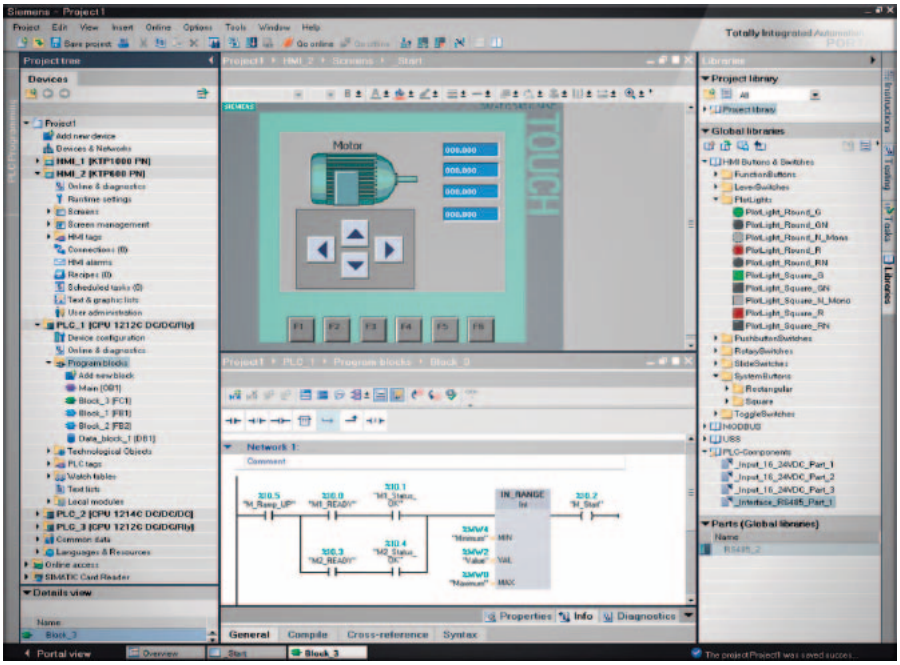
Sterowniki S7-1200
Podstawy przygotowania programów

S7-1200 to nowa, pod wieloma względami przełomowa, rodzina sterowników PLC firmy Siemens. Ich podstawowe możliwości przedstawiliśmy miesiąc temu, teraz skupimy się na podstawach ich programowania.

Sposobów przygotowywania programów dla sterowników PLC jest niemal tyle ilu jest automatyków z nich korzystających. Niezależnie od stylu, programiści korzystają z jednolitego zestawu bloków i funkcji, spośród których najciekawsze przedstawiamy w artykule.

Wybór języka programowania

Pisząc program dla sterownika S7-1200 użytkownik ma możliwość wyboru języka programowania i korzystania albo z LAD (*LADder logic*) albo FBD (*Function Block Diagram*). Obydwa języki programowania są obsługiwane przez nowe, zintegrowane środowisko projektowe TIA (*Totally Integrated Automation*), wprowadzone na rynek przez firmę Siemens wraz ze sterownikami S7-1200 (widok okna pokazano na **rys. 1**).



Rys. 1.

Język programowania LAD

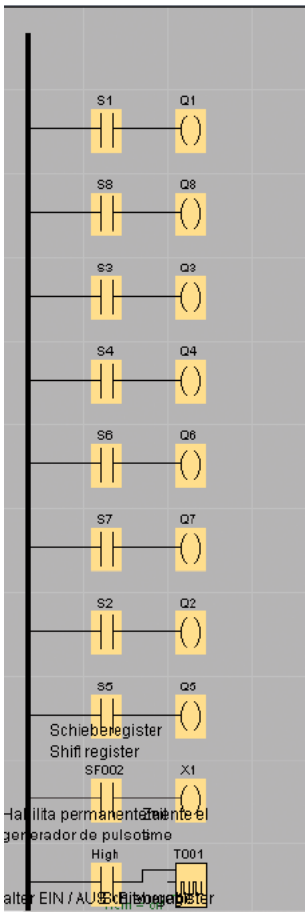
LAD jest graficznym językiem programowania (przykład programu LAD pokazano na **rys. 2**), opartym na schematach obwodów. LAD umożliwia stosowanie instrukcji dla różnych funkcji, takich jak arytmetyczne, czasowe, zliczające oraz związane z ruchem. Podczas tworzenia sieci LAD, należy kierować się następującymi zasadami:

- Każda sieć LAD musi kończyć się cewką lub instrukcją ramkową. Nie należy zakańczać sieci instrukcjami porównania lub wykrywania zboczy (dodatnich lub ujemnych).
- Nie wolno tworzyć gałęzi, w której może nastąpić przepływ mocy w odwrotnym kierunku.
- Nie wolno tworzyć gałęzi, która mogłaby spowodować zwarcie.

W celu zaprojektowania logiki dla złożonych operacji, na schemacie można umieszczać gałęzie dla utworzenia logiki obwodów równoległych. Gałęzie równoległe są na początku rozwarte lub bezpośrednio łączone do zasilania. Od strony zakończeń gałęzi występują połączenia kończące.

Język programowania FBD

Podobnie jak LAD, również FBD jest graficznym językiem programowania (przykład programu FBD pokazano na **rys. 3**). Repre-



Rys. 2.

zentacja logiki jest w nim oparta na graficznych symbolach logicznych stosowanych w algebrze Boole'a.

Zarówno w LDA, jak i w FBD, dla niektórych instrukcji ramkowych stosuje się parametr „power flow" – „zasilanie" (EN i ENO). Niektóre instrukcje (m.in. arytmetyczne) wykorzystują parametry EN i ENO. Te parametry są związane z podawaniem zasilania i określają czy instrukcja jest wykonywana podczas cyklu programu.

- EN (*Enable In*) jest wejściem boolowskim dla ramek W LAD i FBD. Jeżeli instrukcja ramkowa ma być wykonana, to na jej wejściu musi wystąpić zasilanie (EN = 1).
- ENO (*Enable Out*) jest wyjściem boolowskim dla ramek w LAD i FBD. Jeżeli wejście EN bloku LAD jest bezpośrednio połączone do szyny zasilania z lewej strony, to wtedy instrukcja ramkowa zawsze będzie wykonana. Jeżeli na wejściu EN bloku jest zasilanie i funkcje bloku są wykonane bez błędów, to ENO przekazuje zasilanie (ENO = 1) do następnego elementu. Jeżeli zostanie wykryty błąd podczas wykonywania instrukcji z bloku, to przekazanie zasilania jest zatrzymywane (ENO = 0) na tej ramce z instrukcjami, w której został wygenerowany błąd.

Bloki organizacyjne

Bloki organizacyjne wprowadzają w programie strukturę. Służą jako interfejs między systemem operacyjnym i programem użytkownika. OB są sterowane zdarzeniami. Zdarzenie, takie jak przerwanie diagnostyczne lub interwału czasowego, powoduje, że CPU wykonuje OB. Niektóre OB mają predefiniowane zdarzenia startowe i działanie.

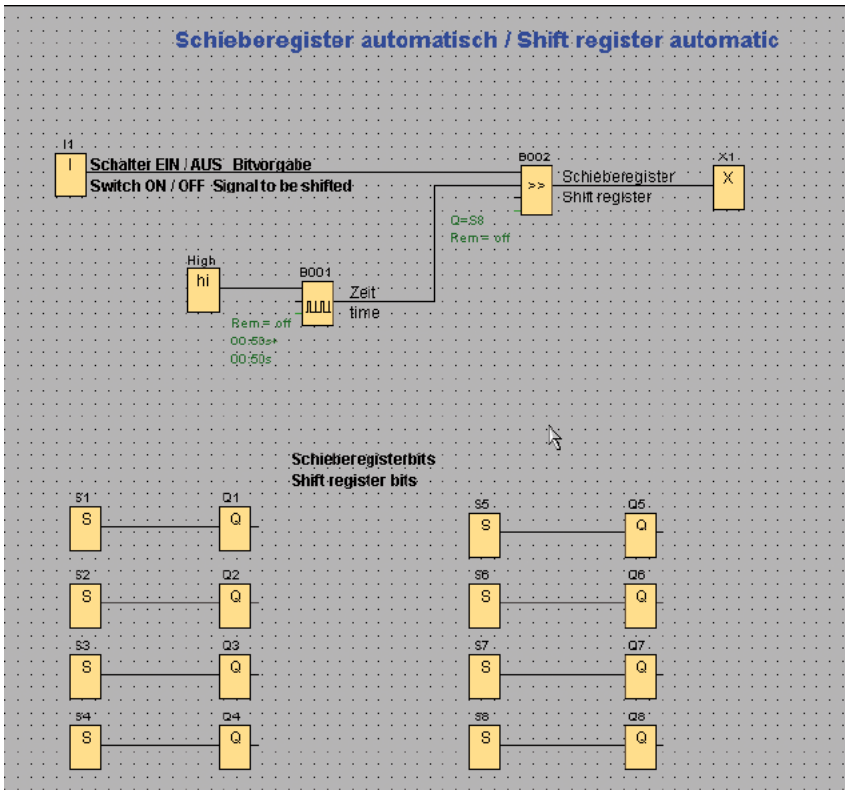
Cykliczny OB zawiera główny program. Użytkownik może włączyć więcej niż jeden cykliczny OB w swój program użytkownika. W trybie RUN, cykliczne OB działają z najniższym priorytetem i mogą być przerwane przez wszystkie inne typy programów. (Rozruchowy OB nie przerywa działania cyklicznego OB, ponieważ CPU wykonuje rozruchowy OB przed wejściem do trybu RUN.)

Po zakończeniu wykonywania cyklicznego OB, CPU natychmiast zaczyna ponownie wykonywać cykliczny OB. To działanie cykliczne jest normalnym sposobem pracy programowanych sterowników logicznych. W wielu aplikacjach cały program użytkownika jest zlokalizowany w jednym cyklicznym OB.

Użytkownik może utworzyć inne OB przewidziane do wykonywania specyficznych zadań, takich jak zadania rozruchowe, obsługa przerwań i błędów lub wykonywanie określonego kodu programu w stałych odstępach czasu. Te OB przerywają działanie OB cyklu programu.

Każde zdarzenie ma swój priorytet obsługi. Kilka zdarzeń wywołujących przerwania może być połączonych w klasę priorytetu. Więcej informacji na ten temat jest podanych w rozdziale Koncepcja PLC, w części opisującej wykonywanie programu użytkownika.

Jeśli do programu użytkownika zostanie utworzonych wiele cyklicznych OB, to CPU wykonuje wszystkie cykliczne OB w kolejności zgodnej z ich numerami, począwszy od głównego OB cyklu (domyślnie; OB 1). Na przykład, po zakończeniu pierwszego cyklicznego OB (OB 1), CPU wykonuje drugi cykliczny OB (np. OB 2 lub OB 200).



Rys. 3.

Funkcje (FC)

Funkcja jest to szybko uruchamiany blok kodu, który zazwyczaj wykonuje określone działania na zbiorze wartości wejściowych. FC przechowuje wyniki operacji w komórkach pamięci.

FC stosuje się do wykonywania następujących zadań:

- Standardowych i powtarzalnych działań, jak na przykład obliczeń arytmetycznych.
- Funkcji technologicznych, takich jak indywidualne sterowanie za pomocą działań logicznych.

FC może być wywoływana wielokrotnie w różnych miejscach programu. Ta możliwość wielokrotnego użycia FC upraszcza programowanie często występujących zadań.

Przeciwnie niż blok funkcji (FB), FC nie jest skojarzona z żadnym blokiem danych *instance* (DB). Dla danych tymczasowych występujących podczas przeprowadzania obliczeń, FC wykorzystuje lokalny stos danych. Dane tymczasowe nie są zapamiętywane. W celu zapamiętania danych należy przypisać wartości wyjściowej miejsce w pamięci, na przykład w pamięci M lub w globalnym DB.

Blok funkcji (FB)

Blok funkcji (FB) jest blokiem kodu, którego wywołanie może być programowane za pomocą parametrów bloku. FB ma zmienną pamięć zlokalizowaną w bloku danych (DB) lub instancji DB. Instancja DB zapewnia blok pamięci skojarzonej z tym egzemplarzem (lub „wywołaniem") FB i przechowuje dane po zakończeniu działania FB. Można skojarzyć różne bloki danych *instans* DB z różnymi wywołaniami FB. Bloki DB pozwalają na użycie tego samego FB do sterowania wielu urządzeń. CPU wykonuje program zawarty w FB i zapamiętuje parametry bloku oraz statyczne dane lokalne w danej instancji DB. Gdy wykonanie FB jest zakończone, wtedy CPU powraca do bloku kodu, z którego FB został wywołany. Instancja DB zachowuje wartości wpisane podczas tego wykonania FB.

FB są zwykle używane do sterowania działaniem zadań lub urządzeń, które nie kończą swojej pracy w jednym cyklu programu. W celu przechowywania parametrów operacyjnych w taki sposób, by były szybko dostępne w kolejnych cyklach programu, każda FB w programie użytkownika ma jeden lub więcej instancji DB. Kiedy FB jest wywoływana, wtedy również jest otwierany DB przechowujący parametry bloku i statyczne dane lokalne dla danego wywołania lub instancji FB. Instancja DB pamięta te wartości po zakończeniu działania FB.

Projektując FB do wykonywania ogólnych zadań sterowania, użytkownik może wykorzystać ten FB z wieloma urządzeniami wybierając różne instancje DB do różnych wywołań FB.

FB przechowuje w instancji DB parametry wejściowe (IN), wyjściowe (OUT) oraz wejściowo/wyjściowe (IN_OUT).

Jeżeli parametry wejściowe, wyjściowe lub wejściowo/wyjściowe bloku funkcji (FB) nie mają przypisanych wartości, to są wykorzystywane wartości pamiętane w instancji bloku danych (DB). W niektórych przypadkach to użytkownik musi ustalić te parametry.

Użytkownik może nadać wartości początkowe parametrom interfejsu FB. Te wartości zostaną przeniesione do skojarzonej instancji DB. Jeśli parametrom nie zostaną nadane wartości, to będą wykorzystane wartości pamiętane w instancji DB.

Wykorzystanie pojedynczego FB z wieloma instancjami DB

Na poniższym rysunku przedstawiono jeden OB, który trzykrotnie wywołuje jeden FB, za każdym razem z innym blokiem danych. Ta struktura pozwala wykorzystać jeden ogólny FB do sterowania kilku podobnych urządzeń, takich jak silniki, poprzez przypisanie mu w każdym wywołaniu różnych instancji bloku danych dla różnych urządzeń. Każda instancja DB przechowuje dane (jak szybkość, czas rozpędzania i całkowity czas pracy) dla indywidualnego urządzenia. W podanym przykładzie FB 22 steruje trzema oddzielnymi urządzeniami, przy czym DB 201 pamięta dane dla pierwszego, DB 202 dla drugiego, a DB 203 dla trzeciego urządzenia.

Blok danych (DB)

Bloki danych (DB) są umieszczane w programie użytkownika po to, by przechowywały dane dla bloków kodu. Wszystkie bloki kodu w programie użytkownika maja dostęp do globalnego DB, ale poszczególne instancje DB przechowują dane dla określonych bloków funkcji (FB).

Program użytkownika może przechowywać dane w specjalizowanych obszarach pamięci CPU, przeznaczonych dla wejścia (I), wyjścia (Q) i pamięci bitowej (M). Ponadto użytkownik może wykorzystywać bloki danych (DB) dla uzyskania szybkiego dostępu do danych przechowywanych w samym programie. Użytkownik może nadać DB status „tylko do odczytu".

Dane pamiętane w DB nie są usuwane po zamknięciu bloku danych lub po zakończeniu wykonywania korzystającego z nich bloku kodu. Są dwa typy DB:

- Globalny DB przechowuje dane dla bloków kodu programu użytkownika. Dostęp do danych zawartych w globalnym DB ma dowolny OB, FB i FC.
- Instancje DB przechowują dane dla określonych FB. Struktura danych w instancji DB odzwierciedla parametry (wejściowe, wyjściowe i wejściowo/wyjściowe) oraz dane statyczne FB. (Pamięć Temp FB nie jest przechowywana w instancji DB).

Andrzej Gawryluk

Artykuł powstał na bazie dokumentacji firmy Siemens.