Dokument ten jest dosłownym i niekompletnym tłumaczeniem „RAPID Reference manual” będącego instrukcją obsługi robota IRB 1400. Tłumaczenie to zostało wykonane na użytek własny i Autor nie odpowiada za szkody spowodowane jego błędami (a zapewne jest ich trochę).
Uwagi wysyłać na klang1@wp.pl

Podstawy RAPID-a

(fragmenty)

Spis treści

Wprowadzenie

To jest instrukcja zawierająca szczegółowe objaśnienie języka programowania oraz wszystkich typów danych, instrukcji i funkcji. Jeśli programujesz off-line, ten podręcznik będzie szczególnie użyteczny w tym względzie.

Kiedy zaczynasz programować robota to jest zwykle lepiej jest zacząć z podręcznikiem użytkownika (User's Guide), aż zaznajomisz się z systemem.

1 Inne podręczniki

Przed użyciem robota po raz pierwszy, powinieneś przeczytać rozdział *Podstawowe operacje*. To zaznajomi cię z podstawami działania robota.
Podręcznik użytkownika *The User's Guide* dostarcza instrukcji krok po kroku jak wykonywać różne zadania, takie jak poruszać robota ręcznie, jak go programować lub jak uruchomić program podczas produkcji.

Instrukcja obsługi (*The Product Manual)* opisuje jak instalować robota, a także procedury serwisowe i usuwanie błędów. Ta instrukcja zawiera także specyfikację produktu (*Product Specification),* która dostarcza ogólne właściwości i osiągi robota.

2 Jak czytać ten podręcznik

Aby odpowiedzieć na pytanie „Którą instrukcję powinienem użyć?" lub „Co ta instrukcja oznacza?" zobacz *Przegląd ogólny RAPID-a (RAPID Overview) rozdział 3: Streszczenie RAPID-a*. Ten rozdział krótko opisuje wszystkie instrukcje , funkcje i typy danych pogrupowane zgodnie z listą wyboru, którą używasz podczas programowania. Zawiera także streszczenie składni, co jest szczególnie użyteczne podczas programowania off-line.

Przegląd RAPID-a rozdział 4: Podstawowe właściwości tłumaczą wewnętrzne szczegóły języka. Raczej nie będziesz czytał tego rozdziału, chyba że jesteś doświadczonym programistą.

Przegląd RAPID-a rozdział 5: Ruch i zasady WE/WY opisuje różne systemy współrzędnych robota , jego szybkość i inne właściwości ruchu podczas różnych trybów pracy.

Systemowe Typy Danych i Procedury (System DataTypes and Routines), rozdziały 1-3 opisują wszystkie typy danych, instrukcje i funkcje. Dla twojej wygody zostały opisane w kolejności alfabetycznej. Ten podręcznik opisuje wszystkie dane i programy dostarczone z robotem. Jako dodatek do tego są pewne predefiniowane dane i programy dostarczone z robotem, albo na dyskietce, albo już załadowane do jego pamięci.

Przegląd RAPID-a Rozdział 7: Predefiniowane dane i programy opisuje co się dzieje, jak się je załaduje do pamięci robota.
Jeśli programujesz off-line , znajdziesz pewne wskazówki w *Przeglądzie RAPID-a, rozdziale 6: Programowanie off-line* .

Aby uczynić rzeczy łatwiejsze do znalezienia i bardziej zrozumiałe, *Przegląd RAPID-a, rozdział 8* zawiera *Indeks, Słownik oraz Systemowe typy danych i procedury, rozdziału 4* zawiera indeks.

2.1 Konwencje typograficzne

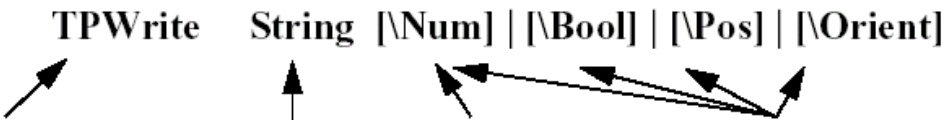
Rozkazy zlokalizowane pod którymkolwiek z pięciu klawiszy menu panelu uczącego są napisane w formie **Menu: Rozkaz**. Na przykład, aby uruchomić drukowanie w menu Plik wybierasz **Plik : Drukuj**. Nazwy na klawiszach funkcyjnych i w polach edycji są wyszczególnione (krój pisma : wytłuszczona kursywa) np. ***Modpos***. Słowa należące do aktualnego języka programowania, takie jak nazwy instrukcji, są pisane kursywą, np. *MoveL*.
Przykłady programów są zawsze pokazane w ten sam sposób, jak są wyprowadzane na dysk elastyczny lub drukarki . Różni się to od tego, co jest pokazywane na panelu uczącym w następujący sposób:
- pewne słowa kontrolne, które są maskowane na panelu uczącym są drukowane, np. słowa wskazujące początek i koniec procedury.
- dane i deklaracje procedur są drukowane w normalnej formie np. *VAR num reg1;*

2.2 Zasady składni

Instrukcje i funkcje są opisywane zarazem w uproszczonej składni lub składni formalnej. Jeśli używasz panelu uczącego do programowania, to w zasadzie powinieneś znać składnię uproszczoną – robot automatycznie upewnia się, że jest używana poprawna składnia.

Uproszczona składnia

Przykład



Instrukcja argument obowiązkowy arg. opcjonalny wzajemnie wykluczające się argumenty

- Opcjonalne argumenty są zamykane w kwadratowych nawiasach [ ]. Te argumenty mogą być opuszczane.
- Argumenty wzajemnie wyłączne nie mogą być tej samej instrukcji jednocześnie , są rozdzielane pionową kreską |.
- Argumenty które mogą być powtarzane arbitralną liczbę razy, są zamykane w nawiasy { }.

2.3 Składnia formalna

Przykład : TPWrite

[String’:=’] <expression (IN) of string>
[’\’Num’:=’ <expression (IN) of num>] |
[’\’Bool’:=’ <expression (IN) of bool>] |
[’\’Pos’:=’ <expression (IN) of pos>] |
[’\’Orient’:=’ <expression (IN) of orient>]’;’

- tekst wewnątrz kwadratowych nawiasów [ ] może być opuszczany.
- Argumenty wzajemnie się wyłączający, tzn. nie mogą być użyte w tej samej instrukcji jednocześnie, są rozdzielane pionową kreską |.
- Argumenty mogące być powtarzane arbitralną liczbę razy są zamykane w nawiasy { }.
Symbole pisane po to, aby otrzymać poprawną składnię, są załączane w apostrofach ' '.
Typ danych argumentu (kursywa) i inne cechy są zamykane w nawiasy <>. Zobacz opis parametrów procedury dla bardziej szczegółowej informacji.
Podstawowe elementy języka i pewnych instrukcji są pisane z użyciem specjalnej Składni, EBNF. Opiera się o te same reguły, ale z pewnymi dodatkami.

Przykład:
GOTO <identyfikator>';'
<identyfikator> ::= <ident>
| <ID>
<ident> ::= <litera> {<litera> | <cyfra> | '_'}

- symbol ::= znaczy „jest definiowany jako”.
- Tekst załączony w nawiasach trójkątnych < > jest definiowany w oddzielnej linii.

Streszczenie RAPID-a

1 Struktura języka

Program składa się z instrukcji opisujących pracę robota. Są określone instrukcje dla różnych poleceń, takie jak ruch robota ,ustawianie wyjścia, itp. Instrukcje ogólnie mają pewna liczbę argumentów, które definiują, co ma miejsce w określonej instrukcji . Dla przykładu, instrukcja resetująca wyjście zawiera argument, który definiuje, które wyjście ma zostać zresetowane. Te argumenty mogą być wyszczególnione w następujący sposób:

- jako wartość numeryczna, np. 5 lub 4.6,
- jako odniesienie do danrj, np. reg1,
- jako wyrażenie, np. 5+reg1*2,
- jako wywołanie funkcji, np. Abs(regt1),
- jako ciąg, np. „Wykonuję część A”

Są trzy typy instrukcji – procedury, funkcje i tzw. „trap routine”.

- procedura jest używana jako podprogram,

-funkcja zwraca wartość określonego typu i jest używana jako argument instrukcji .

-„trap routine” odpowiada na przerwania. „Trap routine” może być kojarzona z określonym przerwaniem np. kiedy jest ustawiane dane wejście.

Informacja może też być przechowywana w danych, np. danych narzędzi (które zawierają wszystkie informacje o narzędziu, takie jak jego TCP i waga) i danych numerycznych (które mogą być użyte np. do policzenia ilości części do przetworzenia). Dane są podzielone na różne grupy danych, które opisują różne typy informacji, jak np. narzędzia, pozycja czy ładunek. Jako że dane mogą być tworzone i mogą być im przydzielane arbitralne nazwy, nie ma limitu (oprócz narzuconego przez pamięć) na liczbę danych. Te dane mogą istnieć zarówno globalnie w programie i lokalnie w procedurze.

Istnieją trzy typy danych – stałe, zmienne i „trwałe” (persistent).

- Stała reprezentuje statyczną wartość i może mieć przydzieloną wartość ręcznie.
- Zmienna może mieć przydzieloną nową wartość podczas wykonywania programu
- „Trwała” może zostać opisana jako trwała zmienna. Kiedy program jest zapisany wartość początkowa odpowiada aktualnej wartości „trwałej”.

Inne cechy języka to:

- parametry procedur,
- arytmetyczne i logiczne wyrażenia,
- automatyczna obsługa błędów,
- modułowe programy,
- wielozadaniowość.

2. Kontrolowanie przepływu programu (program flow control)

Program jest wykonywany sekwencyjnie, tzn. instrukcja po instrukcji. Czasami instrukcje które przerywają sekwencyjnie wykonywanie i wywołują inną instrukcję są wymagane do obsługi różnych sytuacji, które mogą się zdarzyć podczas wykonywania programu.

2.1 Podstawy programowania

Program może być kontrolowany na pięciu różnych sposobów:
- poprzez wywoływanie innej procedury, a następnie po jej wykonaniu kontynuację wykonywanie instrukcji występujących po wywołaniu tej procedury.
- poprzez wykonywanie różnych instrukcji zależnie od spełnionego warunku,
- poprzez powtarzanie sekwencji instrukcji tak długo, aż warunek nie zostanie spełniony,
- poprzez skok do etykiety wewnątrz tej samej procedury,
- poprzez zatrzymanie programu.

2.2 Wywoływanie innej procedury

Instrukcja	Używana do
*ProcCal*l	wywołuje (skacze do) inną procedurę
CallByVar	wywołuje procedurę o specyficznej nazwie
RETURN	wraca do pierwotnej procedury

2.3 Kontrola programu wewnątrz procedury

Instrukcja	Używana do
Compact IF	wykonuje jedną instrukcję jeśli warunek został spełniony,
IF	wykonuje zestaw różnych instrukcji w zależności czy warunek został spełniony, czy nie
FOR	powtarza część programu pewną ilość razy,
WHILE	powtarza zestaw różnych instrukcji tak długo, aż warunek zostanie spełniony,
TEST	wykonuje różne instrukcje w zależności od wartości wyrażenia,
GOTO	skok do etykiety
Label	określa etykietę (nazwa linii).

2.4 Zatrzymywanie wykonywania programu

Instrukcja	używana do
Stop	Zatrzymaj wykonywanie programu
EXIT	Zatrzymaj wykonywanie programu kiedy restart programu jest niedozwolony
Break	Zatrzymaj wykonywanie programu tymczasowo do celu usunięcia błędu (debugging)

2.5 Zatrzymanie aktualnego cyklu

Instrukcja	używana do
EXIT cycle	zatrzymuje aktualny cykl i przesuwa wskaźnik programu do pierwszej instrukcji w głównej procedurze. Kiedy jesy wybrany tryb wykonywania CONT, wykonywanie będzie kontynuowane z następnym cyklem programu.

3 Instrukcje różne

Instrukcje różne są używane do :
- wyznaczania wartości danych,
- czekania danego okresu czasu lub oczekiwania aż warunek zostanie spełniony,
- wstawiania komentarza do programu,
- ładowania modułów programu.

3.1 Nadawanie wartości danych

Danym możne być wyznaczana arbitralna wartość. Mogą być inicjalizowane ze stałą wartością, np. 5 albo zmodyfikowane z wyrażeniem arytmetycznym np. reg1+5*reg3

Instrukcja	używana do
:=	nadanie wartości danej

3.2 Oczekiwanie

Robot może być zaprogramowany do czekania określonego czasu lub czekania, aż pewien warunek jest spełniony; np. do czekania aż wejście jest ustawione.

Instrukcja	używana do
WaitTime	czekaj podany okres czasu lub czekaj, aż robot przestanie się poruszać,
WaitUntil	czekaj aż warunek zostanie spełniony,
WaitDI	czekaj aż cyfrowe wejście jest ustawione,
WaitDO	czekaj aż cyfrowe wyjście jest ustawione

3.3 Komentarze

Komentarzy są wstawiane do programu tylko po to, aby zwiększyć jego czytelność. Wykonywanie programu nie zależy od komentarza.

Instrukcja	zastosowanie
comment	Komentarz programu

3.4 Ładowanie modułów programu

Moduły programowe mogą być ładowane z pamięci masowej do pamięci programu lub usuwane z pamięci programu.
W ten sposób duże programy mogą być uruchamiane z użyciem małej części pamięci.

Instrukcja	zastosowanie
Load	ładuje moduł programowy do pamięci oprogramu,

UnLoad	usuwa moduł programowy z pamięci programu,
Start Load	ładuje moduł programowy do pamięci podczas wykonywania programu
Wait Load	łączy moduł, jeśli został załadowany za pomocą *StartLoad*, z zadaniem programu („program task”)

3.5 Różne funkcje

Instrukcja	zastosowanie
OpMode	czyta aktualny tryb pracy robota
RunMode	czyta aktualny tryb wykonywania programu robota
Dim	pobiera wymiary tablicy
Present	sprawdza, czy opcjonalny argument był obecny przy wywoływaniu procedury
IsPers	sprawdza czy parametr jest „trwały”,
IsVar	sprawdza czy parametr jest zmienną,

3.6 Podstawowe typy danych

Typ danych	żywany do definiowania
bool	dane logiczne (wartości prawda lub fałsz)
num	wartości numerycze (dziesiętne lub integer)
symnum	dane numeryczne z wartością symboliczną
string	ciąg znaków
switch	parametry procedury bez wartosci

3.7 Funkcje konwersji

Funkcja	zastosowanie
StrToByte	konwertuje bajt do ciągu znaków z zdefiniowanym formatem danych bajtu
ByteToStr	konwertuje ciąg ze zdefiniowanym formatem danych do bajtów

4. Ustawienia ruchu

Niektóre z właściwości ruchu robota są określane przez instrukcje logiczne które odnoszą się do wszystkich rodzajów ruchu:
- maksymalna prędkość i przekraczanie prędkości
- przyśpieszenie
- zarządzanie różnymi konfiguracjami robota
- obciążenie
- zachowanie w okolicach szczególnych punktów
- tzw. „soft servo”
- wartości strojenia

4.1 Podstawy programowania

Podstawowe charakterystyki ruchu robota są określane przez dane wyszczególnione dla każdej instrukcji pozycjonowania. Niektóre dane są wyszczególnione w oddzielnych instrukcjach, które odnoszą się do wszystkich ruchów dopóki te dane się nie zmienią.
Ogólne ustawienia ruchu są określane przez instrukcje ale mogą być też odczytywane przez zmienną systemową C+MOTSET lub C_PROGDISP. Wartości domyślne są ustawiane automatycznie (poprzez wykonanie procedury SYS_RESR w module systemowym BASE)
- przy zimnym starcie
- kiedy nowy program jest ładowany
- kiedy program jest startowany od początku

4.2 Definiowanie prędkości

Bezwzględna prędkość jest programowana jako argument w instrukcji pozycjonowania. W dodatku do tego, może być zdefiniowana maksymalna prędkość i przekroczenie prędkości (procent zaprogramowanej prędkości).

Instrukcja	zastosowanie
VelSet	maksymalna prędkość i przekroczenie prędkości

4.3 Definiowanie przyśpieszenia

Kiedy, na przykład, operuje się kruchymi częściami, przyśpieszenie może zostać zredukowane w pewnej części programu

Instrukcja	definiuje
AccSet	maksymalne przyśpieszenie

4.4. Definiowanie zarządzania konfiguracją

Konfiguracja robota jest zazwyczaj sprawdzana podczas ruchu. Jeśli jest stosowany ruch przegubowy osiąga się odpowiednią konfigurację. Jeśli jest stosowany ruch liniowy lub kołowy, robot będzie się przemieszczał w kierunku najbliższej konfiguracji żądanej, ale jest sprawdzane, czy jest to ta sama konfiguracja. Zawsze to można zmienić.

Instrukcja	zastosowanie
ConfJ	kontrola konfiguracji jest włączona/wyłączona podczas ruchu
ConfL	sprawdzanie konfiguracji jest włączone/wyłączone podczas ruchu liniowego

4.5 Definiowanie obciążenia

Aby osiągnąć najlepsze osiągi robota, musi zostać poprawnie zdefiniowane obciążenie.

Instrukcja	zastosowanie
GripLoad	obciążenie chwytaka

4.6 Definiowanie zachowania robota wokół szczególnych punktów

Robot może zostać zaprogramowany do unikania pewnych punktów poprzez automatyczną zmianę orientacji narzędzia.

Instrukcja	definiuje
SingArea	metoda interpolacji przez szczególne punkty

4.7 Przemieszczanie programu

Kiedy część programu musi być przesunięta, można dodać instrukcję przesunięcia programu.

Instrukcja	Używana do
PDispOn	Aktywacja przesunięcia programu
PDispSet	Aktywacja przesunięcia programu przez wyszczególnienie wartości
PDispOff	Dezaktywacja przesunięcia programu
EOffsOn	Aktywacja offsetu zewnętrznej osi
EOffsSet	Aktywacja offsetu zewnętrznej osi przez wyszczególnienie wartości
EOffsOff	Dezaktywacja offsetu zewnętrznej osi

Funkcja	Używana do
DefDFrame	Oblicza przesunięcie programu z 3 pozycji
DefFrame	Oblicza przesunięcie programu z 6 pozycji
ORobT	Usuwa przesunięcie programu z pozycji

4.8 „Soft servo”

Jedna lub więcej osi robota może być ustawiona jako „miękka”. Używając tej funkcji, robot będzie elastyczny (ustępliwy) i może zastąpić, dla przykładu, elastyczne narzędzie

Instrukcja	zastosowanie
SoftAct	włączanie soft servo dla jednej lub więcej osi
SoftDeact	wyłączanie soft servo

4.9 Dopasowywanie wartości strojonych (Adjust the robot tuning values)

Ogólnie rzecz biorąc, osiągi robota są samooptymalizujące, jednak w pewnych ekstremalnych przypadkach może wystąpić np. przekroczenie. Możesz dostosować wartości strojenia, aby otrzymać żądane osiągi.

Instrukcja	Zastosowanie
TuneServo[1]	dostosowuje wartości strojenia robota
TuneReset	reset ustawień strojenia do normalnych wartości
PathResol	dostosowuje rozdzielczość geometrycznej ścieżki

Typ danych	Zastosowanie
Tunetype	reprezentuje wartości strojenia jako stałą symboliczną

4.10 Strefy obszaru roboczego

Do 10 różnych stref można zdefiniować wewnątrz obszaru pracy robota. Mogą być użyte do:
- wskazywania, że punkt środku narzędzia (TCP, tool centre point) jest częścią obszaru roboczego,
- ustalania strefy roboczej dla robota i zapobieganiu kolizjom z narzędziem,
- tworzenia strefy roboczej wspólnej dla dwóch robotów. Tylko jeden robot może pracować w tej strefie w danym momencie.

Instrukcja	zastosowanie
WZBoxDef	definiuje strefę o kształcie prostopadłościanu
WZCylDef	definiuje strefę o kształcie cylindra
WZSphDef	definiuje strefę o kształcie sfery
WZLimSup	aktywuje nadzór granicy
WZDOset	aktywuję strefę do ustawiania cyfrowych wyjść
WZDisable	deaktywuje nadzór nad tymczasową strefą
WZEnable	aktywuje nadzór nad tymczasową strefą
WZFree	wyłącza nadzór nad tymczasową strefą

Typ danych	definiuje
wztemporary	identyfikuje tymczasową strefę
wzstationary	identyfikuje (stałą) strefę
shapedata	opisuje geometrię strefy

4.11 Dane dla ustawień ruchu

Typ danych	używane do definiowania

[1] Tylko gdy robot jest wyposażony w opcję „Advanced Motion”

motsetdata	ustawienia ruchu oprócz przestawiania programu
progdips	przestawienie programu

5. Ruch

Ruchy robota są programowane jako ruchy z jednej pozycji do drugiej. Droga pomiędzy tymi dwoma pozycjami jest automatycznie liczona przez robota.

5.1 Podstawy programowania

Podstawowe charakterystyki ruchu, takie jak rodzaj drogi, są wyszczególniane przez wybieranie odpowiednich instrukcji pozycjonowania. Pozostałe właściwości ruchu są wyszczególniane przez definiowanie danych, które są argumentami instrukcji:

- dane pozycji (końcowa pozycja robota i osi zewnętrznych)
- dane prędkości (żądanej prędkości)
- dane strefy (dokładność pozycji)
- dane narzędzia (np. pozycja TCP- środka narzędzia)
- dane pracy (np. aktualny system współrzędnych)

Niektóre z własności ruchu robota są określane poprzez instrukcje logiczne, które odnoszą się do wszystkich ruchów (patrz rozdział 4: *Ustawienia ruchu*)

- maksymalna prędkość i przekraczanie prędkości
- przyśpieszenie
- zarządzanie różnymi konfiguracjami robota
- obciążenie
- zachowanie w okolicach szczególnych punktów
- tzw. „soft servo”
- wartości strojenia

Robot i jego osie zewnętrzne są pozycjonowane poprzez te same instrukcje. Osie zewnętrzne są przemieszczane ze stałą prędkością, osiągając końcową pozycję w tym samym czasie co robot.

5.2 Instrukcje pozycjonowania

instrukcja	typ ruchu
MoveC	przemieszcza TCP łukiem
MoveJ	ruch przegubu
MoveL	przemieszcza TCP liniowo
MoveAbsJ	bezwzględny ruch przegubu
MoveCDO	przemieszcza TCP okrężnie i ustawia wyjście cyfrowe w połowie drogi
MoveJDO	przemieszcza przegub robota i ustawia wyjście cyfrowe w połowie drogi
MoveLDO	przemieszcza TCP liniowo i ustawia wyjście cyfrowe w połowie drogi
MoveCSync	przemieszcza TCP okrężnie i wykonuje polecenie RAPID-a
MoveJSync	przemieszcza przegub robota i wykonuje polecenie RAPID-a
MoveLSync	przemieszcza TCP liniowo i i wykonuje polecenie RAPID-a

5.3 Przeszukiwanie

Podczas ruchu, robot może np. szukać pozycji danego obiektu. Przeszukana pozycja (wskazana przez sygnał sensora) jest zapamiętywana i może zostać użyta do pozycjonowania robota albo do przemieszczenia programu.

Instrukcja	rodzaj ruchu
SearchC	TCP wzdłuż łuku
SearchL	TCP wzdłuż linii

5.4 Aktywowanie wyjść lub przerwania w określonych pozycjach

Normalnie, logiczne instrukcje są wykonywane w przejściu z jednej instrukcji pozycjonowania do drugiej. Jeśli jednak używana jest specjalna instrukcja ruchu, to może być wykonana logiczna instrukcja gdy robot jest w określonej pozycji.

Instrukcja	zastosowanie
TriggIO	definiuje warunek ustawienia wyjścia w podanej pozycji
TriggInt	definiuje warunek wykonania „trap routine" w podanej pozycji
TriggEquip	definiuje warunek ustawienia wyjścia w podanej pozycji z możliwością kompensacji opóźnienia w zewnętrznym oprzyrządowaniu
TriggC	porusza robota (TCP) kołowo z aktywowanym warunkiem
TriggJ	porusza robota oś-po-osi z aktywowanym warunkiem
TriggL	porusza robota (TCP) liniowo z aktywowanym warunkiem

Typ danych	definiuje
trigdata	warunek wykonania polecenia

5.5 Kontrola ruchu jeśli ma miejsce błąd/przerwanie

Aby porawić błąd albo obsłużyć przerwanie, robot może zostać chwilowo zatrzymany, a następnie uruchomiony od nowa.

Instrukcja	zastosowanie
StopMove	zatrzymaj ruch robota
StartMove	zrestartuj ruch robota
StorePath	zachowaj drogę ostatnio wygenerowaną
RestoPath	odtwórz drogę ostatnio zachowaną

5.6 Kontrola zewnętrznych osi

Robot i zewnętrzne osie są zazwyczaj pozycjonowane za pomocą tych samych instrukcji. Jednakże niektóre instrukcje działają tylko na ruch osi zewnętrznych.

Instrukcja	Zastosowanie
DeactUnit	dezaktywuje zewnętrzną jednostkę mechaniczną
ActUnit	aktywuje zewnętrzną jednostkę mechaniczną

5.7 Niezależne osie

Szósta oś robota lub oś zewnętrzna może być przemieszczana niezależnie od innych ruchów. Obszar roboczy osi może być także kasowany, co skróci czas cykli.

Funkcja	Zastosowanie
IndAMove	Przełącza oś w tryb niezależny i przemieszcza ją do pozycji bezwzględną
IndCMove	Przełącza oś w tryb niezależny i zaczyna ją przemieszczać w sposób ciągły
IndDMove	Przełącza oś w tryb niezależny i przemieszcza ją o odległość delta
IndRMove	Przełącza oś w tryb niezależny i przemieszcza ją do pozycji względną
IndReset	Przełącza oś w tryb zależny i/lub kasuje przestrzeń roboczą
IndInpos	Sprawdza czy niezależna oś osiągnęła pozycję
IndSpeed	Sprawdza czy niezależna oś osiągnęła zaprogramowaną prędkość

5.8 Funkcje pozycji

Funkcja	Zastosowanie
Offs	Dodaje offset do pozycji robota wyrażony we współrzędnych obiektu (tj. przesuwa robota o offset)
RelTool	Dodaje offset do pozycji robota wyrażony we współrzędnych narzędzia
CPos	Odczytuje pozycję robota (tylko współrzędne x, y ,z)
CRobT	Odczytuje pozycję robota (zmienna typu *robtarget*)
CJointT	Odczytuje kąty poszczególnych osi
ReadMotor	Odczytuje kąty poszczególnych silników
CTool	Odczytuje wartość zmiennej typu *tooldata*
CWObj	Odczytuje wartość zmiennej typu *workobject*
ORobT	Usuwa przesunięte programu z pozycji
MirPos	Odbicie lustrzane pozycji względem zadanej płaszczyzny

5.9 Dane ruchu

Dane ruchu są stosowane jako argumenty w instrukcjach pozycjonowania

Typ danych	Definiuje
robtarget	Pozycja końcowa
robtarget	Pozycja końcowa instrukcji *MoveAbsJ*

speeddata	Prędkość
zonedata	Dokładność pozycjonowania (punkt końcowy lub pośredni)
tooldata	System współrzędnych narzędzia i jego obciążenie
wobjdata	System współrzędnych obiektu

5.10 Podstawowe dane ruchu

Typ danych	Definiuje
pos	Pozycja (x, y, z)
orient	Orientacja
pose	Układ współrzędnych (pozycja + orientacja)
confdata	Konfiguracja osi robota
extjoint	Pozycja osi zewnętrznych
robjoint	Pozycja osi robota
o_robtarget	Oryginalna pozycja robota podczas stosowania *Limit ModPos*
o_jointtarget	Oryginalna pozycja robota podczas stosowania *Limit ModPos* dla *MoveAbsJ*
loaddata	Obciążenie
mecunit	Zewnętrzna jednostka mechaniczna

6. Sygnały wejścia/wyjścia

Robot jest wyposażony w pewną liczbę cyfrowych i analogowych wejść/wyjść, których stan może być zmieniany programowo.

6.1 Podstawy programowania

Nazwy sygnałów są zdefiniowane w parametrach systemu i za pomocą tych nazw można je czytać z programu (w przypadku robota w lab. 10 : dox – wyjście cyfrowe, x – jego numer od 1 do 15). Wartość sygnału analogowego lub grupy sygnałów cyfrowych jest podana jako wartość numeryczna

6.2 Zmiana wartości sygnału

Instrukcja	Zastosowanie
InvertDO	odwraca wartość cyfrowego sygnału wyjściowego
PulseDO	generuje impuls na wyjściu cyfrowym
Reset	kasuje wyjściowy sygnał cyfrowy (ustawia 0)
SetAO	zmienia wartość analogowego sygnału wyjściowego
SetDO	zmienia wartość cyfrowego sygnału wyjściowego (wartość symboliczna, np. high/low)
SetGO	zmienia wartość grupy cyfrowych sygnałów wyjściowych

6.3 Odczyt wartości sygnału wejściowego

Wartość sygnału może być odczytywana bezpośrednio, np. IF di1=1 THEN ....

6.4 Odczyt wartości sygnału wyjściowego

Funkcja	używana do odczytu
DOutput	wartości cyfrowego sygnału wyjściowego
GOutput	wartości grupy cyfrowych sygnałów wyjściowych
Aoutput	aktualna wartość analogowego sygnału wyjściowego

6.5 Testowanie wejścia na sygnałach wyjściowych

Instrukcja	używana do
WaitDI	czeka aż wejściowy sygnał cyfrowy jest ustawiony lub skasowany
WaitDO	czeka aż wyjściowy sygnał cyfrowy jest ustawiony lub skasowany

Funkcja	używana do
TestDI	sprawdza, czy cyfrowy sygnał wejściowy jest ustawiony

6.6 Włączanie i wyłączanie modułów WE/WY

Moduły WE/WY są automatycznie włączane przy starcie, ale mogą być wyłączane podczas pracy programu i potem ponownie włączane.

Instrukcja	zastosowanie
IODisable	wyłącza moduł WE/WY
IOEnable	włącza moduł WE/WY

6.7 Definiowanie sygnałów wejściowych i wyjściowych

Typ danych	Definiuje
dionum	Symboliczna wartość sygnału cyfrowego
signalai	Nazwa analogowego sygnału wejściowego
signalao	Nazwa analogowego sygnału wyjściowego
signaldi	Nazwa cyfrowego sygnału wejściowego
signaldo	Nazwa cyfrowego sygnału wyjściowego
signalgi	Nazwa grupy cyfrowych sygnałów wejściowych
signalgo	Nazwa grupy cyfrowych sygnałów wyjściowych

7. Komunikacja

Istnieją cztery sposoby komunikowania poprzez porty szeregowe
- wiadomości mogą być wyprowadzane na zewnątrz za pomocą panelu uczącego i użytkownik może odpowiadać na pytania, takie jak ilość części do przetworzenia
- informacja znakowa może być zapisywana i odczytywana z pamięci masowej. W ten sposób dane produkcji mogą być zachowywane i później przetwarzane na PC. Informacja może też być bezpośrednio drukowana na drukarce połączonej z robotem
- informacje binarne mogą być wysyłane pomiędzy robotem a sensorem
- informacje binarne mogą być wysyłane pomiędzy robotem a innym komputerem np. poprzez protokół komunikacyjny

7.1 Podstawy programowania

Decyzja czy używać informacji znakowej czy binarnej zależy od sposoby w jaki urządzenie, z którym robot komunikuje się, przetwarza informację. W pliku można przechowywać dane w postaci znakowej lub binarnej. Jeśli komunikacja jest wymagana w obu kierunkach równocześnie, binarna transmisja jestem konieczna. Każdy port szeregowy lub plik musi być najpierw otwarty. Podczas otwierania, port/plik otrzymuje deskryptor, który jest potem odnośnikiem podczas zapisu/odczytu. Panel sterujący może być używany cały czas i nie ma potrzeby go otwierać. Tekst i wartości różnych typów mogą być drukowane.

7.2 Komunikowanie z użyciem panelu uczącego

Instrukcja	Zastosowanie
TPErase	czyści ekran panelu uczącego
TPWrite	pisze tekst na ekranie panelu uczącego
ErrWrite	pisze tekst na ekranie panelu uczącego i równocześnie zapisuje tą wiadomość w logu błędów
TPReadFK	nadaje etykiety klawiszom funkcyjnym i czyta który klawisz został wciśnięty
TPReadNum	czyta wartość numeryczną z panelu uczącego
TPShow	wybiera okno na panelu uczącym z RAPID'a

7.3 Czytanie i zapis z urządzenia znakowego

instrukcja	Zastosowanie
Open	otwiera port/plik do odczytu lub zapisu
Write	pisze tekst do portu/pliku
Close	zamyka port/plik

Funkcja	zastosowanie
ReadNum	czyta wartośc numeryczną
ReadStr	czyta ciąg znaków

7.4 Komunikowanie w trybie binarnym

instrukcja	zastosowanie
Open	otwiera port/plik do odczytu lub zapisu
WriteBin	pisze tekst do portu/pliku binarnego
WriteStrBin	pisze ciąg do portu/pliku binarnego
Rewind	ustawia pozycję pliku na jego początku
Close	zamyka port/plik

Funkcja	zastosowanie
ReadBin	czyta z portu szeregowego w trybie binarnym

7.5 Dane dla portów szeregowych

typ danych	definiuje
iodev	odniesienie do portu szeregowego, który może być używany do odczytu lub zapisu

11 Matematyka

Instrukcje i funkcje matematyczne mogą być stosowane do obliczania i zmiany wartości danych.

11.1 Podstawy

Obliczenie są zazwyczaj wykonywane przy użyciu instrukcji przyrównania, np. reg1:=reg2+reg3/reg5. Istnieją też instrukcje do prostych działań – np. do zerowania wartości zmiennej.

11.2 Proste operacje na danych numerycznych

Instrukcja	użycie
Clear	czyści wartość
Add	dodanie lub odjęcie wartości
Incr	zwiększenie wartości o 1
Decr	zmniejszenie wartości o 1

11.3 Bardziej zaawansowane obliczenia

Instrukcja	użycie
:=	wykonaj obliczeń na dowolnym typie danych

11.4 Funkcje arytmetyczne

Funkcja	zastosowanie
Abs	oblicza wartość bezwzględną
Round	zaokrągla wartość numeryczną
Trunc	obcina wartość po przecinku
Sqrt	oblicza pierwiastek kwadratowy
Exp	oblicza eksponentę (potęga o podstawie ”e”)
Pow	oblicza potęgę (dowolna podstawa)
Acos	oblicza arc cos
ASin	oblicza wartość arc sin
ATan	oblicza arc tangens w zakresie [-90,90]
ATan2	oblicza arc tangens w zakresie [-180,180]
Cos	oblicza cosinus
Sin	oblicza sinus
Tan	oblicza tangens
EulerZYX	oblicza kąty Eulera z orientacji (kwaternionów)
OrientZYX	oblicza orientację (kwaterniony) z kątów Eulera
PoseInv	odwraca *pose*
PoseMult	mnoży razy *pose*
PoseVect	iloczyn *pose* i wektora
Vectmagn	długość wektora pozycji.

DotProd	oblicza iloczyn dwóch wektorów pozycji
NOrient	normalizuje nieznormalizowaną orientację (kwaternion)

Uwaga*: pose* – typ danych umożliwiający przekształcanie jednego układu współrzędnych w drugi. Składa się z wektora translacji i rotacji.

15 Komunikacja z komputerem zewnętrznym

Robot może być kontrolowany z zewnętrznego komputera. W tym celu jest używany specjalny protokół komunikacyjny.

15.1 Podstawy programowania

Gdy jest używany wspólny protokół komunikacyjny do przesyłania danych z robota do komputera i z powrotem, urządzenia te rozumieją się nawzajem i nie ma potrzeby tego oprogramowywać. Komputer na przykład zmieniać wartości w danych programu bez programowania (oprócz definiowania danych). Programowanie jest konieczne tylko kiedy informacja z programu musi zostać wysłana z robota do zewnętrznego komputera.

15.2 Wysyłanie wiadomości programowej z robota do komputera

Instrukcja	zastosowanie
SCWrite	wysyła wiadomość do zewnętrznego komputera

17 Operacje na ciągach znaków

Funkcje realizujące operacje na ciągach znaków są stosowane m.in. do : kopiowania, łączenia, porównywania, wyszukiwania, zamiany ciągów

17.1 Podstawowe operacje

Typa danych	używany do definiowania
string	ciąg. Predefiniowane stałe STR_DIGIT, STR_UPPER, STR_LOWER i STR_WHITE

Instrukcja/operacja	używana do
:=	nadanie wartości (kopiowanie ciągu)
+	łączenie ciągów

Funkcja	Używana do
StrLen	podaje długość ciągu
StrPart	pobiera część ciągu

17.2 Porównywanie i przeszukiwanie

Operator	używany do
=	sprawdza, czy jest równe
<>	sprawdza, czy jest nierówne

Funkcja	używana do
StrMemb	sprawdza, czy znak należy do zestawu
StrFind	szuka znaku w ciągu
StrMatch	szuka zadanego wzorca w ciągu
StrOrder	sprawdza czy ciągi są po kolei

17.3 Zamiana

Funkcja	używana do
NumToStr	zamiana wartości numerycznej na ciąg
ValToStr	zamiana wartości na ciąg
StrToVal	zamiana ciągu na wartość
StrMap	przekształca ciąg zgodnie ze wzorcem
StrToByte	zamienia ciąg na bajt
ByteToStr	zamienia bajt na ciąg

18. Wielozadaniowość (multitasking)

Wielozadaniowość RAPID-a jest sposobem na uruchamianie programów (pseudo) równocześnie. Jeden program równoległy może być umieszczony w tle lub na pierwszym planie (in foreground) drugiego. To może też być na tym samym poziomie co inny program (patrz Podstawowe właściwości wielozadaniowości)

18.1 Podstawy

Aby używać tej funkcji, robot musi być odpowiednio skonfigurowany – jedno dodatkowe zadanie dla każdego programu działającego w tle. Do 10-ciu programów może zostać uruchomionych pseudorównolegle Każde zadanie składa się z zestawu modułów, takich samych jak zwykły program. Wszystkie moduły są lokalne w każdym z zadań. Zmienne i stałe są lokalne w każdym zadaniu, ale „trwałe" nie. „Trwałe" o tej samej nazwie i typie są osiągalne we wszystkich zadaniach. Jeśli dwie „trwałe" mają tą samą nazwę, ale różnią się typen lub rozmiarem (wymiar tablicy) wystąpi błąd. A task has its own trap handling and the event routines are triggered only on its own task system states (e.g. Start/Stop/Restart....).
Zadanie ma własną obsługę pułapek i procedur wywoływaną tylko na własnych stanach systemowych.

Ochrona dostępu do zasobów

Funkcja	zastosowanie
TestAndSet	odzyskuje wyłączne prawo do określonych obszarów kodu RAPID-a lub zasobów systemowych