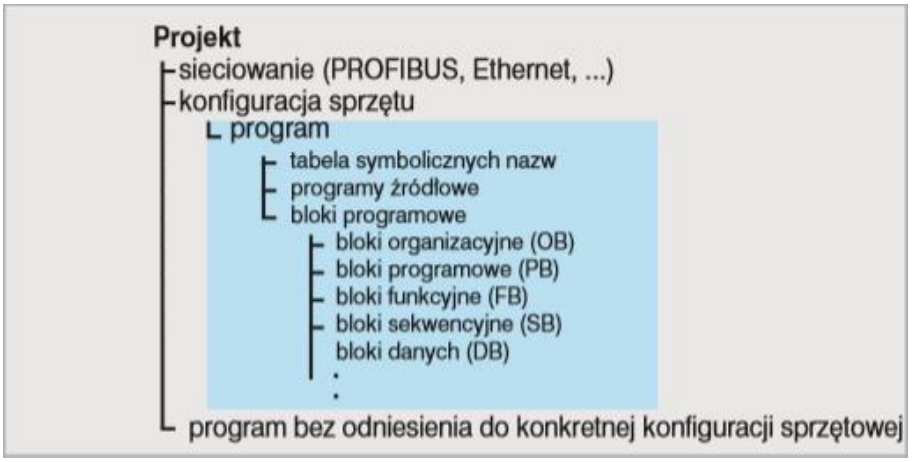
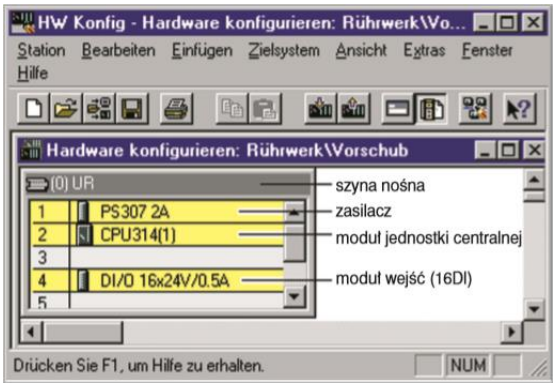
Budowa programu.

Zgodnie z normą IEC 61131 każde zadanie związane z automatyzacją procesu realizowane jest w formie projektu, w którym wstępnie określona jest **hierarchia zadań** w postaci **drzewa projektu** (rys. 1).



Rys. 1. Hierarchia zadań projektu, drzewo projektu.

Oprogramowanie sterowników PLC jest **zorientowane obiektowo**, dlatego też poszczególne części tworzonego programu sterującego użytkownika są obiektami, którym użytkownik przypisuje określone właściwości. Realizacja projektu rozpoczyna się od ustalenia powiązań sieciowych (ang. *network* = sieć) pomiędzy zespołem sterowników, urządzeniami wejść/wyjść procesowych oraz stacjami operatorskimi. Kolejnym zadaniem jest dobór sprzętu, jego skompletowanie i konfiguracja (rys. 2).

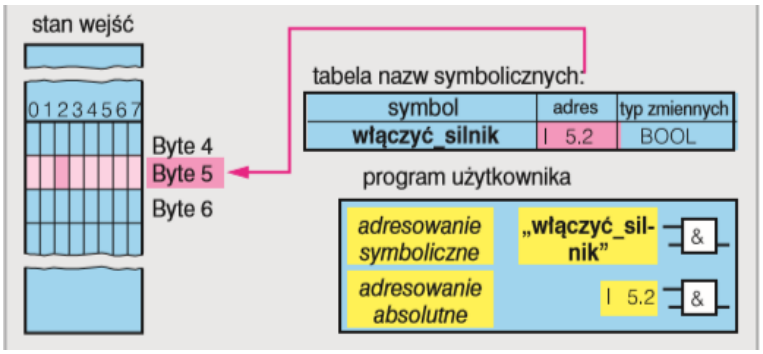


Rys. 2. Przykład konfigurowania sprzętu.

Definiowane są tu parametry zasilacza, modułu jednostki centralnej CPU oraz modułów dyskretnych i analogowych wejść i wyjść procesowych, składające się na sterownik PLC.

Kolejnym etapem jest wybór struktury programu sterującego użytkownika, który powinien zostać podzielony na oddzielne bloki programowe. Można wykorzystać w tym celu istniejące programy źródłowe i typowe bloki programowe, dostępne w oprogramowaniu systemowym (ang. *Firmware*), także te, które zostały napisane wcześniej, bez odniesienia do konkretnej struktury i konfiguracji sprzętowej.

Podczas programowania użytkownik może posługiwać się (rys. 3) **adresowaniem symbolicznym** zmiennych (np. „włączyć silnik”) lub **adresowaniem absolutnym** (np. I 5.2).



Rys. 3. Różnica pomiędzy adresowaniem absolutnym i symbolicznym.

Stosowanie symboli, w odróżnieniu od podobnych do siebie ciągów liter i cyfr, w zdecydowany sposób ułatwia umiejscowienie zmiennych w automatyzowanym procesie. Ponadto w przypadku konieczności przełączenia zmiennej na inne wejścia/wyjścia modułu, np. z powodu uszkodzenia, wiąże się z mniejszymi konsekwencjami. Wystarczy, bowiem zmienić adres zmiennej tylko w tablicy zmiennych, a nie w każdym miejscu programu, w którym ta zmienna została wykorzystana.

Rys. 4. Przykładowa tablica zmiennych grupująca ich symbole, adresy, typy i opisy.

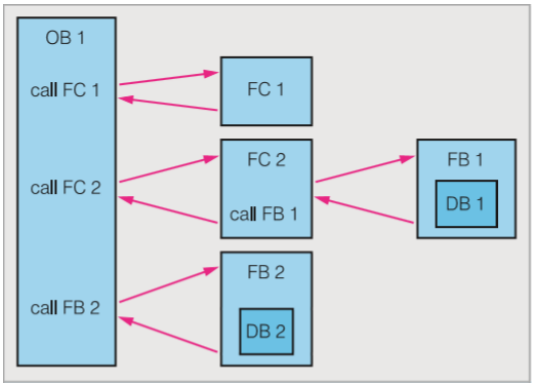
Wszystkie zmienne wykorzystywane w programie sterującym zgrupowane są w tablicy zmiennych (rys. 4). Użytkownik może je deklarować jako zmienne globalne lub zmienne lokalne. W pierwszym przypadku ich symbole są znane we wszystkich blokach programowych. W drugim symbol zmiennej jest rozpoznawany jedynie w bloku programowym, w którym została ona zdeklarowana. Odpowiednie deklaracje używanych zmiennych musi zawierać każdy blok programowy.

Podczas trwania danego cyklu działania sterownika procesor nie ma możliwości reagowania na jakiekolwiek zmiany stanu wejść, gdyż:

- stan sygnału wejściowego jest jednakowy podczas trwania cyklu działania sterownika. Zmiana dowolnego bitu w module wejść będzie uwzględniona dopiero na początku następnego cyklu,
- wielokrotna zmiana sygnału wyjściowego podczas trwania cyklu działania sterownika nie ma znaczenia, ponieważ wyjścia przełączane są dopiero na końcu cyklu i odpowiadają ostatniej zmianie dokonanej w tym cyklu.

Programowanie strukturalne, bloki programowe:

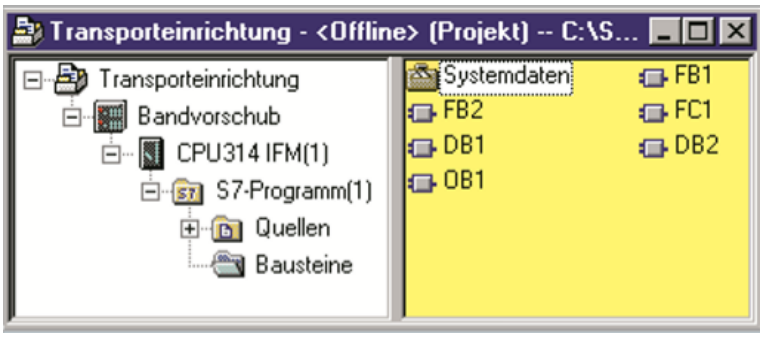
Prawidłowo przygotowany program sterujący powinien mieć strukturę modułową przedstawioną na rys. 5.



Rys. 5. Struktura modułowa programu sterującego.

Użytkownik ma do dyspozycji następujące bloki (moduły) programowe:

- **bloki organizacyjne OB.** (ang. ***O**rganisation **B**lock)* – interfejs programowy pomiędzy systemem operacyjnym sterownika, a programem sterującym użytkownika. Wyróżnia się blok organizacyjny (**OB1**), przeznaczony na główny program sterujący oraz zespół bloków organizacyjnych (**OBxx**, gdzie xx jest numerem bloku), wywoływanych cyklicznie przez system operacyjny po wystąpieniu określonych zdarzeń wyjątkowych (ang. *exception* – np. przerwania, zanik zasilania), jak również bloki wywoływane z programu użytkownika.
- **bloki programowe PB** (ang. ***P**rogram **B**lock)* – wydzielone z programu głównego, powtarzające się sekwencje operacji wykonywanych na tych samych parametrach, podanych przy wywołaniu bloku.
- **bloki funkcyjne FB** (ang. ***F**unction **B**lock)* – podobnie funkcjonujące do bloków PB, lecz stanowiące sekwencje operacji, wykonywane dla różnych wartości parametrów.
- **bloki sekwencyjne SB** (ang. ***S**equential **B**lock)* – szczególna forma bloków programowych, przeznaczonych do realizacji układów logicznych, sekwencyjnych, wyposażonych w elementy pamięci. Sekwencja zadań sterowania definiowana jest w postaci ciągu na przemian występujących fragmentów programów, zwanych krokami oraz warunków przejścia do kolejnych kroków, zwanych tranzycjami,
- **Bloki danych DB** (ang. ***D**ata **B**lock)* – wydzielone obszary pamięci użytkowej, przechowujące odpowiednio pogrupowane wartości zmiennych i parametrów wykorzystywanych w programie, np. zmierzona przez sensor wartość temperatury, wartość zadana licznika czasu lub zbiór nastaw regulatora PID (rys. 6).



Rys. 6. Przykładowe bloki programowe.

Podobnie jak w innych językach programowania, należy przestrzegać zasady kojarzenia ze sobą danych wyłącznie jednakowego typu.

Typ danych określony jest zarówno przez zakres wartości, jakie mogą one przyjmować, jak i zbiór operacji, które można na nich wykonać (tab. 1).

Tab. 1. Elementarne typy danych

Typ danych	Opis		Przykład
BOOL	bit	1 Bit	TRUE FALSE (1, 0)
BYTE	bajt (byte) liczba 8-bitowa w kodzie szesnastkowym	8 Bitów	 B#16#00 B#16#FF
WORD	słowo liczba 16–bitowa w kodzie szesnastkowym wartość licznika 3 dekady w kodzie BCD	16 Bitów	 W#16#0000 W#16#FFFF 2#0000_0000_0000_0000 2#1111_1111_1111_1111 C#000 C#999
INT	liczba stałoprzecinkowa	16 Bitów	–32768 +32767
DINT	liczba stałoprzecinkowa	32 Bity	L#–2 147 483 648 L#+2 147 483 647
REAL	liczba zmiennoprzecinkowa	32 Bity	+123.4567 +1.234567E+02
TIME	wartość czasu w formacie IEC	32 Bity	TIME#24d20h31m23s647ms TIME#24d20h31m23s647ms
S5TIME	wartość czasu	16 Bitów	S5T#0ms S5TIME#2h46m30s

Na przykład typ BOOL określa daną binarną, mającą postać pojedynczego bitu (wartość 0 lub 1) – może to być zmienna wejściowa I 5.2, zmienna wyjściowa Q 0.2 lub zmienna wewnętrzna, tzw. marker M 100.0 (zwana również flagą – F 100). Zmienne typu BYTE, WORD lub DWORD są ciągami bitów zawierających odpowiednio: 8 bitów (byte, bajt), 16 bitów (słowo, liczba całkowita w zakresie od – 32768 do +32767) lub 32 bity (słowo podwójne, liczba całkowita podwójnej precyzji w zakresie od -2147483648 do +2147483647).