074 A

PRZEMYSŁOWY INSTYTUT AUTOMATYKI I POMIARÓW
MERA-PIAP
Al. Jerozolimskie 202 02-222 Warszawa Telefon 23-70-81

Ośrodek Automatyzacji Procesów Produkcji

przy współpracy Warszawskiego Centrum Studenckiego Ruchu Naukowego

Główny wykonawca mgr inż. A. Aderek

Wykonawcy mgr inż. W. Hernik, mgr inż. J. Korytkowski, mgr inż. S. Skoneczny, mgr inż. M. Stodolski

Konsultant

Nr zlecenia RP-27.3

Opracowanie oprogramowania dla modułowych robotów PR-02E.

Wykonanie projektu oprogramowania.

Zleceniodawca CPBR 7.1

Pracę rozpoczęto dnia

zakończono dnia 88.11.30

Kierownik Ośrodka
mgr inż. A. Aderek

Z-ca Dyrektora d/s Automatyki
dr inż. T. Gałązka

Praca zawiera:

stron
rysunków
fotografii
tabel
tablic
załączników

Nr rejestr. 6203

Rozdzielnik - ilość egz:

Egz. 1 BOINTE
Egz. 2 OAP
Egz. 3
Egz. 4
Egz. 5
Egz. 6

Analiza deskryptorowa Robot przemysłowy, oprogramowanie, język programowania.

Analiza dokumentacyjna

W pracy przedstawiono definicję języka programowania i zasady programowania robotów PR-02E oraz projekt oprogramowania realizującego ten język w układzie sterowania opartym o pakiety MM-86.

Tytuły poprzednich sprawozdań

Opracowanie oprogramowania dla modułowych robotów PR-02E. Opracowanie założeń na oprogramowanie.

338.45:62/69].002.1/2 Roboty przemysłowe
681.3.06 Oprogramowanie

UKD

PIAP 21/88 10000

2

SPIS TRESCI

WPROWADZENIE.

Niniejsze opracowanie zawiera projekt oprogramowania robota przemyslowego PR-02E. Zostal on podzielony na trzy czesci. W pierwszej z nich omowiono zasade dzialania programu uzytkowego robota z punktu widzenia uzytkownika. Zostal opisany w nim jezyk programowania robota oraz zasady poslugiwania sie panelem programowania i panelem opearacyjnym. Opisane zostaly wszystkie dozwolone stany panelu, opis funkcyjny klawiszy oraz sposoby zglaszania sytuacji błędnych przez program sterownika robota. Kazda instrukcja programu sterowania zostala opisana zarowno pod wzgledem funkcjonalnym jak i sposobu jej programowania. Na koncu pierwszej czesci zamieszczono spis bledow sygnalizowanych przez program sterujacy robota.

W czesci drugiej sprawozdania zostal przedstawiony schemat programu robota. Wyszczegolnione zostaly w nim zmienne i struktury dostepne dla wszystkich podprogramow takie jak :

- deskryptory opisujace silowniki elektryczne i pneumatyczne,
- struktury zawierajace opisy bledow,
- struktury zawierajace opis programu uzytkowego i sposob dostepu do niego.

W dalszej czesci zostala wyjasniona koncepcja budowy programu, jego podzial na niezalezne moduly i komunikacja miedzy nimi. Niektore z proponowanych modulow zostaly przedstawione bardzo dokladnie, lacznie z fragmentami opisanymi w uzytym jezyku programowania (jezyku C). Poniewaz do stworzenia oprogramowania zostanie uzyta czesc istniejacych procedur dla robotow IRb-6/60 przedstawione zostaly sposobu wykorzystania tych procedur i komunikacja z nimi.

Ostatnia czesc opracowania tworza cztery zalaczniki. Zawieraja one dokladny opis komunikacji miedzy panelem programowania i sterownikiem robota na poziomie programow obslugi panelu. Wyszczegolnione zostaly w nich formaty przechowywania instrukcji w pamieci operacyjnej, symbole znakow itd.

Przedstawione opracowanie stanowi wedlug autorow podstawe do rozpoczecia pracy nad tworzeniem programu sterujacego dla robota przemyslowego PR-02E oraz dokonania podzialu pracy miedzy poszczegolnych programistow.

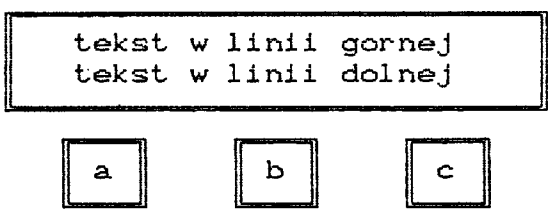
1. WSTEP.

W ponizszej czesci projektu przedstawiono zwiezly opis panelu programowania i panelu operacyjnego oraz przedstawiono sposob poslugiwania sie nimi. Przedstawiono skladnie jezyka robota PR-02E z punktu widzenia uzytkownika oraz zasady tworzenia, poprawiania i wykonywania programow uzytkowych. Przedstawiono rowniez spis sytuacji sygnalizowanych jako bledne.

1.1. Opisy konwencji stosowanych w tekscie.

Przy opisie procesu wykonywania jakiejs operacji na panelu programowania przyjeto nastepujaca konwencje:

1. W ramach opisu realizacji jednej funkcji wybranej z menu glownego wprowadzono numeracje stanow wyswietlaczy, tzn. kazdy stan wyswietlaczy, z ktorym operator moze sie spotkac w trakcie realizacji tej funkcji ma swoj numer (za wyjatkiem sytuacji zwiazanych z sygnalizacja bledow, ktorych opis jest w rozdziale dotyczacym sygnalizacji bledow).
2. Dla stanow wyswietlacza, w ktorych moglyby pojawic sie watpliwosci co do sposobu dalszego operowania, dolaczono odpowiedni komentarz.
3. Stan wyswietlacza ilustrowany jest nastepujacym rysunkiem:



gdzie:

- "tekst w linii gornej" i "tekst w linii dolnej" symbolizuja tekst max. 24-znakowy, wyswietlany w linii gornej/dolnej,
- "a", "b", "c" oznaczaja numery stanow, do ktorych nastapi przejscie po nacisnieciu przycisku funkcyjnego oznaczonego tym numerem. Jezeli przycisk nie jest oznaczony zadnym numerem, to jest on nieaktywny (jego nacisniecie nie powoduje zadnej reakcji). Jezeli na przycisku jest litera "X", to oznacza to przejscie do realizacji innej funkcji menu glownego.

8

2. PANEL PROGRAMOWANIA I PANEL OPERACYJNY

Panel programowania i panel operacyjny sa urzadzeniami umozliwiajacymi komunikacje miedzy operatorem a robotem. Za ich posrednictwem mozliwe jest programowanie robota oraz uruchamianie napisanych poprzednio programow.

Idea programowania robota PR-02E za posrednictwem panelu jest podobna jak dla robotow typu IRp. W przypadku robota PR-02E zmienilo sie znaczenie funkcjonalne przyciskow oraz zakres i skala ruchow wynikajaca z kinematyki manipulatora. Zgodnie z przyjetymi zalozeniami manipulator skladac sie bedzie z maksymalnie trzech modulow elektrycznych wystepujacych w roznych kombinacjach z modulami pneumatycznymi (dwustanowymi lub trzystanowymi). Maksymalna liczba modulow pneumatycznych dwustanowych wynosi 10. Jeden modul trzystanowy od strony programowej traktowany bedzie jako dwa dwustanowe. Wynika z tego, ze maksymalna liczba modulow trzystanowych wynosi 5.

2.1. Panel programowania.

Panel programowania (rys. 1) jest wydzielonym, przenosnym urzadzeniem umozliwiajacym obsluge robota przy ustawieniu sie operatora w pozycji i miejscu najdogodniejszym z punktu widzenia efektywnosci dzialania. Panel programowania, jak to sugeruje nazwa, najbardzej intensywnie wykorzystywany jest w trakcie programowania robota. W odroznieniu od robotow typu IRp wykorzystywany jest on rowniez podczas deklarowania konfiguracji manipulatora oraz sluzy do synchronizacji robota. Schemat rozmieszczenia poszczegolnych elementow panelu podany jest na rysunku 1.

W sklad panelu programowania wchodza nastepujace elementy:

- kilka grup przyciskow,
- wyswietlacze alfanumeryczne,
- pokretlo zmiany predkosci oraz pracy skokowej
- przycisk stopu awaryjnego.

Ze wzgledu na spelniane funkcje mozna wyróznic kilka grup przyciskow (na rysunku 1 zaznaczone linia i odpowiednim numerem). Oznaczenia stosowane w niniejszym podreczniku (skrot podany miedzy znakami "") wraz z krotkim opisem spelnianych funkcji zestawiono ponizej. Jezeli przy przycisku znajduje sie dioda swiecaca, to w opisie umieszczono znak (D).

Grupa 1.

MEN (przegladanie menu) - wyswietlenie na dolnym wyswietlaczu kolejnego fragmentu menu (D),

AUT (start programu) - wyswietlenie na dolnym wyswietlaczu mozliwych opcji uruchomienia programu,

PRZ (przelacz) - umozliwienie poruszania poszczególnymi ~~wszystkimi~~ modulami przy wykorzystaniu odpowiadajacych im grup przyciskow. Predkosc ruchu oraz tryb poruszania manipulatorem ustawiany jest za pomoca pokretla zmiany predkosci i pracy skokowej,

OPE (reczne operowanie systemem) - odczytywanie, zapisywanie, deklarowanie i modyfikowanie parametrow pracy robota (D),

SYM (symulacja) - symulowanie sygnalow wejsia,

STP (stop) - bezwarunkowe zatrzymanie robota w cyklu pracy automatycznej, krokowej lub w czasie synchronizacji,

SYN (synchronizacja) - synchronizacja robota

Grupa 2.

POZ (instrukcja pozycjonowania) - programowanie instrukcji pozycjononowania,

INS (instrukcje inne) - programowanie innych instrukcji niz instrukcja pozycjonowania. Wyboru tych instrukcji dokonujemy "przewijajac" menu wlasciwe dla tego przycisku.

EDY (edycja) - przegladanie i modyfikowanie programow uzytkowych (D).

Grupa 3.

1 - cyfra 1,

2 - cyfra 2,

.

.

9 - cyfra 9,

. - kropka dziesietna,

0 - cyfra 0.

Przyciski z tej grupy sluza do wpisywania wartosci numerycznych podczas tworzenia lub poprawiania programu uzytkowego.

Grupa 4

A - litera A (D),

B - litera B (D),

C - litera C (D),

D - litera D (D),

E - litera E (D),

F - litera F (D),

G - litera G (D),

H - litera H (D),

K - litera K (D),

L - litera L (D).

Grupa przyciskow okreslana nazwa MODULY PNEUMATYCZNE sluzy do deklarowania i poruszania modulami pneumatycznymi oraz do tworzenia instrukcji pozycjonowania tych modulow. Diody luminescencyjne umieszczone obok kazdego z tych przyciskow informuja, ktory przycisk jest aktywny. Kolejne litery okreslaja kolejne moduly w przypadku modulow dwustanowych. Natomiast w przypadku modulow trzystanowych jednemu modulowi odpowiada para kolejnych przyciskow (np.: "A"-"B", "C"-"D").

Grupa 5.

I+ - modul elektryczny nr 1, przesuniecie w gore (prawo),

I- - modul elektryczny nr 1, przesuniecie w dol (lewo),

II+ - modul elektryczny nr 2, przesuniecie w gore (prawo),

II- - modul elektryczny nr 2, przesuniecie w dol (lewo),

III+ - modul elektryczny nr 3, przesuniecie w gore (prawo),

III- - modul elektryczny nr 3, przesuniecie w dol (lewo),

Grupa przyciskow pod nazwa MODULY ELEKTRYCZNE sluzy do porusznia czescia manipulatora z silnikami elektrycznymi oraz do tworzenia instrukcji pozycjonowania. Przyciskow jest szesc, gdyz kazdy z modulow moze poruszac sie w dwie strony (gora-dol lub prawo-lewo).

Grupa 6.

LEW (przesun tekst w lewo) - przesuwanie tekstu o jedna pozycje w lewo.

PRA (przesun tekst w prawo) - przesuwanie tekstu o jedna pozycje w prawo.

WPI (wpisz) - zakonczenie wprowadzania parametru numerycznego lub zakonczenie konstruowania instrukcji,

KAS (kasuj) - skasowanie omylkowo wpisanego znaku numerycznego lub potwierdzenie przyjscia komunikatu o bledzie,

Grupa 7.

PF1 - przycisk funkcyjny 1,

PF2 - przycisk funkcyjny 2,

PF3 - przycisk funkcyjny 3.

Grupa przyciskow pod nazwa PRZYCISKI ZMIENNE sluzy do wyboru

11

funkcji opisanych na dolnym wyswietlaczu.
Grupy przyciskow wraz z wyswietlaczami alfanumerycznymi wykorzystywane sa do komunikowania sie z robotem na zasadzie dialogu. Wcisniecie jednego z przyciskow powoduje ukazanie sie na dolnym wyswietlaczu menu funkcji zwiazanych z tym przyciskiem. Wybor funkcji dokonywany jest przyciskami zmiennymi znajdujacymi sie pod dolnym wyswietlaczem.
Przyciski w grupach 4, 5, 7 sa wykorzystywane do recznego operowania manipulatorem, a takze spelniaja funkcje pomocnicze. Przyciski z grupy 3 stanowia klawiature numeryczna i sa wykorzystywane podczas dialogu do wprowadzania wartosci liczbowych.

2.2. Panel operacyjny.

Panel operacyjny jest integralna czescia szafy sterowniczej robota wyposazona w szereg przyciskow i lampek pozwalajacych operatorowi na wykonanie podstawowych czynnosci zwiazanych z obsluga robota. Oznaczenia odnosnie poszczegolnych przyciskow i lampek stosowane tekscie (zawarte miedzy znakami "") oraz krotki opis realizowanych przez nie funkcji zamieszczono ponizej.

ZAS (zasilanie) - lampka sygnalizujaca podlaczenie zasilania do szafy sterowniczej,

GOT (gotowosc, jest lampka-przycisk) - swiecenie sie sygnalizuje, ze tylko czesc cyfrowa sterownika jest podlaczona do zasilania, wcisniecie powoduje podlaczenie do zasilania tylko czesci cyfrowej sterownika,

PRA (praca, jest to lampka-przycisk) - swiecenie sie sygnalizuje, ze caly sterownik podlaczony jest do zasilania, wcisniecie (jesli swieci sie lampka GOT) powoduje podlaczenie do zasilania wszystkich elementow sterownika,

KAS (lampka-przycisk) - swiecenie sie sygnalizuje wystapienie stopu awaryjnego, a nacisniecie likwiduje stop awaryjny,

AWA - przycisk stopu awaryjnego,

STR (start, jest to lampka-przycisk) - swiecenie sie swiatlem migajacym sygnalizuje, ze robot jest w trakcie wykonywania instrukcji czekania, swiecenie sie swiatlem ciaglym oznacza, ze robot wykonuje instrukcje inna niz instrukcja czekania, wcisniecie przy zgaszonej lampce powoduje wystartowanie

wykonywania programu od pierwszej instrukcji programu glownego,

STP (stop, jest to przycisk) - wcisniecie powoduje zatrzymanie synchronizacji lub automatycznego wykonywania programu,

PK3 (ladowanie programu z PK3, jest to lampka-przycisk) - swiecenie sie sygnalizuje, ze trwa przepisywanie programu z pamieci kasetowej PK3 do pamieci programow uzytkowych, wcisniecie inicjuje przepisywanie z PK1 pamieci sterownika,

UTR (utrata programu, jest to lampka) - swiecenie sygnalizuje, ze w pamieci programow nie ma zadnego programu,

BLA (blad, jest to lampka) - swiecenie sie swiatlem migajacym sygnalizuje wykrycie przez system sytuacji blednej,

TEM (przegrzanie, jest to lampka) - swiecenie oznacza, ze nadmiernie wzrosla temperatura w szafie sterowniczej.

3. OGOLNE ZASADY POSLUGIWANIA SIE PANELEM PROGRAMOWANIA.

3.1. Wprowadzanie parametrow numerycznych.

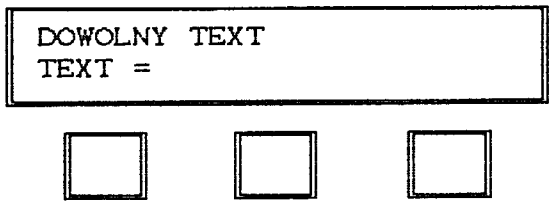
W trakcie komunikacji operatora z ukladem sterowania robota czesto pojawia sie koniecznosc wprowadzania do systemu parametrow numerycznych. Sytuacja taka moze miec miejsce na przyklad w trakcie tworzenia programu uzytkowego, przy edycji programow uzytkowych, podczas definiowania lub zmieniania parametrow pracy robota itp.

Wprowadzenie parametrow numerycznych moze byc:

- obligatoryjne.
- opcjonalne.

3.1.1. Obligatoryjne wprowadzanie wartosci numerycznej.

Obligatoryjne wprowadzanie wartosci numerycznej oznacza, ze jedynie poprawne wykonanie tej czynnosci pozwala na przejscie do dalszych operacji. Koniecznosc obligatoryjnego wprowadzenia wartosci numerycznej sygnalizowana jest operatorowi wyswietleniem na dolnym wyswietlaczu pewnego tekstu zakonczonego znakiem "=" (rownosc) np:



W tym momencie wykorzystujac przyciski numeryczne (grupa 3) nalezy wpisac do dolnej linii wyswietlacza zadana wartosc. Liczba bedzie wpisywana za znakiem "=" (rownosc). Jezeli w trakcie wpisywania operator uzna, ze ostatnio wprowadzony znak lub cala liczba jest bledna, moze wymazac wszystkie wprowadzone przez niego znaki przez nacisniecie przycisku KAS (kasuj). Po skasowaniu zlej liczby, operator moze przystapic do wprowadzania nowej wartosci parametru. Koniec wprowadzania nastepuje po wcisnieciu przycisku WPI (wpisz). W trakcie wprowadzania i po wprowadzeniu liczby dokonywany jest szereg sprawdzen. Po wykryciu nieprawidlowosci sygnalizowany jest blad. Czesc sprawdzen dotyczy przestrzegania zasad obowiazujacych przy wprowadzaniu wszystkich parametrow numerycznych. Jako bledne sygnalizowane sa sytuacje, gdy:

- znak minus "-" nie jest pierwszym znakiem wprowadzanej liczby,
- wprowadzany jest znak minusa "-" lub kropki "." do liczby, do ktorej odpowiednio znak minusa "-" lub kropki "." juz uprzednio wprowadzono,
- znak kropki "." jest pierwszym wprowadzonym znakiem.

Pozostale sprawdzenia dotycza przestrzegania regul wprowadzania okreslonego parametru numerycznego. Sygnalizacja sytuacji blednych bedzie zalezala od typu tego parametru i pojawi sie w nastepujacych okolicznosciach:

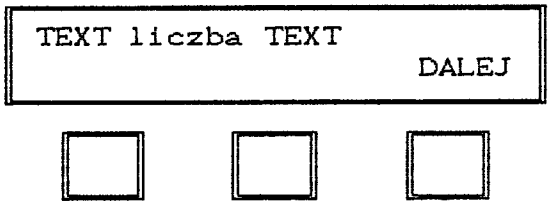
- wprowadzenie wiekszej niz dopuszczalna liczby znakow,
- proba wprowadzenia zbyt duzej liczby cyfr po kropce dziesietnej,
- proba wprowadzenia kropki "." lub minusa "-" dla liczby, ktora takich znakow nie moze zawierac,
- proba wprowadzenia zbyt malej lub zbyt duzej wartosci.

3.1.2. Opcjonalne wprowadzanie wartosci numerycznej.

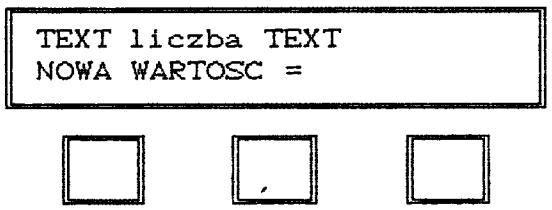
Opcjonalne wprowadzenie wartosci numerycznej oznacza pozostawienie operatorowi wyboru co do koniecznosci wprowadzenia wartosci numerycznej. Sytuacja taka sygnalizowana jest wyswietleniem

14

nastepujacego tekstu:



"TEKST" oznacza tekst wynikajacy z danej sytuacji. "liczba" jest domniemana wartoscia numeryczna, np. wprowadzona uprzednio, wyrozniajaca sie z posrod innych znakow tekstu mruganiem. Jesli wyswietlona (mrugajaca) wartosc numeryczna odpowiada operatorowi, to naciska on przycisk DALEJ, co powoduje przejscie do realizacji kolejnych operacji. Jesli operator chce zmienic ta wartosc, to naciska dowolny przycisk klawiatury numerycznej (grupa 3), co powoduje wyswietlenie sie:



Dalsze wprowadzanie wartosci numerycznej odbywa sie na takich samych zasadach jak dla wprowadzania obligatoryjnego. Jedyna roznice stanowi nacisniecie przycisku KAS (kasuj), po ktorym nie wraca sie juz do w/w postaci wyswietlacza, domniemuje sie, ze operator zrezygnowal z wprowadzania nowej wartosci numerycznej.

3.2. Podglad tekstow instrukcji.

Panel programowania wyposazony jest w dwa alfanumeryczne wyswietlacze o dlugosci 24 znakow kazdy. Wyswietlacz gorny przeznaczony jest glownie do wyswietlania tekstow instrukcji i komunikatow dla operatora. Teksty instrukcji roznia sie miedzy soba dlugoscia i zawieraja od kilku do max. 72 znakow. Oznacza to, ze teksty niektorych instrukcji nie mieszcza sie w okienku wyswietlacza. Istnieje jednak mozliwosc przesuwania okienka po tekscie instrukcji. Jezeli z lewej strony wyswietlanego tekstu znajduje sie strzalka w lewo "<-", to znaczy ze mozna przesunac okienko w lewo (na lewo od strzalki znajduja sie znaki). Jezeli strzalka w prawo "->" znajduje sie z prawej strony wyswietlanego tekstu, to znaczy, ze okienko mozna przesunac w prawo (na prawo od strzalki znajduja sie znaki).

Przesuniecia okienka w lewo dokonujemy przyciskiem LEW (przesun w lewo), aw prawo przyciskiem PRA (przesun w prawo). Przycisniecie i puszczenieprzycisku powoduje przesuniecie okienka o jeden znak w lewo/prawo (znieznacznym opoznieniem). Przy ciaglym nacisnieciu przycisku okienko przesuwasie po jednym znaku w lewo/prawo z niewielka szybkoscia, dopoki przycisk niezostanie zwolniony lub okienko nie osiagnie lewego/prawego konca tekstu.

UWAGA:

Strzalki w tekscie pokazuja kierunek, w ktorym mozna przesunac okienko nad tekstem, a nie kierunek przesuwania tekstu pod okienkiem. Okienko moze byc przesuwane nad tekstem tylko wtedy, gdy na 1-szym lub 24-tym znaku znajduje sie strzalka i tylko w kierunku wskazanym przez strzalke(i).

3.3. Przegladanie menu.

Dolny wyswietlacz panelu programowania przeznaczony jest miedzy innymi do wyswietlania opisow znajdujacych sie pod nim przyciskow funkcyjnych. Opisy te tworza tak zwane menu. Menu moze zawierac wiecej niz trzy elementy. Musi wiec istniec mozliwosc "dostania sie" do pozostalych jego elementow, tzn. tych niewidocznych aktualnie na wyswietlaczu. Do tego celu sluzy przycisk MEN (menu). Przycisk ten jest aktywny, gdy menu ma wiecej niz 3 elementy, a jest to sygnalizowane swieceniem sie diody znajdujacej sie obok przycisku. Tak wiec tylko gdy dioda jest zapalona mozliwe jest przegladanie menu. Kazde nacisniecie przycisku powoduje wyswietlenie kolejnej trojki opisow. Ostatnia trojka moze zawierac mniej niz trzy opisy. Po wyswietleniu ostatniej trojki nastepne nacisniecie przycisku powoduje wyswietlenie pierwszej trojki, tzn. przegladanie jest cykliczne.
Po znalezieniu odpowiedniego opisu operator dla zrealizowania funkcji zwiazanej z tym opisem powinien nacisnac znajdujacy sie pod nim przycisk funkcyjny.

3.4. Stany pracy panelu programowania.

Panel programowania moze znajdowac sie (z punktu widzenia operatora) w dziewieciu stanach pracy:

1. Programowanie innych instrukcji
2. Programowanie instrukcji pozycjonowania
3. Synchronizacja robota

4. Edycja programu
5. Reczne operowanie systemem
6. Praca automatyczna
7. Blad
8. Reczne operowanie manipulatorem

Szczegolowe zestawienie stanow pracy panelu programowania z informacjami jak rozpoznac dany stan i jak do niego przejsc podano w tablicy stanow panelu programowania zamieszczonej ponizej.

Tablica stanow panelu programowania.

Numer stanu	Po czym mozna rozpoznac, ze panel programowania jest w tym stanie	Jak przejsc do tego stanu:	
		z panelu programowania	z panelu operacyjnego
1	Odpowiednie menu na wyswietlaczach	1. Nacisniecie przycisku INS 2. Po skasowaniu bledu (1) 3. Nacisniecie przycisku PRZ (2)	Niemozliwe
2	Odpowiednie menu na wyswietlaczach	1. Nacisniecie przycisku POZ 2. Po skasowaniu bledu (1) 3. Nacisniecie przycisku PRZ (2)	Niemozliwe
3	Zapalona dioda przy przycisku EDI	1. Nacisniecie przycisku EDI 2. Po skasowaniu bledu (1) 3. Nacisniecie przycisku PRZ (2)	Niemozliwe
4	Odpowiednie menu na wyswietlaczach	1. Nacisniecie przycisku SYN 2. Po skasowaniu bledu (1) 3. Nacisniecie przycisku PRZ (2)	Niemozliwe

Tablica stanow panelu programowania cd.

Numer stanu	Po czym mozna rozpoznac, ze panel programowania jest w tym stanie	Jak przejsc do tego stanu:	
		z panelu programowania	z panelu operacyjnego
5	Zapalona dioda przy przycisku OPE	1. Nacisniecie przycisku OPE 2. Po skasowaniu bledu (1) 3. Nacisniecie przycisku PRZ (2)	Niemozliwe
6	Odpowiednie menu na wyswietlaczach oraz swiecaca sie lampka STR na panelu operacyjnym	1. Nacisniecie przycisku AUT 2. Po skasowaniu bledu (1) 3. Nacisniecie przycisku PRZ (2)	Niemozliwe
7	Zapalona lampka BLA na panelu operacyjnym i tekst informujacy o bledzie na wyswietlaczu	Wykrycie blednej sytuacji przy operowaniu z panelu programowania	Wykrycie bledne sytuacji przy przy operowaniu z panelu operacyjnego lub prz pracy automatycznej
8	Odpowiednie menu na wyswietlaczach	1. Nacisniecie przycisku PRZ 2. Po skasowaniu bledu (1)	Niemozliwe

UWAGI:

(1) Przejscie do tego stanu po skasowaniu bledu nastepuje tylko wtedy, gdy przed wystapieniem bledu panel programowania byl w tym stanie i blad nie byl spowodowany bledami w komunikacji.

(2) Przejscie do tego stanu po nacisnieciu przycisku PRZ mozliwe jest tylko wtedy, gdy panel znajdowal sie w tym stanie.

W trakcie operowania robotem istnieje koniecznosc przechodzenia z jednego stanu pracy do drugiego. Nie wszystkie przejscia sa jednak dozwolone. Wszystkie mozliwe przejscia zestawiono ponizej.

Tablica mozliwych przelaczen stanow panelu programowania:

+ oznacza przejscie mozliwe do zrealizowania

- oznacza przejscie nie dajace sie zrealizowac

STAN PRZED PRZELACZENIEM	STAN PO PRZELACZENIU							
	1	2	3	4	5	6	7	8
1	+	+	-	+	+	+	+	+
2	+	+	-	+	+	+	+	+
3	+	+	+	-	+	+	+	+
4	+	+	+	+	+	+	+	+
5	+	+	+	+	+	+	+	+
6	-	-	-	-	-	-	+	+
7	+(1)	+(1)	+(1)	+(1)	+(1)	-	+	-
8	+(2)	+(2)	+(2)	+(2)	+(2)	-	+	+

UWAGI:

(1) Przejscie do tego stanu jest tylko mozliwe wtedy, gdy panel znajdowal sie w nim przed wystapieniem bledu (przejscie w stan 7) i jesli blad nie byl spowodowany bledami w komunikacji miedzy panelem a sterownikiem.

(2) Przejscie do tego stanu jest tylko mozliwe po nacisnieciu przycisku PRZ o ile przed nacisnieciem tego przycisku panel byl w tym stanie.

Po wejsciu do ktoregos z w/w stanow w dolnym wyswietlaczu pojawia sie pierwszy fragment menu. Zawartosc gornego wyswietlacza zalezna jest od numeru stanu. Teraz operator przez nacisniecie wybranego przycisku funkcyjnego, zgodnie z opisem znajdujacym sie nad tym przyciskiem, moze zainicjowac jakas akcje.

W stanach zwiazanych z programowaniem instrukcji obowiazuja nastepujace zasady automatycznej zmiany numeru aktualnie programowanej instrukcji, przy przejsciu do programowania nowej instrukcji:

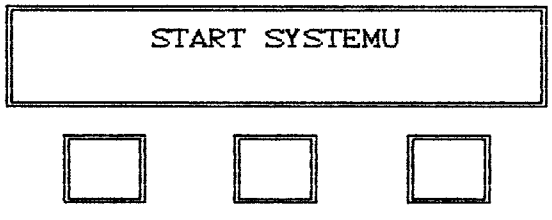
- numer instrukcji nie ulega zmianie, tzn. jest taki jak przed realizacja funkcji wybranej z menu glownego, gdy instrukcja opatrzona tym numerem nie byla poprzednio zaprogramowana,

- numer instrukcji ulegnie zmianie na nastepny, jesli wyswietlona w gornym rzedzie instrukcja byla poprzednio zaprogramowana, tzn. bezposrednio po jej zaprogramowaniu zaczeto programowac nastepna instrukcje (nie bylo przejscia do innego stanu). Zmiana numeru na nastepny oznacza wypisanie numeru instrukcji o numerze wiekszym od wyswietlonego i podzielnym przez 10, jesli poprzednio zaprogramowana instrukcja miala najwiekszy numer w programie. W przeciwnym przypadku nowa instrukcja otrzymuje numer posredni pomiedzy poprzednim, a nastepnym podzielnym przez 10.

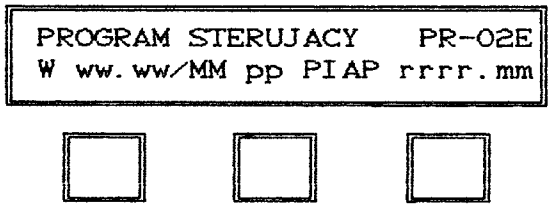
4. ROZPOCZECIE PRACY

4.1. Restart systemu po wlaczeniu zasilania.

Dolaczenie do szafy sterowniczej zasilania nastepuje po wcisnieciu i przekreceniu w prawo kluczyka w stacyjce znajdujacej sie po prawej stronie wneki na panel programowania. Bedzie to zasygnalizowane zaswieceniem sie zoltej lampki nad stacyjka.
Zasilenie ukladow elektronicznych systemu sterowania uzyskuje sie po wlaczeniu przycisku GOT (gotowosc) na panelu operacyjnym. Wtedy tez zostaje zasilony panel programowania. Na wyswietlaczach panelu programowania ukaze sie tekst:



Po okolo 2 sekundach, jezeli istnieje komunikacja miedzy panelem programowania a sterownikiem, zostanie wyswietlona informacja o programie sterujacym:



gdzie:

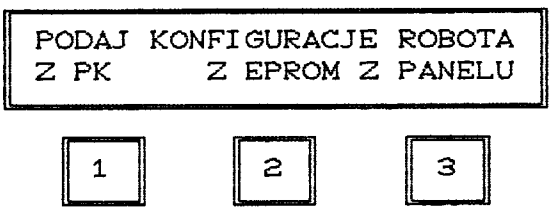
ww.ww - numer wersji programu sterujacego,

pp - typ pakietu mikroprpcesora (16,86),

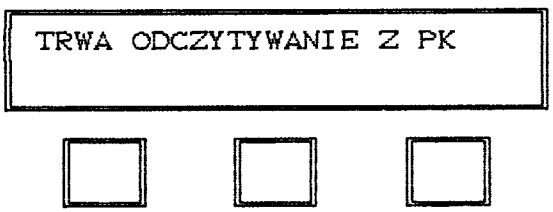
rrrr.mm - rok (rrrr) i miesiac (mm) utworzenia wersji programu sterujacego.

4.2. Okreslanie konfiguracji robota.

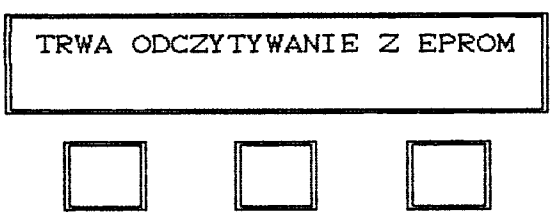
Poniewaz robot PR-02 bedzie mial rozne konfiguracje modulow pneumatycznych program sterujacy bedzie umozliwial zmiane konfiguracji robota przez uzytkownika. Z tego powodu po okolo 2 sekundach pojawi sie na wyswietlaczach nastepujacy napis:



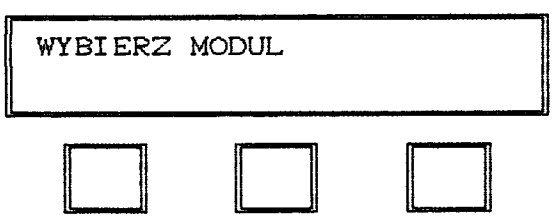
Wybranie wariantu nastepuje po wcisnieciu jednego z trzech przyciskow. Po wybraniu przycisku "1" nastepuje uruchomienie pamieci kasetowej i odczyt konfiguracji

1. 

po zakonczeniu wczytywania konfiguracji nastepuje przejscie do synchronizacji robota.
Po wybraniu przycisku "2" nastepuje odczyt konfiguracji z pamieci EPROM oraz pojawia sie komunikat:

2. 

po zakonczeniu wczytywania konfiguracji nastepuje przejscie do synchronizacji robota.
Po wybraniu przycisku "3" uaktywniaja sie przyciski modulow pneumatycznych, a na gornym wyswietlaczu pojawia sie napis:

3. 

Wcisniecie dowolnego przycisku oznaczonego literami spowoduje wyswietlenie na gornym wyswietlaczu nazwy (litery) wybranego modulu, a na dolnym menu pojawi sie menu w dwoch zestawach (4),

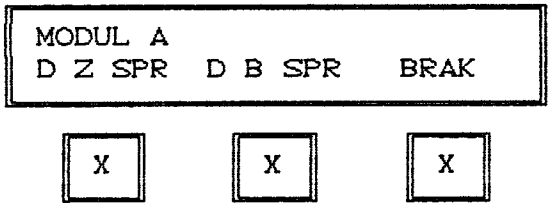
21

(5):

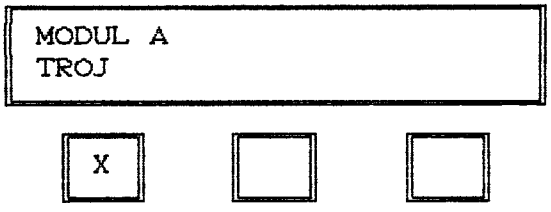
a) D Z SPR - dwupolozeniowy ze sprzezeniem
D B SPR - dwupolozeniowy bez sprzezenia
BRAK - brak modulu

b) TROJ - trojpolozeniowy

4.

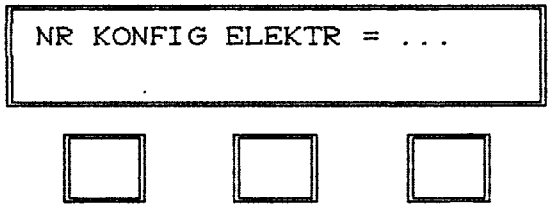


5.

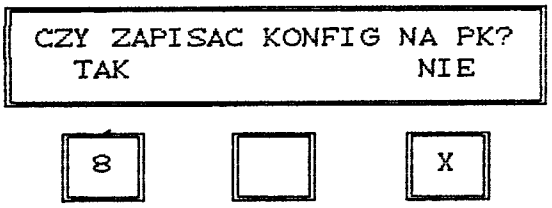


Wybierajac jeden z mozliwych czterech wariantow poprzez jednokrotne wcisniecie przycisku okreslamy czy dany modul istnieje i jesli tak, to jakiego rodzaju. Przycisniecie spowoduje rowniez zapalenie sie diody przy deklarowanym module. Nalezy w tym miejscu zaznaczyc, ze przy deklaracji modulu trojpolozeniowego zapalaja sie diody przy dwoch kolejnych przyciskach. Z tego wzgledu moduly trojpolozeniowe moga byc deklarowany jedynie jako: "A", "C", "E", "G", "K". Po deklaracji kazdego modulu nastepuje powrot do punktu (3) i powtarza sie opisana powyzej procedura deklarowania modulow, az do momentu gdy wszystkie moduly beda zadeklarowane. Po zadeklarowaniu modulow pneumatycznych pojawi sie na gornym wyswietlaczu nastepujacy text:

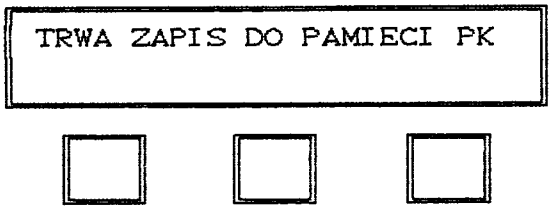
6.



Korzystajac z grupy przyciskow numerycznych nalezy wpisac numer odpowiedni dla aktualnej konfiguracji. Numery konfiguracji modulow elektrycznych odpowiadajace wszystkim mozliwym kombinacjom trzech roznych modulow znajda sie w dokumentacji technicznej.
Zadeklarowanie numeru konfiguracji modulow elektrycznych konczy proces deklarowania konfiguracji z panelu programowania. Wowczas pojawi sie na wyswietlaczy nastepujace pytanie:

7. 

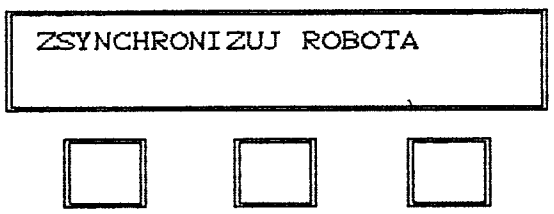
Wcisniecie przycisku "TAK" spowoduje zapisanie do PK informacji (na nowej tasmie lub w miejsce starego zapisu) potwierdzonej napisem na wyswietlaczu:

8. 

Po zakonczeniu pracy PK lub wcisnieciu przycisku "NIE" mozna przogram sterujacy przechodzi do synchronizacji robota.

4.3. Synchronizacja robota.

Zgloszenie sie napisu:



traktowac mozna jako faktyczny restart systemu. W tej chwili uaktywniony zostanie dodatkowo przelacznik predkosci, jak rowniez przyciski MODULY ELEKTRYCZNE, stop - "STP", synchronizacja - "SYN", oraz przelacznik na reczne operowanie manipulatorem - "PRZ", potrzebne do synchronizacji. Pozostale przyciski pozostana nieaktywne, w tym przyciski MODULY PNEUMATYCZNE mimo uzycia przycisku "PRZ". Poruszanie reczne manipulatorem przy pomocy przyciskow modulami robota mozliwe jest po wczesniejszym wcisnieciu przycisku "PRA" znajdujacym sie na panelu operacyjnym, czyli proces synchronizacji musi byc poprzedzony uzyciem tego przycisku.

Synchronizacja informuje program sterujacy, w jakiej pozycji znajduja sie moduly manipulatora po wznowieniu pracy, czyli praktycznie po ponownym wlaczeniu zasilania i przed wykonaniem jakiegokolwiek ruchu zdeterminowanego programem roboczym wczytanym z PK. W przypadku robota PR-02E zsynchronizowane musza byc moduly

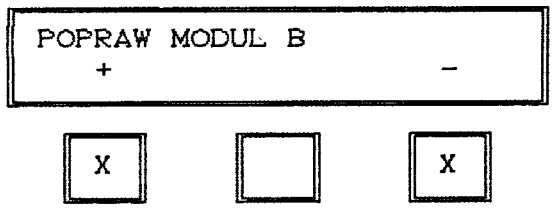
elektryczne i pneumatyczne. Procesowi synchronizacji podlegaja najpierw moduly pneumatyczne, a nastepnie elektryczne.
Pojecie synchronizacji w przypadku modulow pneumatycznych ma sens tylko dla tych, ktore posiadaja przyciski sprzezenia zwrotnego. Uruchomienie synchronizacji od strony obslugi polegac bedzie na wcisnieciu przycisku "SYN" znajdujacym sie na panelu programowania. Program natomiast najpierw odczyta stan modulow pneumatycznych. Kazde zsynchronizowanie osi pneumatycznej ze sprzezeniem polega na podaniu zasilania na wlasciwa cewke elektrozaworow sterujacych ruchem tego modulu. Po zakonczeniu synchronizacji modulow pneumatycznych ze sprzezeniem uzytkownik musi ustalic jeszcze polozenia modulow pneumatycznych bez sprzezenia. Zalozmy, ze modulowi pneumatycznemu bez sprzezenia przyporzadkowano nazwe "H". Wowczas po synchronizacji modulow pneumatycznych ze sprzezeniem na wyswietlaczu pojawi sie komunikat:

1.

ZORIENTUJ MOD. H BEZ SPR		
+		-
X		X

Uzytkownik musi wizualnie sprawdzic, w ktorym z dwoch stanow znajduje sie modul "H" i wcisnac wlasciwy przycisk "+" lub "-". Po zorientowaniu modu zostanie podane zasilanie elektryczne na wlasciwa cewke elektrozaworow sterujacych ruchem tego modulu. Procedura synchronizacji modulow bez sprzezenia (1) powtarza sie, az wszystkie zadeklarowane moduly tego typu beda synchronizowane.
Pozostaje jeszcze do rozwazenia sytuacja, gdy moduly pneumatyczne robota w poprzednim seansie pracy z powodu awarii (np. zanik cisnienia powietrza lub zasilania pradem elektrycznym) zatrzymaja sie w pozycji nieokreslonej. Wtedy program synchronizacji (zaraz po wlaczeniu przycisku "SYN") wykryje taki stan, gdyz zaden z wylacznikow koncowych nie bedzie zalaczony (w modulach ze sprzezeniem zwrotnym). Fakt ten bedzie zasygnalizowany pojawieniem sie komunikatu informujacego o awari danego modulu. Zalozmy ze awaria dotyczy modulu "B". Na wyswietlaczu pojawi sie wtedy nastepujacy komunikat:

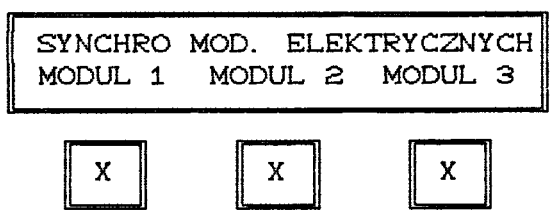
2.



Wciskajac wybrany przycisk zmienny "+" lub "-" powoduje przestawienie modulu na wybrana pozycje. Modul dochodzac do puntku synchronizacji zalacza czujnik ograniczenia (sprzezenia zwrotnego) inicjujac dalsza synchronizacje modulow pneumatycznych, az do momentu wykrycia w kolejnym module ewentualnej usterki i wtedy program synchronizacji przechodzi do punktu (2) lub do momentu zsynchronizowania wszystkich modulow ze sprzezeniem zwrotnym. Nastepnie nastepuje synchronizacja modulow bez sprzezenia zwrotnego (1).

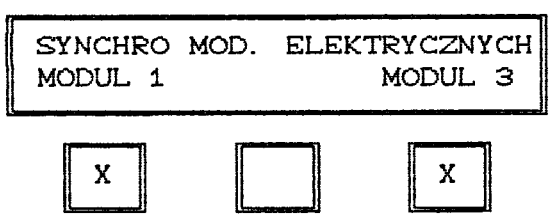
Po zsynchronizowaniu modulow pneumatycznych nastepuje przejscie do synchronizacji modulow elektrycznych. Na wyswietlaczu pojawi sie nastepujacy komunikat:

3.



Na dolnym wyswietlaczu pojawi sie opis tylko tych modulow elektrycznych, ktore byly zadeklarowane i ktore wystepuja faktycznie. (tzn. jezeli zadeklarowane sa jedynie moduly nr 1 i 3 to postac wyswietlacza bedzie wygladala nastepujaco (4)).

4.



Napis na gornym wyswietlaczu miga w przypadku braku synchronizacji modulow elektrycznych. Jezeli chcemy zsynchronizowac np. modul 1, to wciskamy pierwszy przycisk zmienny. Wowczas modul elektryczny nr 1 zaczyna poruszac sie do czujnika umiejscowionego w polowie zakresu jego ruchu. W tym czasie napis na gornym wyswietlaczu jest wyswietlany w sposob ciagly. Gdy modul 1 dotrze do czujnika i zsynchronizuje sie, to zgasnie rowniez napis "MODUL 1" na dolnym wyswietlaczu.

Potem w podobny sposob synchronizujemy pozostale moduly nie koniecznie zachowujac kolejnosc numeracji modulow. Synchronizacja kolejnego modulu sygnalizowana jest na panelu programowania zawsze ciaglym swieceniem napisu na gornym wyswietlaczu, zas koniecznosc zainicjowania tej czynnosci miganiem tego napisu. Dolny wyswietlacz informuje o modulach jeszcze niezsynchronizowanych. Proces synchronizacji modulu elektrycznego mozemy zawsze przerwac wciskajac w dowolnej chwili przycisk "STP" (stop). Kontynuacja synchronizacji nastepuje po ponownym wcisnieciu jednego z przyciskow zmiennych (nie koniecznie tego samego).
Proces synchronizacji modulow elektrycznych nalezy poprzedzic wyprowadzeniem poszczegolnych modulow z punktow synchronizacji tak, aby w trakcie jej trwania widoczny byl ruch w kierunku tego punktu. W tym celu poslugiwac sie mozna pokretlem zmiany predkosci i przyciskami MODULY ELEKTRYCZNE, ktore sa aktywne w fazie synchronizacji po wcisnieciu przycisku "PRZ" (przelacz). Oczywiscie chodzi tu o ustawienie modulow w polozeniu bliskim punku synchronizacji.

UWAGA:
Podczas ruchu przed dokonaniem synchronizacji wskazana jest szczegolna ostroznosc w manipulowaniu robotem, gdyz system nie kontroluje przekroczenia ograniczen. Przekroczenie ograniczen moze spowodowac wlaczenie sie stopu awaryjnego.

Proces synchronizacji konczy etap przygotowywania robota do pracy. Zakonczenie synchronizacji uaktywnia wszystkie przyciski z panelu programowania. Koniec synchronizacji zasygnalizuje pojawienie sie napisow dotyczacych startu programu. Jezeli w pamieci uzytkowej robota jest program, to zostanie wyswietlona jego pierwsza instrukcja. W przeciwnym wypadku w gornej linii wyswietlacza wyswietli sie liczba 10.

5. TWORZENIE PROGRAMU.

5.1. Instrukcja.

Program uzytkowy tworzony jest z instrukcji. Kazda instrukcja ma swoj numer, typ oraz parametry (o ile sa potrzebne).

5.1.1. Numer instrukcji.

Kazda instrukcja ma swoj numer. Numer instrukcji, moze byc dowolna liczba calkowita z przedzialu od 1 do 9999. W trakcie programowania kolejnych instrukcji ich numery automatycznie zwiekszane sa o 10. Na przyklad, jesli po zaprogramowaniu instrukcji o numerze 330 przechodzimy do programowania nastepnej instrukcji, to bedzie ona miala numer 340. Przerwanie programowania instrukcji przez przejscie do innego stanu niz 1 lub 2 spowoduje, ze po ponownym przejsciu do programowania, nowo programowana instrukcja bedzie miala numer taki jaki ostatnio byl na wyswietlaczu. Dowolny numer mozemy nadac instrukcji przez wpisanie go przy uzyciu klawiatury numerycznej (patrz stan 3 - edycja).

5.1.2. Typ instrukcji.

Za numerem instrukcji znajduje sie mnemonik okreslajacy typ instrukcji. Mnemonik oddzielony jest spacja od numeru i od parametrow instrukcji. W trakcie pracy automatycznej wyswietlany jest jedynie numer i mnemonik aktualnie wykonywanej instrukcji.

5.1.3. Parametry instrukcji.

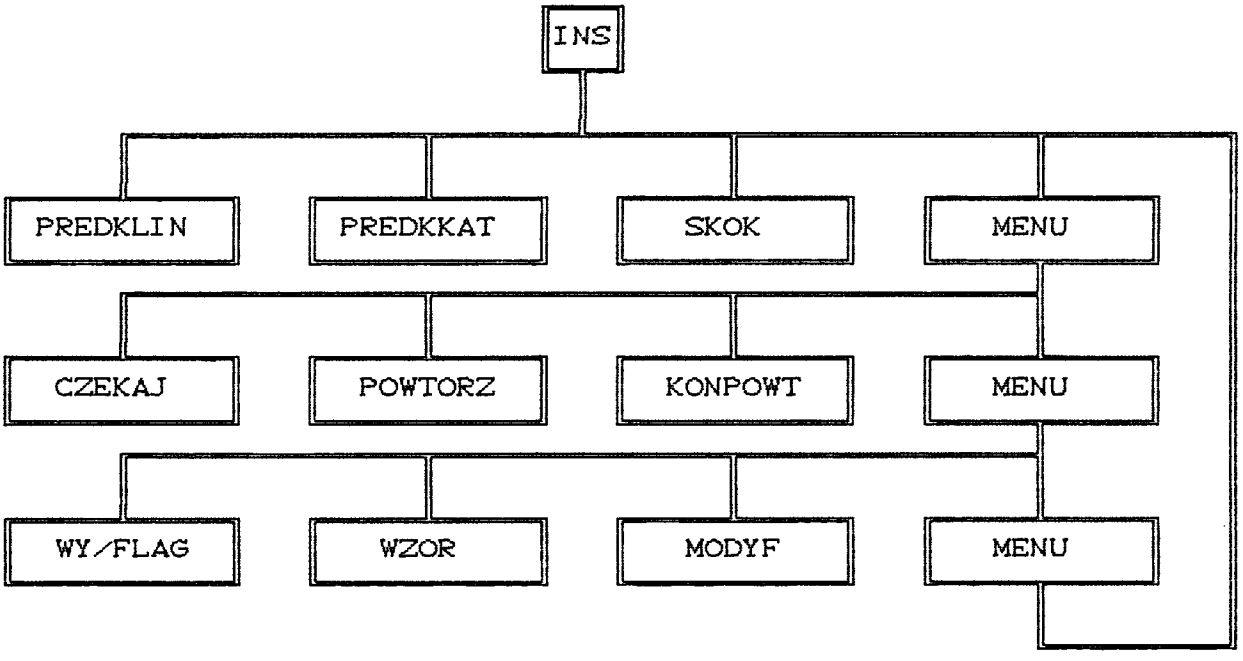
Za mnemonikiem instrukcji znajduja sie parametry instrukcji. Nie wszystkie instrukcje maja parametry, a decyduje o tym typ instrukcji. Parametry moga byc numeryczne i nienumeryczne.

5.2. Programowanie.

Programowanie rozpoczyna sie od wybrania typu instrukcji. W przypadku programowania instrucji poycjonowania nastepuje to przez nacisniecie przycisku POZ, a w przypadku innych instrukcji, przez wybranie odpowiedniego typu z menu zwiazanego z przyciskiem INS. Wybranie typu powoduje wpisanie na pierwszej pozycji numeru instrukcji a nastepnie jej typu. Jesli poprzednio wyswietlony numer instrukcji byl numerem "swiezo" zaprogramowanej instrukcji, to numer aktualnie programowanej instrukcji bedzie najblizszym, wiekszym numerem podzielnym przez 10. Jesli programowanie instrukcji rozpoczelo sie po przejsciu z innych stanow niz 1 lub 2, to instrukcja bedzie miala numer taki, jaki byl wyswietlony przed wybraniem typu instrukcji.

5.2.1. Programowanie innych instrukcji.

Menu dla stanu nr 1 tzn.: "programowanie innych instrukcji" jest nastepujace:



Znaczenie poszczegolnych pozycji menu jest nastepujace:

PREDKLIN - zmiana predkosci liniowej maksymalnej i podstawowej dla modulow elektrycznych,

PREDKKAT - zmiana predkosci katowej maksymalnej i podstawowej dla modulow elektrycznych,

SKOK - skok bezwarunkowy i warunkowy,

CZEKANIE - czekanie bezwarunkowe i warunkowe,

POWTORZ - poczatek petli programowej,

KONPOWT - koniec petli programowej,

WY/FLAG - zmiana stanu wyjsc lub flag,

WZOR - definicja wzoru,

MODYF - definicja modyfikacji.

5.2.1.1. Instrukcja zmiany predkosci liniowej maksymalnej i podstawowej.

W przypadku modulow elektrycznych charakteryzujacych sie ruchem liniowym dla instrukcji pozycjonowania nalezy okreslic predkosc katowa. Predkosc w instrukcji pozycjonowania deklaruje sie jako procent predkosci podstawowej. Predkosc liniowa podstawowa i maksymalna moze byc zmieniana w dowolnym miejscu programu instrukcja PREDKLIN. Podczas wykonywania instrukcji pozycjonowania

sprawdzane jest czy predkosc ruchu nie jest wieksza od aktualnie zadeklarowanej predkosci maksymalnej.
Predkosc podstawowa i maksymalna podaje sie w milimetrach na sekunde (mm/s), a zakres zmiennosci jest od 1 do 1000 mm/s. Predkosc podstawowa musi byc przy tym nie wieksza niz maksymalna. Zalozmy, ze poprzednio zaprogramowana instrukcja byla instrukcja skoku bezwarunkowego o numerze 10:

1.

```
10 SKOK DO 1000
PREDKLIN PREDKKAT  SKOK
```

2	X	X

2.

```
20 PREDKLIN MAX
PREDKLIN MAX MM/S =
```


Po wprowadzeniu predkosci maksymalnej (liczba od 1 do 1000, np. 500) nastepuje przejscie do punktu 3.

3.

```
<N MAX = 500, PREDKLIN =
PREDKLIN MM/S =
```


Po wprowadzeniu predkosci podstawowej (liczba od 1 do predkosci max.) np. 300 nastepuje przejscie do punktu 4.

4.

```
20 PREDKLIN MAX = 500, >
PREDKLIN PREDKKAT  SKOK
```

X	X	X

Instrukcja zmiany predkosci liniowej zostala zaprogramowana.
W celu poprawnego dzialania programu uzytkowego instrukcja deklarowania predkosci liniowej musi poprzedzic ciag instrukcji pozycjonowania, dla ktorych obowiazywac bedzie jej wartosc. Dlatego umieszczenie tej instrukcji na poczatku programu spowoduje, ze uzytkownik uniknie przypadkowych bledow wykonania programu. Ruch modulu elektrycznego manipulatora odbywac sie bedzie z taka predkoscia, jaka otrzymamy po wymnozeniu wartosci

wzietej z instrukcji deklarowania predkosci przez wartosc procentowa umieszczona w instrukcji pozycjonowania.

5.2.1.2. Instrukcja zmiany predkosci katowej maksymalnej i podstawowej.

W przypadku modulow elektrycznych charakteryzujacych sie ruchem obrotowym dla instrukcji pozycjonowania nalezy okreslic predkosc katowa. Postepowanie w tym przypadku jest analogiczne do deklarowania predkosci liniowej dla modulow elektrycznych charakteryzujacych sie ruchem liniowym.
Predkosc katowa podstawowa i maksymalna moze byc zmieniana w dowolnym miejscu programu instrukcja PREDKKAT. Podczas wykonywania instrukcji pozycjonowania sprawdzane jest czy predkosc ruchu nie jest wieksza od aktualnie zadeklarowanej predkosci maksymalnej.
Predkosc podstawowa i maksymalna podaje sie w stopniach na sekunde (deg/s). Predkosc podstawowa musi byc przy tym nie wieksza niz maksymalna.
Zalozmy, ze poprzednio zaprogramowana instrukcja byla instrukcja skoku bezwarunkowego o numerze 10:

1.

```
10 SKOK DO 1000
PREDKLIN PREDKKAT  SKOK
```

X	2	X

2.

```
20 PREDKKAT MAX
PREDKKAT MAX DEG/S =
```


Po wprowadzeniu predkosci maksymalnej (np. 60) nastepuje przejscie do punktu 3.

3.

```
<T MAX =  60, PREDKKAT =
PREDKKAT DEG/S =
```


Po wprowadzeniu predkosci podstawowej (np. 40) nastepuje przejscie do punktu 4.

4.

```
20 PREDKKAT MAX =  60, >
PREDKLIN PREDKKAT  SKOK
```

X	X	X

Instrukcja zmiany predkosci katowej zostala zaprogramowana.
W celu poprawnego dzialania programu uzytkowego instrukcja deklarowania predkosci liniowej musi poprzedzic ciag instrukcji pozycjonowania, dla ktorych obowiazywac bedzie jej wartosc. Dlatego umieszczenie tej instrukcji na poczatku programu spowoduje, ze uzytkownik uniknie przypadkowych bledow wykonania programu. Ruch modulu elektrycznego manipulatora odbywac sie bedzie z taka predkoscia, jaka otrzymamy po wymnozeniu wartosci wzietej z instrukcji deklarowania predkosci przez wartosc procentowa umieszczona w instrukcji pozycjonowania.

5.2.1.3. Instrukcja skoku bezwarunkowego i warunkowego.

W trakcie wykonywania programu czesto zachodzi koniecznosc wykonania skoku do innego miejsca programu. Mozna to zrealizowac przez zaprogramowanie instrukcji skoku. Istnieje mozliwosc zaprogramowania dwoch rodzajow instrukcji skoku:

- instrukcji skoku bezwarunkowego,
- instrukcji skoku warunkowego.

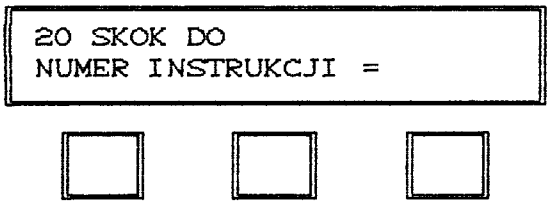
Skok bezwarunkowy wykonywany jest do wskazanej instrukcji bez wzgledu na okolicznosci (oczywiscie jesli program zawiera ta instrukcje).
Skok warunkowy wykonuje sie, jesli spelniony jest zaprogramowany warunek. W przeciwnym przypadku wykonywanie programu kontynuowane jest od instrukcji znajdujacej sie bezposrednio za instrukcja skoku. Skok warunkowy moze byc uwarunkowany stanem flagi, wejscia lub wartosci.
Zalozmy, ze poprzednio programowana instrukcja byla instrukcja czekania bezwarunkowego o numerze 10:

1.

```
10 CZEKAJ 50S
PREDKLIN PREDKKAT  SKOK
```

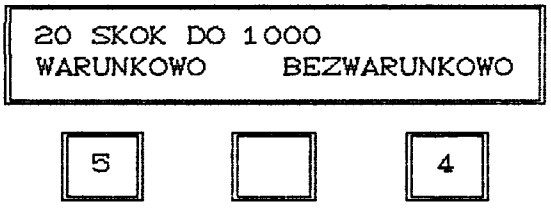
X	X	2

2.

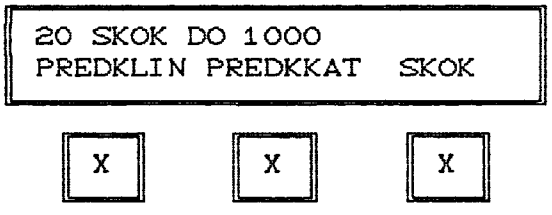


Po wprowadzeniu numeru instrukcji (korzystajac z grupy przyciskow cyfr i przycisku "WPI" na panelu programowania), do ktorej ma byc wykonany skok (liczba od 1 do 9999, np 1000), przechodzimy do punktu 3.

3.

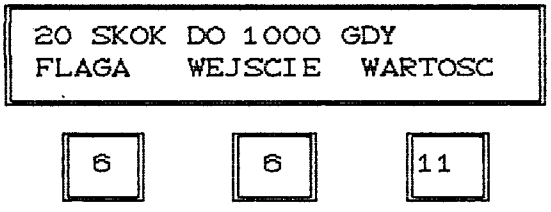


4.



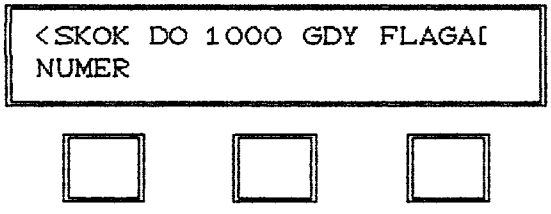
Zaprogramowana zostala instrukcja skoku bezwarunkowego.

5.



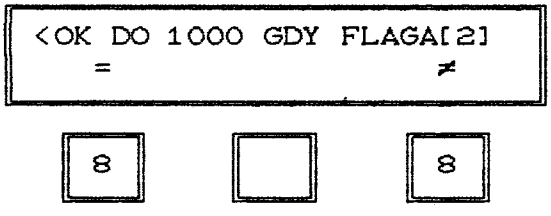
Z dwoch rownoprawnych mozliwosci tj. FLAGA, WEJSCIE, wybierzmy FLAGA.

6.



Po wpisaniu numeru (liczba od 0 do 63, np 2) przechodzimy do punktu 7.

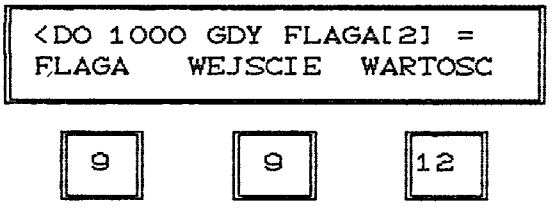
7.



32

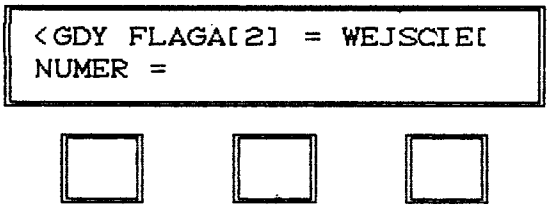
Zalozmy, ze badane bedzie spelninie rownosci, a wiec wybieramy znak "=" i przechodzimy do punktu 8.

8.



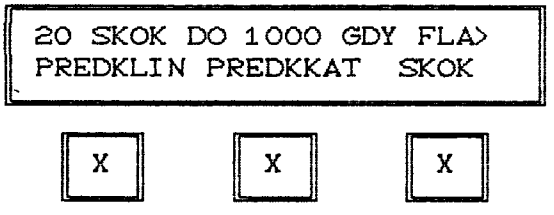
Wybierajac FLAGA lub WEJSCIE (np. WEJSCIE) przechodzimy do pkt. 9, a decydujac sie na WARTOSC przechodzimy do pkt. 12.

9.



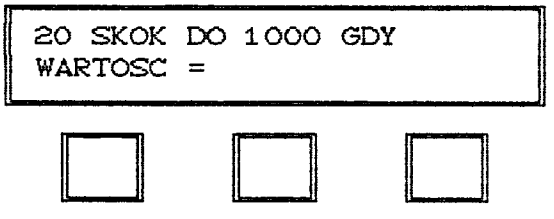
Po wpisaniu numeru (liczba od 0 do 63, np 8) przechodzimy do punktu 10.

10.



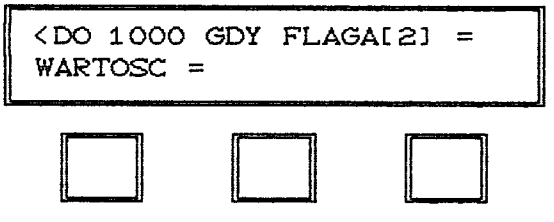
Zaprogramowana zostala instrukcja skoku warunkowego.

11.



Nalezy wprowadzic wartosc, z ktora bedzie porownywana prawa strona warunku (liczba 0 lub 1, np. 1). Dalsze operacje kontynuowane sa od punktu 7, z tym ze zamiast tekstu "FLAGA[2]" bedzie tekst "1".

12.



Po wprowadzeniu wartosci, z ktora porownywana bedzie lewa strona (liczba 0 lub 1, np. 0), dalsze czynnosci beda takie jak w punkcie 10, z tym ze zamiast tekstu "WEJSCIE[8]" bedzie tekst "0".

5.2.1.4. Instrukcja czekanie bezwarunkowego i warunkowego.

W trakcie wykonywania programu czesto zachodzi koniecznosc wytracenia czasu lub zaczekania na jakies zdarzenie. Mozna to zrealizowac przez zaprogramowanie instrukcji czekania. Istnieje mozliwosc zaprogramowania dwoch rodzajow instrukcji czekania:

- instrukcji czekania bezwarunkowego,
- instrukcji czekania warunkowego.

Czekanie bezwarunkowe polega na wstrzymaniu wykonywania ciagu instrukcji na przeciag zadeklarowanego czasu.

Czekanie warunkowe polega na oczekiwaniu na spelnienie zaprogramowanego waruneku. Oznacza to, ze wykonywanie programu uzytkowego bedzie wstrzymane, az do chwili, gdy warunek bedzie spelniony. Czekanie warunkowe moze byc uzaleznione od stanu flagi, wejscia lub wartosci.

O tym, ze robot znajduje sie w stanie oczekiwania na uplyw czasu lub na spelnienie warunku swiadczy mruganie przycisku-lampki STR (start) na panelu operacyjnym.

Zalozmy, ze poprzednio programowana instrukcja byla instrukcja skoku bezwarunkowego o numerze 10:

1.

10 SKOK DO 1000		
CZEKAJ	POWTORZ	KONPOWT
2	X	X

2.

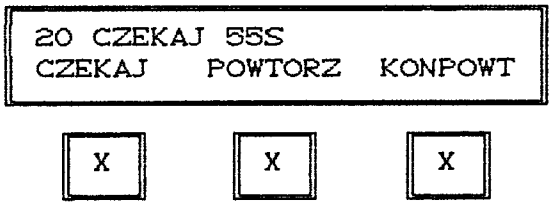
20 CZEKAJ	
CZAS	WARUNEK
3	5

3.

20 CZEKAJ		
CZAS CZEKANIA S =		

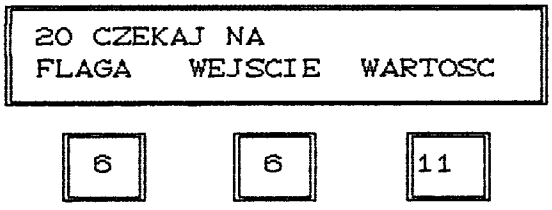
Nalezy wprowadzic czas czekania (liczba od 0.1 do 999.9, np. 55 sek), po czym nastepuje przejscie do pkt. 4.

4.



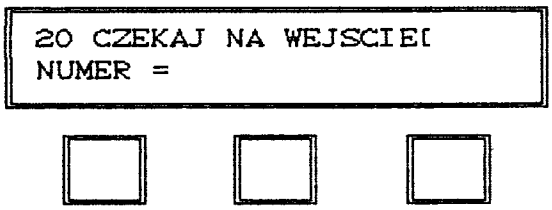
Zaprogramowana zostala instrukcja czekania bezwarunkowego, tzn czekania na upływ czasu.

5.



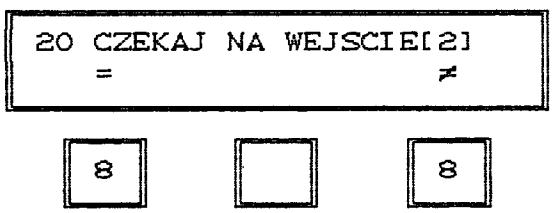
Z dwoch rownoprawnych mozliwosci tj. FLAGA, WEJSCIE, wybierzmy WEJSCIE.

6.



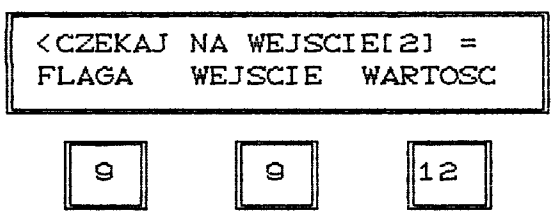
Po wpisaniu numeru (liczba od 0 do 63, np. 2) przechodzimy do punktu 7.

7.



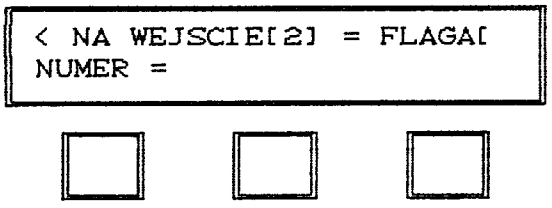
Zalozmy, ze badane bedzie spelninie rownosci, a wiec wybieramy znak "=" i przechodzimy do punktu 8.

8.



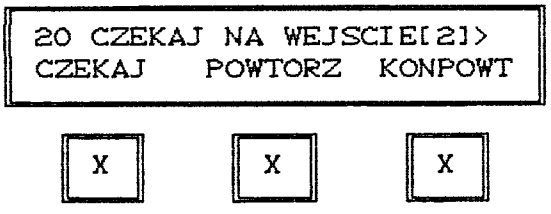
Wybierajac FLAGA lub WEJSCIE (np. FLAGA) przechodzimy do pkt. 9, a decydujac sie na WARTOSC przechodzimy do pkt. 12.

9.



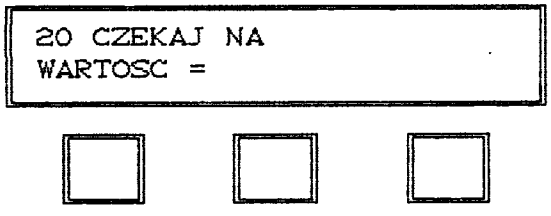
Po wpisaniu numeru (liczba od 0 do 63, np. 8) przechodzimy do punktu 10.

10.



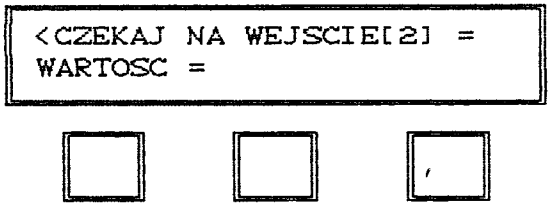
Zaprogramowana zostala instrukcja czekania warunkowego.

11.



Nalezy wprowadzic wartosc, z ktora bedzie porownywana prawa strona warunku (liczba 0 lub 1, np. 0). Dalsze operacje kontynuowane sa od punktu 7, z tym ze zamiast tekstu "WEJSCIE[2]" bedzie tekst "0".

12.



Po wprowadzeniu wartosci, z ktora porownywana bedzie lewa strona (liczba 0 lub 1, np. 1), dalsze czynnosci beda takie jak w punkcie 10, z tym ze zamiast tekstu "FLAGA[8]" bedzie tekst "1".

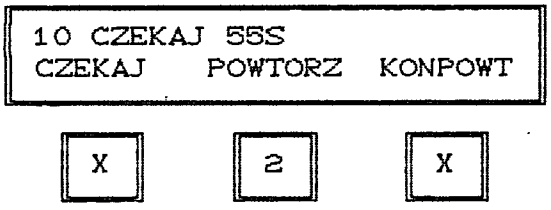
5.2.1.5. Instrukcja poczatku petli programowej.

Instrukcja poczatku petli programowej wraz z opisana w nastepnym punkcie instrukcja konca petli umozliwiaja zaprogramowanie powtarzania wykonywania zamknietej miedzy tymi instrukcjami sekwencji instrukcji.

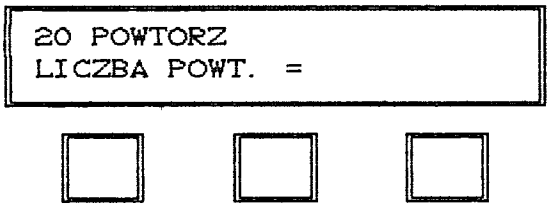
Petla programowa moze zawierac w sobie inne petle programowe, przy czym instrukcja konca petli wewnetrznej musi znajdowac sie przed instrukcja konca petli zewnetrznej. Dopuszcza sie 8 poziomow

zagniezdzen. Niedozwolone jest wykonanie instrukcji konca petli bez uprzedniego wykonania instrukcji poczatku petli (liczba rozpoczec petli nie moze byc mniejsza niz liczba zakonczen).
Petla powinna byc konczona jedynie w sposob naturalny, tzn. przez wykonanie sie zaprogramowana ilosc razy. Wyjscie z petli przez wyskok poza nia powoduje, ze petla zostaje niezamknieta, co w dalszej konsekwencji moze spowodowac bledne dzialanie programu.
Zalozmy, ze poprzednio programowana instrukcja byla instrukcja czekania bezwarunkowego o numerze 10:

1.

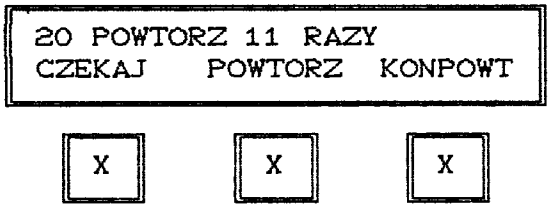


2.



Po wprowadzeniu liczby powtorzen (liczba od 1 do 999, np. 11) przechodzimy do punktu 3.

3.

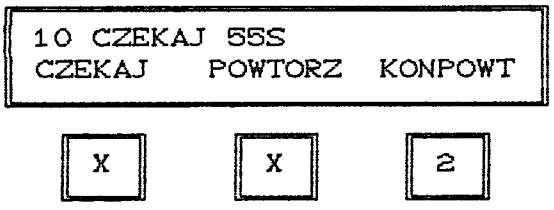


Zostala zaprogramowana instrukcja poczatku petli programowej.

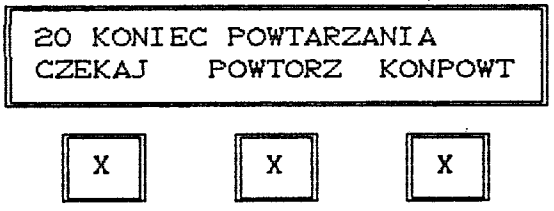
5.2.1.6. Instrukcja konca petli programowej.

Jak wspomniano w poprzednim punkcie, instrukcja konca petli zamyka ciag instrukcji wchodzacych w sklad petli programowej. Wykonanie instrukcji konca petli powoduje zmniejszenie wewnetrznego licznika obiegow petli. Jesli licznik ten osiagnal wartosc zerowa, to petla jest zamykana (zmniejsza sie wewnetrzny licznik poziomow zagniezdzen) i wykonywana jest nastepna instrukcja za instrukcja konca petli.
Zalozmy, ze poprzednio programowana instrukcja byla instrukcja czekania bezwarunkowego o numerze 10:

1.



2.

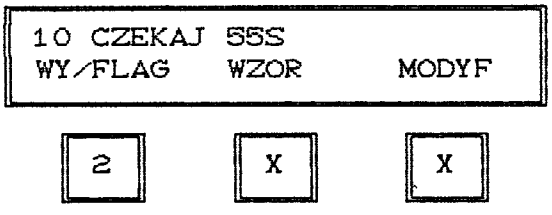


Zostala zaprogramowana instrukcja konca petli programowej.

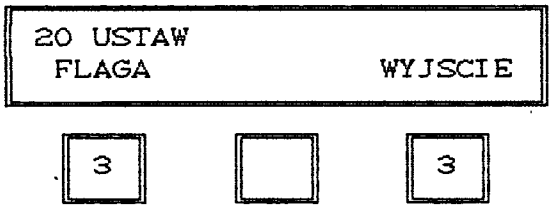
5.2.1.7. Instrukcja ustawiania wyjscia lub flagi.

Dla celow komunikacji z otoczeniem oraz organizacji wykonania programu (czekanie warunkowe, skoki warunkowe) mozliwe jest zaprogramowanie instrukcji ustawiajacej wartosc wyjscia lub flagi na wartosc podana eksplicite lub wartosc reprezentowana przez stan flagi lub wejscia.
Zalozmy, ze poprzednio programowana instrukcja byla instrukcja czekania bezwarunkowego o numerze 10:

1.

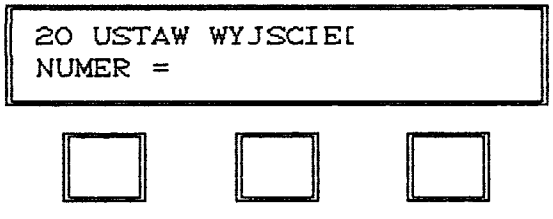


2.



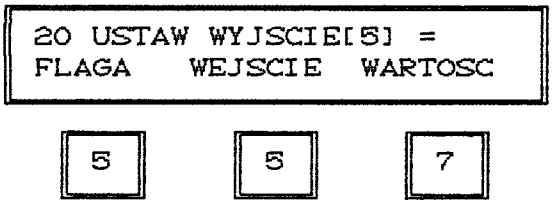
Z dwoch rownoprawnych mozliwosci tj. FLAGA, WYJSCIE, wybierzmy WYJSCIE.

3.



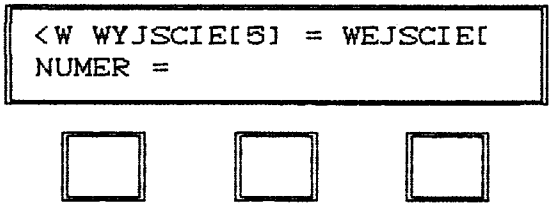
Po wpisaniu numeru (liczba od 0 do 63, np. 5) przechodzimy do punktu 4.

4.



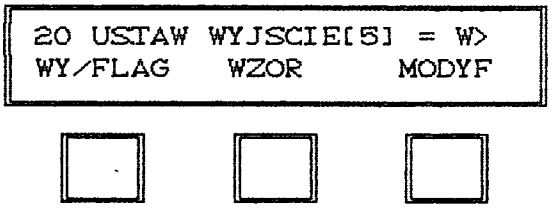
Wybierajac FLAGA lub WEJSCIE (np. WEJSCIE) przechodzimy do pkt. 5, a decydujac sie na WARTOSC przechodzimy do pkt. 7.

5.



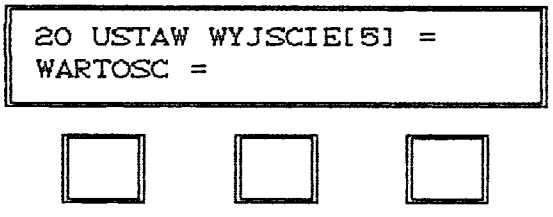
Po wpisaniu numeru (liczba od 0 do 63, np. 20) przechodzimy do punktu 6.

6.



Zaprogramowana zostala instrukcja ustawienia flagi lub wyjscia.

7.



Po wprowadzeniu wartosci, ktora bedzie wyslana na wyjscie lub przypisana fladze (liczba 0 lub 1, np. 0), dalsze czynnosci beda takie jak w punkcie 6, z tym ze zamiast tekstu "WEJSCIE[20]" bedzie tekst "0".

5.2.1.8. Instrukcja deklarowania poczatku wzorca.

Instrukcje deklarowania polozenia poczatku wzorca WZOR oraz odwolania sie do wzorca MODYF umozliwiaja realizacje funkcji paletyzacji. Obie instrukcje stosowane sa w tych programach, w ktorych wystepuja pewne czynnosci powtarzajace sie i inne zmieniajace sie. Typowym przykladem jest zdejmowanie detali z palety i podawanie ich do maszyny.
Ruch robota od palety do maszyny, podawanie detali do maszyny oraz ewentualny powrot do pozycji oczekiwania, sa takie same dla

wszystkich detali znajdujacych sie na palecie. Poszczegolne detale leza w roznych miejscach i ich pobieranie musi byc programowane indywidualnie. Podawanie detali odbywa sie wedlug pewnego wzoru, ktory jest modyfikowany dla kazdego z nich.
Instrukcja WZOR sluzy do oznaczania poczatku programu realizujacego dzialanie wedlug wzoru.
Zalozmy, ze poprzednio programowana instrukcja byla instrukcja czekania bezwarunkowego o numerze 20:

1.

20 CZEKAJ 55S		
WY/FLAG	WZOR	MODYF
X	2	X

2.

30 WZOR		
WY/FLAG	WZOR	MODYF
X	X	X

Zaprogramowana zostala instrukcja deklarowania poczatku wzorca.

5.2.1.9. Instrukcja odwolania do wzorca.

Instrukcja odwolania sie do wzorca MODYF wraz z opisana powyzej instrukcja deklarowania poczatku wzorca WZOR umozliwia, jak juz wspomniano realizacje paletyzacji. Instrukcja MODYF sluzy do wydzielania rozniacych sie od siebie czesci wzoru, jest ona uzywana przed kazda nowa modyfikowana czescia wzoru, ktora moze skladac sie z pewnej liczby instrukcji roznych typow.
Instrukcje zawarte miedzy kolejnymi instrukcjami MODYF realizuja wykonywanie poszczegolnych czesci wzoru. Liczba tych instrukcji moze byc dowolna. Numer instrukcji wystepujacej po instrukcji WZOR, jest stosowany jako adres skoku powrotnego, po wykonaniu kazdej czesci wzoru. Przy kazdej instrukcji MODYF (oprocz pierwszej) jest wykonywany skok do tej czesci wzoru, ktora nie byla jeszcze wykonywana.
Po kazdej czesci wzoru jest wykonywany skok powrotny do wycinka programu miedzy instrukcja WZOR i pierwsza instrukcja MODYF, ktory realizuje czynnosci powtarzajace sie w pracy. Odbywa sie to tak dlugo, az wszystkie czesci wzoru zostana wykonane, po czym program jest kontynuowany.

Instrukcja WZOR musi byc zawsze zaprogramowana przed instrukcja MODYF. Przy podwojnej pracy, wedlug wzoru (pobieranie detali i odkladanie ich na palete) konieczne jest zastosowanie wskaznikow, ktorych stan wskazuje, czy ma byc wykonywana kolejna czesc wzoru pobierania, czy tez ukladania detalu.

UWAGA:

Podczas testowania programu krok po kroku, w pracy recznej, podczas wykonywania wycinka programu realizujacego wzor, nie mozna poza wycinkiem dokonac zadnych poprawek programu.

Zalozmy, ze poprzednio programowana instrukcja byla instrukcja czekania bezwarunkowego o numerze 40:

1.

40 CZEKAJ 55S		
WY/FLAG	WZOR	MODYF
X	X	2

2.

50 MODYF		
WY/FLAG	WZOR	MODYF
X	X	X

Zaprogramowana zostala instrukcja odwolania sie do wzoru.

Przedstawiamy ponizej przyklad zastosowania instrukcji WZOR i MODYF dla sytuacji, w ktorej trzy przedmioty sa pobierane z jednego miejsca, obrabiane, a nastepnie odkladane na palete.

```
 10 WZOR
 20 ...      } Pobieranie przedmiotu i obrobka
 30 ...      } Czesc wspolna wzoru
 40 ...      }
 50 MODYF      Instrukcja MODYF poprzedza kazda czesc wzoru
               rozna dla poszczegolnych przedmiotow
 60 ...      }
 70 ...      } Odkladanie pierwszego przedmiotu
 80 ...      }
 90 MODYF
100 ...      }
110 ...      } Odkladanie drugiego przedmiotu
120 ...      }
130 MODYF
140 ...      }
150 ...      } Odkladanie trzeciego przedmiotu
160 ...      }
170 ...       Kontynuacja programu lub KONIEC
```

Instrukcje sa wykonywane w nastepujacej kolejnosci: 10, 20, 30, 40, 50, 60, 70, 80, 90, 20, 30, 40, 50, 100, 110, 120, 130, 20, 30, 40, 50, 140, 150, 160, 170.

41

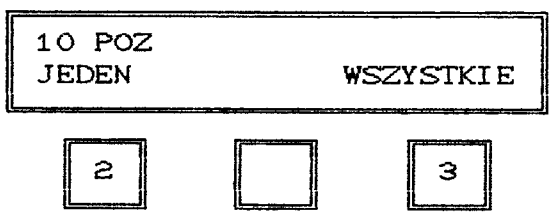
5.2.2. Programowanie instrukcji pozycjonowania (stan 2).

Programowanie instrukcji pozycjonowania, ze wzgledu na swoja wage i czestosc wystepowania w trakcie tworzenia programu uzytkowego, zostalo wyodrebnione w oddzielny stan.
Poniewaz konfiguracja robota PR-02E stanowi dowolna kombinacje modulow elektrycznych i modulow pneumatycznych oddzielnie zostanie przedstawione pozycjonowanie modulow pneumatycznych i elektrycznych. Spowodowane to jest roznicami w sposobie okreslania pozycjonowania dla roznych typow modulow.

5.2.2.1. Pozycjonowanie modulow pneumatycznych.

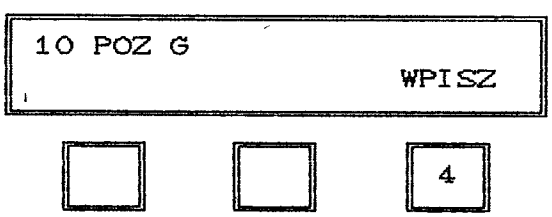
Wcisniecie przycisku "POZ" umieszczonego na panelu programowania powoduje znikniecie uprzednio wyswietlanego tekstu oraz wypisanie na gornym wyswietlaczu nowego numeru instrukcji (w przedstawionym ponizej przykladzie, zakladadamy, ze instrukcja pozycjonowania jest pierwsza wprowadzana instrukcja) wraz z mnemonikiem instrukcji pozycjonowania tj. "POZ":

1.



Wyboru jednej osi dokonujemy wciskajac jeden z przyciskow z grupy MODULY PNEUMATYCZNE (oznaczonych literami alfabetu np. "G"). Wowczas przechodzimy do 2:

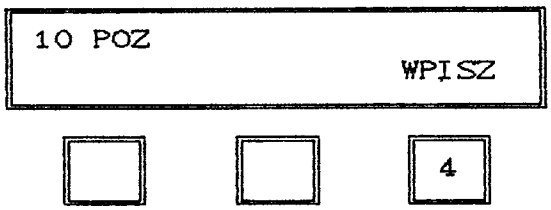
2.



Jezeli modul "G" znajduje sie w docelowym polozeniu, to potwierdzamy ten fakt przyciskajac przycisk zmienny pod tekstem "WPISZ", a nastepnie program przechodzi do punktu 4 zwiekszajac automatycznie numer instrukcji pozycjonowania (w naszym przykladzie na 20). Pozycja modulu "G" zostala tym samym zapamietana. W przypadku, gdy modul "G" nie jest w polozeniu docelowym nalezy przelaczyc sie na reczne operowanie manipulatorem (przycisk "PRZ" na panelu), ustawic go w zadanym polozeniu (patrz rozdzial "RECZNE OPEROWANIE MANIPULATOREM"), a nastepnie powrocic

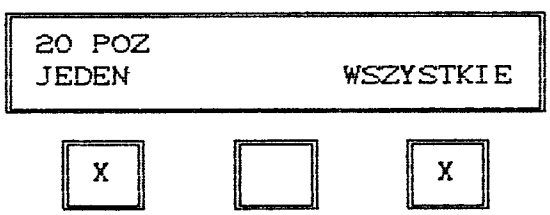
do pozycjonowania wciskajac ponownie przycisk "PRZ". Wowczas ponownie znajdziemy sie w punkcie 2.

3.



W przypadku wybrania przycisku "WSZYSTKIE" system sam dokonuje wyboru modulow. Powyzej przedstawiona jest sytuacja gdy robot posiada jedynie moduly pneumatyczne. Potwierdzenie pozycji docelowej wszyskich obecnych w konfiguracji modulow pneumatycznych osiagamy wciskajac przycisk zmienny pod tekstem "WPISZ". Wowczas zapamietane sa pozycje wszystkich zadeklarowanych i obecnych modulow pneumatycznych. Nastepnie program zwieksza numer instrukcji pozycjonowania i przechodzi do punktu 4. Oczywiscie przed potwierdzeniem pozycji mozliwe jest przelaczenie na reczne operowanie manipulatorem i odpowiednie ustawienie poszczegolnych modulow w taki sam sposob, jak dla jednego modulu.

4.



Instrukcja pozycjonowania modulow (modulu) pneumatycznych zostala zaprogramowana.
Przejscie do programowania innych instrukcji niz pozycjonowania mozna uzyskac wciskajac przycisk "INS" na panelu programowania. Wowczas zniknie mnemonik instrukcji pozycjonowania "POZ", a na dolnym wyswietlaczu pojawi sie odpowiednie menu.

5.2.2.2. Pozycjonowanie modulow elektrycznych.

Przy programowaniu instrukcji pozycjonowania modulow elektrycznych konieczne jest okreslenie sposobu wykonania tej instrukcji. Czyni sie to przez podanie nastepujacych parametrow:

- predkosci lub czasu,
- sposobu dojscia do punktu koncowego (dokladnie, zgrubnie).

Predkosc ruchu okresla predkosc liniowa lub katowa z jaka bedzie poruszal sie modul elektryczny.
W instrukcji pozycjonowania podaje sie ja w procentach predkosci

podstawowej (z dokladnoscia do dziesiatych czesci procenta). Oznacza to, ze przed wykonaniem pozycjonowania brana jest aktualna wartosc predkosci podstawowej mnozona przez procent podany w instrukcji i wykonywana jest kontrola, czy otrzymany wynik jest nie wiekszy od aktualnej predkosci maksymalnej. Jesli tak, to zostaje zasygnalizowany blad. Aktualne wartosci predkosci podstawowej i maksymalnej sa to wartosci okreslone przez ostatnio wykonane instrukcje PREDKLIN i PREDKKAT.

Sposob dojsca do polozenia koncowego, podany w instrukcji pozycjonowania , okresla sposob w jaki zostanie wykonany koncowy fragment zaprogramowanego ruchu. Sa dwie mozliwosci osiagniecia punktu koncowego:

- dokladnie,
- zgrubnie.

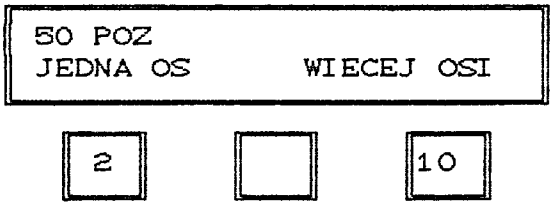
Dojscie dokladne oznacza, ze dopoki uklady wykonawcze nie sygnalizuja osiagniecia wyznaczonego polozenia koncowego, dopoty nie bedzie rozpoczete wykonywanie nastepnej instrukcji pozycjonowania. Oznacza to, ze na koncu ruchu zostanie wykonane hamowanie az do zatrzymania manipulatora w zadanym w instrukcji polozeniu.

Dojscie zgrubne oznacza, ze po wyslaniu do ukladow wykonawczych ostatniej wartosci przyrostu polozenia, jezeli nastepna instrukcja jest instrukcja pozycjonowania, przechodzi sie do jej realizacji bez czekania az dokladnie zostanie osiagniete polozenie zadane poprzednia instrukcja pozycjonowania. Dzieki takiemu typowi dojscia do punktu koncowego otrzymuje sie efekt wygladzenia trajektorii aproksymowanej odcinkami ("obcinanie rogow"). Im wieksza predkosc ruchu, tym beda wieksze niedokladnosci w osiaganiu punktu koncowego. Jezeli nastepna instrukcja po instrukcji pozycjonowania ze zgrubnym dojsciem do punktu koncowego nie jest instrukcja pozycjonowania, to instrukcja pozycjonowania wykonywana jest tak jak instrukcja z dojsciem dokladnym.

Zalozmy, ze chcemy zaprogramowac pozycjonowanie modulow elektrycznych oraz, ze ostatnia instrukcja miala numer 50. Wowczas po nacisnieciu przycisku "POZ" na panelu programowania pojawi sie nastepujacy tekst:

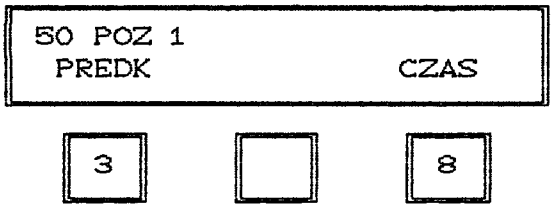
44

1.



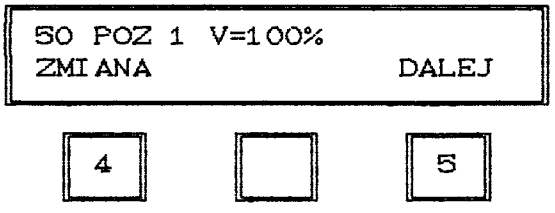
Wyboru jednej osi dokonujemy wciskajac jeden z przyciskow z grupy MODULY ELEKTRYCZNE (oznaczonych cyframi rzymskimi "I+", "I-", "II+", "II-", "III+", "III-", np "I+"). Wowczas przechodzimy do 2:

2.



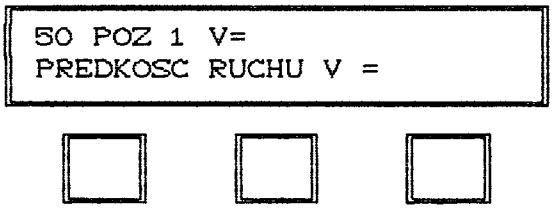
Wciskajac przycisk zmienny po textem "PREDK" przechodzimy do punktu 3, natomiast wybierajac "CZAS" do punktu 8.

3.



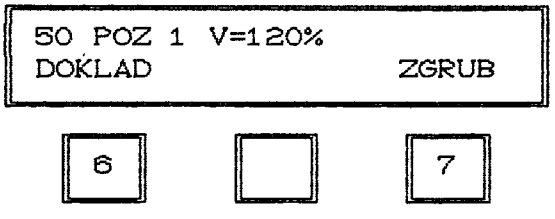
Mozna teraz dokonac wyboru predkosci z jaka ma byc wykonane pozycjonowanie. Liczba oznaczajaca predkosc "mruga". Mozna wiec ja zmienic lub pozostawic niezmieniona, tzn. taka jak w ostatnio programowanej instrukcji deklaracji predkosci (PREDKLIN lub PREDKKAT).

4.



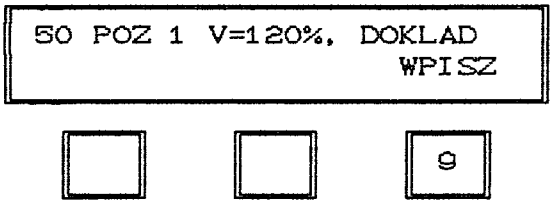
Po wpisaniu nowej wartosci (np. 120) przy uzyciu przyciskow z grupy cyfr wciskamy "WPI" i przechodzimy do punktu 5:

5.



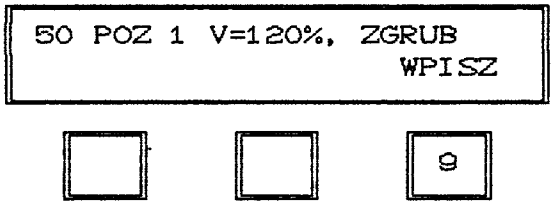
Wyboru sposobu dojscia do punktu koncowego dokonujemy wciskajac odpowiedni przycisk zmienny.

6.



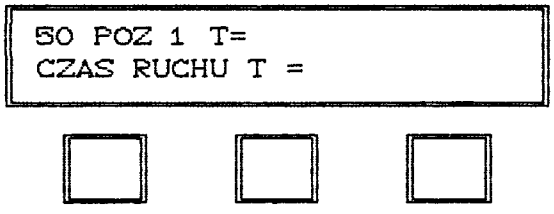
Wciskajac przycisk zmienny pod tekstem "WPISZ" zaprogramowalismy instrukcje pozycjonowania dla modulu elektrycznego nr 1.

7.



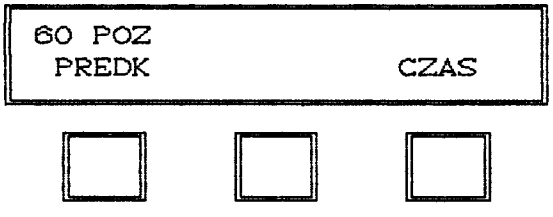
Wciskajac przycisk zmienny pod tekstem "WPISZ" zaprogramowalismy instrukcje pozycjonowania dla modulu elektrycznego nr 1.

8.



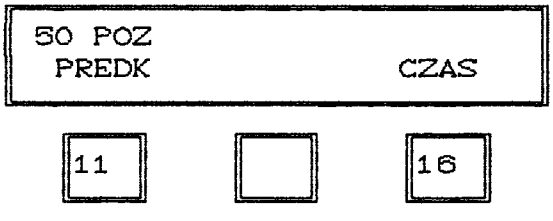
Nalezy wprowadzic czas trwania ruchu (od 0.1 do 999.9 sekund), po czym nastepuje przejscie do punktu 5 z ta roznica, ze zamiast tekstu "V=100%" bedzie tekst "T=11.1S" (jesli wybralismy czas 11.1 sekundy).

9.



Mozemy teraz przejsc do pozycjonowania innych modulow lub innych instrukcji. Oczywiscie podczas pozycjonowania modulow elektrycznych, tak jak przy pozycjonowaniu modulow pneumatycznych aktywny jest przycisk "PRZ" (przelacz) dzieki czemu mozemy reczne nastawiac manipulator w rzadana pozycje.

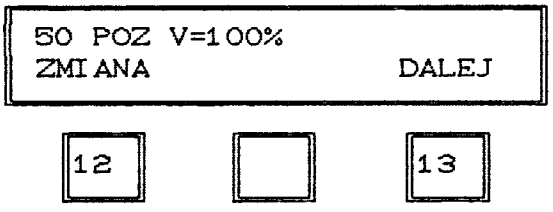
10.



Decydujac sie na wariant "WIECEJ OSI" program pozycjonuje

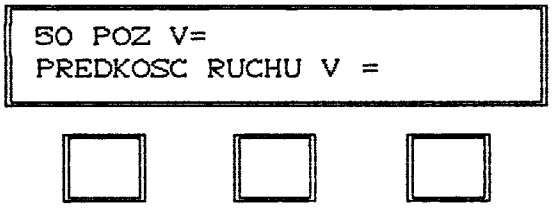
(automatycznie dokonujac wyboru) wszyskie moduly zadeklarowane w konfiguracji. Ponizsza sekwencja pozycjonowania wystepuje jezeli w sklad konfiguracji robota wchodza zarowno moduly pneumatyczne, jak i elektryczne. Oczywiscie przed potwierdzeniem pozycji dla wszyskich modulow mozliwe jest przelaczenie na reczne operowanie manipulatorem i odpowiednie ustawienie poszczegolnych modulow w taki sam sposob, jak dla jednego modulu.
Wciskajac przycisk zmienny po textem "PREDK" przechodzimy do punktu 11, natomiast wybierajac "CZAS" do punktu 16.

11.



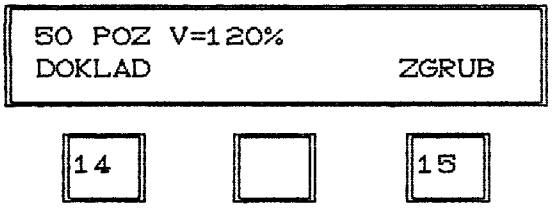
Mozna teraz dokonac wyboru predkosci z jaka ma byc wykonane pozycjonowanie. Liczba oznaczajaca predkosc "mruga". Mozna wiec ja zmienic lub pozostawic niezmieniona, tzn. taka jak w ostatnio programowanej instrukcji deklaracji predkosci (PREDKLIN lub PREDKKAT).

12.



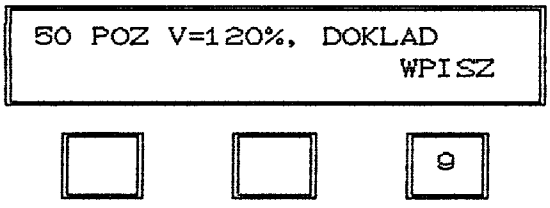
Po wpisaniu nowej wartosci (np. 120) przy uzyciu przyciskow z grupy cyfr wciskamy "WPI" i przechodzimy do punktu 13. Wpisana przez nas wartosc odnosi sie do wszystkich zadeklarowanych modulow elektrycznych.

13.



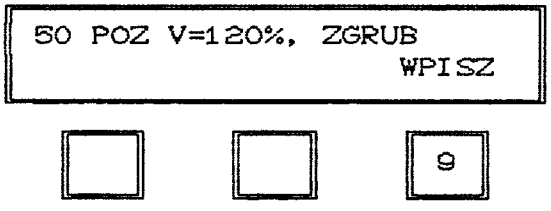
Wyboru sposobu dojscia do punktu koncowego dokonujemy wciskajac odpowiedni przycisk zmienny.

14.



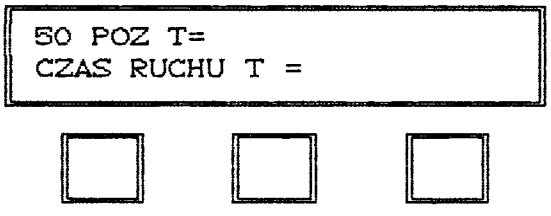
Wciskajac przycisk zmienny pod tekstem "WPISZ" zaprogramowalismy instrukcje pozycjonowania dla wszyskich zadeklarowanych modulow.

15.



Wciskajac przycisk zmienny pod tekstem "WPISZ" zaprogramowalismy instrukcje pozycjonowania dla wszyskich zadeklarowanych modulow.

16.



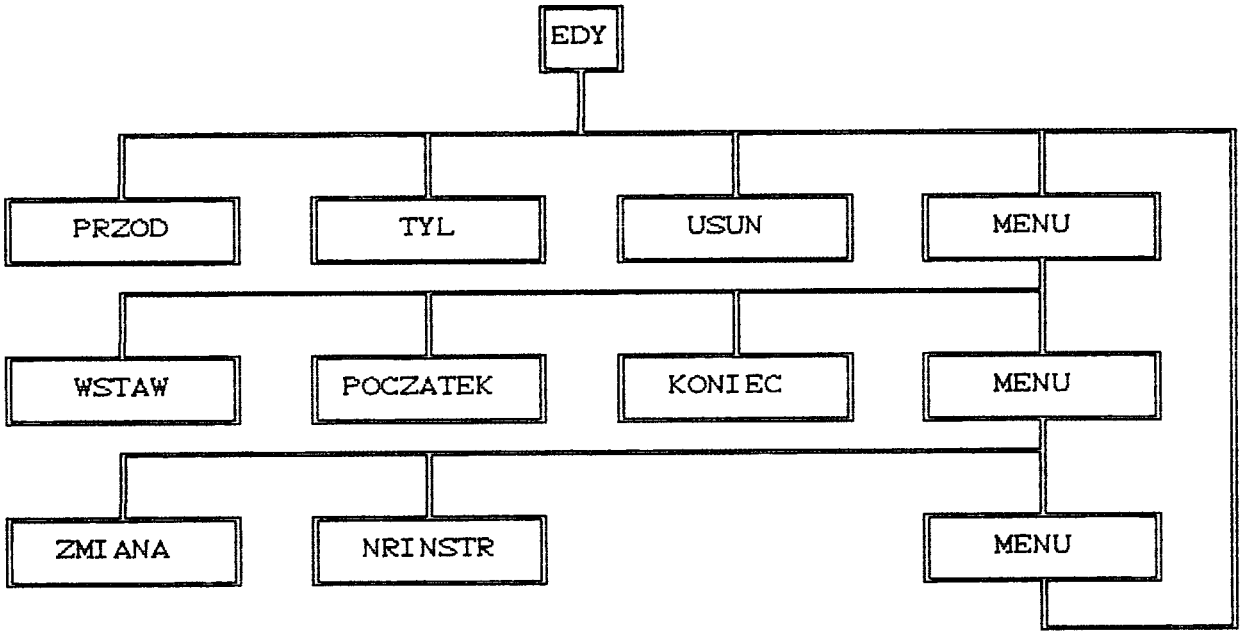
Nalezy wprowadzic czas trwania ruchu (od 0.1 do 999.9 sekund), po czym nastepuje przejscie do punktu 13 z ta roznica, ze zamiast tekstu "V=100%" bedzie tekst "T=11.1S" (jesli wybralismy czas 11.1 sekundy).

6. EDYCJA PROGRAMU.

Znaczenie poszczegolnych pozycji menu jest nastepujace:

PRZOD	- wyswietlenie nastepnej instrukcji
TYL	- wyswietlenie poprzedniej instrukcji
USUN	- usuniecie instrukcji
WSTAW	- wstawianie miedzy dwie instrukcje
POCZATEK	- wyswietlenie pierwszej instrukcji programu
KONIEC	- wyswietlenie ostatniej instrukcji programu
ZMIANA	- zmiana parametrow numerycznych instrukcji
NRINSTR	- ustawienie numeru instrukcji

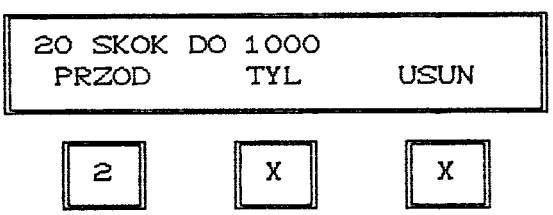
Menu dla stanu "edycja programu" jest nastepujace:



6.1. Wyswietlenie nastepnej instrukcji (PRZOD).

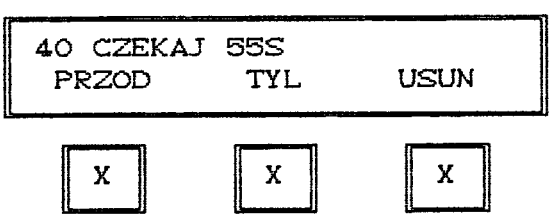
Funkcja PRZOD umozliwia przejscie do nastepnej zaprogramowanej instrukcji programowu. Jezeli istnieje w programie instrukcja o numerze wiekszym od aktualnie wyswietlonego, to jej tekst pojawi sie na gornym wyswietlaczu. W przeciwnym wypadku na ok. 1 sekunde w gornym wyswietlaczu pojawi sie komunikat KONIEC PROGRAMU.

1. Zalozmy, ze przed realizacja funkcji PRZOD stan wyswietlaczy jest:

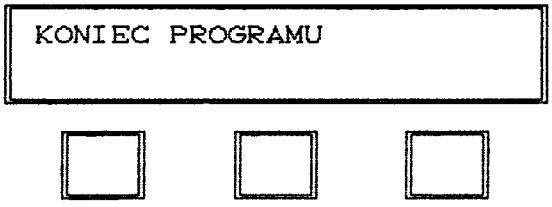


2. W zaleznosci od polozenia w programie instrukcji poprzedzajacej realizacje funkcji PRZOD moga wystapic dwa przypadki:

- instrukcja nie jest ostatnia instrukcja w programie, np. po niej wystepuje instrukcja o numerze 40:



- instrukcja jest ostatnia instrukcja w programie:

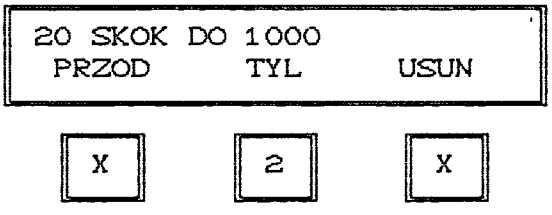


Po okolo 1 sekundzie nastepuje powrot do punktu 1.

6.2. Wyswietlenie poprzedniej instrukcji (TYL).

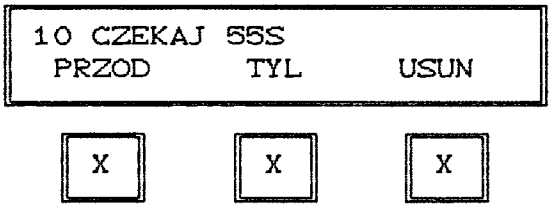
Funkcja TYL umozliwia przejscie do instrukcji poprzedzajacej aktualnie wyswietlana instrukcje. Jezeli jest taka instrukcja, to jej tekst zostaje wyswietlony na wyswietlaczu. Jesli nie, to na okolo 1 sek na gornym wyswietlaczu pojawi sie komunikat: POCZATEK PROGRAMU.

1. Zalozmy, ze przed realizacja funkcji TYL wyswietlacz wyglada nastepujaco:

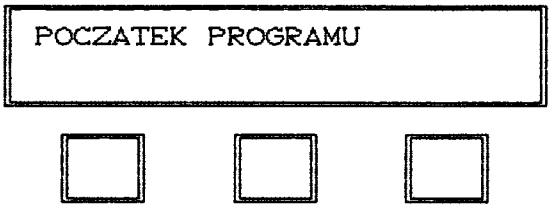


2. W zaleznosci od polozenia w programie instrukcji poprzedzajacej wykonanie funkcji TYL moga wystapic dwa przypadki:

- instrukcja nie jest pierwsza instrukcja w programie np. poprzedza ja instrukcja o numerze 10:



- instrukcja jest pierwsza instrukcja w programie:



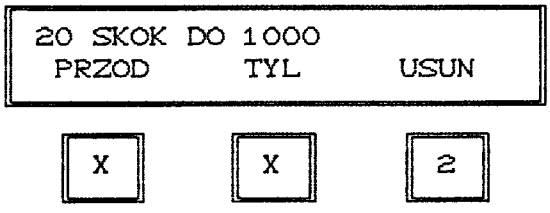
Po okolo 1 sek. nastepuje przejscie do punktu 1.

6.3. Usuniecie instrukcji (USUN).

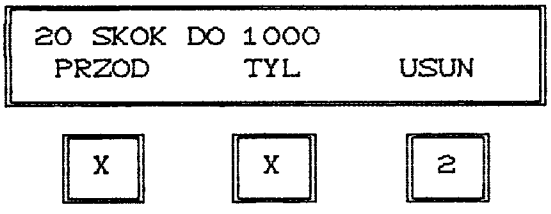
Funkcja USUN daje mozliwosc usuniecia instrukcji, ktorej tekst

pokazany jest na wyswietlaczu. Wykonanie tej funkcji powoduje usuniecie wskazanej instrukcji z ciala programu. Na gornym wyswietlaczu pozostaje tylko numer skasowanej instrukcji.

1. Zalozmy, ze przed realizacja funkcji USUN stan wyswietlacza jest nastepujacy:



2.



Instrukcja zostala usunieta.

6.4. Wstawienie instrukcji miedzy dwie inne (WSTAW).

Funkcja WSTAW umozliwia szybkie wstawienie instrukcji miedzy obecnie wyswietlona a nastepna instrukcja. Wartosc numeru instrukcji, ktory zostanie wyswietlony na gornym wyswietlaczu zalezy od warosci numerow instrukcji biezacej i nastepnej. Numer ten moze byc:

1. Srednia arytmetyczna (z zaokragleniem w dol) numerow instrukcji biezacej i nastepnej,
2. Najblizszym numerowi instrukcji biezacej numerem podzielnym przez 10.

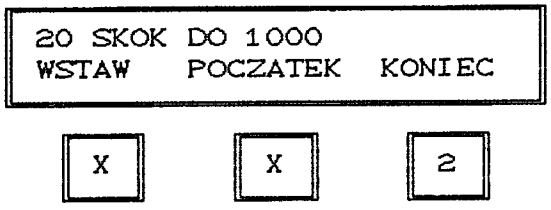
Przypadek 1 bedzie mial miejsce wtedy, gdy:

- roznica miedzy numerem instrukcji biezacej i nastepnej bedzie mniejsza niz 10 i oba numery beda mialy taka sama cyfre dziesiatek, np. B (numer instrukcji biezacej) = 110, N (numer instrukcji nastepnej) = 119, to W (numer instrukcji wstawianej) = 114,
- ostatnia cyfra numeru instrukcji biezacej jest 0, a numer instrukcji nastepnej jest o 10 wiekszy, np. B = 110, N = 120, to W = 115.

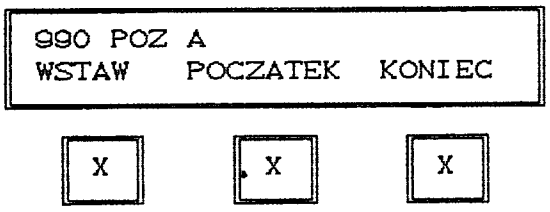
Przypadek 2 wystapi wtedy, gdy:

- roznica miedzy numerem instrukcji biezacej i nastepnej bedzie wieksza niz 10, np. B = 110, N = 121, to W = 120 lub B

1. Zalozmy, ze przed realizacja funkcji KONIEC stan wyswietlaczy jest:



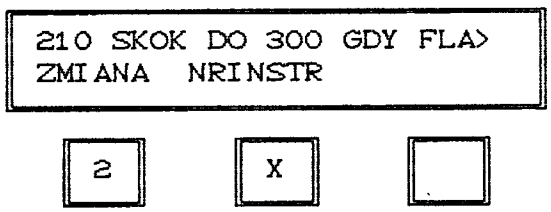
2. Jezeli ostatnia instrukcja w programie bedzie na przyklad instrukcja pozycjonowania o numerze 990, to wyswietlacze beda wygladac nastepujaco:



6.7. Zmiana parametrow numerycznych instrukcji (ZMIANA).

Funkcja ZMIANA pozwala na zmiane parametrow numerycznych instrukcji bez koniecznosci programowania calej instrukcji. Parametr, ktorego wartosc mozna w danej chwili zmieniac "mruga". Po zmianie dowolnego parametru nowa wersja instrukcji wpisywana jest do ciala programu w miejsce starej. Przy wprowadzaniu nowej wartosci parametru sprawdzane sa ograniczenia wlasciwe dla tego parametru. Na przyklad, gdybysmy zmieniali numer narzedzia w instrukcji zmiany narzedzia, to zostanie sprawdzone, czy nowy numer jest wiekszy od zera i mniejszy od dziesieciu.

1. Zalozmy, ze chcemy poprawic parametry instrukcji skoku warunkowego, ktora wyglada nastepujaco: 210 SKOK DO 300 GDY FLAGA[5] = 1. Przed rozpoczeciem realizacji funkcji ZMIANA stan wyswietlaczy jest wiec nastepujacy:



2. Po nacisnieciu ZMIANA liczba 300 (pierwszy numeryczny parametr aktualnie wyswietlonych instrukcji) zaczyna "mrugac". Oznacza to, ze jego wartosc moze byc zmieniana.

= 110, N = 1234, to W = 120,

- roznica miedzy numerem instrukcji biezacej i nastepnej jest niewieksza od 10, ale oba numery maja rozne cyfry dziesiatek, np. B = 119, N = 129, to W = 120,

- biezaca instrukcja jest ostatnia instrukcja w programie, np. B = 110, to W = 120 lub B = 117, to W = 120.

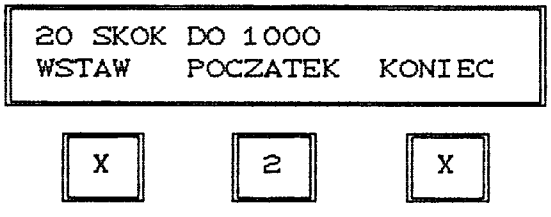
UWAGA:

Nowy numer instrukcji zostanie wyswietlony tylko wtedy, gdy roznica miedzy numerem instrukcji biezacej i nastepnej jest wieksza niz 1. W przeciwnym przypadku bedzie sygnalizowany blad.

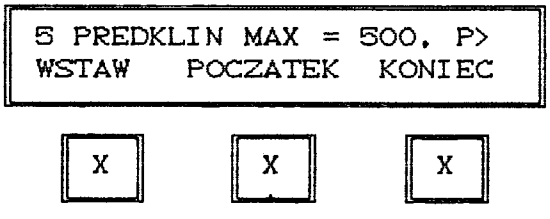
6.5. Wyswietlenie pierwszej instrukcji programu (POCZATEK).

Funkcja POCZATEK pozwala na szybkie przejscie na poczatek tworzonego lub edytowanego programu. Jesli program jest pusty, tzn. nie ma zadnej zaprogramowanej instrukcji, to w gornym wyswietlaczu wyswietlany jest numer 10. W przeciwnym przypadku wyswietlona jest instrukcja o najnizszym numerze.

1. Zalozmy, ze przed realizacja funkcji POCZATEK stan wyswietlaczy jest:

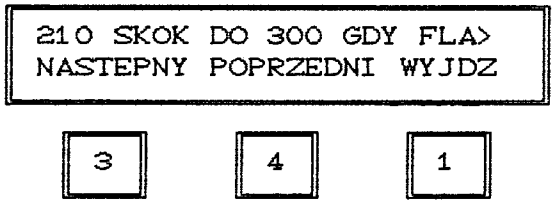


2. Jesli pierwsza instrukcja w programie bedzie instrukcja nr 5, to wyswietlacze beda wygladac nastepujaco:

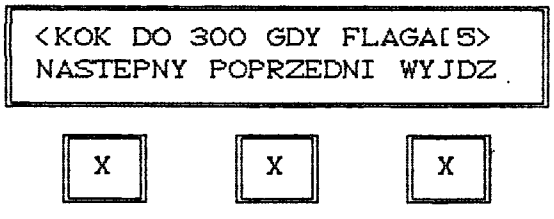


6.6. Wyswietlanie ostatniej instrukcji programu (KONIEC).

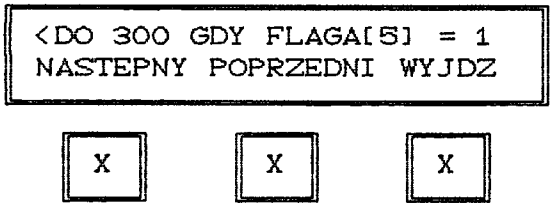
Funkcja KONIEC pozwala na szybkie przejscie na koniec tworzonego lub edytowanego programu. Jezeli program jest pusty tzn. nie ma zadnej zaprogramowanej instrukcji, to w gornym wyswietlaczu wyswietlany jest numer 10. W przeciwnym przypadku wyswietlona jest instrukcja o najwyzszym numerze.



Nacisniecie przycisku NASTEPNY spowoduje "udostepnienie" do zmiany nastepnego parametru (jesli jest). W tym przypadku spowoduje to podswietlenie numeru flagi (pkt 3). Nacisniecie przycisku POPRZEDNI "udostepni" poprzedni parametr (znajdujacy sie na lewo od aktualnie podswietlonego). W tym przypadku bedzie to podswietlenie liczby z ktora porownywana jest wartosc flagi (pkt 4). Przeskakiwanie z parametru na parametr odbywa sie cyklicznie, tzn po nacisnieciu przycisku NASTEPNY z ostatniego parametru przeskakujmy na pierwszy, a w przypadku przycisku POPRZEDNI - z pierwszego na ostatni. Jesli nastepny lub poprzedni parametr nie jest aktualnie wyswietlony, to "okienko" nad tekstem jest tak przesuwane aby stal sie on widoczny. Nacisniecie przycisku WYJDZ zakonczy proces zmiany parametrow i spowoduje wyjscie do menu glownego (pkt 1).

3. 

Numer flagi (tu: liczba 5) jest teraz parametrem, ktorego wartosc moze byc zmieniana.

4. 

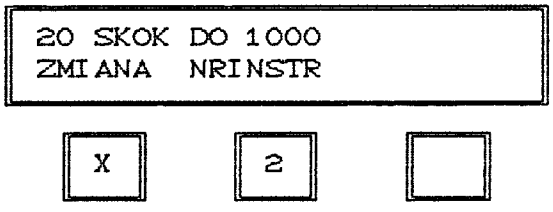
Wartosc, z ktora porownywana jest flaga (tu: liczba 1) jest parametrem, ktory moze byc modyfikowany.

6.8. Ustawienie numeru instrukcji (NRINSTR).

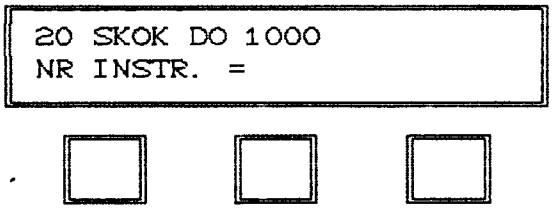
Czesto w trakcie tworzenia lub edycji programu zachodzi koniecznosc zaprogramowania lub obejrzenia instrukcji o wybranym numerze. Do tego celu sluzy funkcja NRINSTR. Po wpisaniu nowego numeru instrukcji na gornym wyswietlaczu ukazuje sie tekst

instrukcji, o ile pod tym numerem jest zaprogramowana jakas instrukcja lub wyswietlany jest tylko numer instrukcji, gdy program nie zawiera instrukcji o tym numerze.

1. Zalozmy, ze przed wykonaniem funkcji NRINSTR stan wyswietlacza byl nastepujacy:



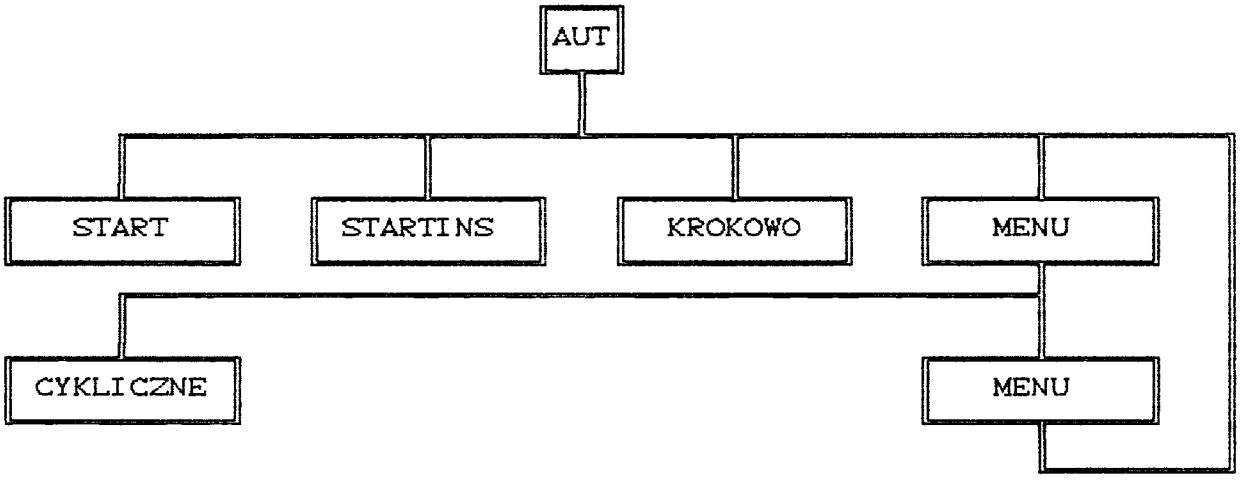
2.



Nalezy wpisac numer instrukcji, ktora ma zostac wyswietlona w gornym wierszu (liczba od 1 do 9999). Po wpisaniu numeru nastepuje przejscie do punktu 1. W gornym wyswietlaczu wyswietlany jest numer i tekst instrukcji jesli taka jest, lub tylko numer instrukcji w przeciwnym wypadku.

7. PRACA AUTOMATYCZNA (STAN 4).

Menu dla stanu "praca automatyczna" wyglada nastepujaco:



Znaczenie pozycji menu jest nastepujace:

START - start programu od pierwszej instrukcji programu,

STARTINS - start programu od instrukcji aktualnie pokazanej na wyswietlaczu,

KROKOWO - wykonanie aktualnie wyswietlonej instrukcji,

CYKLICZNIE - wykonanie programu okreslona ilosc razy.

7.1. Start programu od pierwszej instrukcji (START).

Po nacisnieciu przycisku START rozpoczyna sie automatyczne wykonywanie programu uzytkowego poczawszy od pierwszej instrukcji (instrukcji o najmniejszym numerze). Po dojsciu i wykonaniu ostatniej instrukcji program sterujacy powraca do pierwszej instrukcji i rozpoczyna wykonywanie programu uzytkowego w nieskonczonej petli.

7.2. Start programu od instrukcji aktualnie pokazanej na wyswietlaczu (STARTINS).

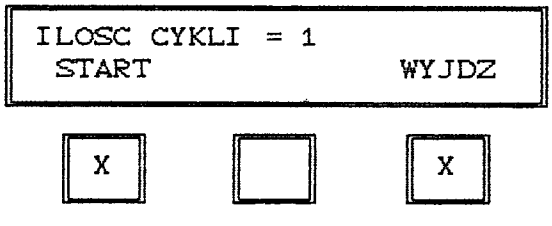
Po nacisnieciu przycisku STARTINS rozpoczyna sie automatyczne wykonywanie programu w nieskonczonej petli poczawszy od instrukcji aktualnie pokazanej na wyswietlaczu.

7.3. Wykonanie aktualnie wyswietlonej instrukcji (KROKOWO).

Nacisniecie przycisku KROKOWO powoduje wykonanie instrukcji wyswietlonej aktualnie na wyswietlaczu. Po wykonaniu instrukcji na wyswietlaczu ukazuje sie nastepna instrukcja do wykonania (np. jesli instrukcja wykonana byla instrukcja skoku bezwarunkowego, to na wyswietlaczu pojawi sie instrukcja do ktorej nastapil skok). Taki tryb pracy nazywa sie praca krokowa, tzn po wykonaniu kazdej instrukcji oczekuje sie na decyzje operatora, czy nastepna instrukcja ma byc wykonana.
Krokowe wykonywanie instrukcji jest szczegolnie przydatny przy tworzeniu i uruchamianiu programu. Pozwala ono na: sprawdzenie dzialania pojedynczej instrukcji, dokladne doprowadzenie manipulatora do pozycji zapamietanej w instrukcji pozycjonowania, kilkakrotne wykonanie tej samej instrukcji dla roznych wartosci parametrow itd.

7.4. Cykliczne wykonywanie programu (CYKLICZNIE).

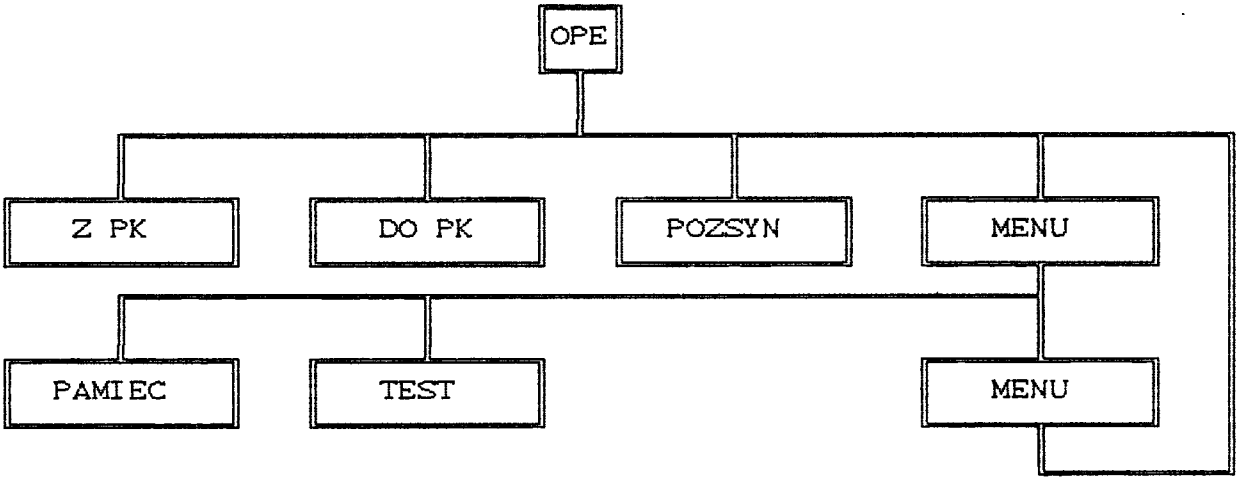
Nacisniecie przycisku "CYKLICZNIE" powoduje ukazanie sie nastepujacego menu:



W miejsce migajacej liczby "1" mozemy wpisac nowa okreslajaca liczbe powtorzen calego programu (przy uzyciu przyciskow z grupy cyfr i "WPI"). Wcisniecie "START" oznaczac bedzie wykonanie programu tyle razy, ile zadeklarowano w gornym wierszu. Po wcisnieciu "WYJDZ" wracamy do glownego menu dla stanu "praca automatyczna" i liczba powtorzen (rozna od 1) zostaje skasowana, gdyz instrukcja ta ma znaczenie dorazne. Inny sposob powtarzania programu moze byc zrealizowany za posrednictwem instrukcji "POWTORZ" tak, ze bedzie to stanowilo wewnetrzna ceche programu, badz za pomoca przerwania pracy programu po zwyklym uzyciu przycisku "START". Po wykonaniu programu z zadeklarowana liczba powtorzen dolny wyswietlacz znow przyjmie postac jak po wcisnieciu przycisku "CYKLICZNIE".
Program uruchomiony kazdym z czterech powyzej opisanych sposobow mozna przerwac przyciskiem "STP" (stop) i wznowic korzystajac z wszystkich czterech mozliwosci rozpoczecia pracy. Podczas pracy automatycznej na panelu programowania aktywny jest tylko przycisk "STP" oraz STOP AWARYJNY. Pozostale przyciski pozostaja nieaktywne. Powyzsza koniecznosc dyktuja wymagania bezpiecznego korzystania z robota. Przerwanie pracy automatycznej przyciskiem "STP" uaktywnia wszystkie przyciski panelu.

8. RECZNE OPEROWANIE SYSTEMEM (STAN 5).

Menu dla stanu "reczne operowanie systemem" wyglada nastepujaco:



Znaczenie poszczegolnych pozycji menu jest nastepujace:

Z PK - przepisywanie programu z pamieci kasetowej PK-3 do pamieci uzytkowej,

DO PK - przepisywanie programu z pamieci uzytkowej do

pamieci kasetowej PK-1,

POZSYN - odczyt i zmiana wspolrzednych wewnetrznych punktu synchronizacji,

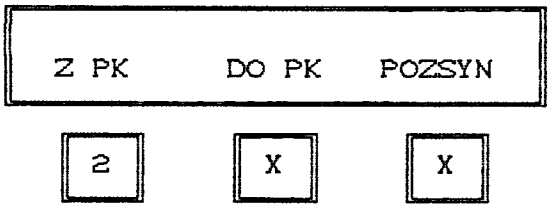
PAMIEC - odczyt wielkosci zajetego obszaru pamieci uzytkowej,

TEST - testowanie panelu programowania.

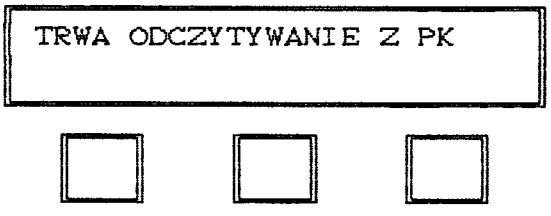
8.1. Przepisywanie programu z pamieci kasetowej do pamieci uzytkowej (Z PK).

Funkcja Z PK przeznaczona jest do odtworzenia, uprzednio zapamietanego na kasecie obrazu pamieci uzytkowej, tzn. programu oraz konfiguracji i pozycji synchronizacj robota (oczywiscie jezeli znajduja sie na kasecie). Na jednej stronie kasety moze znajdowac sie jeden obraz pamieci uzytkowej.

1.



2.



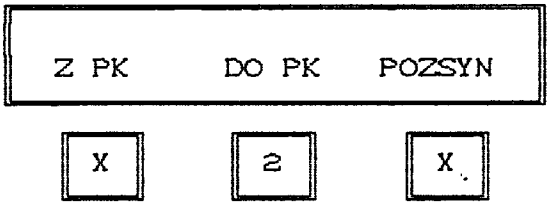
Komunikat ten wyswietlony jest az do chwili zakonczenia operacji przepisywania z pamieci kasetowej do pamieci uzytkowej. Po zakonczeniu wczytywania zostaje wyswietlona pierwsza instrukcja programu i sterowanie jest przekazane w stan "praca automatyczna". Dzieje sie to w przypadku gdy wczytana konfiguracja oraz pozycja synchronizacji odpowiadaja rzeczywistosci. W innym przypadku sygnalizowany jest blad i program sterujacy przechodzi do synchronizacji robota (oczywiscie po skasowaniu bledu).
Program uzytkowy mozna wczytac dopiero po synchronizacji robota. W trakcie pracy PK nieaktywne sa przyciski panelu programowania. Po zakonczeniu wczytywania program gotow jest do natychmiastowego uruchomienia (np. przyciskiem "STA" - start na szafie).

8.2. Przepisywanie programu z pamieci uzytkowej do pamieci kasetowej (DO PK).

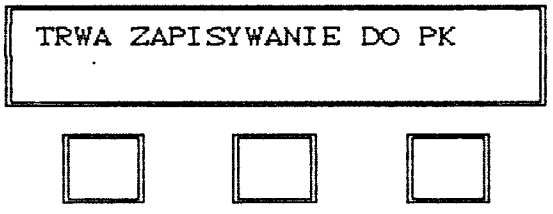
Funkcja DO PK przeznaczona jest do zapamietania w pamieci

kasetowej PK obrazu pamieci uzytkowej, tzn. programu i informacji o konfiguracji robota oraz pozycji synchronizacji geometrycznej. Na jednej stronie kasety moze byc zapisany tylko jeden obraz pamieci uzytkowej.

1.



2.

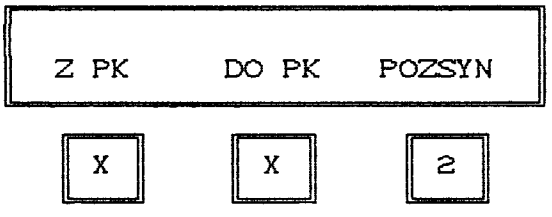


Powyzszy komunikat wyswietlony jest az do chwili zakonczenia przepisywania do pamieci kasetowej, po czym nastepuje przejscie do punktu 1.

8.3. Odczyt i zmiana wspolrzednych wewnetrznych punktu synchronizacji (POZSYN).

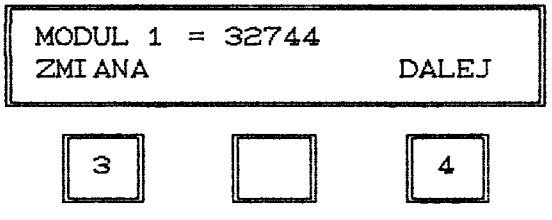
Funkcja POZSYN przeznaczona jest do odczytywania i ewentualnej zmiany wspolrzednych punktu synchronizacji geometrycznej. Punkty synchronizacji dla roznych egzemplarzy robota moga sie roznic miedzy soba.

1.



Wcisniecie przycisku "POZSYN" wywola pojawienie sie na nastepujacego napisu:

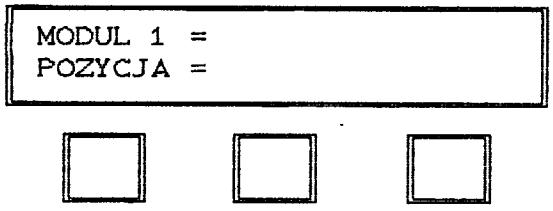
2.



Liczba po znaku "=" oznacza pozycje modulu elektrycznego nr 1 wyrazona w inkrementach, a odczytana z EPROM-ow. Wybranie przycisku "ZMIANA" umozliwia wstawienie nowej pozycji punktu

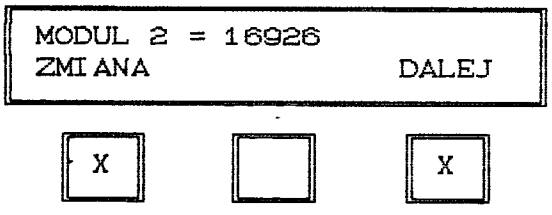
synchronizacji dla modulu elektrycznego numer 1. Przechodzimy wowczas do punktu 3:

3.



Korzystajac z grupy przyciskow cyfr nalezy wpisac nowa pozycje synchronizacji (liczbe z zakresu od 0 do 65 535). Przycisk "WPI" (wpisz) wpisuje nowe dane i wywoluje kolejny napis:

4.

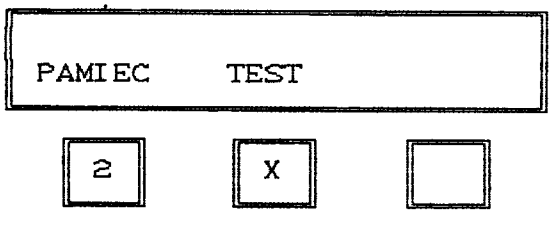


Po zakonczeniu zmian pozycji synchronizacji dla modulow elektrycznych nr 2 i 3 program sterujacy przechodzi do punktu 1. Jezeli ktoregos z modulow nie ma w konfiguracji robota to jest on pomijany podczas zmiany pozycji synchronizacji. Nalezy pamietac, ze jesli chcemy zmieniac pozycje synchronizacji, to nalezy to uczynic przed napisaniem programu uzytkowego, a nastepnie wykonywac ten program dla tych parametrow synchronizacji geometrycznej.

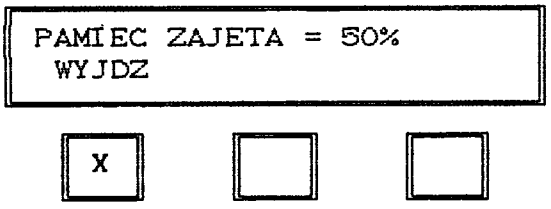
8.4. Odczyt wielkosci wolnego obszaru pamieci uzytkowej (PAMIEC).

Funkcja ta pozwala na odczytanie wielkosci pamieci zajmowanej przez program powstajacy lub wczytany z PK jaka jest jeszcze do wykorzystania. Wielkosc ta podawana jest jako procentowy stosunek wielkosci pamieci zajetej do calkowitej wielkosci pamieci dostepnej uzytkownikowi.

1.



2.



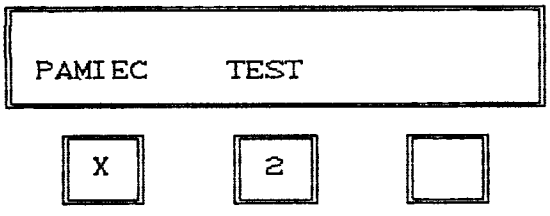
Wciskajac przycisk "WYJDZ" powracamy do menu z punktu 1.

8.5. Testowanie panelu programowania (TEST).

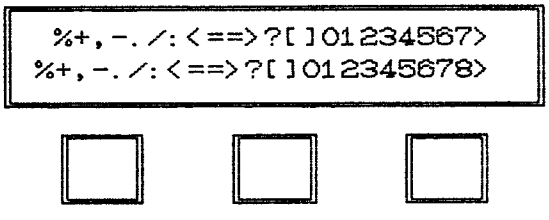
Funkcja TEST pozwala na przetestowanie poprawnosci pracy panelu programowania. Za jej pomoca mozna sprawdzic dzialanie:

- wyswietlaczy alfanumerycznych,
- diod swiecacych.

1.

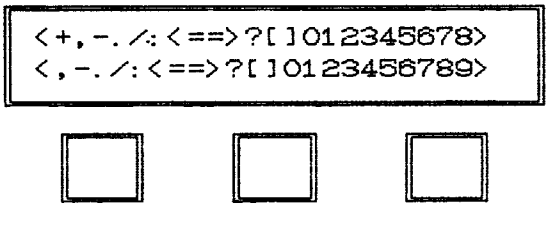


2. Po wejsciu w funkcje TEST na panelu programowania powinny sie zaswiecic wszystkie diody, a na wyswietlaczu powinien ukazac sie nastepujacy ciag znakow alfanumerycznych:

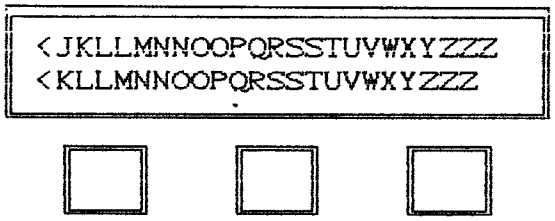


Na wyswietlaczu dolnym wyswietlany jest taki sam ciag znakow jak na wyswietlaczu gornym, tylko przesuniety o jeden znak w lewo.

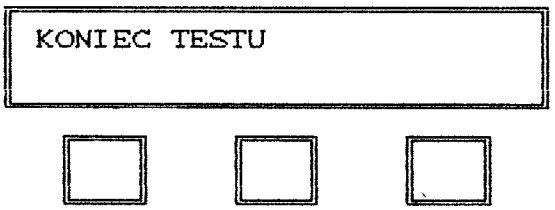
3. Co okolo 0.5 sekundy zawartosci obu wyswietlaczy przesuwaja sie o jeden znak w lewo:



4. Przesuwanie zawartosci w lewo trwa az do wyczerpania sie repertuaru dostepnych znakow:



5. Teraz zmienia sie kierunek przesuwania zawartosci wyswietlacza, tzn. co okolo 0.5 sekundy nastepuje przesuniecie wszystkich znakow o jeden znak w prawo az do osiagniecia sytuacji z punktu 2.
6. Wyjscie z testu nastepuje po rownoczesnym nacisnieciu dwoch dowolnych przyciskow i trzymaniu ich w pozycji wcisnietej, az do momentu pojawienia sie na wyswietlaczach nastepujacego tekstu:

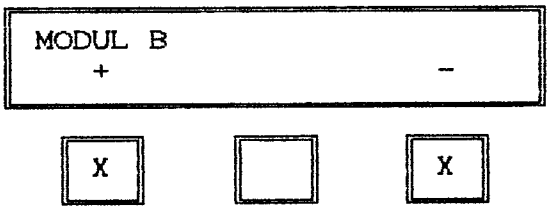


Po szybkim zwolnieniu przyciskow nastapi przejscie do punktu 1. Dluzsze przetrzymywanie wcisnietych przyciskow spowoduje restart panelu programowania dzieki zadzialaniu ukladu wewnetrznego budzika panelu.

9. RECZNE OPEROWANIE MANIPULATOREM ROBOTA.

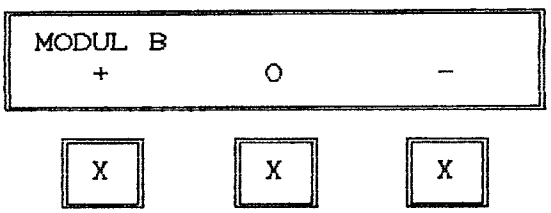
Swobodne, reczne manipulowanie robotem w zakresie modulow pneumatycznych mozliwe jest dopiero po synchronizacji. Do sterowania recznego mozna przejsc na kazdym etapie budowania programu. W trakcie pracy automatycznej niemozliwe jest poslugiwanie sie przyciskami recznego sterowania modulami elektrycznymi i pneumatycznymi (a takze przyciskiem "PRZ" - przelacz). Prace automatyczna mozemy przerwac jedynie przyciskiem "STP" (stop) znajdujacym sie na panelu i na szafie sterownika. Pozostale przyciski panelu i szafy sa w trakcie pracy automatycznej nieaktywne. Po przerwaniu pracy uaktywniaja sie wszystkie przyciski szafy, w tym przycisk "PRZ" (przelacz), ktory umozliwia sterowanie reczne. Uzycie tego przycisku uaktywnia przyciski modulow pneumatycznych, elektrycznych i pokretla. Drugie wcisniecie przycisku oznacza powrot do stanu poprzedniego.
Reczne sterowanie modulami pneumatycznymi polega na wykorzystaniu przyciskow oznaczonych literami A,B,C,D,E,F,G,H,K,L. Po uzyciu "PRZ" wybieramy modul i wciskamy np. przycisk B. Zapala sie dioda

przy tym przycisku, a na gornym wyswietlaczu pojawia sie napis:



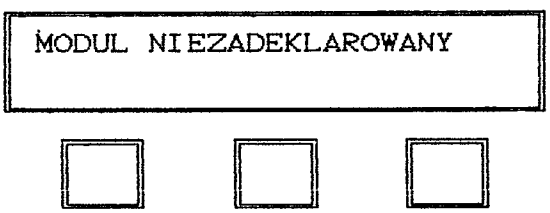
Na dolnym wyswietlaczu w pozycji 1 i 3 przyciskow zmiennych pojawia sie opisy "+" i "-" oznaczajace dwie z mozliwych pozycji, jakie moze przyjac modul dwustanowy. Wcisniecie wybranego przycisku spowoduje ruch modulu do wskazanej pozycji. Wcisniecie niewlasciwego przycisku niczego nie zmieni (zaden ruch nie bedzie wykonany). Zmiane modulu dokonujemy przez wcisniecie przycisku oznaczonego inna litera.

Dla modulow trojstanowych na wyswietlaczu pojawia sie napis:



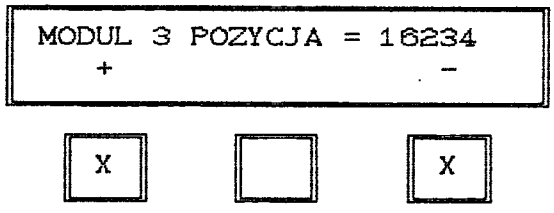
gdzie "+", "-" oznaczaja te same pozycje, jak w przypadku modulu dwustanowego. Natomiast "O" oznacza wybor pozycji posredniej.

Jezeli przez pomylke wcisniemy przycisk nie odpowiadajacy żadnemu z istniejacych modulow (np. C), to na gornym wyswietlaczu pojawi sie napis:



Moduly elektryczne moga pracowac w dwoch rodzajach sterowania recznego: pracy ciaglej i pracy z pojedynczymi przyrostami. Zadany tryb pracy wybieramy pokretlem. Pracy ciaglej odpowiada polozenie pokretla w pozycjach "1.5%", "15%", "50%", natomiast praca przyrostowa w pozycji "INKR". Jezeli chcemy poruszyc ktoryms z modulow elektrycznych w sposob ciagly, to ustawiamy pokretlo w pozycji odpowiadajacej zadanej predkosci (wyrazonej w procentach predkosci maksymalnej) i nastepnie wciskamy jeden z przyciskow odpowiadajacych modulom elektrycznym (np. przycisk "II+"). Gasnie

wtedy dioda przycisku modulu pneumatycznego (o ile taki byl wczesniej uzywany) i rozpoczyna sie ruch (w wybranym kierunku), ktory trwa tak dlugo, jak dlugo przycisk jest wcisniety. Zmiana modulu nastepuje automatycznie w przypadku wcisniecia przycisku odpowiadajacego wybranemu modulowi (i wybranemu kierunkowi ruchu). Jezeli przekrecimy pokretlo do pozycji "INKR" to przejdziemy do pracy w trybie przyrostowym. W tym trybie nacisniecie dowolnego przycisku odpowiadajacego np. modulowi III (tzn. "III+" lub "III-") spowoduje pojawienie sie na gornym wyswietlaczu informacji (pierwsze naciśnięcie nie powoduje zmiany położenia modułu):



gdzie liczba wystepujaca po znaku "=" okresla ilosc przyrostow (inkrementow) odpowiadajaca pozycji, w ktorej aktualnie znajduje sie modul. Wcisniecie przycisku oznaczonego przez "+" spowoduje zwiekszenie liczby przyrostow o jeden i odpowiadajacy temu, niewidoczny ruch manipulatora. Wcisniecie przycisku "-" powoduje zmniejszenie liczby przyrostow i ruch manipulatora w druga strone. Jednemu wcisnieciu przycisku zawsze odpowiada zmiana liczby przyrostow o jeden.

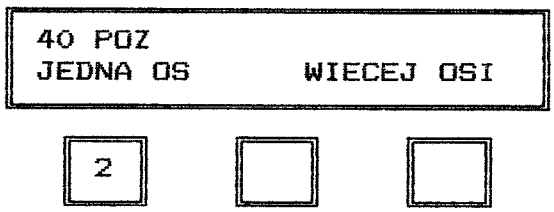
Ponowne przejscie do sterowania modulami pneumatycznymi nastepuje po wcisnieciu wybranego przycisku oznaczonego litera. Prace modulami elektrycznymi i pneumatycznymi mozna wykonywac na zmiane. Przejscie do poprzedniego stanu (np. pisanie programu) wykonujemy poprzez nacisniecie przycisku "PRZ". Podczas pracy recznej aktywne jest tylko pokretlo, przyciski oznaczone literami (odpowiadajace modulom pneumatycznym), przyciski modulow elektrycznych i przycisk "PRZ".

10. BLAD (STAN 7).

W trakcie pracy robota moze pojawic sie wiele sytuacji blednych spowodowanych przez operatora lub bedacych wynikiem awarii sprzetu. Wykrycie przez system sytuacji blednej sygnalizowane jest przez wyswietlenie na wyswietlaczu numeru bledu (na panelu operacyjnym zaczyna mrugac lampka BLA (blad)). Numer bledu pozwala

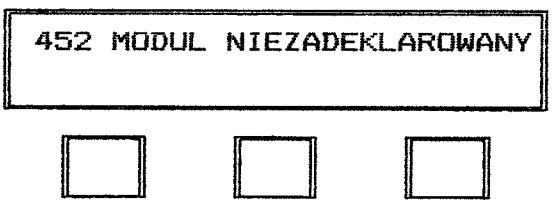
na odszukanie na liscie bledow opisu przyczyny, ktora doprowadzila do jego powstania. Wyjscie ze stanu blad moze nastapic tylko w wyniku interwencji operatora, ktory przez nacisniecie przycisku KAS (kasuj) potwierdza przyjecie komunikatu o bledzie. Po "skasowaniu bledu" nastepuje powrot do stanu z przed bledu (o ile to mozliwe) lub do stanu 4 (praca automatyczna) - po bledach w komunikacji miedzy panelem programowania a sterownikem lub po bledach podczas pracy automatycznej.
Zalozmy na przyklad, ze programujemy instrukcje pozycjonowania:

1.



Po wybraniu jednego modulu (przyciskajac "JEDNA OS"), naciskamy przycisk oznaczony litera "H". Wowczas przechodzimy do punktu 2:

2.



Oznacza to, ze popelnilismy blad polegajacy na probie wybrania modulu niezadeklarowanego (nie istniejacego w konfiguracji).
3. Nacisniecie przycisku KAS (kasuj) spowoduje przywrocenie stanu wyswietlacza sprzed bledu (punkt 1), co pozwoli na dalsze programowanie instrukcji.

10.1. Lista bledow.

Spis bledow sygnalizowanych przez procedury:

10 - NIE MOZNA WSTAWIC INSTRUKCJI
11 - BRAK MIEJSCA W PAMIECI
12 - MODUL NIEZADEKLAROWANY
21 - BRAK ARGUMENTOW NUMERYCZNYCH
23 - ZLA MANIPULACJA
24 - NIEDOZWOLONY PRZYCISK
25 - ZA DLUGA LICZBA
26 - ZLE POLOZENIE KROPKI
27 - LICZBA POZA PRZEDZIALEM
28 - BLEDY W KOMUNIKACJI

29 - TIME-OUT PRZY ODBIORZE PRZESYLKI
30 - TIME-OUT PRZY NADAWANIU PRZESYLKI
31 - INNY NIZ OCZEKIWANY KOD PRZESYLKI
40 - W PRZESYLCE WYKRYTO NIEWLASCIWY ZNAK
41 - NIEZGODNOSC SUMY KONTROLNEJ
42 - BRAK WOLNEGO BUFORA DO ODBIERANIA PRZESYLKI
43 - BLAD PODCZAS PODBIERANIA BAJTU
44 - NALOZENIE SIE NA SIEBIE DWOCH PRZESYLEK
45 - BLEDNY PIERWSZY ZNAK PRZESYLKI
46 - W KODZIE PRZESYLKI WYKRYTO NIEWLASCIWY ZNAK
47 - NIEZNANY KOD PRZESYLKI
50 - ROBOT NIEZSYNCHRONIZOWANY
60 - NIE WLACZONE NAPEDY
61 - STOP AWARYJNY
62 - SPRZET SYGNALIZUJE BLAD
63 - ROBOT JEST JUZ ZSYNCHRONIZOWANY
90 - BRAK GOTOWOSCI STEROWNIKOW OSI
91 - STOP AWARYJNY
92 - ZBYT DUZY BLAD POLOZENIA
93 - NIE WLACZONE NAPEDY
94 - INNE BLEDY WYKONANIA PROGRAMU
95 - ZBYT WYSOKA TEMPERATURA WEWNATRZ SZAFY
96 - UTRATA SYNCHRONIZACJI
97 - BRAK SYGNALU DOJSCIA DO POZYCJI
100 - INSTRUKCJA NIEZAIMPLEMENTOWANA
101 - BRAK WYWOLANEGO PROGRAMU
102 - PROBA REKURSYWNEGO WYWOLANIA PROGRAMU
110 - PROBA SKOKU DO NIEISTNIEJACEJ INSTRUKCJI
120 - ZA DUZO JEDNOCZESNIE OTWARTYCH PETLI
130 - KONEC NIEOTWARTEJ PETLI
140 - NIEZDEFINIOWANA PREDKOSC PODSTAWOWA
141 - NIEDOZWOLONA ZMIANA KONFIGURACJI
142 - PROBA PRZEKROCZENIA OBSZARU ROBOCZEGO
143 - INNE BLEDY POZYCJONOWANIA
144 - ZA DUZY BLAD POLOZENIA W STEROWNIKU OSI
145 - ZA KROTKI CZAS POZYCJONOWANIA
146 - PRZEKROCZENIE MAX PREDKOSCI RUCHU TCP
147 - NIEOSIAGALNY PUNKT KONCOWY
148 - ZA WOLNY RUCH

149 - PRZEKROCZENIE MAKSYMALNEJ PREDKOSCI W OSIACH
160 - BRAK INSTRUKCJI O WSKAZANYM NUMERZE
170 - PROBA PRZEKROCZENIA OBSZARU ROBOCZEGO
180 - PROBA POWROTU Z PROGRAMU GLOWNEGO
190 - BRAK DEFINICJI WEKTORA KOREKCJI
200 - PROBA USTAWIENIA WARTOSCI NATYCHMIASTOWEJ
201 - PROBA USTAWIENIA WEJSCIA
210 - BRAK DEFINICJI NARZEDZIA
300 - BLAD PRZY PRZEWIJANIU TASMY
301 - BLAD PRZY ZAPISIE NA TASMIE
302 - BLAD PRZY PRZEWIJANIU TASMY
303 - BLAD PRZY ODCZYCIE Z TASMY

11. OPEROWANIE ROBOTEM PRZY UZYCIU PANELU OPERACYJNEGO

Przy uzyciu panelu operacyjnego, oprocz czynnosci zwiazanych ze startem systemu, mozna wykonac nastepujace operacje:

- przepisanie programu z pamieci kasetowej do pamieci uzytkowej
- start automatycznego wykonywania programu,
- zatrzymanie automatycznego wykonywania programu.

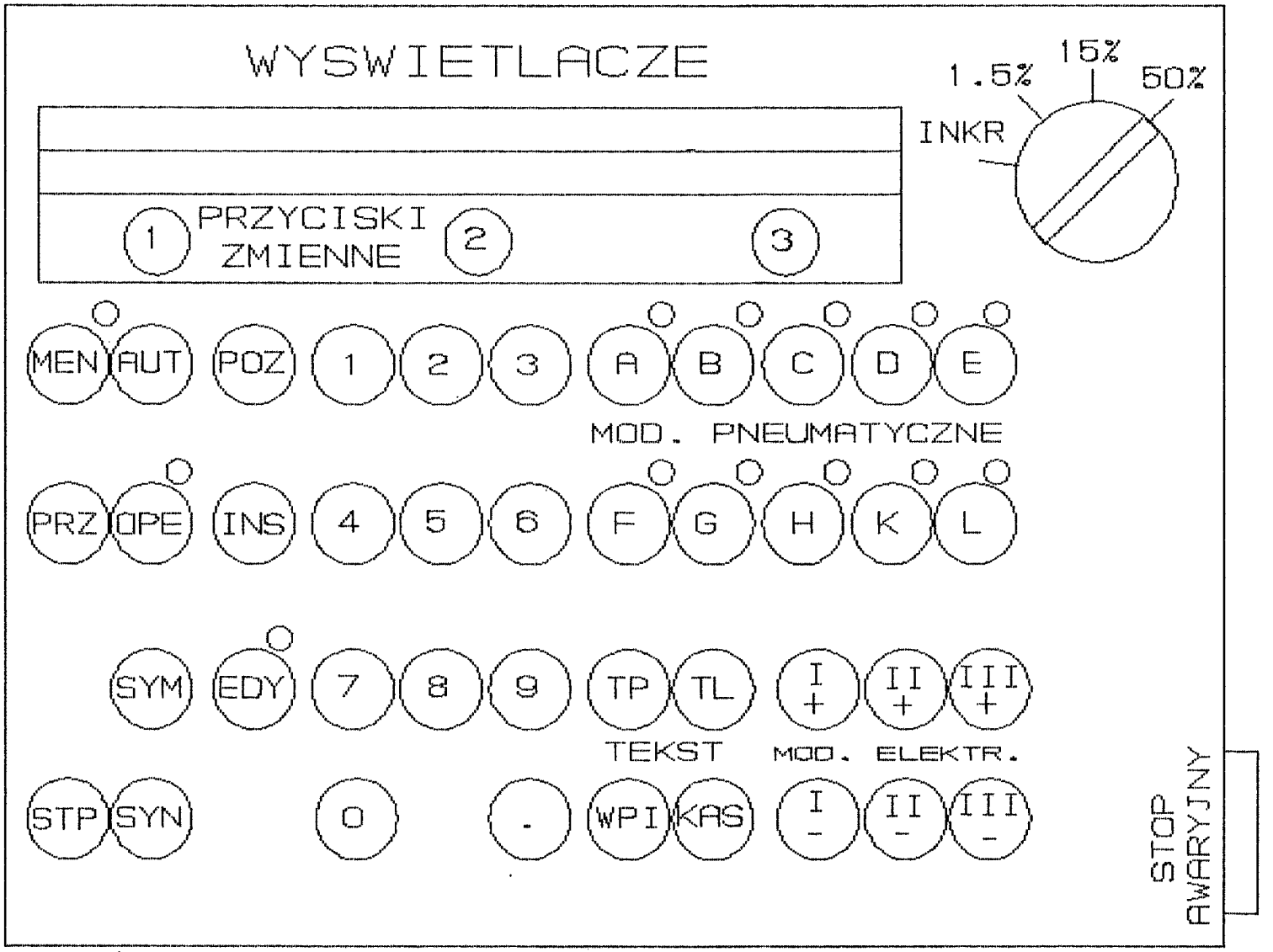
1. Przepisanie programu z pamieci kasetowej do pamieci uzytkowej uzyskuje sie po nacisnieciu zoltego przycisku-lampki PK. Podczas trwania calego procesu przepisywania lampka ta jest zapalona. Prawidlowe zakonczenie przepisywania sygnalizowane jest zgasnieciem lampki PK1. Ewentualne bledy, ktore wystapily w trakcie przepisywania beda sygnalizowane mruganiem lampki BLA (numer i opis bledu wyswietla sie na panelu programowania). Przycisk PK moze byc uzyty tylko wtedy, gdy panel programowania znajduje sie w stanie od 1 do 5. Gdy tak sie stanie panel programowania przechodzi w stan 5 "reczne operowanie systemem" i zachowuje sie tak, jakby to za jego pomoca uruchomiono przepisywanie.

2. Start automatycznego wykonywania programu uzyskuje sie przez nacisniecie zielonego przycisku-lampki STR. Lampka zaczyna swiecic swiatlem ciaglym. Wykonanie programu rozpoczynane jest od pierwszej instrukcji programu. Program wykonywany jest cyklicznie tzn. po wykonaniu ostatniej instrukcji programu wykonywana jest instrukcja pierwsza itd. Po nacisnieciu przycisku nastepuje automatyczne przelaczenie sie panelu programowania w stan 6 "praca automatyczna".

3. Zatrzymanie automatycznego wykonywania programu mozna uzyskac przez nacisniecie przycisku STP (gasnie lampka "start"). Panel programowania automatycznie przechodzi w stan 3 tzn. start programu.

UWAGA:

Nacisniecie przycisku STP (stop) nie zapewnia natychmiastowego zatrzymania ruchu manipulatora. Natychmiastowe zatrzymanie mozna uzyskac za pomoca przycisku AWA (stop awaryjny).


WYSWIETLACZE
1.5%
15%
50%
INKR
1
PRZYCISKI ZMIENNE
2
3
MEN
AUT
POZ
1
2
3
A
B
C
D
E
MOD. PNEUMATYCZNE
PRZ
OPE
INS
4
5
6
F
G
H
K
L
SYM
EDY
7
8
9
TP
TL
I +
II +
III +
TEKST
MOD. ELEKTR.
STP
SYN
0
.
WPI
KAS
I -
II -
III -
STOP AWARYJNY

OGOLNY SCHEMAT OPROGRAMOWANIA ROBOTA PR-02E

Oprogramowanie robota PR-02E bedzie w wiekszosci zrealizowane w jezyku C (kompilator MWC86) oraz asemblerze Intela 8086. W celu rozpisania projektu oprogramowania miedzy podwykonawcow caly system sterowania robotem zostal podzielony na nastepujace elementy :

A) System nadzorujacy wszystkie cznynnosci - w tym interpreter polecen oraz program obslugi bledow.

B) Panel programowania oparty na procesorze Intel 8080 i wszystkie procedury jednostki centralnej komunikujace sie z panelem programowania bezposrednio na styku zlacza. W ich sklad wchodza procedury obslugi zlacza oraz wszystkie procedury uzywane przy edycji, tworzeniu programu, interpretacji przyciskow.

C) Procedury obslugi sprzetu - sterowanie modulami elektrycznymi, pneumatycznymi, obsluga przyciskow panelu operacyjnego.

Ad A)

STRUKTURA PROGRAMU GLOWNEGO

Struktura programu bedzie wygladac w nastepujacy sposob :

1) Zablokowanie przerwan
2) Inicjalizacja zmiennych globalnych
4) Konfiguracja i synchronizacja robota (po odblokowaniu przerwan)
3) Procedura system

W programie system zostana wykorzystane dwa mechanizmy :

- mechanizm "dlugich skokow" dla obsluzenia sytuacji wyjatkowych
- obsluga przerwan generowanych przez uzycie przyciskow zamieszczonych na panelu operacyjnym.

Mechanizm "dlugich skokow" jest dostepny w wielu jezykach programowania. Umozliwia on powrot z dowolnie zagniezdzonych programow w ustalone wczesniej miejsce. Dzieki temu odpada problem propagacji sytuacji wyjatkowych na kolejne procedury. W

standartowej postaci jezyka C do obslugi tych mechanizmow sluza procedury :

```
int  setjmp( jmp_buf  env )
void longjmp( jmp_buf  env,  int  value )
```

W momencie wywolania procedury setjmp w buforze env zostana zachowane niektore rejestry procesora. Funkcja ta zwraca wtedy wartosc zero. Jezeli w dalszej czesci wykonywania programu chcemy powrocic do miejsca w ktorym wywolana zostala procedura setjmp, to uzywamy procedury longjmp. Dzialanie to mozna zilustrowac na nastepujacym przykladzie (opisujacym rowniez dzialanie projektowanego programu) :

```
int main()
{
     switch( setjmp( error_handl ) )
     {
          case 0 :  go( 0 ); break;
          case 1 : error(); break;
          deflaut :break;
                         /* niemozliwa sytuacja */
     }
}

int go( int val )
{
     if ( val == 0 )
          longjmp( error_handl, 1 );
}
```

Program ten spowoduje wywolanie najpierw procedury (zwrocona przez procedure setjmp wartosc 0) go(0), a nastepnie powrot do programu glownego (zwrocona przez setjmp wartosc 1) i wywolanie procedury error.

Dokladnie taka sama strukture bedzie mial program glowny sterowania robota. Bedzie sie on skladal z instrukcji case obejmujacej obsluge bledow :

- sprzetowych
- braku synchronizacji
- koniecznosc stopu awaryjnego

- stop interpretera
- zmiana trybu pracy
- pozostalych - ktorych wygoda obslugi w programie glownym wyniknie w trakcie pracy nad oprogramowaniem

Szkielet programu glownego :

```
int main()
{
    switch( setjmp( error_handle ) )
    {
        case HRDW_ERROR :   hrdw_error_handle(); break;
                            /*  utrata programu, lub blad
                              *  pamieci, lub dowolny blad
                              *  sprzetowy, uniemozliwiajacy
                             *poprawna prace systemu.
                             * W tym miejscu zostanie przeslany
                             * komunikat do panelu programowania
                             * program bedzie czekal na dowolny
                              * klawisz z panelu programowania
                              * lub kasowanie bledu z panelu
                             * operacyjnego. Nastepnie nastapi
                             * przejscie do restartu systemu
                             */
        case RESTART  :
                    initialise();  /* inicjalizacja zmiennych */
                   /* Wczytanie standartowej konfiguracji z pamieci
                    * EPROM. Funkcja get_config moze byc wolana w
                    * dowolnej chwili z panelu, i bedzie miala
                    * opcje pomin ( zachowaj ta konfiguracje ktora
                    * zostala juz wczytana ) dlatego przedtem
                    * trzeba wczytac standartowa konfiguracje.
                    * Struktury _EL_CFG i _PN_CFG sa umieszczone w
                    *  epromach.  Ich  odpowiedniki  (  bez
                    * podkreslenia ) beda uzywane w programie i sa
                    *  w  pamieci  RAM.  Procedura  memcpy  kopiuje
                    * obszary pamieci ( biblioteczna procedura ).
                    *  Opis  struktor  opisujacych  konfiguracje
```


72

```
                    * robota zostal zawarty w dalszej czesci
                    * opisujacej zmienne globalne.
                    */
                    memcpy( EL_CFG, _EL_CFG, sizeof( EL_CFG ) );
                    memcpy( PN_CFG, _PN_CFG, sizeof( PN_CFG ) );
                    get_config();  /* konfigurowanie robota   */
                    /* opis procedury w punkcie b4            */
                    /* synchronizacja robota                  */
            case SYNCHRONIZATION :
                        synchronise(); break;
            case  EMERGENCY_STOP   :
                        /*  wyslanie  do  sterownikow  sygnalu
                         * zatrzymania osi, zapalenie lampek
                          *  sygnalizujacych  stop  awaryjny,
                         * zacisniecie hamulcow
                         */
                        emrg_stop_handle(); break;
            case
                            ...
            default :
                        /* niemozliwa sytuacja oznaczajaca
                         * bledne dzialanie programu
                         */
        }

        system();
}
```

Trzy kropki pomiedzy kolejnymi instrukcjami oznaczaja mozliwe dalsze instrukcje sytuacji blednych. Proponowane sytuacje bledne beda wyszczegolnione w dalszej czesci opisu. Konstrukcja taka umozliwi elastyczna budowe oprogramowania. Dla dowolnych sytuacji moga byc realizowane procedury ich obslugi, lub moga byc tylko ustawiane odpowiednie zmienne globalne wedlug ktorych bedzie postepowala dalej procedura system .

Omowienie tej procedury nastapi po ogolnym omowieniu kolejnych podpunktow (B i C) i przedstawieniu dostepnych globalnych zmiennych.

Ad B)

OPROGRAMOWANIE PANELU PROGRAMOWANIA

Do realizacji tych procedur zostanie wykorzystana pewna czesc istniejacego oprogramowania dla robotow IPR-6/60. Beda to procedury obslugi panelu programowania. Po nizbednych modyfikacjach beda one sluzyly do tworzenia i edycji programu, oraz bezposredniego sterowania robotem z wykorzystaniem panelu programowania.

Tak jak w poprzedniej wersji program umieszczony w panelu (procesor Intel 8080) bedzie obslugiwal wszystkie przyciski i dbal o to ktore z nich moga byc aktywne. Wszelkie przewijania MENU, budowa instrukcji, beda sie odbywaly bez komunikacji z programem sterujacym.Komunikacja odbywa sie przez zlacze szeregowe typu RS232C. Informacja przekazywana jest w postaci przesylek. Format przesylki jest nastepujacy:

:	kod	bajty informacyjne	suma kontrolna

Wszystkie przesylki rozpoczynaja sie znakiem ":" (dwukropka). Odbior przesylki zaczyna sie wiec dopiero po odebraniu tego znaku. Wszyskie znakipoprzedzajace dwukropek sa ignorowane. Kazda przesylka ma przypisany swoj kod identyfikacyjny. Kod ten pozwala naokreslenie oczekiwanej dlugosci przesylki i na zinterpretowanie jej zawartosci.

Przesylka moze zawierac bajty informacyjne. Ilosc bajtow zalezna jest od typu przesylki. Wiele przesylek zawiera jedynie kod a nie ma bajtow informacyjnych. Kazda przesylka zakonczona jest suma kontrolna. Suma kontrolna jestuzupelnieniem do 0 sumy modulo 256 wartosci (przed zakodowaniem) kodu przesylki wartosci bajtow informacyjnych. Kod przesylki, bajty informacyjne i suma kontrolna przesylane sa w postaci zakodowanej. Zakodowanie bajtu oznacza tu przedstawienie go w postaci dwochznakow ASCII z ktorych pierwszy jest przedstawiona w postaci znakowej cyfra heksadecymalna odpowiadajaca starszym czterem bitom, a drugi znak jest przedstawiona w postaci znakowej cyfra heksadecymalna odpowiadajaca mlodszym czterem bitom. Np. bajt o wzorcu bitowym 10110111 pozakodowaniu bedzie mial postac znakowa 'B7' tzn. beda to dwa bajty o wzorcach bitowych 01000010 (kod ASCII litery 'B')

to dwa bajty o wzorcach bitowych 01000010 (kod ASCII litery 'B') i 00110111 (kod ASCII cyfry 7).

Obsluga kolejnych przesylek odbywa sie oczywiscie na poziomie przerwan. Przesylki sa umieszczane w kolejce . Kazda z nich ma okreslony priorytet ktory mowi czy dana przesylka moze byc usunieta, czy tez nie. Przesylki z najwyzszym priorytetem sa ustawiane na poczatku kolejki wraz z jednoczesnym usunieciem z niej przesylek o pozwalajacych na to priorytetach.
Do obslugi przesylek (z punktu widzenia inych programow) wystarcza dwie procedury :

```
typedef parcel *parcelptr;
    void putparcel( parcelptr, mod );
```

Dostepne sa trzy mody :

- wait

  wyjscie z procedury nastapi dopiero po pomyslnym zakolejkowaniu przesylki. Oznacza to, ze jesli kolejka przesylek bedzie pelna to wyjscie z procedury nastapi dopiero po zwolnieniu miejsca w kolejce i wstawieniu do niej przesylki.

- immediate

  Wszystkie przesylki z arybutem "mozliwy do usuniecia" sa usuwane z kolejki (poczynajac od ostatnio wstawionej). Na pierwsze wolne miejsce wstawiana jest aktualna przesylka.

- lack

  Dzialanie jest podbne jak dla przypadku "wait". Roznica polega na tym, ze przesylka otrzymuje atrybut "niemozliwa do usuniecia ", co oznacza, ze na pewno bedzie zrealizowana.

```
    parcelptr getparcel( );
```

Procedura ta zwraca adres pola w ktorym znajduje sie przesylka. Gdy kolejka przesylek jest pusta to pod tym adresem bedzie przesylka o kodzie rownm zero. Dzialanie tej procedury usuwa odebrana przesylke z kolejki przesylek.

```
    int getparcelcode();
```

Funkcja ta zwraca kod przesylki bedacej na poczatku kolejki bez jej usuwania. Oczywiscie brak przesylki daje kod zero.

Opisane procedura stanowi polaczenie pomiedzy innymi programami, a panelem programowania co zapewnia przyslanianie szczegolow implementacyjnych obslugi zlacza. Dokladny opis kodow poszczegolnych przesylek, ich struktury i dlugosci znajduje sie w Dodatku A.

Na poziomie przesylek obslugiwana jest tez budowa programu. Dzieki temu w duzym stopniu w jego budowie uczestniczy procesor 8080 . Budowa programu i jego modyfikacje sa niewidoczne z punktu widzenia glownego programu. Program system czyta kolejne instrukcje programu umieszczone w pamieci w postaci tablicy i interpretuje je. Wszelkie uzupelniania programu i jego modyfikacje powoduja rozsuwanie i sciesnianie obszaru programu. Dokladna charakterystyka obszarow programow zostala zawarta w Dodatku B. Dla ulatwienia pisania pozostalych programow (w tym glownie programu interpretera) przyjmuje sie, ze dla danego programu (zgodnie z zalozeniami zaszyta jest na stale liczba programow rowna jeden) dostepne sa :

```
char *beg_prog;      /* adres pierwszej instrukcji */
char *actual_instr; /* adres aktualnej instrukcji */
```

Do poruszania sie po tablicy instrukcji sluzy tablica dlugosci poszczegolnych instrukcji :

```
int instr_size[ INSTRUCTION_TYPES ]
```

Np. przejscie do kolejnej instrukcji odbedzie sie w nastepujacy sposob :

```
actual_instr += instr_size[ actual_instr->code ];
```

W dodatku C znajduja sie wyszczególnione wszystkie typy instrukcji i ich opis w bajtach. W rozdziale poswieconym programowi interpretera zostaly przedstawione odpowiadajace im typy w jezyku C.

Ad C)

Programy bezposredniego styku z modulami robota.

Do programow tych zaliczane sa programy obslugi panelu operacyjnego i poszczegolnych modulow.

a) obsluga panelu operacyjnego :

Sklada sie z trzech podstawowych procedur :

a1)

funkcji odzytujace kod przycisku . Poniewaz w innych implementacjach kod klawisza moze miec rozna dlugosc, dlatego zdefiniowany zosrtanie typ button_code (w naszej implementacji jest to typ int):

```
button_code read_button()
{
     /* odzytaj wejscie pod ktorym znajduje sie kod przycisku i
      * zwroc znormalizowany kod
      */
}
```

a2)

Programu uruchomianego w trakcie wszystkich czynnosci pracy programu system wymagajacych czasu takich jak :

- wykonywanie instrukcji pozycjonowania
- obslugi pamieci kasetowej
- wykonywanie instrukcji czekania warunkowego

w celu sprawdzenia czy nie nadszedl sygnal przerwania pracy

```
int stop_pressed()
{
     switch( read_button() )  /* pobranie kodu przycisku z panelu
                               * operacyjnego
                               */
     {
          case EMERG_STOP :   EMRG_STOP();
          case STOP       :   return(1);
          default :/* Wez kod przesylki -> usuwajac ja z kolejki,
                    * gdyz w trakcie pracy ignorowane sa wszystkie
                    * klawisze z wyjatkiem klawisza stop
                    */
                    switch( *getparcel() )
                    {
```


47

```
                    case ERG_STOP : EMRG_STOP();
                    case CSTOP    : return(1);
                    default       : break;
                }
        }
        return( 0 );
}
```

W przewidywanej implementacji procedura EMRG_STOP() jest w rzeczywistosci makrem :

```
#define EMRG_STOP()     longjmp( error_handle, EMERGENCY_STOP )
```

Postac ta zachowano dla ewntualnych zmian w koncepcji obslugi stopu awaryjnego.

a3)

Obsluga panelu programowania wolana przez procedure system. program ten interpretuje wszystkie przyciski panelu operacyjnego .

```
void oppanel()
{
    while ( TRUE )
        switch( read_button() )
        {
            case EMERGENCY_STOP : EMRG_STOP(); return;
            case START          :/* uruchom program poczawszy
                                  * od pierwszej instrukcji.
                                  * Wymaga to rowniez pewnych
                                  *  pomocniczych  dzialan
                                  * zamieszczonych w opisie do
                                  * procedury run()
                                  */
                                  run(); break;
            case READ_PK        :
                /*    parametrami    formalnymi    programow
                 * czytajacych i piszacych z ( na ) tasme
                 * magnetyczna sa iloczyny logiczne masek
                 * bitowych . Kazda z nich oznacza pozwolenie
                 * na zapis ( odzczyt ) programu robota i
```

```
                    * konfiguracji.
                    */
                                        read_pk( PRG : CONFIG );
                                        break;
                case WRITE_PK       : write_pk( PRG : CONFIG );
                                        break;
                case STOP           :
                default             :  return;
            }
}
```

b) sterowanie modulami robota

b1) Moduly pneumatyczne

Zmienna globalna opisujaca konfiguracje modulow pneumatycznych :

```
unsigned pn_max_waiting_time;  /* Maksymalny czas oczekiwania na
                                * potwierdzenie w przypadku modulow
                                * ze sprzezeniem. W przypadku jego
                                * uplyniecia sygnalizowany jest blad
                                */
typedef struct {
     unsigned char out_port;  /* numer pierwszego wyjscia pod *
                               * ktorym jest dany modul        */
     unsigned char in_port;/* numer wejscia aktywnego w     *
                               * przypadku istnienia sprzezenia */
     unsigned not_exist : 1;/* modul nie istnieje             */
     unsigned two_states: 1; /* czy jest to modul dwustanowy,  *
                               * czy trojstanowy               */
     unsigned back_feed : 1; /* czy istnieje sygnal sprzezenia *
                               * zwrotnego                     */
     pn_pos_dscr pos_synhr: 2; /* pozycja przy synchronizacji :   *
                               * 0 -> polozenie srodkowe
                               * 1 -> polozenie '-'
                               * 2 -> polozenie '+'
                               */
     unsigned time          ; /* czas czekanaia w przypadku gdy *
```

```
                                        * modul nie posiada sprzezenia    */
} pn_module_descriptor;
pn_module_descriptor PN_CFG[ MAX_PN_MODULES ];
readonly pn_modules_descriptor _PN_CFG[ MAX_PN_MODULES ];
  \
```

Oznaczenie literowe poszczegolnych modulow wystepuje tylko w komunikacji z uzytkownikiem. Wewnatrz oprogramowania kolejnym modulom sa przyporzadkowane kolejne liczby od zera do MAX_PN_MODULES - 1.
Na poczatku pracy oprogramowania tablica ta PN_CFG jest wypelniana standartowymi wartosciami umieszczonymi w pamieci EPROM (tablica _PN_CFG). Do zmian w konfiguracji modulow pneumatycznych sluzy procedura get_config() uruchamiana odpowiednim rozkazem z panelu programowania. Zostanie ona opisana po przedstawieniu opisu modulow elektrycznych. Sterowaniem modulu pneumatycznego zajmuje sie procedura pn_handle :

```
int pn_handle( no, direct )
int no;  /* numer modulu
          */
int direct;     /*  0 -> kierunek '0'
                 *  1 -> kierunek '-'
                 *  2 -> kierunek '+'
                 */
{
```

Program jest za dlugi aby zamiescic go w projekcie .
Dla realizatorow projektu wazna jest jedynie komunikacja z tym programem. Procedura ta zwraca nastepujace wartosci :

0 -> brak bledow
1 -> numer nie istniejacego modulu
2 -> brak sygnalu potwierdzenia -> blad sprzetowy
W obecnej wersji projektu procedura ta nigdy nie zwroci tej wartosci, gdyz w takim przypadku nastapi skok do procedury obslugi bledu. Dla zachowania mozliwosci zmiany tej koncepcji przyjmuje sie we wszystkich procedurach z chwila pojawienia sie bledu sprzetowego wywolanie procedury: .

```
                    HRDW_ERROR_HANDLE( error_code );
```

W obecnej implementacji jest to rownowazne odpowiedniej funkcji :

```
void HRDW_ERROR_HANDLE( code )
{
     ERRORS.hrdw = code;
     longjmp( error_handle, HRDW_ERROR );
}

all_pn_handle( synhr, pos )
int synhr;  /* czy jest to proces synchronizacji */
unsigned pos[]; /* jest to tablica opisujaca docelowe pozycje *
                 * Kazde jej slowo opisuje cztery moduly, poniewaz
                 * pozycje kazdego modulu opisuja dwa bity
                 */
{
     procedura ta przeglada tylko istniejace moduly i ustawia ich
     pozycje zgodnie z zadana. Przy okazji ustawiania modulow bez
     sprzezenia wyznaczany jest maksymalny czas po ktorym mozemy
     uznac ze wszystkie moduly bez sprzezenia uzyskaly zadane
     polozenie.  Dla modulow ze sprzezeniem brak sygnalu
     potwierdzenia w zadanym maksymalnym czasie powoduje
     zgloszenie wyjatku bledu sprzetowego. Opisane bledy nie beda
     sygnalizowane w przypadku synchronizacji. Nastapi wtedy
     wypisanie tekstu "popraw modul nazwa_modulu". Tekst ten
    bedzie sie pojawial na wyswietlaczu dopoki nie nadejcie sygnal
     potierdzenia od modulu, lub uzytkownik nacisnie stop (
     funkcja stop_pressed jest testowana za kazdym obiegiem petli
     po poszczegolnych modulach).
}
```

b2) Moduly elektryczne

Struktury opisujace moduly elektryczne maja nastepujaca postac :

```
unsigned el_max_waiting_time;
readonly unsigned EL_LIN_MAX_SPEED;
```

```
unsigned el_lin_max_speed;     /* Maksymalna predkosc dla modulow
                               * liniowych.
                               */
unsigned el_lin_base_speed;    /* Podstawowa predkosc, dla ktorej
                                * realizowane sa instrukcje poz.
                                */
readonly unsigned EL_ROT_MAX_SPEED;
unsigned el_rot_max_speed;     /* analogicznie jak poprzednio */
unsigned el_rot_base_speed;

typedef
struct {
     unsigned not_exist : 1;   /* modul nie istniejacy */
     unsigned linear    : 1;   /* czy jest to modul liniowy */
     } el_module_descriptor;

el_module_descriptor EL_CFG[ MAX_EL_MODULES ];
readonly el_module_descriptor _EL_CFG[ MAX_EL_MODULES ];
```

Do sterowania modulami elektrycznymi robota beda sluzyly dwie glowne procedury :

```
el_handle( data_descriptor )
struct {
          unsigned    no : 2; /* numer modulu */
          unsigned dummy : 3; /* atrapa       */
          unsigned adapt : 1; /* czy pozycjonowanie adaptacyjne */
          unsigned speed : 1; /* czy zadana predkosc, czy czas  */
          unsigned rough : 1; /* czy pozycjonowanie zgrubne     */
          unsigned data  ;    /* wartosc predkosci lub czasu    */
          unsigned n_incr;    /* pozycja docelowa */
} *data_descriptor;
{
```

Zwracana wartosc : 0 - brak bledow
1 - modul nie istnieje

W przypadku napotkania ograniczen ustawiana jest globalna flaga bledu ERRORS.hrdwr i nastepuje skok do procedury obslugi

bledow sprzetowych. Kroki te nie sa wykonywane w przypadku gdy robot jest w trakcie synchronizacji (synchro_work == TRUE).
)

all_el_handle(int synchr, unsigned pos[])
Analogicznie jak w prypadku modulow pneumatycznych procedura ta moze synchronizowac moduly elektryczne robota, lub dokonywac pozycjonowania modulow. Typy parametrow sa takie same jak i dla modulow pneumatycznych, rozna jest ich wewnetrzna interpetacja, gdyz dla modulow elektrycznych parametrem pozycjonowania jest jedno slowo (dwa bajty).
(

W trakcie synchroniacji sprawdzane jest żadanie panelu programowania przelaczenia w tryb pracy recznej. Wyswietlane jest wtedy ostrzezenie o braku synchronizacji i nie sprawdzane sa ograniczenia. Dla ulatwienia przyjmowany jest jednolity okres calkowania dla wszystkich modulow. W przypadku braku potwierdzenia gotowosci odzytu nowych danych (w zadanym maksymalnym czasie) przez ktorykolwiek modul, lub braku synchronizacji (patrz procedura stop zamieszczona przy opisie interpretera) nastepuje przekazanie sterowania do programu obslugi bledow sprzetowych .
)

b3) Praca reczna

Do sterowania recznego robotem sluzy bezparametrowa procedura:
void hand_work()
(

Interpretuje tylko i wylocznie klawisze stopu z panelu operacyjnego oraz klawisze opisane w zalozeniach oprogramowania. Zakonczenie pracy recznej odbywa sie poprzez nacisniecie przycisku "przelacz" lub nacisniecie klawisza "stop". Dzieki temu do sprawdzania kryterium stopu wystarczy testowac procedure stop_pressed, testujac przedtem nacisniecie klawisza "przelacz" (procedura getparcelcode).
)
b4) Konfiguracja robota

Do konfiguracji sluzy procedura:

void get_config()

Zaklada ona, ze jest juz wczytana standartowa tablica z epromow. Proponuje rozne opcje :

- czytanie konfiguracji z tasmy
- wprowadzanie z panelu
- czytanie z epromow

Podanie konfiguracji wszystkich modulow, lub dowolny przycisk stopu konczy dzialanie tej procedury. W jej trakcie wywolywany jest program oppanel, w celu interpretacji klawiszy panelu operacyjnego.

}

PROCEDURA SYSTEM

Jest to program glowny sterowania robotem. Jego szkic wyglada w nastepujacy sposob :

```
void system ()
{
     "wypisz aktualna instrukcje"
     while ( TRUE )
     {
          oppanel();
          switch( "kod przesylki z panelu programowania " ){
          case EMERGENCY_STOP : EMRG_STOP(); break;
          case PR_RECZNA      : auto_work(); break;
          case PRZELACZ       : hand_work(); break;
          case EDIT           : edit();      break;
          deflaut             :
                                     break;

          }
     }
}
```

Kazda z procedur pisanych malymi literami ma za zadanie interpretacje tylko aktywnych klawiszy (i wolanie procedury oppanel). Dla kazdej z nich inne klawisze funkcyjne powoduja

zakonczenie jej dzialania i powrot do procedury system. Dlatego wewnatrz ich nie powinny byc uzywane procedury getparcel, lecz getparcelcode() i w przypadku odpowiedniego kodu nastapi dopiero wtedy odebranie przesylki.

Zaklada sie, ze w procedurach auto_work i edit mozliwe jest przelaczenie sie na prace reczna. Dlatego z pracy recznej wywolanie innych stanow spowoduje zakonczenie dzialania tej procedury (kod przesylki zostaje tylko sprawdzony ale bez odebrania).

Procedura pracy recznej wola interpreter z parametrami zaleznymi od trybu pracy. W przypadku uruchomienia interpretera poczawszy od pierwszej instrukcji nalezy najpierw wywolac program resetintrpr(). Program ten i program interpretera zostaly opisane w nastepnym rozdziale.

PROJEKT PROGRAMU INTERPRETERA.

Poniewaz przewidziana w zalozeniach liczba instrukcji jest mala, kazdej instrukcji bedzie odpowiadala jedna procedura. W celu przyspieszenia znalezienia odpowiedniej funkcji dla kazdej z instrukcji zostanie wprowadzona tablica wskaxnikow na poszczegolne procedury. Bedzie ona zmienna widoczna tylko przez program interpretera (atrybut STATIC). Postac tej tablicy bedzie nastepujaca:

```
int (*instr_service[])() = {
                                    empty,
                                    pose,
                                    posp,
                                    posa,
                                    defvl,
                                    defva,
                                    set,
                                    wait,
                                    waitco,
                                    jump,
                                    jumpco,
                                    loop,
                                    lpend,
                                    patt,
                                    modify
                                };
```

Parametrem formalnym interpretera beda dwie zmienne :

step_work :

1 - wykonaj tylko jedna instrukcje

0 - praca ciagla

- period :

0 - praca bez zatrzymywania

n - liczba cykli

Liczba n_period zostanie podstawiona pod statyczna (widoczna tylko dla interpretera zmienna :

```
static int n_periods
```

Procedura nie zwraca zadnych wartosci. Ustawiana jest globalna strukture ERORS. Zostala ona opisana w poprzednich rozdzialach. Jest to struktura (czterobajtowa) ktorej pierwsza skladowa oznacza bledy (programu), a druga bledy sprzetowe.

```
struct ERRORS {
                    unsigned prog ;
                    unsigned hrdwr;
              }
```

Taki zapis bledow jest spojny i umozliwia przypisywanie bledow o tym samym numerze innym sytuacjom. W rzeczywistosci wiekszosc bledow sprzetowych spowoduje bezwzgledny skok do procedury obslugi bledow (bedacej w programie glownym).
Inne zmienne z ktorych korzysta interpreter to :

- opis parametrow programu :

```
extern char *actual_instr -> adres aktualnej instrukcji
extern char *first_instr  -> adres pierwszej instrukcji
extern int  instr_size[]  -> tablica rozmiarow instrukcj
```

- opis konfiguracji modulow robota
- flaga stop_signal;

Zmienne statyczne

Zmienne statyczne dla obslugi petli, wzoru i modyfikacji opisane zostaly w realizujacych je procedurach.

Interpreter jak i wszystkie programy korzystajace z procedur sterujacych ruchem robota korzystaja z opisanej procedury stop_pressed() sprawdzajacej nadejscie sygnalu stopu. Z procedury tej korzysta funkcja stop(). Bedzie ona sprawdzala poszczegolne urzadzenia zewnetrzne. W przypadku napotkania bledow sprzetowych sygnalizowanych przez procedure sprawdzajaca stan osi takich jak :

brak synchronizacji

limit bledow

nastapi "dlugi skok" do procedury obslugi bledu. Przedsiebrane wtedy kroki zostaly opisane przy procedurze obslugi bledow. Taki

sam skok (tylko inna obsluga bledu) nastapi w wypadku nacisniecia na dowolnym z paneli (operacyjnym lub programowania) sygnalu natychmiastowego stopu (EMERGENCY_STOP). W przypadku zadania zatrzymania programu zostanie tyko ustawiona flaga stop, ktorej kod bedzie zwrocony. Szkic programu bedzie wygladal nastepujaco

```
int stop()
{
     axis_controler();    /* wola procedure obslugi bledow  */
     check_memory();      /* sprawdza pamiec */

     if ( stop_presssed() )
          stop_signal = TRUE;

     return( stop_signal );
}
```

Procedura stop bedzie wolana w petli glownej programu interpretera oraz we wszystkich procedurach ktore wykonuja sie przez dlugi okres czasu :

- instrukcji czekania
- instrukcjach pozycjonowania

Program glowny interpretera bedzie sie skladal z jednej petli . Bedzie ona wykonywana dopoki nie nastapi sygnal przerywajacy dzialanie interpretera. Z petli tej nastapi wyjscie rowniez w przypadku pracy krokowej (zmienna step_work). Ponizej przedstawiono shemat programu interpretera:

```
void intrprt( int step_work, period )
{
     n_periods = period;
     ERRORS.intrpr = NO_ERRORS ;
     stop_signal = FALSE;
     /* Jezeli aktualna instrukcja jest pierwsza instrukcja
      * programu do dokonaj restartu interpretera polegajacego na
       * usunieciu zawartosci stosu petli i zainicjalizowaniu
```

```
        * zmiennych globalnych z ktorych korzysta interpreter.
        * Zaklada sie, ze czynnosci te bedzie wykonywac funkcja :
        * resetintrpr()
        */

      while ( TRUE )
      {
            instr_code = get_code(actual_instr);
            ((*instr_service[instr_code])(actual_instr))
           /* actual_instr jest adresem poczatku instrukcji
            */
            wyswietl aktualnie wykonywana instrukcje
            if( ERRORS.prog ){
                  wypisz komunikat o bledzie
                  i zakoncz dzialanie petli
            }
            if( stop_signal ||stop() || step_work )
                  break;
      }
}
```

Procedury obslugujace poszczegolne instrukcje.

Dla zwiekszenia czytelnosci programu i efektywnosci kodu wynikowego do przekazywania informacji o instrukcji zostanie uzyty mechnizm jezyka C - rzutowania typow. Do wszystkich procedur jako parametr aktualny przekazywany jest adres poczatku instrukcji (wskaxnik na znak - char*). W kazdej z procedur wskaxnik ten podlega rzutowaniu (zamianie typow) na wskaxnik do struktury zawierajacej opis biezacej instrukcji. Struktury te zostana opisane dla kazdej z procedur. Opis kazdej z procedur bedzie sie skladal z naglowka procedury w standardzie jezyka C oraz krotkiego opisu dzialania procedury.

1. Procedury obslugi petli

a) poczatek petli programowej

```
void loop(instruction) /* sposob wywolania procedury */
struct loopinstr {  char code;     /* kod instrukcji       */
                    int no_instr; /* numer instrukcji      */
                    int n_period; /* liczba powtorzen petli */
                 } *instruction;
```

Dla potrzeb obslugi petli wprowadzone zostaly nastepujace zmienne statyczne (widoczne w calym interpreterze) :

stos obslugiwanych petli :

```
struct {  loopinstr *begin_loop; /* adres poczatku petli */
          int        counter;    /* licznik powtorzen    */
       } loop_stck;
```

wskaznik stosu:

```
int lstck_ptr; /* wskaxnik biezacej petli */
```

Kazde wywolanie procedury loop bedzie powodowalo sprawdzenie czy liczba otwartych petli nie przekracza maksymalnej liczby petli. Tzn. czy lstck_ptr jest mniejszy od stalej N_LOOP. Jezeli tak, to wskaxnik ten jest zwiekszany i dla tej nowej petli sa ustawiane jej skladowe:

```
loop_stck[lstck_ptr].begin_loop = instruction + 1;
loop_stck[lstck_ptr].counter = instruction->n_period;
```

oraz adres biezacej instrukcji jest ustawiany na poczatek kolejnej instrukcji :

```
actual_instr = (char*)(instruction + 1);
```

w przeciwnym razie ustawiana jest flaga bledu:

```
ERRORS.prog = LOOP_OVERFLOW;
```

b) koniec petli programowej

```
void lpend(instruction) /* sposob wywolania procedury */
struct lpndinstr {  char code;     /* kod instrukcji       */
                    int no_instr; /* numer instrukcji      */
                 } *instruction;
```

Jezeli wskaxnik biezacej petli jest mniejszy lub rowny zero, to:

```
ERRORS.prog = NO_OPEN_LOOP;
```

w przeciwnym razie zmniejszny jest licznik cykli dla danej petli. Gdy jest on wiekszy od zera, to wskaxnik biezacej instrukcji bedzie ustawiony na adres poczatku petli:

```
actual_instr = (char*)loop_stck[lstck_ptr].begin_loop;
```

w przeciwnym razie actual_instr staje sie adresem poczatku nastenej instrukcji po instrukcji konca petli:

```
actual_instr = (char*)(instruction + 1);
```

i wskaxnik biezacej petli jest zmniejszany. Jest to rownowazne z zamknieciem tej petli.

2. Procedury obslugi wzoru i modyfikacji.

Procedury te uzywaja trzech zmiennych statycznych (lokalnych, ale widocznych w calym interpreterze)

```
char *patt_addr; /* pierwsza instrukcja po instrukcji wzor */
char *end_patt;  /* adres ostatniej instrukcji wzoru       */
char *modf_addr; /* pierwsza instrukcja  po  ostatniej
                  * instrukcji modify
                  */
```

a) Definiowanie wzoru

```
void patt(instruction)
struct {  char code;
          int no_instr;
       } *instruction;
```

Pod zmienna patt_addr podstawiany jest adres nastepnej instrukcji po instrukcji wzoru:

```
patt_addr = (char *)(instruction + 1);
```

Pod ostatnia instrukcje wzoru podstawiany jest zerowy wskaxnik. Ma to na celu wychwycenie pierwszej instrukcji modyfikacji. Wskaxnik biezacej instrukcji staje sie rowny patt_addr.

b) Definiowanie modyfikacji

```
void modify(instruction)
struct {  char code;
          int no_instr;
       } *instruction;
```

Jezeli patt_addr jest zerowym wskaxnikiem (przy uruchamianiu programow w jezyku C wszystkie zmienne statyczne i globalne sa inicjalizowane jako zerowe) to ustawiana jest globalna flaga bledu

```
ERRORS.prog = NO_PATTERN_FOUND;
```

tzn. przed instrukcja modyfikacji nie wystapila instrukcja wzoru. W przeciwnym razie jezeli end_patt jest wskaxnikiem zerowym, to staje sie on rowny adresowi poczatku biezacej instrukcji:

```
end_patt = actual_instr;
```

,a aktualna instrukcja staje sie nastepna instrukcja po biezacej:

```
actual_instr = (char*)(instruction + 1);
```

Natomiast gdy end_patt jest wskaxnikiem niezerowym to:

- sprawdzamy czy biezaca instrukcja jest instrukcja konca wzoru. Jezeli tak, to wskaznik biezacej instrukcji jest ustawiany na modf_addr.
- w przeciwnym wypadku modf_addr = (char*)(instruction + 1), a adresem biezacej instrukcji bedzie pierwsza instrukcja wzoru:

```
actual_instr = patt_addr;
```

Podsumowujac procedura mialaby nastepujaca budowe:

```
if (patt_addr == NULL)
then {
        ERRORS.prog = NO_PATTERN_FOUND;
     }
else if (end_patt == NULL)
then {
        end_patt = actual_instr;
        actual_instr = (char*)(instruction + 1);
     }
else if (actual_instr == end_patt)
     then
          actual_instr = modf_addr;
     else {
```

```
                modf_addr = (char*)(instruction + 1);
                actual_instr = patt_addr;
            }
```

3. Instrukcje skokow.

a) skok bezwarunkowy

```
void jump(instruction)
struct {  char code;
          int no_instr;
          int dest_instr; /* numer instrukcji docelowej */
       } *instruction;
```

Funkcja korzysta z pomocniczej procedury find_instr, ktorej parametrem jest numer programu i numer instrukcji:
char *find_instr(int no_instr).

Zwraca ona adres poczatku instrukcji lub zerowy wskaxnik (NULL) w przypadku gdy nie ma instrukcji o takim numerze. Glowna czescia tej procedury jest petla przedstawiona ponizej, w ktorej zmienna ptr jest wskaxnikiem na znak (char *ptr).

```
for (ptr = first_instr;
    no_instr != *((int *)(ptr + 1));
    ptr += size_instruction[ *ptr ] )
  {
     if( *ptr == EMPTY ) /* koniec programu */
     then {
               ERRORS.prog = JUMP_NOT_FOUND;
               return(NULL);
          }
  }
return ( ptr );
```

W petli tej zmienna ptr staje sie rowna poczatkom kolejnych instrukcji. Adres poczatku kolejnej instrukcji jest wyliczany za pomoca globalnej tablicy okreslajacej rozmiary poszczegolnych

instrukcji (*ptr okresla kod instrukcji). Warunkiem zakonczenia wykonywania petli jest znalezienie adresu poczatku instrukcji o szukanym numerze. W przypadku napotkania instrukcji konca programu zwracany jest zerowy wskaxnik i ustawiana jest globalna flaga bledu. W przeciwnym razie zwracany jest adres poczatku znalezionej instrukcji (zmienna ptr).

Procedura jump sprawdza czy znaleziony przez funkcje find_instr adres jest zerowym wskaxnikiem:

- Jezeli tak to nastepuje wyjscie z procedury. Wskaxnik aktualnej instrukcji pozostaje niezmieniony.
- Jezeli nie to wartosc globalnej zmiennej okreslajacej biezaca instrukcje (actual_instr) staje sie rowny adresowi znalezionej instrukcji:

```
actual_instr = ptr;
```

b) skok warunkowy

```
jumpco(instruction)
struct (  char code;
          int no_instr;
          int dest_instr; /* numer instrukcji docelowej */
          char relation;  /* kod relacji */
          char l_arg;     /* 1-bitowy lewy argument */
          char r_arg;     /* 1-bitowy prawy argument */
       ) *instruction;
```

Procedura wykonuje te same czynnosci co procedura jump. Jedyna roznica polega na tym, ze ostatnia instrukcja (przypisanie aktualnemu adresowi instrukcji zmiennej ptr) bedzie wykonana tylko i wylacznie w przypadku bezblednego sprawdzenia relacji zachodzacej miedzy obiektami jednobitowymi (sprawdzanej przez funkcje comp_1bval). W przypadku zasygnalizowania przez funkcje blednych argumentow ustawiana jest flaga bledu:

```
ERRORS.prog = INVALID_1BVAL;
```

c) funkcja porownujaca obiekty jednobitowe

int comp_1bval(char l_arg, char r_arg, int *relation)
Funkcja ta zwraca wartosc:

0 - poprawne argumenty
-1 - nieistniejacy lewy argument
1 - nieistniejacy prawy argument

W przypadku sprawdzenia zaleznosci argument relation przyjmuje wartosci:

0 - l_arg == r_arg;
-1 - l_arg < r_arg;
1 - l_arg > r_arg;

Funkcja bedzie korzystac ze zmiennych globalnych (FLAGS[], N_FLAGS - tablica flag oraz ich liczba) oraz z procedur obslugujacych wejscia i wyjscia.

4. Instrukcje czekania.

a) czekanie bezwarunkowe

```
void wait(instruction)
struct {  char code;
          int no_instr;
          int time_wait;
       } *instruction;
```

Funkcja bedzie korzystac ze zrealizowanej dla potrzeb poprzednich wersji robota (IRp 10) procedury zegara. Skladac sie bedzie z jednej petli wykonywanej dopoki odczytana wartosc czasu nie bedzie wieksza lub rowna wartosci czasu w chwili rozpoczecia petli powiekszonej o instruction->time_wait. Po zakonczeniu dzialania petli zwiekszany jest wskaxnik biezacej instrukcji:

```
actual_instr = (char*)(instruction + 1);
```

W petli tej bedzie takze wolana procedura stop() dla sprawdzenia istnienia sygnalow z panelu programowania i operacyjnego.

```
for (t0 = gettime; gettime < t0 + instruction->time_wait; )
{
     if( stop() )
```

```
            return;
    }
```

b) czekanie warunkowe

```
void waitco(instruction)
struct {  char code;
          int no_instr;
          char relation;   /* kod relacji */
          char l_arg;      /* 1-bitowy lewy argument */
          char r_arg;      /* 1-bitowy prawy argument */
       } *instruction;
```

Procedura wywoluje w petli funkcje comp_1bval porownujaca jednobitowe argumenty. Petla ta moze sie zakonczyc w dwoch przypadkach:

- jeden z argumentow jest bledny co spowoduje ustawienie flagi bledu:

```
ERRORS.prog = INVALID_1BVAL;
```

- spelnienia relacji co spowoduje przejscie do kolejnej instrukcji:

Poniewaz w petli tej wolana jest kazdorazowo procedura stop. Spelnien relacji moze byc
zasymulowane z panelu programowania przysiskiem symul. :

```
while ( !condition )
{
     if( stop() )
          return;
     if( parcelcode() == CSYM )
          break;
}

actual_instr = (char*)(instruction + 1);
```

5. Instrukcja ustawiania flag lub wyjsc.

96

```
void set(instruction)
struct {  char code;
          int no_instr;
          char l_arg;      /* 1-bitowy lewy argument */
          char r_arg;      /* 1-bitowy prawy argument */
       } *instruction;
```

Procedura korzystac bedzie z procedury comp_1bval ignorujac zwracana przez nia wartosc relacji pomiedzy argumentami, w celu ustalenia dopuszczalnosci argumentow. W przypadku wystapienia bledu ustawiana jest flaga:

```
ERRORS.prog = INVALID_1BVAL;
```

w przeciwnym razie wartosc prawego argumentu bedzie podstawiona pod lewy argument.

6) Instrukcja konczaca program

```
void empty(instruction)
struct {  char code;
          int no_instr;
       } *instruction;
```

Funkcja ta ustawia zmienna globalna actual_instr na pierwsza instrukcje w danym programie. Sprawdzany bedzie stos petli. Niepusty stos spowoduje ustawienie flagi bledu :

```
ERRORS.prog = NO_CLOSE_LOOP;
```

Nastepnie wolana jest procedura resetintrpr dla zainicjalizowania zmiennych i struktur uzywanych przez interpreter. Funkcja ta nie niszczy zawartosci stosu petli. lecz zmienia tylko wskaznik do poczatku stosu. Dzieki temu mozliwe jest przy podaniu komunikatu o bledzie wskazanie numeru niezamknietej petli.
W przypadku gdy n_periods jest rozne od zera sprawdzany jest licznik cykli. W przypadku gdy jest on mniejszy lub rowny zero ustawiana jest zmienna stop_intrpr na wartosc TRUE.
Licznik cykli jest ustawiany standartowo na wartosc jeden. Zmiana trybu pracy spowoduje zatrzymanie interpretera zgodnie, ze schematem programu zamieszczonym na poczatku tego rozdzialu.
Gdy zaden z powyzej opisanych wariantow nie zachodzi, procedura

nie wykonuje zadnych dalszych czynnosci.

7) Instrukcje ustawiania predkosci

a) Ustawianie predkosci liniowej

```
void defvl( instruction )
struct {  char code;
          int no_instr;
          int base_speed;
          int max_speed;
       } *instruction;
```

Procedura ta ustawia dwie globalne zmienne (opisane w porzednim rozdziale) :

```
el_lin_base_speed = instruction->base_speed;
el_lin_max_speed  = instruction->max_speed;
```

Sprawdzane sa dwie mozliwe sytuacje bledne :

```
el_lin_base_speed > el_lin_max_speed;
el_lin_max_speed > EL_LIN_MAX_SPEED;
```

obie sytuacje spowoduja ustawienie flagi bledu na wartosc

```
ERRORS.prog = INVALID_SPEED;
```

a) Ustawianie predkosci katowej

```
void defvl( instruction )
struct {  char code;
          int no_instr;
          int base_speed;
          int max_speed;
       } *instruction;
```

Procedura ta ustawia dwie globalne zmienne (opisane w porzednim rozdziale) :

```
el_rot_base_speed = instruction->base_speed;
el_rot_max_speed  = instruction->max_speed;
```

Sprawdzane sa dwie mozliwe sytuacje bledne :

```
el_rot_base_speed > el_rot_max_speed;
el_rot_max_speed > EL_ROT_MAX_SPEED;
```

el_rot_max_speed > EL_ROT_MAX_SPEED;

obie sytuacje spowoduja ustawienie flagi bledu na wartosc

ERRORS.prog = INVALID_SPEED;

8) Instrukcje pozycjonowania

Instrukcje te beda korzystaly z procedur nadzorujacych ruch robota opisanych w poprzednich rozdzialach przy opisie procedur obslugi modulow.

Procedury tamte wymagaja podania predkosci w m/s. Dlatego predkosc ta bedzie obliczona jako procent wielkosci podstawowej zaleznej od modulu (liniowego lub obrotowego). Gdy predkosc ta bedzie wieksza od maksymalnej dopuszczalnej to nastapi sygnalizacja tego samego bledu co w przypadku procedury ustawiania predkosci. Ten sam blad bedzie ustawiany w przypadku zwrocenia przez procedure pozycjonowania sygnalu niemoznosci zrealizowania zadanego polozenia w wyznaczonym czasie.

UWAGA !

Proponujemy w wypadku za duzych wartosci predkosci, lub zlego czasu nie sygnalizowac sytuacji blednych, lecz przyjmowac najblizsze dopusczalne wartosci. Predkosc podstawowa przy kazdym restarcie programu interpretera bedzie ustawiana na 50% predkosci maksymalnej.

DODATEK A

Protokol komunikacji miedzy panelem programowania a sterownikiem.

Zestawienie formatow przesylek z panelu programowania do sterownika.

W zestawieniu podano mnemonik przesylki i opis bajtow informacyjnych (jesli takie wystepuja).

CSTINS - wpisz do programu zaprogramowana instrukcja
    byte 0: kod typu instrukcji
         1: mlodszy] - numer instrukcji
         2: starszy]
         3:]
         .] - n bajtow parametrow instrukcji (n zalezy od typu
       2+n:]   instrukcji)

CINSDE - usun instrukcje w podanym zakresie
    byte 0: mlodszy] - numeru pierwszej instrukcji do usuniecia
         1: starszy]
         2: mlodszy] - numeru ostatniej instrukcji do usuniecia
         3: starszy]

CTHISI - przyslij parametry instrukcji o wskazanym numerze
    byte 0: mlodszy] - bajt numeru instrukcji
         1: starszy]

CNEXTI - przyslij numer i parametry nastepnej instrukcji
    byte 0: mlodszy] - bajt numeru instrukcji
         1: starszy]

CPREVI - przyslij numer i parametry poprzedniej instrukcji
    byte 0: mlodszy] - bajt numeru instrukcji
         1: starszy]

CLASTI - przyslij numer i parametry ostatniej instrukcji programu
         brak bajtow informacyjnych

CFRSTI - przyslij numer i parametry pierwszej instrukcji programu
         brak bajtow informacyjnych

100

CMIDNU - przyslij numer instrukcji miedzy biezaca a nastepna
byte 0:] numer biezacej instrukcji
1:]

CSTART - wystartuj program glowny od pierwszej instrukcji
brak bajtow informacyjnych

CSTAIN - wystartuj (kontynuuj) wykonywanie programu od aktualnie wyswietlanej instr.
brak bajtow informacyjnych

CSTEP - wykonaj aktualnie wyswietlana instrukcje (praca krokowa do przodu)
brak bajtow informacyjnych

CCYCLE - wykonaj program okreslona ilosc razy
byte 0: liczba cykli

CSTOP - zatrzymaj automatyczne wykonywanie programu ("stop")
brak bajtow informacyjnych

CSYMUL - symuluj spelnienie warunku
brak bajtow informacyjnych

CFRRAM - przepisz program z RAM do PK1
brak bajtow informacyjnych

CTORAM - przepisz program z PK1 do RAM
brak bajtow informacyjnych

CTAPHO - wstrzymanie zapisu z/do pamieci kasetowej
brak bajtow informacyjnych

CISERR - sygnalizuj blad (kasuj lampke "blad" na panelu operacyjnym)
brak bajtow informacyjnych

CNOERR - kasuj blad (zgas lampke "blad" na panelu operacyjnym)
brak bajtow informacyjnych

CMEMOC - przyslij informacje o zajetosci pamieci programow
brak bajtow informacyjnych

CMANP - zmien polozenie modulu pneumatycznego
bajt 0: bity 0-3 numer modulu pneumatycznego

modul	A	B	C	D	E	F	G	H	K	L
numer	0	1	2	3	4	5	6	7	8	9

101

bity 4-5 kod polozenia

0 - wybranie modulu

1 - polozenie "-" (minus)

2 - polozenie "0" (centralne)

3 - polozenie "+" (plus)

CMANE - zmien polozenie modulu elektrycznego

bajt 0: bity 0-1 numer modulu elektrycznego (1-3)

bity 2-3 kod predkosci ruchu

0 - ruch inkrementalny

1 - 1.5% predkosci maksymalnej

2 - 15% predkosci maksymalnej

3 - 50% predkosci maksymalnej

CGETCO - przyslij konfiguracje robota

brak bajtow informacyjnych

CSETCO - pamietaj konfiguracje robota

bajt 0: typ modulu pneumatycznego A

1: typ modulu pneumatycznego B

2: typ modulu pneumatycznego C

3: typ modulu pneumatycznego D

4: typ modulu pneumatycznego E

5: typ modulu pneumatycznego F

6: typ modulu pneumatycznego G

7: typ modulu pneumatycznego H

8: typ modulu pneumatycznego K

9: typ modulu pneumatycznego L

10: typ modulu elektrycznego nr 1

11: typ modulu elektrycznego nr 2

12: typ modulu elektrycznego nr 3

gdzie: typ modulu pneumatycznego

0 - brak modulu w konfiguracji

1 - modul dwupolozeniowy ze sprzezeniem

2 - modul dwupolozeniowy bez sprzezenia

3 - modul trojpolozeniowy

typ modulu elektrycznego

0 - brak modulu w konfiguracji

1 - modul katowy

2 - modul liniowy

CSYNCH - synchronizuj modul

byte 0: bity 0-3, gdy bit 7 = 0, to numer modulu pneumatycznego,
gdy bit 7 = 1, to numer modulu elektrycznego
bity 4-5 kod polozenia modulu (wazne tylko dla modulow dwupolozeniowych bez sprzezenia)
1 - polozenie "-" (minus)
3 - polozenie "+" (plus)

CSTPSY - stop synchronizacji (wazne tylko dla modulow elektrycznych)

brak bajtow informacyjnych

Zestawienie formatow przesylek ze sterownika do panelu programowania.

W zestawieniu podano mnemonik przesylki i opis bajtow informacyjnych (jesli takie wystepuja).

PTHISI - parametry instrukcji o wskazanym numerze

byte 0: kod typu instrukcji
1: mlodszy] - bajt numeru instrukcji
2: starszy]
3:]
.] - n bajtow parametrow instrukcji
2+n:]

PNEXTI - numer i parametry nastepnej instrukcji

byte 0: kod typu instrukcji
1: mlodszy] - bajt numeru instrukcji
2: starszy]
3:]
.] - n bajtow parametrow instrukcji
2+n:]

PPREVI - numer i parametry poprzedniej instrukcji

byte 0: kod typu instrukcji
1: mlodszy] - bajt numeru instrukcji
2: starszy]

```
          3: ]
          .  ] - n bajtow parametrow instrukcji
        2+n: ]
PLASTI - numer i parametry ostatniej instrukcji programu
    byte 0: kod typu instrukcji
         1: mlodszy ] - bajt numeru instrukcji
         2: starszy ]
         3: ]
         .  ] - n bajtow parametrow instrukcji
       2+n: ]
PFRSTI - numer i parametry pierwszej instrukcji programu
    byte 0: kod typu instrukcji
         1: mlodszy ] - bajt numeru instrukcji
         2: starszy ]
         3: ]
         .  ] - n bajtow parametrow instrukcji
       2+n: ]
PMIDNU - numer instrukcji miedzy biezaca a nastepna
    byte 0: ] nowy numer instrukcji
         1: ]
PINSDE - potwierdzenie usuniecia instrukcji
         brak bajtow informacyjnych
PMEMOC - informacje o zajetosci pamieci programow
    byte 0: zajetosc pamieci (w procentach)
PCONIN - informacja o konfiguracji robota
    bajt 0:  typ modulu pneumatycznego A
         1:  typ modulu pneumatycznego B
         2:  typ modulu pneumatycznego C
         3:  typ modulu pneumatycznego D
         4:  typ modulu pneumatycznego E
         5:  typ modulu pneumatycznego F
         6:  typ modulu pneumatycznego G
         7:  typ modulu pneumatycznego H
         8:  typ modulu pneumatycznego K
         9:  typ modulu pneumatycznego L
        10:  typ modulu elektrycznego nr 1
        11:  typ modulu elektrycznego nr 2
```

12: typ modulu elektrycznego nr 3

gdzie: typ modulu pneumatycznego

0 - brak modulu w konfiguracji

1 - modul dwupolozeniowy ze sprzezeniem

2 - modul dwupolozeniowy bez sprzezenia

3 - modul trojpolozeniowy

typ modulu elektrycznego

0 - brak modulu w konfiguracji

1 - modul katowy

2 - modul liniowy

PSTART - start programu

brak bajtow informacyjnych

PSTOP - stop programu

bajt 0:] numer nastepnej instrukcji

1:]

PSYNST - stop synchronizacji

brak bajtow informacyjnych

PTAPRE - wspolpraca z pamiecia kasetowa

byte 0: = 0 - stop zapisu/odczytu

= 1 - odczyt z PK1

= 2 - zapis do PK1

PERROR - sygnalizuj blad (wyswietl numer bledu)

byte 0: mlodszy] - bajt numeru bledu

1: starszy]

PRESER - kasowanie bledu na panelu programowania

brak bajtow informacyjnych

PDOINS - numer aktualnie wykonywanej instrukcji w pracy automatycznej

byte 0: kod instrukcji

1: mlodszy] bajt numeru instrukcji

2: starszy]

PAXISP - potwierdzenie zmiany polozenia modulu pneumatycznego

bajt 0: bity 0-3 numer modulu pneumatycznego

modul	A	B	C	D	E	F	G	H	K	L
numer	0	1	2	3	4	5	6	7	8	9

bity 4-5 kod polozenia

0 - brak modulu

1 - polozenie "-" (minus)

2 - polozenie "0" (centralne)

3 - polozenie "+" (plus)

PAXISE - potwierdzenie zmiany polozenia modulu elektrycznego

bajt 0: bity 0-1 numer modulu elektrycznego (1-3)

bity 2-3 kod stanu modulu

0 - brak modulu

1 - polozenie zostalo zmienione

2 - modul doszedl do ograniczenia

1: mlodszy] - bajt wspolrzednej polozenia modulu

(wazne, gdy

2: starszy] bity 2-3 maja wartosc 1)

Zestawienie kodow i dlugosci przesylek.

Dlugosc przesylki obliczana jest w nastepujacy sposob:

1 bajt ze znakiem dwukropka (:)	1
2 bajty z zakodowanego kodu przesylki	2
2*n bajtow zakodowanych n bajtow informacyjnych	2*n
2 bajty zakodowanej sumy kontrolnej	2
razem:	5+2*n

a wiec minimalna dlugosc przesylki jest 5.

Przesylki z panelu programowania do sterownika.

Mnemonik	Kod	Dlugosc	Uwagi
	0	5	rezerwa
CSTINS	1	35	maksymalna dlugosc instrukcji jest 15 bajtow
CINSDE	2	13	
CTHISI	3	9	
CNEXTI	4	9	
CPREVI	5	9	
CLASTI	6	5	
CFRSTI	7	5	
CMIDNU	8	9	
CSTART	9	5	
CSTAIN	10	5	
CSTEP	11	5	
CCYCLE	12	7	
CSTOP	13	5	
CSYMUL	14	5	
CFRRAM	15	5	
CTORAM	16	5	
CTAPHO	17	5	
CISERR	18	5	
CNOERR	19	5	
CMEMOC	20	5	
CMANP	21	7	
CMANE	22	7	
CGETCO	23	5	
CSETCO	24	31	
CSYNCH	25	7	
CSTPSY	26	5	

MNEMONIK	KOD	DLUGOSC	UWAGI
	0	5	rezerwa
PTHISI	1	35	max. dl. instrukcji - 15 bajtow
PNEXTI	2	35	max. dl. instrukcji - 15 bajtow
PPREVI	3	35	max. dl. instrukcji - 15 bajtow
PLASTI	4	35	max. dl. instrukcji - 15 bajtow
PFRSTI	5	35	max. dl. instrukcji - 15 bajtow
PMIDNU	6	9	
PINSDE	7	5	
PMEMOC	8	7	
PCONIN	9	31	
	10	5	rezerwa
	11	5	rezerwa
	12	5	rezerwa
	13	5	rezerwa
	14	5	rezerwa
	15	5	rezerwa
PSTART	16	5	
PSTOP	17	9	
PSYNST	18	5	
PTAPRE	19	7	
PERROR	20	9	
PRESER	21	5	
PDOINS	22	11	
PAXISP	23	7	
PAXISE	24	7	

DODATEK B

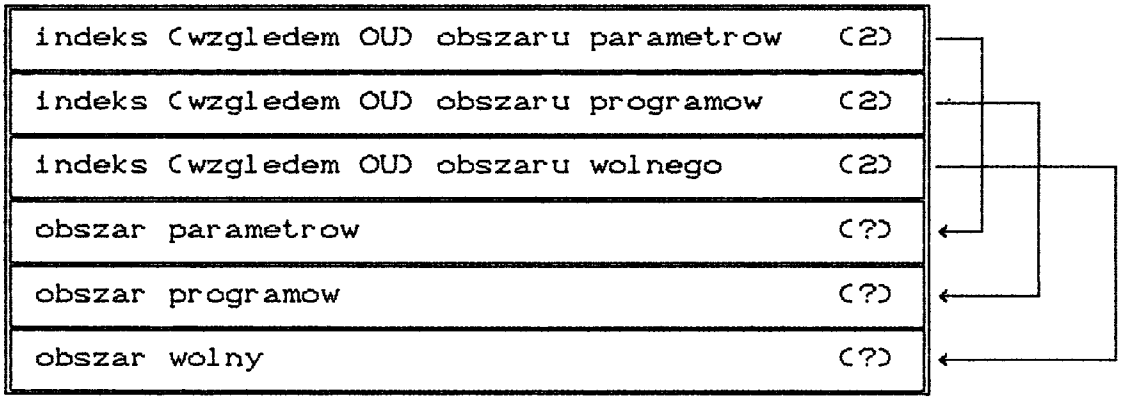
Postac obszaru programow uzytkowych.

Obszar programow uzytkowych jest tak zaprojektowany, aby pozniejsze modyfikacje programu sterujacego nie powodowaly koniecznosci modyfikowania procedurwykorzystujacych dane zawarte w obszarze programow uzytkowych. W strukturze obszaru przewidziane sa takie modyfikacje jak:

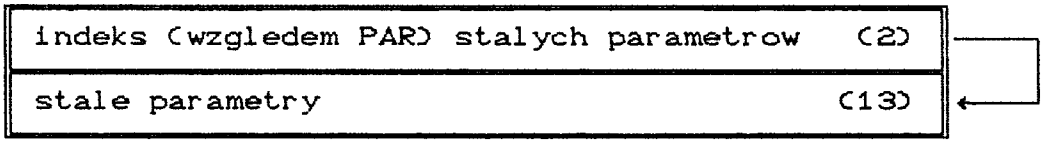
- dodawanie nowych parametrow,
- dodanie mozliwosci przechowywanie w pamieci wielu programow,
- dodanie mechanizmu wywolywania podprogramow.

Przy opisie kazdego pola podana jest dlugosc tego pola w bajtach. Znak ? zamiastdlugosci oznacza, ze pole ma zmienna dlugosc.

obszar uzytkowy (OUD



obszar parametrow (PAR)



stale parametry

typ modulu pneumatycznego:

0 - brak modulu w konfiguracji
1 - modul dwupolozeniowy ze sprzezeniem
2 - modul dwupolozeniowy bez sprzezenia
3 - modul trojpolozeniowy

typ modulu elektrycznego:

0 - brak modulu w konfiguracji

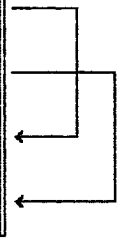
1 - modul katowy

2 - modul liniowy

typ modulu pneumatycznego A	(1)
typ modulu pneumatycznego B	(1)
typ modulu pneumatycznego C	(1)
typ modulu pneumatycznego D	(1)
typ modulu pneumatycznego E	(1)
typ modulu pneumatycznego F	(1)
typ modulu pneumatycznego G	(1)
typ modulu pneumatycznego H	(1)
typ modulu pneumatycznego K	(1)
typ modulu pneumatycznego L	(1)
typ modulu elektrycznego nr 1	(1)
typ modulu elektrycznego nr 2	(1)
typ modulu elektrycznego nr 3	(1)

obszar programow (PRG)

indeks (wzgledem PRG) tablicy opisow programow	(2)
indeks (wzgledem PRG) obszaru cial programow	(2)
tablica opisow programow	(?)
obszar cial programow	(?)



tablica opisow programow

ilosc	(2)
opis programu	(8)
.....	
opis programu	(8)

110

opis programu

numer programu	(2)
indeks (wzgledem PRG) poczatku programu	(2)
indeks (wzgledem PRG) programu wywolujacego	(2)
indeks (wzgledem PRG) kolejnej instrukcji	(2)

obszar cial programow

cialo programu	(?)
.....	
cialo programu	(?)

cialo programu

instrukcja	(?)
.....	
instrukcja	(?)
instrukcja konczaca	(3)

instrukcja konczaca

0	(1)
10000	(2)

instrukcja

Dowolna instrukcja.

obszar wolny

	(?)

111

DODATEK C

Kody i dlugosci instrukcji.

Mnemonik	Kod	Dlugosc	Opis
EMPTY	0	3	Instrukcja pusta
POSE	1	8	Poz. jednego modulu elektrycznego
POSP	2	4	Poz. jednego modulu pneumatycznego
POSA	3	15	Pozycjonowanie wszystkich modulow
DEFVL	4	7	Okreslenie predkosci liniowej
DEFVA	5	7	Okreslenie predkosci katowej
SET	6	5	Ustawienie flagi lub wyjscia
WAIT	7	5	Czekanie
WAITCO	8	6	Czekanie warunkowe
JUMP	9	5	Skok
JUMPCO	10	8	Skok warunkowy
LOOP	11	5	Poczatek petli programowej
LPEND	12	3	Koniec petli programowej
PATT	13	3	Poczatek wzoru
MODIFY	14	3	Poczatek modyfikacji

Forma przechowywania instrukcji.

1. Instrukcja pozycjonowania jednego modulu elektrycznego.

bajt 0 - POSE (kod instrukcji)

bajty 1-2 - numer instrukcji (liczba calkowita)

bajt 3

bity 0-1: numer modulu elektrycznego, ktory ma byc pozycjonowany (liczba od 1 do 3)

bit 5: 0 - pozycjonowanie bez szukania
1 - pozycjonowanie z szukaniem

bit 6: 0 - w bajtach 4-5 zapamietana jest predkosc ruchu
1 - w bajtach 4-5 zapamietany jest czas ruchu

bit 7: 0 - pozycjonowanie zgrubne
1 - pozycjonowanie dokladne (nastepna instrukcja ruchu zostaje rozpoczeta po pojawieniu sie sygnalu INPOS poprzedniej instrukcji)

bajty 4-5 - predkosc lub czas pozycjonowania (decyduje o tym bit 5 bajtu 3)
- predkosc jest wyrazona w dziesiatych czesciach % wartosci predkosci podstawowej
- czas ruchu jest w dziesiatych czesciach sekundy

bajty 6-7 - wspolrzedna polozenia docelowego modulu (w inkrementach)

2. Instrukcja pozycjonowania jednego modulu pneumatycznego.

bajt 0 - POSP (kod instrukcji)

bajty 1-2 - numer instrukcji (liczba calkowita)

bajt 3

bity 0-3: numer modulu pneumatycznego, ktory ma byc pozycjonowany (liczba od 1 do 3)

bity 4-5: kod polozenia docelowego modulu
1 - polozenie "-" (minus)
2 - polozenie "0" (centralne).
3 - polozenie "+" (plus)

3. Instrukcja jednoczesnego pozycjonowania wszystkich modulow:

bajt 0 - POSA (kod instrukcji)

bajty 1-2 - numer instrukcji (liczba calkowita)

bajt 3

bit 5: 0 - pozycjonowanie bez szukania
1 - pozycjonowanie z szukaniem

bit 6: 0 - w bajtach 4-5 zapamietana jest predkosc ruchu
1 - w bajtach 4-5 zapamietany jest czas ruchu

bit 7: 0 - pozycjonowanie zgrubne
1 - pozycjonowanie dokladne (nastepna instrukcja ruchu zostaje rozpoczeta po pojawieniu sie sygnalu INPOS poprzedniej instrukcji)

bajty 4-5 - predkosc lub czas pozycjonowania (decyduje o tym bit 5 bajtu 3)
- predkosc jest wyrazona w dziesiatych czesciach % wartosci predkosci podstawowej

- czas ruchu jest w dziesiatych czesciach sekundy

bajty 6-7 - wspolrzedna polozenia docelowego modulu elektrycznego nr 1 (w inkrementach)

bajty 8-9 - wspolrzedna polozenia docelowego modulu elektrycznego nr 2 (w inkrementach)

bajty 10-11 - wspolrzedna polozenia docelowego modulu elektrycznego nr 3 (w inkrementach)

bajt 12

bity 6-7: kod polozenia docelowego modulu pneumatycznego A

bity 4-5: kod polozenia docelowego modulu pneumatycznego B

bity 2-3: kod polozenia docelowego modulu pneumatycznego C

bity 0-1: kod polozenia docelowego modulu pneumatycznego D

bajt 13

bity 6-7: kod polozenia docelowego modulu pneumatycznego E

bity 4-5: kod polozenia docelowego modulu pneumatycznego F

bity 2-3: kod polozenia docelowego modulu pneumatycznego G

bity 0-1: kod polozenia docelowego modulu pneumatycznego H

bajt 14

bity 6-7: kod polozenia docelowego modulu pneumatycznego K

bity 4-5: kod polozenia docelowego modulu pneumatycznego L

gdzie kod polozenia docelowego modulu pneumatycznego:

1 - polozenie "-" (minus)

2 - polozenie "0" (centralne)

3 - polozenie "+" (plus)

4. Instrukcja deklaracji predkosci liniowej ruchu modulu elektrycznego:

bajt 0 - DEFVL (kod instrukcji)

bajty 1-2 - numer instrukcji

bajty 3-4 - predkosc podstawowa (w mm/s)

bajty 5-6 - predkosc maksymalna (w mm/s)

5. Instrukcja deklaracji predkosci katowej ruchu modulu elektrycznego:

bajt 0 - DEFVA (kod instrukcji)

bajty 1-2 - numer instrukcji

bajty 3-4 - predkosc podstawowa (w deg/s)

bajty 5-6 - predkosc maksymalna (w deg/s)

6. Instrukcja ustawiania flagi lub wyjscia:

bajt 0 - SET (kod instrukcji)

bajty 1-2 - numer instrukcji

bajt 3 - ustawiany obiekt 1-bitowy

bajt 4 - obiekt 1-bitowy, ktorego wartosc podstawiamy pod obiekt w bajcie 3

UWAGA: Obiekt 1-bitowy jest okreslony nastepujaco:

bity 0-5 - numer obiektu (numer flagi, wejscia lub wyjscia) lub wartosc natychmiastowa (0 lub 1)

bity 6-7 - typ obiektu, a mianowicie:

0 - wartosc natychmiastowa

1 - flaga

2 - wejscie

3 - wyjscie

7. Instrukcja czekania bezwarunkowego:

bajt 0 - WAIT (kod instrukcji)

bajty 1-2 - numer instrukcji

bajty 3-4 - czas czekania (w dziesiatych czesciach sekundy)

8. Instrukcja czekania warunkowego:

bajt 0 - WAITCO (kod instrukcji)

bajty 1-2 - numer instrukcji

bajt 3 - kod relacji, ktora ma byc spelniona

0 - rownosc

1 - nierownosc

bajt 4 - lewy argument relacji (obiekt 1-bitowy)

bajt 5 - prawy argument relacji (obiekt 1-bitowy)

9. Instrukcja skoku bezwarunkowego:

bajt 0 - JUMP (kod instrukcji)

bajty 1-2 - numer instrukcji

bajty 3-4 - numer instrukcji, do ktorej ma nastapic skok

10. Instrukcja skoku warunkowego:

bajt 0 - JUMPCO (kod instrukcji)

bajty 1-2 - numer instrukcji

bajt 3-4 - numer instrukcji, do ktorej ma nastapic skok

bajt 5 - kod relacji, ktora ma byc spelniona
0 - rownosc
1 - nierownosc
bajt 6 - lewy argument relacji (obiekt 1-bitowy)
bajt 7 - prawy argument relacji (obiekt 1-bitowy)

11. Instrukcja programowania poczatku petli:
bajt 0 - LOOP (kod instrukcji)
bajty 1-2 - numer instrukcji
bajty 3-4 - liczba powtorzen

12. Instrukcja programowania konca petli:
bajt 0 - LPEND (kod instrukcji)
bajty 1-2 - numer instrukcji

13. Instrukcja programowania wzoru:
bajt 0 - PATT (kod instrukcji)
bajty 1-2 - numer instrukcji

14. Instrukcja programowania modyfikacji:
bajt 0 - MODIFY (kod instrukcji)
bajty 1-2 - numer instrukcji

DODATEK D

Kody przyciskow na panelu programowania.

Program obslugi klawiatury panelu programowania ma mozliwosc wybrania grupy przyciskow i odczytania stanu przyciskow wchodzacych w sklad tej grupy. Przyporzadkowanie przyciskow grupom i pozycjom w grupie jest natepujace:

Kod grupy	Numer bitu w grupie	Znaczenie przycisku	Nr diody zwiazanej z przyc.
0	0 .. 7	niewykorzystane	
1	0	niewykorzystany	
	1	programowanie innych instrukcji	5
	2	programowanie instr.pozycjonowania	4
	3	edycja programu	3
	4	praca auto	2
	5	praca reczna	1
	6 .. 7	niewykorzystane	
2	0	stop programu	
	1	przycisk funkcyjny nr 1	
	2	przycisk funkcyjny nr 2	
	3	przycisk funkcyjny nr 3	
	4	synchronizacja	
	5	uaktywnia przyciski manipulacyjne (przelacz)	
	6	przegladanie menu	
	7	niewykorzystane	
3	0	cyfra 0	
	1	cyfra 1	
	2	cyfra 2	
	3	cyfra 3	
	4	cyfra 4	
	5	cyfra 5	
	6	cyfra 6	
	7	cyfra 7	
4	0	cyfra 8	
	1	cyfra 9	
	2	znak kropki	
	3	kasowanie	
	4	wpis	
	5 .. 7	niewykorzystane	

Kod grupy	Numer bitu w grupie	Znaczenie przycisku	Nr diody zwiazanej z przyc.
5	0	modul pneumatyczny A	15
	1	modul pneumatyczny B	14
	2	modul pneumatyczny C	13
	3	modul pneumatyczny D	12
	4	modul pneumatyczny E	11
	5	modul pneumatyczny F	10
	6	modul pneumatyczny G	9
	7	modul pneumatyczny H	8
6	0	modul pneumatyczny K	7
	1	modul pneumatyczny L	6
	2	ruch modulu elektr. nr 1 na "+"	
	3	ruch modulu elektr. nr 1 na "-"	
	4	ruch modulu elektr. nr 2 na "+"	
	5	ruch modulu elektr. nr 2 na "-"	
	6	ruch modulu elektr. nr 3 na "+"	
	7	ruch modulu elektr. nr 3 na "-"	
7	0	przesuwanie okienka w lewo	
	1	przesuwanie okienka w prawo	
	2 .. 7	niewykorzystane	

Pokretlo ustawiania predkosci ruchu modulow elektrycznych widziane jest przez program jako stan cztwerech najstarszych bitow odczytywanych z portu B ukladu U121. Jednoczesnie moze byc ustawiony tylko jeden z bitow. Przyporzadkowanie wartosciom odczytanym z portu B kodow predkosci jest nastepujace:

Wartosc odczytana z portu (x oznacza bit nieistotny)	Przyporzadkowanie kodow predkosci
0000xxxx	predkosc 0 (nie mozna sterowac modulami elektrycznymi)
1000xxxx	ruch w inkrementach
0100xxxx	predkosc 1.5%
0010xxxx	predkosc 15%
0001xxxx	predkosc 50%

Zestawienie znakow wyswietlanych na panelu programowania.

Tablica kodow znakow.

Kod(dec)	Symbol	Uwagi	Kod(dec)	Symbol
0		spacja	31	C
1	_		32	C'
2	%		33	D
3	->	strzalka prawa	34	E
4	<-	strzalka lewa	35	E,
5	+		36	F
6	,		37	G
7	-		38	H
8	.		39	I
9	/		40	J
10	:		41	K
11	<		42	L
12	=		43	L/
13	rozny	przekreslony	44	M
		znak rownosci	45	N
14	>		46	N'
15	?		47	O
16	[		48	O'
17	]		49	P
18	0		50	Q
19	1		51	R
20	2		52	S
21	3		53	S'
22	4		54	T
23	5		55	U
24	6		56	V
25	7		57	W
26	8		58	X
27	9		59	Y
28	A		60	Z
29	A,		61	Z-
30	B		62	Z'

Wzorce znakow i odpowiadajace im sekwencje kodow.

. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	o o o o o	xxx00000B,	00H
spacja	(0)		_	(1)	
o o . . o	xxx00110B,	06H	. . . . .	xxx11111B,	1FH
o o . . o	xxx00110B,	06H	. . o . .	xxx11011B,	1BH
. . . o .	xxx11101B,	1DH	. . o o .	xxx11001B,	19H
. . o . .	xxx11011B,	1BH	o o o o o	xxx00000B,	00H
. o . . .	xxx10111B,	17H	. . o o .	xxx11001B,	19H
o . . o o	xxx01100B,	0CH	. . o . .	xxx11011B,	1BH
o . . o o	xxx01100B,	0CH	. . . . .	xxx11111B,	1FH
%	(2)		->	(3)	
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . o . .	xxx11011B,	1BH	. . o . .	xxx11011B,	1BH
. o o . .	xxx10011B,	13H	. . o . .	xxx11011B,	1BH
o o o o o	xxx00000B,	00H	o o o o o	xxx00000B,	00H
. o o . .	xxx10011B,	13H	. . o . .	xxx11011B,	1BH
. . o . .	xxx11011B,	1BH	. . o . .	xxx11011B,	1BH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
<-	(4)		+	(5)	
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . . . .	xxx11111B,	1FH	. . . . .	xxx11111B,	1FH
. . o o .	xxx11001B,	19H	o o o o o	xxx00000B,	00H
. . o o .	xxx11001B,	19H	. . . . .	xxx11111B,	1FH
. . . o .	xxx11101B,	1DH	. . . . .	xxx11111B,	1FH
. . o . .	xxx11011B,	1BH	. . . . .	xxx11111B,	1FH
,	(6)		-	(7)	

120

```
. . . . .    xxx11111B,  1FH      . . . . o    xxx11110B,  1EH
. . . . .    xxx11111B,  1FH      . . . . o    xxx11110B,  1EH
. . . . .    xxx11111B,  1FH      . . . o .    xxx11101B,  1DH
. . . . .    xxx11111B,  1FH      . . o . .    xxx11011B,  1BH
. . . . .    xxx11111B,  1FH      . o . . .    xxx10111B,  17H
. . o o .    xxx11001B,  19H      o . . . .    xxx01111B,  0FH
. . o o .    xxx11001B,  19H      o . . . .    xxx01111B,  0FH
   .            ( 8 )                /            ( 9 )

. . . . .    xxx11111B,  1FH      . . . . .    xxx11111B,  1FH
. . . . .    xxx11111B,  1FH      . . . o .    xxx11101B,  1DH
. . o o .    xxx11001B,  19H      . . o . .    xxx11011B,  1BH
. . o o .    xxx11001B,  19H      . o . . .    xxx10111B,  17H
. . . . .    xxx11111B,  1FH      . . o . .    xxx11011B,  1BH
. . o o .    xxx11001B,  19H      . . . o .    xxx11101B,  1DH
. . o o .    xxx11001B,  19H      . . . . .    xxx11111B,  1FH
   :            ( 10 )               <            ( 11 )

. . . . .    xxx11111B,  1FH      . . . . .    xxx11111B,  1FH
. . . . .    xxx11111B,  1FH      . . . . o    xxx11110B,  1EH
o o o o o    xxx00000B,  00H      o o o o o    xxx00000B,  00H
. . . . .    xxx11111B,  1FH      . . o . .    xxx11011B,  1BH
o o o o o    xxx00000B,  00H      o o o o o    xxx00000B,  00H
. . . . .    xxx11111B,  1FH      o . . . .    xxx01111B,  0FH
. . . . .    xxx11111B,  1FH      . . . . .    xxx11111B,  1FH
   =            ( 12 )             rozny          ( 13 )

. . . . .    xxx11111B,  1FH      . . o o .    xxx11001B,  19H
. o . . .    xxx10111B,  17H      . o . . o    xxx10110B,  16H
. . o . .    xxx11011B,  1BH      . . . . o    xxx11110B,  1EH
. . . o .    xxx11101B,  1DH      . . . o .    xxx11101B,  1DH
. . o . .    xxx11011B,  1BH      . . o . .    xxx11011B,  1BH
. o . . .    xxx10111B,  17H      . . . . .    xxx11111B,  1FH
. . . . .    xxx11111B,  1FH      . . o . .    xxx11011B,  1BH
   >            ( 14 )               ?            ( 15 )
```

```
. o o o .    xxx10001B,  11H
. o . . .    xxx10111B,  17H
. o . . .    xxx10111B,  17H
. o . . .    xxx10111B,  17H
. o . . .    xxx10111B,  17H
. o . . .    xxx10111B,  17H
. o o o .    xxx10001B,  11H
   [           ( 16 )
```

```
. o o o .    xxx10001B,  11H
. . . o .    xxx11101B,  1DH
. . . o .    xxx11101B,  1DH
. . . o .    xxx11101B,  1DH
. . . o .    xxx11101B,  1DH
. . . o .    xxx11101B,  1DH
. o o o .    xxx10001B,  11H
   ]           ( 17 )
```

```
. o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH
o . . o o    xxx01100B,  0CH
o . o . o    xxx01010B,  0AH
o o . . o    xxx00110B,  06H
o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H
   0           ( 18 )
```

```
. . o . .    xxx11011B,  1BH
. o o . .    xxx10011B,  13H
. . o . .    xxx11011B,  1BH
. . o . .    xxx11011B,  1BH
. . o . .    xxx11011B,  1BH
. . o . .    xxx11011B,  1BH
. o o o .    xxx10001B,  11H
   1           ( 19 )
```

```
. o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH
. . . . o    xxx11110B,  1EH
. . . o .    xxx11101B,  1DH
. . o . .    xxx11011B,  1BH
. o . . .    xxx10111B,  17H
o o o o o    xxx00000B,  00H
   2           ( 20 )
```

```
o o o o o    xxx00000B,  00H
. . . . o    xxx11110B,  1EH
. . . o .    xxx11101B,  1DH
. . o o .    xxx11001B,  19H
. . . . o    xxx11110B,  1EH
o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H
   3           ( 21 )
```

```
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o o o o o    xxx00000B,  00H
. . . . o    xxx11110B,  1EH
. . . . o    xxx11110B,  1EH
. . . . o    xxx11110B,  1EH
   4           ( 22 )
```

```
o o o o o    xxx00000B,  00H
o . . . .    xxx01111B,  0FH
o o o . o    xxx00001B,  01H
. . . . o    xxx11110B,  1EH
. . . . o    xxx11110B,  1EH
o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H
   5           ( 23 )
```

```
. . . o .    xxx11101B,  1DH
. . o . .    xxx11011B,  1BH
. o . . .    xxx10111B,  17H
o o o o .    xxx00001B,  01H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H
```

6 (24)

```
o o o o o    xxx00000B,  00H
. . . . o    xxx11110B,  1EH
. . . . o    xxx11110B,  1EH
. . . o .    xxx11101B,  1DH
. . o . .    xxx11011B,  1BH
. o . . .    xxx10111B,  17H
o . . . .    xxx01111B,  0FH
```

7 (25)

```
. o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H
```

8 (26)

```
. o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
. o o o o    xxx10000B,  10H
. . . . o .    xxx11101B,  1DH
. . . o . .    xxx11011B,  1BH
. o . . .    xxx10111B,  17H
```

9 (27)

```
. o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o o o o o    xxx00000B,  00H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . . o .    xxx01101B,  0DH
```

A (28)

A, (29)

```
o o o o .    xxx00001B,  01H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o o o o .    xxx00001B,  01H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o o o o .    xxx00001B,  01H
```

B (30)

```
. o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH
o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH
o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H
```

C (31)

```
. o o o o    xxx10000B,  10H      o o o o .    xxx00001B,  01H
o . . o .    xxx01101B,  0DH      o . . . o    xxx01110B,  0EH
o . . . .    xxx01111B,  0FH      o . . . o    xxx01110B,  0EH
o . . . .    xxx01111B,  0FH      o . . . o    xxx01110B,  0EH
o . . . .    xxx01111B,  0FH      o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH      o . . . o    xxx01110B,  0EH
. o o o .    xxx10001B,  11H      o o o o .    xxx00001B,  01H
   C'             ( 32 )              D             ( 33 )

o o o o o    xxx00000B,  00H      o o o o o    xxx00000B,  00H
o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o o o . .    xxx00011B,  03H      o o o . .    xxx00011B,  03H
o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH      o . . . o    xxx01110B,  0EH
o o o o o    xxx00000B,  00H      o o o o .    xxx00001B,  01H
   E             ( 34 )              E,            ( 35 )

o o o o o    xxx00000B,  00H      . o o o .    xxx10001B,  11H
o . . . .    xxx01111B,  0FH      o . . . o    xxx01110B,  0EH
o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o o o . .    xxx00011B,  03H      o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH      o . . o o    xxx01100B,  0CH
o . . . .    xxx01111B,  0FH      o . . . o    xxx01110B,  0EH
o . . . .    xxx01111B,  0FH      . o o o .    xxx10001B,  11H
   F             ( 36 )              G             ( 37 )

o . . . o    xxx01110B,  0EH      . o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH      . . o . .    xxx11011B,  19H
o . . . o    xxx01110B,  0EH      . . o . .    xxx11011B,  19H
o o o o o    xxx00000B,  00H      . . o . .    xxx11011B,  19H
o . . . o    xxx01110B,  0EH      . . o . .    xxx11011B,  19H
o . . . o    xxx01110B,  0EH      . . o . .    xxx11011B,  19H
o . . . o    xxx01110B,  0EH      . o o o .    xxx10001B,  11H
   H             ( 38 )              I             ( 39 )
```


12H

```
. . . . o    xxx11110B,  1EH      o . . . o    xxx01110B,  0EH
. . . . o    xxx11110B,  1EH      o . . o .    xxx01101B,  0DH
. . . . o    xxx11110B,  1EH      o . o . .    xxx01011B,  09H
. . . . o    xxx11110B,  1EH      o o . . .    xxx00111B,  07H
. . . . o    xxx11110B,  1EH      o . o . .    xxx01011B,  09H
o . . . o    xxx01110B,  0FH      o . . o .    xxx01101B,  0DH
. o o o .    xxx10001B,  11H      o . . . o    xxx01110B,  1EH
    J            ( 40 )               K            ( 41 )

o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH      o . o . .    xxx01011B,  09H
o . . . .    xxx01111B,  0FH      o o . . .    xxx00111B,  07H
o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o . . . .    xxx01111B,  0FH      o . . . .    xxx01111B,  0FH
o o o o o    xxx00000B,  00H      o o o o o    xxx00000B,  00H
    L            ( 42 )               L/           ( 43 )

o . . . o    xxx01110B,  0EH      o . . . o    xxx01110B,  0EH
o o . o o    xxx00100B,  04H      o . . . o    xxx01110B,  0EH
o . o . o    xxx01010B,  0AH      o o . . o    xxx00110B,  06H
o . . . o    xxx01110B,  0EH      o . o . o    xxx01010B,  0AH
o . . . o    xxx01110B,  0EH      o . . o o    xxx01100B,  0CH
o . . . o    xxx01110B,  0EH      o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH      o . . . o    xxx01110B,  0EH
    M            ( 44 )               N            ( 45 )

. o . . o    xxx10110B,  16H      . o o o .    xxx10001B,  11H
o . . . o    xxx01110B,  0EH      o . . . o    xxx01110B,  0EH
o o . . o    xxx00110B,  06H      o . . . o    xxx01110B,  0EH
o . o . o    xxx01010B,  0AH      o . . . o    xxx01110B,  0EH
o . . o o    xxx01100B,  0CH      o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH      o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH      . o o o .    xxx10001B,  11H
    N'           ( 46 )               O            ( 47 )
```

. o o . o	xxx10010B,	12H	o o o o .	xxx00001B,	01H
o . . o .	xxx01101B,	0DH	o . . . o	xxx01110B,	0EH
o . . . o	xxx01110B,	0EH	o . . . o	xxx01110B,	0EH
o . . . o	xxx01110B,	0EH	o o o o .	xxx00001B,	01H
o . . . o	xxx01110B,	0EH	o	xxx01111B,	0FH
o . . . o	xxx01110B,	0EH	o	xxx01111B,	0FH
. o o o .	xxx10001B,	11H	o	xxx01111B,	0FH
O'	(48)		P	(49)	
. o o o .	xxx10001B,	11H	o o o o .	xxx00001B,	01H
o . . . o	xxx01110B,	0EH	o . . . o	xxx01110B,	0EH
o . . . o	xxx01110B,	0EH	o . . . o	xxx01110B,	0EH
o . . . o	xxx01110B,	0EH	o o o o .	xxx00001B,	01H
o . . . o	xxx01110B,	0EH	o . o . .	xxx01011B,	09H
o . . o .	xxx01101B,	0DH	o . . o .	xxx01101B,	0DH
. o o . o	xxx10010B,	12H	o . . . o	xxx01110B,	0EH
Q	(50)		R	(51)	
. o o o .	xxx10001B,	11H	. o o o o	xxx10000B,	10H
o . . . o	xxx01110B,	0EH	o . . o .	xxx01101B,	0DH
o	xxx01111B,	0FH	o	xxx01111B,	0FH
. o o o .	xxx10001B,	11H	. o o o .	xxx10001B,	11H
. . . . o	xxx11110B,	1EH	. . . . o	xxx11110B,	1EH
o . . . o	xxx01110B,	0EH	o . . . o	xxx01110B,	0EH
. o o o .	xxx10001B,	11H	. o o o .	xxx10001B,	11H
S	(52)		S'	(53)	
o o o o o	xxx00000B,	00H	o . . . o	xxx01110B,	0EH
. . o . .	xxx11011B,	1BH	o . . . o	xxx01110B,	0EH
. . o . .	xxx11011B,	1BH	o . . . o	xxx01110B,	0EH
. . o . .	xxx11011B,	1BH	o . . . o	xxx01110B,	0EH
. . o . .	xxx11011B,	1BH	o . . . o	xxx01110B,	0EH
. . o . .	xxx11011B,	1BH	o . . . o	xxx01110B,	0EH
. . o . .	xxx11011B,	1BH	. o o o .	xxx10001B,	11H
T	(54)		U	(55)	

```
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
. o . o .    xxx10101B,  15H
. . o . .    xxx11011B,  19H
```

V (56)

```
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
o . o . o    xxx01010B,  0AH
o o . o o    xxx00100B,  04H
o . . . o    xxx01110B,  0EH
```

W (57)

```
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
. o . o .    xxx10101B,  15H
. . o . .    xxx11011B,  1BH
. o . o .    xxx10101B,  15H
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
```

X (58)

```
o . . . o    xxx01110B,  0EH
o . . . o    xxx01110B,  0EH
. o . o .    xxx10101B,  15H
. . o . .    xxx11011B,  1BH
. . o . .    xxx11011B,  1BH
. . o . .    xxx11011B,  1BH
```

Y (59)

```
o o o o o    xxx00000B,  00H
. . . . o    xxx11110B,  1EH
. . . o .    xxx11101B,  1DH
. . o . .    xxx11011B,  1BH
. o . . .    xxx10111B,  17H
o . . . .    xxx01111B,  0FH
o o o o o    xxx00000B,  00H
```

Z (60)

```
o o o o o    xxx00000B,  00H
. . . . o    xxx11110B,  1EH
. . . o .    xxx11101B,  1DH
. o o o .    xxx10001B,  11H
. o . . .    xxx10111B,  17H
o . . . .    xxx01111B,  0FH
o o o o o    xxx00000B,  00H
```

Z- (61)

```
. o o o o    xxx10000B,  10H
o . . . o    xxx01110B,  0EH
. . . o .    xxx11101B,  1DH
. . o . .    xxx11011B,  1BH
. o . . .    xxx10111B,  17H
o . . . .    xxx01111B,  0FH
o o o o o    xxx00000B,  00H
```

Z' (62)