Szkolenie PLC cz. 11
Podstawowe instrukcje sterownika TECO TP03

Po ogólnikowej charakterystyce sterownika TP03 dokonanej w poprzedniej odsłonie szkolenia, czas na analizę konkretnych instrukcji arytmetycznych, logicznych oraz sterujących przepływem programu w jednostce centralnej. Przedstawiony zostanie sposób wykorzystania przykładowych funkcji w rzeczywistych rozwiązaniach funkcjonalnych.

STRUKTURA PAMIĘCI WEWNĘTRZNEJ STEROWNIKA

Zanim przejdziemy do charakterystyki wybranych instrukcji, należy napisać kilka słów na temat konstrukcji rejestrów tworzących pamięć jednostki centralnej. Podstawową i zarazem najczęściej wykorzystywaną komórką pamięci jest rejestr 16-bitowy. W przypadku liczb całkowitych (INT) obejmuje on zakres od -32767 do 32767 (ostatni bit wykorzystywany jest jako bit znaku). Jeśli podany zakres jest niewystarczający dla danej aplikacji, istnieje możliwość wykorzystania dwóch kolejnych rejestrów aby obsłużyć liczby całkowite 32-bitowe. Podobnie jest w przypadku liczb rzeczywistych, gdzie użytkownik ma możliwość określenia dwóch kolejnych rejestrów 16-bitowych jako miejsce w pamięci przechowujące 32-bitową liczbę typu REAL. Wybór typu liczb przechowywanych w rejestrach nie odbywa się na poziomie parametrów samej pamięci, ale na poziomie wykorzystywanych instrukcji arytmetycznych. W zależności od zastosowanej funkcji (16 lub 32-bitowej), sterownik traktuje wskazane rejestry jako niezależne lub powiązane.

OPERACJE LOGICZNE

Operacje logiczne należą do podstawowych działań realizowanych na sterownikach PLC. Można wśród nich wyróżnić wiele rodzajów, od prostych do bardzo złożonych, jednak te najprostsze sprowadzają się do analizy pojedynczego wejścia binarnego oraz ustawienia pojedynczego wyjścia. Są to więc kluczowe operacje przy projektowaniu funkcjonalności oprogramowania tworzonego na sterowniki. W przypadku instrukcji wejściowych możemy wyróżnić:

- **instrukcję analizy stanu aktywnego wejścia binarnego – cewka normalnie otwarta,**
- **instrukcję analizy stanu nieaktywnego wejścia binarnego – cewka normalnie zamknięta,**
- **instrukcję analizy zmiany ze stanu aktywnego na nieaktywny – wykrywanie zbocza opadającego,**
- **instrukcję analizy zmiany ze stanu nieaktywnego na aktywny – wykrywanie zbocza narastającego.**

W przypadku cewek analizujących statyczny stan wejścia, instrukcje znajdujące się za nimi wykonywane są cały czas gdy dana cewka jest zwarta lub rozwarta. W przypadku elementów wykrywających dynamiczną zmianę stanu, instrukcje znajdujące się za nimi wykonywane są tylko raz, w momencie konkretnej zmiany stanu styku wejściowego.

OPERACJE ARYTMETYCZNE ORAZ OPERACJE PRZESUNIĘCIA DANYCH

Najczęściej wykorzystywanymi funkcjami arytmetycznymi są dodawanie, odejmowanie, mnożenie oraz dzielenie. Wszystkie wymienione funkcje występują w dwóch wersjach. W przypadku operacji na liczbach całkowitych wykorzystuje się instrukcje:

- **add – dodawanie,**
- **sub – odejmowanie,**
- **mul – mnożenie,**
- **div – dzielenie.**

O ile instrukcje dodawania, odejmowania oraz mnożenia nie generują nieścisłości związanych z wynikiem, o tyle instrukcja dzielenia może wprowadzić programistę w błąd w momencie dzielenia liczb, które dzielą się z resztą. Posługując się wyżej wymienioną instrukcją, uzyskamy wynik dzielenia zaokrąglony w dół do liczby wielokrotności dzielników zawierających się w dzielnej. Przykładowo dzieląc liczbę 10 przez liczbę 7, wynikiem będzie liczba 1. W przypadku gdy liczby dzielą się przez siebie bez reszty z dzielenia, wyżej wymieniony problem/osobliwość nie występuje.

W przypadku konieczności stosowania liczb zmiennoprzecinkowych, wykorzystujemy poniższe instrukcje:

- **deadd – dodawanie liczb typu REAL,**
- **desub – odejmowanie liczb typu REAL,**
- **demul – mnożenie liczb typu REAL,**
- **dediv – dzielenie liczb typu REAL.**

Stosowanie tych funkcji umożliwia wykonanie wszystkich operacji arytmetycznych z wymaganą dokładnością. Należy natomiast pamiętać o odpowiednim podzieleniu pamięci w momencie planowania struktury programu, po to aby nie wykorzystywać zajętych już miejsc (liczby typu REAL przechowywane są w dwóch kolejnych rejestrach). W przypadku tym, najwygodniej jest posługiwać się tylko parzystymi lub nieparzystymi indeksami rejestrów.

Aby dokonać konwersji liczby całkowitej na liczbę zmiennoprzecinkową lub odwrotnie posługujemy się następującymi instrukcjami:

- **int – przy przejściu na liczbę całkowitą,**
- **flt – przy konwersji na liczbę zmiennoprzecinkową.**

Szkolenie prowadzi:
Dominik Szewczyk

tel.: 32 789 00 13

Pierwsza z instrukcji rezerwuje jeden rejestr wyjściowy, natomiast druga, jak zostało wspomniane wyżej, rezerwuje dwa kolejne miejsca w pamięci.

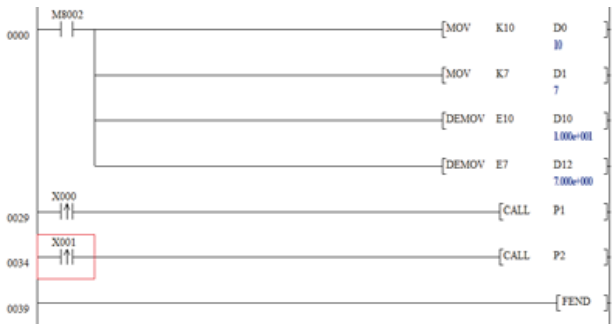
Jedną z najistotniejszych operacji sterujących jest instrukcja przesunięcia danych z jednej komórki rejestru do drugiej lub operacja inicjująca dane, czyli wprowadzająca je do pamięci sterownika. W zależności od rodzaju oraz rozmiaru liczb, na których przeprowadzane są operacje, wykorzystujemy instrukcję mov (w przypadku liczb całkowitych) lub demov (w przypadku liczb zmiennoprzecinkowych). Podobnie jak miało to miejsce przy instrukcjach arytmetycznych, pierwsza z nich rezerwuje jeden rejestr, natomiast druga dwa. Z punktu widzenia programu, wykorzystanie operacji przesunięcia jest takie samo w przypadku inicjacji rejestru oraz w przypadku przenoszenia danych z jednego rejestru do drugiego. W pierwszym przypadku źródłem danych jest stała liczba, natomiast w drugim zawartość wskazanego rejestru początkowego.

STEROWANIE WYKONANIEM PROGRAMU

Środowisko TP03- PCLink umożliwia utworzenie ośmiu podprogramów, z których jeden ma rangę programu nadrzędnego (main). Wywołanie pozostałych programów odbywa się z programu głównego lub z innych programów podrzędnych. Istotne jest zachowanie odpowiedniej hierarchii wywołań. Skok warunkowy do konkretnego podprogramu odbywa się poprzez wykorzystanie instrukcji call, w której jako parametr podawany jest numer programu do którego należy się odwołać. Wywołanie programu odbywa się bez podania argumentów wejściowych, ponieważ wszystkie dane wykorzystywane w projekcie są deklarowane globalnie. W momencie zakończenia skanowania kodu programu podrzędnego, wskaźnik programu głównego wraca na miejsce z którego wykonywany był skok i kontynuowane jest skanowanie programu nadrzędnego.

WYKORZYSTANIE OPISANYCH STRUKTUR NA PRZYKŁADZIE RZECZYWISTEGO PROGRAMU

Na kilku poniższych ilustracjach znajduje się kod źródłowy programu głównego (main) oraz programów podrzędnych.

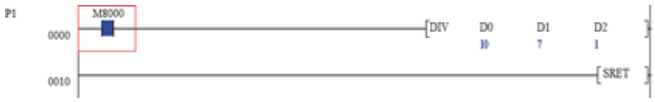


◀◀ *Rysunek 1 Program główny*

Analiza działania: Skanowanie rozpoczyna się od pierwszej linii programu głównego. Znacznik M8002 jest znacznikiem systemowym, aktywnym tylko podczas pierwszego skanowania kodu. Dlatego też na samym początku wykonania programu, do odpowiednich rejestrów wpisywane są stałe liczby – całkowite, wartości 10 oraz 7 do rejestrów kolejno D0 oraz D1, a także zmiennoprzecinkowe liczby 10.0 oraz 7.0 do rejestrów kolejno D10 oraz D12. Przerwa w numeracji podyktowana jest wspomnianym problemem zajętości pamięci, gdyż liczby zmiennoprzecinkowe zajmują dwa kolejne rejestry.

Poniżej znajdują się dwie linie kodu odpowiedzialne za skok do podprogramów P1 oraz P2. Ostatnia instrukcja programu głównego jest informacją dla sterownika o tym, iż w tym miejscu kończy się skanowanie pętli głównej.

W momencie wykrycia zbocza narastającego na wejściu binarnym X0, wskaźnik wykonywania programu przechodzi do początku podprogramu P1, natomiast w momencie wykrycia takiego zbocza na wejściu X1, program wykonuje skok do podprogramu P2.



◀ *Rysunek 2 Podprogram 1*

W podprogramie P1 znajdują się tylko dwie linie kodu. Jedna z nich zapoczątkowana jest znacznikiem M8000. Jest to znacznik aktywny przez cały czas wykonania programu – konieczny ze względu na przymus obecności instrukcji wejściowej przed każdą standardową instrukcją wyjściową. Linia ta realizuje dzielenie zawartości rejestru D0 przez zawartość rejestru D1. Wcześniej wpisane do tych rejestrów, całkowite liczby to kolejno 10 oraz 7. Wynikiem dzielenia jest więc liczba 1, która zostaje wpisana do rejestru wynikowego D2. Druga linia realizuje powrót do programu głównego. Jak można zobaczyć, operacja powrotu jako jedna z nielicznych nie wymaga instrukcji wejściowej.

◀ *Rysunek 3 Podprogram 2*

W podprogramie P2 zrealizowano podobne operacje jak miało to miejsce w P1. Różnica tkwi w typie danych, które biorą udział w obliczeniach. Dzielenie dotyczy liczb rzeczywistych, więc wynik także jest liczbą rzeczywistą. Instrukcja obecna w drugiej linii podprogramu dokonuje konwersji liczby zmiennoprzecinkowej na całkowitą. Z punktu widzenia optymalizacji kodu, takie rozwiązanie ma sens tylko w momencie, gdy większość operacji arytmetycznych, ze względu na ich charakter, realizujemy na liczbach rzeczywistych; natomiast wynik chcemy przedstawić w postaci liczby całkowitej. W tym momencie zaokrąglania dokonujemy tylko raz. W przypadku gdy realizowalibyśmy skomplikowane obliczenia i zastosowalibyśmy instrukcje dla liczb całkowitych, w sytuacji gdy cząstkowe wyniki także nie byłyby całkowite, zaokrąglanie miałoby miejsce w każdym kroku. Mogłoby to prowadzić do znacznych błędów w obliczeniach.