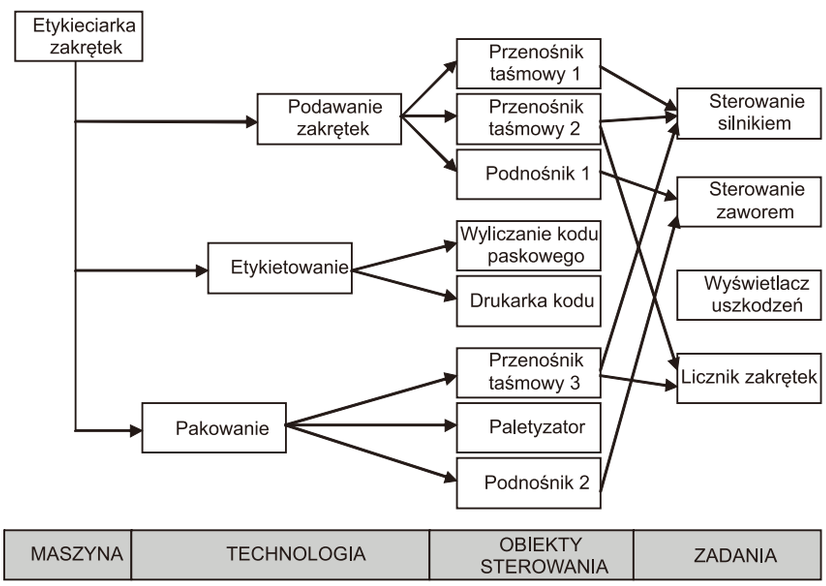
S7-1200 – podstawy PLC: teoria i praktyka

Miesiąc temu poruszyliśmy teoretyczne podstawy działania sterownika PLC. Teraz – bazując na sterownikach z rodziny S7-1200 – przedstawiamy nieco praktyki.

W linii technologicznej lub maszynie produkcyjnej często można wyróżnić fragmenty ściśle związane z określoną technologią. Na przykład dla technologa maszyna do etykietowania zakrętek realizuje następujące procesy technologiczne: podawania zakrętek, etykietowanie i pakowanie (**rysunek 1**). Projektant automatyki przydzieli do każdego procesu technologicznego obiekty, takie jak: przenośnik taśmowy, podnośnik itd. i niezależne zadania sterownicze. W obiektach występują te same zadania sterownicze, korzysta się często z tych samych algorytmów, np. algorytm załączania i wyłączania pracy silnika i wtedy celowe jest podejście strukturalne. W przytoczonym przykładzie na rysunku 1 będą to m.in.: sterowanie silnikiem, sterowanie zaworem, zliczanie zakrętek lub wyświetlanie informacji o uszkodzeniach. Ta forma strukturyzacji jest bazą do prostszego



Rysunek 1. Procesy realizowane przez maszynę

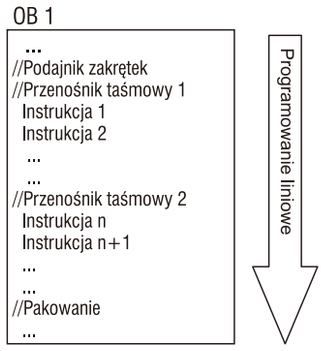
rozwiązania tego złożonego zadania. Program STEP 7 umożliwia podział programu użytkownika na zadania, ich programowanie i stosowanie bloków. Bloki w STEP 7 są częścią programu użytkownika i w nich można określać funkcje, struktury, cele, które pozwalają realizować wielostronną strukturyzację i redukują złożoność zadań sterowniczych. Użytkownik z tego tytułu ma takie korzyści:

- łatwiejsze i szybsze programowanie,
- lepszy wgląd w cały proces sterowania, łatwiejsze testowanie i utrzymanie w ruchu każdego składnika procesu,
- możliwość ponownego użycia raz napisanego oprogramowania, przez wygenerowanie bibliotek.

Do przytoczonego tu przykładu można napisać program instrukcja po instrukcji umieszczony w jednym bloku np. OB1. Będzie to tzw. program liniowy (**rysunek 2**). Ten typ programu ma jednak wiele niedogodności:

- jest nieprzejrzysty,
- stąd podatny na błędy,
- ma duży rozmiar plików,
- wymaga dużo pamięci operacyjnej.

Z tych względów nie jest on zalecany do stosowania. SIMATIC S7 zaleca stosować architekturę blokową. W przetwarzaniu blokowym bloki są wywoływane sukcesywnie i skojarzone są z technologią lub funkcjami, które mają realizować. Po zakończeniu przetwarzania ostatniego bloku (kilku zagnieżdżeniach) program automatycznie wraca do poprzednio wywoływanego adresu bloku. STEP 7 oferuje różne typy bloków



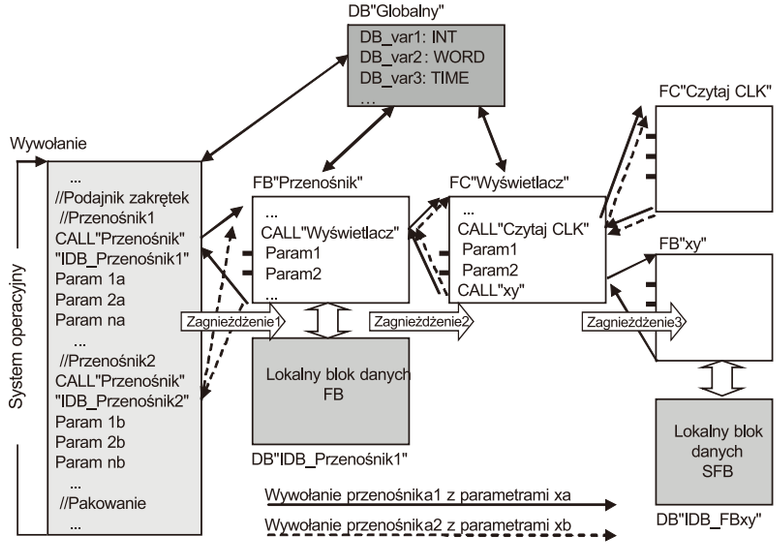
Rysunek 2. Program liniowy

do gromadzenia naszych podzadań w zależności od wymagań aplikacji. W systemie wykonawczym blok organizacyjny jest interfejsem pomiędzy systemem operacyjnym a programem użytkownika. Główny program jest zawsze umieszczany w cyklicznie wywoływanym bloku OB1. Na **rysunku 3** ilustrującym architekturę blokową, OB1 wywołuje technologiczną jednostkę pierwszego przenośnika taśmowego. To podzadanie jest realizowane jako blok funkcji FB. Ten blok może być elastycznie parametryzowany i ma swoją pamięć w formie lokalnego bloku danych. Ten blok danych jest generowany przez edytor programu w pierwszym wywołaniu FB z pierwszą instancją i jest dostępny tylko dla tego bloku FB. Wyjaśnić tu należy pojęcie instancji nazywanej też konkretem lub elementem. Instancja to nazwa i struktura danych wywoływana z bloków danych (lokalnych lub globalnych) dla konkretnego zadania. W przytoczonym tu przykładzie dla pierwszego przenośnika taśmowego, druga instancja będzie dotyczyć drugiego przenośnika. Zastosowana tu instrukcja wywołania CALL (ze sterowników S7-300) nie występuje w sterownikach S7-1200. Wywołanie w oprogramowaniu TIA polega na przeniesieniu myszką bloku FB lub FC z drzewa projektu do programu. Wynik obliczeń i operacji logicznych w bloku funkcji może być zależny od danych bieżących, jak również od wcześniejszych cykli w zależności od statusu wywoływanych parametrów. Z bloku funkcji mogą być wywoływane kolejne bloki. W przykładzie na rysunku 3 mamy zadanie wyświetlenia i przetworzenia sygnału błędu z możliwością ponownego wykorzystania. Funkcja (FC) jest stosowana do programowania często powtarzających się operacji, Może być parametryzowana. Nie ma żadnego elementu pamiętającego i o stanie wyjścia decydujące stan wejść i wyjść w danej chwili. Dane użytkownika są strukturyzowane i zakodowane i mogą być umieszczone w blokach danych. Są dwa typy bloków danych: a) globalne bloki danych, które mogą być używane jako samodzielne bloki danych przez wszystkie inne bloki, b) lokalne bloki danych z instancjami, które są zawsze przyporządkowane do określonego bloku funkcji i zawierają tylko dane lokalne dotyczące tego bloku. W przykładzie z rysunku 3 po zakończeniu obsługi pierwszego przenośnika (bloki FB, FC) program przejdzie do obsługi drugiego przenośnika w tych samych blokach, ale z instancją dla drugiego przenośnika, czyli nowymi parametrami I/O. Kolejne wywołanie nowego bloku z poprzedniego bloku powoduje zagnieżdżenie (*nesting*). Sterowniki mają ograniczoną głębokość zagnieżdżania, tutaj to zagnieżdżenie wynosi 3. Ta droga tworzenia programu użytkownika ilustruje, jak dzięki architekturze blokowej upraszcza się programowanie, staje się ono bardziej przejrzyste, zużywa mniej pamięci i można je ponownie wykorzystać w innych projektach. Prosty program z **rysunku 4** ilustruje, jak blok funkcji FB komunikuje się z różnymi obszarami pamięci przez parametryzowane interfejsy. Przedstawiony diagram drabinkowy wykorzystuje prosty, często stosowany blok funkcji do dwustanowego sterowania silnikiem. Silnik jest załączany i wyłączany dwoma przyciskami: ZAŁ i WYŁ. Dodatkowo stan załączenia silnika jest sygnalizowany migającą lampką. Blok może być wykorzystany wielokrotnie do innych takich samych silników (z innymi instancjami). Określony silnik jest wybrany tylko przez przyporządkowany mu interfejs. Program w bloku FB i jego funkcje pozostają niezmienione. Wejścia są adresowane przez obszar pamięci obrazu procesu wejściowego i połączone z parametrami bloku. Sygnał zezwolenia dla wszystkich silników jest umieszczony w bloku danych globalnych i również jest podłączony do parametrów wejściowych. Migający sygnał jest automatycznie przyporządkowany do bitu w pamięci globalnej i również przyłączony do parametrów wyjściowych. Sterowany stycznik silnika i migająca lampka są połączone z adresami w obrazie procesu wyjść przez parametry wyjściowe. Stan elementu pamiętającego (SR) również musi być obsługiwany, dlatego należy go zadeklarować w sekcji deklaracyjnej jako zmienną statyczną. W bloku FB wszystkie parametry, zmienne chwilowe i statyczne są zapamiętywane w bloku danych z instancjami. Każdy program składa się z instrukcji, które obok skróconej nazwy mają co najmniej jeden operand (zmienną lub stałą). Operandy programu użytkownika stosowane w bloku są załączane wyłącznie przy użyciu nazw symbolicznych parametrów i statycznych zmiennych sekcji deklaratywnej do tego bloku. Funkcjonalnie blok FB jest cały czas gotowy do sterowania, a fizycznie blok istnieje tylko jeden raz w pamięci CPU, ale może być wywoływany w programie wiele razy z użyciem innych danych z bloków DB z instancjami.

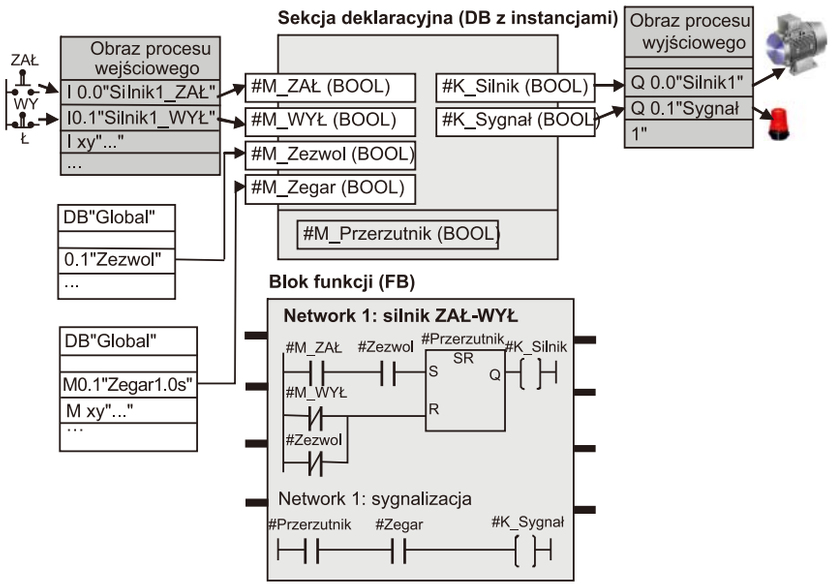
Wykonywanie programu użytkownika rozpoczyna się od jednego lub więcej opcjonalnych startowych (rozruchowych) bloków organizacyjnych (OB), które są wykonywane jednokrotnie po wejściu CPU w tryb RUN, a po nich jest wykonywany cyklicznie jeden lub więcej pozostałych OB cyklu programu. OB może być również skojarzony z przerwaniem wywołanym albo standardowym zdarzeniem, albo wykrytym błędem i jest wykonywany za każdym razem, kiedy wystąpi odpowiednie standardowe zdarzenie lub błąd. Funkcja (FC) lub blok funkcji (FB) jest blokiem kodu programu, który może być wywołany z OB albo innej FC lub FB, przy czym głębokość zagnieżdżenia takich wywołań wynosi:

- 16 z cyklu programu lub OB startowego,
- 4 z OB przerwania opóźnienia, przerwania cyklicznego, przerwania sprzętowego,
- przerwania błędu czasu lub przerwania błędu diagnostycznego.

Funkcje (FC) nie są skojarzone z jakimś szczególnym blokiem danych (DB), podczas gdy bloki funkcji (FB) są bezpośrednio związane z DB i wykorzystują ten DB do przekazywania parametrów i przechowywania



Rysunek 3. Architektura blokowa



Rysunek 4. Przykładowy program demonstrujący komunikację z pamięcią

tymczasowych wartości i wyników. Rozmiar programu użytkownika, danych i konfiguracji jest ograniczony wielkością dostępnej pamięci ładowania CPU. Liczba obsługiwanych bloków nie jest ograniczona. Jedynym ograniczeniem jest wielkość pamięci. Każdy cykl obejmuje zapisywanie stanu wyjść, odczytywanie stanu wejść, wykonanie instrukcji programu użytkownika oraz wykonanie obsługi systemu lub przetwarzania w tle. Taki cykl jest nazywany cyklem programu.

Płytka sygnałowa (SB), moduły rozszerzeń (SM) i moduły komunikacyjne (CM i CB) są wykrywane i rejestrowane tylko w trakcie włączenia zasilania. Wkładanie i wyjmowanie płytki sygnałowej, modułów rozszerzeń i modułów komunikacyjnych pod napięciem nie jest obsługiwane. Jedynym wyjątkiem jest karta pamięci SIMATIC Memory Card, która może być wkładana i wyjmowana wtedy, kiedy CPU jest zasilane.

W warunkach standardowych, wszystkie punkty I/O analogowe i cyfrowe są uaktualniane synchronicznie z cyklem programu wykorzystującym obszar pamięci wewnętrznej zwanym obrazem procesu. Obraz procesu zawiera chwilowy stan fizyczny wejść i wyjść (fizycznych punktów I/O CPU, płytki sygnałowej i modułów rozszerzeń).

Tuż przed wykonaniem programu użytkownika CPU odczytuje stan fizycznych wejść i zapamiętuje te wartości wejściowe w obszarze wejściowym pamięci obrazu procesu. Dzięki temu uzyskuje się pewność, że te dane pozostają stałe w trakcie wykonywania instrukcji użytkownika. CPU wykonuje zadania określone instrukcjami użytkownika i – nie zmieniając stanu fizycznych wyjść – uaktualnia wartości wyjściowe w obszarze wyjściowym pamięci obrazu procesu.

Po wykonaniu programu użytkownika, CPU przepisuje stany wyjść z obszaru wyjściowego pamięci obrazu procesu do fizycznych wyjść. Ten proces zapewnia zachowanie spójności logiki poprzez wykonywanie w danym cyklu instrukcji użytkownika i zapobiega zmianom stanu fizycznych punktów wyjściowych, w wyniku mogących występować wielokrotnie w cyklu zmian w obszarze wyjściowym pamięci obrazu procesu.

Użytkownik może zmienić standardowe działanie modułu, wyłączając to automatyczne uaktualnianie stanu punktów wyjściowych. Można również bezpośrednio odczytywać i zapisywać cyfrowe i analogowe stany I/O modułów podczas wykonywania instrukcji. Bezpośredni odczyt stanu fizycznych wejść nie uaktualnia obszaru wejściowego pamięci obrazu procesu. Bezpośredni zapis stanu do fizycznych wyjść uaktualnia zarówno obszar wyjściowy pamięci obrazu procesu, jak i stan fizycznych punktów wyjściowych.

Tomasz Starak