mgr inż. Marek WADECKI
Przemysłowy Instytut Automatyki
i Pomiarów MERA–PIAP
Warszawa

SPOSÓB PROGRAMOWANIA MIKROPROCESOROWEGO STEROWNIKA MSEP

W artykule zaprezentowano język FAST umożliwiający zaprogramowanie przemysłowego sterownika mikroprocesorowego MSEP. Podano podstawowe informacje niezbędne do samodzielnego pisania programów użytkowych. Autor artykułu jest twórcą opisanego języka i translatora FAST.

1. WPROWADZENIE

Programowalny mikroprocesorowy sterownik MSEP, opracowany w Przemysłowym Instytucie Automatyki i Pomiarów, jest przeznaczony do sterowania maszynami i urządzeniami technologicznymi, których sygnały wejściowe i wyjściowe mają charakter dwustanowy. Szczególnie przydatny jest do średnio skomplikowanych układów wykorzystujących pneumatyczne urządzenia wykonawcze przesterowywane za pomocą elektrozaworów. Możliwe jest również sterowanie dwoma silnikami krokowymi. Sterownik MSEP składa się z zespołu podstawowego, który może być uzupełniony o zespół rozszerzający. Każdy z zespołów ma 32 wejścia i 16 wyjść binarnych. Komunikację użytkownika ze sterownikiem umożliwia komputer typu IBM PC. Pełni on jednocześnie rolę programatora sterownika i daje możliwość pisania programów, sprawdzania poprawności leksykalnej i syntaktycznej napisanego programu, translacji tekstu programu na postać zakodowaną i zaprogramowania pamięci typu EPROM. Komputer może być też podłączony do sterownika w czasie uruchamiania programu na obiekcie, co umożliwia odczyt lub zapis wyjść, wejść oraz pamięci sterownika. Choć w sterowniku jest wykorzystywana pamięć typu EPROM, możliwe jest również sprawdzenie działania programu zapisanego bezpośrednio z komputera do pamięci typu RAM. Obecnie oprogramowanie sterownika składa się z translatora FAST, programu służącego do testowania sterownika przy użyciu komputera IBM XT/AT i programu programatora pamięci EPROM. Niniejszy artykuł poświęcony jest tylko części tego oprogramowania a mianowicie translatorowi FAST i jego językowi.

2. TRANSLATOR FAST

Translator FAST służy do zamiany programu źródłowego napisanego w języku FAST na równoważny mu funkcjonalnie program w języku asemblera Z80. Translator może być uruchamiany na mikrokomputerze IBM PC XT/AT (lub kompatybilnym) pracującym pod kontrolą systemu operacyjnego PC DOS. Jego obsługa ogranicza się do wywołania programu poleceniem FAST i po ukazaniu się na ekranie odpowiedniego komunikatu wpisania pełnej nazwy zbioru zawierającego wersję źródłową programu. Kod źródłowy jest wówczas tłumaczony na kod wynikowy, który jest zapisywany w pliku o nazwie identycznej z nazwą kodu źródłowego, ale zaopatrzonej w rozszerzenie BIN. Cała operacja translacji i zapisu trwa najwyżej kilka sekund. W przypadku napotkania błędów w programie źródłowym translator nie tworzy kodu wynikowego. Na ekranie ukazuje się wówczas lista popełnionych błędów zaopatrzona w stosowne komentarze (w języku polskim) oraz numery wierszy umożliwiające odszukanie błędów w tekście programu. Program poprawiony za pomocą edytora tekstu i przetłumaczony na kod wynikowy za pomocą translatora FAST umieszcza się następnie w pamięci sterownika MSEP.

3. PRZYGOTOWANIE PROGRAMU APLIKACYJNEGO

Chcąc zautomatyzować proces technologiczny za pomocą sterownika MSEP należy najpierw dokonać analizy automatyzowanego procesu i napisać jego algorytm przypisując wszystkim sygnałom występującym w procesie nazwy odpowiadające nazwom wejść i wyjść sterownika. Najczęściej do realizacji konkretnego programu nie wykorzystuje się wszystkich wejść i wyjść sterownika MSEP. W związku z tym można sobie pozwolić na wybranie zacisków sterownika o nazwach zapewniających możliwie największą czytelność programu. W typowych układach elektropneumatycznych występuje najczęściej szereg siłowników, które mogą być przesterowywane za pomocą elektrozaworów. Przełączenie zaworu następuje w wyniku wciśnięcia odpowiedniego przycisku na pulpicie sterowniczym bądź na skutek sygnału pochodzącego z układu sterowania automatycznego. Potwierdzeniem wykonania przez siłownik żądanego ruchu jest najczęściej przełączenie łącznika krańcowego. Niejednokrotnie spotyka się też oznaczenie urządzeń wykonawczych kolejnymi literami alfabetu.
Wszystko to wpłynęło na metodę programowania sterownika MSEP. W języku FAST wszystkie sygnały wyjściowe wysłane do urządzeń wykonawczych oznaczane są dużymi literami alfabetu. W celu ułatwienia powiązania zmiennych wejściowych z określonymi wyjściami sterownika oznaczono je takimi samymi lecz małymi literami alfabetu. Zaś w celu wyróżnienia sygnałów pochodzących z pulpitu sterowniczego do ich oznaczenia dodana jest na początku mała litera „p". Na przykład ruch siłownika

oznaczonego symbolem C w kierunku „1" jest wywoływany pojawieniem się sygnału wyjściowego C1, zaś sygnałem potwierdzającym wykonanie tej czynności będzie sygnał wejściowy c1. Analogicznie dla ruchu powrotnego siłownika w kierunku „0" są to sygnały C0 i c0. Natomiast sygnały pochodzące od przycisków sterowania ręcznego mają dla tego siłownika w zależności od wymaganego kierunku ruchu oznaczenia pc1 i pc0.

Działanie sterownika polega na wykonywaniu rozkazów (instrukcji) umieszczonych w programie. Pojedyncze rozkazy składają się na szereg wyrażeń logicznych opisujących zależności pomiędzy stanem panującym na wejściach lub w pamięci sterownika a stanem wyjść sterownika. Argumentami tych wyrażeń logicznych są między innymi zmienne odpowiadające sygnałom wejściowym sterownika. Pojawienie się napięcia na wejściach sterownika jest odbierane przez program jako przyjęcie wartości logicznej PRAWDA przez zmienną o nazwie identycznej z nazwą danego wejścia. Niektóre z wejść sterownika nie mają swoich odpowiedników wśród symboli wykorzystywanych podczas programowania. Są to wejścia umożliwiające uruchomienie sterownika, realizację funkcji stopu awaryjnego, czasowe wstrzymanie wykonywania programu, przełączanie trybów pracy itp. W zależności od stanu panującego na tych wejściach wykonywane są różne fragmenty programu napisanego w języku FAST.

4. JĘZYK FAST

4.1. Uwagi ogólne

Język FAST powstał w celu umożliwienia szybkiego i wygodnego pisania programów aplikacyjnych dla sterownika MSEP. Aby ułatwić opanowanie języka osobom, które miały styczność z programowaniem komputerów i z językami typu Pascal czy Fortran, w języku FAST podobne są funkcje niektórych symboli jak na przykład: WHEN, IF, THEN, ELSE, NOT, AND, OR, GOTO, END, ENDS. Stosowane są też symbole ułatwiające identyfikację pełnionych przez nie funkcji w programie, jak np. CZAS, ewentualnie symbole identyczne z nazwami zacisków sterownika MSEP. Struktura programu została ściśle podporządkowana możliwości przejrzystego opisu pracy sterownika w trybie automatyki i sterowania ręcznego. W przypadku pracy automatycznej możliwa jest zarówno typowa praca sekwencyjna jak i natychmiastowe wykonywanie poleceń opisanych logicznymi funkcjami kombinacyjnymi. Język FAST umożliwia łatwe programowanie silników skokowych, a także stosowanie funkcji czasowych nie wymagających instalowania jakichkolwiek dodatkowych modułów. Jednocześnie program napisany w języku FAST ułatwia tworzenie dokumentacji układu sterowania dzięki możliwości umieszczania prktycznie nieograniczonej liczby komentarzy.

4.2. Struktura programu

Tekst programu powinien rozpoczynać się słowem PROGRAM i kończyć słowem END. Translator FAST analizuje tylko tę część tekstu, która znajduje się po pierwszym napotkanym słowie PROGRAM i przed pierwszym napotkanym słowem END. Pomiędzy tymi słowami znajduje się segment programu opisujący pracę sterownika w trybie automatyki sekwencyjnej. Segment ten może występować samodzielnie lub wspólnie z innymi segmentami, o ile są one niezbędne do opisu pracy sterownika. Program w najbardziej rozbudowanej postaci zbudowany jest z następujących części:

```
PROGRAM

.....          }   deklaracje parametrów pracy
.....          }
.....          }   silników krokowych

START          }
.....          }
.....          }
.....          }
ENDS           }   segmenty opisujące blokady

START          }
.....          }
.....          }
.....          }
ENDS           }

HAND           }
.....          }   segment opisujący pracę w trybie sterowania
.....          }
.....          }   ręcznego
ENDS           }

AUTO           }   segment opisujący zjawiska wymagające
.....          }
.....          }   natychmiastowej reakcji podczas pracy
.....          }
ENDS           }   w trybie automatyki

.....          }   cześć opisująca pracę sekwencyjną
.....          }
.....          }   w trybie automatyki

END
```

W tekście programu może występować nieograniczona liczba komentarzy. Za komentarz uważany jest dowolny ciąg znaków znajdujący się w jednej linii programu i poprzedzony średnikiem. Translator FAST nie przenosi tych komentarzy w żadnej postaci do kodu wynikowego. Umieszczenie dodatkowego tekstu może jednak znacznie poprawić czytelność programu i być pomocne przy tworzeniu dokumentacji układu sterowania.

4.3. Symbole wykorzystywane w języku FAST

W tekście programu mogą być używane jedynie symbole przewidziane dla języka FAST. Oczywiście ograniczenie to nie stosuje się do fragmentów stanowiących komentarze. Poszczególne symbole zapisane w jednym wierszu muszą być oddzielone przynajmniej jednym znakiem spacji (odstępu). Poniżej znajduje się wykaz symboli, które są nazwami instrukcji stosowanych w języku FAST, bądź są argumentami tych instrukcji. Wykaz ten nie obejmuje symboli będących nazwami segmentów oraz niektórych symboli wykorzystywanych do sterowania silnikami skokowymi.

4.3.1. Symbole stosowane do tworzenia wyrażeń logicznych

WHEN DO
IF THEN ELSE
AND OR NOT ( ) – iloczyn i suma logiczna, negacja, nawiasy

4.3.2. Symbole reprezentujące instrukcje wykonawcze zmieniające stan wyjść sterownika

A1_1 A0_1 B1_1 B0_1 C1_1 C0_1 D1_1 D0_1 E1_1 E0_1 F1_1 F0_1 G1_1 G0_1 H1_1 H0_1 I1_1 I0_1 J_1 K_1 T_1 R_1 S_1 X1_1 X0_1 Y1_1 Y0_1 IX_1 IY_1 – instrukcje te zmieniają na aktywny stan następujących wyjść sterownika: A1, A0, B1, B0, C1, C0, D1, D0, E1, E0, F1, F0, G1, G0, H1, H0, I1, I0, J, K, T, R, S, X1, X0, Y1, Y0, IX, IY

A0_0 B1_0 B0_0 C1_0 C0_0 D1_0 D0_0 E1_0 E0_0 F0_0 G1_0 G0_0 H1_0 H0_0 I1_0 I0_0 J_0 R_0 S_0 X1_0 X0_0 Y1_0 Y0_0 IX_0 IY_0 – instrukcje te kasują stan aktywny na następujących wyjściach sterownika: A0, B1, B0, C1, C0, D1, D0, E1, E0, F0, G1, G0, H1, H0, I1, I0, J, R, S, X1, X0, Y1, Y0, IX, IY

4.3.3. Symbole reprezentujące instrukcje wykonawcze nie zmieniające stanu wyjść sterownika

CZAS1 CZAS2 CZAS3 CZAS4 CZAS5 CZAS6 CZAS7 CZAS8 – instrukcje te rozpoczynają proces odmierzania odstępów czasowych o długości określonej liczbą umieszczoną po każdym z podanych symboli.

FL1_1 FL2_1 FL3_1 FL4_1 FL5_1 FL6_1 FL7_1 FL8_1 FL9_1 FL10_1 FL11_1 FL12_1 FL13_1 FL14_1 FL15_1 FL16_1 – instrukcje te nadają znacznikom FL1 ÷ FL16 wartość logiczną PRAWDA.

FL1_0 FL2_0 FL3_0 FL4_0 FL5_0 FL6_0 FL7_0 FL8_0 FL9_0 FL10_0 FL11_0 FL12_0 FL13_0 FL14_0 FL15_0 FL16_0 – instrukcje te nadają znacznikom FL1 ÷ FL16 wartość logiczną FAŁSZ.

LICZN1 LICZN2 LICZN3 LICZN4 – powodują załadowanie wartości liczbowej umieszczonej za symbolem do licznika nr 1, 2, 3 lub 4.

L1–1 L2–1 L3–1 L4–1 – powodują zmniejszenie o 1 zawartości licznika 1, 2, 3 lub 4
GOTO /etykieta/ – instrukcja powodująca skok do miejsca wskazanego przez etykietę.

4.3.4. Symbole służące do zapisu instrukcji wykonawczych związanych z obsługą silników krokowych

SX1 SY1 SX1/2 SY1/2 SX1/4 SY1/4 SX0 SY0 SXSTOP SYSTOP PREDKX PREDKY DROGAX DROGAY

4.3.5. Symbole reprezentujące zmienne binarne o stanie logicznym odpowiadającym obecności napięcia na wejściach sterownika

a1 a0 b1 b0 c1 c0 d1 d0 e1 e0 f1 f0 g1 g0 h1 h0 i1 i0 j k l m n o r s t u w z x1 x0 y1 y0 start

pa1 pa0 pb1 pb0 pc1 pc0 pd1 pd0 pe1 pe0 pf1 pf0 pg1 pg0 ph1 ph0 pi1 pi0 px1 px0 py1 py0

4.3.6. Symbole reprezentujące zmienne binarne nie związane bezpośrednio ze stanem wejść sterownika MSEP

czas1 czas2 czas3 czas4 czas5 czas6 czas7 czas8 – zmienne przyjmujące stan logiczny PRAWDA po zakończeniu procesu odmierzania czasu CZAS1 ÷ CZAS8, zaś stan FAŁSZ po rozpoczęciu odmierzania.

fl1 fl2 fl3 fl4 fl5 fl6 fl7 fl8 fl9 fl10 fl11 fl12 fl13 fl14 fl15 fl16 – zmienne mające stan logiczny zgodny z aktualnym stanem znaczników FL1 ÷ FL16.

liczn1 liczn2 liczn3 liczn4 – mają wartość logiczną PRAWDA, gdy odpowiadający im licznik zawiera zero tzn. zakończył odliczanie.

silx sily – instrukcje stosowane przy wykorzystaniu silników krokowych.

4.4. Sposób tworzenia wyrażeń logicznych

Translator FAST zapewnia zachowanie ogólnie uznawanych priorytetów dla wyrażeń logicznych. Kolejność wykonywania działań jest więc następująca:

1. wyrażenia nawiasowe
2. NOT – negacja
3. AND – koniunkcja
4. OR – alternatywa

Zapis (x0 OR x1) AND (y0 OR NOT y1) jest równoważny wyrażeniu: $(x0 \vee x1) \wedge (y0 \vee \overline{y1})$. Najpierw zanegowany jest sygnał y1, następnie obliczone wartości w obu nawiasach, a na końcu jest wyznaczany iloczyn wyrażeń w nawiasach. Z kolei wykonanie instrukcji: IF (l_OR m) AND NOT start THEN H1_1 polega na uaktywnieniu wyjścia H1 tylko wtedy, gdy jednocześnie wystąpi brak napięcia na wejściu „start" sterownika i obecność napięcia przynajmniej na jednym z dwu wejść: „l" lub „m".

4.5. Opis instrukcji typu WHEN ... DO ... i IF ... THEN ... ELSE

Typowy program składa się z ciągu instrukcji typu:
WHEN /wyrażenie logiczne/ DO /ciąg instrukcji wykonawczych/ oraz instrukcji typu:
IF /wyrażenie logiczne/ THEN /ciąg instrukcji wykonawczych/
ELSE /ciąg instrukcji wykonawczych/

Wykonanie takich instrukcji polega na sprawdzeniu wartości wyrażenia logicznego i w przypadku otrzymaniu wyniku PRAWDA wykonaniu ciągu instrukcji wykonawczych znajdujących się po symbolu DO lub po symbolu THEN a przed symbolem ELSE. Instrukcje umieszczone po symbolu ELSE są wykonywane w przypadku otrzymania wyniku FAŁSZ. Dopuszcza się również pominięcie ciągu instrukcji wykonawczych po symbolach DO, THEN lub ELSE. Różnica między omawianymi instrukcjami polega na tym, że konstrukcja WHEN /wyrażenie logiczne/ DO ... zapewnia wstrzymanie wykonywania części programu znajdującej się za symbolem DO do czasu, aż wyrażenie logiczne przybierze wartość PRAWDA. Można więc powiedzieć, że każda instrukcja typu WHEN ... DO ... wyznacza pojedynczy krok programu podczas pracy sekwencyjnej. Natomiast instrukcja: IF /wyrażenie logiczne/ THEN ... ELSE ... oraz jej skrócona postać: IF /wyrażenie logiczne/ THEN ... powodują jednokrotne sprawdzenie wartości wyrażenia logicznego.

W przypadku otrzymania wyniku PRAWDA wykonywane są tylko instrukcje umieszczone pomiędzy symbolami THEN a ELSE. Natomiast wskutek otrzymania wyniku FAŁSZ wykonywane są tylko instrukcje znajdujące się po symbolu ELSE. Następnie, niezależnie od otrzymanego wyniku, dokonywane jest natychmiastowe przejście do następnej instrukcji programu. Ilustruje to przykład:

```
PROGRAM
START: WHEN start DO
PETLA: WHEN b0 DO B1_1
       WHEN a1 b1 DO B1_0
       IF s THEN GOTO START ELSE GOTO PETLA
END
```

Wykonując ten program sterownik czeka na sygnał od przycisku podającego napięcie na wejście „start". Następnie po pojawieniu się napięcia na wejściu „b0" uaktywnia wyjście B1. Gdy pojawi się napięcie na wejściu „b1" stan aktywny wyjścia B1 jest kasowany, a potem znowu przywracany po ponownym pojawieniu się napięcia na wejściu „b0". Czynności te powtarzane są w pętli do momentu pojawienia się napięcia na wejściu „s". Wtedy bowiem wykonywany jest skok w programie do miejsca oznaczonego etykietą START i dokonywanie zmian na wyjściu B1 zostaje zawieszone do czasu ponownego pojawienia się sygnału „start".

4.6. Instrukcja GOTO i etykiety

Wykonanie instrukcji GOTO /etykieta/ sprawia, że następną wykonywaną instrukcją będzie ta, która znajduje się za etykietą o nazwie podanej po symbolu GOTO. W programie można wykonywać skoki do przodu i do tyłu w obrębie danego segmentu lub części programu. Zakazane jest wykonywanie skoków do innych segmentów. Etykietą w programie jest dowolny ciąg zawierający co najwyżej 7 znaków, po których następuje bezpośrednio dwukropek np.:

```
ETYKIET: WHEN a1 DO B1_1 B0_0
12$ab:   WHEN b1 DO C0_1 C1_0
         ......................
e:       IF a0 THEN B1_0 B0_1 ELSE GOTO ETYKIET
```

4.7. Programowe odmierzanie opóźnienia czasowego

W języku FAST istnieje możliwość jednoczesnego mierzenia ośmiu różnych wartości opóźnienia czasowego. Rozpoczęcie odmierzania czasu następuje w wyniku wykonania instrukcji składającej się z symbolu CZAS zakończonego numerem licznika czasu (1...8) oraz wartości liczbowej (z zakresu 1...65535) równej wymaganej zwłoce czasowej wyrażonej w setnych częściach sekundy. W ten sposób można odmierzać czas o wartości z zakresu od 0,01 sekundy do 10 minut 55,35 sekundy. O upłynięciu zadanego czasu informuje zmiana oznaczana symbolem „czas" zakończonym numerem licznika.

Przykładowo rozpoczęcie odliczania czasu nr 5 równego 4 sekundom następuje po wykonaniu instrukcji: CZAS5 400. Zmienna czas5 przyjmuje wtedy wartość FAŁSZ. Po zakończeniu odliczania czasu nr 5 zmienna o nazwie czas5 przyjmuje wartość PRAWDA.

Inny przykład:

```
PROGRAM

POCZ: WHEN start DO CZAS8 100
      WHEN czas8 DO CZAS6 6000 D1_1
      WHEN czas6 DO D1_0 GOTO POCZ

END
```

Powyższy program będzie wykonywany następująco: po pojawieniu się sygnału od przycisku podającego napięcie na wejście „start" sterownika MSEP rozpocznie się odmierzanie czasu nr 8 równego 1 sek. Po upływie tego czasu rozpocznie się odmierzanie czasu nr 6 równego 1 min. i jednocześnie zostanie uaktywnione wyjście D1 sterownika. Po upływie 1 minuty stan aktywny na wyjściu D1 zostanie skasowany i rozpocznie się oczekiwanie na ponowne pojawienie się sygnału „start".

4.8. Korzystanie ze znaczników binarnych

W programie można korzystać z maksymalnie 16 znaczników ponumerowanych od 1 do 16. Zmiany stanu znaczników dokonuje się za pomocą instrukcji rozpoczynających się od liter „FL", zaś stan znaczników reprezentują zmienne o nazwach zaczynających się od liter „fl". Przykładowo znacznik nr 15 można ustawić instrukcją FL15_1. Wówczas zmienna fl15 przyjmie wartość PRAWDA. Po wykonaniu instrukcji FL15_0 zmienna fl15 przyjmuje wartość FAŁSZ.

4.9. Programowanie funkcji licznikowych

W programie możliwe jest wykorzystanie czterech liczników o maksymalnej pojemności 65535. Są to liczniki zliczające w dół od wartości zadanej aż do zera. Wprowadzenie do licznika wartości początkowej dokonywane jest przy zastosowaniu instrukcji składającej się z symbolu LICZN1, LICZN2, LICZN3, lub LICZN4 i wartości liczbowej. Licznik zmniejsza swą zawartość o 1 aż do osiągnięcia wartości zerowej każdorazowo po napotkaniu instrukcji L1–1, L2–1, L3–1 lub L4–1. Osiągnięcie wartości zerowej sygnalizowane jest przyjęciem wartości PRAWDA przez zmienną liczn1, liczn2, liczn3 lub liczn4 w zależności od numeru wykorzystywanego licznika.

```
START: B1_0
       LICZN3 50
PETLA: WHEN a1 DO L3-1
       WHEN NOT a1 DO
       IF NOT liczn3 THEN GOTO PETLA ELSE B1_1
```

Wykonanie powyższego fragmentu programu spowoduje, że wyjście B1 sterownika będzie się znajdowało w stanie nieaktywnym, dopóki licznik numer 3 nie zliczy 50 impulsów na wejściu a1 sterownika.

4.10. Opis poszczególnych segmentów programu

4.10.1. Część opisująca pracę sekwencyjną

W zależności od stanu panującego na wejściu „auto" sterownika MSEP wykonywany jest program pracy w trybie automatyki lub w trybie sterowania ręcznego. Program pracy w trybie „automatyki sekwencyjnej" składa się z ciągu instrukcji typu WHEN ... DO ... oraz instrukcji typu: IF ... THEN ...ELSE ....
Instrukcje typu WHEN ... DO ... wyznaczają, jak już wspomniano, kolejne kroki programu, co może być wykorzystane przy pracy automatycznej krokowej. Wykonanie kolejnej instrukcji typu WHEN ... DO ... nie jest nigdy możliwe bez wykonania po-

przedniej. Wynika stąd, że kolejność występowania instrukcji w tekście programu musi być zgodna z kolejnością, w jakiej powinny one wpływać na sterowany proces. W ten sposób można uzyskać typową pracę sekwencyjną sterownika.
Część programu, opisująca pracę automatyczną sekwencyjną, jako najbardziej typowa dla sterownika MSEP nie jest wyróżniana żadnym symbolem rozpoczęcia ani zakończenia segmentu.

4.10.2. Segment opisujący zjawiska wymagające natychmiastowej reakcji podczas pracy w trybie automatyki

Niektóre zjawiska pojawiające się podczas pracy sekwencyjnej wymagają natychmiastowego wykonania określonych zadań niezależnie od realizowanego w danej chwili kroku programu. Język FAST pozwala na zaprogramowanie takich działań i opisanie sytuacji, w jakich powinny być podjęte. Do tego celu służy segment programu rozpoczynający się symbolem AUTO i kończący się symbolem ENDS. W segmencie tym występuje szereg instrukcji typu IF ... THEN ... ELSE ... realizujących funkcje kombinacyjne. Zabronione jest natomiast używanie instrukcji WHEN ... DO ... oraz instrukcji skoku do etykiety występującej w innym segmencie programu. Wszystkie instrukcje występujące w tym segmencie programu są wykonywane, o ile nie są celowo ominięte za pomocą instrukcji GOTO i, oczywiście, o ile spełnione są warunkujące ich wykonanie równania logiczne. Po wykonaniu ostatniej instrukcji występującej w tym segmencie nastąpi powrót do pierwszej instrukcji. Nie należy zatem umieszczać na końcu instrukcji skoku na początek segmentu. A oto przykład programu zawierającego segment AUTO.

```
PROGRAM

AUTO

IF l OR m THEN E1_0 E0_1

IF b0 AND NOT ( l OR m ) THEN E0_0 E1_1

END

KROK1: WHEN start DO
       ..............

END
```

Przedstawiony fragment programu zapewnia natychmiastową zmianę wyjść E1 E0 w reakcji na określony stan wejść b0, l i m. Jest to reakcja niezależna od wykonania dowolnej części programu pracy sekwencyjnej zaczynającej się od etykiety KROK1.

4.10.3. Segment opisujący pracę w trybie sterowania ręcznego

Często rozpoczęcie pracy w cyklu automatycznym wymaga uprzedniego sprowadzenia urządzeń wykonawczych do położenia wyjściowego. Osiąga się to za pomocą przycisków na pulpicie sterowniczym. Przyciski te ułatwiają też sprawdzenie poprawności działania nowo opracowanego układu oraz sprawdzenie poprawności podłączenia i działania poszczególnych urządzeń. Praca w trybie sterowania ręcznego ma miejsce wówczas, gdy do wejścia „auto" sterownika nie jest doprowadzone napięcie. W takiej sytuacji wykonywane są instrukcje znajdujące się w segmencie rozpoczynającym się symbolem HAND i zakończonym symbolem ENDS. Struktura segmentu HAND i sposób wykonywania zamieszczonych w nim instrukcji są identyczne jak w przypadku segmentu AUTO. Nie może on zawierać instrukcji WHEN ... DO ... ani instrukcji skoku do etykiety występującej w innym segmencie programu. Przykład:

```
PROGRAM
    .
HAND
IF pc1 THEN C0_0 C1_1
IF pc0 THEN C1_0 C0_1
IF pd1 THEN D0_0 D1_1
IF pd0 THEN D1_0 D0_1
ENDS
AUTO
IF NOT ( l AND m OR start ) THEN H0_1 ELSE H0_0
.....
```

W podanym przykładzie wciśnięcie przycisków podłączonych do wejść pc1,pc0, pd1, pd0 serownika powoduje przesterowanie siłowników C, D podłączonych do wyjść C0, C1, D0, i D1.

4.10.4. Segmenty opisujące blokady

Ze względu na bezpieczeństwo pracy obsługujących sterowany układ lub w celu zabezpieczenia współpracujących urządzeń przed zniszczeniem powinno się uniemożliwić start niektórych urządzeń w sytuacjach, gdy nie są spełnione pewne warunki. W języku FAST można każdemu z wyjść przypisać wyrażenie logiczne, które warunkuje zmianę napięcia na tym wyjściu. Realizuje się to wstawiając do programu krótkie segmenciki rozpoczynające się symbolem START, po którym występuje symbol blokowanego wyjścia, wyrażenie logiczne i symbol ENDS. Na przykład:

```
PROGRAM

START A0_1
      b0 AND NOT a0
ENDS
START A0_0
      a0
ENDS
START C0_1
      ( b0 OR b1 ) AND NOT a1
ENDS
HAND
IF pa1 THEN A0_0 A1_1
.....................
```

W przedstawionym przykładzie zarówno w trybie pracy ręcznej jak i automatycznej sygnał wyjściowy A0 może pojawić się tylko przy obecności sygnału wejściowego b0 i braku sygnału wejściowego a0. Natomiast wyłączenie sygnału wyjściowego A0 możliwe jest tylko przy obecności sygnału wejściowego a0. Z kolei uaktywnienie wyjścia C0 możliwe jest tylko przy obecności sygnałów wejściowych na wejściu b0 lub b1 i braku sygnału na wejściu a1.

4.11. Przykładowy program napisany w języku FAST

Poniżej przedstawiono przykładowy program składający się z omówionych powyżej segmentów. Program zapewnia sterowanie dwoma siłownikami dwustronnego działania B i C, jednym urządzeniem oznaczonym literą D oraz dodatkowo podaje do sterowanego procesu sygnał E. Każdorazowe przestawienie siłownika C wymaga obecności sygnału b0 na wejściu sterownika. Po przełączeniu sterownika na tryb pracy ręcznej wszystkie wymienione urządzenia mogą być przesterowywane z pulpitu sterowniczego przyciskami. W czasie programu opisującej pracę w cyklu automatycznym pokazane jest użycie licznika czasu i licznika zdarzeń. Zrozumienie treści programu powinny ułatwić zamieszczone komentarze.

```
PROGRAM

START   C0_1     ;blokada wyjścia C0
        b0       ;o ile nie ma sygnału wejściowego b0
ENDS

START   C1_1     ;pojawienie się sygnału na wyjściu C1
        b0       ;możliwe jest tylko przy obecności
ENDS             ;sygnału wejściowego b0
;  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

HAND    IF pb1 THEN B0_0 B1_1 ;sterowanie urządzeniami
        IF pb0 THEN B1_0 B0_1 ;przy pomocy
        IF pc1 THEN C0_0 C1_1 ;przycisków ręcznych
        IF pc0 THEN C1_0 C0_1
        IF pd1 THEN D1_1
        IF pd0 THEN D1_0
ENDS
;  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *

AUTO    IF b0 OR c0 OR c1 THEN E1_1 ELSE E1_0
;sygnał wyjściowy E1 jest alternatywą
;trzech sygnałów wejściowych c0, c1 lub b0
ENDS
;  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
;Segment opisujący pracę sekwencyjną w cyklu automatycznym:

START: WHEN start AND c0 AND b0 DO
                    ;rozpoczęcie pracy jest uwarunkowane jednoczesnym
                    ;podaniem sygnałów wejściowych: start, c0 i b0
           LICZN1 4 ;załadowanie liczby 4 do licznika obiegu pętli
```

```
PĘTLA:          C0_0 C1_1 ;uruchomienie siłownika C
                          ;sterowanego dwustronnie sygnałami C0 i C1
      WHEN c1 DO B0_0 B1_1 ;uruchomienie siłownika B po potwierdzeniu
                          ;wykonania ruchu przez siłownik C
      WHEN b1 DO D1_1 CZAS1 50
                  ;po potwierdzeniu wykonania ruchu przez siłownik B
                  ;załączenie urządzenia D
                  ;i rozpoczęcie odmierzania czasu 0,5 sek
    WHEN czas1 DO B1_0 B0_1 ;po upływie 0,5 sek wycofanie siłownika B
    WHEN b0 DO C1_0 ;po wycofaniu siłownika B wycofanie siłownika C
    WHEN c0 DO D1_0 ;po wycofaniu siłownika C wyłączenie urządzenia D
            L1-1 ;zmniejszenie zawartości licznika obiegu pętli
     IF NOT liczn1 THEN GOTO PETLA ;skok o ile pętla nie była
                                   ;wykonana czterokrotnie
                  ELSE GOTO START ;w przeciwnym razie skok
                                  ;na początek programu
END
;koniec programu
;  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
```