Wstęp

W ostatnich latach daje się zauważyć bardzo szybki rozwój dziedzin automatyki ściśle związanych z zastosowaniem sterowników PLC (ang. Programmable Logic Controllers – Programowalne Sterowniki Logiczne). Przyczyną tego jest łatwość implementacji algorytmów sterowania automatycznego w oparciu o sterowniki PLC, ich niezawodność sprzętowa, modułowa budowa umożliwiająca łatwą skalowalność oraz łatwość serwisowania.

Ponieważ firma GE Fanuc nie oferuje oprogramowania symulującego pracę swoich sterowników, co znacznie utrudnia testowanie poprawności zaimplementowanego dla nich oprogramowania, postanowiliśmy stworzyć takie oprogramowanie, dzięki któremu takie testowanie będzie możliwe. Dodatkowo, dzięki symulatorowi będzie możliwa nauka programowanie sterowników bez potrzeby ich posiadania.

Oprogramowanie prezentowane w niniejszej pracy łączy ze sobą cechy edytora, symulatora i kompilatora, co sprawia, że jest w pełni funkcjonalnym i wygodnym w użyciu narzędziem. Pomimo, iż w internecie dostępne są programy tego typu pisane również jako prace dyplomowe, to posiadają one następujące wady:

- zbyt mała liczba dostępnych zmiennych
- słaba funkcjonalność edytora programu
- brak możliwości współpracy z programem LM 90 (komercyjne oprogramowanie narzędziowe służące do programowanie sterowników GE Fanuc)
- ograniczone możliwości symulacji
- brak możliwości wyboru typu symulowanego sterownika
- nieergonomiczne wykonanie interfejsu użytkownika

1. Rys historyczny

Historia programowanych sterowników logicznych sięga 1968 roku, kiedy to firma Gemeral Motors rozpoczęła prace nad budową sterowników przyjmując następujące założenia:

- Łatwość programowania i przeprogramowywania stosownie do zmieniających się warunków przemysłowych.
- Łatwość w utrzymaniu w ruchu produkcyjnym z możliwością napraw przez wymianę pojedynczych modułów.
- Większa niezawodność w warunkach przemysłowych przy mniejszych wymiarach niż urządzenia przekaźnikowe
- Koszty porównywalne z urządzeniami dotychczas stosowanymi.

Sterowniki programowalne, które rozwinęły się na początku lat 70 – tych stały się integralną częścią systemów sterowania automatycznego. Pomimo, iż początkowo ich celem było zastąpienie układów przekaźnikowych zostały one od razu zaakceptowane w przemyśle samochodowym, a później w pozostałych gałęziach przemysłu.

W latach 70-tych sterowniki wyposażono w moduły komunikacyjne oraz koprocesory, które umożliwiły przyśpieszenie wykonywania operacji obliczeniowych.

Już w latach 80-tych układy przekaźnikowe, regulatory analogowe, a nawet mikrokomputery zostały prawie całkowicie wyparte przez sterowniki PLC.

Sterowniki PLC rozwinęły się głównie dlatego, ponieważ umożliwiają szybkie reagowanie na zmiany wymagań aplikacyjnych poprzez łatwiejszą możliwość przeprogramowania bez potrzeby zmian sprzętowych.

Duże znaczenie w rozwoju PLC odegrały również następujące fakty:

- podobieństwo schematów drabinkowych używanych w programowaniu PLC do stosowanych schematów stykowo – przekaźnikowych
- zwiększenie niezawodności komputerów przemysłowych

- możliwość kontrolowania poprawności obwodów wejściowych i wyjściowych
- posiadanie specjalnego zbioru instrukcji uwzględniające warunki przemysłowe
- możliwość komunikacji operatorów procesu z systemami sterowania poprzez urządzenia typu HMI (Human Machine Interface).

Duże znaczenie dla rozwoju PLC miało pojawienie się w latach 90 – tych systemów do sterowania nadrzędnego i archiwizacji danych SCADA (ang. Supervisory Control and Data Aquisition). Systemy te dopełniają i rozszerzają możliwości sterowników poprzez następujące funkcje:

- zbieranie, przetwarzanie oraz archiwizację danych
- opracowywanie raportów dotyczących bieżącego stanu procesu przemysłowego
- wizualizacja zmiennych procesowych aktualnych i historycznych
- generowanie sygnałów alarmowych przy określonych warunkach
- wypracowywanie danych dla warstw operatywnego sterowania produkcją i warstw zarządzania

W chwili obecnej sterowniki PLC znajdują zastosowanie praktycznie wszędzie tam, gdzie istnieje konieczność sterowania wykonaniem jakiejś czynności. Przez ich rozpowszechnienie i zwiększającą się konkurencję na rynku stają się one coraz tańsze, a przez co bardziej dostępne. Umożliwiając one również zabezpieczenie najważniejszych procesów przez redundancję sterowników PLC realizujących sterowanie tymi procesami.

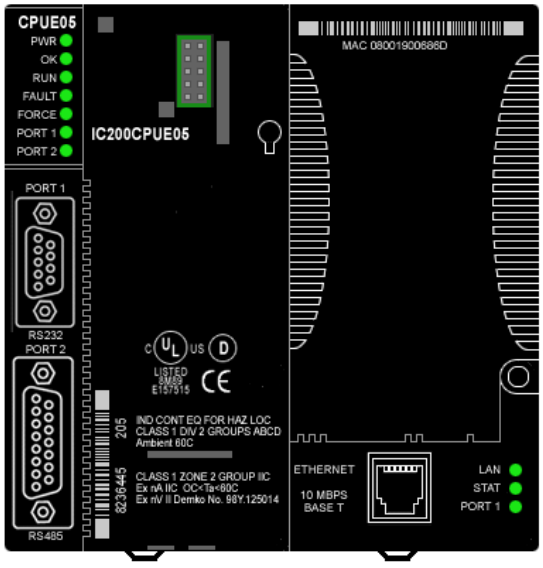
Coraz większe rozpowszechnienie sterowników PLC wymusiło ich standaryzację. W 1993 roku została wydana przez Międzynarodową Komisję Elektroniki norma IEC 1131 „Programmable Controllers”, która łączy w sobie doświadczenia wielu producentów i użytkowników sterowników PLC. Norma ta standaryzuje sprzęt i określa wymagania testowe. Definiuje ona również podstawowe pojęcia związane z językiem programowania oraz standaryzację w zakresie wymiany informacji.

2. Budowa i zasada działania sterowników PLC.

Jedną z zalet sterowników PLC jest ich modułowa budowa. Pozwala to na dołączanie określonych modułów rozszerzających w zależności od realizowanego zadania. Do zmontowania sterownika niezbędna jest płyta łączeniowa (kaseta) która wyposażona jest w gniazda służące do połączenia wybranych modułów (pełni rolę magistrali wymiany danych pomiędzy modułami wejść/wyjść a jednostką centralną). Do działania sterownika potrzebne są dwa podstawowe moduły: jednostka centralna oraz zasilacz. Poprzez dołączenie dodatkowych modułów zwiększa się jego funkcjonalność tak, aby umożliwić realizację zadań sterowania. Do modułów dodatkowych zaliczamy m.in.:

- moduły wejść, wyjść dwustanowych DI, DO (Discrete Input, Discrete Output)
- moduły wejść, wyjść analogowych AI, AO (Analog Input, Analog Output)
- moduły szybkich liczników HSC (ang. High Speed Counter)
- moduły pozycjonowania osi
- moduły komunikacyjne CMM
- moduły komunikacyjne (Genius, Ethernet, Profibus DP i wiele innych)
- moduły programowalnych koprocesorów arytmetyczno/komunikacyjnych PCM

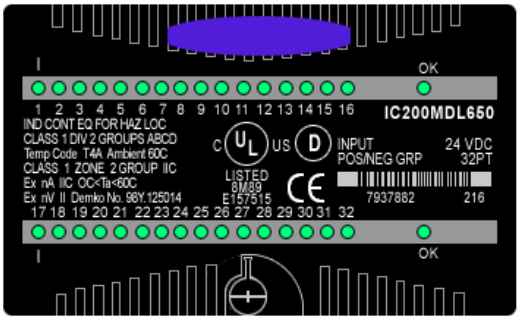
Moduł jednostki centralnej CPU – jest jednym z dwóch niezbędnych modułów



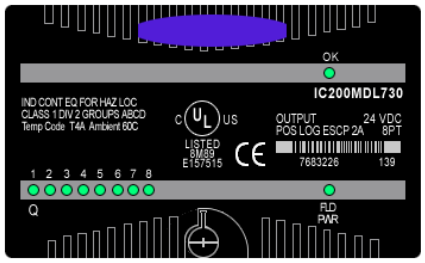
sterownika, umożliwia on wykonanie programu użytkownika, zapewnia komunikację między modułami, koordynuje jego pracę, odpowiada za transmisję programu do pamięci sterownika przez protokół SNP. Odpowiada za bezpieczeństwo danych w pamięci RAM dzięki zapewnieniu zasilania z baterii podczas awarii. Umożliwia również zapisanie konfiguracji sterownika i zabezpieczenie go kluczem programowym. Jednostki centralne różnią się częstotliwością zegara, pojemnością pamięci rejestrów i

pamięci przeznaczonej na program użytkownika. Dodatkowo niektóre modele umożliwiają wykonywanie operacji zmiennoprzecinkowych. Parametrem charakteryzującym szybkość działania poszczególnych jednostek CPU jest czas wykonania 1000 instrukcji bitowych lub czas wykonania 1000 instrukcji mieszanych.

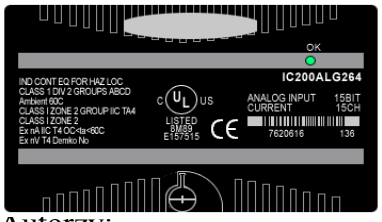
Moduły wejść dyskretnych DI (Discrete Input) – są to moduły wejść dwustanowych oraz trójstanowych (bardzo rzadko stosowane). Moduły te umożliwiają zamianę sygnału prądu stałego lub zmiennego pochodzące z urządzeń zewnętrznych na sygnały akceptowane przez sterownik. Zazwyczaj zbudowane są one z wykorzystaniem przetwornika optycznego, który jednocześnie zapewnia izolację między obwodami. Polaryzacja źródła zasilania zależy od typu modułu, ale coraz częściej stosuje się moduły niewrażliwe na polaryzację. Moduły te zapewniają filtrację zakłóceń sygnału wynikających z działania czujników obiektowych.



Moduły wyjść dyskretnych DO – stosowane są do transformacji sygnałów logicznych sterownika na sygnały prądu stałego lub zmiennego potrzebne do wysterowania urządzeń zewnętrznych. Używane są do tego w zależności od potrzeb różne urządzenia elektroniczne (przekaźniki, tyrystory, tranzystory, itp.). Ich styki mogą być normalnie otwarte lub normalnie zamknięte lub przełączane. Najczęstszymi obciążeniami dla modułów wyjść dyskretnych są obciążenia indukcyjne stwarzające niebezpieczeństwo ich urządzenia (cewki, silniki) z tego powodu moduły te są zabezpieczone przed przepięciami. Podczas projektowania systemu należy zwrócić uwagę na dane dotyczące ograniczeń liczby przełączeń przekaźników. Dane te dostępne są w katalogach.



Moduły wejść analogowych AI – umożliwiają one przetwarzanie analogowych sygnałów wejściowych o wartościach ciągłych na sygnały o zakresie wartości akceptowane przez sterownik. Przetworzone sygnały umieszczone są w obszarze danych



oznaczonych jako %AI. Wartości te są proporcjonalne do wartości napięć lub prądów w poszczególnych obwodach modułów. Istnieją dwa podstawowe rodzaje tych modułów:

- z wejściami jednokońcówkowymi
- z wejściami różnicowymi (mniej czułe na zakłócenia)

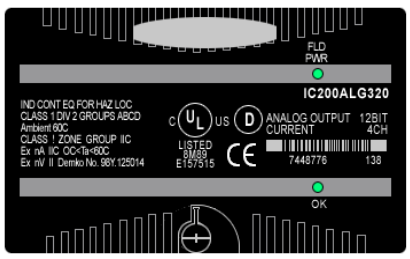
Główne parametry kanałów analogowych:

- rozdzielczość – (błąd kwantyzacji) jest jednakowa w całym przedziale napięcia wejściowego, jego maksymalna wartość określana jest przez bit LSB rejestru przetwornika ADC (analogowo - cyfrowego)
- dokładność – zależy od tolerancji elementów użytych do budowy modułów. Wyznaczana jest przez określenie maksymalnej różnicy pomiędzy wartością oczekiwaną a wartością mierzoną
- tłumienie napięcia wspólnego w kanale CMRR – wyznacza się w dB
- tłumienie zakłóceń międzykanałowych – określa wpływ zmian w poprzednim kanale na kanał badany, wyrażone w dB
- czas uaktualniania

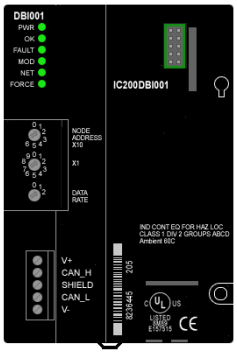
Moduły wejść analogowych dodatkowo posiadają zdolność filtracji sygnałów, które mogłyby je uszkodzić. Dodatkowo stosuje się ekranowanie przewodów użytych w tych obwodach. Rozróżnia się analogowe moduły prądowe i napięciowe.

Sygnały wejściowe prądowe mieszczą się w zakresie 0 – 20 mA lub 4 – 20 mA. Sygnały wejściowe napięciowe mieszczą się w przedziale 0 – 10 V lub -10 – 10V.

Moduły wyjść analogowych – posiadają przetworniki C/A przetwarzające wartości umieszczone w obszarze danych %AQ na wartości wyjściowe (prąd 0-20mA lub napięcie -10 – 10 VDC). Najczęściej zapewniona jest optoizolacja obwodów modułu od magistrali sterownika. Zakresy wyjść prądowych i napięciowych konfigurowane są programowo i mogą przyjmować standardowe wartości (takie jak w modułach wejść analogowych). Po awarii zasilania sterownika wyjścia modułów ustawiane są na ostatnią wartość lub wartość zadeklarowaną w konfiguracji.



Zasilacz – jest to niezbędny moduł sterownika. Główny nacisk kładzie się na jego niezawodność, możliwość pracy w warunkach przemysłowych oraz stabilność napięcia zasilania. Najczęściej zasilane są napięciem 100 – 240VAC lub 10 – 48VDC. Są one źródłem zasilana 5 oraz 24VDC. Dodatkowo każdy zasilacz wyposażony jest we wskaźnik statusu pracy sterownika oraz gniazda RS 485/422. W razie zaniku zasilania zewnętrznego za podtrzymanie danych odpowiedzialna jest bateria litowa o dużej trwałości.



Moduł komunikacyjny (na przykładzie GFanuc) – służy do połączenia sterowników w sieć oraz do zapewnienia funkcji komunikacyjnych i kontrolnych. Tworzą one sieć z wykorzystaniem asynchronicznej transmisji szeregowej między sterownikami oraz urządzeniami HMI. Najbardziej popularny jest protokół Madbus RTU.

Moduł komunikacji szeregowej CCM – udostępnia dwa łącza szeregowe RS 232C i RS 422/485 wraz z zaimplementowanymi protokołami komunikacyjnymi.

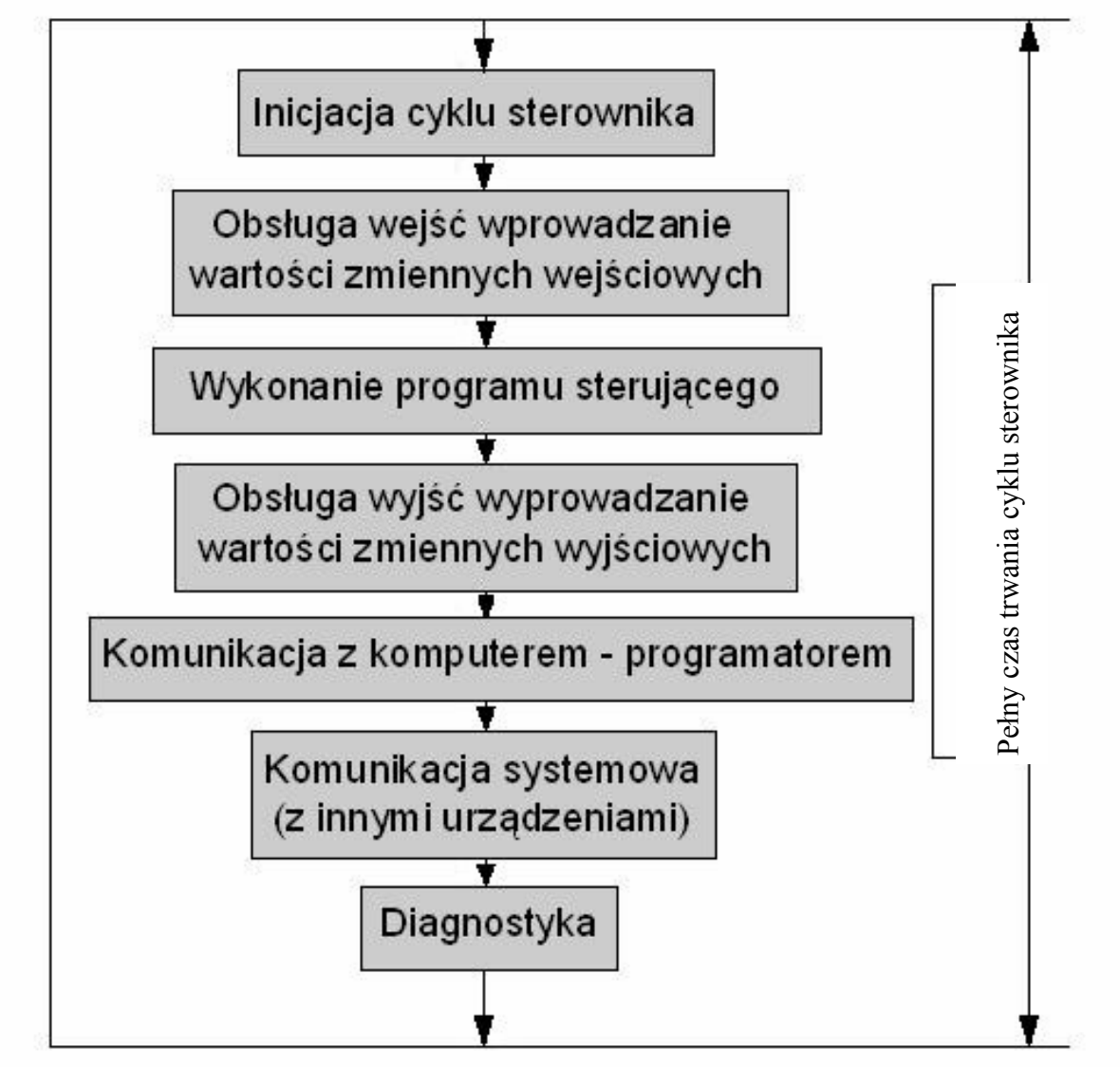
Moduł GCM – wykorzystywany do łączenia sterowników serii 90-30 do lokalnej sieci o topologii magistralowej Genius I/O Communication Bus za pomocą skrętki dwu – przewodowej. Każde z urządzeń podłączone do sieci charakteryzuje się pełnoprawnym dostępem w którym każdy z każdym może wymieniać dane. Moduł komunikacyjny GCM umożliwia komunikację z wykorzystaniem zmiennych globalnych. Możliwości te rozszerza GBC, który ma znacznie większe możliwości komunikacyjne i dodatkowo umożliwia przesyłanie paczek informacji oraz diagnostykę sieci Genius.

Zasada działania sterowników PLC

Program sterujący wykonywany jest cyklicznie aż do zatrzymania, którego powodem może być instrukcja z komputera (programatora) lub innego urządzenia zewnętrznego. Tryby pracy sterownika:

- STANDARD PROGRAM SWEEP – standardowy cykl pracy, jest to tryb w którym zwykle działają sterowniki 90-30 i Micro

- STOP WITH I/O DISABLED – tryb zatrzymania z nieczynnymi wejściami i wyjściami
- STOP WITH I/O ENABLED – tryb zatrzymania z aktualizacją zmiennych wejściowych i ustawianiem wyjść
- tryb ze stałym czasem trwania cyklu
  - ✓ Inicjalizacji cyklu
  - ✓ Obsługi wejść
  - ✓ Wykonania programu sterującego
  - ✓ Obsługi wyjść
  - ✓ Obsługi programatora
  - ✓ Obsługi innych urządzeń
  - ✓ Diagnostyki



Standardowy cykl pracy sterownika

3. Przegląd sprzętu, oprogramowania dostępnego na rynku oraz obszary ich zastosowań

3.1. Rodzaje sterowników dostępnych obecnie na rynku.

90-30

Sterowniki tej serii są bardzo łatwe w instalacji, użytkowaniu oraz programowaniu. Mają bardzo szeroki obszar zastosowań i są jednymi z najłatwiej dostępnymi sterownikami w Polsce. Można je stosować do automatyzacji prostych systemów jak i bardziej złożonych poprzez łączenie ich w sieć. Są najszybszymi urządzeniami i oferują wiele dodatkowych funkcji poprzez zastosowanie modułów rozszerzających. Umożliwiają one również:

- zabezpieczenie dostępu do programu sterującego oraz pamięci sterownika,
- diagnostykę i konfigurowanie modułów sterownika,
- programowanie w językach: Microsoft C, Instruction List, Ladder Logic, State Logic.

90-70

Znajdują zastosowanie w dużych systemach o dużej liczbie wejść i wyjść. Charakteryzują się dużą szybkością przetwarzania danych, skutecznymi zabezpieczeniami oraz bardzo dokładnymi obliczeniami. Używane są do pracy sieciowej. Są łatwe w użytkowaniu, instalowaniu i programowaniu. Jednostka centralna realizuje zadania związane z kontrolą systemu, posiada koprocesor logiczny oraz automatycznie rejestruje błędy w systemie.

VersaMax

Jest to jeden z nowszych sterowników. Może pracować osobno lub być częścią większego systemu. Jego CPU obsługuje do ośmiu kaset, każda z nich może zawierać do ośmiu modułów wejścia – wyjścia.

VersaMax Micro

Jest to nowa rodzina sterowników stosowanych do prostych (maszyny pakujące, rozdzielnie elektryczne) lub złożonych systemów sterowania. Są to niewielkie sterowniki do których można dołączyć maksymalnie cztery moduły rozszerzające. Programowanie tych strowników możliwe jest tylko pod systemem Windows.

VersaMax Nano

Pomimo niewielkich wymiarów realizuje on wiele funkcji dostępnych w większych jednostkach centralnych. Posiada funkcje umożliwiające sterowanie napędami w prostych systemach transportowych lub maszynach pakujących. Jego CPU posiada 4kB pamięci programowej RAM do której ładowany jest program sterujący w trakcie uruchomienia z nielotnej pamięci FLASH.

OCS

Jest to sterownik zintegrowany z panelem operatorskim. Moduły rozszerzające montowane są z tyłu i może ich być maksymalnie cztery. Szybkość jego wynika z tego, że:

- jeden program dla sterownika i panelu operatorskiego,

- panel operatorski połączony ze sterownikiem bez użycia łączy szeregowych,
- 64 kB pamięci dla programu sterownika i 64 kB pamięci dla zawartości ekranów,
- czas skanu programu 0,7 ms/kB programu.

Sterowniki te posiadają zegar czasu rzeczywistego i mogą być stosowane do sterowania małymi urządzeniami i zbierania danych w rozproszonych systemach.

3.2. Aplikacje do programowania sterowników PLC.

Obecnie na rynku dostępne są dwie najpopularniejsze aplikacje do programowania sterowników PLC firmy GE Fanuc, są to LM90 oraz Versa Pro.

LM90 jest programem stosunkowo starym i obecnie dosyć rzadko używanym. Jest to aplikacja działająca w środowisku DOS. Program dla sterownika tworzy się za pomocą klawiszy funkcyjnych, które służą również do konfiguracji i poruszania się po programie. Jego cechą jest to, że konstruowanie programu dla sterownika polega na tworzeniu pojedynczych rungów. Po skonstruowaniu rungu musi on być zapisany i w tym momencie możemy przystąpić do tworzenia kolejnego rungu. W programie zaimplementowano także typowe operacje edycyjne (kopiowanie rungów, wklejanie, wycinanie). Poprawność tworzonego programu, jak i skojarzonych zmiennych z funkcjami jest sprawdzana w momencie zapisania aktualnego rungu. Jeśli zostanie wykryty błąd w rungu, kursor ustawi się w miejscu zawierającym błąd. Stosowany jest głównie do programowania sterowników serii 90-30. Program nie posiada kilku funkcji, które są dostępne w nowszych programach. Efektem działania programu jest utworzenie katalogu o nazwie odpowiadającej nazwie projektu w którym znajdują się cztery pliki w których przechowywana jest informacja o budowie programu sterownika.

VersaPro jest nowszym i bardziej funkcjonalnym narzędziem do programowania sterowników. Przeznaczona jest dla systemów Windows. Główne różnice pomiędzy LM90 to:

- większy asortyment dostępnych bloków funkcyjnych w VersaPro
- większe możliwości konfiguracji sterownika
- sposób obsługi
- różne metody pisania programu
- format zapisu plików
- możliwości edycyjne

Posiada ona standardowe opcje charakterystyczne dla programów Windowsowych. Program integruje w sobie narzędzia do edycji programu, konfiguracji sterownika, deklaracji zmiennych, edytora listy instrukcji, wizualizacji tablicy błędów, itp. Pomimo wysokiej funkcjonalności program jest narzędziem stosunkowo prostym w użyciu, co świadczy o poprawnej i dobrze przemyślanej architekturze aplikacji.

4. Język LD dla sterowników GE Fanuc.

4.1. Zasady tworzenia programu drabinkowego

Język schematów drabinkowych LD (*ladder diagrams*) zaliczany jest do języków graficznych, co oznacza, że umożliwia on zrealizowanie zadania sterowania za pomocą standaryzowanych symboli graficznych. Symbole te umieszcza się w obwodach podzielonych na **rungi**. Każdy rung może mieć maksymalnie osiem lini oraz dziesięć kolumn. Wzajemnie połączone symbole wewnątrz rungu tworzą obwód, który ograniczony jest zarówno z lewej jak i z prawej strony przez tzw. szyny prądowe. Wykonanie programu sterownika polega na określeniu sposobu przepływu sygnału, od lewej do prawej strony poprzez znajdujące się w poszczególnych rungach przekaźniki oraz inne elementy. Rungi analizowane są według kolejności pojawienia się na schemacie, zaczynając od pierwszego a kończąc na zawierającym instrukcję END. Dodatkowo obowiązują następujące zasady:

- Jeżeli w rungu występują połączenia równoległe to najpierw analizowana jest linia położona najniżej.
- Jeżeli występuje konieczność umieszczenia w rungu większej ilości elementów niż dziesięć, to można użyć styków kontynuacji i przenieść sygnał do następnej linii.
- Jeżeli rung zawiera cewkę załączaną zboczem sygnału to musi to być jedyna cewka w tym rungu.
- Rung musi zawierać przynajmniej jeden styk przed cewką, blokiem funkcyjnym, instrukcją skoku lub linią pionową.
- Obwód musi być poprawnie skonstruowany z logicznego punktu widzenia – nie może zawierać rozgałęzień mających początek lub koniec w innej gałęzi.

Jeżeli występuje konieczność uruchamiania bloku funkcyjnego w każdym cyklu sterownika to nie należy łączyć go z lewą szyną prądową, lecz użyć styku z przypisaną zmienną ALW_ON.

4.2. Typy danych i rodzaje zmiennych.

Pamięć sterowników GE Fanuc dzieli się na dwie zasadnicze grupy – pamięć bitową i pamięć rejestrową. Pamięć rejestrowa ma organizację 16 – bitową. W przypadku użycia tej pamięci mamy do dyspozycji następujące typy: WORD, INT, DINT. Gdy zmienna zostanie zadeklarowana jako DINT należy pamiętać o tym, że zmienna ta znajduje się w dwóch kolejnych rejestrach. Zmienna zapisana w pamięci bitowej może przyjmować wszystkie typy. W przypadku zadeklarowania zmiennej jako typ inny niż BIT traktowana jest jako ciąg bitów o określonej długości. Istnieje w sterowniku pewna grupa zmiennych nazywanych systemowymi, które dostarczają nam informacji o stanie sterownika, błędach oraz czasie. Zmienne typu %S mogą tylko odczytywane, natomiast zmienne typu %SA, %SB, %SC mogą być również zapisywane. Wszystkim zmiennym systemowym zostały przyporządkowane nazwy symboliczne. Nazwy symboliczne mogą być przyporządkowywane wszystkim zmiennym, ale należy pamiętać o tym, że nazwy te muszą być unikatowe i powinny odzwierciedlać potencjalne zastosowanie.

Typ	Nazwa	Opis
BIT	Bit	Zmienna logiczna przyjmująca wartości 0 lub 1
BYTE	Bajt	Zmienna zawierająca 8 kolejnych bitów. Zakres wartości od 0 do 16#FF
WORD	Słowo	Zmienna zawierająca 16 kolejnych bitów. Zakres wartości od 0 do 16#FFFF
INT	Liczba całkowita 16 – bitowa	Liczba całkowita w kodzie dopełnienie do dwóch zajmująca 16 bitów. Za kres wartości od -32768 do 32767.
DINT	Liczba całkowita 32 – bitowa	Liczba całkowita w kodzie dopełnienie do dwóch zajmująca 32 bity. Zakres wartości od -2147483648 do 2147483647
BCD-4	Czterocyfrowa cyfra dziesiętna w formacie BCD	Liczba dziesiętna, której każda z czterech cyfr jest kodowana binarnie na czterech bitach. Zakres zawartości od 0 do 9999.

Tabela 1. Typy danych.

Nazwa	Opis	Symbol	Przykład
Wejścia binarne	Zmienna opisująca fizyczne wejście dwustanowe. Może być przyporządkowana stykowi.	%I	%I0012
Wyjścia binarne	Zmienna reprezentujące fizyczne wyjście dwustanowe. Może być przyporządkowane cewce lub stykowi.	%Q	%Q0023
Wewnętrzna zmienna binarna	Reprezentuje wewnętrzną zmienną binarną programu sterującego. Może być przyporządkowane cewce lub stykowi.	%M %T	%M0215 %T0010
Zmienne systemowe	Zmienne reprezentujące dane systemowe (błędy działania, aktualny czas, itp.). Mogą być przyporządkowane stykom, a niektóre z nich stykom lub cewkom.	%S	%S0007 %SA0006
Zmienne globalne	Zmienne reprezentujące dane binarne wspólne dla kilu sterowników pracujących w sieci.	%G	%G0016

Tabela 2. Zmienne binarne.

Nazwa	Opis	Symbol	Przykład
Wejścia analogowe	Zmienna reprezentująca rejestr wejścia analogowego.	%AI	%AI0007
Wyjścia analogowe	Zmienna reprezentująca rejestr wyjścia analogowego.	%AQ	%AQ0012
Rejestr	Zmienna oznaczająca komórkę pamięci.	%R	%R1234

Tabela 3. Zmienne rejestrowe.

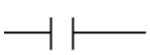
Symbol	Nazwa	Opis
%S0001	FST_SCN	=1, jeżeli bieżący cykl pracy jest pierwszym cyklem programu sterownika
%S0002	LST_SCN	=1, jeżeli bieżący cykl pracy jest ostatnim cyklem programu sterownika
%S0003	T_10MS	Styk generatora sygnału prostokątnego o okresie 10ms
%S0004	T_100MS	Styk generatora sygnału prostokątnego o okresie 100ms
%S0005	T_SEC	Styk generatora sygnału prostokątnego o okresie 1s
%S0006	T_MIN	Styk generatora sygnału prostokątnego o okresie 1min
%S0007	ALW_ON	Zmienna zawsze równa 1
%S0008	ALW_OFF	Zmienna zawsze równa 0.
%S0009	SY_FULL	=1, gdyby tablica błędów PLC wypełniona całkowicie
%S0010	IO_FULL	=1, gdyby tablica błędów I/O wypełniona całkowicie
%S0011	OVR_PRE	=1, jeżeli wykonywane jest forsowanie wartości z blokadą dla zmiennych %I, %Q, %M, %G
%SA0001	PB_SUM	=1, jeśli obliczona wartość sumy kontrolnej programu użytkownika jest różna od wartości odniesienia
%SA0002	OV_SWP	=1, gdy poprzedni cykl sterownika trwał dłużej, niż zadeklarowana wartość.
%SA0003	APL_FLT	=1, gdy wystąpił błąd w programie użytkownika
%SA0009	CFG_MM	=1, gdy zostało wykryte niedopasowanie konfiguracji
%SA0010	HRD_CPU	=1, gdy gdy wykryto błąb sprzętowy CPU
%SA0011	LOW_BAT	=1, gdy zbyt niskie napięcie baterii
%SA0014	LOS_IOM	=1, gdy brak komunikacji CPU z modułem I/O
%SB0010	BAD_RAM	=1, gdy wykryto błąd pamięci RAM w czasie załączania sterownika
%SB001	BAD_PWD	=1, gdy wykryto błędne hasło dostępu
%SC0009	AN_FLT	=1, gdy wykryty został jakikolwiek błąd
%SC0010	SY_FLT	=1, gdy wystąpił jakikolwiek błąd, który powinien być zapisany w tablicy błędów PLC
%SC0011	IO_FLT	=1, gdy wystąpił jakikolwiek błąd, który powinien być zapisany w tablicy błędów I/O

Tabela 4. Wybrane zmienne systemowe sterowników GE Fanuc

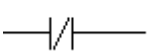
5. Opisy podstawowych elementów języka LD.

5.1. Funkcje przekaźnikowe.

Są podstawowymi elementami występującymi w schemacie drabinkowym. Składają się z cewek i styków. Głównym zadaniem styków jest sterowanie przebiegiem programu. Rozróżniamy dwa rodzaje styków: normalnie otwarty i normalnie zamknięty. W przypadku zmienna skojarzona ze stykiem ma wartość równa 1 sygnał zostaje przekazany na prawo od styku. W przeciwnym wypadku sygnał nie zostaje przesłany. Styk normalnie zamknięty działa dokładnie odwrotnie.

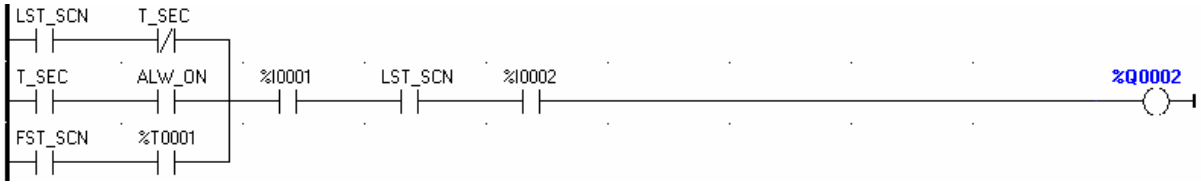


Styk normalnie otwarty



Styk normalnie zamknięty

Możliwe jest łączenie szeregowe równoległe oraz mieszane styków, przez co otrzymuje się odpowiednie funkcje logiczne.



Cewki przekaźników używane są do ustawiania wartości zmiennych z nimi skojarzonych. Wzbudzenie cewki uwarunkowane jest dopływem sygnału z poprzedzającego ją obwodu. Cewki znajdują się zawsze w ostatniej kolumnie. Istnieje kilka rodzajów cewek: normalnie otwarte, normalnie zamknięte, załączane zboczem narastającym, załączane zboczem opadającym, ustawiające i resetujące oraz styki kontynuacji.

5.2. Funkcje arytmetyczne.

Funkcje arytmetyczne służą do wykonywania operacji matematycznych. Wszystkie bloki tego typu posiadają wejście zezwalające od którego uwarunkowane jest zadziałanie funkcji. Funkcje arytmetyczne dają nam możliwość operacji tylko na danych typu INT lub DINT. Istnieje kilka typów bloków: ADD, SUB, MUL, DIV,MOD, SQRT. Ich nazwy pochodzą od angielskich słów odpowiadających realizującym przez nie

działaniom. Każdy z tych bloków posiada wejścia służące do określenia argumentów wejściowych oraz jedno wyjście służące do zapisu rezultatu działania funkcji. Oprócz tego posiadają wyjście świadczące o powodzeniu lub niepowodzeniu przeprowadzonej operacji. W przypadku, gdy operacja wykona się prawidłowo, na wyjściu ok ustawiany jest sygnał. W przypadku niepowodzenia na wyjściu ok nie jest dostępny sygnał, a dodatkowo na wyjściu służącym do zapisu wyniku operacji ustawiana jest wartość maksymalna wartość górna lub dolna w zależności od kierunku przekroczenia zakresu. W sterownikach, które udostępniają operacje zmiennoprzecinkowe wszystkie te funkcje mogą być używane dla zmiennych typu REAL.

5.3. Funkcje relacji.

Funkcje relacji wykonywane są wtedy, gdy na wejście zezwalające dopływa sygnał. Jeżeli argumenty funkcji spełniają relację określoną przez rodzaj bloku to na wyjściu wystawiany jest sygnał. Bloki tego typu nie posiadają wyjścia typu ok. Argumentami mogą być liczby INT i DINT, a dodatkowo funkcja RANGE umożliwia porównanie danych typu WORD.

5.4. Timery i liczniki.

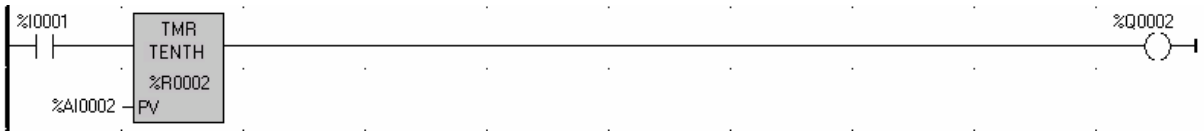
Timery umożliwiają realizację uwarunkowań czasowych. Zliczanie czasu odbywa się w określonych jednostkach czasu (0,1s 0,01s 0,001s) z dokładnością do jednej jednostki. Liczniki dają nam możliwość zliczania liczby zboczy narastających występujących w dopływającym do nich sygnale. Każdy timer lub licznik wykorzystuje trzy kolejne słowa pamięci do przechowywania następujących danych:

- wartość bieżąca CV
- wartość zadana PV

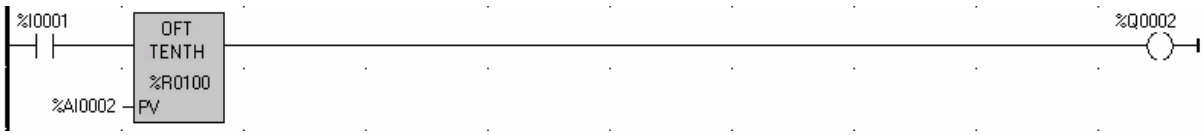
- słowo kontrolne

Do poprawnego działania timera lub licznika konieczne jest przyporządkowanie mu adresu pierwszego z trzech rejestrów. Należy pamiętać, aby nie zmieniać wartości tych rejestrów w innych miejscach programu. Do timerów zaliczamy:

- TMR – jego wartość aktualizowana jest w każdym cyklu pracy sterownika w którym timer jest aktywny czyli pokazuje on czas kiedy na wejście zezwalające dopływa sygnał, przeciwnym wypadku timer jest resetowany. Po osiągnięciu wartości zadanej przesyłany jest sygnał na prawo od bloczka.

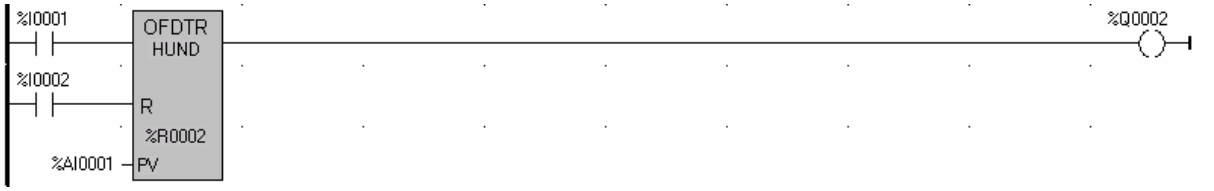


- OFDT - jego wartość aktualizowana jest w każdym cyklu pracy sterownika w którym timer jest nieaktywny czyli pokazuje on czas kiedy na wejście zezwalające nie dopływa sygnał, przeciwnym wypadku timer jest resetowany. Przed osiągnięciem wartości zadanej przesyłany jest sygnał na prawo od bloczka.



- ONDTR – (timer z pamięcią) działa podobnie jak TMR. Różnica polega na tym, że po zaniku sygnału zezwalającego wartość bieżąca jest podtrzymywana. Po wznowieniu tego sygnału kontynuowane jest odmierzanie czasu. Wartość bieżąca jest resetowana w momencie dopłynięcia na wejście resetujące sygnału.

- OFDTR – (timer z pamięcią) działa podobnie jak OFDT. Różnica polega na tym, że Przy ustawieniu sygnału zezwalającego wartość bieżąca jest podtrzymywana. Po zaniku tego sygnału kontynuowane jest odmierzanie czasu. Wartość bieżąca jest resetowana w momencie dopłynięcia na wejście resetujące sygnału.

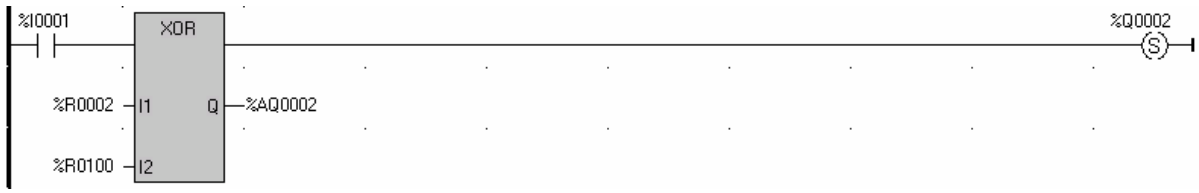


- UPCTR – służy do zliczania impulsów. Jest to licznik, który inkrementuje wartość bieżącą po pojawieniu się odpowiedniego przebiegu sygnału na wejściu (zbocze narastające). Po osiągnięciu wartości zadanej na wyjściu pojawia się sygnał. Wartość bieżąca zwiększana jest do momentu osiągnięcia wartości maksymalnej lub zresetowania licznika.

- DNCTR – służy do zliczania impulsów. Jest to licznik, który dekrementuje wartość bieżącą po pojawieniu się odpowiedniego przebiegu sygnału na wejściu (zbocze narastające). Po osiągnięciu wartości równej zero na wyjściu pojawia się sygnał. Wartość bieżąca zmniejszana jest do momentu osiągnięcia wartości minimalnej (-32768) lub zresetowania licznika.

5.5. Funkcje na ciągach bitów.

Grupa ta obejmuje funkcje logiczne, takie jak: mnożenie i dodawanie logiczne, negacja, rotacja ciągu bitów, testowanie, ustawianie i resetowanie określonego bitu w słowie. Operacje te wykonywane są na danych typu WORD. Część z nich może być wykonywana na ciągach słów określonych parametrem Len. Funkcje AND, OR, XOR mogą również służyć do tworzenia masek.



5.6. Funkcje przesyłania danych i operacje tablicowe

Do tej grupy zalicza się takie funkcje jak: MOVE, BLKMOV, BLKCLR, SHFR, BITSEQ, COMMREQ służące do przemieszczania danych w pamięci oraz funkcje ARRAYMOV, SRCH_EQ, SRCH_NE, SRCH_GT, SRCH_GE, SRCH_LT, SRCH_LE związane z działaniami na tablicach.

5.7. Funkcje konwersji.

Można tutaj zaliczyć takie funkcje, jak: BCD4_TO_INT, INT_TO_BCD4. Konwertują one dane wejściowe na dane wyjściowe. Zawsze kiedy dopływa sygnał na wejście zezwalające, jest on przesyłany na wyjście ok.

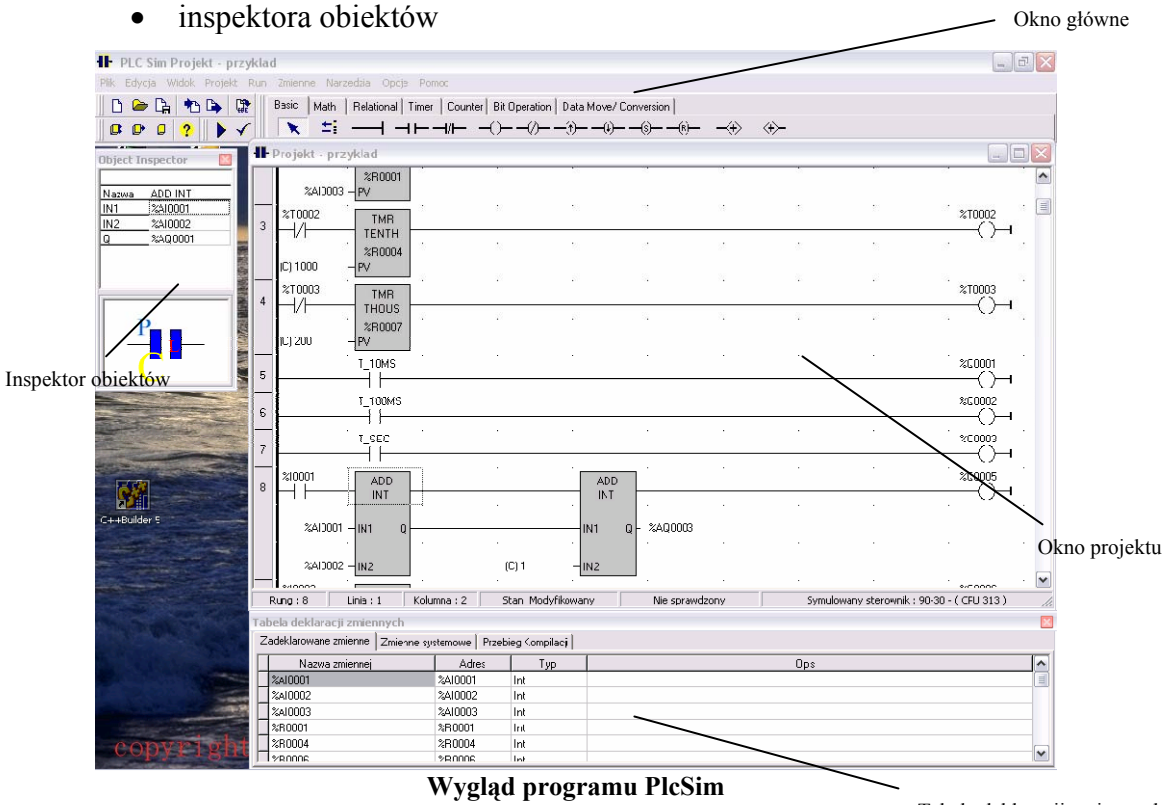
6. Opis symulatora PLC Sim.

Program przeznaczony jest do symulowania działania rzeczywistego sterownika. Umożliwia on sprawdzanie przebiegu napisanego w nim programu oraz przetestowanie go w różnych sytuacjach krytycznych. Poprzez symulację zwiększa się prawdopodobieństwo, że program sprawdzi się w warunkach rzeczywistych. Dzięki wymuszonej zmianie określonych zmiennych można analizować zachowanie się sterownika w różnych sytuacjach i skutki wykonania programu na obiekcie rzeczywistym.

6.1. Budowa programu.

Program gotowy do edycji składa się z czterech podstawowych okien:

- głównego (PlcSim)
- projektu zawierającego obszar roboczy programu
- tabeli deklaracji zmiennych
- inspektora obiektów

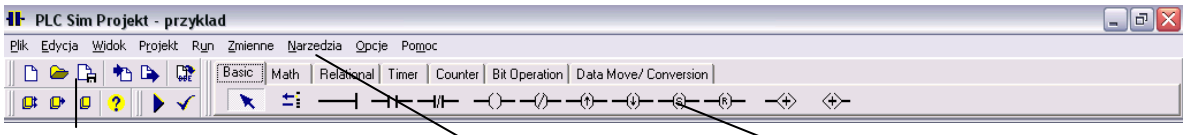


Wygląd programu PlcSim

Poszczególne okna mogą być dowolnie rozmieszczane oraz mogą być modyfikowane ich rozmiary. Możliwe jest również ukrycie niewykorzystywanych okien, a w razie potrzeby ich przywrócenie.

6.2. Opis elementów menu programu.

Po uruchomieniu programu pojawia się główne okno programu, które dzieli się na trzy podstawowe części.



Paski narzędzi Menu główne Menu funkcyjne

Menu główne zawiera:

- Plik – mieszczą się w nim wszystkie operacje związane z działaniami na plikach, takie jak: zapis, odczyt, import, drukowanie
- Edycja – odpowiada za operacje na rungach i liniach programu, takie jak dodawanie, usuwanie czy modyfikacja.
- Widok – umożliwia wyświetlenie nieaktywnych okien programu.
- Projekt – dostarcza informacji o aktualnym projekcie.
- Run – umożliwia przetestowanie programu oraz jego symulację.
- Zmienne – umożliwia operacje na deklarowanych zmiennych.
- Narzędzia – zawiera dodatkowe narzędzia przydatne podczas użytkowania programu.
- Opcje – umożliwia konfigurowanie aplikacji.
- Pomoc – zawiera informacje na temat programu oraz pomoc.

Paski narzędzi dają możliwość szybkiego dostępu do najczęściej używanych funkcji menu, przez co podnoszą komfort i szybkość pracy.

Menu bloków funkcyjnych – udostępnia wszystkie elementy z których możemy budować program dla sterownika. Podzielone są one na kategorie w zależności od typu. Znajduje się tam również opcja służąca do kasowania elementu.

6.3. Przeznaczenie i działanie poszczególnych okien aplikacji.

6.3.1. Okno deklaracji zmiennych

Podzielone jest na trzy główne sekcje. W pierwszej z nich znajduje się tabela, w której użytkownik może zadeklarować używane przez siebie zmienne. Należy tam zapisać nazwę zadeklarowanej zmiennej, jej typ, adres oraz opcjonalnie opis. Jeżeli nazwa zmiennej ma być taka sama jak jej adres należy rozpocząć deklarowanie zmiennych od drugiej kolumny. Poprawna składnia powinna wyglądać następująco:

%	Typ adresu	Adres

Przykład poprawnie wpisanej wartości: %R4, %q5, %I9, %AQ0016

Procedura deklarowania nowej zmiennej zaczyna się w momencie kliknięcia w siatce okna deklaracji zmiennych. Jeśli użytkownik zaczyna deklarację od drugiej kolumny, wtedy zostaje sprawdzana zawartość pierwszej. Jeżeli jest pusta, to wartość jest kopiowana, w przeciwnym wypadku wartość ta pozostaje bez zmian. Sprawdzana jest również poprawność wprowadzanego adresu, jeżeli jest niepoprawny, to proces zostaje przerwany. W przypadku próby wypełnienia kolumny trzeciej lub czwartej zostaje sprawdzona zawartość drugiej. Jeżeli w kolumnie drugiej znajduje się pusty string, to próba zakończy się niepowodzeniem. Jeżeli użytkownik chce zmodyfikować zawartość któregokolwiek wiersza, to musi liczyć się z tym, że wartość z pierwszej kolumny zostanie niezmieniona. Podczas wpisywania adresów analizowana jest wartość wpisana w to pole i zostaje ona uzupełniona o odpowiednią ilość zer, ewentualnie zostaje dokonana konwersja liter małych na wielkie. W momencie uaktywnienia trzeciej kolumny, pojawia się pole kombi zapobiegające wpisaniu niewłaściwego typu zmiennej. Tą kolumnę także należy wypełnić w celu zapewnienia prawidłowego działania programu.

W drugiej sekcji znajduje się tabela z zadeklarowanymi w niej zmiennymi systemowymi. Jej edycja jest niemożliwa, ponieważ ma wyłącznie charakter informacyjny.

Trzecia sekcja poświęcona jest na wyświetlanie informacji dotyczących procesu analizy poprawności programu. Jeśli wywoływana jest procedura sprawdzania projektu i napotkany zostanie błąd, wtedy do okna dodawana jest linia informująca o rodzaju błędu i

miejscu jego wystąpienia. Dodatkowo na pasku statusu jest wypisywana informacja o wyniku sprawdzania projektu.

6.3.2. Okno projektu

Umożliwia edycję programu. Jest to obszar, w którym poprzez złączenie określonych elementów otrzymujemy obwód.

Związane jest z tym oknem szereg zdarzeń. Jednym z nich jest wybór aktywnej komórki. Najpierw na pasku statusu są wyświetlone odpowiednie informacje. Następnie sprawdzany jest aktywny przycisk w menu funkcyjnym i w zależności od niego wykonywana jest odpowiednia procedura. Jeśli jest to przycisk kasowania, to blok ten zostaje usunięty a komórki, które on zajmował są repetowane. Natomiast jeśli jest to wstawianie nowego bloku to program sprawdza, czy w tych komórkach znajdują się już elementy. Jeśli zajdzie taka sytuacja, to element znajdujący się w tych komórkach zostaje wykasowany i zastąpiony nowym. Jeśli wybranym elementem jest cewka, to automatycznie zostaje ona umieszczona w ostatniej kolumnie i zostaje dorysowane połączenie do poprzedniego elementu. Operacje te dokonywane są na komórkach, które dany blok ma zajmować. Ich liczba zależna jest od rodzaju bloku. Po wykonaniu tych operacji obszar zostaje przerysowany.

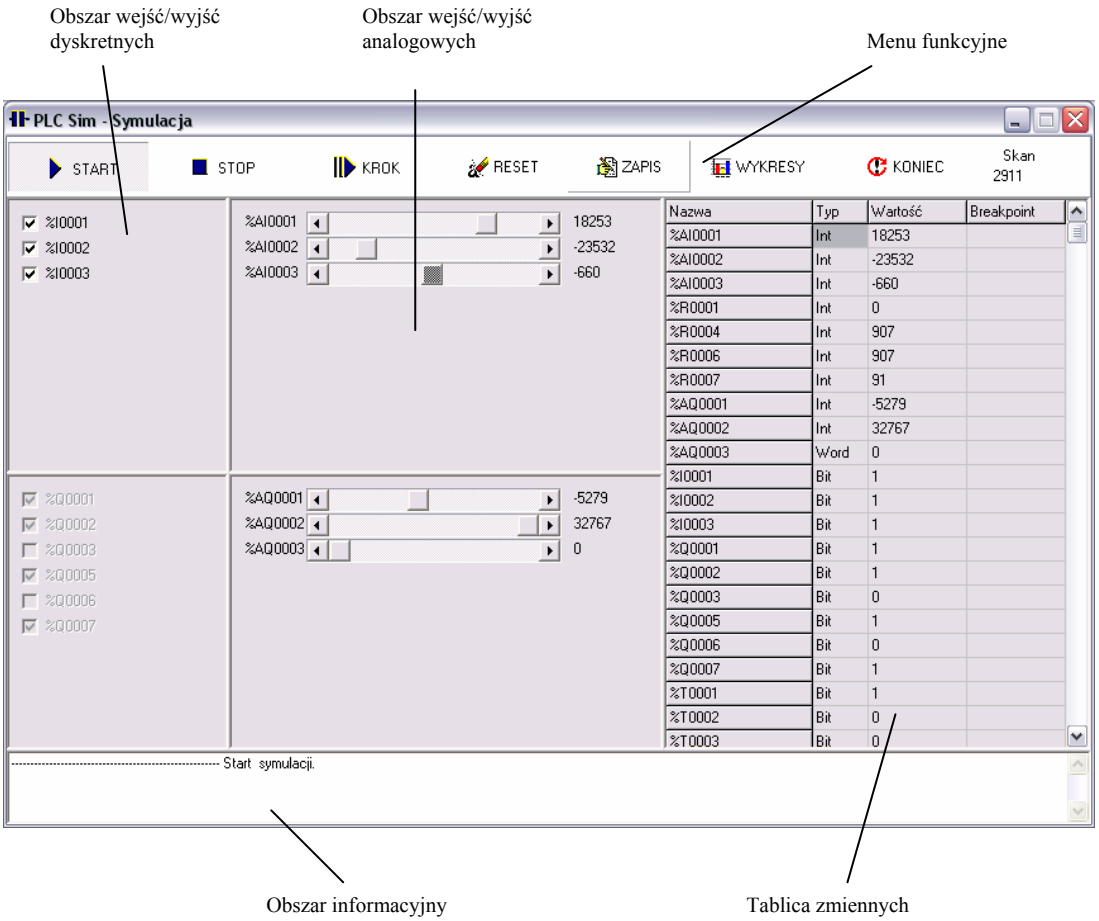
Poszczególne części bloków i opisy do nich są wyświetlane jako odrębne elementy. Okno to prezentuje graficzną postać programu użytkownika.

6.3.2. Inspektor obiektów

Okno, w którym użytkownik w sposób łatwy i przyjemny nadaje poszczególnym funkcjom parametry niezbędne do prawidłowej symulacji i działania programu. Przyporządkowanie zmiennych polega na wybraniu spośród zadeklarowanych zmiennych odpowiedniego adresu. Możliwe jest ręczne wpisanie adresu nie zadeklarowanego w tabeli (nie będzie on wtedy widoczny podczas symulacji). Zasada wpisywania adresów jest taka sama jak w przypadku tabeli deklaracji zmiennych.

Zdarzenia związane z tym formularzem odnoszą się przede wszystkim do czynności wyboru wartości z pola kombi i przepisania ich do parametrów danego bloku. Do zbioru wartości tego pola są przepisywane wyselekcjowane wartości z drugiej kolumny tabeli deklaracji zmiennych.

6.3.3. Okno symulacji programu.



Okno to przeznaczone jest do przeprowadzania analizy napisanego programu. Umożliwia ono uruchomienie symulacji, zatrzymanie oraz pracę krokową. Poprzez manipulację suwakami i polami wyboru możliwa jest modyfikacja zmiennych wejściowych. Istnieje możliwość zaobserwowania sygnałów pojawiających się na wyjściu. Podczas analizy możliwa jest obserwacja zadeklarowanych w programie zmiennych. Dostępna jest również opcja debugowania programu przez ustawianie punktów zatrzymania tzw. breakpointów oraz modyfikacja wartości zmiennych.

Okno jest podzielone na następujące obszary:

Menu funkcyjne – znajdują się tam przyciski do sterowania procesem symulacji.

Obszar elementów do ustawiania wartości wejść – obszar, w którym możliwa jest regulacja wartości zmiennych wejściowych analogowych i dyskretnych. Dzieje się to przez obsługę kontrolek, które są tam dynamicznie wstawiane podczas ładowania formularza.

Obszar elementów do prezentacji wyjść – obszar, w którym możliwa jest obserwacja wartości zmiennych wyjściowych analogowych i dyskretnych. Dzieje się to przez obsługę kontrolek, które są tam dynamicznie wstawiane podczas ładowania formularza.

Tabela zmiennych – tabela zawierająca informacje o adresie zmiennej, jej typie oraz aktualnej wartości w czasie procesu symulacji.

Obszar informacyjny – wyświetla informacje dotyczące stanu symulatora. Ma charakter informacyjny.

Podczas uruchamiania formularza resetowane są wartości zadeklarowanych zmiennych. Następnie program jest konwertowany, tzn. wszystkie użyte bloki wraz z parametrami konwertowane na format zrozumiały przez symulator, podobne operacje są wykonywane na liście zmiennych programu. Kolejną czynnością jest wyświetlenie tabeli zmiennych wraz z opisami. Całość procedury kończy umieszczenie w odpowiednich obszarach kontrolek za pomocą których użytkownik może obserwować zachowanie się programu w konkretnych stanach.

Opcje menu

Start – włącza symulację ciągłą programu. Zostaje uruchomiony proces symulacji programu. Jeśli symulacja została wcześniej zatrzymana to funkcja ta wznawia jej działanie od poprzedniego stanu.

Stop – pozwala zatrzymać proces symulacji. Możliwe jest w tym momencie dokładne przeanalizowanie aktualnego stanu. Istnieje możliwość zapisania wartości zmiennych do pliku.

Krok – włącza tryb symulacji krokowej. Wykonywany jest jeden cykl programu. Pozwala to na zaobserwowanie jak zmieniają się poszczególne zmienne w kolejnych cyklach. W tym trybie nie działają timery, ponieważ nie jest możliwa ich krokowa symulacja.

Reset – funkcja ta ustawia wszystkie wartości zmiennych na stan wejściowy. Zerowana jest również liczba wykonanych cykli sterownika.

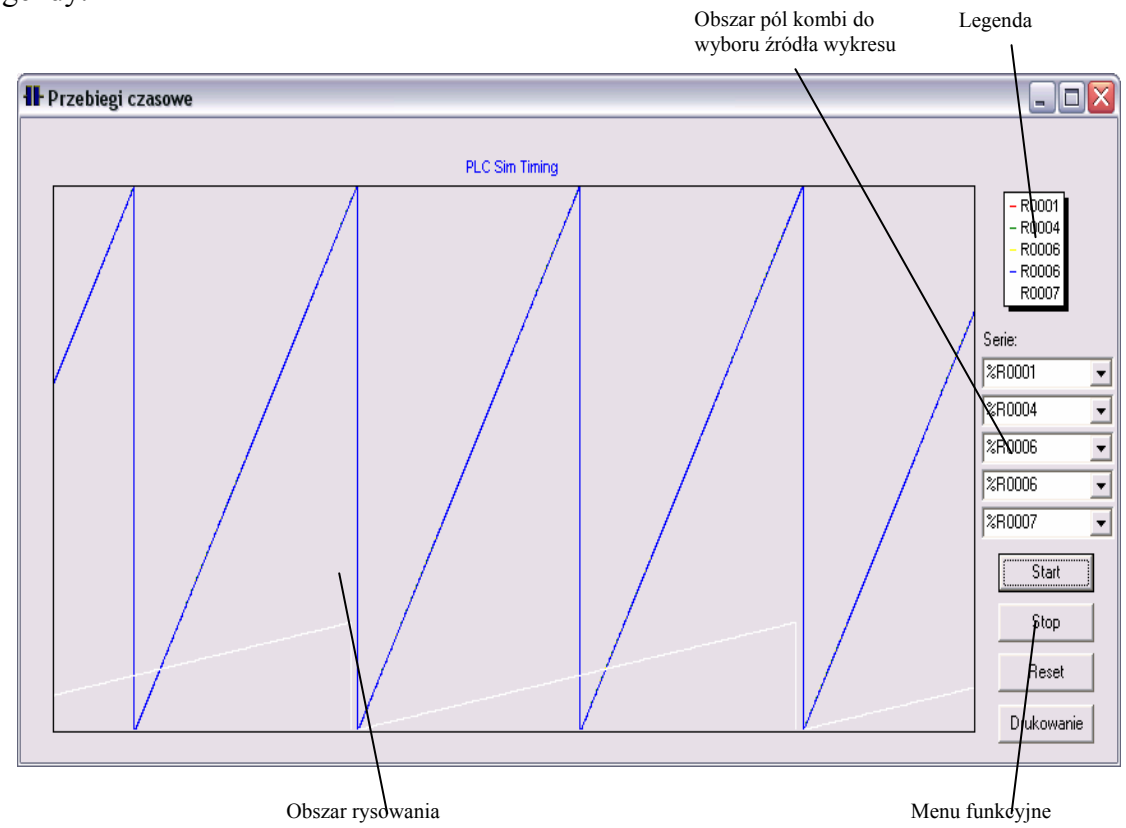
Zapis – zapisuje aktualne wartości wszystkich zadeklarowanych zmiennych do pliku w formacie html.

Wykresy – pozwalają graficznie pokazać przebiegi maksymalnie do pięciu zmiennych. Aby umożliwić rysowanie wykresów należy najpierw włączyć symulację programu.

Koniec – zamyka okno symulacji i program wraca do trybu edycji programu użytkownika.

6.3.4. Przebiegi czasowe

Możliwość prezentacji graficznej przebiegów wybranych sygnałów jest bardzo użytecznym i pomocnym narzędziem podczas analizy programu użytkownika. Możliwe jest obserwowanie maksymalnie pięciu zmiennych w tym samym czasie. Przebiegi są prezentowane jako wykresy o kolorach linii odpowiadających kolorom zmiennych z legendy.



Obszar rysowania – jest przeznaczony do rysowania wybranych przebiegów. Dla każdej zmiennej przyporządkowany jest inny kolor wykresu.

Obszar pól kombi do wyboru źródła wykresu – pozwala na wybranie zestawu zmiennych, które mają być wyświetlone w obszarze rysowania.

Legenda – obrazuje kolory przyporządkowane do poszczególnych zmiennych.

Menu funkcyjne – zestaw funkcji do sterowania oknem. Możliwe są następujące operacje:

- Start – rozpoczyna proces rysowania wykresu,

- Stop – zatrzymuje proces rysowania wykresu

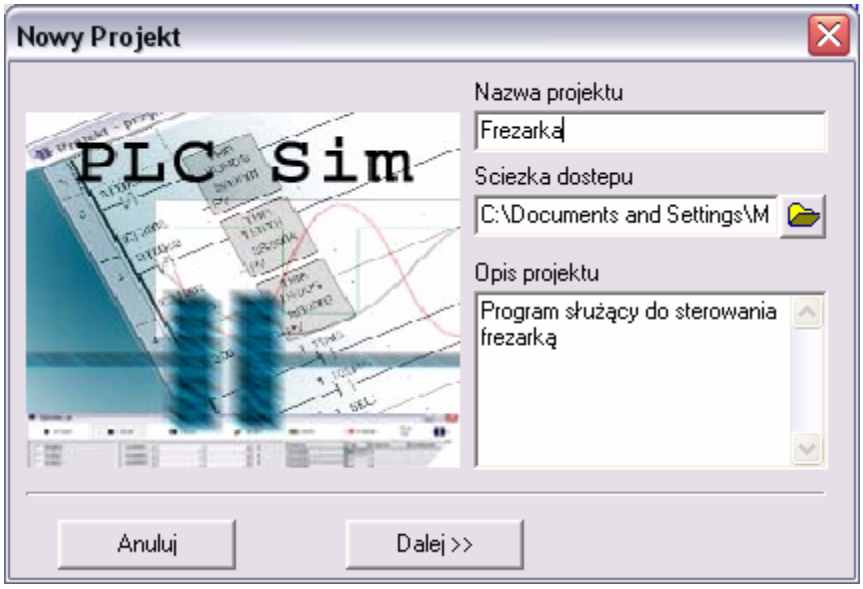
- Reset – czyści obszar wykresu i resetuje funkcję aktualizującą

- Drukowanie – pozwala wydrukować przebiegi zmiennych przedstawionych graficznie.

6.4. Programowanie i symulacja z programem PLCSim.

6.4.1.Rozpoczęcie pracy z programem.

Po uruchomieniu pojawia się główne okno aplikacji. W tej chwili użytkownik musi się zdecydować czy tworzy pojedynczy plik czy pełny projekt. W zależności od podjętej decyzji z menu plik musi wybrać zakładkę Nowy projekt lub Nowy plik. W przypadku wybrania projektu prowadzony jest przez kreator.



W pierwszym kroku definiowana jest nazwa projektu, która służy również do nadania nazwy folderowi zawierającego pliki projektu. Konieczne jest również wprowadzenie ścieżki dostępu do tego katalogu, oraz opcjonalnie informacji dotyczących projektu.

W drugim kroku konieczny jest wybór jednostki centralnej. Dostępnych jest kilkanaście jednostek oraz ustawienia domyślne. W zależności od rodzaju wybranego sterownika użytkownik będzie miał do dyspozycji określoną liczbę zmiennych. W przypadku wybrania ustawień domyślnych liczby zmiennych ustawiane są na wartości maksymalne. Gdy projekt musi być zgodny z programem LM90 należy pamiętać o tym, że symulator będzie miał ograniczoną liczbę funkcji.

Dane dotyczące projektu oraz zastosowanego sterownika dostępne są w menu Projekt -> Opis projektu.

6.4.2. Działanie podstawowych funkcji menu

Plik-> Nowy – jeśli był otwarty inny plik, to zostaje on najpierw zamknięty, a wszystkie okna ukryte. Następnie są ustawiane zmienne reprezentujące tytuły okien, a zawartość tabeli deklaracji zmiennych jest resetowana. Na zakończenie otwierane są puste okna i do tabeli deklaracji zmiennych wczytywane są zmienne systemowe sterownika.

Plik-> Otwórz – jeśli był otwarty inny plik, to zostaje on najpierw zamknięty, a wszystkie okna ukryte. Następnie wczytywana jest pamięć systemowa sterownika, po czym następuje otwarcie okna dialogowego pozwalające wybrać odpowiedni plik zapisany na dysku. W przypadku potwierdzenia przyciskiem Otwórz następuje odczytanie tego pliku.

Zastosowany został odpowiedni filtr, aby udogodnić użytkownikowi wybór właściwego pliku.

Plik-> Zapisz – zostaje otwarte odpowiednie okno dialogowe, w którym użytkownik może wybrać nazwę i położenie pliku/projektu na dysku. Po zatwierdzeniu zostaje on zapisany fizycznie na dysk. Zapis pliku został zoptymalizowany objętościowo poprzez zastosowanie skompresowanej formy zapisu poszczególnych składników klasy.

Edycja-> Czyść Aktualną Komórkę – odczytuje położenie kursora w siatce. Następnie wszystkie wartości zmiennych przyporządkowanych do tej komórki są zerowane. Jeśli jest to element o budowie wielokomórkowej, to zerowane są wszystkie komórki do niego przynależne. W miejsce usuniętego elementu zostają załadowane odpowiednie elementy graficzne.

Edycja-> Czyść Aktualny Szczebel – zostają wyszukane wszystkie linie programu przynależne do danego rungu, następnie komórki składające się na jego budowę zostają zerowane podobnie jak w przypadku czyszczenia aktualnej komórki.

Zmienne-> Zerowanie Wszystkich Zmiennych – uruchamiana jest pętla, w której zostają przypisane do zmiennych wartości zerowe. Operacja ta jest wykonywana zarówno dla danych bitowych jak i dla danych rejestrowych. Na końcu cała tabela zostaje odświeżona. Działanie to przeprowadzane jest tylko dla zmiennych zadeklarowanych przez użytkownika.

Zmienne-> Usuwanie Deklaracji Zmiennych – uruchamiana jest pętla, w której zerowana jest cała tablica zmiennych zadeklarowanych przez użytkownika. Operacja ta wykonywana jest tylko dla zadeklarowanych zmiennych.

Plik-> Import – na początku zostaje otwarte okno służące do ustalenia katalogu z którego dane mają być importowane. Sprawdzane jest wypełnienie odpowiedniego pola wyboru na formularzu. Kolejną czynnością jest przygotowanie pliku/projektu do procedury importu (wczytywane są zmienne systemowe i otwierane odpowiednie okna). Następnie odczytywany jest rozmiar importowanego pliku i do przygotowanej tablicy są wczytywane wartości bajtów reprezentujących program użytkownika. Główną częścią funkcji jest pętla,

która sprawdza kolejne ciągi bajtów, a następnie odpowiednio je interpretuje i zamienia na wartości zrozumiałe dla programu. Wartości te są następnie przyporządkowywane do odpowiednich zmiennych. W przypadku występowania połączeń równoległych, do osobnej tablicy zapisywane są węzły. Konieczne to było ze względu na budowę pliku. Niezbędne okazało się także zastosowanie dodatkowego pliku wykonywalnego ze względu na duże trudności związane z prawidłowym odczytaniem wszystkich bajtów pliku. W przypadku napotkania bloku nieidentyfikowalnego przez program, analiza zostaje przerwana. Informuje o tym odpowiedni komunikat. Gdy cała procedura importu zakończy się prawidłowo wyświetlana jest informacja.

Plik-> Eksport - na początku zostaje otwarte okno służące do ustalenia katalogu do którego dane mają być eksportowane. Sprawdzane jest wypełnienie odpowiedniego pola w formularzu. Powinien to być katalog, w którym znajduje się program LM90. Następnie przygotowywana jest tablica, w której umieszczone będą dane przeznaczone do eksportowania oraz tablice, w których znajdują się informacje o ilości styków w rungu i współrzędnych połączeń. Jest to wykonywane tylko raz dla danego rungu. Kolejnym krokiem jest wyznaczenie maksymalnych współczynników dla każdego typu adresu. Następnie element zostaje identyfikowany i zapisany do tabeli danych eksportowanych. Po tym zostaje podjęta decyzja o adresie kolejnej komórki do identyfikacji. Później do tablicy danych dopisywane są informacje stanowiące resztę zawartości pliku. Tutaj też konieczne okazało się zastosowanie odrębnego pliku wykonywalnego do zapisu zawartości programu. Na końcu tworzone są jeszcze dwa pozostałe pliki niezbędne do prawidłowego przeprowadzenia procedury eksportu i wykonywane są czynności porządkowe.

6.4.2. Podstawowe zasady tworzenia i rozwijania projektu.

W czasie pracy z programem należy pamiętać o pewnych zasadach, których przestrzeganie ułatwi pracę. Projekt powinien posiadać nazwę związaną w pewien sposób z jego przeznaczeniem. Programowanie należy rozpocząć od zadeklarowania podstawowych zmiennych, należy zastanowić się nad ich typami. Nazwy zmiennych powinny ułatwiać ich identyfikację i wykorzystanie w projekcie. Po sporządzeniu programu należy sprawdzić jego poprawność i w razie potrzeby usunąć błędy. Kolejnym krokiem jest uruchomienie symulacji. Gdy symulacja nie przebiega prawidłowo należy wykorzystać narzędzia dostępne w symulatorze takie jak debugowanie lub praca krokowa. W razie potrzeby

możliwe jest takie zmodyfikowanie zmiennych sterownika w zależności od okoliczności. W przypadku skomplikowanego projektu najlepszą metodą jest tworzenie niewielkich części i sukcesywne go rozwijanie.

6.4.3. Sposoby symulacji.

Istnieje kilka metod symulacji, co pozwala na wybranie odpowiedniej – stosownej do potrzeb projektu. Podstawowa metoda – czyli symulacja ciągła pozwala obserwować zmiany rejestrów jednostki centralnej w czasie rzeczywistym. Jest najprostszą i najszybszą metodą, jednak w pewnych sytuacjach nie daje spodziewanych rezultatów. Zmieniające się szybko wartości zmiennych nie mogą być zauważone. Sposobem nie mającym tych wad jest praca krokowa. Wykonanie kolejnych kroków odbywa się poprzez uruchomienie odpowiedniej funkcji, która wykonuje jeden cykl pracy sterownika. Proces ten jest długotrwały lecz skuteczny i umożliwia dokładne prześledzenie zmian wartości zmiennych. Należy jednak pamiętać o tym, aby w testowanym programie nie występowały timery, gdyż nie wszystkie otrzymane dane będą aktualne i zgodne z oczekiwaniami. Można łączyć te metody ze sobą, co wpływa na zwiększenie skuteczności analizy programu. Trzecia metoda łączy w sobie cechy dwóch poprzednich. Symulacja przeprowadzana jest w sposób ciągły. Dodatkowo definiowane są tzw. pułapki – to znaczy określone wartości zmiennych, przy których następuje zatrzymanie działania programu. W tym momencie możliwe jest prześledzenie wszystkich wartości interesujących nas zmiennych. Dalej praca może być kontynuowana w sposób krokowy lub ciągły aż do następnego zatrzymania.

7. Współpraca z innymi aplikacjami.

7.1. Program LM 90.

Program PlcSim umożliwia import oraz eksport danych z/do plików tworzonych w programie LM90. Umożliwia to zbudowanie obwodu, przetestowanie go i ewentualne zapisanie go w określonym wyżej formacie. Następnie plik ten może być wysłany do sterownika za pomocą programu LM90, a konkretnie za pośrednictwem protokołu SNP. Import danych daje możliwość symulowania zapisanych plików i ewentualne poprawienie błędów w nich znalezionych. Należy pamiętać o zaznaczeniu pola wyboru trybu zgodności z LM90 podczas działania kreatora projektu oraz o ograniczeniach wynikających z zaznaczenia tego pola. Wpływa na to także wybór typu jednostki centralnej (ilość poszczególnych zmiennych). Przy niektórych układach funkcje te mogą nie działać poprawnie.

7.2. Komunikacja poprzez DDE

DDE (Dynamic Data Exchange) jest to dynamiczna wymiana danych pomiędzy uruchomionymi aplikacjami. PlcSim jest serwerem co oznacza, że inne aplikacje mogą pobierać informacje przez niego udostępniane. Może także pełnić rolę klienta czyli odbierać informacje od innych aplikacji. Aplikacja serwera musi być uruchomiona wcześniej niż aplikacja klienta, ponadto w celu połączenia należy stosować następujące parametry: Name – nazwa procesu aplikacji (Splc), Topic – nazwa tematu (Splc), Item – nazwa zmiennej (bez symbolu %). Przykład instrukcji dla aplikacji MS Excel: =Splc|Splc!R0001 – powoduje to odczytanie do komórki wartości zmiennej %R1.

7.3. Ograniczenia programu.

Program działa w środowisku Windows, a co za tym idzie istnieją problemy z odzwierciedleniem czasu rzeczywistego. Podczas testowania zdarzało się, że system zawieszał wykonanie programu na czas 10 milisekund. Wymusiło to ustalenie czasu skanu symulatora na ten czas. Ma to poważne konsekwencje przy projektowaniu programów zawierających timery. Program nie obsługuje zmiennych typu REAL, nie posiada bloków służących do regulacji i innych rzadko stosowanych funkcji. Przeznaczony jest głównie do

symulowania sterowników GE Fanuc. Jest w pierwszej wersji, więc może zawierać błędy, które nie zostały jeszcze zidentyfikowane.

8. Opis poszczególnych faz projektu.

8.1. Określenie wymagań

Program z założenia miał spełniać następujące funkcje:

- Umożliwienie edycji, tworzenia programu sterownika
- Sprawdzenie jego poprawności
- Symulacja działania
- Dostarczenie metod służących do poprawy skuteczności działania programu.
- Łatwość i intuicyjność użytkowania aplikacji.
- Możliwość modernizacji aplikacji w przyszłości
- Import/eksport plików z/do programu LM90
- Zapis programu do plików

Wymagania sprzętowe i programowe

- Środowisko Windows 98 lub lepszy.
- Pamięć RAM minimum 128 MB
- Procesor minimum Pentium 200 MHz

8.2. Projektowanie

- Projektowanie klas
- Projektowanie pamięci
- Projektowanie timerów
- Opracowanie sposobu zapisu funkcji.
- Projektowanie interfejsu

8.3. Kodowanie

- Środowisko programistyczne – język C++
- Programowanie obiektowe i zastosowanie najnowszych technik programistycznych – wielowątkowość, obsługa wyjątków.

8.4. Testowanie

- Umieszczenie kolejnych wersji programu na ogólnodostępnej stronie WWW
- Testowanie poszczególnych funkcji modułów programu.

9. Proces powstawania programu

Po zebraniu wszystkich potrzebnych informacji na temat budowy, zasady działania oraz sposobów programowania sterowników PLC. Rozpoczęta została analiza dostępnych kompilatorów pod kątem ich możliwości oraz funkcjonalności. Wybrany został kompilator stworzony przez firmę Borland i działający w środowisku Windows o nazwie C++Builder 6.0. Główną jego zaletą, obok funkcjonalności, nowoczesności i ergonomiczności jest fakt, że jest on dostępny w wersji demonstracyjnej na stronie internetowej www.borland.com . Rozpoczęty został etap projektowania podstawowych algorytmów programu. Równocześnie zostały rozpoczęte prace nad prototypem interfejsu użytkownika.

9.1. Ograniczenia systemu Windows i ich wpływ na budowę programu

Z założenia program miał dotrzeć do jak najszerszego grona użytkowników. W związku z tym podjęta została decyzja, że zostanie napisany dla systemu Windows. Jak każdy produkt ma on wady i zalety. Do najważniejszych wad tego systemu wpływających na budowę i działanie symulatora należy zaliczyć problem z implementacją programów działających w czasie rzeczywistym.

System Windows jest systemem wielozadaniowym z wywłaszczaniem. Oznacza to, że pomimo iż system realizuje wiele procesów to tak naprawdę w danej chwili wykonywany jest tylko jeden z nich. Co określony czas zegar generuje przerwania do jądra sytemu, a to zawiesza wykonanie aktualnego procesu bez względu na stan w jakim on się znajduje i nie ma możliwości zablokowania takiego działania. Technika ta została nazwana wywłaszczeniem. Gwarantuje to stabilność systemu wieloprocesowego. W niektórych przypadkach (np. w programie PLCSim) uznać to należy za główne ograniczenie. Niemożliwe jest ustalenie czasu, jaki upłynie od zawieszenia wątku do jego ponownego uaktywnienia. Istnieje możliwość nadawania priorytetów wątkom, jednak operacja ta nie daje pełnej kontroli nad tym procesem. Zwiększenie stabilności wykonywania poszczególnych wątków może przynieść zminimalizowanie liczby uruchomionych aplikacji. Bezpieczny czas odświeżania aplikacji powinien wynosić minimum 100 ms. W programie PLCSim czas ten został ustalony na 10 ms, więc należy się liczyć z niejednokrotnym przekraczaniem tego czasu co jest sygnalizowane podczas symulacji

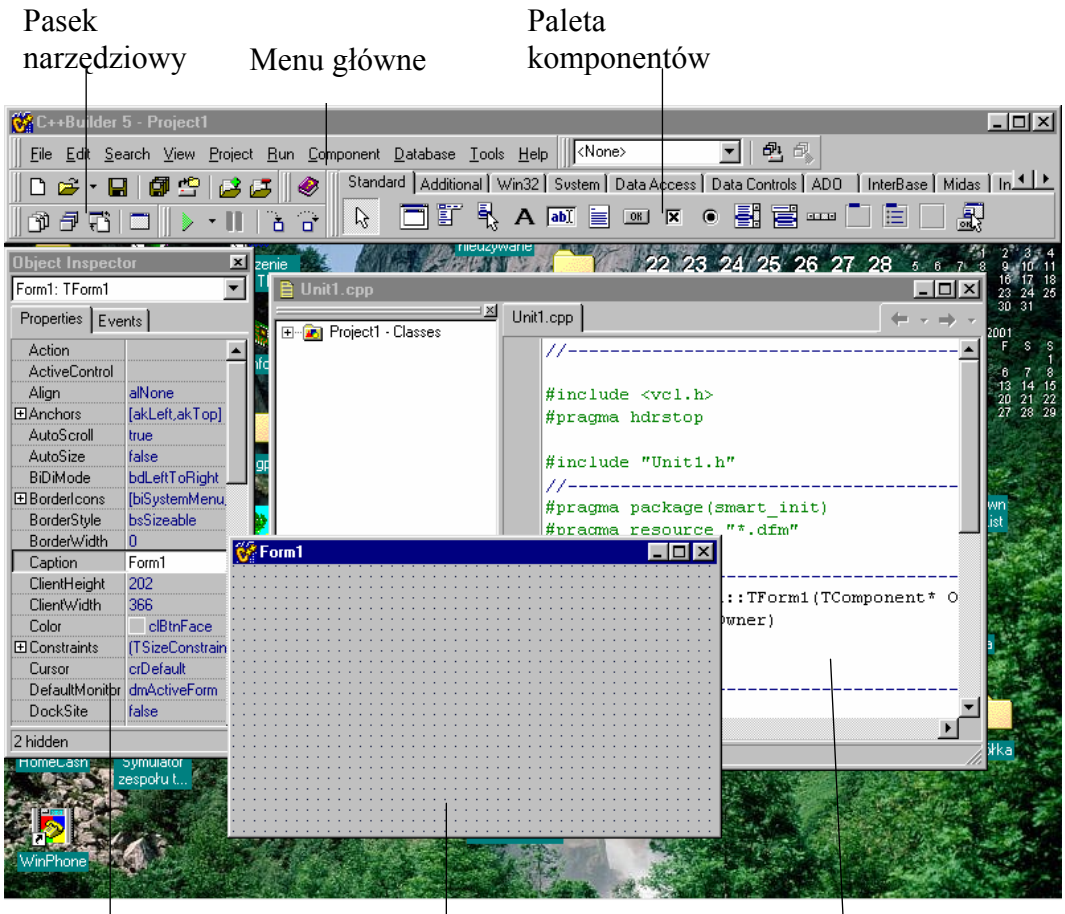
przez odpowiedni komunikat, jednak aplikacja działa poprawnie. Ze względu na możliwość zawieszenia się komputera zaleca się uruchamianie go na systemach Windows 2000 i XP.

9.2. Opis środowiska programistycznego – C++Builder 6.0 (wersja testowa)

Zaletą wybranego przez nas oprogramowania jest fakt, że łączy ono w sobie cechy środowiska RAD (ang. *Rapid Applications Development* – błyskawiczne tworzenie aplikacji) z funkcjonalnością ANSI C++. Jest on przeznaczony zarówno dla amatorów jak i profesjonalistów.

W porównaniu z innymi programami (Visual C++) charakteryzuje się on zwiększoną łatwością tworzenia złożonych aplikacji .

Po uruchomieniu C++Buildera ukaże się zintegrowane środowisko projektowe. Zbudowane jest ono z trzech podstawowych okien przy pomocy których można wykonywać podstawowe operacje na projekcie.



Biblioteka VCL (Visual Component Library)

Stanowi ona zbiór komponentów używanych podczas tworzenia aplikacji. Zawierają one w sobie dużą część kodu programu dzięki czemu programista może się skupić nad merytoryczną częścią tworzonej aplikacji. Dzięki możliwości zmiany właściwości obiektów (Object Inpector) możliwe jest sterowanie ich zachowaniem. Dodatkowo możliwe jest tworzenie własnych komponentów.

Formularze

Po utworzeniu nowego projektu automatycznie pojawia się pusty formularz główny. Odzwierciedla on główne okno programu, na którym programista może umieścić komponenty VCL (widzialne i niewidoczne) tworząc w ten sposób interfejs aplikacji. Możliwe jest wywoływanie z tego formularza okien potomnych. Są podstawową i nierozłączną częścią każdej aplikacji windowsowej.

Paski narzędziowe

Umożliwiają szybki dostęp do najczęściej wykorzystywanych opcji menu. Użytkownik ma możliwość dostosowania zawartości pasków narzędziowych do swoich potrzeb.

C++Builder zawiera następujące paski narzędziowe:

- Standard
- View
- Debug
- Custom
- Component Palette (paleta komponentów)

Paleta komponentów

Pod menu głównym znajduje się Paleta komponentów, która jest magazynem komponentów VCL. Podzielone są one na strony, które reprezentują poszczególne kategorie dzięki czemu użytkownik może łatwo znaleźć interesujący go element.

Zdarzenia i ich obsługa

Działanie programu jest sterowane za pomocą obsługi zdarzeń. Komponent może obsłużyć różne zdarzenia w zależności od akcji jaka jest wykonywana. To właśnie zdarzenia decydują o zachowaniu i właściwościach tworzonego oprogramowania.

Zmienne dostępne w środowisku C++Builder

Typ	Rozmiar (w bajtach)	C++Builder
Liczba całkowita ze znakiem	1	char
	2	short, short int
	4	int, long
	8	__int64
Liczba całkowita bez znaku	1	BYTE, unsigned short
	2	unsigned short
	4	unsigned long
Liczba zmiennoprzecinkowa	4	float
	8	double
	10	long double
Variant	16	OleVariant, VARIANT
Znak	1	char
	2	WCHAR
Łańcuch dynamiczny	–	AnsiString
Łańcuch dynamiczny znaków dwubajtowych	–	WideString
Łańcuch z zerowym ogranicznikiem	–	char *
Łańcuch znaków dwubajtowych z zerowym ogranicznikiem	–	LPCWSTR
Wskaźnik amorficzny	4	void *
Wartość logiczna (boolowska)	1	bool

Tabela 5. Typy zmiennych C++Buildera

Typy plików w programie

Podczas pracy z programem generowane są liczne pliki w których zapisane są wszystkie informacje dotyczące kompilacji, kodu programu, zawartości projektu. Są one przedstawione w tabeli poniżej.

C++Builder	Przeznaczenie pliku
BPR	Główny plik projektu
CPP	Plik kodu źródłowego modułu (w C++Builderze również główny plik źródłowy projektu)
DFM	Plik binarny opisujący kompletną strukturę formularza wraz z komponentami
H lub HPP	Plik nagłówkowy
RES	Skompilowany plik zasobów
OBJ	Skompilowany plik modułu
BPG	Plik opisujący grupę projektów
BPK	Plik zawierający listę modułów pakietu
BPI	Plik importowy biblioteki tworzony dla każdego pakietu
BPL	Biblioteka czasu wykonania (*runtime*); fizycznie jest to biblioteka DLL wzbogacona o elementy specyficznie dla Delphi (C++Buildera)
LIB	Statyczna biblioteka
~*	Kopie zapasowe rozmaitych plików
MAK	Plik tekstowy zawierający scenariusz budowania binarnej postaci projektu na podstawie jego plików źródłowych.
MAP	Plik tekstowy zawierający informacje symboliczne dla śledzenia niskopoziomowego
TDS	Plik zawierający informacje symboliczne dla Turbo Debuggera.

Tabela 6. Typy plików

Zalety i wady C++Buildera

Zalety:

- Duża łatwość projektowania formularzy
- Szybkie komponentów wydajne tworzenie aplikacji różnego typu
- Duża elastyczność biblioteki VCL
- Rozbudowany i wygodny w użyciu mechanizm bazodanowy
- Wygoda w użyciu aplikacji internetowych
- Możliwość konwersji projektów z innych kompilatorów C++

Wśród najczęściej wymienianych niedostatków C++Buildera wymienia się:

- Możliwość utraty kontroli nad źle konstruowanym projektem
- Niepełna kompatybilność pomiędzy kolejnymi wersjami
- Duże problemy podczas konwersji do innych kompilatorów

9.3. Opis najważniejszych fragmentów programu

PLCSim jest dosyć rozbudowanym systemem i jest w nim wiele fragmentów, które pochłonęły wiele czasu i wysiłku. Na projekt składa się xxx plików źródłowych, xxx formatek. Do najważniejszych funkcji w programie należą:

- funkcja służąca do badania stanu logicznego wejścia bloku funkcyjnego

  bool Logic(int start, int szczebel, int linia,int parent);

  Parametry funkcji:

  a) start – kolumna w której rozpoczyna się analiza stanu logicznego
  b) szczebel – rung w którym rozpoczyna się analiza
  c) linia – linia w rungu
  d) parent – flaga zabezpieczająca poprawność wykonania funkcji

- funkcja arytmetyczna – pobiera i zapisuje w pamięci wartości zmodyfikowanych zmiennych

  bool ModInt(String IN1,String IN2,String Q);

  a) IN1 – adres pierwszej zmiennej wejściowej
  b) IN2 – adres drugiej zmiennej wejściowej
  c) Q – adres zmiennej wyjściowej

- funkcja służąca do wykonywania programu napisanego przez użytkownika, przeprowadza symulację poprzez uruchamianie poszczególnych funkcji (ADD, BIT POS, ROL, SHIFT LEFT itp.)
  analiza_programu();

9.4. Metody wykrywania błędów w programie

Podczas tworzenia programu pojawiały się błędy. Najprostsze do wychwycenia były błędy syntaktyczne. Wszystkie z nich zostały wychwytywane przez kompilator i szybko usuwane. Znacznie trudniejsze do zlokalizowania i zlikwidowania były błędy związane z źle opracowanymi i zaimplementowanymi algorytmami. Większość zastosowanych algorytmów była unikatowa, a co za tym idzie konieczne było ich opracowanie, zaimplementowanie i przetestowanie. Bardzo przydatne w tym procesie okazało się stosowanie narzędzi dostarczanych przez C++Bulidera. Znajdujący się tam debugger umożliwia dokładne prześledzenie programu, wygodny podgląd wartości zmiennych przez umieszczenie wskaźnika myszy na danej zmiennej. Możliwe jest również nadawanie wartości zmiennym, co także okazało się przydatne. Stosowane były również inne metody, co było wynikiem poznawania nowych możliwości kompilatora.

9.5. Analiza budowy pliku LM90

Program LM90 jest aplikacją stosowaną do programowania sterowników PLC. Działa on w środowisko DOS. Do prawidłowego otworzenia projektu niezbędny jest zestaw trzech plików. Mają one takie same nazwy dla wszystkich projektów, jednak każdy projekt umieszczony jest w odrębnym katalogu. Jeden z tych plików odpowiedzialny jest za udostępnianie projektu, drugi odpowiada za zapisywanie zmiennych przyporządkowanych cewkom, trzeci natomiast zawiera właściwe dane, w których pośród innych informacji zapisany jest program przeznaczony do wysłania do sterownika. Pliki te mają następujące nazwy:

- Lmfolder.30 – plik ten odpowiedzialny jest za udostępnienie projektu dla aplikacji. Ma on rozmiar 22 bajty.
- _main.dec – w pliku tym zapisywane są między innymi następujące informacje: nazwa projektu, suma kontrolna tego pliku, obszar danych w którym zaszyte są adresy przyporządkowywane cewkom. W zależności od typu cewki oraz przypisanego jej adresu stosowany jest określony sposób zapisu tego wyjścia. Jego rozmiar podczas analizy we wszystkich przypadkach wynosił 660 bajtów.

- _main.pdt – na budowę tego pliku składają się między innymi: kod programu przeznaczonego dla sterownika, informacja o rozmiarze pliku, informacje o adresach zmiennych przypisanych do poszczególnych elementów, informacje o ilości elementów tworzących program

Analiza była przeprowadzana metodą małych kroków. Tworzone były programy od najprostszych do bardziej skomplikowanych i na bieżąco były obserwowane zmiany w poszczególnych plikach. Różnice pomiędzy projektami pozwoliły stopniowo wydobywać informacje na temat ich struktury. Wymagało to wiele cierpliwości i wytrwałości, ponieważ w chwili rozpoczęcia pracy wiedza na ten temat była znikoma i niemożliwe było przewidzenie jakich technik do osiągnięcia tego celu używać. Poziom trudności podnosił fakt, iż konieczne było stworzenie narzędzi, których zadaniem była pomoc w analizowaniu problemu.

9.5.1. Budowa pliku _MAIN.DEC

Budowę tego pliku można przedstawić w formie tabeli.

1	44	47	51	59	85	660
Nagłówek pliku	Suma kontrolna	Dane stałe podczas analizy	Nazwa projektu	Dane kontrolne	Dane zawierające informacje o adresach przyporządkowanych cewkom	

Suma kontrolna jest liczona z zakresu od pięćdziesiątego pierwszego bajtu do końca pliku. Mieści się w trzech bajtach. Nie zastał dokładnie poznany mechanizm wyznaczania tych liczb. Aby w programie było możliwe eksportowanie pliku do formatu programu LM90 należało ustalić nazwę projektu. Jeszcze jednym ograniczeniem musiało być zawężenie przypisania adresów do cewek, ponieważ te dane także wpływają na procedurę liczenia sumy kontrolnej.

Nazwa projektu w tym pliku mieści się w ośmiu bajtach począwszy od pięćdziesiątego pierwszego licząc od początku pliku. Długość nazwy związana jest z ograniczeniem systemu DOS, w którym ten program działa. Zapisywana jest tylko nazwa katalogu, w którym zawarte są pliki programu.

Obszar danych kontrolnych podczas wszystkich testów był jednakowy. Nie jest on bez znaczenia, ponieważ tak jak nazwa projektu on również ma wpływ na procedurę liczenia sumy kontrolnej pliku.

Obszar danych zawierających informacje o adresach przyporządkowanych cewkom jest ciągiem bajtów, w którym w określony sposób są zapisywane dane świadczące o typie adresu i rodzaju zastosowanej cewki. Podzielony jest na podobszary.

85	213	469	533	660
Pierwszy obszar adresów typu Q	Pierwszy obszar adresów typu M	Drugi obszar adresów typu Q	Drugi obszar adresów typu M	

W pierwszym obszarze w każdym kolejnym bajcie są zapisywane kolejne cztery adresy począwszy od pierwszego aż do maksymalnego, natomiast w drugim obszarze w pojedynczym bajcie są zapisywane informacje o ośmiu kolejnych adresach. Każdy bajt z tego zakresu liczony jest jako suma logiczna liczb odpowiadających adresom, które mają być w tym bajcie zawarte.

Plik ten charakteryzuje się tym, że jeżeli zostanie otworzony plik projektu istniejącego już na dysku, wtedy obszar danych odpowiedzialny za adresy przyporządkowane cewkom nie jest resetowany, lecz zostaje modyfikowany nowymi wartościami.

9.5.2. Budowa pliku _MAIN.PDT

Plik ten jest kolejnym plikiem generowanym przez program. Zawiera on w sobie więcej informacji, niż _MAIN.DEC. Jego budowę można przedstawić następująco:

1	25	27	50	54	59	63	69	73	88
Nagłówek pliku	Dane odpowiedzialne za rozmiar pliku	Stałe dane	Dane zależne od długości pliku	Stałe dane	Rozmiar pliku	Stałe dane	Rozmiar pliku	Stałe dane	Zależne od długości pliku

92	104	107	108	111	128	131	164	223	długość-3
Stałe dane	Dane kontrolne	Stałe dane	Dane związane z długością pliku	Stałe dane	Dane związane z długością pliku	Stałe dane	Dane reprezentujące użyte adresy	Program użytkownika	Dane odpowiedzialne za prawidłowe zakończenie pliku

Plik ten jest bardziej rozbudowany. Po nagłówku występują dane, które zawierają bajty stałe w pliku podczas analizy, jak i takie, które są związane z długością pliku i liczbą elementów w nim występujących. Począwszy od 164 bajtu są zapisywane maksymalne adresy, jakie pojawiają się w programie. Na końcu pliku występują trzy bajty będące znacznikiem końce pliku.

W pliku tym zawarte są także informacje o jego rozmiarze. Dane dotyczące rozmiaru są dublowane (zapisywane są dwa razy w strukturze pliku). Jeśli wystąpi niezgodność pomiędzy tą informacją a rzeczywistym rozmiarem będzie generowany błąd.

Główną częścią pliku jest obszar danych zaczynający się od 223 bajtu. W nim są zawarte informacje na temat budowy programu użytkownika. Każdy element programu użytkownika jest reprezentowany za pomocą określonego ciągu bajtów charakterystycznego dla danego bloku funkcyjnego. Zawsze pierwszy jest unikatowy znacznik bloku, a następnie zapisywane są parametry, które odnoszą się do danego bloku. W przypadku pracy z kontaktami występują dodatkowe elementy mające za zadanie zachować prawidłową strukturę tej części programu. Są one reprezentowane w zależności od struktury rungu: może to być dodatkowy bajt w pliku lub odpowiednio zmodyfikowany identyfikator elementu.

9.5.3. Inne pliki projektu

Do prawidłowego otwarcia projektu niezbędne są trzy pliki. Dwa z nich zostały omówione, trzeci (Lmfolder.30) ma budowę niezmienną we wszystkich projektach i jest od odpowiedzialny za udostępnianie projektu do aplikacji. Istnieje także plik o nazwie _MAIN.LH1, który jest tworzony na podstawie plików _MAIN.PDT i _MAIN.DEC. Podczas zapisywania projektu programu LM90 jego obecność w procesie eksportu nie jest wymagana.

9.6. Testowanie programu i zbieranie informacji o błędach

Ze względu na fakt, iż tylko użytkowanie programu pozwala na wychwycenie wszystkich jego błędów, został on udostępniony szerokiemu gronu użytkowników poprzez umieszczenie go na stronie internetowej. Pozwoliło to na łatwiejsze testowanie i poprawianie błędów o których informacje dostarczane były przez użytkowników za pośrednictwem poczty elektronicznej. Pozwoliło to również ocenić zainteresowanie programem i uwzględnić potrzeby użytkowników. Sugestie przesyłane na pocztę pozwoliły na zwiększenie funkcjonalności aplikacji, a przez to dostosowanie jej do jeszcze szerszego grona użytkowników. Informacje te pozwoliły nie tylko usuwać i lokalizować błędy ale wytyczyły także kierunek rozwoju aplikacji. Co kilka dni, w zależności od intensywności zmian projektu na stronie pojawiała się nowa wersja symulatora. Ogólną niezawodność

programu podniósł fakt iż każda z funkcji została gruntownie przetestowana w czasie projektowania i implementacji.

9.7. Zasada działania programu

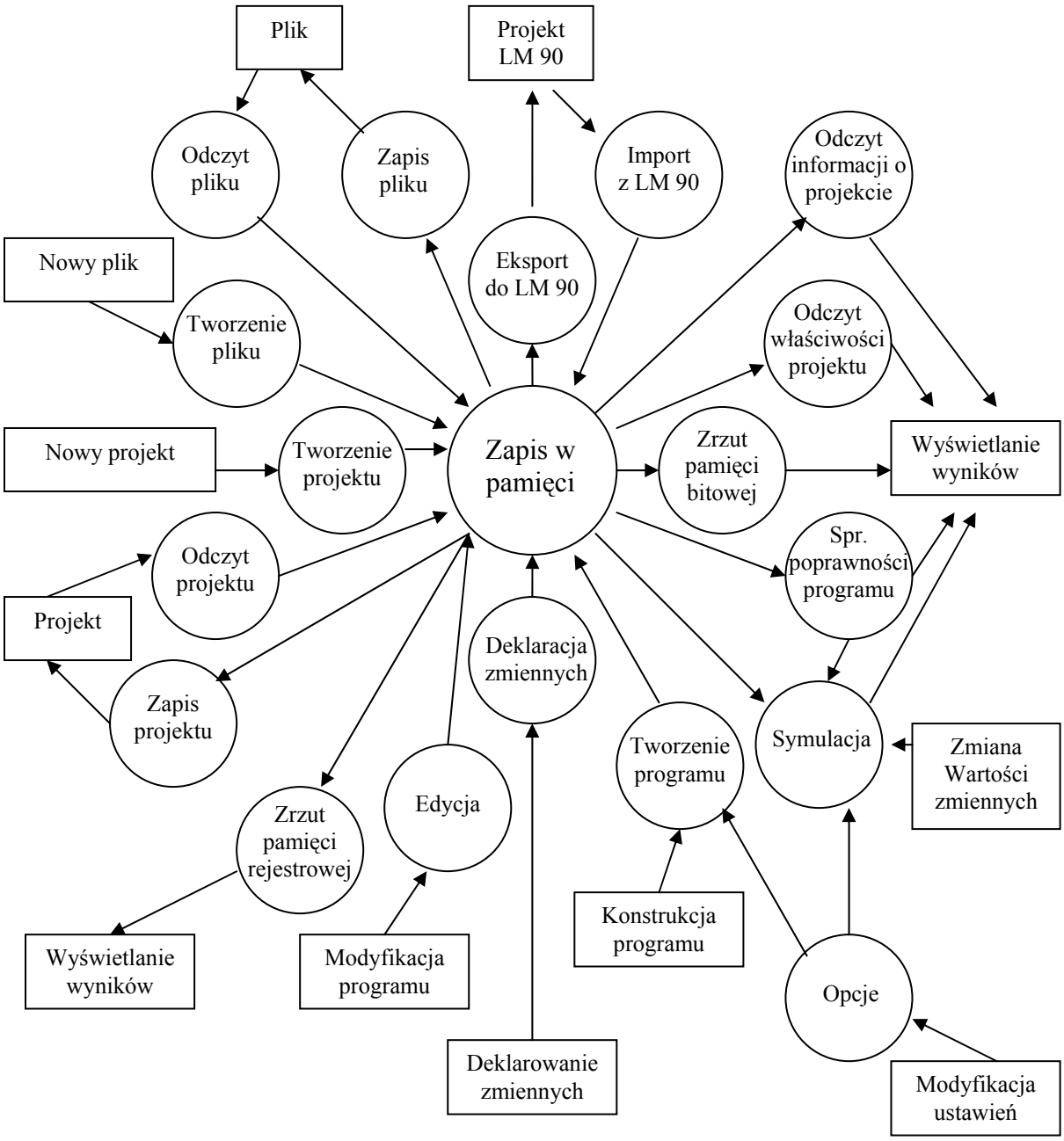
Program PLC Sim można podzielić na dziesięć zasadniczych części wykonujących następujące operacje:

- wejścia, wyjścia
- edycyjne
- tworzenia programu
- deklaracji i modyfikacji zmiennych
- testu poprawności programu
- symulacji
- wyświetlania wyników symulacji
- zmiany parametrów symulacji
- działania serwera DDE
- inne operacje

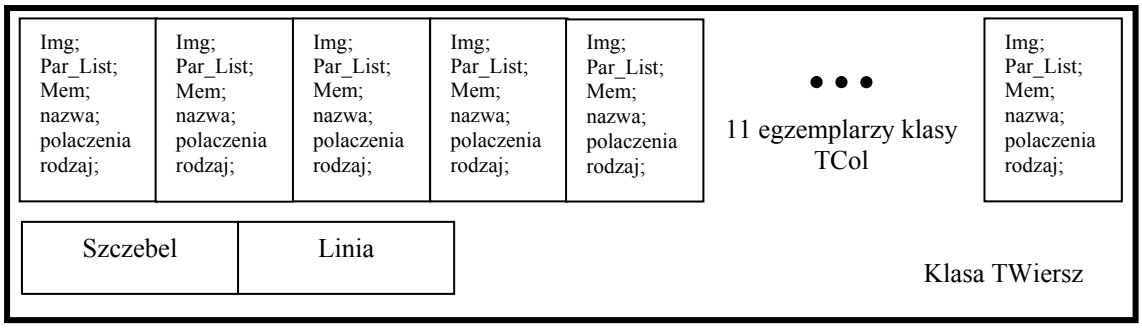
Ważniejsze z nich zostaną przedstawione w postaci diagramów oraz opisane.

Ogólna struktura programu

Poniżej przedstawiony jest ogólny diagram przepływu danych (DFD) w programie PLC Sim



Wszystkie operacje za wyjątkiem symulacji wykonywane są na pamięci programu, której budowa oparta jest na klasie TWiersz. Zawiera ona identyfikator rungu oraz linii w rungu. Ponadto każda klasa TWiersz zawiera w sobie informacje o jedenastu klasach TCol, w których znajdują się dane jednoznacznie określające zawartość każdej komórki wyświetlanej na siatce. Znajdujący się poniżej rysunek ilustruje budowę pojedynczego wiersza, w którym znajdują się wszystkie potrzebne dane.



Wiersze są tworzone dynamicznie, a w zależności od potrzeb ich ilość jest modyfikowana, jednak zgodnie z zasadami tworzenia programu w języku drabinkowym może ich być maksymalnie 8 w każdym rungu. Natomiast w wierszu może być maksymalnie 10 kolumn. Powodem zadeklarowania w klasie TWiersz 11 egzemplarzy klasy TCol jest konieczność przechowywania informacji o bocznym pasku widocznym z lewej strony siatki. Spowodowało to, iż poszczególne komórki (kolumny) siatki adresowane są do 1 do 10 i aby zachować zgodność adresowania został stworzony dodatkowy i niewyświetlany egzemplarz klasy TWiersz. Wypisywanie na ekran odbywa się w funkcji obsługi zdarzenia przerysowania siatki:

*StringGrid1DrawCell (TObject *Sender, int ACol, int ARow, TRect &Rect)*

Argumenty:

Rect - parametry płótna (canvas) siatki

ACol, ARow – kolumna i wiersz

Funkcja ta wywoływana cyklicznie dla kolejnych parametrów ARow i ACol (wiersz i kolumna) umożliwia przepisanie do odpowiedniej komórki przypisanych jej parametrów z klasy TCol. Wypisanie tekstu odbywa się przy pomocy funkcji:

StringGrid1->Canvas->TextOutA(Left, Top, wiersz[ARow].col[ACol].mem);

Argumenty:

Top, Left – pozycja tekstu

wiersz[ARow].col[ARow].mem – wartość wyświetlana.

Grafika natomiast wyświetlana jest przy pomocy funkcji:

StringGrid1->Canvas->CopyRect(Rect,wiersz[ARow].col[ACol].Img->Picture->Bitmap->Canvas,wiersz[ARow].col[ACol].Img->Canvas->ClipRect);

Argumenty:

Rect - parametry płótna (canvas) siatki

wiersz[ARow].col[ACol].Img->Picture->Bitmap->Canvas – kopiowane płótno

col[ACol].Img->Canvas->ClipRect- parametry kopiowanego płótna

Zapis parametrów do klas odbywa się przy pomocy funkcji:

*StringGrid1SelectCell(TObject *Sender, int ACol, int ARow)*

Argumenty:

ACol, ARow – kolumna i wiersz

W zależności od rodzaju wybranej funkcji z menu, wstawiane są odpowiednie parametry do klasy skojarzonej z komórką siatki, na której nastąpiło kliknięcie. Gdy funkcja zajmuje więcej niż jedną komórkę, odpowiednie parametry wstawiane są do wszystkich a w zmiennej „Mem" zapisywany jest numer kolejnej komórki dla danego bloku funkcyjnego w formacie:

Mem=AB

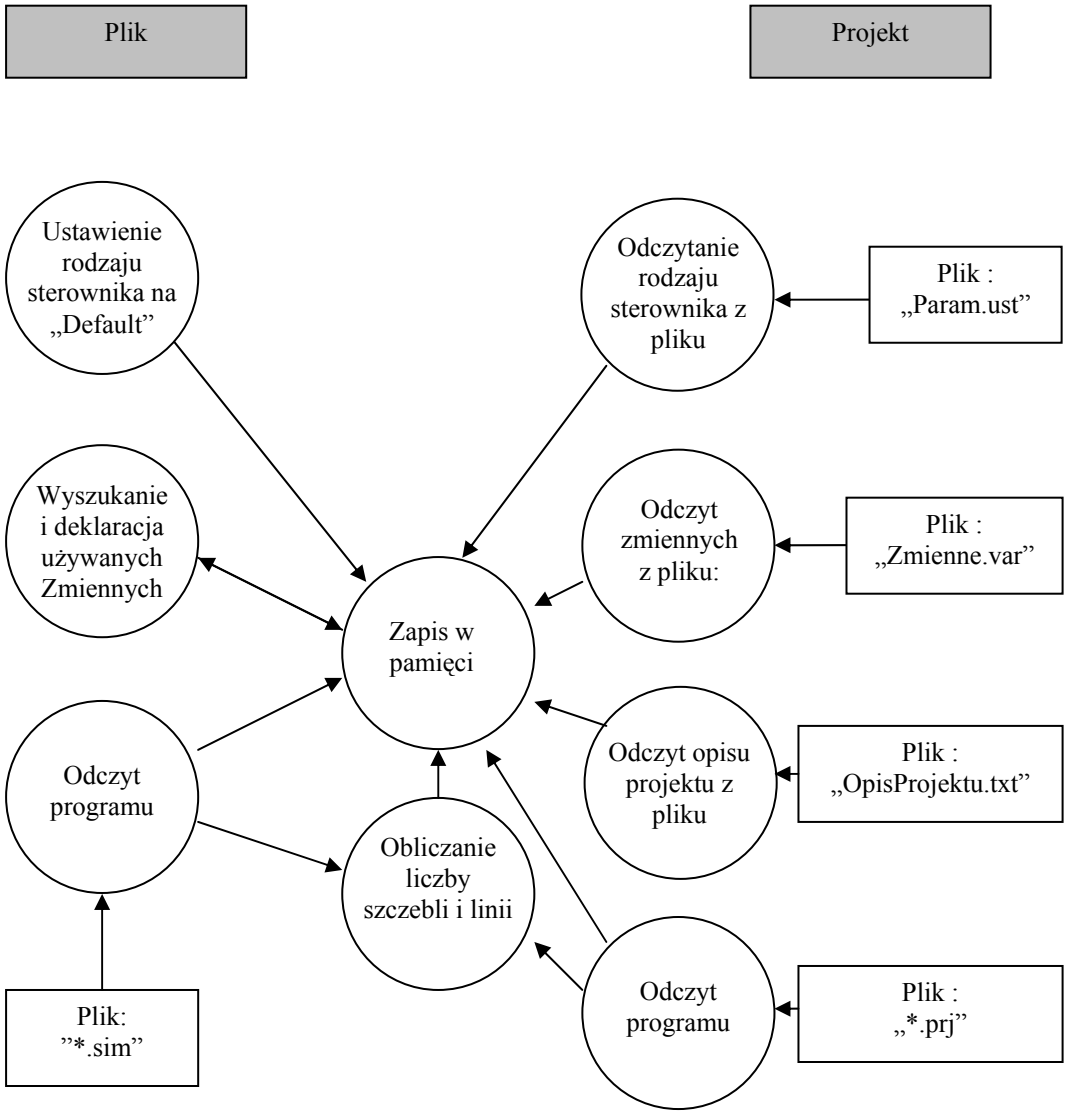
gdzie: A –numer komórki w bloku funkcyjnym

B –rozmiar bloku funkcyjnego

Przykład: blok ADD INT mieści się w trzech komórkach, co za tym idzie parametr B ma wartość 3 a parametr A kolejno 1, 2, 3. Zastosowanie tej operacji zaowocowało łatwością i zwiększeniem wygody programowania funkcji edycyjnych.

Operacje wejścia/wyjścia

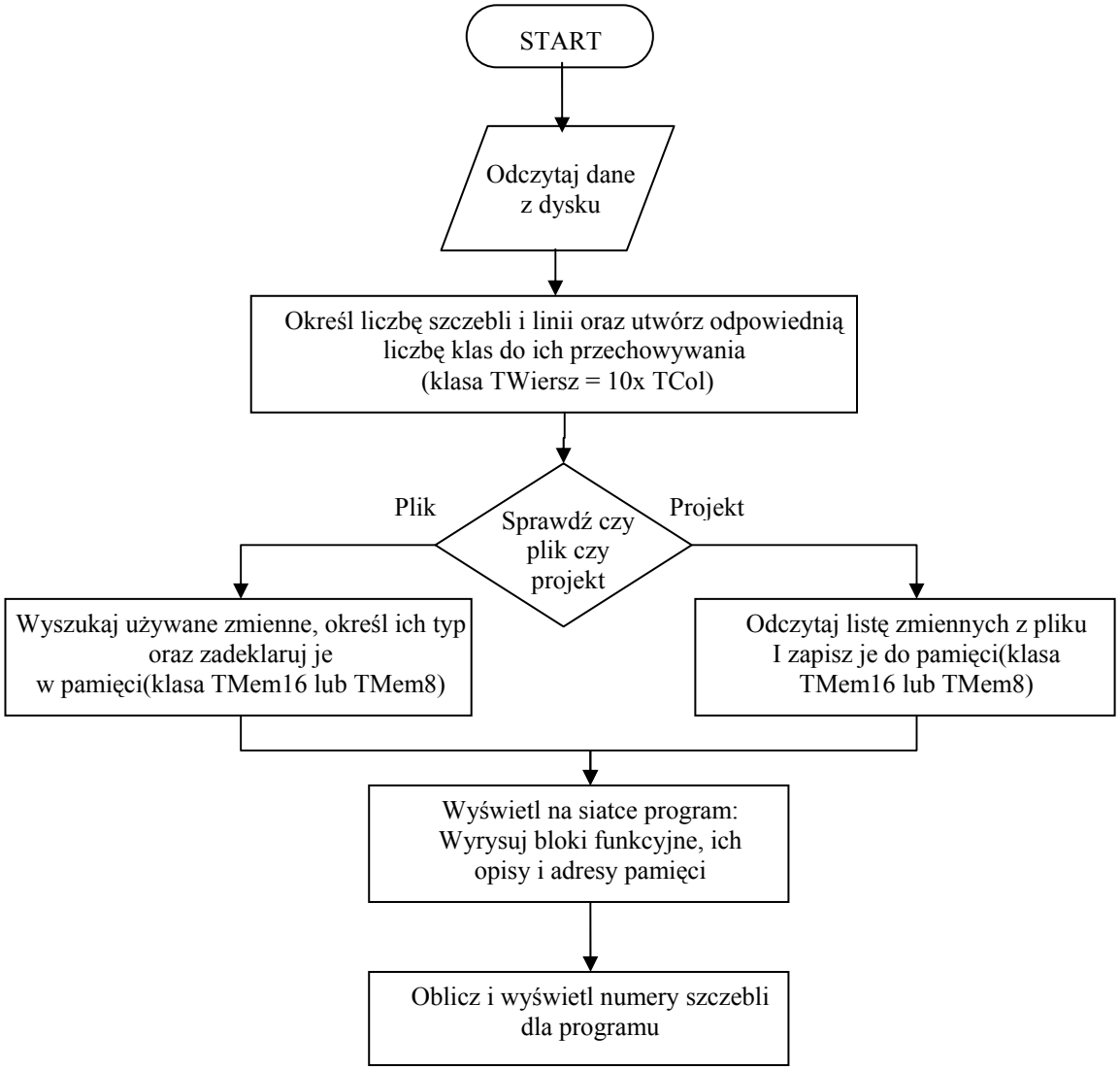
Odczyt z dysku podzielić można na dwa rodzaje: odczyt pliku lub projektu. Obrazuje to poniższy diagram DFD



Różnica pomiędzy plikiem a projektem jest znaczna. Polega ona na tym, że podczas korzystania z pliku wszystkie informacje dotyczące programu zapisane są w nim samym. Nie ma możliwości symulacji określonego typu sterownika a ilości wszystkich zmiennych ustawione są na wartości maksymalne (tryb default). W projekcie natomiast zarówno zmienne jak i sam program zapisane są w osobnych plikach. Dodatkowo dostępny jest plik z opisem projektu, istnieje możliwość wyboru rzeczywistego sterownika a przez to ograniczenie ilości zmiennych do wartości charakterystycznych dla danego typu jednostki centralnej. W trybie tym, po zaznaczeniu odpowiedniej opcji możliwe jest zadeklarowanie

zgodności z programem LM90 a przez to uzyskanie dostępu do opcji importu i eksportu do pliku w formacie tego programu.

Podczas zapisu programu do pamięci stosowany jest następujący algorytm



Nazwy plików wraz z ich opisami używane do przechowywania informacji o programie umieszczone są w tabeli:

Nazwa pliku lub rozszerzenia	Przeznaczenie	Katalog
*.sim	Zawiera program	Nie
*.prj	Zawiera program	Tak
Zmienne.var	Zawiera listę zadeklarowanych w projekcie zmiennych, ich nazwy, typy oraz krótkie opisy	Tak
OpisProjektu.txt	Zawiera krótki opis projektu	Tak
Param.ust	Zawiera informacje o rodzaju sterownika i trybie zgodności z LM90	Tak

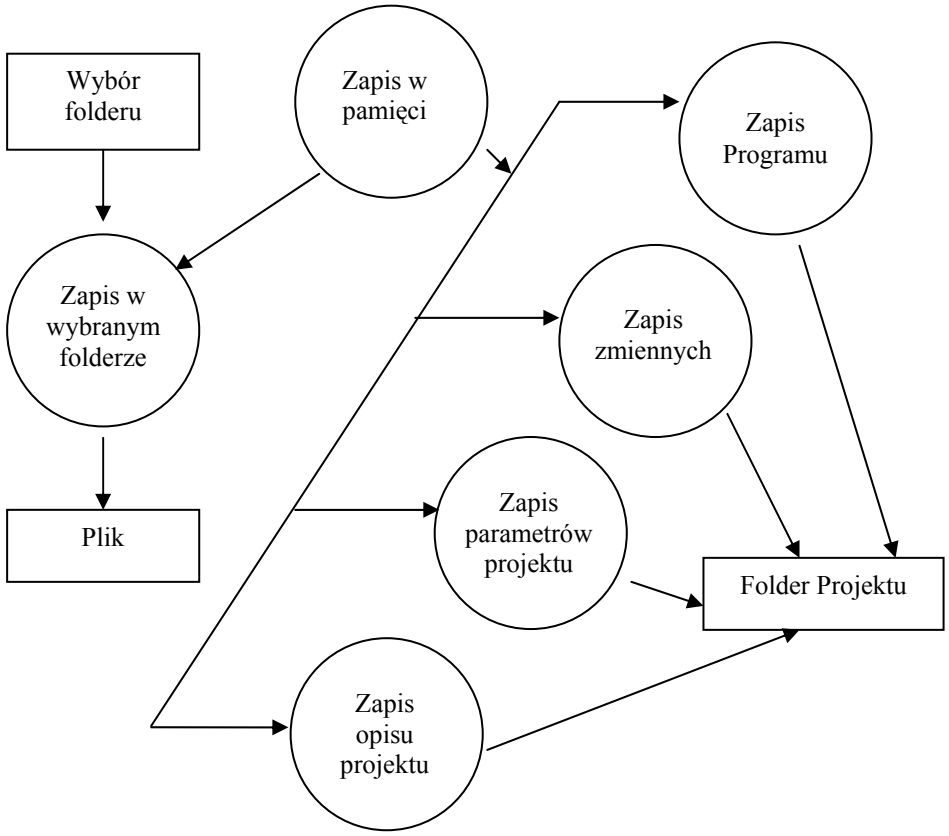
Tabela 7. Typy plików i ich opis.

Zapis programu

Przebieg zapisu programu:

- wybór sposobu zapisu (projekt czy plik)
- wybór folderu (w przypadku pliku)
- zapis parametrów dodatkowych(w przypadku projektu)

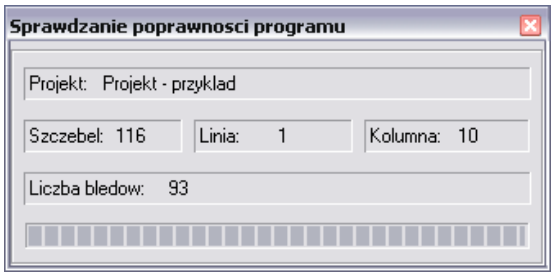
Szczegółowo przebieg zapisu przedstawiony jest na poniższym diagramie:



Istotne znaczenie ma również fakt, iż edytowany projekt można zapisać w postaci pliku, natomiast niemożliwe jest zapisanie pliku jako projekt. Decyzję o wyborze sposobu zapisu programu należy podjąć podczas rozpoczynania pracy nad nim. Opcja zapisu do pliku została stworzona w celu szybkiego przetestowania pewnej części projektu, bez zwracania uwagi na zgodność z rzeczywistymi sterownikami.

Sprawdzanie poprawności programu.

Funkcja ta uruchamiana jest przed symulacją w celu wykrycia błędów, które mogłyby spowodować nieprawidłowe wykonanie. Możliwe jest również uruchomienie jej w dowolnym momencie poprzez wybranie z menu głównego.



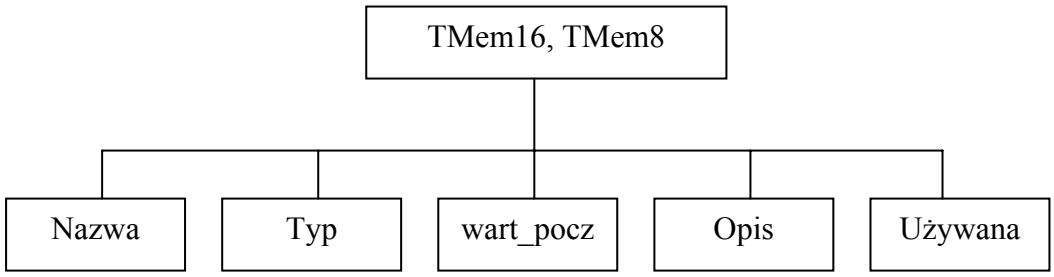
Wygląd okna funkcji

Zadanie tej funkcji polega na wykryciu błędów opisanych w rozdziale 4.1. Sprawdzane jest również czy wszystkim funkcjom przypisane są parametry oraz czy znajdują się one w odpowiednim miejscu. Sprawdzanie zakończone jest wyświetleniem raportu o przebiegu testowania, ilości i lokalizacji błędów.

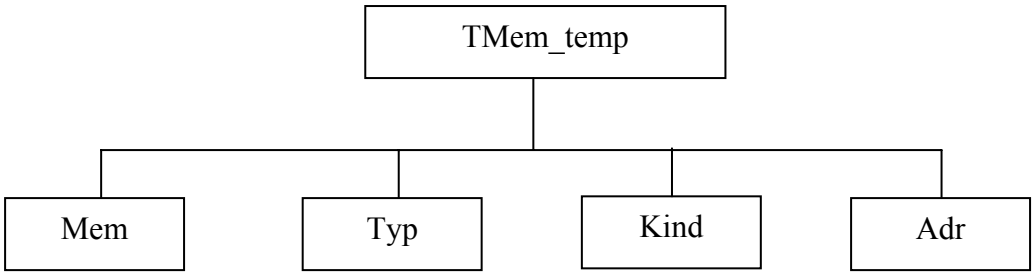
Symulacja programu.

Symulacja programu zawsze poprzedzona jest sprawdzaniem poprawności. Wyjątek stanowi przypadek, gdy program nie został zmodyfikowany od ostatniego sprawdzania. Informacja o tym wyświetlana jest na pasku statusu w oknie projektu:

W przypadku, gdy sprawdzanie poprawności programu przebiegnie pomyślnie rozpoczynany jest etap konwersji programu z klas TWiersz na postać zrozumiałą dla symulatora. Rozwiązanie takie okazało się konieczne, ponieważ zawierająca zmienne typu *String* klasa TWiersz była zbyt wolna i symulacja w oparciu o nią była nieekonomiczna. Z podobnych przyczyn została zastosowana konwersja klas przechowujących zmienne sterownika i klasa przechowująca dane tajmerów.



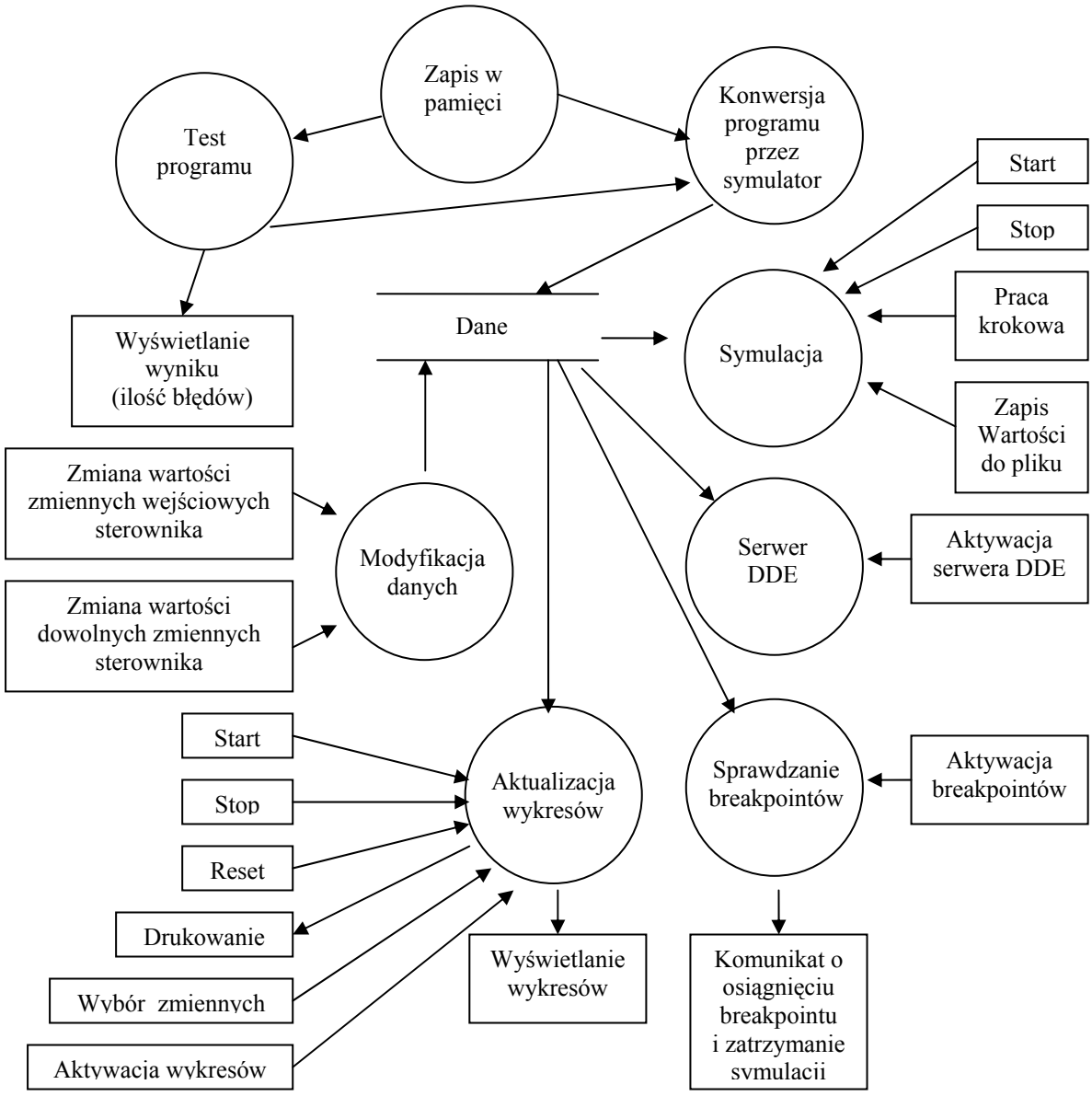
Budowa klasy przechowującej zmienne



Klasa przechowująca zmienne w symulatorze.

Dodatkowym atutem zastosowania tej konwersji jest możliwość przepisania tylko używanych w programie użytkownika zmiennych, co sygnalizuje zmienna *Używana* z klasy *TMem16 lub TMem8* .Adresy zmiennych z kasy *TMem_temp* zapisane są w postaci dwóch liczb typu *int* określających adres (Mem) oraz typ (Typ %AI=0, %R=1, %AQ=2, %I=3, %S=4, %Q=5, %M=6, %T=7, %G=8) Zmienna *Kind* służy do przechowywania informacji o typie zmiennej (np. int, dint), Natomiast w zmiennej *Adr* przechowywany jest adres w postaci stringu (przydatny podczas wyświetlania).

Przepływ danych podczas symulacji ilustruje poniższy diagram (DFD)



Zasadniczy proces symulacji polega na sekwencyjnym wykonywaniu kodu zawartego w pętli *for* poniższej funkcji:

```
void __fastcall sym::Execute()
{
        FILETIME CreationTime,ExitTime,KernelTime;
        union{
        LARGE_INTEGER iUT;
        FILETIME fUT;
        }UserTimeS, UserTimeE;
        ile=0;
        for(;;)
        {
        try
        {
```

```
        GetThreadTimes((HANDLE)Handle, &CreationTime,
&ExitTime,&KernelTime,&UserTimeS.fUT);
        Synchronize(DDE_IN);
        Synchronize(Skan_WejsciaF);
        ile++;
        TimeF();
        Symul->alaliza_programu();
        Synchronize(DaneF);
        Synchronize(DDE_OUT);
        Synchronize(EDIT_REF);
        Symul->pierwszy_skan=false;
        }
        catch (...)
        {
        Synchronize(Memo_komunikaty_err);
        }
        GetThreadTimes((HANDLE)Handle, &CreationTime,
&ExitTime,&KernelTime,&UserTimeE.fUT);
        int total=UserTimeE.iUT.QuadPart - UserTimeS.iUT.QuadPart;
        total /=10*1000;
        if(total<=10)
        Sleep(10-total);            //stabilizacja czasu
        else
        Synchronize(Memo_komunikaty);
        }
        Symul->Memo1->Lines->Add("Symulacja zakończona.");
}
```

Nazwa procedury	Realizowana funkcja
DDE_IN()	Odczyt danych z serwera DDE
SkanWejscia()	Odczyt pozycji suwaków i stanu pól wyboru
TimeF()	Inkrementacja zmiennych skojarzonych z tajmerami
Analiza programu();	Wykonanie programu
DaneF()	Aktualizacja wyjść i wypisywanie wartości zmiennych
DDE_OUT()	Zapis danych do serwera DDE
EDIT_REF()	Wyświetlanie liczby skanów symulatora

Tabela 8. Lista funkcji według kolejności ich wykonania.

W celu zapewnienia możliwości sterowania w programie funkcja ta została umiejscowiona w specjalnie do tego przystosowanym wątku, którego priorytet został podniesiony do najwyższego poziomu - *tpTimeCritical*. Dodatkowo zastosowana została pewnego rodzaju stabilizacja czasu do wartości 10ms. Funkcja ta realizowana jest poprzez procedurę *GetThreadTimes*, która bardzo precyzyjnie oblicza czas rozpoczęcia i zakończenia cyklu. Informacje te służą do podjęcia decyzji o wywołaniu dodatkowego opróżnienia i jego wartości. W przypadku przekroczenia wartości 10ms wyświetlany jest komunikat informujący o tym użytkownika.

Podczas symulacji możliwe jest sterowanie wykonaniem programu. Realizowane jest to przy pomocy funkcji sterujących wątkiem:

symu->Resume() - wznowienie wątku

symu->Suspend() - zatrzymanie wątku

Praca krokowa jest zrealizowana poprzez umieszczenie w funkcji obsługi przycisku (TButton) wywołań procedur dla jednego skanu. Ważny jest fakt, iż podczas pracy krokowej nie są uaktualniane tajmery. Zatrzymanie programu po przekroczeniu przez określoną zmienną zadanej wartości (breakpoint) również powoduje wywołanie funkcji:

symu->Suspend()

Dodatkowo wyświetlany jest stosowny komunikat informujący o tym użytkownika. Aby uaktywnić w programie sprawdzanie breakpointów należy w ustawieniach programu odznaczyć opcję *Full Debug* lub *Run i Breakpoints* znajdujące się w menu **Opcje** następnie w tabeli z zmiennymi w oknie *Symulacja* należy przy pomocy menu kontekstowego ustawić breakpoint. Wyłączenie tej opcji zmniejsza prawdopodobieństwo przekroczenia czasu skanu (10ms).

Serwer DDE

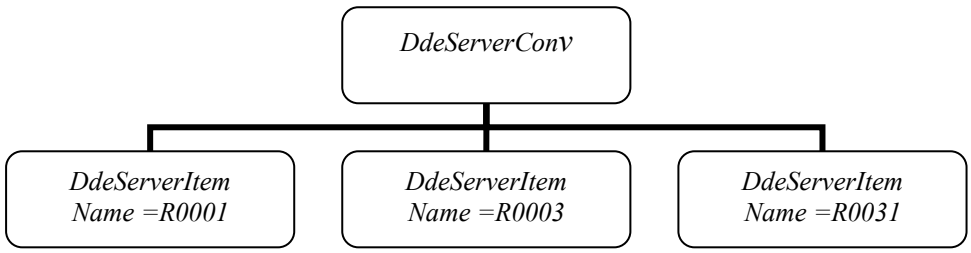
Serwer DDE zbudowany został przy użyciu komponentów udostępnianych przez C++Builder6.0. Są to

DdeServerConv -który poprzez właściwość *Name* ustala nazwę topicu połączenia.

DdeServerItem -który poprzez właściwość *Name* ustala nazwę item'u.

DdeServerItem -tworzone dynamicznie w zależności od ilości zmiennych zadeklarowanych w programie użytkownika. Jeżeli nazwa zmiennej odpowiada jej adresowi to nazwa *DdeServerItem* ustalana jest jako nazwa zmiennej bez znaku „%" (np. R0001, AQ0002). Odświeżanie serwera DDE polega na przepisywaniu wartości zmiennych do zmiennej *Text* należącej do klasy *DdeServerItem.*. Aby uaktywnić w programie sprawdzanie serwer DDE należy w ustawieniach programu odznaczyć opcję *Full Debug* lub *Run -> DDE.* Wyłączenie tej opcji zmniejsza prawdopodobieństwo przekroczenia czasu skanu (10ms).

Powyższy diagram obrazuje sposób tworzenia serwera DDE.



Głównym celem podczas pisania programu PLCSim obok ograniczenia zużywania nadmiernych ilości zasobów komputera była optymalizacja wykonania funkcji pod względem czasowym. Wynikało to z konieczności przeprowadzania dokładnych symulacji programów użytkownika. Sprawa optymalizacji okazała się kluczową podczas pierwszych testów programu zawierającego tajmery. Można było zauważyć znaczne różnice w czasie wykonania programu użytkownika przy ich zastosowaniu. Zastosowane optymalizacje i konwersja programu przez symulator pozwoliły w znacznym stopniu przyspieszyć jego wykonanie.

Zakończenie

Wiedza zdobyta zarówno podczas gromadzenia informacji dotyczących sterowników PLC jak i procesu projektowania oraz implantacji programu PlcSim pozwoliła nam na szersze spojrzenie na zagadnienia związane z budowaniem większych programów narzędziowych. Temat umożliwił nam sprawdzenie się w wielu trudnych sytuacjach wymagających samodzielnego opracowania skomplikowanych algorytmów. Pozwoliło to nam wykazać się pomysłowością oraz wyszukiwaniem najprostszych metod dążenia do celu.

Praca była o tyle ciekawsza, że wiele osób było zainteresowanych tematem i na bieżąco sprawdzało stronę internetową na której umieszczaliśmy kolejne wersje programu. Mamy nadzieję, że program przez nas napisany znajdzie swoje miejsce w praktyce, że pozwoli nowym użytkownikom przybliżyć problemy związane z programowaniem PLC. Jesteśmy zadowoleni z wyniku naszej pracy i mamy nadzieję, że zainteresowanie programem sprawi, że będą powstawały jego kolejne wersje.

Bibliografia

[1] Tadeusz Jegierski, Janusz Wyrwał, Jerzy Kasperczyk, Janusz Hajda, „Programowanie sterowników PLC”, Wydawnictwo Pracowni Komputerowej Jacka Skalmierskeigo, 1998

[2] Jarrod Hollingworth, Dan Butterfield, Bob Swart, Jamie Allsop, “C++ Builder 5. Vademecum profesjonalisty. Tom I”, Helion 2001

[3] Jarrod Hollingworth, Dan Butterfield, Bob Swart, Jamie Allsop, “C++ Builder 5. Vademecum profesjonalisty. Tom II”, Helion 2002

[4] www.plcs.pl

[5] www.astor.com.pl

[6] www.automatyka.pl

[7] www.gefanuc.com

[8] www.borland.nq.pl

[9] Pomoc programu VersaPro firmy GE-Fanuc.