Programowanie PLC zgodnie z normą IEC61131 – standardy i środowisko programistyczne

Praktycznie w każdej dziedzinie techniki istnieją pewne dobre zasady postępowania, do których warto się stosować tworząc niemal każdy projekt. Nie inaczej jest w przypadku programowania sterowników PLC. W temacie tym istotną rolę odgrywa norma IEC61131, której zasadom podporządkowali się praktycznie wszyscy producenci sterowników. Aby zademonstrować praktyczne aspekty programowania, posłużono się zintegrowanym środowiskiem Control FPWinPro 6.106 firmy Panasonic Electric Works, które pozwala w pełni wykorzystać założenia standardu.

Różnorodność dostępnych na rynku sterowników oraz sposobów ich programowania była przez lata przyczyną licznych problemów związanych z ich użytkowaniem. Każdy z producentów sterowników oferował unikalne oprogramowanie narzędziowe dla swojego produktu. Powodowało to szereg niedogodności, zarówno dla programistów, jak i użytkowników systemu. Po wielu latach starań powstała norma IEC61131, która definiuje cechy sterowników PLC, a w tym sposób ich programowania.

Norma IEC61131-3

Z punktu widzenia programisty najważniejsza jest część trzecia normy dotycząca języków programowania. Dzięki wprowadzonym zasadom ujednolicenia oprogramowania użytkownik może programować dowolny system PLC. W normie IEC61131 przedstawiono pięć języków programowania: IL (*Instruction List*), LD (*Ladder Diagram*), ST (*Structured Text*), FBD (*Function Block Diagram*) oraz SFC (*Sequential Function Chart*). Język IL jest podobny do asemblera, w którym dostępne są instrukcje logiczne, arytmetyczne oraz funkcje czasomierzy, liczników, itd. Język LD jest najczęściej wykorzystywanym ze względu na swoje podobieństwo do obwodów schematów elektrycznych. Dopuszcza się w nim wykorzystanie oprócz symboli styków i cewek również funkcji oraz bloków funkcjonalnych. W języku ST stosuje się strukturę programu zbliżoną do konstrukcji używanych w językach wysokiego poziomu, takich jak Pascal i C. W języku FBD schemat jest przedstawiony w postaci połączonych symboli bramek, funkcji oraz bloków funkcjonalnych. Zdefiniowano również sposób tworzenia struktury programu w postaci SFC, która umożliwia przedstawienie sterowania jako grafów zawierających kroki i warunki przejścia pomiędzy tymi krokami. Definicje kroków i warunków przejścia są programowane w jednym z czterech wymienionych wyżej języków. W normie IEC61131-3 przedstawiono następujące elementy języków programowania:

- typy danych określają strukturę, wielkość i zakres wartości danych stałych i zmiennych, potrzebnych do ich zapamiętania,
- jednostki organizacyjne oprogramowania, które są podstawowym elementem aplikacji użytkownika i składają się z funkcji, bloków funkcjonalnych oraz programów,
- elementy sekwencyjnego schematu funkcjonalnego, które stanowią kroki i wzajemnie powiązane przejścia za pomocą połączeń bezpośrednich,
- elementy konfiguracji, które określają zasoby, zadania, zmienne globalne oraz ścieżki dostępu.

Za pomocą konfiguracji definiuje się system sterownika PLC, który obejmuje wszystkie elementy oprogramowania. Zasobem jest programowy odpowiednik sprzętu, który realizuje funkcje przetwarzania sygnałów. Zadania określają własności wykonywanego programu lub FB, takie jak np. priorytet i sposób wykonywania. Konfiguracja może mieć więcej niż jeden zasób, który z kolei może zawierać więcej niż jeden program. Zadania kontrolują wykonywanie programów, które mogą zawierać elementy napisane we wcześniej wymienionych językach. Konfiguracje i zasoby mogą być uruchamiane i zatrzymywane za pomocą funkcji realizowanych przez system operacyjny, program lub poprzez interfejs operatora. Uruchomienie konfiguracji powinno inicjować zdefiniowane zmienne globalne, a następnie uruchomienie wszystkich należących do niej zasobów. Po uruchomieniu zasobu powinny zostać zainicjowane wszystkie występujące zmienne, a następnie powinny móc zadziałać wszystkie zadania zdefiniowane w danym zasobie.

Sposób wymiany danych w oprogramowaniu

W systemie sterowania wymiana danych pomiędzy elementami oprogramowania może się odbywać w następujące sposoby:

Pomiędzy elementami jednego programu – wartości zmiennych wewnątrz jednego programu są przekazywane pomiędzy różnymi elementami bezpośrednio poprzez połączenie wyjścia jednego elementu z wejściem drugiego.

Pomiędzy programami wewnątrz jednej konfiguracji – wartości zmiennych są przekazywane za pomocą zmiennych globalnych. W konfiguracji powinny być one zadeklarowane jako VAR_GLOBAL, a w programach jako zmienne zewnętrzne VAR_EXTERNAL.

Pomiędzy różnymi konfiguracjami – wartości zmiennych są przekazywane poprzez bloki komunikacyjne w tej samej lub innej konfiguracji, pomiędzy programami w sterowniku PLC i innym systemem sterowników.

Zalety stosowania normy IEC61131-3

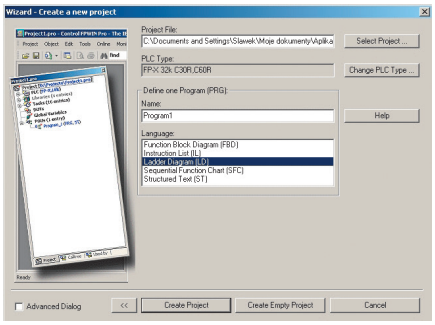
Wprowadzenie normy IEC do programowania sterowników PLC skutkuje licznymi korzyściami. Najważniejsze z nich to:

Uniwersalna biblioteka funkcji i bloków funkcjonalnych – oprogramowanie może być tworzone niezależnie od konkretnego systemu PLC, gdyż pakiety narzędziowe umożliwiają tworzenie uniwersalnych bibliotek zawierających funkcje i bloki funkcjonalne, których można później wielokrotnie używać w różnych aplikacjach. W celu ujednolicenia korzystania z bloków i funkcji w normie zdefiniowano standardowe bloki.

Uporządkowana struktura oprogramowania – elementy języków programowania umożliwiają tworzenie jasnej struktury oprogramowania, zaczynając od deklaracji zmiennych, danych, jednostek organizacyjnych oprogramowania. Język oprogramowania SFC umożliwia tworzenie skomplikowanych zadań sterowania w bardzo przejrzysty sposób dzięki strukturyzacji programów.

Wiele języków oprogramowania – w normie jest zdefiniowanych pięć różnych języków programowania, w tym jeden SFC, który umożliwia tworzenie programu strukturalnego.

Uproszczone struktury i typy danych – zmiana sposobu podejścia do adresowania



Rysunek 1. Widok okna konfigurującego aplikację użytkownika

zmiennych umożliwiła zmniejszenie zaangażowania programisty w procesie deklaracji zmiennych. Podczas deklaracji zmiennej, oprócz nazwy wprowadzany jest jej typ oraz wartość początkowa. Programista może również zdefiniować własne typy danych, które mogą być tablicami lub innymi złożonymi strukturami danych.

Oprogramowanie Control FPWinPro

W dalszej części artykułu przedstawione zostaną praktyczne przykłady programowania PLC w środowisku Control FPWinPro. Wersję demonstracyjną tego pakietu można pobrać ze strony internetowej firmy Panasonic. Wersja demo jest ograniczona tylko długością kodu kompilowanego programu.

Proces instalacji pakietu jest prosty. Po jej zakończeniu należy uruchomić oprogramowanie i utworzyć nowy projekt, po czym wprowadzić nazwę projektu, wybrać jednostkę centralną oraz podać nazwę programu i określić wykorzystywany język programowania. Sposób konfiguracji został przedstawiony na **rysunku 1**. Na **rysunku 2** pokazano widok środowiska oprogramowania przygotowanego do wprowadzania aplikacji.

Jednostki organizacyjne oprogramowania – POU

Jednostki organizacyjne oprogramowania (w skrócie POU) zawierają najmniejsze jednostki oprogramowania aplikacji, takie jak program (PRG), blok funkcjonalny (FB) lub funkcję (FUN). W porównaniu z konwencjonalnym programowaniem nie występuje tu pojęcie podprogramu. Pojawiają się za to takie określenia jak: program, który może wywoływać z danego POU blok funkcyjny FB lub funkcję FUN. Z Jednostki FB można wywołać tylko inny blok funkcyjny lub funkcję, natomiast funkcja może tylko wywołać inną funkcję. Nie jest możliwe wywołanie rekurencyjne, a więc POU nie może wywołać siebie pośrednio ani bezpośrednio.

Podstawowa różnica pomiędzy blokiem funkcyjnym FB i funkcją FUN polega na tym, że jeśli zostanie wywołana funkcja z tymi samymi parametrami wejściowymi to wartość wyjściowa zawsze będzie identyczna. W przypadku bloku funkcyjnego nie musi tak być. Jeśli ten sam blok FB z takimi samymi argumentami zostanie w programie wywołany kilkukrotnie, to za każdym razem może dać inny wynik wyjściowy. Wynika to z tego, że blok funkcyjny zawiera zmienne wewnętrzne, które przechowują informacje o jego stanie. Funkcja nie zawiera wewnętrznej informacji o stanie z poprzedniego wywołania. Wartości w bloku funkcyjnym są pamiętane pomiędzy kolejnymi wywołaniami funkcji, dlatego też każdy blok FB występujący w programie musi mieć własną nazwę oraz strukturę danych. W normie bloki te określane są instancjami. Jeśli dany blok jest użyty pięć razy to musi być zdefiniowanych pięć bloków FB tego typu.

Funkcja FUN może mieć dostęp do zmiennych zewnętrznych poprzez odniesienia: VAR_EXTERNAL, VAR_EXTERNAL_RETAIN i VAR_EXTERNAL_CONSTANT. Ponieważ funkcja nie zawiera informacji o stanie wewnętrznym, to zawsze daje na wyjściu VAR_IN_OUT lub VAR_OUTPUT tą samą wartość, zależną jedynie od wartości argumentów VAR_INPUT lub VAR_IN_OUT. W oprogramowaniu FPWinPro jest dostępna biblioteka zdefiniowanych funkcji. Są to funkcje zdefiniowane przez producenta sterowników oraz zgodne z normą IEC. Użytkownik może również tworzyć własną bibliotekę funkcji lub też korzystać z dodatkowych bibliotek dostarczanych przez innych użytkowników lub producentów urządzeń. W przypadku jeśli użytkownik będzie sam tworzył bibliotekę funkcji ma do wyboru cztery języki programowania (IL, LD, ST i FBD).

Każdy jednostka POU zawiera następujące elementy:

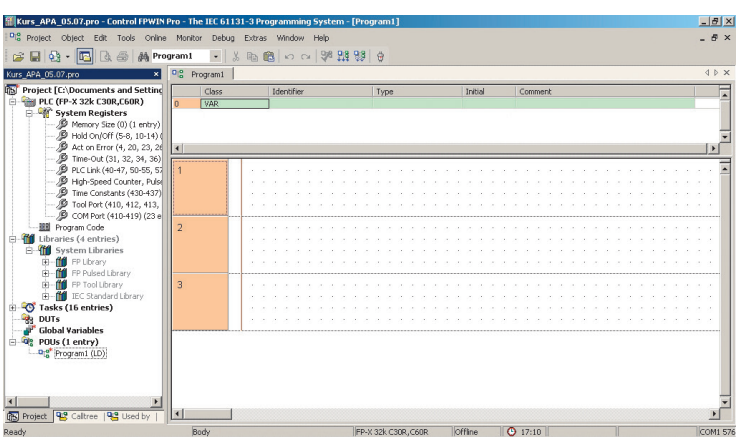
- typ i nazwę POU, które są określane w momencie wyboru nowego POU; dodatkowo w przypadku funkcji określony jest również typ funkcji,
- deklarację zmiennych, która zawiera wszystkie zmienne używane w danym POU,
- ciało *body* POU, które zawiera algorytm działania stworzony w odpowiednim języku programowania.

Na **rysunku 3** zamieszczono widok okna podczas tworzenia nowego programu, w którym można wybrać język programowania: IL, LD, ST, FBD lub SFC. Podczas tworzenia nowej funkcji lub bloku funkcyjnego można wybierać tylko z czterech języków (oprócz SFC), co zostało przedstawione na **rysunku 4**.

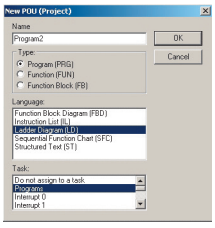
Deklaracja zmiennych

Zmienne służą do gromadzenia i przetwarzania informacji i są umieszczane w pamięci danych sterownika PLC. W POU wystarczy podać tylko nazwę symboliczną oraz informację o typie danej. Określenie typu danych jest konieczne, aby system zarezerwował odpowiedni obszar pamięci. Zmienne mogą być deklarowane jako globalne i są deklarowane poza POU, natomiast wewnątrz POU należy podać zmienne lokalne. Dodatkowo, przy deklaracji zmiennej można określić atrybuty, takie jak zapamiętanie zmiennej po zaniku zasilania.

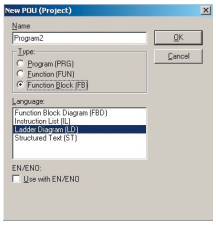
Przykład deklaracji zmiennych przedstawiono na **rysunku 5**. Należy wybrać jej klasę – ustalić czy jest to zmienna lokalna, czy globalna. W **tabeli 1** umieszczono słowa kluczowe określające typ zmiennej. Dodatkowo, za słowem kluczowym można umieścić atrybut informujący kompilator, że jest to zmienna pamiętana po wyłączeniu zasilania (RETAIN) lub że ma być ona traktowana jako stała (CONSTANT). Jeśli konkretna zmienna była dostępna dla innych aplikacji, to należy ją zadeklarować jako VAR_EXTERNAL. Musi ona również



Rysunek 2. Widok okna aplikacji przygotowanej do wprowadzania aplikacji



Rysunek 3. Widok okna podczas tworzenia nowego programu

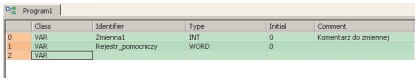


Rysunek 4. Widok okna podczas tworzenia nowego bloku funkcyjnego FB

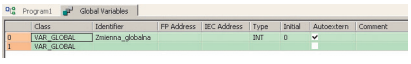
być też zadeklarowana w liście *Global Variables*. Aby uniknąć podwójnego wprowadzania zmiennej, warto najpierw wprowadzić ją w *Global Variables*, a następnie zaznaczyć pole *Autoextern* (**rysunek 6**). Wszelkie modyfikacje zmiennej należy zatwierdzić uaktualniając POU. Aby tego dokonać, trzeba ją zaznaczyć w liście deklaracji, kliknąć prawym przyciskiem myszy i wybrać z listy *Update Variables* lub z menu górnego *Extras* → *Update Variables*.

W polu *Identifier* wprowadza się symboliczną nazwę zmiennej. Sama nazwa może być dowolna z tym, że nie wolno używać znaku spacji do rozdzielenia wyrazów. Wobec tego stosowany jest znak podkreślenia „_" w roli separatora wyrazów. Kolumna *Type* określa typ danej i jest to pole obowiązkowe. Kompilator musi wiedzieć ile miejsca w pamięci zajmie dana zmienna. Na **rysunku 7** pokazano widok okna podczas wyboru typu dla zmiennej. Można wybrać elementarny typ, który został zdefiniowany w normie. W **tabeli 2** umieszczono typy danych dostępne w oprogramowaniu FPWinPro, zakresy ich wartości oraz liczbę bitów zajmowanych w pamięci. Taka koncepcja deklaracji zmiennych umożliwia sprawdzenie w trakcie kompilowania czy na zmiennej są wykonywane prawidłowe działania. Jeśli pojawią się błędy w deklaracji, to zostaną one przekazane programiście w oknie dialogowym. W polu *Initial* wprowadza się wartość początkową. Można pozostawić je puste, wówczas wartość początkowa będzie wynikała z typu danej. W większości przypadków dla zmiennych typu BOOL, INT, WORD, DINT, DWORD i REAL, wynosi ona 0. Zmienne typu STRING i związane z czasem przyjmują wartość różną od 0.

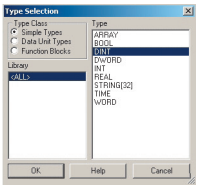
Deklaracja zmiennej reprezentowanej bezpośrednio odbywa się inaczej, niż zmiennych



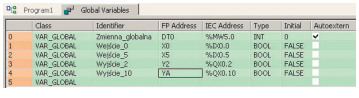
Rysunek 5. Przykład deklaracji zmiennych lokalnych



Rysunek 6. Przykład deklaracji zmiennych globalnych



Rysunek 7. Widok okna podczas wyboru typu danej



Rysunek 8. Przykład adresacji fizycznej zmiennych globalnych

Tabela 1. Wykaz słów kluczowych definiujących zmienne

Typ zmiennej	Opis
VAR	Zmienna deklarowana wewnątrz POU
VAR_EXTERNAL	Zmienna deklarowana wewnątrz POU, jest też zmienną zewnętrzną VAR_GLOBAL
VAR_GLOBAL	Deklaracja zmiennej globalnej
VAR_INPUT	Zmienna wejściowa do POU, nie może być zmieniana wewnątrz POU
VAR_IN_OUT	Zmienna wejściowa i wyjściowa do POU, może być zmieniana przez POU
VAR_OUTPUT	Zmienna wyjściowa z POU, wyprowadza dane i może być zmieniana przez POU

Tabela 2. Elementarne typy danych

Typ danej	Oznaczenie	Zakres wartości	Liczba bitów
Boolowski	BOOL	0 (FALSE) lub 1 (TRUE)	1 bit
Liczba całkowita	INT	-32768 do 32768	16 bitów
Liczba całkowita	DINT	-2147483648 do 2147483647	32 bity
Słowo	WORD	16#0000...16#FFFF	16 bitów
Podwójne słowo	DWORD	16#00000000...16#FFFFFFFF	32 bity
Ciąg znaków	STRING	1 do 255 bajtów (ASCII)	8 bitów na bajt
Czas trwania	TIME	T#0,00s do T#21 474 836,47s	32 bity
Liczba rzeczywista	REAL	$-1{,}175494\times10^{-38}\ldots-3{,}402823\times10^{38}$ i $1{,}175494\times10^{-38}\ldots3{,}402823\times10^{38}$	32 bity

lokalnych, gdyż jest konieczne podanie jej adresu fizycznego. Do wprowadzania zmiennych globalnych w FPWinPro służy okno *Global Variables*, gdzie oprócz nazwy, typu i wartości początkowej wprowadza się również adres fizyczny. W sterownikach z serii FP istnieje możliwość podawania adresów w formacie firmy Panasonic lub zgodnie z normą IEC. Jeśli zostanie wprowadzony adres w formacie sterowników FP, to automatycznie zostanie dopisany jego odpowiednik zgodny z normą IEC. Procedura ta działa również w drugą stronę.

Następujące przykłady prezentują zasady adresowania sterowników:

- %MW10.0 – oznacza słowo o adresie 10 w pamięci (w sterowniku FP – rejestr DT10),
- %MD5.102 – oznacza podwójne słowo o adresie 102 (DDT102),
- %MX0.1.1 – oznacza bit 1. w rejestrze 1 (bit pomocniczy R11),
- %IX5.0 – oznacza bit 50. na wejściu (wejście X50),
- %QX6.2 – oznacza bit 62. na wyjściu (wyjście Y62).

Tablice i struktury danych

Tablica to zbiór elementów tego samego typu. Dostęp do elementów tablicy jest możliwy poprzez indeks. Deklaracja zmiennej tablicowej sprowadza się do wprowadzenia nazwy tablicy (np. *Tablica1*), a następnie wyboru w polu *Type* typu *ARRAY* i określeniu typu elementów tablicy (np. *INT*). Po akceptacji w polu *Type* zostaje wprowadzone oznaczenie tablicy w postaci *ARRAY [0...2] OF INT*. Aby zaadresować trzeci element tablicy w programie należy wpisać *Tablica1[2]*. Tablice mogą być jedno-, dwu- i trzywymiarowe. Zmienna typu tablica nie może być używana w innej tablicy jako typ danych. Wykaz typów tablic i innych elementów wieloelementowych został przedstawiony w **tabeli 3**.

Struktury danych to natomiast zmienne, które składają się z elementów różnego typu. Przy adresowaniu kolejnych elementów struktury, jako separator stosuje się kropkę. Przykładowo: struktura o nazwie *Urzadzenie1* może zawierać elementy: *Stan1*, *Stan2*, *Stan3*. Aby zaadresować drugi element należy użyć składni *Urzadzenie1.Stan2*.

W oprogramowaniu FPWinPro struktury danych są określane jako *Data Units Type – DUT*. Występują ich dwa rodzaje: z elementami nakładającymi *overlapping elements* oraz z ciągła strukturą *non-overlapping*. Różnica pomiędzy nimi polega na tym, że dla pierwszego typu występuje nałożenie adresów fizycznych dla różnych typów elementarnych. Przykładowo, jeśli w strukturze została zadeklarowana zmienna typu *Integer*, a za nią następnie 16 zmiennych typu *BOOL*, to fizycznie w pamięci zostaną one umieszczone w tej samej komórce. W omawianym przypadku będzie to komórka pamięci, która umożliwia bezpośredni dostęp bitowy do rejestru. Dla struktury *non-overlapping* występuje adresowanie ciągłe, a więc dane będą umieszczane w kolejnych komórkach pamięci.

Sławomir Kacprzak

Artykuł ukazał się w EP+

Tabela 3. Zmienne wieloelementowe

Typ	Znaczenie	Rozmiar	Komentarz
ARRAY[...]OF...	Tablica elementów tego samego typu	1-255 bajtów	Tablica maksymalnie w trzech wymiarach
FB-Nazwa	Używany do tworzenia egzemplarza bloku funkcyjnego	Zmienny	Lokalny lub globalny blok funkcyjny
DUT-Nazwa	Egzemplarz struktury danych DUT	Zmienny	Globalny DUT