono na rysunku 3.7. Transmisja składa się z polecenia wysyłanego przez urządzenie master do urządzenia slave oraz z odpowiedzi przesyłanej z urządzenia slave do mastera. Odpowiedź zawiera dane żądane przez mastera lub potwierdzenie realizacji jej polecenia. Jest to więc odpytywanie (*ang. polling*) przez mastera wszystkich urządzeń typu slave. Master może adresować indywidualnych odbiorców lub też przesyłać wiadomości przeznaczone dla wszystkich urządzeń slave pracujących w sieci.



Rys.3.7. Metoda master-slave dostępu do łącza sieciowego

W metodzie z przekazywaniem uprawnień (żetonu) urządzenia (stacje) w sieci pomiędzy którymi są przesyłane informacje tworzą albo pierścień fizyczny (topologia pierścieniowa) lub pierścień logiczny w topologii magistralowej. Każdemu urządzeniu (stacji) jest przypisywany numer oraz tworzona jest zamknięta sekwencja tych numerów zgodnie z którą przekazywane są uprawnienia (żeton). W sekwencji zamkniętej ostatnia stacja w pierścieniu przekazuje prawo dostępu do łącza pierwszej stacji. Każda stacja zna numer następnej stacji w pierścieniu. Pierścień fizyczny powstaje w strukturze pierścieniowej

(pokazanej na rysunku 3.6) a pierścień logiczny przedstawiono na rysunku 3.8. Kolejność fizyczna stacji w pierścieniu logicznym jest nieistotna i niezależna od kolejności logicznej. Stacja posiadająca żeton dysponuje ściśle określonym czasem na przesłanie danych. Po upływie tego czasu musi przerwać transmisję i przekazać żeton następnej stacji w pierścieniu, niezależnie od tego czy przesłała wszystkie dane czy nie. Pozostałe do przesłania dane będzie mogła wysłać po następnym otrzymaniu żetonu. Metoda ta nie dopuszcza do powstania kolizji oraz zapewnia determinizm czasowy wymiany danych w sieci poprzez ograniczenie na czas przetrzymywania żetonu przez jedną stację.



Rys.3.8. Metoda dostępu do łącza sieciowego z przekazywaniem żetonu w pierścieniu logicznym

Protokół komunikacyjny

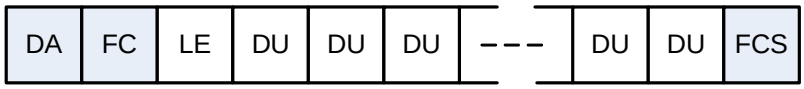
Przykład protokołu komunikacyjnego (Suconet K), wykorzystywanego w przemysłowej sieci miejscowej, omówiono w podrozdziale 3.4.

3.4. Sieć miejscowa Suconet K

Suconet K został opracowany przez firmę Klockner-Moeller (obecnie Moeller w koncernie Eaton) i jest otwartym standardem przemysłowej sieci miejscowej wykorzystywanym głównie w systemach sterowania czasu rzeczywistego ze sterownikami programowalnymi. Suconet K [18] wykorzystuje dwie pierwsze warstwy wzorcowego modelu łączenia systemów otwartych ISO/OSI. Sieć Suconet K ma topologię magistralową i wykorzystuje deterministyczną metodę dostępu do łącza sieciowego typu master-slave. W zależności od tego, jaki sterownik programowalny jest sterownikiem głównym (masterem) w

sieci, może on obsługiwać do 8 lub do 30 urządzeń podporządkowanych (slaves). Połączenie jest realizowane poprzez sprzęgi komunikacji szeregowej RS485 z medium transmisyjnym w postaci ekranowanej skrętki. Informacja jest przesyłana w ramkach o zmiennej długości; w ramkach tych przesyłane są znaki o długości 11 bitów. Szybkość transmisji dla Suconet K wynosi 187,5 kBodów (przy długości sieci do 600 m) lub 375 kBodów (przy długości sieci do 300 m).

Protokół Suconet K wykorzystuje standardową strukturę ramki przesyłanej informacji. Strukturę ramki w protokole Suconet K przedstawiono na rysunku 3.9.



Rys.3.9. Ramka przesyłanej informacji w protokole Suconet K

Ramka składa się z trzech części: nagłówka, pola danych i stopki.
W nagłówku można wyróżnić dwie części:

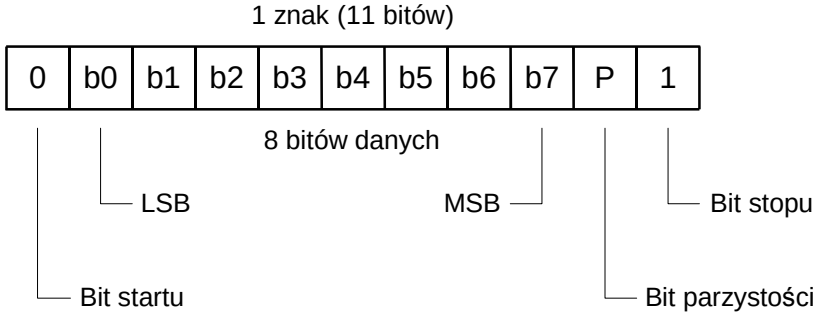
- adres docelowy (*ang. destination address –DA*);
- znak kontrolny ramki (*ang. frame control – FC*).

W przesyłanej informacji czyli w polu danych można wyróżnić:

- informację o długości pola danych (*ang. length character – LE*);
- przesyłane dane (*ang. dala unit – DU*).

Stopkę stanowi sekwencja kontrolna ramki (*ang. frame check sequence – FCS*) wyznaczana jako funkcje logiczne XOR dla DA, FC, LE i wszystkich DU.

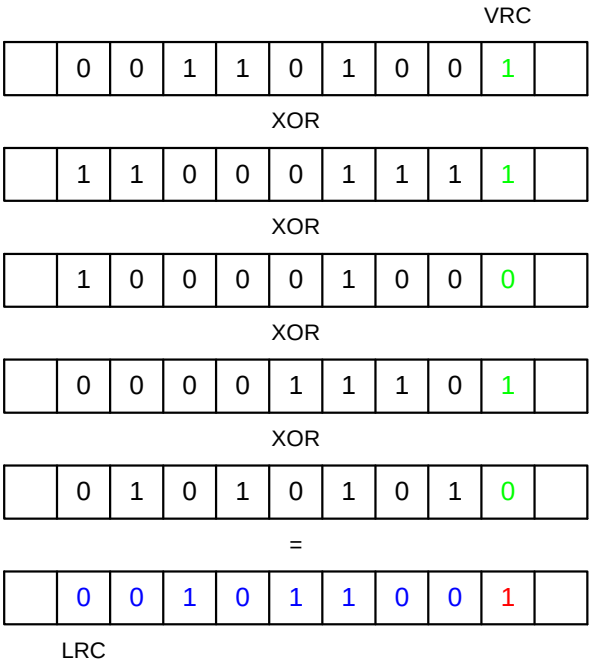
Każdy znak przesyłany szeregowo w protokole Suconet K składa się z 11 bitów. Ramkę dla znaku przesyłanego w protokole Suconet K przedstawiono na rysunku 3.10.



Rys.3.10. Ramka pojedynczego znaku w protokole Suconet K

Pierwszym bitem jest bit startu (o wartości logicznej 0), kolejne osiem bitów to bity danych (rozpoczynając od bitu najmłodszego LSB a kończąc na bicie najstarszym MSB), następnie jeden bit kontrolny (parzystości) i jeden bit stopu (o wartości logicznej 1).

Sekwencja (suma) kontrolna ramki (FCS) przesyłanej informacji jest obliczana w sposób przedstawiony na rysunku 3.11.



Rys.3.11. Przykład obliczania sumy kontrolnej

Wyznaczenie bitów parzystości jest realizowane dla pojedynczych znaków przesyłanych w ramce (*ang. Vertical Redundancy Check – VRC*) i dla odpowiednich bitów wszystkich znaków w ramce (*ang. Longitudinal Redundancy Check – LRC*).

3.5. Praca w sieci sterowników programowalnych serii PS4-150 i PS4-200

Sterowniki programowalne serii PS4-150 i PS4-200 posiadają dwa wbudowane sprzęgi komunikacji szeregowej: RS-232 i RS-485. Sprzęg standardu RS-232 jest przeznaczony głównie do podłączenia urządzenia programującego (komputer typu IBM/PC z programem narzędziowym Sucosoft S40), a sprzęg standardu RS-485 umożliwia budowę sieci [1,2].

W sterownikach tych wykorzystywany jest system komunikacji szeregowej pracujący na zasadzie kontrolowanego dostępu do magistrali (typu Master – Slave), w którym sterownik główny (master) ma uprawnienia do inicjatywy komunikacyjnej z urządzeniami podporządkowanymi (slaves). Wymiana danych odbywa się ekranowaną skrętką z szybkością

375 kBodów na odległość do 300 m lub 187,5 kBodów na odległość do 600 m. Wykorzystywany jest protokół Suconet K będący jednym ze standardów przemysłowych.

Sprzęgi RS-232 i RS-485 mogą być również wykorzystane do wymiany danych z innymi urządzeniami w kodzie przezroczystym (tj. przesyłanie bitów informacji bez dokonywania jakichkolwiek zmian w przesyłanych ramkach). Funkcja ta jest realizowana programowo przez specjalny blok funkcyjny SCO (*ang. Serial COmmunication function block*).

Sterowniki programowalne serii PS4-200 można lokalnie rozszerzać poprzez dołączanie modułów rozszerzenia lokalnego LE4; do sterownika PS4-201-MM1 można dołączyć lokalnie sześć, a do sterownika PS4-271-MM1 pięć modułów LE4. Sterowniki serii PS4-150 nie mogą być rozszerzane lokalnie.

Zarówno sterowniki z serii PS4-150 jak i z serii PS4-200, pracując jako sterowniki główne w sieci, mogą adresować maksymalnie osiem urządzeń podporządkowanych [1,2,22]. Tymi urządzeniami mogą być: kolejne sterowniki PS4 (pracujące jako aktywne lub pasywne urządzenia podporządkowane), moduły zewnętrzne EM4, moduły zewnętrzne EM4 z modułami rozszerzenia lokalnego LE4 (maksymalnie do sześciu modułów), oraz inne urządzenia wyposażone w Suconet K (wyświetlacze tekstu, panele operatorskie, przemienniki częstotliwości, urządzenia łagodnego rozruchu, przetworniki obrotowo-impulsowe i wiele innych).

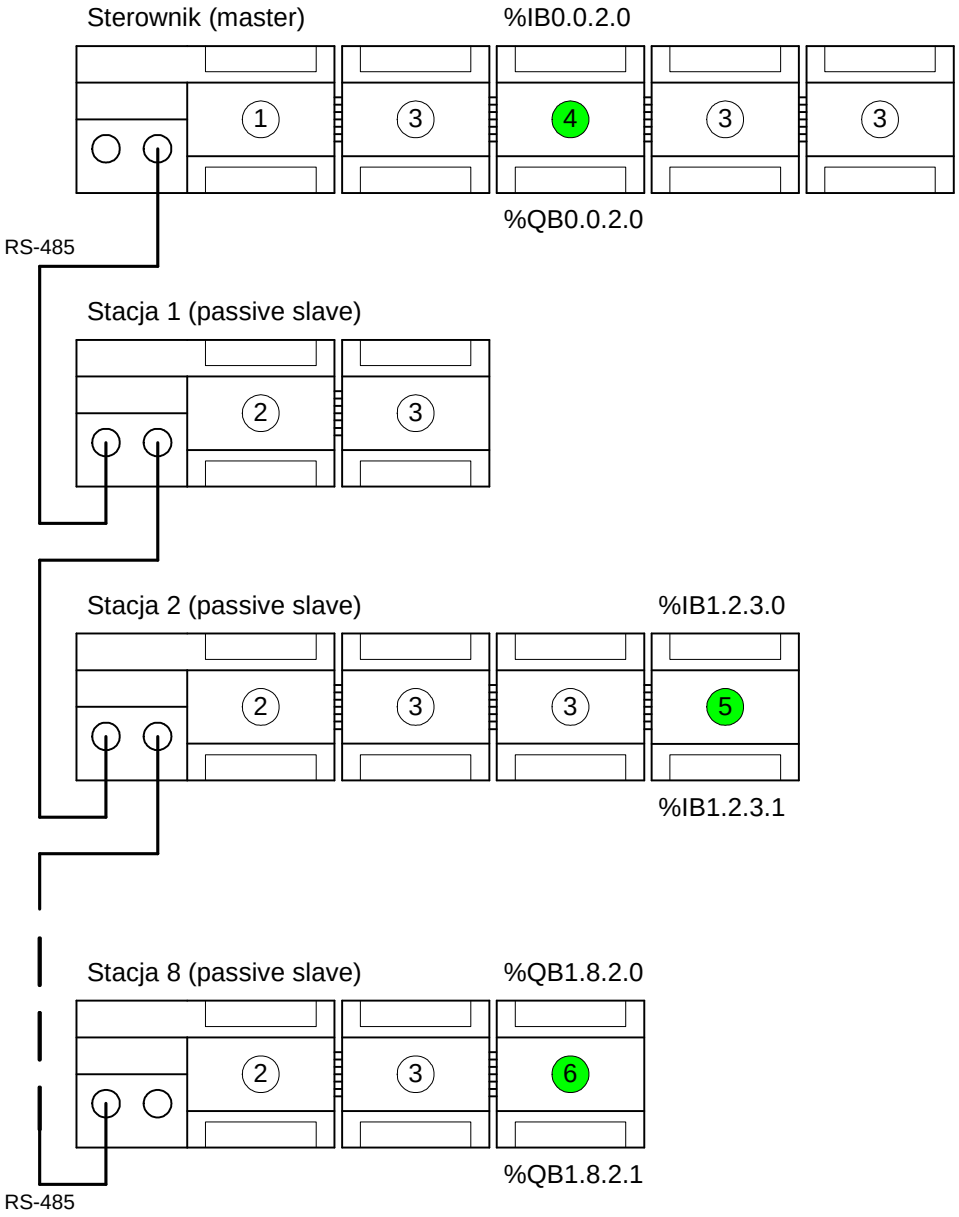
Na rysunku 3.12 przedstawiono przykład sieci zbudowanej w oparciu o sterownik PS4-201-MM1, moduły zewnętrzne EM4 i moduły rozszerzenia lokalnego LE4. Oprócz programu sterowania tworzony jest również plik konfiguracyjny (topologia) układu sterowania. Na podstawie danych zawartych w pliku konfiguracyjnym sterownik rozpoznaje typ każdego z urządzeń podporządkowanych pracujących w sieci i automatycznie nawiązuje z nim łączność.

Na przykładzie sterownika głównego i trzech modułów rozszerzenia lokalnego LE4 oznaczonych na rysunku 3.12 numerami 4, 5 i 6 omówiono poniżej zasady adresacji.

Przedstawiona na rysunku 3.12 sieć ze sterownikiem programowalnym PS4-201-MM1, z punktu widzenia wymiany danych w sieci, składa się z dwóch linii komunikacyjnych.

Pierwszą linię, ale adresowaną jako linia 0, stanowi sterownik główny i wszystkie moduły rozszerzenia lokalnego LE4 dołączone do niego. Sterownik i wszystkie dołączone do niego moduły LE4 są równocześnie urządzeniem (stacją) o numerze 0.

Drugą linię, ale adresowaną jako linia 1, tworzą wszystkie urządzenia podporządkowane (stacje) podłączone do sterownika poprzez sprzęg komunikacji szeregowej RS485.



Rys.3.12. Przykładowa sieć Suconet K z jednym sterownikiem PS4-201-MM1

Są to więc, jak pokazano na rysunku 3.12, moduły EM4 z ich rozszerzeniami lokalnymi LE4.

Adresy zmiennych są powiązane z topologią systemu sterowania, a ogólna zasada adresacji zmiennej jest następująca:

A R L . S . M . x . y

gdzie: **A** - argument operacji;

R - typ danej;

L - numer linii komunikacyjnej (0 lub 1)2;

S - numer stacji (od 0 do 8);

M - numer modułu (od 0 do 6);

x - numer bajtu (0 lub 1)3;

y - numer bitu (od 0 do 7).

Uwaga: W powyższym zapisie adresu zmiennej, dla większej czytelności tekstu, wprowadzono spacje, których przy poprawnej adresacji zmiennej nie powinno być (ARL.S.M.x.y).

Argumentami operacji mogą być:

%I - wejście binarne;

%IP - wejście binarne (odczyt bezpośredni z pominięciem pamięci obrazu wejść);

%Q - wyjście binarne;

%QP - wyjście binarne (aktualizacja bezpośrednia z pominięciem pamięci obrazu);

%IA - wejście analogowe;

%QA - wyjście analogowe;

%M - element (komórka) pamięci;

%SD - zmienna wysyłana w sieci;

%RD - zmienna odbierana w sieci.

Oznaczenia typów danych w adresacji zmiennej mogą być następujące: brak oznaczenia lub X – bit, B – bajt (8 bitów), W – słowo (16 bitów), a dla większych sterowników także D – słowo podwójne (32 bity).

Sterownik programowalny PS4-201-MM1 posiada 8 wejść binarnych, 6 wyjść binarnych, 2 zadajniki potencjometryczne (rozdzielczość przetwornika A/C 10 bitów), 2 wejścia analogowe (rozdzielczość przetwornika A/C 10 bitów) i 1 wyjście analogowe (rozdzielczość przetwornika C/A 12 bitów).

Wejścia binarne sterownika są adresowane od %I0.0.0.0.0 do %I0.0.0.0.7 (lub od

[2] W adresacji możliwe są również i wyższe numery linii komunikacyjnej (2 i 3) jeśli w systemie sterowania wykorzystuje się moduły komunikacyjne LE4.

[3] Numeracja 0 i 1 odnosi się do modułów z 16 wejściami lub wyjściami binarnymi. W ogólnym przypadku numery w numeracji mogą być także wyższe np. dla modułów z wejściami i/lub wyjściami analogowymi, lub też przy adresacji pamięci.

%IP0.0.0.0.0 do %IP0.0.0.0.7), a także mogą być adresowane jako bajt wejściowy, czyli %IB0.0.0.0 (lub %IPB0.0.0.0).

Analogicznie wyjścia binarne sterownika są adresowane od %Q0.0.0.0.0 do %I0.0.0.0.5 (lub od %QP0.0.0.0.0 do %QP0.0.0.0.5), a także mogą być adresowane jako bajt wyjściowy, czyli %QB0.0.0.0 (lub %QPB0.0.0.0).

Adresacja wejść i wyjść analogowych jest następująca:

%IAW0.0.0.0 - zadajnik potencjometryczny P1;

%IAW0.0.0.2 - zadajnik potencjometryczny P2;

%IAW0.0.0.4 - wejście analogowe U0;

%IAW0.0.0.6 - wejście analogowe U1;

%QAW0.0.0.0 - wyjście analogowe U10.

Pamięć można adresować bitowo (znacznik), bajtowo (bajt znacznikowy) lub słownie (słowo znacznikowe). Adresacja w sterownikach PS4 jest zorientowana bajtowo i dlatego bajty są numerowane od 0 kolejnymi liczbami całkowitymi, a słowa od 0 kolejnymi liczbami całkowitymi parzystymi.

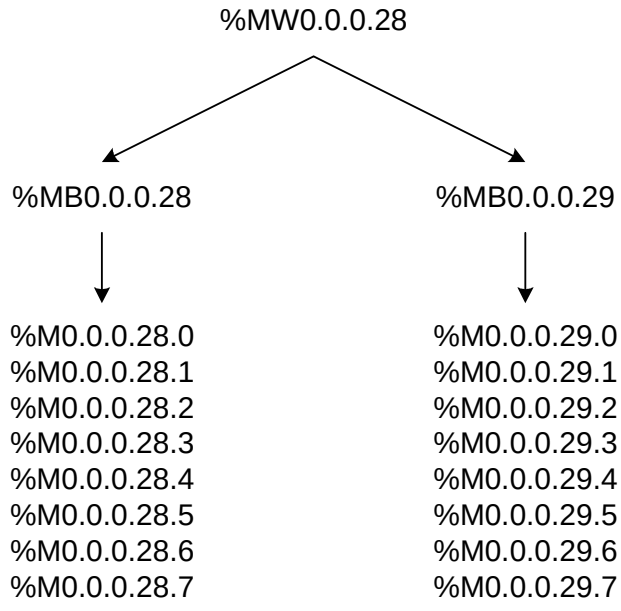
Znaczniki od %M0.0.0.0.0 do %M0.0.0.0.7 tworzą bajt znacznikowy %MB0.0.0.0 ;

znaczniki od %M0.0.0.1.0 do %M0.0.0.1.7 tworzą bajt znacznikowy %MB0.0.0.1 ;

dwa bajty znacznikowe %MB0.0.0.0 i %MB0.0.0.1 tworzą słowo znacznikowe %MW0.0.0.0 a kolejne %MB0.0.0.2 i %MB0.0.0.3 tworzą kolejne słowo znacznikowe %MW0.0.0.2 .

Należy przy tym zwrócić uwagę, że np. słowo znacznikowe %MW0.0.0.13 nie istnieje (jest to błędna adresacja bo numer słowa jest nieparzysty).

Przykład adresacji pamięci w sterownikach PS4 przedstawiono na rysunku 3.13.



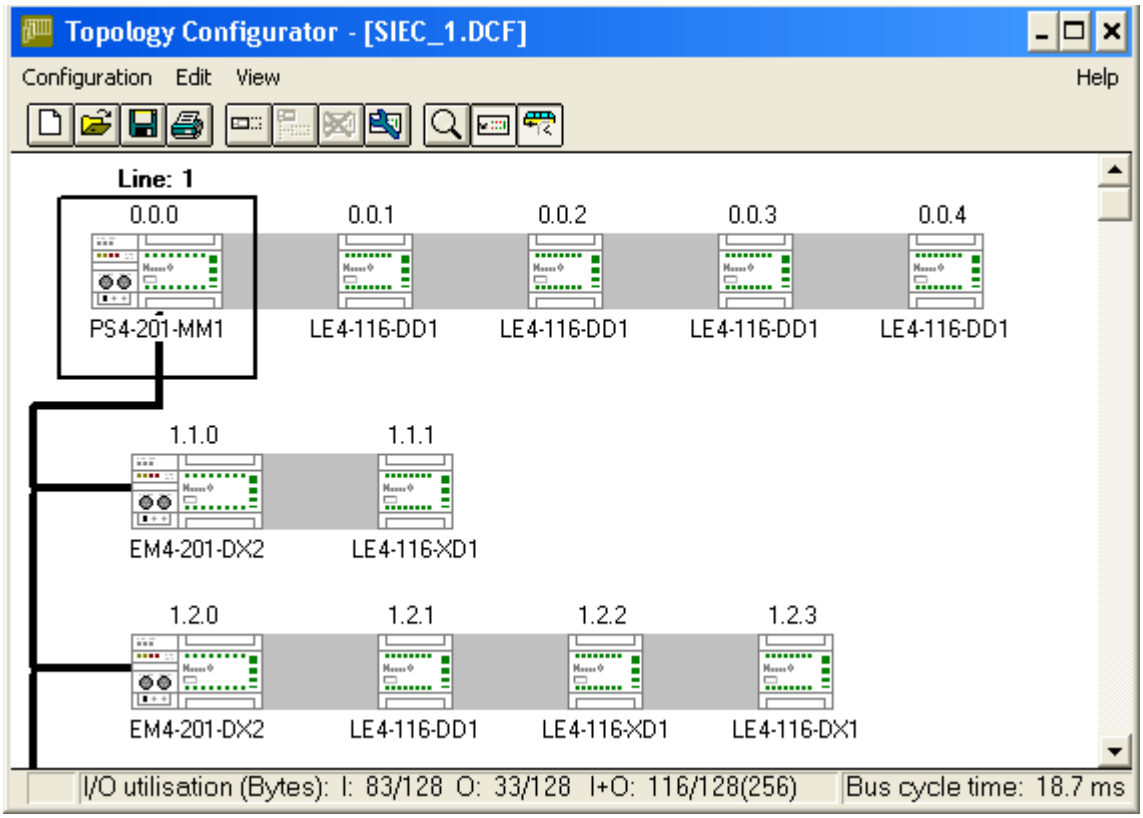
Rys.3.13. Przykład adresacji pamięci w sterownikach PS4 (słowo → bajty → bity)

Numerem 4 na rysunku 3.12 oznaczono moduł rozszerzenia lokalnego LE4-116-DD1. Jest to moduł 8 wejść binarnych i 8 wyjść binarnych.
Adresy jego wejść rozpoczynają się od %I0.0.2.0.0 a kończą na %I0.0.2.0.7 , ale wejścia mogą być również odczytane jako bajt wejściowy %IB0.0.2.0 (co pokazano na rysunku).
Adresy jego wyjść rozpoczynają się od %Q0.0.2.0.0 a kończą na %Q0.0.2.0.7 , ale wyjścia mogą być również wystawione jako bajt wyjściowy %QB0.0.2.0 (co również pokazano na rysunku).
Numerem 5 na rysunku 3.12 oznaczono moduł rozszerzenia lokalnego LE4-116-DX1. Jest to moduł 16 wejść binarnych.
Adresy jego wejść rozpoczynają się od %I1.2.3.0.0 a kończą na %I1.2.3.1.7 , ale wejścia mogą być również odczytane jako dwa bajty wejściowe %IB1.2.3.0 i %IB1.2.3.1 (co pokazano na rysunku) lub jako słowo wejściowe %IW1.2.3.0 .
Numerem 6 na rysunku 3.12 oznaczono moduł rozszerzenia lokalnego LE4-116-XD1. Jest to moduł 16 wyjść binarnych.
Adresy jego wyjść rozpoczynają się od %Q1.8.2.0.0 a kończą na %Q1.8.2.1.7 , ale wyjścia mogą być również wystawione jako dwa bajty wyjściowe %QB1.8.2.0 i %QB1.8.2.1 (co pokazano na rysunku) lub jako słowo wyjściowe %QW1.8.2.0 .

W sieci przedstawionej na rysunku 3.12 wszystkie moduły zewnętrzne EM4 z modułami rozszerzenia lokalnego LE4 zwiększają liczbę wejść/wyjść systemu sterowania adresowanych przez sterownik główny. Są więc pasywnymi urządzeniami podporządkowanymi (passive slaves).

Na rysunku 3.14 pokazana jest topologia tej sieci (początkowy fragment) utworzona za pomocą konfiguratora topologii w programie narzędziowym Sucosoft S40. Na dolnym pasku stanu podane są: liczba bajtów danych przesyłanych w sieci oraz obliczony czas potrzebny na wymianę tych danych w sieci. Liczba bajtów przesyłanych w sieci zależy od liczby i rodzaju modułów pracujących w sieci; jest ona sumą liczby bajtów wynikających z liczby i rodzaju wejść i wyjść tych modułów, oraz bajtów diagnostycznych.

Inny przykład sieci przedstawiono na rysunku 3.15. W tej sieci pracują cztery sterowniki programowalne PS4-201-MM1: jeden sterownik główny (master) i trzy sterowniki będące aktywnymi urządzeniami podporządkowanymi (active slaves). Oprócz sterownika głównego, także każdy sterownik pracujący jako active slave realizuje swój program sterowania; odczytuje stany swoich wejść, wykonuje program sterowania i aktualizuje stany swoich wyjść.

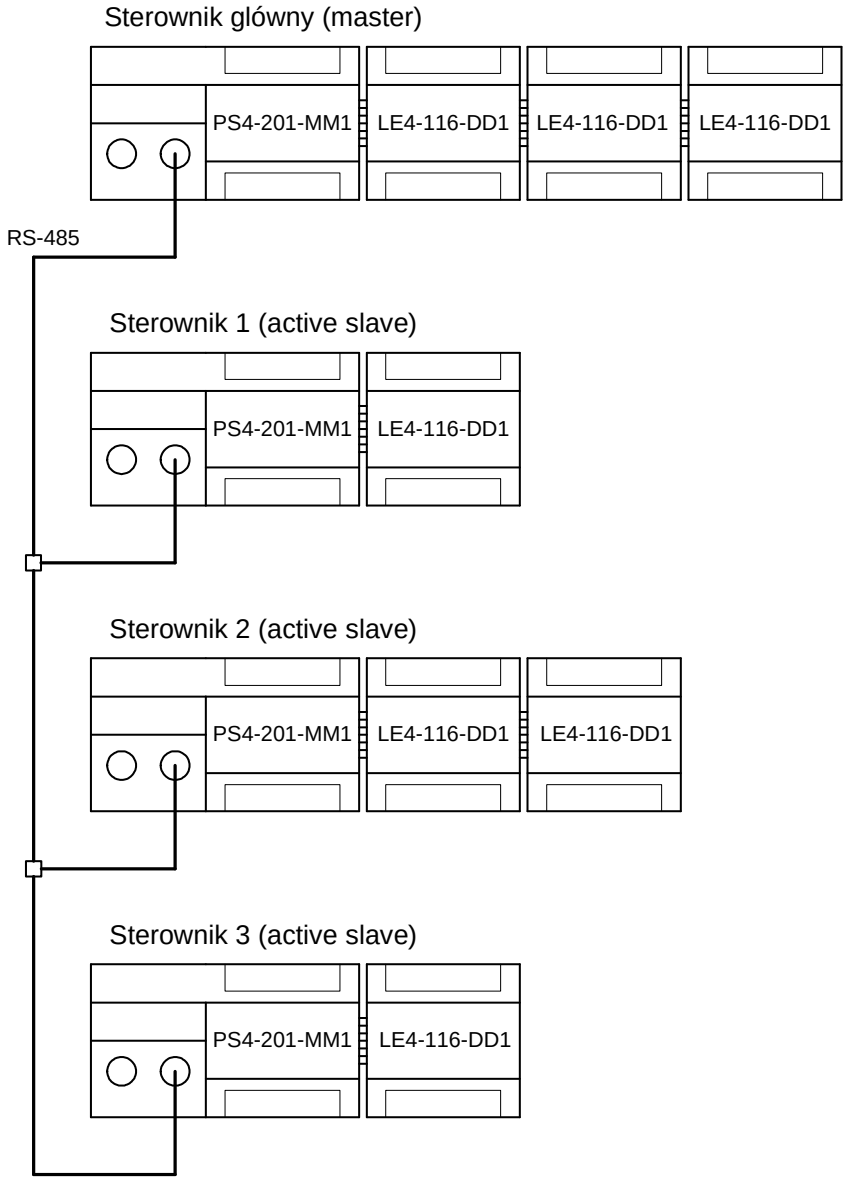


Rys.3.14. Topologia sieci (fragment początkowy) z rysunku 3.12 utworzona za pomocą konfiguratora w programie narzędziowym Sucosoft S40

Oprócz wykonywania swoich programów sterowania pracujące w sieci sterowniki prowadzą także wymianę danych w sieci. Oczywiście, zgodnie z metodą sterowania dostępem do łącza sieciowego typu master-slave, inicjatywę komunikacyjną posiada tylko sterownik główny (master), a pozostałe sterowniki (active slaves) odpowiadają tylko na zapytania sterownika głównego.

Sterownik główny (master) z serii PS4-150 lub PS4-200 może w sieci Suconet K wysyłać maksymalnie 128 bajtów i odbierać maksymalnie 128 bajtów. Sterownik podporządkowany (active slave) może w sumie odbierać i wysyłać maksymalnie 78 bajtów (np. 10 bajtów wysyłać i 10 bajtów odbierać lub 28 wysyłać i 50 odbierać).

Zasada adresacji zmiennych wysyłanych i odbieranych w sieci jest analogiczna do adresacji omówionej na przykładzie sieci ze sterownikiem głównym PS4-201-MM1, modułami zewnętrznymi EM4 i modułami rozszerzenia lokalnego LE4, przedstawionej na rysunku 3.12. Należy pamiętać, że dla danych przesyłanych w sieci adresacja również jest zorientowana bajtowo.



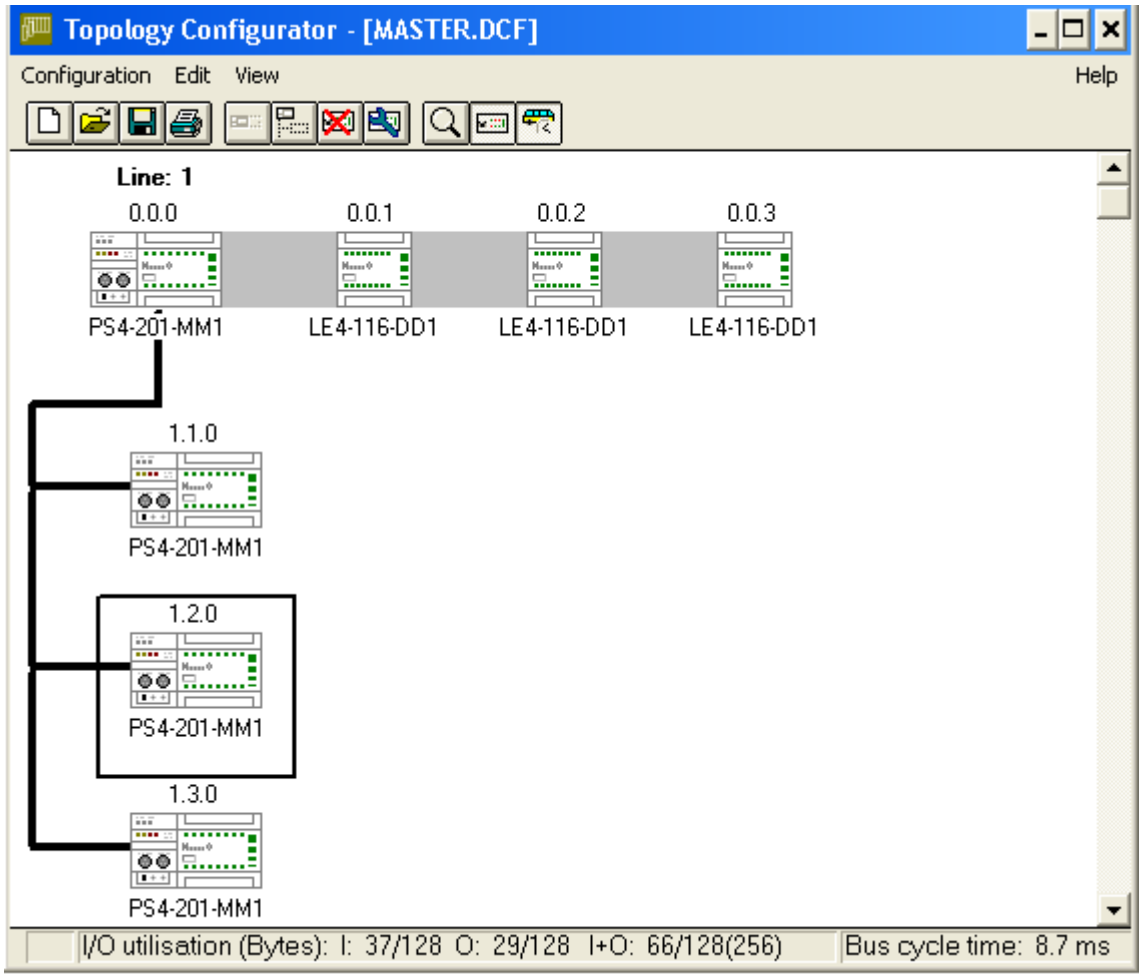
Rys.3.15. Przykładowa sieć Suconet K z czterema sterownikami PS4-201-MM1

Za pomocą argumentu operacji %SD (%RD) z rozszerzeniem B, oznaczającym zmienne bajtowe przesyłane w sieci, można adresować dowolne zmienne dla których typ danych do zapisu ich wartości wymaga 8 bitów, czyli adresacja %SDB (%RDB) odpowiada zmiennym o następujących typach: BYTE, SINT, USINT.

Za pomocą tego samego argumentu operacji z rozszerzeniem W, oznaczającym zmienne słowne przesyłane w sieci, można adresować dowolne zmienne dla których typ danych do zapisu ich wartości wymaga 16 bitów, czyli adresacja %SDW (%RDW) odpowiada zmiennym o następujących typach: WORD, INT i UINT.

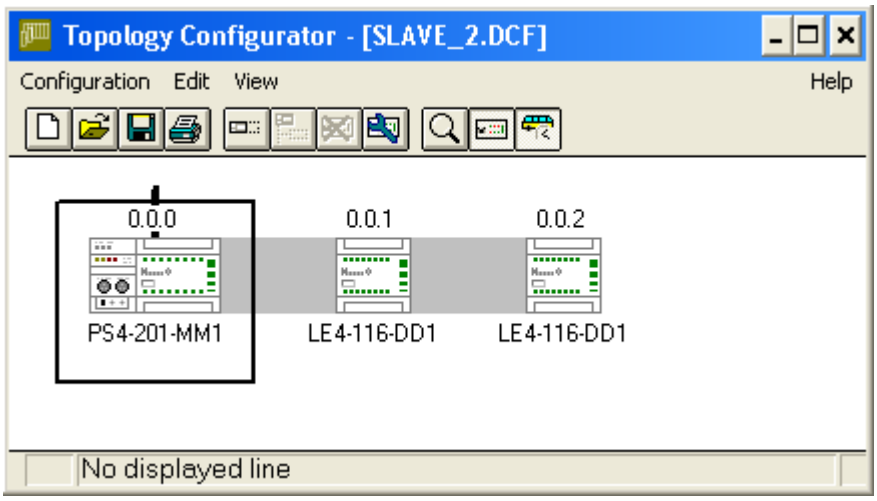
Każdy sterownik pracujący w sieci i wykonujący program sterowania wymaga stworzenia zbioru konfiguracyjnego zawierającego jego topologię systemu sterowania.

Topologia dla każdego sterownika obejmuje tylko te elementy sieci które ten sterownik obsługuje (których wejścia odczytuje i których wyjścia aktualizuje, oraz te którymi zarządza w sieci). Dlatego w topologii dla sterownika głównego są uwzględnione podłączone do niego moduły rozszerzenia lokalnego LE4 i pozostałe sterowniki pracujące w sieci, ale bez ich modułów rozszerzenia lokalnego LE4. Topologia sieci dla sterownika głównego jest przedstawiona na rysunku 3.16.



Rys.3.16. Topologia sieci z rysunku 3.15 dla sterownika głównego

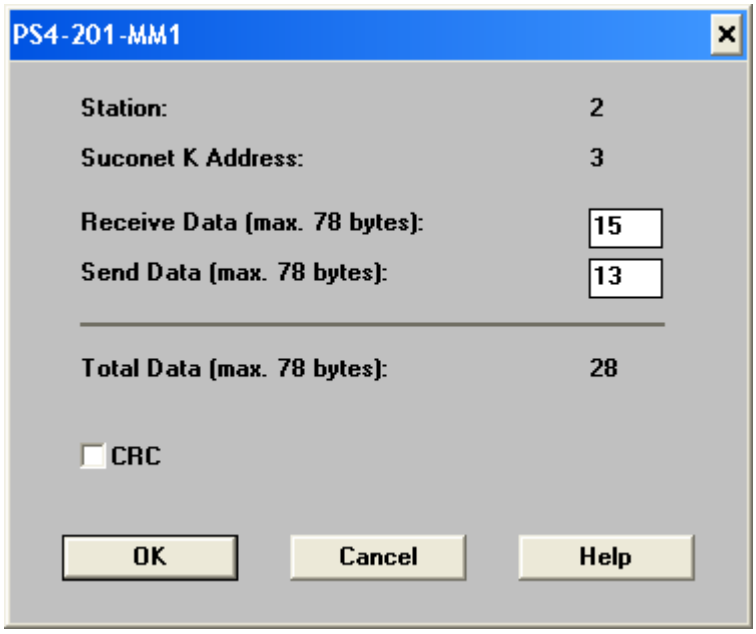
Zgodnie z powyższą zasadą topologie sieci dla sterowników podporządkowanych od 1 do 3 zawierają tylko sam sterownik (skonfigurowany jako active slave) oraz podłączone do niego moduły rozszerzenia lokalnego LE4. Na rysunku 3.17 przedstawiono (dla przykładu) topologię sieci dla sterownika 2.



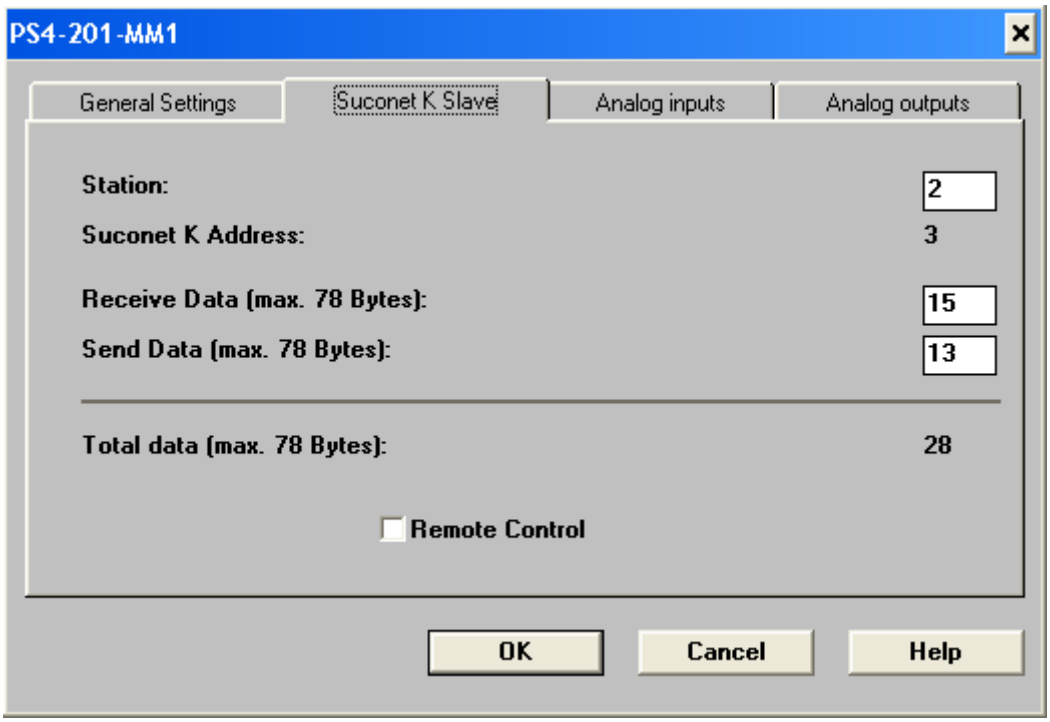
Rys.3.17. Topologia sieci z rysunku 3.15 dla sterownika 2

W topologiach dla każdego ze sterowników należy zadeklarować liczby bajtów danych przesyłanych w sieci dla każdego sterownika (wysyłanych i odbieranych). W topologii dla sterownika master i w topologii dla sterownika active slave zadeklarowane liczby bajtów danych wysyłanych i odbieranych muszą być takie same.

Na rysunku 3.18 przedstawiono deklarowanie przesyłanych w sieci danych dla sterownika 2 w topologii sterownika głównego (master), a na rysunku 3.19 przedstawiono deklarowanie przesyłanych w sieci danych dla tego samego sterownika 2, ale w topologii sterownika 2 (active slave).



Rys.3.18. Deklarowanie liczby bajtów danych przesyłanych w sieci dla sterownika 2 w topologii sterownika głównego



Rys.3.19. Deklarowanie liczby bajtów danych przesyłanych w sieci dla sterownika 2 w topologii sterownika 2

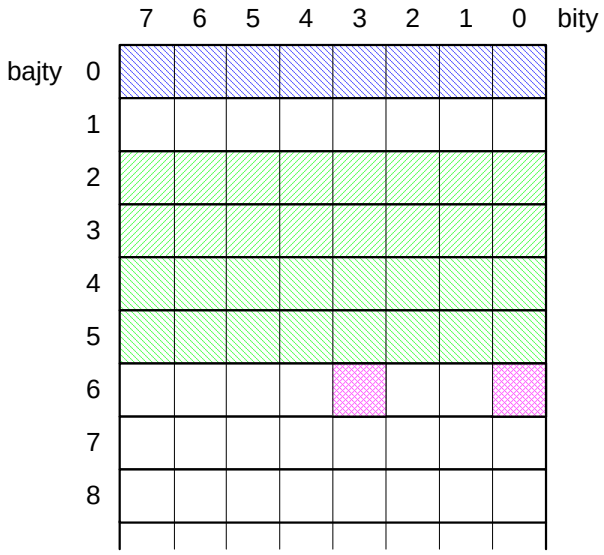
Przykład 1.

W sieci przedstawionej na rysunku 3.15 należy ze sterownika głównego (master) do sterownika 1 (active slave) przesłać: jedną zmienną typu BYTE, dwie zmienne typu INT i dwie zmienne typu BOOL. Jeśli zachowa się kolejność przesyłania i przyporządkuje im się poniżej podane adresy

BYTE → %SDB1.1.0.0 ;
INT → %SDW1.1.0.2 ;
INT → %SDW1.1.0.4 ;
BOOL → %SD1.1.0.6.0 ;
BOOL → %SD1.1.0.6.3 ,

to zmienne te będą wpisane do bufora komunikacyjnego tak, jak pokazano na rysunku 3.20.

Sterownik 1 odczyta je jako: %RDB0.0.0.0 ;
%RDW0.0.0.2 ;
%RDW0.0.0.4 ;
%RD0.0.0.6.0 ;
%RD0.0.0.6.3 .



Rys.3.20. Przykład pierwszy przesyłania danych w sieci Suconet K

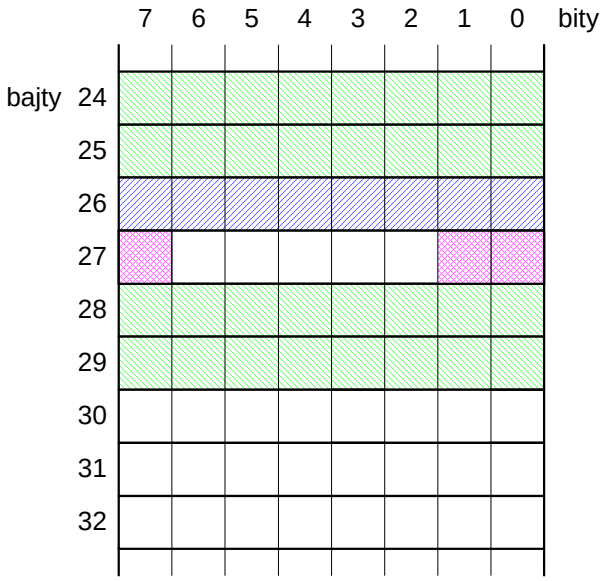
Przykład 2.

W sieci przedstawionej na rysunku 3.15 należy ze sterownika głównego (master) do sterownika 3 (active slave) przesłać: jedną zmienną typu WORD, jedną zmienną typu USINT trzy zmienne typu BOOL i jedną zmienną typu UINT. Jeśli zachowa się kolejność przesyłania i przyporządkuje im się poniżej podane adresy

WORD	→	%SDW1.3.0.24 ;
USINT	→	%SDB1.3.0.26 ;
BOOL	→	%SD1.3.0.27.0 ;
BOOL	→	%SD1.3.0.27.1 ;
BOOL	→	%SD1.3.0.27.7 ;
UINT	→	%SDW1.3.0.28 ,

to zmienne te będą wpisane do bufora komunikacyjnego tak, jak pokazano na rysunku 3.21.

Sterownik 3 odczyta je jako: %RDW0.0.0.24 ;
%RDB0.0.0.26 ;
%RD0.0.0.27.0 ;
%RD0.0.0.27.1 ;
%RD0.0.0.27.7 ;
%RDW0.0.0.28 .



Rys.3.21. Przykład drugi przesyłania danych w sieci Suconet K

Przesyłanie danych pomiędzy sterownikami podporządkowanymi (active slaves) wymaga pośrednictwa sterownika głównego (mastera).

Przykład 3.

Jeśli ze sterownika 3 trzeba przesłać do sterownika 2 jedną zmienną typu BYTE, to można to zrobić następująco:

Sterownik 3		Sterownik główny		Sterownik 2
%SDB0.0.0.0	→	%RDB1.3.0.0		
		%SDB1.2.0.0	→	%RDB0.0.0.0

Dane w sieci mogą być przesyłane nie tylko pomiędzy sterownikiem głównym (master) i sterownikami podporządkowanymi (active slaves), ale również pomiędzy sterownikiem głównym a innymi urządzeniami podłączonymi do sieci.

Gdy sterowniki programowalne pracują w sieci, w niektórych zadaniach automatyzacji bardzo istotne jest określenie czasu reakcji układu sterowania w odniesieniu do sygnałów doprowadzonych i wyprowadzanych ze sterownika (master) oraz w odniesieniu do sygnałów doprowadzonych i wyprowadzanych z modułów podłączonych do sieci (passive slave).

Można wyróżnić dwa przypadki:

- cykl wykonania programu użytkownika trwa dłużej niż cykl wymiany danych w sieci;
- cykl wykonania programu użytkownika trwa krócej niż cykl wymiany danych w sieci.

W pierwszym przypadku w każdym cyklu wykonywania programu użytkownika sterownik dysponuje zaktualizowanymi danymi w pamięci obrazu procesu (sygnały z wejść sterownika i wszystkich podłączonych do niego modułów rozszerzenia lokalnego) oraz zaktualizowanymi danymi z bufora komunikacyjnego (sygnały z wejść wszystkich modułów sieciowych i innych układów automatyki podłączonych poprzez sieć). W przypadku wyjść aktualizacja wyjść w modułach podłączonych poprzez sieć, w porównaniu z aktualizacją wyjść sterownika, następuje z opóźnieniem czasu jednego cyklu wymiany danych w sieci.

W drugim przypadku sterownik dysponuje zaktualizowanymi danymi w pamięci obrazu procesu w każdym cyklu programu, ale nie w każdym cyklu programu będzie dysponował zaktualizowanymi danymi z bufora komunikacyjnego. Czas reakcji układu sterowania dla sygnałów z sieci (odczytywanych i wysyłanych) będzie dłuższy niż dla sygnałów dostępnych w sterowniku i podłączonych do niego modułach rozszerzenia lokalnego.

Dla topologii z rysunku 3.14 czas cyklu wymiany danych w sieci wynosi 18,7 ms, a dla topologii z rysunku 3.16 wynosi 8,7 ms. Zakładając, że czas cyklu wykonania programu wynosi 10 ms, w sieci z rysunku 3.14 ma miejsce drugi przypadek, a w sieci z rysunku 3.16 pierwszy z nich.

Na Rys.3.22. przedstawiono skrótowe podsumowanie charakterystyki sieci miejscowej Suconet K.

Struktura sieci → magistrala
Metoda dostępu do łącza sieciowego → master – slave
Sprzęg komunikacyjny → RS 485
Medium transmisji → ekranowana skrętka
Max długość sieci → 600 m (2400 m – z trzema wzmacniaczami)
Szybkość transmisji → 187,5 kB/s (przy długości sieci do 600 m)
→ 375 kB/s (przy długości sieci do 300 m)
Jedno urządzenie nadrzędne (master) w sieci
Max liczba urządzeń podporządkowanych (slave) →
→ 8 (dla PS4-150 i PS4-200) i 30 (dla PS4-300)
Max liczba przesyłanych bajtów informacji (dla PS4-150 i PS4-200) →
→ Master 128/128 bajtów, Slave 78 bajtów (Σ)
Informacja przesyłana w polu danych o zmiennej długości
Pojedynczy znak w ramce o długości 11 bitów

Rys.3.22. Charakterystyka sieci miejscowej Suconet K – podsumowanie

3.6. System i protokół komunikacyjny AS-i

W klasycznym systemie sterowania poziom drugi (w strukturze informatycznej przedsiębiorstwa) jest ostatnim, na którym wykorzystuje się sieci do łączenia poszczególnych urządzeń i elementów w systemie sterowania. Połączenia pomiędzy pracującymi na tym poziomie sterownikami programowalnymi są realizowane za pomocą sieci miejscowych, natomiast pomiędzy sterownikami programowalnymi a czujnikami, urządzeniami wykonawczymi i układami wejścia/wyjścia za pomocą pojedynczych przewodów prowadzonych do każdego elementu równolegle. Połączenia są wykonywane pomiędzy układami wejściowymi i wyjściowymi sterowników programowalnych (lub dołączonych do nich modułów wejściowych i wyjściowych) a poszczególnymi czujnikami i urządzeniami wykonawczymi układu sterowania. Wszystkie te połączenia tworzą okablowanie układu sterowania. Biorąc pod uwagę, że każdy element nie wymagający własnego zasilania (np. przycisk sterowania lub wyłącznik krańcowy) do podłączenia do sterownika wymaga dwóch przewodów, a każdy element wymagający własnego zasilania (np. czujnik indukcyjny lub optyczny) do podłączenia do sterownika wymaga trzech przewodów, liczba takich przewodów w układzie sterowania może być bardzo duża. Niezależnie od tego, że połączenia tymi przewodami wykonuje się w odpowiedni uporządkowany sposób (łączy w wiązki przewodów, prowadzi w korytkach kablowych, itp.) często mówi się o tzw. „plątaninie przewodów” (ang. *cable harness*). Jedną z niewielu zalet takiego łączenia czujników i urządzeń wykonawczych jest to, że nie ma praktycznie opóźnienia w doprowadzeniu sygnałów z czujników do wejść sterownika oraz w doprowadzeniu sygnałów z wyjść sterownika do urządzeń wykonawczych.

Na początku lat dziewięćdziesiątych ubiegłego wieku firmy z branży automatyki przemysłowej (początkowo 11, ale w krótkim czasie dołączyło do nich wiele następnych) postanowiły opracować wspólny system sieciowy, który umożliwiałby podłączenie do niego czujników i urządzeń wykonawczych. Opracowanie takiego systemu było bardzo celowe, gdyż z badań przeprowadzonych w przemyśle wynikało [53], że ponad 80% sygnałów w układach sterowania to sygnały binarne, oraz możliwe, gdyż w przypadku 80% tych sygnałów wystarczające było ich uaktualnianie w czasie nie przekraczającym 10 ms. Opracowany system został nazwany magistralą czujników i urządzeń wykonawczych AS-i (*ang. Actuator Sensor Interface*). Produkcję i sprzedaż rozpoczęto na przełomie 1993 i 1994 roku i w pierwszym roku sprzedano ok. 300 tysięcy elementów tego systemu [53].

Z założeń, które zostały przyjęte przy opracowywaniu magistrali czujników i urządzeń wykonawczych wynikało, że system powinien:

- umożliwiać aktualizację sygnałów w czasie nie dłuższym niż 5 ms;
- być prosty w montażu i uruchomieniu oraz umożliwiać łatwą rozbudowę;
- umożliwiać podłączenie nie tylko elementów systemu AS-i ale i tradycyjnych czujników i elementów wykonawczych;
- gwarantować większą niezawodność niż tradycyjne łączenia przewodami, a w przypadku uszkodzenia elementu umożliwiać łatwą i szybką jego wymianę;
- być odporny na zakłócenia elektromagnetyczne;
- zapewniać odpowiednią klasę ochrony elementów systemu (instalacja na obiekcie w warunkach przemysłowych);
- zapewniać łatwą diagnostykę elementów systemu – zarówno bezpośrednio na obiekcie, jak i poprzez sieć;
- po opracowaniu stać się (w krótkim czasie) standardem międzynarodowym;

Z perspektywy czasu można stwierdzić, że wszystkie powyższe założenia zostały spełnione i obecnie jest to najbardziej rozpowszechniony w przemyśle standard magistrali szeregowej na pierwszym poziomie w hierarchii sterowania.

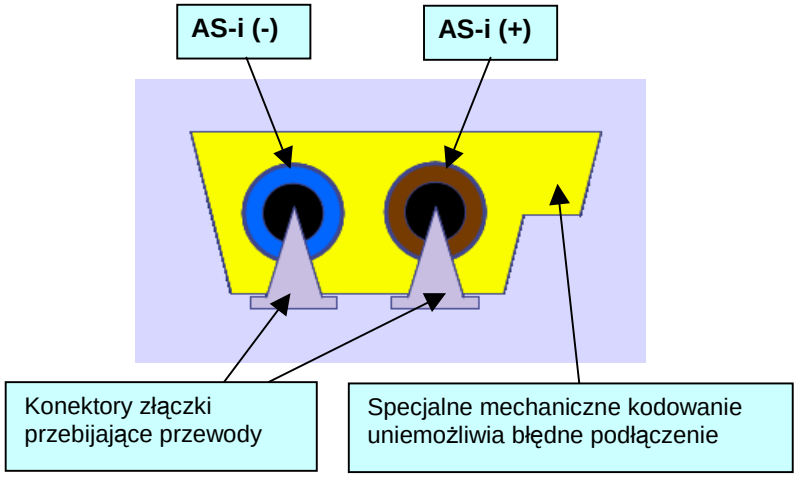
Magistrala czujników i urządzeń wykonawczych może mieć dowolną topologię, ale długość tej magistrali jest ograniczona do 100 m. Możliwe jest jednak zastosowanie jednego lub dwóch wzmacniaczy sygnału, aby zwiększyć jej maksymalną długość (o kolejne 100 m dla każdego wzmacniacza). W pojedynczej sieci mogą pracować maksymalnie 32 urządzenia: 1 urządzenie nadrzędne (master) oraz 31 urządzeń podporządkowanych (slaves); wykorzystywaną metodą dostępu do magistrali jest metoda master – slave.

Każdy element systemu AS-i jest wyposażony w taki sam specjalny układ scalony ASIC (*ang. Application Specific Integrated Circuit*), który realizuje komunikację w tej sieci. Dzięki temu elementy AS-i różnych producentów mogą bez problemu współpracować ze sobą w jednej sieci (elementy AS-i podlegają międzynarodowej certyfikacji).

Elementem systemu AS-i może być pojedynczy czujnik lub urządzenie wykonawcze (posiadające układ scalony AS-i) lub tzw. moduł interfejsu (specjalna skrzynka rozdzielcza z układem scalonym AS-i), do której można podłączyć do 4 typowych czujników lub urządzeń wykonawczych.

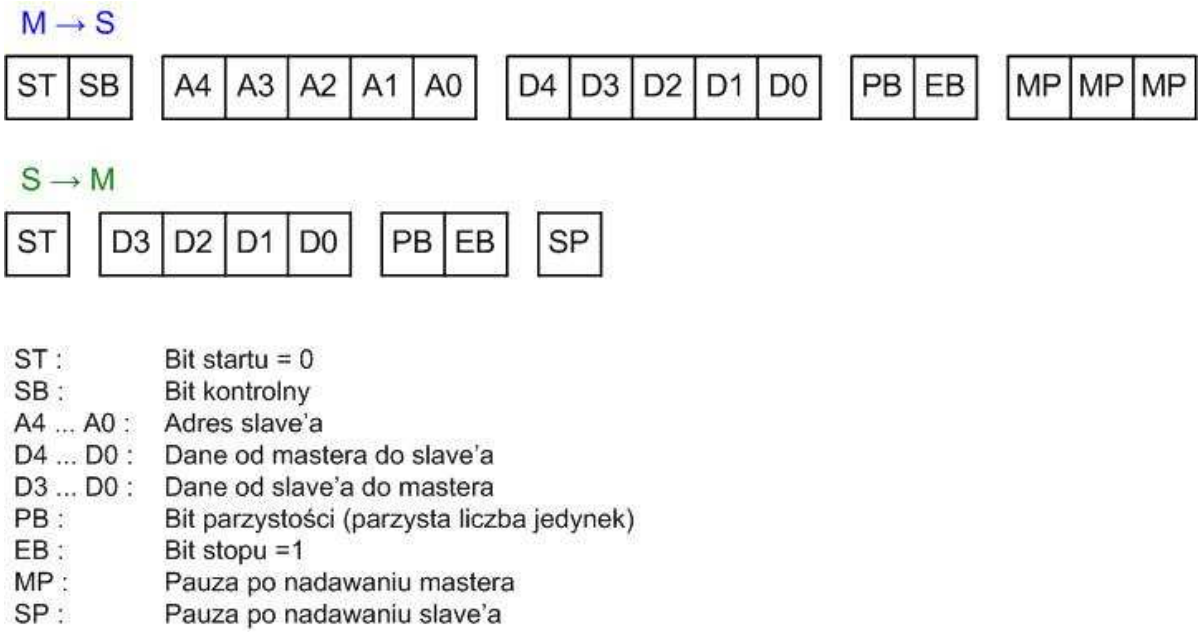
Połączenie elementów AS-i w sieć może być wykonane albo specjalnym, płaskim przewodem profilowym, którego przekrój pokazano na rysunku 3.23, lub też okrągłym,

nieekranowanym przewodem dwużyłowym (o przekroju żył od 1,5 do 2,5 mm^2). Przewody te są nie tylko medium transmisji, ale również doprowadzają napięcie zasilania (24 V=) do podłączonych elementów. Ponieważ tymi samymi przewodami doprowadzane jest napięcie i przesyłane są dane, w systemie AS-i niezbędny jest specjalny zasilacz, który, oprócz funkcji zasilania, realizuje również separację galwaniczną magistrali AS-i od pozostałych obwodów elektrycznych systemu sterowania.



Rys. 3.23. Przekrój specjalnego przewodu profilowego systemu AS-i

Format ramek przesyłanych po magistrali czujników i urządzeń wykonawczych za pomocą protokołu AS-i przedstawiono na rysunku 3.24.



Rys.3.24. Format ramek w protokole AS-i

Masterem sieci AS-i najczęściej jest moduł komunikacyjny modułowego sterownika programowalnego, moduł rozszerzenia lokalnego sterownika kompaktowego, lub też brama (*ang. gateway*) połączona ze sterownikiem programowalnym poprzez sieć miejscową [3,43].

Master do slave'a przesyła 17 bitów (wliczając w to 3 bity pauzy po nadawaniu mastera), a slave do mastera przesyła w odpowiedzi 8 bitów (wliczając w to 1 bit pauzy po nadawaniu slave'a). Łącznie jest to więc 25 bitów przesyłanych w komunikacji pomiędzy masterem i jednym urządzeniem slave.

Szybkość transmisji w sieci AS-i wynosi 167 kb/s , więc przesłanie jednego bitu trwa 6 μs , a czas potrzebny na wymianę danych z jednym urządzeniem slave wynosi 150 μs. Przy maksymalnej liczbie elementów w sieci AS-i (czyli 31) czas potrzebny na wymianę danych pomiędzy masterem i wszystkimi urządzeniami slave wynosi 4650 μs.

Dla 31 urządzeń slave pełny cykl komunikacji w sieci AS-i składa się z:

- 31 telegramów wysyłanych do urządzeń slave;
- jednego telegramu rozpoznającego nowe urządzenie slave;
- jednego telegramu adresującego nowe urządzenie slave lub jednego telegramu parametryzującego.

Czas potrzebny na dwa dodatkowe, wymienione powyżej, telegramy wchodzące w skład pełnego cyklu komunikacji wynosi 300 μs.

Pełny cykl komunikacji w sieci AS-i trwa więc 4950 μs. Spełnione jest więc założenie, że pełna aktualizacja danych dokonywana jest w czasie nie przekraczającym 5 ms.

Powyższe rozważania dotyczą systemu AS-i w wersji 1.1. W systemie AS-i w wersji 2.1 do pojedynczej sieci mogą być podłączone maksymalnie 62 urządzenia slave, więc czas wymiany danych będzie też dwukrotnie dłuższy.

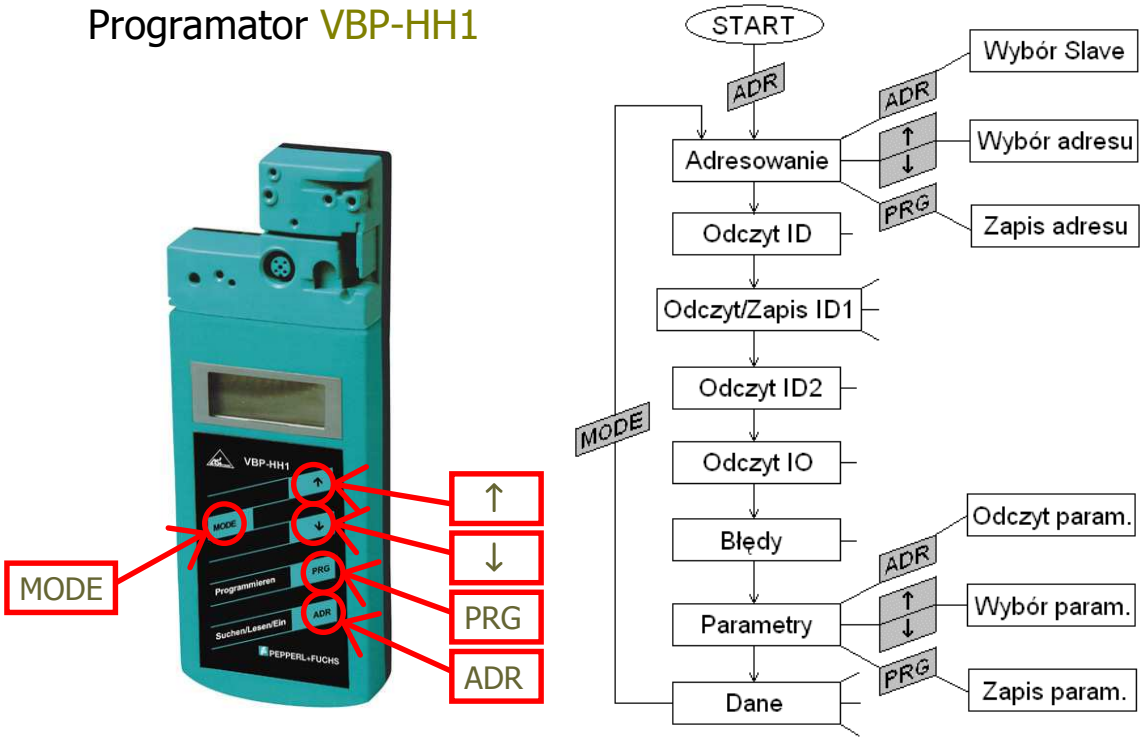
Specjalny układ scalony AS-i w urządzeniu slave jest identyfikowany przez trzy parametry:

- adres urządzenia (*slave address*);
- kod we/wy (*I/O code*) określający co przesyłane jest w bitach danych;
- kod identyfikacyjny (*ID code*) dla rozróżnienia, gdy taki sam jest I/O code.

Dwa ostatnie parametry są nazywane profilem (np. 1.1 dla czujników). Pierwsza jedynka w oznaczeniu to kod we/wy (w tym przypadku 3 bity wejściowe i 1 bit wyjściowy); druga jedynka oznacza kod identyfikacyjny.

Każde wyprodukowane urządzenie slave systemu AS-i ma fabrycznie nadany adres 0. Urządzenia slave w sieci AS-i mogą mieć adresy od 1 do 31 (pięć bitów adresowych w ramce przesyłanej od mastera do slave'a określa adres slave'a).

Do nadania adresu urządzeniu slave o adresie 0 można wykorzystać programator ręczny. Przykładowy programator ręczny oraz opis realizowanych przez niego funkcji przedstawiono na rysunku 3.25.



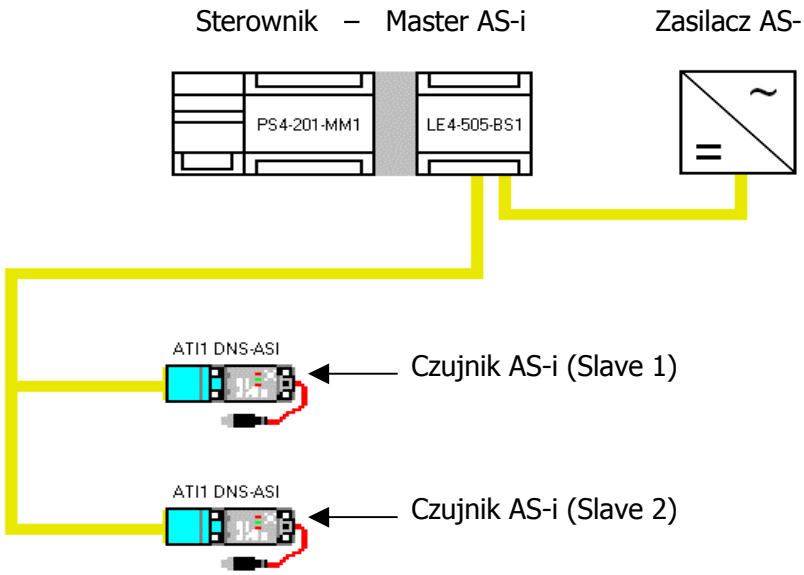
Rys.3.25. Przykładowy ręczny programator elementów systemu AS-i oraz jego funkcje

Do nadania adresu urządzeniu slave o adresie 0 można także wykorzystać mastera sieci AS-i. Master, podobnie jak programator ręczny może nadać adres pojedynczemu urządzeniu slave podłączonemu do sieci AS-i. Wówczas adresy wszystkim urządzeniom slave nadaje się podłączając kolejno pojedyncze urządzenia slave do magistrali i programując je. Można również zaprogramować nie jedno, a wszystkie urządzenia slave o adresach 0 podłączone do sieci. Wówczas master nada im kolejne adresy (rozpoczynając od adresu 1) w sposób automatyczny.

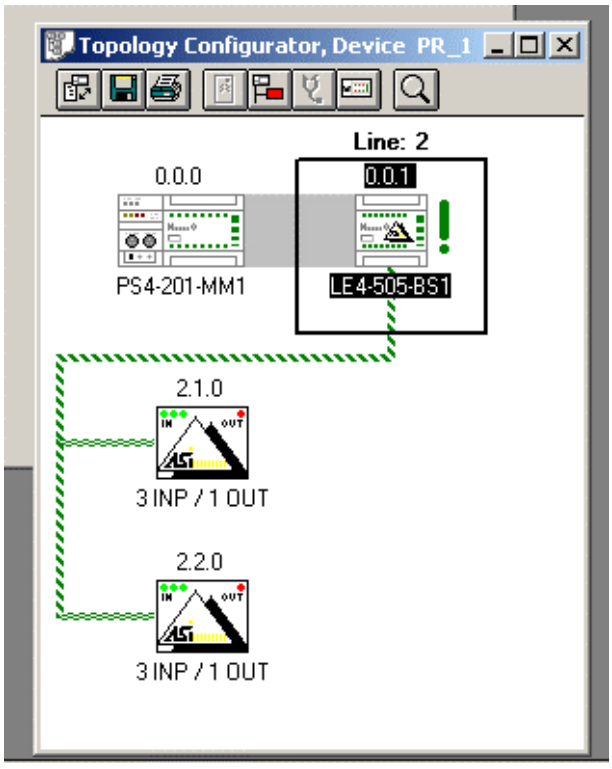
Na rysunku 3.26 przedstawiono przykładową magistralę AS-i. Układ składa się ze sterownika programowalnego PS4-201-MM1, dołączonego do niego modułu rozszerzenia lokalnego LE4-505-BS1, który jest masterem AS-i, zasilacza AS-i SN4-024-DI7 oraz dwóch czujników indukcyjnych AS-i ATI1 DNS-ASI.

Na rysunku 3.27 przedstawiono utworzoną w oprogramowaniu Sucosoft S40 topologię układu w sposób automatyczny. Jak widać z informacji wyświetlonej w topologii urządzeniom slave przyporządkowane zostały kolejne adresy 1 i 2 (w adresach wyświetlonych w topologii 2.1.0 i 2.2.0 druga cyfra odnosi się do adresu slave'a) oraz

wyświetlone zostały informacje o liczbie ich wejść i wyjść (3 INP / 1 OUT), co wynika z odczytanego profilu 1.1 tych czujników. Typ czujnika (oznaczenie producenta) nie został wyświetlony, gdyż informacja ta nie jest zapisana w profilu czujnika.

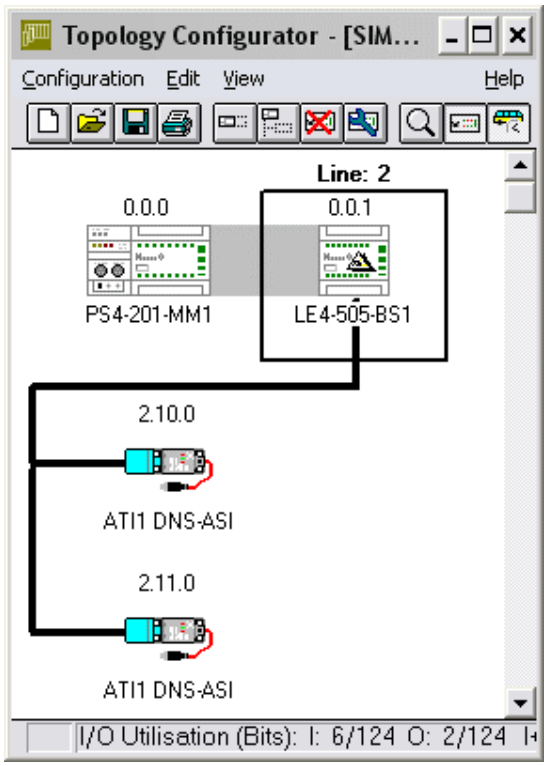


Rys.3.26. Przykładowa magistrala AS-i z dwoma czujnikami indukcyjnymi AS-i



Rys.3.27. Magistrala AS-i z dwoma czujnikami o profilu 1.1

Na rysunku 3.28 przedstawiono utworzoną w oprogramowaniu Sucosoft S40 topologię układu dołączając do niej czujniki indukcyjne konkretnego typu ATI1 DNS-ASI oraz deklarując ich adresy 10 i 11 (w adresach wyświetlonych w topologii 2.10.0 i 2.11.0 druga cyfra odnosi się do adresu slave'a). Znajomość typu czujnika jest oczywiście równoznaczna ze znajomością profilu tego czujnika.



Rys.3.28. Magistrala AS-i z dwoma czujnikami indukcyjnymi ATI1 DNS-ASI

Widoczny w topologii czujnik indukcyjny ATI1 DNS-ASI oraz jego parametry przedstawiono na rysunku 3.29.

Z czterobitowego pola danych przesyłanego w ramce protokołu AS-i dla w przypadku tego czujnika wykorzystano trzy bity danych:

- bit danych D0 – w tym bicie przesyłany jest sygnał zadziałania czujnika (w zależności od tego jak czujnik został sparametryzowany);
- bit danych D1 – w tym bicie przesyłany jest sygnał o niepewnej pracy czujnika (np. takie zamontowanie czujnika, że wykrywany element metalowy przemieszcza się przed czołem czujnika w odległości bardzo różniącej się od znamionowej strefy działania);
- bit danych D2 – w tym bicie przesyłany jest sygnał gotowości do pracy (m. in. poprawna wartość napięcia zasilania czujnika).

Z czterobitowego pola danych, w którym w czasie ostatniego telegramu w cyklu komunikacji przesyłane są dane w celu parametryzacji urządzenia slave, wykorzystano tylko jeden bit P1. Za pomocą tego bitu programuje się sposób działania czujnika, tzn. czy sygnał zadziałania jest wysyłany w przypadku braku (styk NC w czujniku tradycyjnym) czy też obecności (styk NO w czujniku tradycyjnym) metalowego elementu przed czołem czujnika.

Czujnik ATI1 DNS-ASI

Przyporządkowanie logiczne

bit danych D0: wejście sygnału zadziałania
bit danych D1: wejście niepewna praca
bit danych D2: wejście gotowość do pracy
bit danych D3: wyjście -

bit parametru P0: -
bit parametru P1: przełączanie zwierny / rozwierny
bit parametru P2: -
bit parametru P3: -

Rys.3.29. Czujnik indukcyjny ATI1 DNS-ASI i przyporządkowanie logiczne jego sygnałów

Master AS-i tworzy i przechowuje w pamięci następujące listy urządzeń:

- LPS (*list of configured slaves*) – listę skonfigurowanych urządzeń;
- LES (*list of detected slaves*) – listę rozpoznanych urządzeń;
- LAS (*list of active slaves*) – listę aktywnych urządzeń.

Na podstawie analizy tych list master AS-i może na bieżąco prowadzić diagnostykę sieci lub np. nowemu urządzeniu slave o adresie 0 przyporządkować adres brakującego urządzenia.

Rozbieżność w liczbie urządzeń slave na tych listach świadczy o błędach – np. o uszkodzeniu urządzeń slave. Przykładowo, jeśli na liście aktywnych urządzeń zabraknie czujnika, który jest na liście skonfigurowanych urządzeń i wcześniej był na liście rozpoznanych urządzeń, świadczy to o jego uszkodzeniu. W takim przypadku postępowanie w celu usunięcia awarii w układzie sterowania jest bardzo proste. Wystarczy odłączyć uszkodzony czujnik i podłączyć w jego miejsce analogiczny o takim samym profilu (z fabrycznie ustawionym adresem 0, przy czym może to być również czujnik innego

producenta). Po podłączeniu czujnika w najbliższym cyklu komunikacji w przedostatnim telegramie zostanie wykryta jego obecność a w ostatnim telegramie zostanie mu przydzielony brakujący adres z listy aktywnych urządzeń slave.

W przypadku równoczesnego uszkodzenia dwóch lub więcej czujników takie postępowanie nie jest możliwe, gdyż master nie będzie wiedział jaki adres z brakujących trzeba nadać nowo dołączonemu czujnikowi. Przed wymianą czujnika należy nadać mu więc odpowiedni adres – np. za pomocą programatora ręcznego.

Oprócz wszystkich wymienionych wcześniej zalet magistrali czujników i urządzeń wykonawczych AS-i na podkreślenie zasługują jeszcze dwie kolejne zalety: bardzo wysoka niezawodność przesyłania informacji po magistrali AS-i, oraz sprowadzenie możliwości diagnostyki systemu sterowania na najniższy w hierarchii sterowania poziom – czujników i urządzeń wykonawczych. Wysoką niezawodność przesyłania informacji osiągnięto dzięki zastosowaniu metody kodowania typu Manchester (*APM – Alternating Pulse Modulation*). Master AS-i powtarza nadawanie ramek w których wykryto błędy. Jeśli w ramce wystąpią nie więcej niż trzy błędy, to takie sytuacje są zawsze wykrywane, a w przypadku czterech lub pięciu błędów są one wykrywane z prawdopodobieństwem większym niż 99,9%. Dzięki temu określony czas pomiędzy dwoma przesłanymi i nie wykrytymi błędnymi ramkami wynosi ponad 8 lat [53].

Na rysunku 3.30 przedstawiono skrótowe podsumowanie charakterystyki magistrali czujników i urządzeń wykonawczych AS-i.

Struktura sieci → dowolna
Metoda dostępu do łącza sieciowego → master – slave
Medium transmisji → profilowy lub okrągły przewód dwużyłowy
Max długość sieci → 100 m (300 m – z dwoma wzmacniaczami)
Szybkość transmisji → 167 kb/s
Napięcie zasilania → 29,5 – 31,6 V
Jedno urządzenie nadrzędne (master) w sieci
Max liczba urządzeń podporządkowanych (slave) → 31
Max 124 bity danych (po 4 bity dla każdego urządzenia slave)
Specjalny układ scalony AS-i jest identyfikowany przez trzy parametry:
- **adres urządzenia (slave address);**
- **kod we/wy (I/O code) określający co przesyłane jest w bitach danych;**
- **kod identyfikacyjny (ID code) dla rozróżnienia, gdy taki sam jest I/O code.**

Dwa ostatnie parametry są nazywane profilem (np. 1.1 dla czujników).

Rys.3.30. Charakterystyka systemu AS-i – podsumowanie

3.7. Ethernet przemysłowy

Sieć Ethernet jest powszechnie wykorzystywana na trzecim poziomie sterowania w hierarchii sterowania w przedsiębiorstwach przemysłowych – czyli na poziomie sterowania nadrzędnego. Na tym poziomie sterowania nie jest bezwzględnie wymagany determinizm czasowy przy przesyłaniu informacji w sieci, a warunki środowiskowe są zbliżone do tzw. warunków biurowych.

Ze względu na szerokie rozpowszechnienie sieci Ethernet, duże szybkości transmisji, szeroką dostępność urządzeń i elementów do budowy tych sieci (przy niewygórowanych, i stale malejących cenach komponentów służących do budowy sieci Ethernet) w drugiej połowie lat dziewięćdziesiątych ubiegłego wieku podjęto zakończone powodzeniem próby wykorzystania sieci Ethernet na drugim, a nawet pierwszym poziomie sterowania.

Aby zastosować sieć Ethernet na drugim poziomie sterowania – czyli na poziomie na którym wykorzystywane są sieci miejscowe – należało spełnić dwa podstawowe warunki:

- dostosować urządzenia i elementy do budowy sieci Ethernet do warunków przemysłowych;
- zapewnić quasi-determinizm czasowy dla danych przesyłanych w sieci Ethernet.

Sieci Ethernet wykorzystywane poniżej trzeciego poziomu sterowania w odniesieniu do których zostały spełnione powyższe warunki zostały nazwane sieciami Ethernetu przemysłowego (*ang. Industrial Ethernet*).

Spełnienie pierwszego warunku wiązało się z zastosowaniem takich rozwiązań technicznych które byłyby dostosowane do zastosowań przemysłowych, ale nie stanowiło to istotnego problemu. W środowisku przemysłowym należy liczyć się z występowaniem takich niekorzystnych czynników jak: zapylenie, zabrudzenie, wysoka wilgotność, szeroki zakres zmian temperatury otoczenia, wibracje i wstrząsy, zakłócenia elektromagnetyczne. Czynniki te nakładają wysokie wymagania na sprzęt i osprzęt wykorzystywany do budowy sieci Ethernetu przemysłowego. Wysoką niezawodność działania sieci Ethernetu przemysłowego można osiągnąć tylko wówczas, jeśli w stosunku do wszystkich komponentów sieci spełnione są odpowiednie wymagania mechaniczne, klimatyczne, dotyczące stopnia ochrony (najlepiej IP67) oraz odporności na zakłócenia elektromagnetyczne. Obecnie sprzęt i zunifikowany osprzęt do budowy sieci Ethernetu przemysłowego jest powszechnie dostępny.

Spełnienie drugiego warunku wiązało się z opracowaniem nowych rozwiązań programowych, a w niewielkim tylko stopniu z opracowaniem nowych rozwiązań

sprzętowych. Drugi warunek wynika z faktu, że systemy sterowania pracujące na drugim poziomie są systemami czasu rzeczywistego, a nie jest możliwa praca systemu sterowania w czasie rzeczywistym, jeśli nie można przewidzieć czasu wymaganego na aktualizację danych przesyłanych w sieci łączącej poszczególne elementy tego systemu.

Sieć Ethernet wykorzystuje stochastyczną metodę dostępu do łącza komunikacyjnego CSMA/CD. Sieć z taką metodą dostępu nie może zastąpić sieci miejscowych, które wykorzystują deterministyczną metodę dostępu do łącza komunikacyjnego. Jeśli jednak założyć, że dla większości aplikacji przemysłowych wystarczające jest, gdy maksymalny czas aktualizacji danych przesyłanych w sieci nie przekracza 5-10 ms, wówczas możliwa jest budowa takiej sieci na bazie sieci Ethernet, która będzie Ethernetem przemysłowym. Jest to przypadek takiej kategorii czasu rzeczywistego, gdzie określone zdarzenia muszą zaistnieć w określonym czasie (nie później niż). Może się to odnosić zarówno do przesyłania danych, jak i do wyprowadzenia zaktualizowanego sygnału sterowania. Druga kategoria czasu rzeczywistego odnosi się do zdarzeń, które muszą zaistnieć dokładnie w określonej chwili (z zadaną fluktuacją). W systemach sterowania ze sterownikami programowalnymi częściej rozpatruje się pierwszą kategorię. Druga odnosi się przede wszystkim do systemów sterowania ruchem (*ang. motion control*).

Wymagania wobec sieci Ethernet, aby była ona siecią Ethernetu przemysłowego, dotyczą jej budowy oraz wykorzystania specjalnych protokołów komunikacyjnych.

W sieciach Ethernetu przemysłowego wykorzystywane są następujące topologie sieci: gwiazda, drzewo, linia oraz pierścień (często redundantny). Biorąc pod uwagę, że linia tak naprawdę stanowi część pierścienia, można stwierdzić, że to, co odróżnia Ethernet przemysłowy od typowych sieci miejscowych, jest częste wykorzystywanie topologii gwiazdy. Wybór topologii zależy od wymagań poszczególnych elementów sieci (stacji) jak i obiektu sterowania (maszyna, linia produkcyjna, proces technologiczny).

W strukturze gwiazdy centralnym elementem jest przełącznik (switch) od którego prowadzone są połączenia do pozostałych elementów sieci. Struktura gwiazdy jest wykorzystywana wówczas, gdy duża jest liczba elementów do połączenia przy niewielkich odległościach pomiędzy nimi; np. przy sterowaniu pojedynczą maszyną.

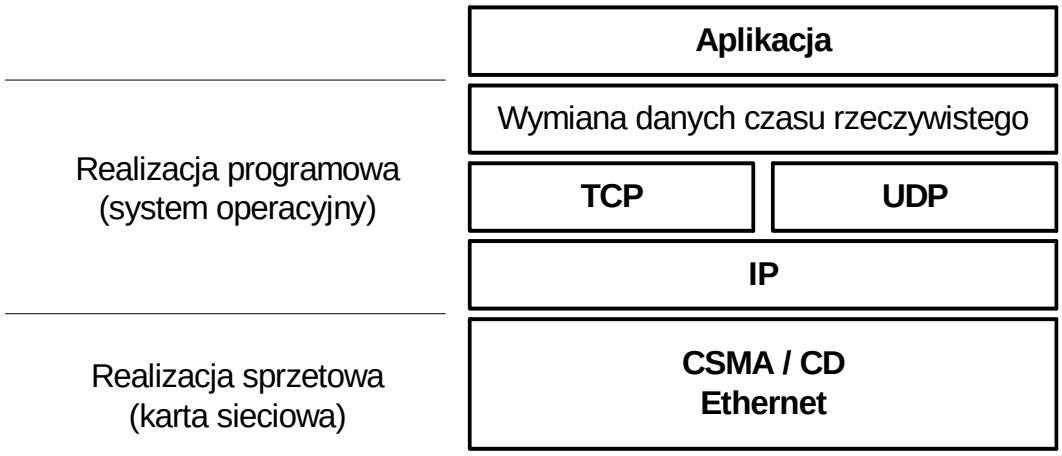
Topologię drzewiastą stanowią połączone ze sobą struktury gwiazdy. Jest wykorzystywana przy dużej liczbie elementów, gdy poszczególne grupy elementów są od siebie oddalone; np. przy sterowaniu linią produkcyjną.

Topologia linii powstaje poprzez połączenie ze sobą przełączników (switch). Jest stosowana w rozległych systemach sterowania; np. długie linie produkcyjne, oddalone gniazda produkcyjne lub rozległy proces technologiczny.

Struktura pierścieniowa powstaje poprzez połączenie końców w strukturze linii. Jest wykorzystywana w podobnych sytuacjach jak linia, ale przy zwiększonych wymaganiach niezawodności przesyłania danych.

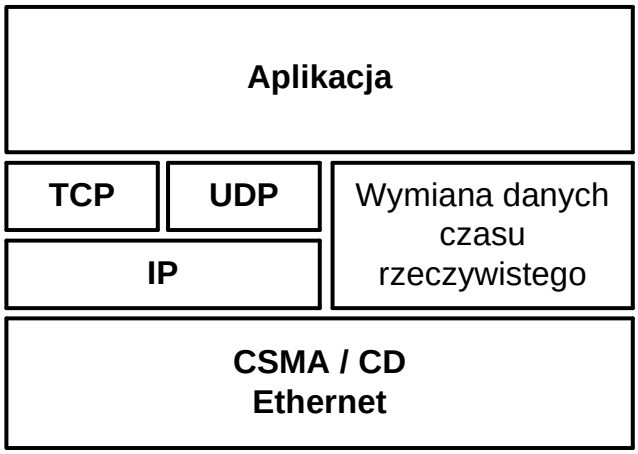
Połączenia w sieciach Ethernetu przemysłowego mogą być wykonywane zarówno kablami jak i łączami światłowodowymi. Stosuje się również kable hybrydowe, posiadające przewody do komunikacji (włókna optyczne) i przewody do doprowadzenia zasilania.

W sieciach Ethernetu przemysłowego wykorzystuje się różne protokoły komunikacyjne, np.: Modbus/TCP, ETHERNET Powerlink, Profinet, Ethernet/IP, EtherCAT, SERCOS III. Nie ma w tych protokołach jednego standardu organizacji nawiązywania połączenia i przesyłania danych w sieci [7,24]. W różny sposób wykorzystują one funkcje poszczególnych warstw wzorcowego modelu ISO/OSI łączenia systemów otwartych. Wyróżnić można trzy architektury warstw modelu pokazane na rysunkach 3.31, 3.32 i 3.33.



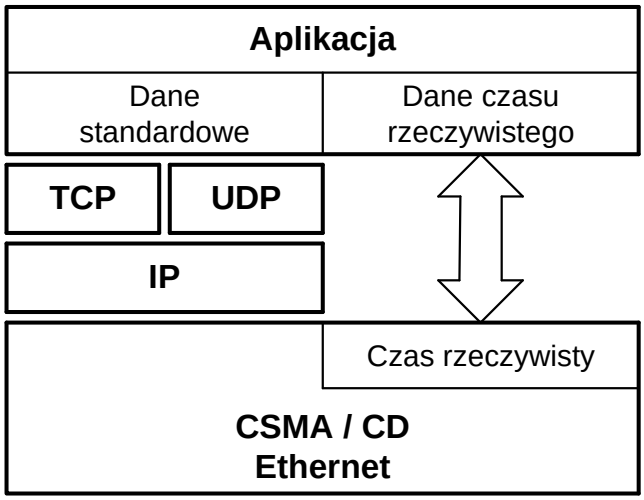
Rys. 3.31. Architektura dla protokołów: Modbus/TCP; Ethernet/IP

W architekturze pokazanej na rysunku 3.31 zarówno przesyłanie danych standardowych jak i danych czasu rzeczywistego jest realizowane z wykorzystaniem 3 i 4 warstwy modelu i standardowego stosu dla protokołów TCP/IP i UDP/IP. Przykładowo, protokół Modus/TCP oprócz wykorzystywania metody dostępu do sieci Ethernet typu master-slave wykorzystuje jeszcze specjalny protokół RTPS (ang. Real-Time Publish Subscribe) korzystający ze stosu UDP/IP w celu synchronizacji pracy aplikacji rozproszonych.



Rys.3.32. Architektura dla protokołów: ETHERNET Powerlink, Profinet Ver.2

W architekturze pokazanej na rysunku 3.32 dane czasu rzeczywistego są przesyłane z pominięciem 3 i 4 warstwy modelu i nie wykorzystują standardowego stosu dla protokołów TCP/IP i UDP/IP. Dane czasu rzeczywistego są przesyłane w podobny sposób jak to się dzieje przy wykorzystywaniu protokołów typowych sieci miejscowych. W sieci wydzielony jest węzeł zarządzający sieci i w każdym cyklu komunikacji w wydzielonym czasie tzw. izochronicznym (równookresowym) są przesyłane dane czasu rzeczywistego; w pozostałym czasie są przesyłane tzw. dane niekrytyczne, które nie wymagają determinizmu czasowego.



Rys.3.33. Architektura dla protokołów: Profinet Ver.3, SERCOS III, EtherCAT

W architekturze pokazanej na rysunku 3.33 dane czasu rzeczywistego są również przesyłane z pominięciem 3 i 4 warstwy modelu i nie wykorzystują standardowego stosu dla protokołów TCP/IP i UDP/IP. W tej architekturze dane czasu rzeczywistego też są przesyłane

w podobny sposób jak to się dzieje przy wykorzystywaniu protokołów typowych sieci miejscowych, z taką jednak różnicą, że ich przesyłanie odbywa się poprzez sprzętowy kanał czasu rzeczywistego. Wykorzystywana karta sieciowa, w odróżnieniu od dwóch poprzednich architektur, nie jest typową kartą sieciową. Tak jak w przypadku poprzedniej architektury przesyłane dane są podzielone na dane czasu rzeczywistego i pozostałe dane niekrytyczne, z tą różnicą, że kanał przesyłania danych czasu rzeczywistego jest zrealizowany w sposób sprzętowy.

Sposoby stosowane w sieciach Ethernetu przemysłowego, aby spełnić wymagania systemu komunikacyjnego pracującego w czasie rzeczywistym, są następujące:

- przydzielenie odpowiedniego priorytetu do przesyłanej informacji (wyższy priorytet dla informacji sterującej);
- zwiększenie szybkości przesyłania danych (10Mb/s, 100Mb/s, 1Gb/s);
- odpowiedni podział sieci na segmenty (segmenty spełniające wymagania komunikacji w czasie rzeczywistym);
- ustawienie różnych szybkości transmisji dla segmentów czasu rzeczywistego i dla informacji przesyłanej w sieci pomiędzy tymi segmentami;
- odpowiednie wykorzystanie protokołów TCP (polecenia, zdarzenia i alarmy przesyłane w chwili zaistnienia; z potwierdzeniem) i UDP (dane przesyłane okresowo, bez potwierdzenia);
- wykorzystanie (dla niektórych protokołów) niestandardowych kart sieciowych, dla których kanał przesyłania danych czasu rzeczywistego realizowany jest w sposób sprzętowy.

Obecnie Ethernet przemysłowy jest coraz częściej stosowany w systemach sterowania ze sterownikami programowalnymi, jest wykorzystywany jako sieć miejscowa, ale nic nie wskazuje na to, aby w najbliższym czasie miał wyprzeć i całkowicie zastąpić pracujące na poziomie drugim sterowania tradycyjne sieci miejscowe.

Podsumowanie

W części 2 opracowanych materiałów pomocniczych w uporządkowany sposób przedstawiono dwa ważne zagadnienia dotyczące tworzenia oprogramowania na sterowniki programowalne, a mianowicie: strukturyzację oprogramowania oraz opracowywanie, przedstawianie i realizację algorytmów sterowania sekwencyjnego. W dalszej części materiałów przedstawiono wybrane zagadnienia dotyczące pracy sterowników programowalnych w sieci, gdyż obecnie coraz rzadziej w układach sterowania pracują pojedyncze sterowniki. Coraz częściej w systemach sterowania pracuje więcej sterowników programowalnych połączonych między sobą, a także z innymi urządzeniami automatyki, siecią (lub sieciami) komunikacji szeregowej.

Czynne uczestnictwo studentów w zajęciach z przedmiotu oraz zrozumienie i dobre opanowanie przez nich zagadnień omówionych w obu częściach tych materiałów powinno sprawić, że zostaną osiągnięte zamierzone efekty kształcenia, oraz powinno przygotować studentów do dalszego – również samodzielnego – uzupełniania wiedzy dotyczącej sterowników programowalnych i ich zastosowań w układach i systemach sterowania.

Literatura

1. Automation Systems and Drives. Main Catalogue 2001/2002. Moeller GmbH, Bonn, 01/2001, HPL0213-2001/2002GB.
2. Automation systems. Main Catalogue 2004/2005. Moeller GmbH, Bonn, HPL0213-2004/2005GB-INT MM/We 10/04.
3. Bauernfeind D., Dung O., Skupin J.: CM4-505-GS1 AS-Interface Master: Gateway for Suconet K – Actuator Sensor Interface. Moeller GmbH, Bonn, AWB 27-1271-GB, 09/96.
4. Berger H.: Automatyzacja za pomocą sterownika SIMATIC S5-115U. SIEMENS AG, Erlangen, 1987.
5. Brock S., Muszyński R., Urbański K., Zawirski K.: Sterowniki Programowalne. Wydawnictwo Politechniki Poznańskiej, Poznań, 2000.
6. Broel-Plater B.: Układy wykorzystujące sterowniki PLC. Projektowanie algorytmów sterowania. Wydawnictwa Naukowe PWN, Warszawa, 2010.
7. Cena G., Valenzano A., Vitturi S.: Hybrid Wired/Wireless Networks for Real-Time Communications. IEEE Industrial Electronics, Vol.2, Nr 1, 2008, pp. 8-20.
8. DL405 User Manual, 2nd Edition. Direct Logic Koyo, PLC *Direct*™ Inc., 1995.
9. Dirnfeldner M.: Automation with Programmable Controllers. An Introduction for Beginners. LS 27-064 GB, Klockner-Moeller 12/88.
10. Flaga S.: Programowanie sterowników PLC w języku drabinkowym. Wydawnictwo BTC, Legionowo, 2010.
11. Gomółka W.: Realizacja algorytmów sterowania sekwencyjnego za pomocą sieci działań GRAFCET. PAK,1990, Nr 7, s. 136-141.
12. Hitachi Programmable Controller - E Series. Operation Manual, No. NJI 008A.
13. Kamiński K.: Programowanie sterownika S7. Norkom, Gdańsk, 2000.
14. Kamiński K.: Programowanie w Step 7 MicroWin. Wydawnictwo GRYF, Gdańsk, 2006.
15. Kamiński K.: Podstawy sterowania z PLC. Wydawnictwo GRYF, Gdańsk, 2009.
16. Kasprzyk J.: Programowanie sterowników przemysłowych. WNT, Warszawa, 2006.
17. Kiełtyka L., Stoiński D., Fengler W.F.: Wykorzystywanie sieci Petriego do sterowania procesami sekwencyjnymi. Prace IX Sympozjum SPD-9 „Symulacja procesów dynamicznych", Polana Chochołowska, 10-14 czerwiec 1996, s. 125-129.
18. Konen P.L., Leuchs K.: Networking Programmable Controllers. An Introduction for Beginners. G 27-2100-GB, Klockner-Moeller 5/90.

19. Król A., Moczko-Król J.: S5/S7 Windows. Programowanie i symulacja sterowników PLC firmy SIEMENS. Wydawnictwo Nakom, Poznań, 2000.
20. Kwaśniewski J.: Programowalne sterowniki przemysłowe w systemach sterowania. J.Kwaśniewski & Fundacja Dobrej Książki, Kraków, 1999.
21. Kwaśniewski J.: Sterowniki PLC w praktyce inżynierskiej. Wydawnictwo BTC, Legionowo, 2008.
22. Kwaśniewski J.: Programowalny sterownik SIMATIC S7-300 w praktyce inżynierskiej. Wydawnictwo BTC, Legionowo, 2009.
23. Legierski T., Kasprzyk J., Hajda J., Wyrwał J.: Programowanie sterowników PLC. Wydawnictwo Pracowni Komputerowej Jacka Skalmierskiego, Gliwice, 1998.
24. Lüder A., Lorentz K.: IAONA Handbook – Industrial Ethernet. IAONA e.V., Magdeburg, 2005.
25. Maczyński A.: Sterowniki programowalne PLC. Budowa systemu i podstawy programowania. Astor Sp. z o.o., Kraków, 2002.
26. Menden R., Petrick B.: Manual. Structuring of User Programs. Klockner-Moeller, Bonn 5/91, AWB 27-1011-GB.
27. Mikulczyński T., Samsonowicz Z.: Synteza sekwencyjnych układów sterowania zautomatyzowanych procesów technologicznych. PAK, Nr 7, 1994.
28. Mikulczyński T., Samsonowicz Z.: O syntezie sekwencyjnych układów sterowania ZPT. PAK, Nr 10, 1994.
29. Mikulczyński T., Samsonowicz Z.: Analiza wybranych metod modelowania i programowania dyskretnych procesów produkcyjnych. PAK, Nr 3, 1995.
30. Mikulczyński T., Samsonowicz Z.: Automatyzacja dyskretnych procesów produkcyjnych. WNT, Warszawa, 1997.
31. Mosoń I., Karkosiński D.: Programowalne sterowniki logiczne w dydaktyce. „Zastosowanie komputerów w dydaktyce ’94”, Cykl seminariów zorganizowanych przez PTETiS Oddział w Gdańsku, Zesz. Nauk. Wydz. Elektrycznego Politechniki Gdańskiej Nr 6, 1994, s.89-94.
32. Mosoń I.: Sterowniki programowalne PS4 w zdecentralizowanych systemach sterowania. Elektroinstalator (Automatyka) 1996, Nr 11, s.27-30.
33. Mosoń I., Karkosiński D.: Program sterowania, diagnozowania stanu technicznego i monitorowania parametrów pracy linii technologicznej okleinowania płyt meblowych. Opis algorytmu. 1998, 15 s. 2 rys. 3 tabl. bibliogr. 6 poz. maszyn.

34. Mosoń I.: Język GRAFCET a weryfikacja algorytmów i programów sterowania. Materiały: Seminarium Naukowo-Techniczne „Zastosowanie nowoczesnej aparatury Schneider Electric w urządzeniach i instalacjach elektroenergetycznych niskiego napięcia", Gdańsk, 9 czerwiec 1999, s. 4-14.
35. Mosoń I.: Wybrane aspekty wprowadzenia do dydaktyki przedmiotu Sterowniki Programowalne. „Zastosowanie komputerów w dydaktyce '99", IX cykl seminariów zorganizowanych przez PTETiS Oddział w Gdańsku, Zesz. Nauk. Wydz. Elektrotechniki i Automatyki Politechniki Gdańskiej Nr13, 1999, s.101-106.
36. Mosoń I.: Sterowniki programowalne – strukturyzacja programów sterowania. „Zastosowanie komputerów w nauce i technice 2001", XI cykl seminariów zorganizowanych przez PTETiS Oddział w Gdańsku, Zesz. Nauk. Wydz. Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 17, 2001, s.145-152.
37. Mosoń I., Żukowski K.: Symulator Sym-PS4 sterownika programowalnego PS4-201-MM1. „Zastosowanie komputerów w nauce i technice 2002", XII cykl seminariów zorganizowanych przez PTETiS Oddział w Gdańsku, Zesz. Nauk. Wydz. Elektrotechniki i Automatyki Politechniki Gdańskiej Nr 18, 2002, s.125-130.
38. Mosoń I.: Control systems with programmable controllers – teaching aspects. IX Konferencja Naukowo-Techniczna „Zastosowania Komputerów w Elektrotechnice" ZKwE'2004, Poznań/Kiekrz, 19-21 kwietnia 2004, s. 621-624.
39. Orłowski H.: Komputerowe układy automatyki. WNT, Warszawa, 1987.
40. PN-EN 61131-1:2004. Sterowniki programowalne – Część 1: Postanowienia ogólne.
41. PN-EN 61131-3:2004. Sterowniki programowalne – Część 3: Języki programowania.
42. Practical Aspects of Industrial Control Technology. Telemecanique Technical Collection. Editions CITEF, 1994.
43. Roersch P.: LE4-505-BS1 AS-I Network Module. Hardware and Engineering. Moeller GmbH, Bonn, AWB 27-1314 GB, 02/98.
44. Ruda A., Olesiński R.: Sterowniki Programowalne PLC. Wydawnictwo COSiW SEP, Warszawa, 2003.
45. Sałat R., Korpysz K., Obstawski P.: Wstęp do programowania sterowników PLC. Wydawnictwo Komunikacji i Łączności, Warszawa, 2010.
46. Schunemann U.: Programming PLCs with an Object-Oriented Approach. ATP International, Automation Technology In Practice. Nr 2, 2007, pp. 59-63.
47. Seta Z.: Wprowadzenie do zagadnień sterowania – wykorzystanie programowalnych sterowników logicznych PLC. Wydawnictwo NIKOM, Warszawa, 2002.

48. Simatic S5-115U Programmable Controller. Manual. Siemens AG, 1991, EWA 4NEB 811 6130-02a.
49. Stańczak W.: Przemysłowe sieci lokalne. Sieciowe systemy komunikacyjne integrujące automatyzację wytwarzania. Podręczniki Nr 6. PIAP, Warszawa, 1998.
50. Sterownik VersaMax. Podręcznik użytkownika. Astor Sp. z o.o., Kraków, 1999.
51. Tassin A.: Structured Programming and Utilities for the Series PCD. User's Guide. SAIA AG, 1992, Edition 26/732 E4 06.94.
52. Trybus L.: Regulatory wielofunkcyjne. WNT, Warszawa, 1992.
53. Volberg J.: Actuator-Sensor Interface. Networking I/O on the fieldbus. Klockner-Moeller, Bonn, 1996.