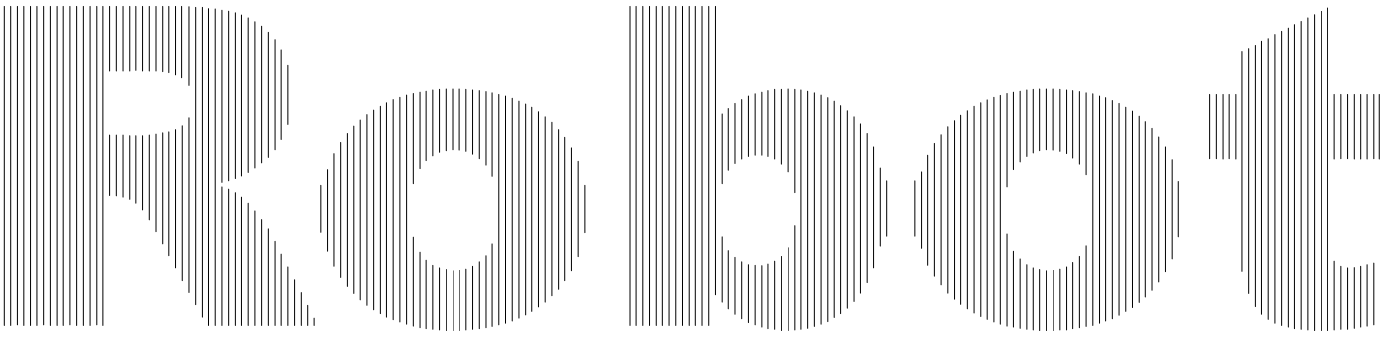
Simple and friendly

Kawasaki

Kontroler serii D

# Programowanie w języku AS




Kawasaki Heavy Industries, Ltd.

90209 - 1017DED - PL

## PRZEDMOWA

W niniejszej instrukcji opisano zagadnienia związane z językiem AS* używanym w Kontrolerach Robotów Serii D firmy Kawasaki. Celem instrukcji jest dostarczenie szczegółowych informacji w zakresie systemu AS, podstawowych zastosowań, rodzajów danych, kontroli trajektorii robota oraz wszystkich poleceń/instrukcji umożliwiających użytkowanie systemu AS. Procedury obsługi robota nie zostały tutaj uwzględnione, informacje w tym zakresie dostępne są w Instrukcji obsługi. Instrukcja niniejsza powinna być czytana po zaznajomieniu się z wymienionymi poniżej, powiązanymi instrukcjami. Użytkowanie robotów możliwe jest po zapoznaniu się z treścią instrukcji i po jej zrozumieniu.

1. Instrukcja bezpieczeństwa
2. Instrukcja instalacji i podłączenia robota
3. Instrukcja instalacji i podłączenia kontrolera
4. Instrukcja dotycząca zewnętrznych sygnałów we/wy (dla połączeń w urządzeniami zewnętrznymi)
5. Instrukcja kontroli i utrzymania

Treść niniejszej instrukcji została przedstawiona z założeniem, że instalacja i podłączenie robota są wykonywane zgodnie z wyżej wymienionymi instrukcjami.

Wyjaśnienia zawarte w instrukcji zawierają informacje o funkcjach dodatkowych, ale w zależności od specyfikacji określonego urządzenia, nie każda przedstawiona opcja może być funkcją robota. W razie jakichkolwiek niewyjaśnionych pytań lub problemów powstałych w trakcie obsługi robota, gorąco zachęcamy do kontaktu z firmą Kawasaki Machine Systems. Dane kontaktowe najbliższych biur firmy Kawasaki Machine Systems, patrz lista zamieszczona z tyłu okładki niniejszej instrukcji.

Uwaga* AS wymawiamy [ez].

---

1. Niniejsza instrukcja nie stanowi gwarancji systemów, w których wykorzystywany jest robot. Odpowiednio, firma Kawasaki nie ponosi odpowiedzialności za żadne wypadki, straty i/lub problemy związane z prawami własności przemysłowej, wynikłe z użytkowania przedmiotowych systemów.
2. Zaleca się szkolenie, prowadzone przez firmę Kawasaki, dla wszystkich pracowników przydzielonych do uruchomienia, uczenia, utrzymania lub kontroli robotów, przed podjęciem ich obowiązków.

3. Firma Kawasaki zastrzega sobie prawo zmiany, poprawiania lub uaktualniania niniejszej instrukcji bez wcześniejszego powiadomienia.
4. Niniejsza instrukcja nie może być, w całości, ani w części, przedrukowywana lub kopiowana bez wcześniejszej, pisemnej zgody firmy Kawasaki.
5. Instrukcja powinna być przechowywana w miejscu dostępnym o każdej porze. Jeśli robot jest reinstalowany, przenoszony na inne miejsce lub sprzedawany innemu użytkownikowi, instrukcja powinna wędrować wraz z robotem. W przypadku zgubienia lub zniszczenia instrukcji, należy skontaktować się z firmą Kawasaki.



## SYMBOLE

Pozycje wymagające w niniejszej instrukcji szczególnej uwagi są oznaczone następującymi symbolami.

Aby zagwarantować prawidłową i bezpieczną obsługę robotów oraz uniknąć urazów fizycznych I szkód materialnych, stosuj zasady bezpieczeństwa podane w oknach opatrzonych tymi symbolami.

**⚠ NIEBEZPIECZEŃSTWO**

**Niezastosowanie się do wskazanych kwestii może skutkować nieuchronnym urazem lub śmiercią.**

**⚠ OSTRZEŻENIE**

**Niezastosowanie się do wskazanych kwestii może skutkować ewentualnym urazem lub śmiercią.**

**⚠ UWAGA**

**Niezastosowanie się do wskazanych kwestii może skutkować ewentualnym urazem i/lub śmiercią.**

[**UWAGA**]

Wskazuje środki ostrożności w zakresie specyfikacji robota,
przenoszenia, uczenia, obsługi i utrzymania.

**OSTRZEŻENIE**

1. **Dokładność i efektywność wykresów, procedur oraz wyjaśnionych szczegółów, zawartych w niniejszej instrukcji nie może być potwierdzona z całą pewnością. W razie jakichkolwiek niewyjaśnionych pytań lub problemów, gorąco zachęcamy do kontaktu z firmą Kawasaki Machine Systems.**
2. **Treść dotycząca bezpieczeństwa opisana w niniejszym podręczniku znajduje zastosowanie do każdej wykonywanej pracy, nie tylko do tej związanej z opisywanym robotem. W celu bezpiecznego wykonywania pracy, zapoznaj się z niniejszym podręcznikiem, wszelkie związane z tematem zasady, regulacje prawne i istotne materiały, a także opisane w każdym rozdziale zasady bezpieczeństwa, a w końcu, podejmij niezbędne kroki do zapewnienia bezpieczeństwa każdej konkretnej pracy.**

# SPIS TREŚCI

# 1.0 INFORMACJE OGÓLNE DOTYCZĄCE SYSTEMU AS

Roboty firmy Kawasaki są sterowane za pomocą oprogramowania opartego na systemie noszącym nazwę AS. W niniejszym rozdziale przedstawiono ogólne informacje dotyczące systemu AS.

## 1.1 IINFORMACJE OGÓLNE DOTYCZĄCE SYSTEMU AS

W systemie AS możesz komunikować się z robotem lub tworzyć programy za pomocą języka AS. System AS jest zapisany w pamięci trwałej modułu sterującego robota. Po włączeniu zasilania, system AS uruchamia się i oczekuje na wprowadzenie poleceń.

System AS steruje robotem zgodnie z zadanymi poleceniami i programami. W czasie działania programu można także wykonywać kilka rodzajów funkcji. Oto niektóre z funkcji, które mogą być wykorzystywane w czasie działania programu: wyświetlanie statusu systemu lub ustawienia robota (położenia i pozycji), zapisywanie danych w zewnętrznej pamięci oraz pisanie/edycja programów.

## 1.2 CHARAKTERYSTYKA SYSTEMU AS

W systemie AS robot jest sterowany i obsługiwany w oparciu o program przygotowany przed rozpoczęciem operacji, opisujący zadania niezbędne dla tej operacji. (Teaching Playback Method - metoda uczenia odtwarzania.)

Język AS może być podzielony na dwa typy: polecenia wprowadzane z ekranu (monitor commands) oraz instrukcje programu (program instructions).

Polecenia wprowadzone z ekranu: wykorzystywane do pisania, edycji i wykonywania programów. Są one wprowadzane po wyświetleniu na ekranie znaku zachęty (>) i natychmiast wykonywane. Niektóre polecenia wprowadzane z ekranu są wykorzystywane wewnątrz programu jako instrukcje programu.

Instrukcje programu: wykorzystywane do kierowania w programach ruchami robota, nadzoru i kontroli zewnętrznych sygnałów, itp. Program jest zbiorem instrukcji programu.

W niniejszym podręczniku, polecenia wprowadzone z ekranu są określane poleceniami a instrukcje programu instrukcjami.

System AS jest wyjątkowy ze względu na następujące cechy:

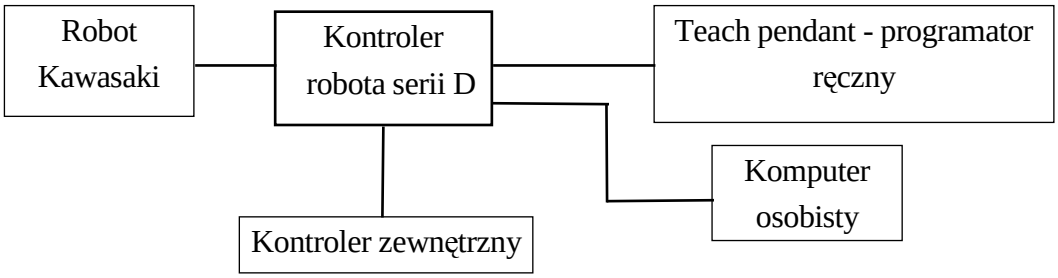
1. Robot może być przesuwany wzdłuż ciągłych trajektorii ścieżki (CP motion: Continuous Path motion - ciągła ścieżka ruchu).
2. Istnieją dwa układy współrzędnych, współrzędne globalne i układ współrzędnych narzędzia, umożliwiające bardziej precyzyjne sterowanie ruchami robota.
3. Współrzędne mogą być przesuwane lub obracane, odpowiednio do zmian ustawienia przedmiotu obrabianego.
4. Podczas uczenia lokalizacji lub wykonywania odtwarzania, robot może być przesuwany wzdłuż ścieżki liniowej z jednoczesnym zachowaniem pozycji narzędzia.
5. Programy mogą być nazywane w dowolny sposób i zapisywane w dowolnych ilościach.
6. Każda jednostka operacyjna może być określona jako program; programy te mogą być łączone w celu utworzenia jednego, złożonego programu. (Subroutine - podprogram standardowy.)
7. Poprzez kontrolę sygnałów, programy mogą być przerywane i rozgałęziane do innych programów, powodując zawieszenie bieżących ruchów po wejściu sygnału zewnętrznego. (Interruption - przerwanie)
8. Program sterujący procesem (PC program - program sterujący procesem działającym

równolegle) może być wykonywany jednocześnie z programem sterującym robotem.

9. Programy i dane ustawienia (lokalizacja) mogą być wyświetlane na terminalach różnych urządzeń, np. kart PC.
10. Programowanie może być przeprowadzane z wykorzystaniem komputerów osobistych, na których załadowano oprogramowanie dostarczone przez firmę Kawasaki (KRterm lub KCwin32). ( Off-line programming - programowanie w trybie offline )

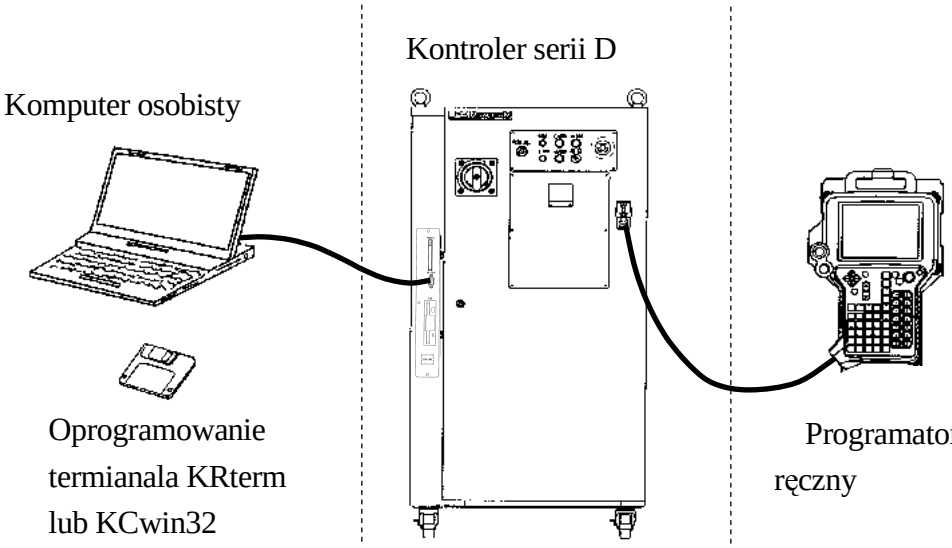
## 1.3 KONFIGURACJA SYSTEMU AS

Kontroler robota serii D firmy Kawasaki składa się z następujących elementów:




Połączenie komputera osobistego, na którym znajduje się oprogramowanie terminala dostarczone przez firmę Kawasaki KRterm lub KCwin32 z kontrolerem serii D, umożliwia wykonanie poniższych operacji:

- Zapisywanie poleceń języka AS
- Zapisywanie i ładowanie na i z komputerów osobistych.




| Komputer osobisty | robota serii D | Programator ręczny |
|---|---|---|
| · Wprowadza polecenia języka AS<br>· Tworzy programy AS<br>· Zapisuje/ładuje programy | Codzienne operacje | · Wybiera program<br>· Wyświetla nazwy programów i kroków<br>· Manualnie steruje robotem<br>· Kontroluje sygnały<br>· Ustawia warunki odtwarzania<br>· Uczy danych ustawienia (lokalizacji)<br>· Uczy danych pomocniczych |

**[UWAGA]**

Oprogramowanie monitorujące dla PC współdziała z 95/98/Me/2000/XP. Przygotuj odpowiedni system operacyjny.

## 2.0 SYSTEM AS

W niniejszym rozdziale przedstawiono status system AS, parametry systemowe oraz konfigurację systemu AS.

## 2.1 STATUS SYSTEMU AS

System AS posiada następujące trzy tryby:

1. Monitor mode - tryb monitorowania
   Jest to podstawowy tryb systemu AS. W trybie tym wykonywane są polecenia wprowadzane z ekranu. Z tego trybu możliwy jest dostęp do trybu edycji (Editor) i odtwarzania (Playback Mode).

2. Tryb uczenia
   Tryb ten umożliwia tworzenie nowych programów lub modyfikowanie istniejących. W trybie tym system wykonuje wyłącznie polecenia edycji.

3. Playback mode - tryb odtwarzania
   System pracuje w trybie odtwarzania podczas wykonywania programu. Gdy CPU nie jest zajęty, wykonywane są obliczenia dotyczące sterowania ruchem i przetwarzane są polecenia wprowadzane z terminala. W trybie tym niektóre polecenia wprowadzane z ekranu nie mogą być wykonywane.

## 2.2 PARAMETRY SYSTEMU AS

Przy pomocy polecenia wprowadzanego z ekranu SWITCH możliwe jest konfigurowanie następujących parametrów systemowych. Status oraz warunki skonfigurowane dla każdego z parametrów mogą być sprawdzane lub zmieniane z terminala.

1. CHECK.HOLD
   Określa, czy odpowiadać na polecenia EXECUTE, DO, STEP, MSTEP i CONTINUE, gdy przełącznik HOLD/RUN jest ustawiony w pozycji HOLD.

2. CP
   Włącza lub wyłącza ciągłą ścieżkę ruchu. Gdy znajduje się w pozycji ON, robot wykonuje płynne przejścia pomiędzy odcinkami ruchu. Gdy wyłącznik jest w pozycji OFF, robot zwalnia i zatrzymuje się na końcu każdego odcinka ruchu.

3. CYCLE.STOP
   Określa, czy CYCLE START ma się znajdować w pozycji ON czy OFF, gdy wprowadzany jest zewnętrzny sygnał wstrzymania w celu zatrzymania ruchu robota.

4. MESSAGES
   Włącza lub wyłącza wychodzenie wiadomości do terminala w odpowiedzi na polecenia PRINT lub TYPE.

5. OX.PREOUT
   Ustawia taktowanie sygnałów wyjścia OX w instrukcjach blokowych, umożliwiając wcześniejsze wyjście sygnału podczas zmiany pamięci, zamiast podczas konfiguracji dokładności.

6. PREFETCH.SIGINS
   Konfiguruje taktowanie sygnałów wyjścia programów języka AS i wywiera taki sam wpływ na taktowanie sygnałów, co OX.PREOUT.

7. QTOOL
   W trybie uczenia (TEACH) określa, czy dokonać zmiany przekształcenia narzędzia zgodnie z numerem narzędzia wyuczonym w instrukcji blokowej.

8. REP_ONCE (Repeat Once - odtwórz raz)
   Jeśli przełącznik ten znajduje się w pozycji ON, program jest wykonywany raz. Jeśli przełącznik ten znajduje się w pozycji OFF, program jest wykonywany w sposób ciągły.

9. RPS (Remote Program Selection - zdalny wybór programu)
   Włącza lub wyłącza wybór programów na podstawie statusu binarnego sygnałów zewnętrznych.

10. SCREEN
    Włącza lub wyłącza przewijanie ekranu, gdy informacja jest zbyt duża, aby zmieścić się na jednym ekranie.

11. STP_ONCE
    Konfiguruje program w taki sposób, że na raz wykonywany jest jeden krok lub też, że następuje wykonywanie ciągłe.

Dalsze informacje dotyczące konfiguracji parametrów systemowych, patrz 5.6 Polecenia wprowadzane z ekranu SWITCH, ON, OFF.

## 3.0 WYRAŻENIA W JĘZYKU AS

W niniejszym rozdziale przedstawiono rodzaje informacji i zmienne używane w języku AS.

## 3.1 ZAPIS I KONWENCJE

1. Wielkie i małe litery
   Dla lepszego zrozumienia, poniższe zasady dotyczą stosowania w niniejszym podręczniku wielkich i małych liter. Wszystkie słowa kluczowe systemu AS (polecenia, instrukcje, itp.) są pisane dużymi literami. Zmienne i inne pozycje są pisane małymi literami. Jednakże, podczas wprowadzania na terminalu AS mogą być używane zarówno duże, jak i małe litery.

2. Klawisze i przełączniki
   Nazwy klawiszy na programatorze ręcznym lub na klawiaturze komputera oraz przełączników na kontrolerze są otaczane w niniejszej instrukcji przez □.

   **Przykład** RUN/HOLD, Backspace

3. Skróty
   słowa kluczowe mogą być skracane. Np. polecenie EXECUTE może być skracane jako EX. Patrz załącznik 2 Lista języka AS.

4. Space, Tab
   Co najmniej jedno puste miejsce lub tabulacja są wymagane jako ogranicznik pomiędzy poleceniem (lub instrukcją) a parametrem*. Spacje lub tabulacja są także wymagane pomiędzy tymi parametrami, które nie są oddzielone przecinkami lub innymi ogranicznikami. Nadmiarowe spacje lub tabulacje są ignorowane przez system.

   Uwaga* Parametr stanowi niezbędne dane uzupełniające polecenie lub inną funkcję. Np. w przypadku polecenia SPEED, parametr danych jest potrzebny do określenia prędkości robota. Jeśli polecenie lub funkcja używa kilku parametrów, przecinek lub spacja oddziela każdy z nich.
   **Przykład** SPEED 50

5. Klawisz ENTER
   Polecenia wprowadzane z ekranu i instrukcje programu są przetwarzane poprzez naciśnięcie klawisza ENTER. W niniejszej instrukcji klawisz ENTER jest przedstawiany jako ↵.

6. Pomijane parametry
   Wiele poleceń wprowadzanych z ekranu i instrukcji programu posiada parametry, które mogą być pominięte. Jeśli pomiędzy tymi opcjonalnymi parametrami znajduje się przecinek, powinien on być zachowany, nawet jeśli parametr jest pomijany. Jeśli pomijane są wszystkie kolejne parametry, przecinek również może być pominięty.

7. Wartości liczbowe

Wartości są wyrażane w zapisie dziesiętnym, chyba że zastrzeżono inaczej. Wyrażenia matematyczne mogą być używane do określania tych wartości jako argumentów. Jednakże zauważ, iż akceptowane wartości są ograniczone. Poniższe zasady wyjaśniają sposób, w jaki interpretowane są wartości w różnych przypadkach.

( 1 ) Odległość

Używana do określania długości ruchu robota pomiędzy dwoma punktami Jednostką odległości jest milimetr (mm); podczas wprowadzania, jednostka jest pomijana. Wprowadzane wartości mogą być albo ujemne, albo dodatnie.

( 2 ) Kąty

Określa i modyfikuje pozycję robota w określonym ustawieniu (lokalizacji) i opisuje wartość obrotu osi robota. Wartości mogą być ujemne lub dodatnie, z maksymalnym kątem pomiędzy 180 a 360 stopni, w zależności od używanych poleceń.

( 3 ) Zmienne skalarne

Zmienne te określają rzeczywiste wartości, chyba że zastrzeżono inaczej. Wartości dla zmiennym mogą się mieścić w zakresie od –3.4E+38 do 3.4E+38 ($-3.4\times10^{38}$ do $3.4+10^{38}$). Jeśli przekraczają ±999999, są wyrażane jako xE+y (x jest mantysą, a y jest wykładnikiem).

( 4 ) Numer osi

Reprezentuje osie robota w liczbach całkowitych od 1 do osiągalnej liczby osi (standardowy typ posiada 6 osi). Osie są numerowane poczynając od base joint - osi podstawowej. (Zwykle wyrażane jako JT1, JT2 ....).

( 5 ) Numer sygnału

Identyfikuje sygnały binarne (ON/OFF). Wartości są wyrażane w liczbach całkowitych i mieszczą się w poniższym zakresie.

| | Zakres standardowy | Zakres maksymalny |
|---|---|---|
| Zewnętrzny sygnał wyjścia | 1 – 32 | 1 – 96 |
| Zewnętrzny sygnał wejścia | 1001 – 1032 | 1001 – 1096 |
| Sygnał wewnętrzny | 2001 – 2256 | 2001 – 2256 |

Ujemna wartość numeru sygnału wskazuje na wyłączenie (OFF).

8. Słowa kluczowe
   Zasadniczo, nazwy zmiennych w ramach systemu AS mogą być przydzielane swobodnie. Jednakże, słowa kluczowe określające polecenia, instrukcje, itp. w systemie AS są zarezerwowane i nie mogą być używane jako nazwy danych ustawienia, zmiennych, itp.

## 3.2 INFORMACJE O POZYCJI ROBOTA, INFORMACJE LICZBOWE, INFORMACJE ZAWIERAJĄCE ZNAKI

W systemie AS istnieją trzy rodzaje informacji: informacje o ustawieniu*, informacje liczbowe i informacje zawierające znaki.

UWAGA* "Ustawienie" nazywano dawniej "lokalizacją", jednak zgodnie z międzynarodowymi standardami (ISO), w niniejszym podręczniku używa się określenia ustawienie, które oznacza łącznie położenie oraz pozycję robota.

### 3.2.1 INFORMACJE O POZYCJI ROBOTA

Informacja o ustawieniu, znana także jako informacja o pozycji jest wykorzystywana do określania położenia i pozycji robota w określonym obszarze pracy. Położenie i pozycja robota odnoszą się do położenia i pozycji punktu centralnego narzędzia (TCP) robota. Położenie i pozycja razem są nazywane ustawieniem (lokalizacją) robota.

Ustawienie jest określane poprzez miejsce, w którym znajduje się robot i kierunek, w którym jest ustawiony; dlatego gdy robot otrzymuje instrukcję ruchu, obie te rzeczy są wykonywane jednocześnie:

1. Punkt centralny narzędzia robota przesuwa się do określonego położenia.
2. Układ współrzędnych narzędzia robota obraca się do określonej pozycji.

Dane ustawienia są określone poprzez zestaw wartości przesunięcia osi lub wartości przekształcenia:

1. Joint displacement values - wartości przesunięcia osi
   Informacja o ustawieniu jest określana poprzez kątowe lub liniowe wartości przesunięcia każdej z osi robota. Korzystając z wartości enkodera, osie obrotowe uzyskują kątowe przesunięcie wyrażane w stopniach, a osie liniowe uzyskują przesunięcie wyrażane w milimetrach.

   **Przykład** Osie są określane w kolejności od JT1,…JT6, a wartości przesunięcia każdej z nich są prezentowane poniżej numeru osi.

```
   T1    JT2       JT3       JT4    JT5    JT6
   #pose = 0.00,  33.00,   -15.00,    0,    -40,    30
```

2. Transformation value - wartość przekształcenia

Opisuje ustawienie współrzędnych w relacji do współrzędnych odniesienia. Jeśli nie określono inaczej, odnosi się do wartości przekształcenia układu współrzędnych narzędzia względem współrzędnych globalnych robota. Położenie jest określane wartościami XYZ na współrzędnych globalnych, pozycja kątami Eulera (O, A, T = Orientation, Azimuth, Tool)*. Do najczęściej używanych wartości przekształcenia należą: wartości przekształcenia narzędzia, określające ustawienia współrzędnych narzędzia względem współrzędnych wyjściowych narzędzia oraz wartości przekształcenia względem przedmiotu obrabianego, określające ustawienie współrzędnych narzędzia względem współrzędnych przedmiotu obrabianego.
Uwaga* Patrz załącznik 5 Kąty Eulera (O, A, T = Orientation, Azimuth, Tool).

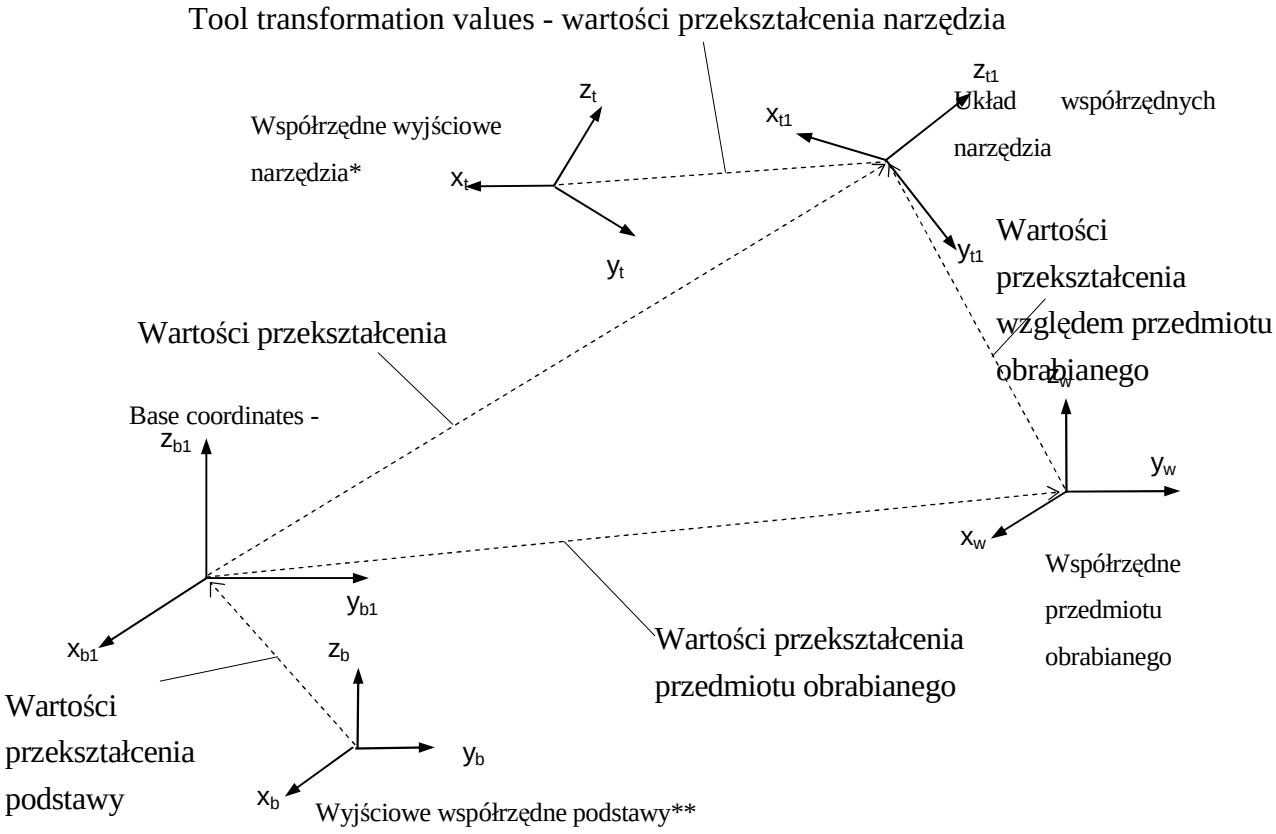
**Przykład**

```
                    X     Y       Z     O     A     T
pose (ustawienie) =    0,  1434,  300,    0,    0,    0
```

Jeśli robot posiada więcej niż sześć osi, wartości dodatkowych osi są prezentowane wraz z wartościami przekształcenia.

**Przykład**

```
                    X     Y       Z     O     A     T    JT7
pose (ustawienie) =    0,  1434,  300,    0,    0,    0   1000
```




Uwaga * Współrzędne wyjściowe narzędzia posiadają punkt początkowy położony w środku

powierzchni kołnierza mocującego narzędzie i są określane wartościami przekształcenia narzędzia (0,0,0,0,0,0).

| | Wartości przesunięcia osi | Wartości przekształcenia |
|---|---|---|
| Zalety | · Precyzja odtwarzania oraz jednoznaczna konfiguracja robota w określonym ustawieniu | · Punkt początkowy układu współrzędnych narzędzia wykorzystywany w trybie odtwarzania nie zmienia się nawet po zmianie narzędzia. (przesunięcie współrzędnych wyjściowych narzędzia)<br>· Możliwość użycia współrzędnych względnych (np. współrzędnych przedmiotu obrabianego).<br>· Wygoda przy przetwarzaniu, jako że dane są prezentowane w wartościach XYZOAT. |
| Wady | · Punkt centralny narzędzia (TCP) zmienia się (współrzędne wyjściowe narzędzia pozostają takie same).<br>· Brak możliwości użycia współrzędnych względnych (np. współrzędnych przedmiotu obrabianego). | · Współrzędne będą się zmieniały zgodnie z wartościami przekształcenia podstawy lub narzędzia, więc dla zachowania bezpieczeństwa niezbędne jest pełne rozeznanie, co do skutków każdej zmiany.<br>· Konfiguracja robota może się zmieniać, jeśli nie została ustawiona przed rozpoczęciem odtwarzania. |
| Proponowane wykorzystanie | · Konfiguracja ustawienia początkowego programu<br>· Ustawianie konfiguracji robota w lub tuż przed ustawieniem określonym wartościami przekształcenia<br>· Inne powszechne ustawienia | · Opis współrzędnych względnych, jak np. współrzędnych przedmiotu obrabianego<br>· Opis ustawienia, które ma być zmienione przy użyciu wartości liczbowych z funkcją SHIFT lub podobną.<br>· Opis ustawienia, które ma być zmienione przez informacje pochodzące z czujnika. |

Uwaga** Współrzędne wyjściowe podstawy są konfigurowane jako wartości domyślne robota i są określane wartościami przekształcenia podstawy (0,0,0,0,0,0).

Wartości przesunięcia osi oraz wartości przekształcenia posiadają swoje zalety i wady. Wykorzystaj je zgodnie ze swoimi potrzebami.

[**UWAGA**]

1. W przeciwieństwie do sytuacji, gdy ustawienie jest definiowane poprzez wartości przesunięcia osi, gdzie konfiguracja robota jest niepowtarzalna, robot może przyjmować różne konfiguracje w zakresie tego ustawienia. Dzieje się tak ze względu na fakt, iż wartości przekształcenia określają wyłącznie wartości XYZOAT układu współrzędnych narzędzia robota, a nie definiują wartości każdej z osi. Z tego względu, przed uruchomieniem robota w trybie odtwarzania, upewnij się, że konfiguracja robota została ustalona przy pomocy poleceń konfiguracji (LEFTY, itp.) lub poprzez zapisanie wartości przesunięcia osi.

2. Ponieważ wartości przekształcenia są określane przez współrzędne globalne, jeśli współrzędne globalne są zmieniane przy pomocy polecenia/instrukcji BASE, punkt centralny narzędzia (TCP) robota zmienia się o tą samą wartość. Stanowi to jedną z zalet wykorzystywania wartości przekształcenia, należy jednak pamiętać o wpływie, jaki zmiana współrzędnych globalnych będzie miała na przekształcane punkty. Uchybienie w tym zakresie może spowodować kolizję z urządzeniami zewnętrznymi.

   Zachowaj także ostrożność podczas korzystania z polecenia/instrukcji TOOL.

### 3.2.2 INFORMACJE LICZBOWE

W systemie AS, wartości liczbowe i wyrażenia mogą być wykorzystywane jako informacje liczbowe. Wyrażenie liczbowe jest wartością wyrażoną w cyfrach i zmiennych połączonych z operatorami i funkcjami. Wyrażenia liczbowe są wykorzystywane nie tylko do obliczeń matematycznych, ale także jako parametry monitoringu poleceń i instrukcji programu.

Np. polecenie DRIVE określa trzy parametry: numer osi, stopień i prędkość. Parametry mogą być wyrażane w wartościach liczbowych albo też w wyrażeniach, jak w poniższych przykładach:

DRIVE 3,45,75 Przesuwa oś numer 3 o 45° z prędkością 75%.

DRIVE joint, (start+30)/2, 75 Jeśli określono joint=2, start=30, joint 2 (oś 2) jest przemieszczana +30° z prędkością 75%.

Wartości liczbowe wykorzystywane w systemie AS można podzielić na trzy rodzaje:

1. Liczby rzeczywiste

   Liczby rzeczywiste mogą zawierać zarówno liczby całkowite, jak i ułamki. Mogą to być wartości dodatnie oraz ujemne pomiędzy –3.4 E+38, a 3.4 E+38($-3.4\times10^{38}$, a $3.4\times10^{38}$) lub zero. Liczby rzeczywiste mogą być prezentowane w notacji naukowej. Symbol E wyróżnia mantysę od wykładnika. Wykładnik może być liczbą ujemną (potęga 1/10) lub dodatnią

(potęga 10).

| | | | |
|---|---|---|---|
| **Przykład** | 8.5E3 | $8.5\times10^{3}$ | (pominięto + w wykładniku) |
| | 6.64 | $6.64\times10^{0}$ | (pominięto E, 0) |
| | –9E-5 | $-9.0\times10^{-5}$ | (pominięto przecinek dziesiętny) |
| | –377 | $-377\times10^{0}$ | (pominięto przecinek dziesiętny, E, 0) |

Pamiętaj, że ważnych jest pierwszych siedem cyfr, jednak liczba ważnych cyfr może się zmniejszać w procesie obliczeń.

Wartości rzeczywiste bez ułamków są nazywane liczbami całkowitymi. Zakres waha się w od–16 777 216 do +16 777 215, a w przypadku wartości przekraczających ten limit, ważnych jest pierwszych siedem cyfr. Liczby całkowite są zazwyczaj wprowadzane w notacji dziesiętnej, chociaż czasami wygodnie jest się posługiwać notacją binarną lub szesnastkową. ^B oznacza, że liczba została wprowadzona w notacji binarnej. ^H oznacza, że liczba została wprowadzona w notacji szesnastkowej.

| | | |
|---|---|---|
| **Przykład** | ^B101 | (5 w notacji dziesiętnej) |
| | ^HC1 | (193 w notacji dziesiętnej) |
| | –^B1000 | (–8 w notacji dziesiętnej) |
| | –^H1000 | (–4096 w notacji dziesiętnej) |

2. Wartości logiczne

Wartości logiczne posiadają tylko dwie opcje - ON (włączony) i OFF (wyłączony) lub TRUE (prawda) i FALSE (fałsz). Wartość –1.0 jest przypisana do opcji TRUE lub ON, a wartość 0 (lub 0.0) jest przypisana do opcji FALSE lub OFF. ON, OFF, TRUE i FALSE są zarezerwowane dla języka AS.

Logiczna prawda = TRUE, ON, –1.0
Logiczny fałsz = FALSE, OFF, 0,0

3. Wartości kodu ASCII

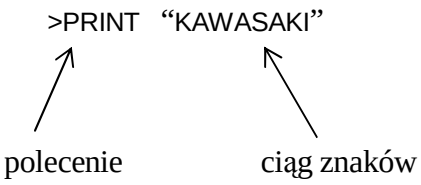
Przedstawiają wartość liczbową jednego znaku kodu ASCII. Znak ten jest poprzedzany apostrofem (’), który odróżnia go od innych wartości.

’A ’1 ’v ’%

### 3.2.3 INFORMACJE ZAWIERAJĄCE ZNAKI

Informacja zawierająca znaki dotycząca systemu AS jest przedstawiana jako ciąg znaków kodu ASCII ujęty w cudzysłów (“”). Jako że znak cudzysłów wskazuje początek i koniec ciągu, nie może on być użyty jako część ciągu. Ponadto, znaki kontrolne kodu ASCII (CTRL, CR, LF, itp.) nie mogą być zawarte w ciągu.

**Przykład**



## 3.3 ZMIENNE

W systemie AS, nazwy mogą być przypisane informacjom o ustawieniu, informacjom liczbowym i informacjom zawierającym znaki. Nazwy te są nazywane zmiennymi, które można podzielić na dwa rodzaje: zmienne globalne i lokalne. Jeśli nie zastrzeżono inaczej, zmienne globalne są określane jako zmienne.

### 3.3.1 ZMIENNE (ZMIENNE GLOBALNE)

Zmienne dla informacji o ustawieniu, informacji liczbowej i informacji zawierającej znaki są nazywane odpowiednio zmiennymi ustawienia, zmiennymi rzeczywistymi*, zmiennymi ciągu. Kilka wartości można zgrupować i używając zmiennej tablicowej zdefiniować jedną nazwą.

Uwaga* Jako że większość wartości liczbowych używanych w systemie AS to liczby rzeczywiste, zmienne liczbowe są nazywane zmiennymi liczb rzeczywistych. Jednakże pamiętaj, iż liczby całkowite, wartości logiczne i wartości kodu ASCII są wyrażane przy pomocy liczb rzeczywistych. Z tego względu zmienne rzeczywiste mogą odnosić się do tych wartości.

Po zdefiniowaniu, zmienna jest zapisywana w pamięci wraz z określoną wartością. Dlatego też może być użyta w dowolnym programie.

### 3.3.2 ZMIENNE LOKALNE

W przeciwieństwie do zmiennych globalnych przedstawionych powyżej, zmienne lokalne są definiowane ponownie podczas każdego wykonania programu i nie są zapisywane w pamięci. Zmienna z “.” (kropką) na początku nazwy jest uważana za zmienną lokalną.

Zmienne lokalne są pomocne w przypadkach, gdy kilka programów używa tej samej nazwy zmiennej, której wartość zmienia się podczas każdego uruchomienia programu. Zmienne lokalne mogą być także używane jako parametry podprogramów standardowych. (Patrz także 4.4.2 Podprogramy standardowe z parametrami.)

**UWAGA**

1. Zmienna lokalna nie może być definiowana przy pomocy polecenia wprowadzanego z ekranu.

2. Ponieważ zmienne lokalne nie są zapisywane w pamięci, wartość zmiennej lokalnej .pose nie może być wyświetlona przy pomocy poniższego polecenia.

   ```
   >POINT. pose
   ```

   Aby wyświetlić bieżącą wartość zmiennej lokalnej, ustaw jej wartość na zmienną globalną, w miejscu gdzie zdefiniowana jest zmienna lokalna, a następnie użyj polecenia POINT.

   ```
   POINT a=.pose
   ```
   Wykonuje program definiujący zmienną lokalną przed użyciem polecenia POINT.

   ```
   >POINT a
   X[mm]     Y[mm]     Z[mm]     O[deg]     A[deg]     T[deg]
   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx    xxxxxxx    xxxxxxx
   Change? (Zmienić?)  ( Jeśli nie, naciśnij klawisz RETURN. ) ↵
   ```

## 3.4 NAZWY ZMIENNYCH

Nazwy zmiennych muszą zaczynać się od znaków alfabetycznych i mogą zawierać tylko litery, cyfry, kropki i znaki podkreślenia. Dopuszczalne są wielkie i małe litery (na ekranie wyświetlacza wyświetlane będą małe litery). Długość zmiennej jest ograniczona do piętnastu znaków. W przypadku nazw dłuższych, tylko piętnaście pierwszych znaków będzie ważnych. Poniżej zamieszczono kilka przykładów niedopuszczalnych nazw:

| | |
|---|---|
| 3p | pierwszy znak nie jest znakiem alfabetycznym |
| part#2 | ”#” jest pierwszym członem nazwy zmiennej przesunięcia i nie może być użyta w środku nazwy zmiennej |
| random | słowo kluczowe |

**UWAGA**

1. Zmienne opisujące wartości przesunięcia osi są poprzedzane symbolem “#”, który odróżnia je od wartości przekształcenia. Zmienne ciągu znaków (w języku AS) są poprzedzane symbolem “$”, który odróżnia je od zmiennych rzeczywistych.

| | | | |
|---|---|---|---|
| pick | (wartość przekształcenia) | #pick | (wartość przesunięcia osi) |
| count | (zmienna rzeczywista) | $count | (zmienna ciągu) |

2. Wszystkie zmienne mogą być używane jako zmienne tablicowe. Tablice składają się z kilku wartości noszących tę samą wartość; wartości te są odróżniane od siebie przy pomocy wartości indeksowej. Każda wartość w tablicy jest nazywana elementem tablicy. Aby określić element tablicy, dołącz wartość indeksową elementu otoczoną nawiasami. Np. “part [7]” wskazuje siódmy element tablicy “part”. W odniesieniu do indeksów używaj liczb całkowitych w zakresie od 0 do 9999. W przypadku tablic trójwymiarowych używaj składni podobnej do poniższej: part [7, 1,1]=1.

3. Po zdefiniowaniu zmienna ta może być używana w różnych programach. Z tego powodu zachowaj ostrożność, aby nie dokonać niepotrzebnych zmian w zmiennych, które są używane w różnych programach.

## 3.5 DEFINIOWANIE ZMIENNYCH POZYCJI

Zmienne opisujące informacje o ustawieniu są nazywane zmiennymi pozycji. Zmienna pozycji jest definiowana, jeśli jest jej przypisana wartość. Natomiast do momentu przypisania jej wartości, zmienna ta pozostaje niezdefiniowana, a jeśli niezdefiniowana zmienna jest wykorzystywana przez wykonywany program, występuje błąd.

Zmienne pozycji są pomocne gdyż:

1. Te same dane ustawienia mogą być wykorzystywane w sposób powtarzalny bez potrzeby przeprowadzania za każdym razem uczenia.
2. Określona zmienna ustawienia może być wykorzystywana w różnych programach.
3. Określona zmienna pozycji może być wykorzystywana lub zmieniana, w celu zdefiniowania różnych ustawień.
4. Obliczone wartości mogą być użyte jako informacja o ustawieniu, zamiast przeprowadzania czasochłonnego procesu uczenia robota pozycji przy pomocy programatora ręcznego.
5. Zmienne pozycji mogą być nazywane dowolnie, tak iż program może być tworzony w sposób bardziej czytelny.

Zmienne pozycji są definiowane w sposób przedstawiony poniżej.

### 3.5.1 DEFINIOWANIE PRZY POMOCY POLECEŃ WPROWADZANYCH Z EKRANU

1. Polecenie HERE zapisuje aktualne dane pozycji robota pod określoną nazwą.

**Przykład 1** Wykorzystywanie wartości przesunięcia osi

Poprzedź nazwę zmiennej symbolem #, aby odróżnić ją od wartości przekształcenia.
Po poleceniu pojawią się wartości przesunięcia osi bieżącego ustawienia:

```
> HERE   #pose
  JT1       JT2          JT3       JT4       JT5       JT6
  xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx
  Change? (Zmienić?) (Jeśli nie, naciśnij klawisz RETURN.)↵
>
```

**Przykład 2** Wykorzystywanie wartości przekształcenia

Po poleceniu pojawią się wartości przekształcenia bieżącego ustawienia:

```
>HERE pose
  X[mm]    Y[mm]    Z[mm]    O[deg]    A[deg]    T[deg]
xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx   xxxxxxx
  Change? (Zmienić?) (Jeśli nie, naciśnij klawisz RETURN.)
>
```

2. Polecenie POINT jest wykorzystywane do definiowania pozycji przy pomocy innej zdefiniowanej zmiennej pozycji lub w celu zdefiniowania go poprzez dane wprowadzane z terminala.

**Przykład 1** Wykorzystywanie wartości przesunięcia osi

(1) Definiowanie nowej, niezdefiniowanej zmiennej

```
>POINT   #pose
      JT1      JT2      JT3      JT4      JT5      JT6
      0.000    0.000    0.000    0.000    0.000    0.000
   Change? (Zmienić?) (Jeśli nie, naciśnij klawisz RETURN.)
     >
```

Wprowadź nowe wartości, oddzielając każdą z nich przecinkiem:

xxx, xxx, xxx, xxx, xxx, xxx

(2) Zmiana wartości zdefiniowanej zmiennej

```
>POINT    #pose
      JT1      JT2      JT3      JT4      JT5      JT6
      10.000   20.000   30.000   40.000   50.000   40.000
   Change? (Zmienić?) (Jeśli nie, naciśnij klawisz RETURN.)
```

Wprowadź wartość, którą chcesz zmienić:

30, , , ,20, ;zmienia wartość JT1 i JT 5 na 30 i 20

(3) Zastępowanie wartości zdefiniowanej zmiennej

```
>POINT    pose_1=pose_2
      JT1      JT2      JT3      JT4      JT5      JT6
      10.000   20.000   30.000   40.000   50.000   40.000
   Change? (Zmienić?) (Jeśli nie, naciśnij klawisz RETURN.)
```

Pojawia się wartość, która ma być zdefiniowana jako pose_1 (ostatnia wartość pose_2). Naciśnij ↵, aby ustawić wartości lub zmień je, stosując identyczną procedurę jak w punkcie (2) powyżej.

**Przykład 2** Wykorzystywanie wartości przekształcenia

Przeprowadź procedurę jak powyżej, z tym wyjątkiem, że nazwa zmiennej nie powinna zaczynać się symbolem #.

### 3.5.2 DEFINIOWANIE PRZY POMOCY INSTRUKCJI PROGRAMU

1. Instrukcja HERE zapisuje aktualne dane pozycji robota pod określoną nazwą.

```
HERE    pose
```

> [ UWAGA ]
> Jeśli używana jest nazwa zmiennej ustawienia zaczynająca się od #, zmienna opisuje wartości przesunięcia osi (tj. #pick, #start).
>
> Jeśli nazwa zmiennej nie posiada prefiksu i zaczyna się od dowolnej litery alfabetu, zmienna opisuje wartości przekształcenia (tj. pick, start).

2. Instrukcja POINT zastępuje zmienną pozycji wartością pochodzącą z uprzednio zdefiniowanego ustawienia.

```
POINT    pose_1=pose_2
```

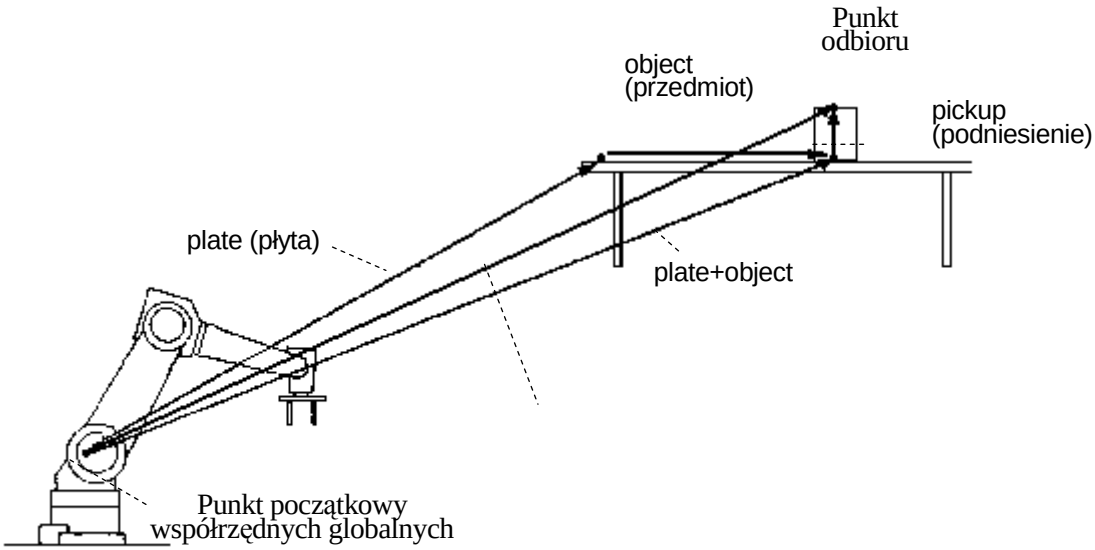
Wartości “pose _1” są zastępowane wartościami zdefiniowanej zmiennej “pose_2”. Jeśli “pose _2” nie jest zdefiniowana, występuje błąd.

### 3.5.3 WYKORZYSTYWANIE ZŁOŻONYCH WARTOŚCI PRZEKSZTAŁCENIA

Wartości przekształcenia pomiędzy dwoma współrzędnymi mogą być wyrażane jako połączenie wartości przekształcenia pomiędzy dwoma lub większą ilością przejściowych współrzędnych. Nosi to nazwę złożonych wartości przekształcenia lub względnych wartości przekształcenia.

Np. załóżmy, iż “plate” jest nazwą zmiennej dla wartości przekształcenia odnoszącej się do współrzędnych globalnych, określających współrzędne płyty. Następnie, jeśli ustawienie przedmiotu względem pozycji “plate” jest definiowane jako “object”, złożone wartości przekształcenia przedmiotu względem współrzędnych globalnych robota mogą być opisane jako “plate+object”.

W poniższym przykładzie, nawet jeśli ustawienie “plate” zmienia się (np. płyta przesuwa się), korekty wymagają tylko wartości przekształcenia “plate”, natomiast pozostałe mogą być użyte bez zmian.




Złożone wartości przekształcenia mogą być definiowane przy użyciu dowolnego polecenia lub instrukcji używanych do zdefiniowania zmiennej pozycji. (Najłatwiej jest używać polecenia HERE.)

Najpierw, używając programatora ręcznego, przesuń krokowo narzędzie robota do ustawienia, które ma być nazwane “plate”. Następnie wprowadź jak poniżej, aby zdefiniować ustawienie płyty (plate).

```
>HERE   plate  ↵
```

Przesuń krokowo narzędzie robota do ustawienia, które ma być nazwane “object” i wprowadź:

```
>HERE   plate + object
```

Wartości przekształcenia “object” definiują teraz bieżące ustawienie względem “plate”* (Jeśli w tym punkcie “plate” nie jest zdefiniowana, “object” nie zostanie zdefiniowany i wystąpi błąd).

Uwaga * Po wprowadzeniu polecenia HERE na ekranie pojawia się wartość przekształcenia ustawienia dla zmiennej położona najbardziej na prawo (tj. w tym przypadku “object”). Wartość nie odnosi się do “plate + value”. Aby wyświetlić wartości dla “plate +object”, użyj polecenia WHERE, gdy robot znajduje się w tym ustawieniu.

W końcu, przesuń krokowo chwytak robota do ustawienia, w którym ma on podnieść przedmiot i wprowadź:

```
>HERE   plate + object + pickup  ↵
```

To ostatnie polecenie określa “pickup” odnoszące się do wartości przekształcenia “object”.

Jak przedstawiono powyżej, złożone wartości przekształcenia są definiowane przez kombinację kilku wartości przekształcenia oddzielonych "+". Nie wprowadzaj żadnych spacji pomiędzy znak "+" a wartości przekształcenia. Używając tej metody, możesz zestawiać dowolną niezbędną ilość wartości przekształcenia.
Jeśli robot ma podnieść przedmiot w ustawieniu określonym jako "pickup", zdefiniowanym w relacji do "object", program zostanie zapisany jak poniżej:

```
        JMOVE   plate+object+pickup
lub     LMOVE   plate+object+pickup
```

**[ UWAGA ]**

1. Nie zmieniaj kolejności, w której wyrażona jest względna wartość przekształcenia. Np. jeśli zmienna wartości przekształcenia "b" jest zdefiniowana względem zmiennej wartości przekształcenia "a", "a+b" daje oczekiwany wynik, jednak "b+a" niekoniecznie.
2. Dane ustawienia "object" i "pickup" z powyższego przykładu są zdefiniowane względem innych danych ustawienia. Z tego względu, nie używaj poleceń jak "JMOVE object" lub "LMOVE pickup", jeśli nie masz pewności co do ich celu i wpływu na program.

Jeśli wykorzystujesz złożone wartości przekształcenia w sposób powtarzalny, użyj polecenia POINT, w celu skrócenia czasu obliczeń złożonych wartości przekształcenia. Np. aby zbliżyć się do ustawienia "pickup" i następnie przejść do niego możesz wprowadzić:

```
JAPPRO  plate + object + pickup, 100    przybliż się na 100 mm powyżej "pickup".
LMOVE   plate + object + pickup         przejdź ruchem liniowym do "pickup".
```

Zamiast tego, jeśli wprowadzisz jak poniżej, oszczędzi to czasu na obliczenia:

```
POINT   x = plate + object + pickup     obliczaj ustawienie docelowe.
JAPPRO  x, 100                          przybliż się na 100 mm powyżej celu.
LMOVE   x                               przejdź ruchem liniowym do celu.
```

Oba powyższe programy skutkują tym samym ruchem, jednak ostatni z nich oblicza złożone przekształcenie tylko raz, więc czas wykonania jest krótszy. W przypadku tak prostych przykładów, różnica będzie nieznaczna, jednak im bardziej złożony program, wzrasta ona, a wraz z nią skróceniu ulega ogólny czas cyklu.

**[ UWAGA ]**

W przypadku robotów z 7 osiami, zapamiętaj co następuje:

1. Używając polecenia POINT zwróć uwagę na wartość JT7. Na przykład w:

   ```
   POINT p=p1+p2
   ```

   Wartość JT7 przypisana do “p” będzie wartością JT7 w “p2”. Wartość zmiennej położonej najbardziej na prawo od wyrażenia jest przypisana do zmiennej po lewej od “=”.

2. Przypisując określoną wartość JT7, dodaj “/7” na końcu polecenia POINT. Na przykład,

   ```
   POINT/7 p = TRANS(,,,,,,value)
   ```

   przypisując “value”(wartość) do wartości JT7 dla zmiennej “p”.

## 3.6 DEFINIOWANIE ZMIENNYCH RZECZYWISTYCH

Zmienne rzeczywiste są definiowane poprzez instrukcje przyporządkowania (=). Dla zmiennej rzeczywistej obowiązuje następujący format:

Real_variable_ name = numeric_value (Nazwa_zmiennej_ rzeczywistej = wartość_liczbowa)

**Przykład**

```
a=10.5
count=i*2+8
Z[2]=Z[1]+5.2
```

Zmienna po lewej może być zmienną skalarną (tj., liczbą) lub elementem tablicy (tj., Z[2]). Zmienna jest definiowana, jeśli jest jej przypisana wartość. Natomiast, do momentu przypisania jej wartości, zmienna ta pozostaje niezdefiniowana, a jeśli niezdefiniowana zmienna jest wykorzystywana przez wykonywany program, występuje błąd.

Wartość liczbowa po prawej może być stała, zmienna lub też może być wyrażeniem liczbowym. Jeśli instrukcja przyporządkowania jest przetwarzana, wartość po prawej od instrukcji przyporządkowania obliczana jest w pierwszej kolejności, następnie wartość ta jest przypisywana zmiennej znajdującej się po lewej.

Jeśli zmienna po lewej od instrukcji jest nowa i nigdy wcześniej nie przypisano jej wartości, wartość znajdująca się po prawej przypisywana jest tej zmiennej automatycznie. Jeśli zmienna znajdująca się po lewej jest już zdefiniowana, nowa wartość zastępuje wartość bieżącą.

Np. instrukcja “x=3” przypisuje wartość 3 zmiennej “x”. Czytaj, “przydziel 3 do x”, nie “x jest równe 3”. Poniższy przykład ilustruje w sposób czytelny kolejność przetwarzania:

x= x+1.

Gdyby przykład ten stanowił równanie matematyczne, byłoby ono czytane jako “x jest równe x plus 1”, co jest pozbawione sensu. Instrukcja przyporządkowania jest czytana “przydziel wartość x plus 1 do x”. W takim przypadku obliczana jest suma bieżącej wartości “x” i 1, a następnie jest ona przydzielana do “x” jako nowa wartość. Równanie takie wymaga uprzedniego zdefiniowania x, jak poniżej:

```
x=3
x=x+1
```

W takim przypadku, wartością wyniku “x” jest 4.

## 3.7 DEFINIOWANIE ZMIENNYCH CIĄGU ZNAKÓW

Zmienne ciągu znaków są definiowane poprzez instrukcje przyporządkowania (=). Dla zmiennej znaków obowiązuje następujący format:

$string_variable=string_value ($zmienna_ciągu znaków=wartość_ciągu znaków)

**Przykład**

```
$a1=$a2
$error mess[2]="time over"
```

Zmienna ciągu po lewej może być zmienną (tj., $name) lub elementem tablicy (tj., $line[2]). Zmienna jest definiowana, jeśli jest jej przypisana wartość. Natomiast, do momentu przypisania jej wartości, zmienna ta pozostaje niezdefiniowana, a jeśli niezdefiniowana zmienna jest wykorzystywana przez wykonywany program, występuje błąd.

Ciąg znaków po prawej może być stałą typu łańcuchowego, zmienną łańcuchową lub też może być wyrażeniem ciągu. Jeśli instrukcja przyporządkowania jest przetwarzana, wartość po prawej obliczana jest w pierwszej kolejności, następnie wartość ta jest przypisywana zmiennej znajdującej się po lewej.

```
$name = "KAWASKI  HEAVY  INDUSTRIES  LTD."
```

W powyższej instrukcji ciąg ujęty w cudzysłów “” zostanie przypisany do zmiennej “$name”. Jeśli zmienna po lewej od instrukcji nigdy wcześniej nie była wykorzystywana, ciąg zostanie przypisany automatycznie. Jeśli zmienna znajdująca się po lewej jest już zdefiniowana, instrukcja ta w miejsce bieżącego ciągu wstawia nowy ciąg po prawej.

## 3.8 WYRAŻENIA LICZBOWE

Wyrażenia liczbowe mogą zawierać cyfry, zmienne, specjalne funkcje lub inne wyrażenia liczbowe łączone razem operatorami. Wszystkie wyrażenie liczbowe, których wartość jest wyznaczana przez system, dają wynik w liczbach rzeczywistych. Wyrażenie liczbowe mogą być używane dowolnie w miejsce wartości liczbowych. Mogą one być wykorzystywane jako parametry w poleceniach wprowadzonych z ekranu, w instrukcjach programu lub też jako indeksy tablic.

Interpretacja wartości zależy od kontekstu, w którym pojawia się wyrażenie. Np. wyrażenie określone dla indeksu tablicy jest interpretowane jako dające liczbę całkowitą. Wyrażenie określone dla wartości logicznej jest interpretowane jako fałsz (false), jeśli jego wartością wynikową jest 0 i prawda (true), jeśli wartość ta jest różna od 0.

### 3.8.1 OPERATORY

Do określania wyrażeń służą operatory arytmetyczne, logiczne i binarne. Wszystkie operatory łączą dwie wartości, aby uzyskać pojedynczą wartość wynikową. Wyjątki: dwa operatory (NOT i COM) pojawiają się z jedną wartością, a operator (–) pojawia się z jedną lub z dwoma wartościami. Poniżej opisano poszczególne operatory.

| | | |
|---|---|---|
| Operatory arytmetyczne | + | Dodawanie |
| | – | Odejmowanie lub negacja |
| | * | Mnożenie |
| | / | Dzielenie |
| | ^ | Potęga |
| | MOD | Reszta |
| Operatory relacyjne | < | Mniej niż |
| | <=, =< | Mniej niż lub równe |
| | == | Równe |
| | <> | Różne od |
| | >=, => | Większe niż lub równe |
| | > | Większe niż |
| Operatory logiczne | AND | Logiczne I |
| | NOT | Logiczne uzupełnienie |
| | OR | Logiczne LUB |
| | XOR | Wyłączające logiczne LUB |
| Operatory binarne | BAND | Binarne I |
| | BOR | Binarne LUB |
| | BXOR | Binarne XOR |
| | COM | Uzupełnienie |

**[ UWAGA ]**

1. Operator relacyjny "==" jest operatorem sprawdzającym, czy dwie wartości są równe i czy różnią się od wskaźnika przypisania "=".

2. Operator binarny BOR wykonuje operacje OR w odniesieniu do odpowiedniego bita binarnego dwóch wartości liczbowych. ( W tym przypadku wartość jest wyrażona w notacji binarnej, jednak może być wykorzystana w dowolnej notacji.

   ^B101000 BOR ^B100001 → ^B101001

   Ten wynik różni się od wyniku z operacji OR.

   ^B101000 OR ^B100001 → -1(TRUE)

   W tym przypadku, ^B101000 i ^B100001 są interpretowane jako wartości logiczne i jako że żadna nie jest 0 (FALSE), wartość jest oceniana jako TRUE.

### 3.8.2 KOLEJNOŚĆ OPERACJI

Wyrażenia są oceniane zgodnie z sekwencją priorytetów. Priorytety te wyszczególniono poniżej, przypisując im wartości od 1 do 14. Zauważ, iż kolejność operacji może być sterowana dzięki zastosowaniu nawiasów grupujących składniki wyrażenia. W przypadku wyrażeń zawierających nawiasy, pierwsze obliczane jest wyrażenie znajdujące się w nawiasach znajdujących się najgłębiej, a następnie system przetwarza nawiasy znajdujące się coraz bardziej na zewnątrz.

1. Oceniaj funkcje i tablice.
2. Przetwarzaj operatory relacyjne dotyczące ciągów znaków (Patrz 3.9 Wyrażenia ciągów).
3. Przetwarzaj operatory potęgi "^".
4. Przetwarzaj operatory jednoargumentowe "-"(negacja), NOT, COM.
5. Przetwarzaj mnożenie "*" i dzielenie"/" od lewej do prawej.
6. Obliczaj resztę (operacja MOD) od lewej do prawej.
7. Przetwarzaj dodawanie"+" i odejmowanie"-" od lewej do prawej.
8. Przetwarzaj od lewej do prawej operatory relacyjne.
9. Przetwarzaj od lewej do prawej operatory BAND.
10. Przetwarzaj od lewej do prawej operatory BOR.
11. Przetwarzaj od lewej do prawej operatory BXOR.
12. Przetwarzaj od lewej do prawej operatory AND.
13. Przetwarzaj od lewej do prawej operatory OR.
14. Przetwarzaj od lewej do prawej operatory XOR

### 3.8.3 WYRAŻENIA LOGICZNE

Wyrażenia logiczne dają wartości logiczne TRUE lub FALSE. Wyrażenia logiczne mogą być wykorzystywane w programach jako warunki determinujące kolejne operacje programów. W poniższym przykładzie proste wyrażenie logiczne, "x>y", jest wykorzystywane w podprogramie standardowym w celu określenia, która z dwóch zmiennych ma być przypisana do zmiennej "max".

```
IF x>y GOTO 10
max=y
GOTO 20
10          max=x
20 RETURN
```

Podczas oceny wyrażenia logicznego, wartość zero została uznana jako FALSE, a wszystkie wartości niezerowe jako TRUE. Dlatego też wszystkie wartości rzeczywiste lub wyrażenia wartości rzeczywistych mogą być używane jako wartość logiczna.

Np. dwie poniższe instrukcje posiadają identyczne znaczenia, ale druga z nich jest łatwiejsza do zrozumienia.

```
IF x   GOTO   10
IF x<>0 GOTO 10
```

## 3.9 WYRAŻENIA CIĄGU ZNAKÓW

Wyrażenia ciągu mogą zawierać ciągi znaków, zmienne ciągu, specjalne funkcje lub inne wyrażenia ciągu łączone razem operatorami. Poniżej przedstawiono operatory używane z wyrażeniami ciągu.

| Operator ciągu | + | Zestawia |
|---|---|---|
| Operatory relacyjne | <<br><=, =<<br>==<br><><br>>=, =><br>> | Mniej niż<br>Mniej niż lub równe<br>Równe<br>Różne od<br>Większe niż lub równe<br>Większe niż |

W wyniku użycia operatora ciągu otrzymasz ciąg, a w przypadku użycia operatora relacyjnego, wartość rzeczywistą.

Jeśli użyjesz operatory relacyjne wraz z ciągami znaków, ciągi będą porównywane znak ze znakiem, zaczynając od pierwszego znaku ciągu. Jeśli wszystkie znaki okażą się identyczne, dwa ciągi zostaną uznane za równe, jeśli jednak będzie istniała choćby tylko jedna różnica, ciąg posiadający wyższy kod znaku zostanie oceniony jako ciąg większy. Jeśli jeden z ciągu będzie krótszy, zostanie on niżej oceniony. W przypadku operatorów relacyjnych z ciągami, spacje i tabulacje są traktowane jako znaki.

```
"AA"        <     "AB"
"BASIC" == "BASIC"
"PEN."    >    "PEN"
"DESK"    <    "DESKS"
```

**[ UWAGA ]**

W przypadku wyrażeń ciągu wielkie i małe litery są uznawane za różne znaki.

## 4.0 PROGRAMY W JĘZYKU AS

Niniejszy rozdział poświęcony jest programowaniu w języku AS. Wyjaśniono w nim programowanie i wykonywanie programów oraz ruchy robota. Po zapoznaniu się z treścią tego rozdziału, dla pogłębienia wiedzy, zaleca się rzeczywistą obsługę systemu lub PC-ROSET[*].

Uwaga[*] PC-ROSET jest zgodnym z systemem AS symulatorem robota, wykorzystującym komputer osobisty.

4.1 Typy programów AS
4.2 Tworzenie i edycja programów
4.3 Wykonywanie programów
4.4 Przepływ (flow) wykonywania programów
4.5 Ruch robota

## 4.1 TYPY PROGRAMÓW AS

Program jest serią instrukcji mówiących robotowi, w jaki sposób ma się poruszać, wysyłać sygnały, wykonywać obliczenia, itp. w danym procesie. Nazwa programu składa się z nie więcej niż 15 znaków, przy czym pierwszy z nich to znak alfabetyczny i może on zawierać tylko litery, cyfry i kropki. Możesz tworzyć dowolną ilość programów, jeśli tylko mieszczą się one w pamięci. Programy są zazwyczaj tworzone przy pomocy trybu edycji systemu AS, ale możesz także używać osobnego komputera, na którym załadowano oprogramowanie terminala KRterm, KCwin32 lub oprogramowanie PC-ROSET, a następnie ładować gotowe programy do pamięci robota.

### 4.1.1 PROGRAM STERUJĄCY ROBOTEM

Program sterujący robotem to program, który steruje ruchami robota. Do tworzenia tego typu programu możesz używać wszystkich instrukcji programu, w tym instrukcji ruchu robota.

### 4.1.2 PROGRAMY DZIAŁAJĄCE RÓWNOLEGLE (PC PROGRAM )

Programy kontrolne procesu działające równolegle lub programy sterujące procesem mogą być wykonywane jednocześnie z programami sterującymi robotem. Programy kontrolne procesu działające równolegle są używane do kontroli lub monitoringu urządzeń zewnętrznych przy pomocy zewnętrznych sygnałów we/wy. Program kontrolny procesu działający równolegle i program sterujący robotem mogą komunikować się ze sobą przy pomocy zwykłych zmiennych lub sygnałów wewnętrznych.

Programy kontrolne procesu działające równolegle i programy sterujące robotem wykorzystują wspólne instrukcje. Dlatego, w niektórych przypadkach, program kontrolny procesu działający równolegle może być wykonywany jako program sterujący robotem. Jednakże, instrukcje ruchu inne niż instrukcja BRAKE nie mogą być używane w programach kontrolnych procesu działających równolegle. Również funkcje BASE i TOOL nie są dostępne dla programów kontrolnych procesu działających równolegle.

### 4.1.3 AUTOSTART

Program kontrolny procesu działający równolegle może być skonfigurowany w taki sposób, że po włączeniu zasilania zostanie on uruchomiony automatycznie.

1. Włącz parametr systemowy AUTOSTART.PC (lub AUTOSTART2.PC – AUTOSTART5.PC).

2. Utwórz program, który ma być uruchomiony automatycznie i nazwij go AUTOSTART.PC (lub AUTOSTART2.PC – AUTOSTART5.PC).

Niektóre polecenia wprowadzane z ekranu mogą być wykonywane przy pomocy instrukcji programu MC;
np. MC CONTINUE, itp. (Patrz 6.9 Instrukcje programu MC.)

Poniżej pokazano przykładowy program, który jest uruchamiany automatycznie (autostart program). W niniejszym przykładzie robot monitoruje MOTOR POWER (zasilanie silnika) i wykonuje program pg1, jeśli włączone jest zasilanie. Aby ułatwić zrozumienie, pominięto kontrole bezpieczeństwa (safety checks), jednak w sytuacjach rzeczywistych upewnij się, że uwzględnione zostały także procedury kontroli bezpieczeństwa.

```
autostart.pc( )
1  WAIT SWITCH (POWER) ;oczekuje na włączenie MOTOR POWER (zasilania silnika)
2  WAIT SIG(27) ;sprawdza, czy robot znajduje się w pozycji wyjściowej (home position)*
3  MC EXECUTE pg1;Wykonuje pg1(robot motion program - program ruchu robota)
```

Uwaga * Przed wykonaniem niniejszego programu skonfiguruj pozycję wyjściową (home position) i przydziel sygnałowi 27 sygnał dedykowany HOME1.

## 4.2 TWORZENIE I EDYCJA PROGRAMÓW

W niniejszej sekcji utworzono prosty program instruujący robot w wykonaniu określonego zadania. Program ten stanowi listę procedur, które robot będzie musiał przeprowadzić. Podczas wykonywania programu w systemie AS, kroki programu (wiersze) są przetwarzane od góry do dołu, a operacje określone w każdym kroku są wykonywane przez robota.

### 4.2.1 FORMAT PROGRAMU AS

Każdy wiersz (step - krok) programu języka AS jest wyrażany w następującym formacie.

**step number (numer kroku) label (etykieta) program instruction (instrukcja programu) ;comment (komentarz)**

1. Step number (numer kroku)
   Numery kroków są przydzielane automatycznie do wszystkich wierszy programu. Kroki są numerowane kolejno, zaczynając od 1, a po każdorazowym wstawieniu lub usunięciu wierszy ich numeracja jest zmieniana.

2. Label (etykieta)
   Etykiety są używane w programie do rozgałęziania programów. Etykieta może być liczbą całkowitą w zakresie od 1 do 9999 lub też ciągiem składającym się z maksymalnie 15 znaków alfabetycznych, kropek i podkreśleń (rozpoczynającym się znakiem alfabetycznym), po którym następuje dwukropek (:). Etykiety są wstawiane na początku wiersza programu, bezpośrednio po numerze kroku. Etykiety mogą być wykorzystywane jako miejsca docelowe gałęzi z dowolnego miejsca w programie.

3. Comment (komentarz)
   Średnik (;) wskazuje, iż wszystkie informacje znajdujące się po jego prawej stronie stanowią komentarz. Komentarze nie są przetwarzane jak instrukcje programu podczas wykonywania programu i używane są wyłącznie do objaśniania treści programu. Możliwe jest tworzenie wierszy programu zawierających wyłącznie komentarz, bez etykiety lub instrukcji. Możliwe jest także pozostawianie pustych wierszy w celu poprawienia czytelności programu. (Wiersz pusty składa się co najmniej z jednej spacji lub tabulacji następującej po średniku.)

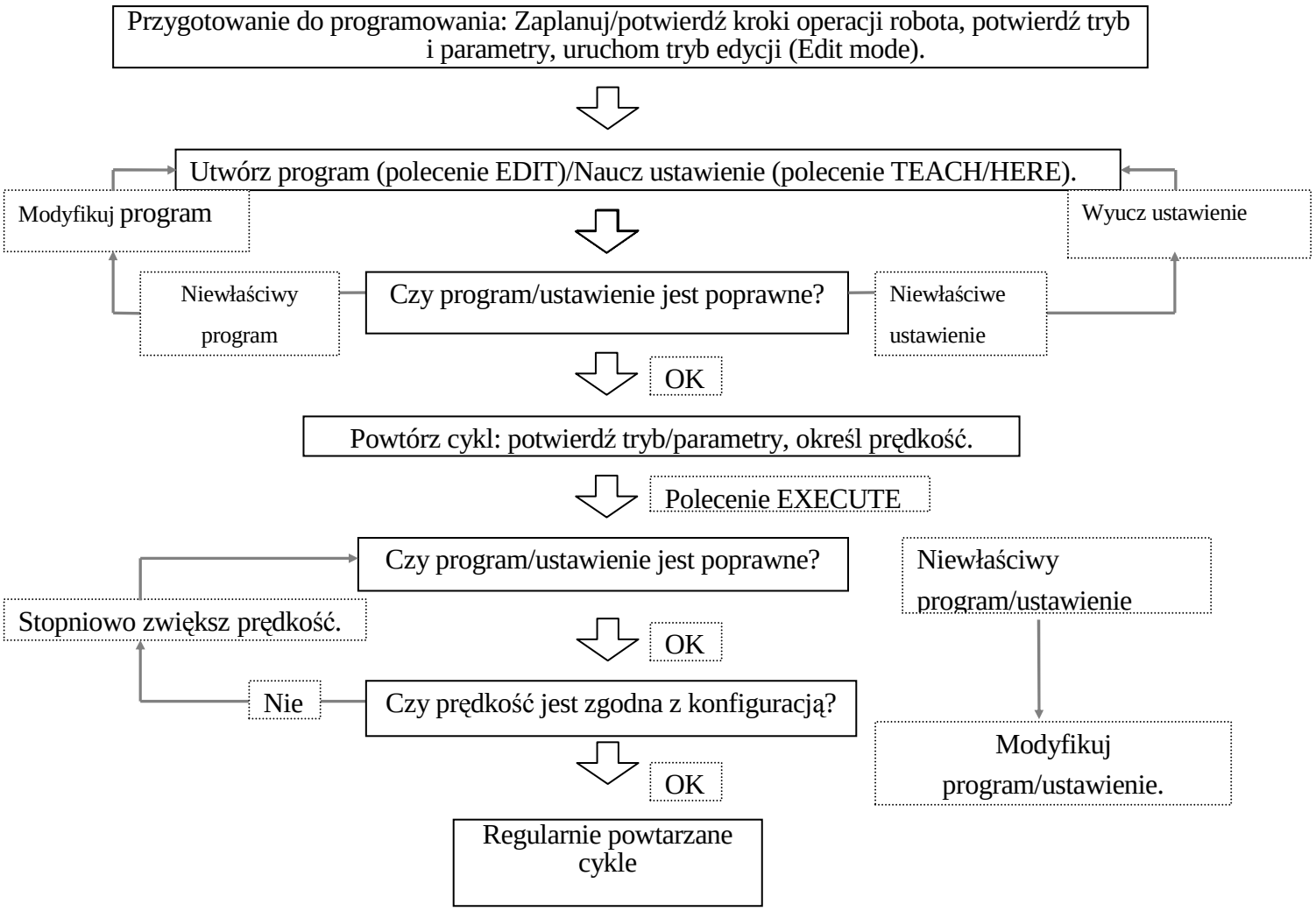
### 4.2.2 EDITOR COMMANDS (POLECENIA EDYCJI)

Do tworzenia i edycji programów wykorzystywane są następujące polecenia edycji. (Parametry wyróżnione mogą być pominięte.)

| | |
|---|---|
| EDIT program name, step (EDYTUJ nazwa programu, numer kroku) | Uruchamia tryb edycji. |
| Program instructions (instrukcje programu) | W miejsce bieżącego kroku wstawia nową instrukcję. |
| ENTER key (↵) (klawisz ENTER) | Przechodzi do kolejnego kroku, bez zmieniania bieżącego. |
| D step count (D liczba kroków) | Usuwa kroki programu (Delet - usuń). |
| E | Kończy tryb edycji i powraca do trybu monitorowania. |
| F character string (F ciąg znaków) | Wyszukuje znaki i wyświetla wiersz, w którym się znajdują. (Find - znajdź) |
| I | Wstawia nowy krok (Insert - wstaw). |
| L | Wyświetla poprzedni krok. (Last - ostatni) |
| M /existing characters/ new _characters (M/istniejące znaki/ nowe _znaki) | W miejsce istniejących znaków wstawia nowe. (Modify - modyfikuj) |
| O | Umieszcza kursor na bieżącym kroku w celu dokonania edycji. (Jeden wiersz). |
| P step count (P liczba kroków) | Wyświetla określoną liczbę kroków programu. (Print - druk) |
| R character string (R ciąg znaków) | Zastępuje znaki w kroku. |
| S step number (numeru kroku) | Wybiera krok programu. (Step - krok) |
| XD | Wycina i zapisuje wybrany krok lub kroki w schowku. |
| XY | Kopiuje i zapisuje wybrany krok lub kroki w schowku. |
| XP | Wkleja zawartość schowka. |
| XQ | Wkleja zawartość schowka w odwrotnej kolejności. |
| XS | Pokazuje zawartość schowka. |
| T | Uczy w trybie edycji (opcja). |

### 4.2.3 PROCEDURY PROGRAMOWANIA

Programowanie jest wykonywane w przedstawionych poniżej krokach:




### 4.2.4 TWORZENIE PROGRAMÓW

W programie AS niezbędne jest wyuczenie robota dwóch rzeczy:

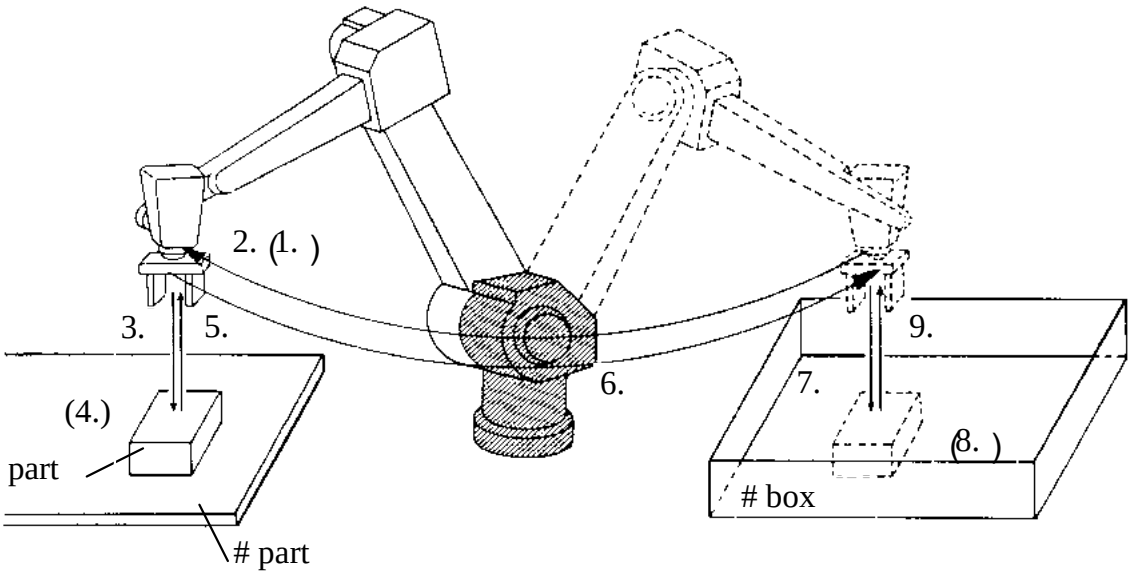
1. Warunków pracy (work conditions) robota.
2. Ścieżki (path) (pose - ustawienia) dla narzędzia robota.

Poniżej zaprezentowano przykładowy program. Robot wykona zadanie przedstawione na następnej stronie: podniesie część dostarczoną przez zsuwnię (przenośnik taśmowy) i umieści ją w pudełku.

Najpierw określ wszystkie ruchy wymagane do ukończenia zadania:

1. Sprawdź, czy chwytak jest otwarty.
2. Przesuń do ustawienia 50 mm powyżej części (#part) znajdującej się na zsuwni.
3. Przesuń prosto w dół do części (#part).
4. Zamknij chwytak i chwyć część.
5. Przesuń prosto w górę 150 mm ponad zsuwnię.
6. Przesuń do położenia 200 mm powyżej pudełka (#box).
7. Przesuń część w dół do pudełka.
8. Otwórz chwytak i zwolnij część.
9. Przesuń z powrotem do położenia 180 mm powyżej pudełka.

Zmienne #part i #box określające położenie oraz pozycję robota są zwane ustawieniem (pose) (location - lokalizacją) danych w systemie AS. Przed wykonaniem programu określ nazwę zmiennej ustawienia, jak pokazano w rozdziale 3.




Programy są tworzone i edytowane w edytorze AS (Editor). Aby utworzyć program o nazwie "demo", wprowadź "EDIT demo ↵". Powinien pojawić się ekran jak poniżej:

```
> EDIT demo ↵
.PROGRAM demo
1 ?
```

Teraz AS oczekuje na pierwszy krok, który ma być wprowadzony. Wprowadź "OPENI ↵ " po "1?"

```
> EDIT demo
  .PROGRAM demo
  1?OPENI ↵
  2 ?
```

Następnie, dla drugiego kroku, wprowadź “JAPPRO #part, 50 ↵”.

```
> EDIT demo
  .PROGRAM demo
  1 ? OPENI
  2 ? JAPPRO #part, 50 ↵
  3 ?
```

Wprowadź pozostałą część programu w identyczny sposób. Przed naciśnięciem ↵, popraw wszystkie błędy powstałe podczas wprowadzania kroków, naciskają klawisz cofający Backspace.

Jeśli na końcu błędnego kroku naciśnięty zostanie klawisz ↵, pojawia się komunikat błędu i krok jest odrzucany. W takim przypadku, ponownie wprowadź krok. Po wprowadzeniu całego programu powinien pojawić się ekran jak poniżej:

```
> EDIT demo
     .PROGRAM
    1 ? OPENI
    2 ? JAPPRO #part,50
    3 ? LMOVE #part
    4 ? CLOSEI
    5 ? LDEPART 150
    6 ? JAPPRO #box,200
    7 ? LMOVE #box
    8 ? OPENI
    9 ? LDEPART 180
   10 ? E   ↵
>
```

Ostatni krok “E ↵” nie jest poleceniem przeznaczonym dla robota, a poleceniem zakończenia trybu edycji (patrz tabela w 4.2.1). Program został ukończony. Jeśli program jest wykonywany, system AS przetwarza kroki w kolejności od 1 do 9.

Dalsze informacje na temat sposobów tworzenie programów, patrz 11. Przykładowe programy.

## 4.3 WYKONYWANIE PROGRAMÓW

Programy sterujące robotem i programy kontrolne procesu działające równolegle są wykonywane w odmienny sposób.

### 4.3.1 WYKONANIE PROGRAMU STERUJĄCEGO ROBOTEM

Aby wykonać program, ustaw przełącznik TEACH/REPEAT w pozycji REPEAT. Następnie upewnij się, że przełącznik TEACH LOCK, na programatorze ręcznym, jest w pozycji OFF. Potem włącz zasilanie silnika (MOTOR POWER) i ustaw przełącznik HOLD/RUN w pozycji RUN.

1. Wykonywanie programu przy użyciu polecenia EXECUTE

   Po pierwsze, ustaw prędkość monitorowania. Robot podczas wykonywania programu będzie się poruszał z tą prędkością. Prędkość powinna być ustawiona na wartość mniejszą od 30%, przy konfiguracji początkowej 10%.

   > SPEED 10 ↵

   Aby rozpocząć wykonywanie, użyj polecenia EXECUTE. Napisz jak poniżej:

   > EXECUTE demo ↵

   Robot powinien wykonać wybrane zadanie. Jeśli nie porusza się w przewidziany sposób, ustaw przełącznik HOLD/RUN w pozycji HOLD. Robot zwolni i zatrzyma się. W nagłych przypadkach, naciśnij przycisk EMERGENCY STOP na panelu kontrolera lub na programatorze ręcznym. Włączone zostaną hamulce i robot natychmiast się zatrzyma.

   Jeśli robot porusza się prawidłowo z prędkością 10%, następuje stopniowe przyspieszanie.

   > SPEED 30 ↵
   > EXECUTE demo ↵ Robot działa z prędkością 30%.

   > SPEED 80 ↵
   > EXECUTE demo ↵ Robot działa z prędkością 80%.

   Po co najmniej jednokrotnym wydaniu polecenia EXECUTE, do wykonania programów może być użyty, znajdujący się na panelu kontrolera, przycisk CYCLE START.

   Aby wykonać program więcej niż jeden raz, wprowadź po nazwie programu liczbę powtórzeń:

   > EXECUTE demo,5 ↵ Wykonuje 5 razy.
   > EXECUTE demo,–1 ↵ Wykonuje program w sposób ciągły.

2. Wykonywanie programu przy użyciu polecenia PRIME

Ustaw prędkość monitorowania w taki sam sposób, jak w przypadku polecenia EXECUTE I wykonaj polecenie PRIME.

>PRIME demo ↵

Robot jest teraz gotowy do wykonania programu. Naciśnięcie CYCLE START na panelu obsługi rozpoczyna wykonywanie. Wykonywanie można także uruchomić przy pomocy polecenia CONTINUE.

3. Wykonywanie programu przy pomocy polecenia STEP lub klawisza CHECK

Możliwa jest kontrola ruchu i treści programu poprzez wykonanie programu krok po kroku. Użyj polecenia wprowadzonego z ekranu STEP lub klawisza CHECK * znajdującego się na programatorze ręcznym.

Uwaga* W momencie używania klawisza CHECK, wykonywanie programu jest zatrzymywane na końcu każdej instrukcji ruchu.

Podczas wykonywania programu sterującego robotem, niektóre polecenia wprowadzane z ekranu są wyłączane. Podobnie, podczas wykonywania, polecenie EXECUTE nie może być wprowadzone dwukrotnie.

### 4.3.2 ZATRZYMYWANIE PROGRAMÓW

Istnieje kilka sposobów zatrzymywania programów, które są w toku wykonania. Przedstawione poniżej trzy z nich są opisane w kolejności od najbardziej do najmniej naglącego.

1. Naciśnij przycisk EMERGENCY STOP na panelu kontrolera lub na programatorze ręcznym. Włączone zostaną hamulce i robot zatrzyma się natychmiast. Poza nagłymi przypadkami, korzystaj z metody 2 i 3.

2. Przełącz wyłącznik HOLD/RUN (wstrzymaj/włącz) na panelu obsługi na pozycję HOLD. Robot zwolni i zatrzyma się.

3. Wprowadzenie polecenia ABORT zatrzymuje wykonanie programu po ukończeniu przez robota bieżącego kroku (instrukcji ruchu).

> ABORT ↵

Do zatrzymania wykonywania można także użyć polecenia HOLD.

> HOLD ↵

### 4.3.3 WZNAWIANIE PROGRAMU STERUJĄCEGO ROBOTEM

W zależności od tego, w jaki sposób zatrzymano program, istnieje kilka metod wznawiania programu.

1. Po zatrzymaniu robota przy pomocy przycisku EMERGENCY STOP, zwolnij blokadę EMERGENCY STOP i naciśnij MOTOR POWER, aby włączyć zasilanie silnika. Po naciśnięciu CYCLE START, robot rozpoczyna ruch.

2. Jeśli do zatrzymania robota użyto przełącznika HOLD/RUN, wykonanie wznawiamy ustawiając go w pozycji RUN.

3. Aby wznowić wykonanie po użyciu polecenia ABORT lub HOLD, lub też gdy program został zawieszony z powodu błędu, użyj polecenia CONTINUE. ( Jeśli wznawiasz wykonanie po wystąpieniu błędu, należy wyzerować błąd przed ponownym uruchomieniem programu.

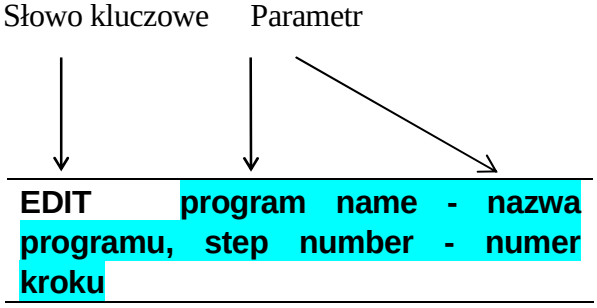
   > CONTINUE ↵

## 5.0 POLECENIA WPROWADZANE W TRYBIE „MONITOR”

W niniejszym rozdziale dokonano grupowania poleceń wprowadzanych w trybie „monitor” według poniższych kategorii i opisano szczegółowo każde polecenie. Polecenie wprowadzane w tym trybie składa się ze słowa kluczowego wyrażającego polecenie i parametru(ów) następującego(ych) po nim, jak pokazano w przykładzie poniżej.

5.1 Polecenia edycji
5.2 Polecenia sterujące programem i danymi
5.3 Polecenia dotyczące przechowywania programu i danych
5.4 Polecenia sterujące programem
5.5 Polecenia dotyczące informacji o pozycji robota
5.6 Polecenia sterujące systemem
5.7 Polecenia sygnałów binarnych
5.8 Polecenia dotyczące wyświetlania komunikatów

**PRZYKŁAD**




Parametry oznaczone mogą być pominięte.

Pomiędzy słowami kluczowymi a parametrami zawsze wprowadzaj spacje (space).

↵ oznacza klawisz Enter w niniejszym przykładzie.

## 5.1 POLECENIA EDYCJI

| | |
|---|---|
| EDIT | Uruchamia edytor programu. |
| C | Kończy edycję bieżącego programu i przechodzi do innego programu (Change - zmiana). |
| S | Wybiera krok programu, który ma być wyświetlany (Step - krok). |
| P | Wyświetla określony numer kroku programu (Print - druk). |
| L | Wybiera poprzedni krok (Last - ostatni). |
| I | Wstawia nowy krok (Insert - wstaw). |
| D | Usuwa krok programu (Delet - usuń). |
| F | Szuka znaku (Find - znajdź). |
| M | Zastępuje znaki (Modify - modyfikuj). |
| R | Zastępuje znaki (Replace - zastąp). |
| O | Umieszcza kursor na bieżącym kroku (One line - jedna linia). |
| E | Kończy działanie edytora (Exit - zakończ). |
| XD | Wycina i zapisuje wybrany krok lub kroki w schowku. |
| XY | Kopiuje i zapisuje wybrany krok lub kroki w schowku. |
| XP | Wkleja zawartość schowka. |
| XQ | Wkleja zawartość schowka w odwrotnej kolejności. |
| XS | Pokazuje zawartość schowka. |
| T | Uczy instrukcji ruchu w trybie edycji. (Opcja) |

## EDIT program name - nazwa programu, step number - numer kroku

**Funkcja**

Wchodzi w tryb edycji, umożliwiając tworzenie i edycję programów.

**Parametr**

1. Program Name (nazwa programu)

Wybiera program, który ma być edytowany. Jeśli nie określisz nazwy programu, otwarty do edycji zostanie program ostatnio edytowany lub wstrzymany (lub też zatrzymany z powodu błędu). Jeśli określony program nie istnieje, utworzony zostanie nowy program.

2. Step number - numer kroku

Określa numer kroku, który ma być edytowany. Jeśli nie określisz żadnego kroku, edycja zostanie rozpoczęta od ostatnio edytowanego kroku. Jeśli podczas ostatnio wykonywanego programu wystąpił błąd, wybierany jest krok, w którym wystąpił błąd.

[**UWAGA**]

Program nie może być edytowany podczas wykonywania.

Program nie może być wykonany lub usunięty podczas edycji. Jeśli program wywołuje aktualnie edytowany program, występuje błąd i wykonanie programu zostaje wstrzymane.

## C nazwa programu , numer kroku

**Funkcja**

Zmienia aktualnie wybrany program w trybie edycji.

**Parametr**

1. Program Name (nazwa programu)
Wybiera program, który ma być edytowany.

2. Step number - numer kroku
Określa numer kroku, który ma być edytowany. Jeśli nie określono żadnego kroku, wybierany jest pierwszy krok programu.

## S step number - numeru kroku

**Funkcja**

Wybiera i wyświetla określony krok do edycji. (Step - krok)

**Parametr**

Step number - numer kroku
Jeśli nie określono żadnego kroku, wybierany jest pierwszy krok programu. Jeśli numer kroku jest wyższy niż liczba kroków określonego programu, wybierany jest nowy krok następujący po ostatnim kroku programu.

## P step count - liczba kroków

**Funkcja**

Wyświetla określoną liczbę kroków, zaczynając od bieżącego kroku.

**Parametr**

Step count - liczba kroków

Określa liczbę kroków, które mają być wyświetlone. Jeśli liczba kroków nie jest określona, wyświetlany jest tylko bieżący krok.

**Objaśnienie**

Wyświetla tylko określoną liczbę kroków. Ostatni krok na liście jest gotowy do edycji.

## L

**Funkcja**

Wyświetla do edycji poprzedni (ostatni) krok. ([Current step number] −1=[step number of the step to be displayed]) - ([Numer bieżącego kroku] −1=[numer kroku, który ma być wyświetlony])

## I

**Funkcja**

Wstawia linię przed bieżącym krokiem.

**Objaśnienie**

Krokom następującym po wstawionej linii przydzielane są nowe numery. Aby zakończyć tryb wstawiania, naciśnij klawisz ↵. Wszystkie linie zapisane przed zakończeniem trybu wstawiania są wstawione do programu.

**Przykład**

Polecenie CLOSEI jest wstawione pomiędzy krokami 3 a 4.

```
1?OPENI
2?JAPPRO #PART, 500
3?LMOVE #PART
4?LDEPART 1000
 5? S 4                          ;Wyświetl krok 4, aby wstawić przed nim linię.
4     LDEPART 1000
4? I                             ;Napisz polecenie I.
  4I CLOSEI                      ;Napisz w instrukcjach dla wstawionej linii.
5I Błąd! Nie można tworzyć obiektów przez edycję kodów pól.   ;Naciśnij enter,
aby zakończyć wstawianie linii.
 5 LDEPART 1000                  ;Krok 4 posiada teraz nowy numer jako krok 5.
 5?
```

[**UWAGA**]

Aby wstawić pustą linię, naciśnij Spacebar (spację) lub TAB (tabulację), a następnie ↵, będąc w trybie wstawiania.

## D step count - liczba kroków

**Funkcja**

Usuwa określoną liczbę kroków, włącznie z krokiem bieżącym.

**Parametr**

Step count - liczba kroków

Określa liczbę kroków, które mają być usunięte, zaczynając od bieżącego kroku. Jeśli liczba kroków nie jest określona, usuwany jest tylko bieżący krok.

**Objaśnienia**

Usuwa tylko określoną liczbę kroków, zaczynając od bieżącego kroku. Po usunięciu, wszystkim pozostałym krokom przypisywane i wyświetlane są nowe numery.

[**UWAGA**]

Jeśli określona liczba kroków jest większa od liczby kroków programu, usuwane są wszystkie kroki następujące po bieżącym kroku.

## F character string - ciąg znaków

**Funkcja**

Wyszukuje (finds) w bieżącym programie określony ciąg, począwszy od bieżącego kroku do ostatniego i wyświetla pierwszy krok, który zawiera określony ciąg.

**Parametr**

Character string - ciąg znaków

Określa ciąg znaków do wyszukania.

**Przykład**

Wyszukuje w kroku następującym po bieżącym kroku ciąg znaków “abc” oraz wyświetla krok zawierający ten ciąg.

```
1?F abc ↵
3          JMOVE abc
3?
```

## M /existing characters/ new characters - /istniejące znaki/ nowe znaki

**Funkcja**

Modyfikuje znaki w bieżącym kroku.

**Parametr**

1. Istniejące znaki

Określa, które znaki są w bieżącym kroku nadpisywane.

2. Nowe znaki

Określa znaki, które zastępują istniejące znaki.

**Przykład**

Modyfikuje krok 4, wstawiając zamiast zmiennej ustawienia abc, zmienną def.

```
4          JMOVE abc
4?M/abc/def ↵
4          JMOVE def
4?
```

## R character string - ciąg znaków

**Funkcja**

Zastępuje istniejące znaki w bieżącym kroku, określonymi znakami.

**Parametr**

Character string - ciąg znaków
Określa nowe znaki, które zastępują istniejące znaki.

**Objaśnienie**

Poniżej przedstawiono procedurę wykorzystywania polecenia R:

1. Używając Spacebar przesuń kursor pod pierwszy znak, który ma być zastąpiony.

2. Naciśnij klawisz R i następnie Spacebar.

3. Wprowadź nowe znaki. Pamiętaj, że nowowprowadzone znaki nie zastępują znaków położonych powyżej kursora, tylko te, które znajdują się o dwie spacje na lewo, zaczynając powyżej R. (Patrz poniższy przykład)

4.Naciśnij ↵.

Po naciśnięciu ↵ system AS sprawdza, czy linia jest poprawna. W przypadku błędu, wpis jest ignorowany.

**Przykład**

Przy pomocy polecenia R prędkość jest zmieniana z 20 do 35.

```
1       SPEED   20 ALWAYS
1?              R 35 ↵
1       SPEED   35 ALWAYS
1?
```

## O

**Funkcja**

Umieszcza kursor na bieżącym kroku w celu dokonania edycji. (“O” od “one line” - jedna linia, nie zero).

**Przykład**

Przy pomocy polecenia O zmienna ustawienia abc jest zmieniana na zmienną def. Kursor jest przesuwany przy pomocy klawisza ← lub →.

```
3        JMOVE abc
3?O ↵
3     JMOVE abc BackSpace          ; usuń “abc” używając Backspace
3     JMOVE def ↵                  ; Wprowadź “def”
3        JMOVE def
3?
```

[**UWAGA**]

Polecenie to nie może być stosowane z programatora ręcznego.

## E

**Funkcja**

Kończy tryb edycji i powraca do trybu monitorowania.

## XD step count - liczba kroków

**Funkcja**

Wycina określoną liczbę kroków z programu i zapisuje je w pamięci buforowej.

**Parametr**

Step count - liczba kroków
Określa liczbę kroków, które mają być wycięte i zapisane w pamięci buforowej, zaczynając od bieżącego kroku. Wyciąć można do dziesięciu kroków. Jeśli nie określono, wycinany jest tylko bieżący krok.

**Objaśnienie**

Wycina określoną liczbę kroków i zapisuje je w pamięci buforowej.

Polecenie XY kopiuje i nie wycina kroków, a polecenie XD wycina kroki. Pozostałe kroki w programie otrzymują, odpowiednio, nowe numery.

## XY step count - liczba kroków

**Funkcja**

Kopiuje określoną liczbę linii i zapisuje je w pamięci buforowej.

**Parametr**

Step count - liczba kroków
Określa liczbę kroków, które mają być skopiowane i zapisane w pamięci buforowej. Skopiować można do dziesięciu kroków. Jeśli liczba kroków nie jest określona, kopiowany jest tylko bieżący krok.

**Objaśnienia**

Kopiuje określoną liczbę kroków, w tym bieżący krok i zapisuje je w pamięci buforowej.

Polecenie XD wycina kroki, a polecenie XY kopiuje kroki. Program pozostaje taki sam, a liczba kroków nie zmienia się po zastosowaniu polecenia XY.

## XP

**Funkcja**

Wstawia zawartość pamięci buforowej przed bieżącym krokiem.

**Objaśnienie**

Aby zapisać pożądaną zawartość w pamięci buforowej, użyj polecenia XD lub XY przed omawianym poleceniem

## XQ

**Funkcja**

Wstawia zawartość pamięci buforowej w odwróconej kolejności przed bieżącym krokiem.

**Objaśnienie**

Wstawia zawartość pamięci buforowej w odwróconej kolejności, jak w przypadku polecenia XP.

## XS

**Funkcja**

Wyświetla zawartość pamięci buforowej.

**Objaśnienie**

Wyświetla bieżącą zawartość pamięci buforowej. Jeśli pamięć buforowa jest pusta, nic nie zostanie wyświetlone.

## T variable name - nazwa zmiennej

Opcja

**Funkcja**

Umożliwia, w trybie edycji, uczenie instrukcji ruchu (JMOVE, LMOVE, itp.), przy pomocy ręcznego programatora.

**Parametr**

Nazwa zmiennej

Określa nazwę zmiennej ustawienia docelowego, które ma być uczone, wyrażoną w wartościach przekształcenia lub w wartościach przesunięcia osi. Jeśli jest ona określona w formie A[ ], to jest odczytywana jako zmienna tablicowa. W takim przypadku, zmienne nie mogą być wykorzystywane w numerach elementów. Jeśli są one pomijane, bieżące wartości przesunięcia osi są uczone jako ustawienie stałe.

**Objaśnienie**

Wprowadź to polecenie w trybie edycji. W trakcie wykonywania, ręczny programator wyświetla specjalny ekran uczenia. Wyuczone w tym miejscu ruchy są zapisywane jako instrukcje w programie w kroku, gdzie wprowadzane jest polecenie T. Jeśli przeprowadzane jest uczenie więcej niż jednego kroku, zmienna otrzymuje nową nazwę w taki sposób, iż zwiększana jest ostatnia liczba w nazwie zmiennej. Dalsze informacje, patrz Instrukcja obsługi.

**Przykład**

Z parametrem

2 JAPPRO #a

3? T pos Wyucz przy pomocy TP (ręcznego programatora). Aby wrócić do systemu AS, naciśnij Cancel.

(W tym miejscu przeprowadzane jest uczenie 3 kroków.)

3 JMOVE pos0

4 JMOVE pos1

5 LMOVE pos2

⋮

Bez parametru

2 JAPPRO #a

3? T Wyucz przy pomocy TP (ręcznego programatora) wartości osi. Aby zakończyć, naciśnij Cancel.

(W tym miejscu przeprowadzane jest uczenie 2 kroków.)

3 JMOVE #[0,10,20,0,0,0]

4 JMOVE #[10,10,20,0,0,0]

⋮

**⚠ OSTRZEŻENI**

**Aby korzystać z omawianego polecenia, niezbędne jest połączenie programatora ręcznego z kontrolerem. Ponadto, robot musi być w trybie uczenia (Teach mode) i posiadać włączoną blokadę w trakcie uczenia - tj. TEACH LOCK w pozycji ON.**

## 5.2 POLECENIA STERUJĄCE PROGRAMEM I DANYMI

| | |
|---|---|
| CARD_FDIR* | Sporządza wykaz nazw programów i zmiennych na karcie PC. |
| LIST | Wyświetla wszystkie kroki programu i wartości zmiennych. |
| LIST/P | Wyświetla wszystkie kroki programu. |
| LIST/L | Wyświetla wszystkie zmienne ustawienia i ich wartości. |
| LIST/R | Wyświetla wszystkie zmienne rzeczywiste i ich wartości. |
| LIST/S | Wyświetla wszystkie zmienne ciągu i ich dane. |
| DELETE | Usuwa programy i zmienne z pamięci robota. |
| DELETE/P | Usuwa programy z pamięci robota. |
| DELETE/L | Usuwa zmienne ustawienia z pamięci robota. |
| DELETE/R | Usuwa zmienne rzeczywiste z pamięci robota. |
| DELETE/S | Usuwa zmienne ciągu z pamięci robota. |

Uwaga* Polecenia te są używane do sterowania pamięcią kart PC, jednak współpracują one z dyskietkami, jeśli CARD_ jest zmienione na FD_. Dalsze informacje, patrz objaśnienia dla każdego polecenia.

## CARD_FDIR

## FD_FDIR

**Funkcja**

Wyświetla nazwy programów lub zmiennych. CARD_ FDIR odnosi się do danych na kartach PC, a FD_ FDIR na dyskietkach.

**Objaśnienie**

Użycie poleceń CARD_FDIR i FD_FDIR powoduje wyświetlenie wszystkich podprogramów standardowych i zmiennych w programach.

**Przykład**

| | |
|---|---|
| >FD_FDIR ↵ | Wyświetla nazwy wszystkich programów, podprogramów standardowych i zmiennych na dyskietce. |
| >CARD_FDIR | Wyświetla nazwy wszystkich programów, podprogramów standardowych i zmiennych na karcie PC. |

Jeśli przełącznik SCREEN jest w pozycji ON, wyświetlacz nie wykonuje przewijania i zatrzymuje się na końcu ekranu. Aby kontynuować wyświetlanie, naciśnij Spacebar (spację). Aby zakończyć wyświetlanie, naciśnij ↵.

[**UWAGA**]

Nazwy programów i zmiennych z symbolem ∗ lub ~ wyświetlonym na początku nazwy oznaczają, że treść tych programów lub zmiennych wciąż nie została zdefiniowana.

| | | |
|---|---|---|
| **LIST** | **program name,** | (nazwa programu) |
| **LIST/P** | **program name,** | (nazwa programu) |
| **LIST/L** | **pose (location) variable, .....** | (zmienna ustawienia (lokalizacji)) |
| **LIST/R** | **real variable, .....** | (zmienna rzeczywista) |
| **LIST/S** | **string variable, .....** | (zmienna ciągu) |

**Funkcja**

Wyświetla określony program i dane.

**Parametry**

Program name - nazwa programu (/P), pose variable - zmienna ustawienia (/L), real variable - zmienna rzeczywista (/R), string variable - zmienna ciągu (/S)
Określa typ danych, które mają być wyświetlone. Jeśli nie określono, wyświetlane są wszystkie dane zawarte w pamięci. Jeśli dokonano wyboru zmiennej tablicowej, na ekranie są wyświetlane wszystkie jej elementy.

**Objaśnienie**

Polecenie LIST powoduje wyświetlenie wszystkich nazw programów, ich podprogramów standardowych i zmiennych. Z drugiej strony, polecenie LIST/P powoduje wyświetlenie wyłącznie treści programu głównego.

**Przykład**

| | |
|---|---|
| >LIST ↵ | Wyświetla zawartość wszystkich programów, w tym zmienne i ich wartości.. |
| >LIST/L ↵ | Wyświetla wszystkie zmienne ustawienia (lokalizacji) i ich wartości. |
| >LIST/R ↵ | Wyświetla wszystkie zmienne rzeczywiste i ich wartości. |
| >LIST test* ↵ | Wyświetla zawartość wszystkich programów zaczynających się od “test”, w tym ich podprogramy standardowe i ich wartości. |

Jeśli przełącznik SCREEN jest w pozycji ON, wyświetlacz nie wykonuje przewijania i zatrzymuje się po zapełnieniu ekranu. Aby kontynuować wyświetlanie, naciśnij Spacebar

(spację). Aby zakończyć wyświetlanie, naciśnij ↵.

**DELETE program name, .....(nazwa programu)**
**DELETE/P program name, .....(nazwa programu)**
**DELETE/L pose variable, ..... (zmienna ustawienia)**
**DELETE/R real variable [array elements] , ..... (zmienna rzeczywista [elementy tablicy])**
**DELETE/S string variable [array elements] , ..... (zmienna ciągu [elementy tablicy])**

**Funkcja**

Usuwa określone dane z pamięci.

**Parametry**

Program name - nazwa programu (/P), pose variable - zmienna ustawienia (/L), real variable - zmienna rzeczywista (/R), string variable - zmienna ciągu (/S)
Określa typ danych, które mają być usunięte.

**Objaśnienie**

Polecenie DELETE usuwa całkowicie określone programy; tj. program główny oraz następujące dane, jeśli są w nim używane. (Jednakże, dane wykorzystywane w innych programach nie są usuwane).

· Wszystkie podprogramy standardowe wywoływane przez program lub przez jego podprogramy standardowe.
· Wszystkie zmienne ustawienia używane w programie i w jego podprogramach standardowych.
· Wszystkie zmienne rzeczywiste używane w programie i w jego podprogramach standardowych.
· Wszystkie zmienne ciągu używane w programie i w jego podprogramach standardowych.

Polecenie DELETE/P, w przeciwieństwie do polecenia DELETE, usuwa wyłącznie program, bez podprogramów standardowych i zmiennych wykorzystywanych przez program.

Jeśli wraz z poleceniami DELETE/R i DELETE/S nie określono elementów tablicy, usuwane są wszystkie elementy zawarte w określonej zmiennej tablicowej. Jeśli elementy zostały określone, usuwane są wyłącznie określone elementy.

**Przykład**

>DELETE test ↵ Usuwa program “test” i wszystkie podprogramy standardowe oraz

zmienne używane w programie.

| | |
|---|---|
| >DELETE/P pg11,pg12 ↵ | Usuwa program “PG11” oraz “PG12”. (Podprogramy standardowe i zmienne nie są usuwane.) |
| >DELETE/R a ↵ | Usuwa wszystkie elementy zmiennej tablicowej “a”. |
| >DELETE/R a[10] ↵ | Usuwa 10-ty element zmiennej tablicowej “a”. |

## 5.3 POLECENIA DOTYCZĄCE PRZECHOWYWANIA PROGRAMU I DANYCH

| | |
|---|---|
| SAVE | Zapisuje określone programy. |
| SAVE/P * | Zapisuje programy. |
| SAVE/L * | Zapisuje zmienne ustawienia. |
| SAVE/R * | Zapisuje zmienne rzeczywiste. |
| SAVE/S * | Zapisuje ciągi znaków. |
| SAVE/A * | Zapisuje informacje pomocnicze. |
| SAVE/SYS * | Zapisuje dane systemowe. |
| SAVE/ROB * | Zapisuje dane dotyczące robota. |
| SAVE/ELOG * | Zapisuje dane dziennika błędów. |
| LOAD * | Ładuje programy i dane do pamięci robota. |

| |
|---|
| **SAVE/SEL file name program name, …… (nazwa pliku = nazwa programu)** |
| **CARD_SAVE/SEL file name = program name, …… (nazwa pliku = nazwa programu)** |
| **FD_SAVE/SEL file name = program name, …… (nazwa pliku = nazwa programu)** |

**Funkcja**

Polecenie SAVE zapisuje programy i zmienne w komputerze. (Stosuj wyłącznie, gdy PC jest połączony z kontrolerem robota.)
Polecenie CARD_SAVE zapisuje programy i zmienne na kartach PC.
Polecenie FD_SAVE zapisuje programy i zmienne na dyskietkach.

**Parametry**

1. File Name (nazwa pliku)

Zapisuje określone programy pod tą nazwą pliku. Jeśli nie określono rozszerzenia, do nazwy pliku automatycznie dodawane jest rozszerzenie “.as”.

2. Program Name (nazwa programu)

Wybiera program, który ma być zapisany. Jeśli nie określono, zapisywane są wszystkie programy istniejące w pamięci.

**Objaśnienie**

Polecenia SAVE/P, SAVE/L, SAVE/R, SAVE/S, SAVE/SYS zapisują każdy typ danych (odpowiednio programy, zmienne ustawienia, zmienne rzeczywiste, zmienne ciągu oraz dane systemowe) w oddzielnych plikach. Użycie samego polecenia SAVE powoduje zapis wszystkich pięciu typów danych w jednym pliku.

Polecenie SAVE (bez /SEL) zapisuje określone programy, w tym dowolne zmienne i podprogramy standardowe używane przez te programy.

Polecenie SAVE/SEL zapisuje wyłącznie program, bez podprogramów standardowych i zmiennych wykorzystywanych przez program.

**Przykład**

| | |
|---|---|
| >SAVE f3=cycle,motor ↵ | Zapisuje dane systemowe pod nazwą pliku “f3.as”, dwa programy “cycle” i “motor”, podprogramy standardowe nazwane od tych programów oraz zmienne używane przez te programy. |

[**UWAGA**]

Jeśli określona nazwa pliku istnieje już w pamięci, do nazwy istniejącego pliku automatycznie dodawana jest literka “b” przed rozszerzeniem pliku. Np., jeśli “file1.as”istnieje już w pamięci i wykonywane jest polecenie >SAVE file1 ↵ , plik ten otrzymuje nową nazwę “file1.bas”. Nowo utworzony plik zostanie nazwany “file1.as”.

**SAVE/P/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**SAVE/L/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**SAVE/R/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**SAVE/S/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**SAVE/A file name (nazwa pliku)**
**SAVE/SYS file name (nazwa pliku)**
**SAVE/ROB file name (nazwa pliku)**
**SAVE/ELOG file name (nazwa pliku)**

**CARD_SAVE/P/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**CARD_SAVE/L/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**CARD_SAVE/R/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**CARD_SAVE/S/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**CARD_SAVE/A file name (nazwa pliku)**
**CARD_SAVE/SYS file name (nazwa pliku)**
**CARD_SAVE/ROB file name (nazwa pliku)**
**CARD_SAVE/ELOG file name (nazwa pliku)**

**FD_SAVE/P/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**FD_SAVE/L/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**FD_SAVE/R/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**FD_SAVE/S/SEL file name=program name, …… (nazwa pliku = nazwa programu)**
**FD_SAVE/A file name (nazwa pliku)**
**FD_SAVE/SYS file name (nazwa pliku)**
**FD_SAVE/ROB file name (nazwa pliku)**
**FD_SAVE/ELOG file name (nazwa pliku)**

**Funkcja**

Zapisuje w pliku na dysku program (/P), pose variable - zmienną ustawienia (/L), real variable - zmienną rzeczywistą (/R), string variable - zmienną ciągu (/S), auxiliary information - informacje pomocnicze (/A), system data - dane systemowe (/SYS), robot data - dane robota (/ROB) oraz error log - dziennik błędów (/ELOG).

Podobnie jak w przypadku polecenia SAVE, CARD_SAVE/ i FD_SAVE/ są używane

odpowiednio do zapisywania plików na kartach PC i dyskietkach (Stosuj polecenie SAVE/ wyłącznie, gdy PC jest połączony z kontrolerem robota). Patrz 2.6.2 Przesyłanie i pobieranie danych).

**Parametry**

1. File Name (nazwa pliku)

Zapisuje dane pod tą nazwą pliku. Jeśli nie określono rozszerzenia, do nazwy pliku automatycznie dodawane są poniższe rozszerzenia, odpowiednio do typu danych zawartych w pliku:

| | | | |
|---|---|---|---|
| program | .PG | system data - dane systemowe | .SY |
| informacja o pozycji | .LC | robot data (dane robota) | .RB |
| zmienne rzeczywiste | .RV | error log - dziennik błędów | .EL |
| zmienne ciągu | .ST | | |
| informacje pomocnicze | .AU | | |

2. Program Name (nazwa programu)

Określa nazwę programu, który ma być zapisany. Jeśli nie określono, w pliku na dysku zapisywane są wszystkie programy i dane istniejące w pamięci.

**Objaśnienie**

1. SAVE/P

Zapisuje w określonym pliku na dysku określone programy i podprogramy standardowe wywołane przez programy (w tym podprogramy standardowe wywołane przez podprogramy standardowe).

Nazwy zapisanych w pliku programów są wyświetlane na terminalu systemowym. Mogą się także pojawiać nazwy dodatkowych programów, takich które nie zostały określone przy pomocy polecenia SAVE. Są to nazwy podprogramów standardowych wywoływanych przez określony program. Podprogramy te są zapisywane w tym samym pliku co programy.

Programy są zapisywane w pliku w kolejności alfabetycznej, niezależnie od kolejności, w której były zapisane.

2. SAVE/L, SAVE/R, SAVE/S

Zapisuje wyłącznie zmienne używane w określonych programach i podprogramach standardowych wywołane przez te programy. (/L: zapisuje tylko zmienne ustawienia, /R: zapisuje

tylko zmienne rzeczywiste, /S: zapisuje tylko zmienne ciągu)

3. SAVE/A
Zapisuje informacje pomocnicze.

4. SAVE/SYS
Zapisuje dane systemowe.

5. SAVE/ROB
Zapisuje dane dotyczące robota (dane robota).

6. SAVE/ELOG
Zapisuje dziennik błędów. Polecenie to nie może być wprowadzane razem z innymi poleceniami SAVE/. Np., SAVE/ELOG/R nie działa.

7. Jeśli wprowadzisz /SEL z /P,/L,/R,/S, zapisane zostaną wyłącznie główny program i zmienne wykorzystywane w głównym programie. Podprogramy standardowe i zmienne w podprogramach standardowych nie są zapisywane.

Jeśli określona nazwa pliku istnieje już w pamięci, do nazwy istniejącego pliku automatycznie dodawana jest literka “b” przed rozszerzeniem pliku. (Patrz okno UWAGA dotyczące polecenia SAVE).

**Przykład**

| | |
|---|---|
| >SAVE/L file2=pg1, pg2 ↵ | Zmienne ustawienia używane przez programy pg1 i pg2 są zapisane pod nazwą pliku “file2.lc”. |

## LOAD/Q file name (nazwa pliku)

## CARD_LOAD/Q file name (nazwa pliku)

## FD_LOAD/Q file name (nazwa pliku)

**Funkcja**

Polecenie LOAD ładuje pliki z pamięci komputera do pamięci robota. (Stosuj wyłącznie, gdy PC jest połączony z kontrolerem robota).
Polecenie CARD_LOAD ładuje pliki z karty PC do pamięci robota.
Polecenie FD_LOAD ładuje pliki z dyskietki do pamięci robota.

**Parametry**

File Name (nazwa pliku)
Zapisuje określone programy pod tą nazwą pliku. Jeśli nie określono rozszerzenia, do nazwy pliku automatycznie dodawane jest rozszerzenie “.as”.

**Objaśnienie**

Polecenie to ładuje dane (dane systemowe, programy i zmienne) z określonego pliku do pamięci robota. Próba ładowania nazwy programu, który już istnieje w pamięci wywołuje błąd i wykonanie polecenia LOAD jest przerywane.

**[UWAGA]**

W czasie ładowania zmiennych ustawienia, zmiennych rzeczywistych lub zmiennych ciągu, które już istnieją w pamięci, dane przechowywane w pamięci są nadpisywane bez ostrzeżenia. (Programy nie są nadpisywane.)

Jeśli polecenie LOAD jest anulowane w trakcie nadpisywania danych w pamięci, oryginalne dane są usuwane.

W przypadku polecenia LOAD z /Q, przed ładowaniem dowolnych danych systemowych lub programów pojawia się następujący komunikat:

```
Load? (1:Yes, 0:No, 2:Load all, 3:Exit)
```

Poniżej przedstawiono dostępne opcje:

1: Ładowanie danych.
0: Dane nie są ładowane, następuje przejście do następnych danych.

2: Ładowanie danych i pozostałych danych w pliku bez dodatkowych zapytań.
3: Dane nie są ładowane, następuje zakończenie polecenia LOAD.

Jeśli w programie znajduje się nieczytelny lub niepoprawny krok, wyświetlany jest poniższy komunikat: “The step format is incorrect (0:Continue load 1: Delete program and exit)” (Format kroku jest niepoprawny (0: Kontynuuj ładowanie 1: Usuń program i zakończ)). Jeśli operacja jest kontynuowana po wprowadzeniu “0”, po załadowaniu programu użyj edytora w celu poprawienia kroku.

**Przykład**

```
>LOAD    pallet ↵
Loading… (Ładowanie...)
System data - dane systemowe
Program     a1()
Program     test()
 :
Transformation values - wartości przekształcenia
Joint interpolation values - wartości interpolacji osiowej
Real values - wartości rzeczywiste
Loading done - wykonano ładowanie.
>
```

Ładuje dane z pliku “pallet.as” do pamięci.

```
>LOAD    f3.pg ↵
```

Ładuje wszystkie programy z pliku “f3.pg” do pamięci.

## 5.4 POLECENIA STERUJĄCE PROGRAMEM

| | |
|---|---|
| SPEED | Ustawia prędkość w trybie “Monitor” |
| EXECUTE | Wykonuje program. |
| STEP | Wykonuje jeden krok programu. |
| MSTEP | Wykonuje jedną instrukcję ruchu. |
| ABORT | Zatrzymuje wykonanie po ukończeniu bieżącego kroku. |
| HOLD | Zatrzymuje wykonanie. |
| CONTINUE | Wznawia wykonanie programu. |

## SPEED monitor speed (prędkość w trybie “monitor”)

**Funkcja**

Ustawia prędkość w trybie „monitor” w procentach.

**Parametr**

Monitor speed (prędkość w trybie „monitor”)

Ustawia prędkość w procentach. Jeśli wartość ta jest równa 100, prędkość będzie ustawiona na 100% prędkości maksymalnej. Jeśli wartość ta jest równa 50, prędkość będzie ustawiona na połowę prędkości maksymalnej.

**Objaśnienie**

Iloczyn prędkości w trybie „monitor” (ustawianej przy pomocy tego polecenia) i prędkości programu (ustawianej w programie przy pomocy instrukcji SPEED) określa prędkość ruchu robota. Np., jeśli prędkość w trybie „monitor” jest ustawiona na 50, a prędkość w programie jest ustawiona na 60, maksymalna prędkość robota będzie wynosiła 30%.

[**UWAGA**]

Jeśli iloczyn prędkości w trybie „monitor” i prędkości programu ma wyższą wartość niż 100, maksymalna prędkość robota jest automatycznie ustawiana na 100%.

Wartość domyślna prędkości w trybie „monitor” to 10%.

Polecenie to nie ma wpływu na prędkość ruchu, który jest aktualnie wykonywany. Nowa konfiguracja prędkości jest stosowana po ukończeniu bieżącego i zaplanowanego ruchu.

**Przykład**

Jeśli prędkość programu jest ustawiona na 100%:

| | |
|---|---|
| >SPEED 30 ↵ | Prędkość maksymalna jest ustawiona na 30%. |
| >SPEED 50 ↵ | Prędkość maksymalna jest ustawiona na 50%. |
| >SPEED 100 ↵ | Prędkość maksymalna jest ustawiona na 100%. |
| >SPEED 200 ↵ | Prędkość maksymalna jest ustawiona na 100%. |

## EXECUTE program name, execution cycles, step number (nazwa programu, cykle wykonania, numer kroku)

**Funkcja**

Wykonuje program robota.

**Parametr**

1. Program Name (nazwa programu)

Wybiera program, który ma być wykonywany. Jeśli nie określono, wybierany jest program ostatnio wykonany (przy pomocy polecenia EXECUTE, PRIME, STEP lub MSTEP).

2. Execution cycles - cykle wykonywania

Określa, ile razy ma być wykonywany program. Jeśli nie określono, przyjmowane jest 1. Aby wykonywać program w sposób ciągły, wprowadź liczbę ujemną (–1).

3. Step number - numer kroku

Określa krok, od którego zaczyna się wykonywanie. Jeśli nie określono, wykonywanie zaczyna się od pierwszego wykonywalnego kroku programu. Jeśli program jest wykonywany więcej niż jeden raz, od drugiego cyklu, program jest wykonywany od pierwszego kroku.

**Objaśnienie**

Wykonuje określony program robota od określonego kroku. Wykonywanie jest powtarzane określoną liczbę cykli.

**⚠ UWAGA**

**Jeśli używane jest to polecenie, poniższe warunki są konfigurowane automatycznie:**

**SPEED 100 ALWAYS**

**ACCURACY 1 ALWAYS**

**Koniec cyklu jest zaznaczany instrukcją STOP lub ostatnim krokiem programu.**

**Przykład**

>EXECUTE test,-1↵ Wykonuje program o nazwie “test” w sposób ciągły. (Program jest wykonywany ciągle, aż do wprowadzenia np. polecenia HALT lub do momentu wystąpienia błędu.)

>EXECUTE ↵ Wykonuje ostatnio wykonywany program. (Tylko jeden cykl).

## STEP program name, execution cycles, step number (nazwa programu, cykle wykonania, numer kroku)
## MSTEP program name, execution cycles, step number (nazwa programu, cykle wykonania, numer kroku)

**Funkcja**

Wykonuje jeden krok programu robota.

**Parametr**

1. Program Name (nazwa programu)

Wybiera program, który ma być wykonywany. Jeśli nie określono, wybierany jest aktualnie zawieszony lub ostatnio wykonany program.

2. Execution cycles - cykle wykonywania

Określa, ile razy ma być wykonywany program. Jeśli nie określono, przyjmowane jest 1.

3. Step number - numer kroku

Określa krok, od którego zaczyna się wykonywanie. Jeśli nie określono, wykonywanie zaczyna się od pierwszego wykonywalnego kroku programu. Jeśli nie określono żadnego parametru, wybierany jest krok następujący po ostatnio wykonanym.

**Objaśnienie**

Polecenie to może być wykonywane bez parametrów, jedynie w następujących warunkach:

1. po instrukcji PAUSE,
2. jeśli program był zatrzymany z innych powodów niż błąd,
3. gdy poprzednia instrukcja programu była wykonana przy pomocy polecenia STEP.

Polecenie MSTEP wykonuje jeden segment ruchu (tj. jedną instrukcję ruchu i kroki przed kolejną instrukcją ruchu). Polecenie STEP wykonuje tylko jeden krok programu (robot nie musi wykonywać ruchu).

**Przykład**

| | |
|---|---|
| >STEP assembly,,23 ↵ | Wykonuje wyłącznie krok 23 programu “assembly”. Ponowne wprowadzenie STEP bez parametru, bezpośrednio potem, powoduje wykonanie kroku 24. |

## ABORT

**Funkcja**

Zatrzymuje wykonywanie określonego programu robota.

**Objaśnienie**

Zatrzymuje wykonywanie programu robota po ukończeniu bieżącego kroku. Jeśli robot jest w ruchu, wykonywanie jest zatrzymywane po ukończeniu ruchu. Wykonywanie programu jest podejmowane ponownie przy pomocy polecenia CONTINUE.

**[UWAGA]**

W systemie AS ruchy robota i aktualnie wykonywane kroki nie zawsze są identyczne. Z tego względu, jeśli przetwarzanie kroków jest szybsze niż ruch robota, robot może wykonywać przed zatrzymaniem, po bieżącym ruchu o jeden ruch więcej.

## HOLD

**Funkcja**

Natychmiast zatrzymuje wykonywanie określonego programu robota.

**Objaśnienie**

Ruch robota jest natychmiast zatrzymywany. W przeciwieństwie do zastosowania wyłącznika EMERGENCY STOP, nie następuje wyłączenie zasilania silnika. Polecenie to daje taki sam efekt, gdy przełącznik HOLD/RUN jest ustawiony w pozycji HOLD. Wykonywanie programu jest podejmowane ponownie przy pomocy polecenia CONTINUE.

## CONTINUE NEXT

**Funkcja**

Podejmuje wykonywanie programu zatrzymanego instrukcją PAUSE, ABORT lub poleceniem HOLD, bądź też z powodu błędu. Polecenie to może być także używane do uruchomienia programów przygotowanych poleceniami PRIME, STEP lub MSTEP.

**Parametr**

NEXT

Jeśli nie wprowadzono NEXT, wykonywanie jest podejmowane od kroku, w którym je zatrzymano. Jeśli wprowadzono NEXT, wykonanie jest podejmowane od kroku następującego po kroku, w którym je zatrzymano.

**Objaśnienie**

Wpływ, jaki na restart programu posiada słowo kluczowe NEXT różni się w zależności od tego, w jaki sposób nastąpiło zatrzymanie programu

1. Program zatrzymany w czasie wykonania kroku lub ruchu:
   CONTINUE restartuje program i powoduje ponowne wykonanie przerwanego kroku.
   CONTINUE NEXT restartuje od kroku następującego po kroku, w którym program został zatrzymany.
2. Wykonywanie programu zatrzymane po ukończeniu kroku lub ruchu:
   CONTINUE i CONTINUE NEXT restartują program od tego kroku natychmiast po ukończeniu kroku, niezależnie od NEXT.
3. Program zawieszony instrukcją WAIT, SWAIT lub TWAIT:

CONTINUE NEXT przeskakuje instrukcje i wznawia wykonywanie od następnego kroku.

**[UWAGA]**

Wykonywanie programu nie może być ponownie podjęte przy pomocy polecenia CONTINUE, gdy:

- Program zakończył się prawidłowo.
- Program został zatrzymany przy pomocy instrukcji HALT.
- Użyto polecenia KILL.

## 5.5 POLECENIA DOTYCZĄCE INFORMACJI O POZYCJI ROBOTA

| | |
|---|---|
| HERE | Definiuje zmienną pozycji jako pozycji bieżącej. |
| POINT | Definiuje zmienną pozycji. |
| POINT/X | Określa wartość X zmiennej pozycji. |
| POINT/Y | Określa wartość Y zmiennej pozycji. |
| POINT/Z | Określa wartość Z zmiennej pozycji. |
| POINT/OAT | Określa wartości OAT zmiennej pozycji. |
| POINT/O | Określa wartość O zmiennej pozycji. |
| POINT/A | Określa wartość A zmiennej pozycji. |
| POINT/T | Określa wartość T zmiennej pozycji. |
| POINT/7 | Określa wartość siódmej osi dla zmiennej pozycji. |

## HERE pose variable name (nazwa zmiennej pozycji)

**Funkcja**

Definiuje zmienną pozycji (lokalizacji) jako pozycja bieżąca. Pozycja ta może być wyrażana w wartościach przekształcenia, przesunięcia osi lub w złożonych wartościach przekształcenia.

**Parametr**

Nazwa zmiennej pozycji (lokalizacji)

Może być określana w wartościach przekształcenia, przesunięcia osi lub w złożonych wartościach przekształcenia.

**Objaśnienie**

Pozycja ta może być wyrażana w wartościach przekształcenia, przesunięcia osi lub w złożonych wartościach przekształcenia.

**[UWAGA]**

Określana jest tylko zmienna położona najdalej na prawo w złożonych wartościach przekształcenia. (Patrz przykład poniżej). Jeśli inne zmienne używane w wartościach złożonych nie są określone, polecenie to powoduje wystąpienie błędu.

Wartości zmiennej są wyświetlane na terminalu z następującymi, a po nich pojawia się komunikat “Change?” (zmienić?). Wartości mogą być zmienione poprzez wprowadzenie wartości oddzielanych przecinkiem. Wartość, która nie jest zmieniana może być pominięta. Aby zakończyć edycję wartości, po komunikacie “Change?” naciśnij ↵.

Jeśli zmienna jest określana w wartościach przesunięcia osi (nazwa zmiennej zaczynająca się symbolem #), wyświetlane są wartości osi pozycji bieżącego. Jeśli zmienna jest wartością przekształcenia, wyświetlane są wartości XYZOAT. Wartości XYZ określają położenie punktu początkowego układu współrzędnych narzędzia, w odniesieniu do współrzędnych globalnych. Wartości kątów OAT określają pozycję układu współrzędnych narzędzia.

**Przykład**

| | |
|---|---|
| >HERE #pick ↵ | Określa pozycję bieżącą robota jako “#pick” (wartości przesunięcia osi). |
| >HERE place ↵ | Określa pozycję bieżącą robota jako “place” (wartości przekształcenia). |
| HERE plate+object ↵ | Określa bieżące ustawienie robota jako “object ” w odniesieniu do pozycji “plate”(złożone wartości przekształcenia). Jeśli “plate” jest niezdefiniowane, występuje błąd. |

| | | |
|---|---|---|
| **POINT** | **pose variable name** | **= pose values, joint displacement values (nazwa zmiennej pozycji = wartości pozycji, wartości przesunięcia osi)** |

**Funkcja**

Przydziela informacje o ustawieniu po prawej od “=” do zmiennej pozycji po lewej od “=”.

**Parametr**

1. Nazwa zmiennej pozycji

Określa nazwę zmiennej informacji o ustawieniu, która ma być określona (może być określona w wartościach przesunięcia osi, przekształcenia lub w złożonych wartościach przekształcenia).

2. Wartości pozycji

Jeśli nie określono, znak “=” jest pomijany.

3. Wartości przesunięcia osi

Parametr ten musi być określony, jeśli nazwa zmiennej pozycji po lewej jest określona przez wartości przesunięcia osi, a wartości informacji o ustawieniu po prawej są wartościami przekształcenia (jeśli parametr po lewej nie jest określony przez wartości przesunięcia osi, omawiany parametr nie wymaga pozycji). Określone tutaj wartości przesunięcia osi konfigurują ustawienie robota. Jeśli nie określono, do zdefiniowania zmiennej pozycji używana jest bieżąca konfiguracja.

**Objaśnienie**

Przydziela wartości pozycji określone po prawej od zmiennej pozycji. Jeśli nie określono wartości pozycji, na terminalu wyświetlane są zdefiniowane już, edytowalne wartości dla określonej pozycji. Jeśli zmienna pozycji jest niezdefiniowana, wyświetlane będą wartości: 0, 0, 0, 0, 0, 0.

Po wykonaniu POINT wyświetlane są wartości pozycji, a po nich pojawia się komunikat “Change?” (zmienić?) oraz znak zachęty. Wartości te mogą być następnie edytowane. Zakończ, naciskając samo ↵ po znaku zachęty.

Jeśli zmienna jest określona przez wartości przesunięcia osi, wyświetlane są wartości osi. Jeśli zmienna jest zdefiniowana wartością przekształcenia, wyświetlane są wartości XYZOAT. Wartości XYZ określają położenie punktu początkowego układu współrzędnych narzędzia, w odniesieniu do współrzędnych globalnych. Wartości kątów OAT określają pozycję układu współrzędnych narzędzia. Jeśli zmienna jest wyrażana w złożonych wartościach przekształcenia, określana jest zmienna położona najdalej na prawo w złożonych wartościach przekształcenia. Jeśli inne zmienne używane w wartości złożonej nie są określone, polecenie to powoduje wystąpienie błędu.

[**UWAGA**]

Jeśli typy wartości po prawej i lewej stronie znaku ”=” różnią się omawiane polecenie działa w następujący sposób:

1. POINT transformation values=joint displacement values (wartości przekształcenia=wartości przesunięcia osi)
   Wartości przesunięcia osi po prawej są przekształcane w wartości przekształcenia i przydzielane zmiennej po lewej.
2. POINT joint displacement values=transformation values, joint displacement values (wartości przesunięcia osi=wartości przekształcenia, wartości przesunięcia osi)
   Wartości przekształcenia po prawej są przekształcane w wartości przesunięcia osi i przydzielane zmiennej po lewej. Jeśli wartości przesunięcia osi po prawej są określone, wartości przekształcenia są przekształcane przez robota przyjmującego konfigurację określonych wartości przesunięcia osi. Jeśli nie określono, wartości przekształcenia są przekształcane przez robota w jego bieżącej konfiguracji.

Podczas określania wartości, można wprowadzić maksymalnie dziewięć cyfr. Dokładność wpisów zawierających więcej niż dziewięć cyfr nie jest gwarantowana.

**Przykład**

```
>POINT   #park                          Wyświetla wartość pozycji “#park”.
                                        (jeśli nie zdefiniowano, wyświetlane jest 0,0,0,0,0,0)
JT1    JT2      JT3      JT4    JT5    JT6
10.000   15.000   20.000   30.000   50.000   40.000
Change?(If not, hit RETURN only) (Zmienić? (Jeśli nie, naciśnij klawisz RETURN.))
,,,-15
JT1    JT2      JT3      JT4    JT5    JT6
10.000   15.000   20.000  -15.000   50.000   40.000
Change?(If not, hit RETURN only) (Zmienić? (Jeśli nie, naciśnij klawisz RETURN.)) ↵
>POINT   pick1=pick ↵                   Przydziela wartości przekształcenia “pick” do
                                        wartości przekształcenia “pick1” i wyświetla wartości,
                                        które muszą być poprawione.

>POINT   pos0=#pos0 ↵                   Przekształca wartości przesunięcia osi “#pos0” na
                                        wartości przekształcenia i przydziela je do “pos0”.

>POINT   #pos1=pos1,#pos2 ↵             Przekształca wartości przekształcenia “pos1” na
                                        wartości przesunięcia osi, używając konfiguracji
                                        robota zadanej przez #pos2 i przydziela wartości do
                                        “#pos1”.
```

**POINT/ X transformation variable name transformation values (nazwa zmiennej przekształcenia = wartości przekształcenia)**
**POINT/ Y transformation variable name transformation values**
**POINT/ Z transformation variable name transformation values**
**POINT/ OAT transformation variable name transformation values**
**POINT/ O transformation variable name transformation values**
**POINT / A transformation variable name transformation values**
**POINT / T transformation variable name transformation values**
**POINT/ 7 transformation variable name transformation values**

**Funkcja**

Przydziela elementy wartości przekształcenia określone po prawej stronie znaku “=” do odpowiednich elementów wartości przekształcenia po lewej od “=”. Wartości te będą wyświetlane do edycji, na terminalu.

**Parametr**

1. Nazwa zmiennej

Wybiera nazwę zmiennej przekształcenia, która ma być zdefiniowana (wartości przekształcenia lub złożone wartości przekształcenia).

2. Transformation values - wartości przekształcenia

Jeśli nie określono, znak “=” jest pomijany.

**Objaśnienie**

Przydziela wyłącznie określone elementy (X, Y, Z, O, A, T) wartości przekształcenia. Po wykonaniu omawianego polecenia wyświetlane są wartości każdego elementu, a po nich pojawia się komunikat “Change?” (zmienić?) oraz znak zachęty. Wartości te mogą być następnie edytowane. Zakończ naciskając klawisz ↵ po znaku zachęty.

**Przykład**

Poniższe polecenie przydziela wartości kąta OAT a1 do a2. Wartości przekształcenia a1 i a2 są następujące:

a1 = (1000, 2000, 3000, 10, 15, 30)

a2 = niezdefiniowano

```
>POINT/OAT a2 = a1
X[mm]      Y[mm]      Z[mm]      O[deg]     A[deg]     T[deg]
  0.         0.         0.        10.        15.        30.
Change?(If not, hit RETURN only) (Zmienić? (Jeśli nie, naciśnij klawisz RETURN.)) ↵
```

## 5.6 POLECENIA STERUJĄCE SYSTEMEM

| | |
|---|---|
| WHERE | Wyświetla bieżące ustawienia robota. |
| IO | Wyświetla status sygnałów binarnych. |
| FREE | Wyświetla informację na temat wielkości wolnej pamięci. |
| TIME | Wyświetla i ustawia bieżący czas i datę. |
| ULIMIT | Ustawia górną granicę ruchu robota. |
| LLIMIT | Ustawia dolną granicę ruchu robota. |
| BASE | Zmienia wartości przekształcenia podstawy. |
| TOOL | Określa wartości przekształcenia narzędzia |
| SETHOME | Konfiguruje pozycję domową. |
| SET2HOME | Konfiguruje pozycję domową nr 2. |

## WHERE display mode (tryb wyświetlacza)

**Funkcja**

Wyświetla bieżące ustawienie robota.

**Parametr**

Display mode (tryb wyświetlacza)

Określa tryb wyświetlania danych. Istnieje 16 trybów, zaprezentowanych poniżej (tryby od 7 do 16 stanowią opcję). Jeśli nie określono trybu, wyświetlane są wartości przekształcenia punktu centralnego narzędzia (TCP) współrzędnych globalnych oraz kąty osi (JT1, JT2, …, JT3). Tryb wyświetlacza nie zmienia się do momentu ponownego naciśnięcia ↵.

WHERE …… Wyświetla bieżące ustawienie robota w wartościach przekształcenia we współrzędnych globalnych oraz kąty osi (JT1, JT2, …, JT3).

WHERE 1 …… Wyświetla bieżące ustawienia przy pomocy kątów osi.

WHERE 2 …… Wyświetla bieżące ustawienie XYZOAT we współrzędnych globalnych (mm, st.).

WHERE 3 …… Wyświetla bieżące zlecone wartości (st.).

WHERE 4 …… Wyświetla odchylenia od zleconych wartości (bit).

WHERE 5 …… Wyświetla wartości enkodera każdej osi (bit).

WHERE 6 …… Wyświetla prędkość każdej osi (st./s).

WHERE 7 …… Wyświetla bieżące ustawienia robota, w tym osi zewnętrznej. (Opcja)

WHERE 8 …… Wyświetla bieżące ustawienie w układzie współrzędnych nieruchomego przedmiotu obrabianego. (Opcja)

WHERE 9 …… Wyświetla zlecone wartości każdej osi dla wartości przekształcenia.

WHERE 10 …… Wyświetla prąd silnika.

WHERE 11 …… Wyświetla prędkość silnika.

WHERE 12 …… Wyświetla bieżące wartości przekształcenia wyrażone we współrzędnych globalnych innego robota. (Opcja)

WHERE 13 …… Wyświetla bieżące wartości przekształcenia wyrażone we współrzędnych narzędzia innego robota. (Opcja)

WHERE 14 …… Wyświetla zlecone wartości prądu silnika.

WHERE 15 …… Wyświetla oryginalne dane enkodera.

WHERE 16 …… Wyświetla prędkość punktu centralnego narzędzia (TCP).

**Przykład** >WHERE ↵

```
JT1     JT2       JT3       JT4     JT5     JT6
9.999   0.000     0.000     0.000   0.000   0.000
X[mm]     Y[mm]     Z[mm]     O[deg]    A[deg]    T[deg]
15.627  88.633  930.000   -9.999    0.000     0.000
```

## IO/E signal number (numer sygnału)

**Funkcja**

Wyświetla bieżący status wszystkich zewnętrznych i wewnętrznych sygnałów we/wy.

**Parametr**

Numer sygnału

1……Wyświetla 1–32, 1001–1032, 2001–2032

2……Wyświetla 33–64, 1033–1064, 2033–2064

3……Wyświetla 65–96, 1065–1096, 2065–2096

4……Wyświetla 97–128, 1097–1128, 2097–2128

Jeśli nie określono……Wyświetla 1–32, 1001–1032, 2001–2032

**Objaśnienie**

Jeśli parametr systemowy DISPIO_01 znajduje się w pozycji OFF, dla sygnałów będących w pozycji ON wyświetlane będzie “o”, natomiast dla sygnałów będących w pozycji OFF odpowiednie jest “x”. Sygnały dedykowane są wyświetlane wielkimi literami (“O” i “X”). Jeśli parametr systemowy DISPIO_01 znajduje się w pozycji ON, dla sygnałów będących w pozycji ON wyświetlane będzie “1”, natomiast dla sygnałów będących w pozycji OFF odpowiednio “0”. Dla zewnętrznych sygnałów we/wy, które nie są zainstalowane, wyświetlane jest “-”.

Jeśli wraz z poleceniem wprowadzane jest “/E” numery sygnału o wartości 3001 i wyższych są wyświetlane wraz z sygnałami o numerach 1–, 1001–, 2001–. (Opcja)

Wyświetlacz jest aktualizowany w sposób ciągły, aż do momentu zakończenia wyświetlania przy pomocy klawisza ↵.

(Patrz 7.0 DISPIO_01 parametr systemowy)

**Przykład**

Gdy DISPIO_01 jest on w pozycji OFF.

```
>IO ↵
  32 -    1   xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxo  xxxo
1032 - 1001   xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  oxxx
2032 - 2001   xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx  xxxx
>
>IO/E ↵
```

```
  32 -      1     xxxx    xxxx    xxxx    xxXX    xxxx    XXXX   XXXO   XXXO
1032 -     1001    xxxx    xxxx    xxxx    xxXX    xxxx    XXXX   XXXO   XXXO
2032 - 2001    xxxx    xxxx    xxxx    xxxx    xxxx    xxxx    xxxx    xxxx
3032 - 3001    xxxx    xxxx    xxxx    xxxx    xxxx    xxxx    xxxx    xxxx
>
```

Gdy DISPIO_01 jest on w pozycji ON

```
>IO ↵
  32 -      1     0000    0000    0000    0000    0000    0000    0001    0001
1032 - 1001    0000    0000    0000    0000    0000    0000    0000    1000
2032 - 2001    0000    0000    0000    0000    0000    0000    0000    0000
```

| | |
|---|---|
| **ULIMIT** | **Joint displacement values (wartości przesunięcia osi)** |
| **LLIMIT** | **Joint displacement values (wartości przesunięcia osi)** |

**Funkcja**

Ustawia i wyświetla górną/dolną granicę zakresu ruchu robota.

**Parametr**

Wartości przesunięcia osi

Ustawia ograniczenia programowe (górne lub dolne) w wartościach przesunięcia osi. Jeśli nie określono tego parametru, wyświetlana jest bieżąca wartość.

**Objaśnienie**

Jeśli nie określono omawianego parametru, wyświetlane są wprowadzone wartości, a po nich pojawia się komunikat “Change?” (zmienić?). Po tym komunikacie wprowadź żądane wartości, jak w przypadku polecenia POINT. Aby zakończyć polecenie, naciśnij klawisz ↵.

Jeśli nie określono omawianego parametru, wyświetlane są aktualnie ustawione wartości limitu, a po nich pojawia się komunikat “Change?” (zmienić?).

**Przykład**

```
>ULIMIT                                  Wyświetla bieżące ustawienia.
                JT1     JT2     JT3      JT4      JT5      JT6
  Maximum   120.00   60.00   60.00    190.00   115.00   270.00   (Maksymalny dopuszczalny limit)
  Current    30.00   15.00   25.00   -40.00    60.00    15.00   (Bieżące ustawienia)
  Change? (zmienić?)  ( If not, hit RETURN only ) (Jeśli nie, naciśnij klawisz RETURN)
>110,50
                JT1     JT2      JT3      JT4     JT5     JT6
  Maximum   120.00   60.00   60.00   190.00   115.00   270.00
  Current   110,00    50,00   25.00   -40.00    60.00    15.00
  Change? (zmienić?)  ( If not , hit RETURN only ) (Jeśli nie, naciśnij klawisz RETURN.)↵

>ULIMIT #upper ↵                         Konfiguruje górne ograniczenia programowe dla ustawienia
        zdefiniowanego jako
                                         “#upper”.
>LLIMIT #low ↵                           Konfiguruje dolne ograniczenia programowe dla ustawienia
zdefiniowanego jako “#low”.
```

## BASE transformation values (wartości przekształcenia)

**Funkcja**

Definiuje wartości przekształcenia podstawy, które określają relację ustawienia pomiędzy współrzędnymi globalnymi, a współrzędnymi wyjściowymi podstawy.

**Parametr**

Wartości przekształcenia (lub złożone wartości przekształcenia)
Określa nowe współrzędne globalne. Omawiane wartości przekształcenia określają ustawienie współrzędnych globalnych w odniesieniu do współrzędnych wyjściowych podstawy, wyrażone we współrzędnych wyjściowych podstawy. Jeśli nie wprowadzono żadnych wartości, wyświetlane są bieżące wartości przekształcenia podstawy.

**Objaśnienie**

Jeśli dla parametru określono “NULL”, wartości przekształcenia podstawy są skonfigurowane jako “null base” (XYZOAT=0, 0, 0, 0, 0, 0,). Podczas inicjacji systemu, wartości przekształcenia podstawy są automatycznie ustawiane jako null base (wartości wyjściowe podstawy).

Po ustawieniu nowych wartości przekształcenia podstawy, wyświetlane są wartości ( XYZOAT )

oraz komunikat “Change?” (zmienić?). Aby zmienić omawiane wartości, wprowadź nowe, oddzielając je przecinkami i naciśnij ↵. Jeśli nie określono żadnego parametru, wyświetlane są bieżące wartości.

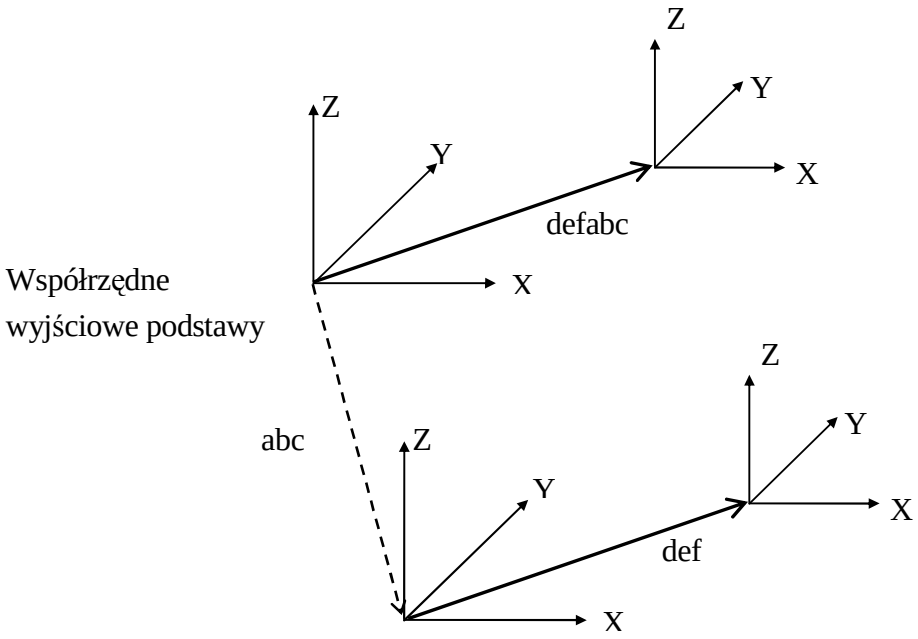
Jeśli robot przesuwa się do ustawienia zdefiniowanego przy pomocy wartości przekształcenia lub jest obsługiwany ręcznie w trybie podstawowym, system automatycznie oblicza ustawienie robota, biorąc pod uwagę określone tutaj wartości przekształcenia podstawy.
Gdy jako parametr używana jest zmienna ustawienia i jeśli zmienna ustawienia jest określana ponownie, pamiętaj iż musisz ponownie zdefiniować także wartości przekształcenia podstawy przy pomocy polecenia BASE oraz jako parametr nowe ustawienie. Zmiana dokonana w zakresie zmiennej ustawienia zostanie odzwierciedlona w wartości przekształcenia podstawy.

Polecenie BASE nie wywiera wpływu na ustawienia określone przy pomocy wartości przesunięcia osi.

[**UWAGA**]

| | |
|---|---|
| BASE abc ↵ | BASE NULL ↵ |
| DO JMOVE def ↵ | DO JMOVE def ↵ |

Nawet jeśli w obu powyższych przykładach informacje o ustawieniu “def” są identyczne, cel robota będzie się różnił zgodnie z wartościami przekształcenia podstawy. Patrz wykres poniżej.




**Przykład**

>BASE ↵ Wyświetla bieżące wartości przekształcenia podstawy.

| X[mm] | Y[mm] | Z[mm] | O[deg] | A[deg] | T[deg] |
|---|---|---|---|---|---|
| -300. | 0. | 0. | 0. | 0. | 0. |

Change? (zmienić?) (If not , hit RETURN only) (Jeśli nie, naciśnij klawisz RETURN.)↵

>BASE NULL ↵ Zmienia wartości przekształcenia podstawy na null base (wartości wyjściowe podstawy).
(X, Y, Z, O, A, T)=(0, 0, 0, 0, 0, 0)

>BASE abc ↵ Zmienia ustawienie współrzędnych globalnych na ustawienie określone zmienną ustawienia “abc”.

## TOOL transformation values (wartości przekształcenia)

**Funkcja**

Definiuje wartości przekształcenia narzędzia, które określają relację ustawienia pomiędzy współrzędnymi narzędzia, a współrzędnymi wyjściowymi narzędzia.

**Parametr**

Wartości przekształcenia (lub złożone wartości przekształcenia)

Określa nowe współrzędne narzędzia. Omawiane wartości przekształcenia określają ustawienie współrzędnych narzędzia w odniesieniu do współrzędnych wyjściowych narzędzia, wyrażone we współrzędnych wyjściowych narzędzia. Jeśli nie wprowadzono żadnych wartości, wyświetlane są bieżące wartości przekształcenia narzędzia.

**Objaśnienie**

Jeśli dla parametru określono “NULL”, wartości przekształcenia narzędzia są skonfigurowane jako “null base” (XYZOAT=0, 0, 0, 0, 0, 0,). Współrzędne wyjściowe narzędzia posiadają punkt początkowy położony w środku powierzchni kołnierza mocującego narzędzie, a osie są równoległe do ostatniej osi robota. Podczas inicjacji systemu, wartości przekształcenia narzędzia są automatycznie ustawiane jako null tool (współrzędne wyjściowe narzędzia).

Po ustawieniu nowych wartości przekształcenia narzędzia, wyświetlane są wartości ( XYZOAT )

oraz komunikat “Change?” (zmienić?). Aby zmienić omawiane wartości, wprowadź nowe, odzielając je przecinkami i naciśnij ENTER. Jeśli nie określono żadnego parametru, wyświetlane są bieżące wartości.

Jeśli robot przesuwa się do ustawienia zdefiniowanego przy pomocy wartości przekształcenia lub jest obsługiwany ręcznie w trybie podstawowym lub w trybie pracy wg współrzędnych narzędzia, system automatycznie oblicza ustawienie robota, biorąc pod uwagę określone tutaj wartości przekształcenia narzędzia.

Gdy jako parametr używana jest zmienna ustawienia i jeśli zmienna ustawienia jest określana ponownie, pamiętaj iż musisz ponownie zdefiniować także wartości przekształcenia narzędzia, przy pomocy polecenia TOOL oraz jako parametr nowe ustawienie. Zmiana dokonana w zakresie zmiennej ustawienia zostanie odzwierciedlona w wartości przekształcenia narzędzia. Patrz, 11.4 Przekształcanie narzędzia.

**Przykład**

| | |
|---|---|
| >TOOL grip ↵ | Zmienia ustawienie współrzędnych narzędzia na ustawienie określone zmienną ustawienia “grip”. |
| >TOOL NULL ↵ | Zmienia wartości przekształcenia narzędzia na wartości |

wyjściowe narzędzia (null tool).

(X, Y, Z, O, A, T)=(0, 0, 0, 0, 0, 0)

| | |
|---|---|
| **SETHOME** | **accuracy, HERE** |
| **SET2HOME** | **accuracy, HERE** |

**Funkcja**

Konfiguruje i wyświetla pozycję wyjściową HOME.

**Parametr**

1. Accuracy - dokładność

Konfiguruje zakres dokładności (accuracy) pozycji HOME w milimetrach. Robot znajduje się w pozycji HOME, gdy znajduje się obok punktu HOME o określoną tu odległość. Jeśli nie określono,1 mm jest przyjmowany jako wartość domyślna.

2. HERE

Konfiguruje bieżące ustawienie jako HOME.

**Objaśnienie**

Jeśli nie określono żadnego parametru, wyświetlane są bieżące wartości, a po nich pojawia się komunikat “Change?” (zmienić?). Wprowadź żądane wartości i naciśnij klawisz ENTER. Jeśli nie wprowadzono żadnych zmian, naciśnij ENTER.

W systemie AS możliwe jest ustawienie dwóch pozycji HOME (HOME1 i HOME2). HOME 1 jest ustawiane przy pomocy polecenia SETHOME, HOME 2 przy pomocy polecenia SET2HOME.

**Przykład**

```
>SETHOME 2 ↵
   JT1   JT2     JT3     JT4     JT5     JT6   accuracy[mm]
   0.     0.      0.      0.      0.      0.        2.
   Change? (zmienić?)  ( If not , hit RETURN only ) (Jeśli nie, naciśnij klawisz RETURN.)
   ,90,-90
        JT1     JT2     JT3     JT4     JT5     JT6   accuracy[mm]
   0.    90.    -90.     0.      0.      0.        2.
   Change? (zmienić?)  ( If not , hit RETURN only ) (Jeśli nie, naciśnij klawisz RETURN.)↵
   >
```

Konfiguruje dokładność na wartości 2 mm i zmienia pozycję HOME, przez wprowadzenie nowych wartości.

```
>SETHOME  10,HERE ↵
```

Konfiguruje bieżące ustawienie jako pozycję HOME. Dokładność jest ustawiana na 10 mm; tj. sygnał dedykowany HOME będzie wysyłany, gdy robot osiągnie pozycję oddaloną od pozycji HOME o 10 mm.

## 5.7 POLECENIA SYGNAŁÓW BINARNYCH

Kontrolery robotów serii D używają dwóch typów sygnałów binarnych: zewnętrznych sygnałów we/wy pomiędzy robotem a urządzeniami zewnętrznymi i wewnętrznych sygnałów we/wy wykorzystywanych wewnątrz robota. Wewnętrzne sygnały we/wy są używane pomiędzy robotem a programami kontrolnymi procesu działającymi równolegle lub jako test sygnałów sprawdzający programy przed rzeczywistym podłączeniem urządzeń zewnętrznych.

Sygnały binarne są sterowane lub definiowane przy pomocy następujących poleceń.

| | |
|---|---|
| RESET | Wyłącza wszystkie zewnętrzne sygnały we/wy. |
| SIGNAL | Włącza (lub wyłącza) sygnały wyjściowe. |
| PULSE | Włącza sygnały wychodzące na określony czas. |
| DLYSIG | Włącza sygnały wychodzące po upływie określonego czasu. |

## RESET

**Funkcja**

Wyłącza wszystkie zewnętrzne sygnały wychodzące. Polecenie to nie ma wpływu na sygnały dedykowane, sygnały włączenia/wyłączenia narzędzia oraz na sygnały antynomii dla wielofunkcyjnych sygnałów wyjścia OX/sygnałów zewnętrznych WX.

Dzięki wykorzystaniu opcjonalnych ustawień, sygnały używane w ekranie interfejsu są niezależne od niniejszego polecenia. (Opcja)

**[UWAGA]**

Pamiętaj, iż polecenie to wyłącza wszystkie sygnały za wyjątkiem tych wspomnianych powyżej, nawet w trybie odtwarzania.

## SIGNAL signal number, …… (numer sygnału)

**Funkcja**

Włącza lub wyłącza określone zewnętrzne lub wewnętrzne sygnały we/wy.

**Parametr**

Numer sygnału

Wybiera numer zewnętrznego sygnału wychodzącego lub sygnału wewnętrznego. Liczba dodatnia włącza sygnał, ujemna wyłącza go.

**Objaśnienie**

Numer sygnału określa, czy jest on zewnętrzny, czy też wewnętrzny.

Dopuszczalne numery sygnału

| | |
|---|---|
| External output signal (zewnętrzny sygnał wyjścia) | 1 – actual number of signals (rzeczywisty numer sygnału) |
| Internal signal (sygnał wewnętrzny) | 2001–2256 |
| External input signal (zewnętrzny sygnał wejścia) | Nie może być określony |

Jeśli numer sygnału jest liczbą dodatnią, sygnał jest włączany; jeśli jest liczbą ujemną, sygnał jest wyłączany. Jeśli wprowadzane jest “0”, sygnały wychodzące nie są zmieniane. Jeśli wybierzesz numer sygnału, który jest już ustawiony jako sygnał dedykowany, wystąpi błąd.

**Przykład**

>SIGNAL -1,4,2010 ↵ External output signal 1 jest w pozycji OFF, 4 jest w pozycji ON, Internal signal 2010 jest w pozycji ON.

>SIGNAL -reset,4 ↵ Jeśli zmienna “reset” jest liczbą dodatnią, wyłączany jest sygnał wychodzący określany przez tę wartość, a włączany jest sygnał wychodzący (output signal) 4.

## PULSE signal number, time (numer sygnału, czas)

**Funkcja**

Włącza określony sygnał na określony czas.

**Parametr**

1. Numer sygnału

Wybiera numer zewnętrznego sygnału wychodzącego lub sygnału wewnętrznego (tylko wartości dodatnie). Jeśli wybierzesz numer sygnału, który jest już używany jako sygnał dedykowany, wystąpi błąd.

Dopuszczalne numery sygnału

| | |
|---|---|
| External output signal (zewnętrzny sygnał wyjścia) | 1 – rzeczywisty numer sygnału lub 64 (mniejsza wartość) |
| Internal signal (sygnał wewnętrzny) | 2001–2256 |

2. Time - czas

Konfiguruje czasokres wysyłania sygnału (w sekundach). Jeśli nie określono, parametr ten jest automatycznie ustawiany na 0,2 sekundy.

## DLYSIG signal number, time (numer sygnału, czas)

**Funkcja**

Wysyła określone sygnały, po upływie określonego czasu.

**Parametr**

1. Numer sygnału

Wybiera numer zewnętrznego sygnału wychodzącego lub sygnału wewnętrznego. Jeśli numer sygnału jest liczbą dodatnią, sygnał jest włączany; jeśli jest liczbą ujemną, sygnał jest wyłączany. Jeśli wybierzesz numer sygnału, który jest już używany przez sygnał dedykowany, wystąpi błąd.

Dopuszczalne numery sygnału

| | |
|---|---|
| External output signal (zewnętrzny sygnał wyjścia) | 1 – rzeczywisty numer sygnału lub 64 (mniejsza wartość) |
| Internal signal (sygnał | 2001–2256 |

| | |
|---|---|
| wewnętrzny) | |

2. Time - czas

Określa w sekundach czas wstrzymania wysyłania sygnału.

## 5.8 POLECENIA DOTYCZĄCE WYŚWIETLANIA KOMUNIKATÓW

| | |
|---|---|
| PRINT | Wyświetla dane. |
| TYPE | Wyświetla dane. |

| | |
|---|---|
| **PRINT** | **device number: print data, …….** (numer urządzenia: dane druku) |
| **TYPE** | **device number: print data, …….** (numer urządzenia: dane druku) |

**Funkcja**

Wyświetla na terminalu dane druku określonego parametru.

**Parametr**

1. Device number (numer urządzenia)

Wybiera urządzenie, którego dane mają być wyświetlane:

1: Komputer osobisty

2: Teach pendant - programator ręczny

Jeśli nie określono, dane są wyświetlane na aktualnie wybranym urządzeniu.

2. Print data (dane druku)

Wybierz jedną lub więcej z poniższych opcji. Wybierając więcej niż jedną, oddzielaj dane przecinkiem.

(1) character string (ciąg znaków) np. “count =”

(2) real value expressions (wyrażenia wartości rzeczywistych) (wartość jest obliczana i wyświetlana) np. count

(3) Format information (informacja dotycząca formatu) (kontroluje format wychodzącego komunikatu) np. /D, /S

Jeśli nie określono parametru, wyświetlana jest pusta linia.

**Objaśnienie**

Jeśli dla numeru urządzenia wprowadzono “2”, ekran programatora ręcznego zmienia się automatycznie na ekran klawiatury. Naciśnij <NEXT PAGE>, aby powrócić do regularnego ekranu.

Poniżej wyszczególniono kody używane do określania formatu wyjścia (output format) wyrażeń liczbowych (numeric expressions). Aż do momentu określenia odmiennego kodu, wykorzystywany jest ten sam format. W przypadku każdego formatu, jeśli wartość jest zbyt duża do wyświetlenia na danej szerokości, przestrzeń zostanie wypełniona gwiazdkami (∗). W takim przypadku zmień liczbę znaków na taką, która może zostać wyświetlona. Maksymalna liczba

znaków, które mogą być wyświetlone w jednej linii wynosi 128. Aby w jednej linii wyświetlić więcej niż 128 znaków, użyj kodu /S objaśnionego na następnej stronie.

**[UWAGA]**

Jeśli przełącznik MESSAGES jest w pozycji OFF, na ekranie terminala nie zostaną wyświetlone żadne komunikaty.

Kody specyfikacji formatu

| | |
|---|---|
| /D | Używa domyślnego formatu. Jest równoznaczne z określeniem formatu jako /G15.8, z tą różnicą, iż usuwane są zera występujące za wartościami liczbowymi, a pomiędzy nimi pozostawiana jest zawsze tylko jedna spacja. |
| /Em.n | Wyświetla wartości liczbowe w notacji naukowej np. -1.234 02. “m” określa ogólną liczbę znaków wyświetlanych na terminalu, a “n” liczbę miejsc dziesiętnych. “m” powinno być co najmniej pięciokrotnie większe od n. |
| /Fm.n | Wyświetla wartości liczbowe w zapisie stałopozycyjnym np. -1.234). “m” określa ogólną liczbę znaków wyświetlanych na terminalu, a “n” liczbę cyfr części dziesiętnych. |
| /Gm.n | Jeśli wartość jest większa, niż 0,01 i może być wyświetlona w formacie Fm.n z m cyframi, wartość jest wyświetlana w tym formacie. W innym przypadku, wartość jest wyświetlana w formacie Gm.n. |
| /Hn | Wyświetla wartości jako liczby notacji szesnastkowej w polu cyfr n. |
| /In | Wyświetla wartości jako liczby dziesiętne w polu n cyfr. |

Poniższe parametry są używane do wstawiania określonych znaków pomiędzy ciągi znaków.

| | |
|---|---|
| /Cn | Wstawia przesuw o wiersz w miejscu, gdzie wprowadzany jest omawiany kod, przed lub też za danymi druku. Jeśli kod ten jest umieszczany wewnątrz danych druku, wstawianych jest n-1pustych linii. |
| /S | Linia nie jest wstawiana. |
| /Xn | Wstawia n spacji. |
| /Jn | Wyświetla wartość jako liczbę notacji szesnastkowej w polu n cyfr. W miejsce pustych znaków wstawiane są zera. (Opcja) |

/Kn Wyświetla wartość jako liczbę dziesiętną w polu n cyfr. W miejsce pustych znaków wstawiane są zera. (Opcja)

/L Identycznie, jak w przypadku /D, z tym wyjątkiem, że w przypadku tego kodu usuwane są wszystkie spacje. (Opcja)

**Przykład**

W niniejszym przykładzie wartość zmiennej rzeczywistej “i” równa się 5 a piąty element zmiennej tablicowej “point” 12,66666.

```
>PRINT   "point", i, "=" , /F5.2, point [i] ↵
```

```
point 5 = 12.67
```

Wyświetlacz powinien wyglądać w następujący sposób.

```
point 5 = *
```

Jeśli wartość point[5] równa się 1000 (1000,00), wyświetlacz powinien wyglądać w następujący sposób. Wartość jest zbyt duża, aby mogła być wyświetlona (tj. liczba cyfr jest większa niż 5).

W poniższym przykładzie kod /S jest używany do wyświetlania danych bez zmiany następujących po nich linii.

```
>PRINT “ABC”
>PRINT/S, “DEF”
>PRINT “GHI” ↵
```

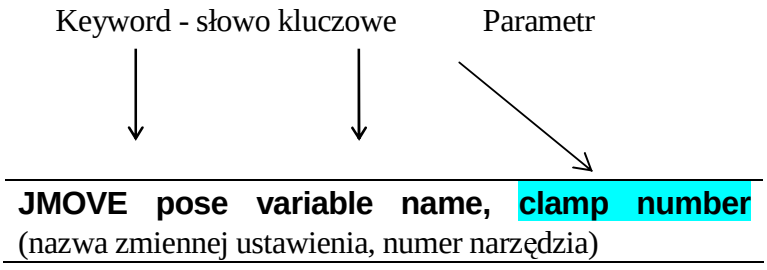
```
| ABC
| DEFGHI
```

| Wyświetlacz powinien wyglądać w następujący sposób, z “GHI” w tej samej linii co “DEF”.

# 6.0 INSTRUKCJE PROGRAMOWANIA JĘZYKA AS



**Przykład**




Parametry oznaczone mogą być pominięte.

Pomiędzy słowami kluczowymi, a parametrami zawsze wprowadzaj spacje (space).

↵ oznacza w podanych przykładach klawisz Enter.

## 6.1 INSTRUKCJE RUCHU

| | |
|---|---|
| JMOVE | Przesuwa robota w ruchu z interpolacją osiową. |
| LMOVE | Przesuwa robota w ruchu z interpolacją liniową. |
| DELAY | Zatrzymuje ruch robota na określony czas. |
| STABLE | Zatrzymuje ruch robota na określony czas, po osiągnięciu zbieżności osi.. |
| JAPPRO | Zbliża robota do punktu docelowego, w ruchu z interpolacją osiową. |
| LAPPRO | Zbliża robota do punktu docelowego, w ruchu z interpolacją liniową. |
| JDEPART | Opuszcza bieżące ustawienie, w ruchu z interpolacją osiową. |
| LDEPART | Opuszcza bieżące ustawienie, w ruchu z interpolacją liniową. |
| HOME | Przesuwa do pozycji wyjściowej. |
| DRIVE | Przesuwa w kierunku pojedynczej osi. |
| DRAW | Przesuwa o określony dystans w kierunku osi X, Y, Z współrzędnych globalnych. |
| TDRAW | Przesuwa o określony dystans w kierunku osi X, Y, Z układu współrzędnych narzędzia. |
| ALIGN | Ustawia w linii oś narzędzia (tool) Z z osią współrzędnych globalnych. |
| HMOVE | Przesuwa w ruchu z interpolacją liniową (oś kiści przesuwa się w ruchu z interpolacją osiową). |
| XMOVE | Przesuwa ruchem liniowym do określonego ustawienia. |
| C1MOVE | Przesuwa w ruchu z interpolacją kołową. (Opcja) |
| C2MOVE | Przesuwa w ruchu z interpolacją kołową. (Opcja) |

**JMOVE pose variable name, clamp number** (nazwa zmiennej ustawienia, numer narzędzia)
**LMOVE pose variable name, clamp number** (nazwa zmiennej ustawienia, numer narzędzia)

**Funkcja**

Przesuwa robota do określonego ustawienia.

JMOVE: Przesuwa w ruchu z interpolacją osiową.

LMOVE: Przesuwa w ruchu z interpolacją liniową.

**Parametr**

1. Nazwa zmiennej ustawienia

Określa punkt docelowy robota. (Dopuszczalne są wartości przekształcenia, złożone wartości przekształcenia, wartości przesunięcia osi lub funkcje informacji o ustawieniu.)

2. Clamp number (numer narzędzia)

Określa numer narzędzia, które ma być otwarte lub zamknięte w ustawieniu docelowym. Liczba dodatnia zamyka narzędzie, ujemna otwiera je. Możliwe jest skonfigurowanie dowolnego numeru narzędzia, aż do maksymalnego numeru ustawionego przy pomocy polecenia HSETCLAMP (lub funkcja pomocnicza 0605). Jeśli pominięto, narzędzie nie otwiera się lub nie zamyka.

**Objaśnienie**

Robot porusza się ruchem z interpolacją osiową podczas wykonywania instrukcji JMOVE. Robot porusza się w taki sposób, że stosunek przebytej odległości do całkowitej odległości pozostaje równy dla wszystkich osi, w czasie całego ruchu od ustawienia początkowego do ustawienia końcowego.

Robot porusza się ruchem z interpolacją liniową podczas wykonywanie instrukcji LMOVE. Punkt początkowy układu współrzędnych narzędzia (TCP) przesuwa się wzdłuż trajektorii liniowych.

**Przykład**

| | |
|---|---|
| JMOVE #pick | Przesuwa ruchem z interpolacją osiową do ustawienia określonego wartościami przesunięcia osi “#pick”. |
| LMOVE ref+place | Przesuwa ruchem z interpolacją liniową do ustawienia określonego złożonymi wartościami przekształcenia “ref + place”. |
| LMOVE #pick,1 | Przesuwa ruchem z interpolacją liniową do ustawienia określonego wartościami przesunięcia osi “#pick”. Po osiągnięciu ustawienia, narzędzie (clamp) 1 jest zamykane. |

## DELAY time (czas)

**Funkcja**

Zatrzymuje ruch robota na określony czas.

**Parametr**

Time - czas

Określa w sekundach czas zatrzymania ruchu robota.

**Objaśnienie**

W systemie AS, instrukcja DELAY jest traktowana jak instrukcja ruchu, która “moves to nowhere - przesuwa do nikąd”.

Nawet jeśli ruch robota jest zatrzymywany przez instrukcję DELAY, wszystkie kroki programu przed następną instrukcją ruchu są wykonywane przed zatrzymaniem.

**Przykład**

DELAY 2.5 Zatrzymuje ruch robota na 2,5 sekundy.

## STABLE time (czas)

**Funkcja**

Odracza wykonanie następnej instrukcji ruchu, aż do upłynięcia określonego okresu czasu po osiągnięciu zbieżności osi. (Oczekuje do momentu, w którym robot jest stabilny.

**Parametr**

Time - czas

Określa w sekundach czas zatrzymania ruchu robota.

**Objaśnienie**

Jeśli po zatrzymaniu robota przy pomocy omawianego polecenia osie przestają być zbieżne, czas jest liczony od momentu ponownego osiągnięcia przez osie zbieżności.

**JAPPRO pose variable name, distance** (nazwa zmiennej ustawienia, odległość)
**LAPPRO pose variable name, distance** (nazwa zmiennej ustawienia, odległość)

**Funkcja**

Przesuwa w kierunku narzędzia (tool) Z o określoną odległość od wyuczonego ustawienia.
JAPPRO: Przesuwa w ruchu z interpolacją osiową.
LAPPRO: Przesuwa w ruchu z interpolacją liniową.

**Parametr**

1. Pose variable names - nazwy zmiennej ustawienia
Określa ustawienie końcowe (w wartościach przekształcenia lub wartościach przesunięcia osi).

2. Odległość
Określa (w milimetrach) wartość przesunięcia pomiędzy ustawieniem końcowym a ustawieniem faktycznie osiągniętym przez robota na osi kierunku Z układu współrzędnych narzędzia. Jeśli określona odległość jest wartością dodatnią, robot przesuwa się w kierunku ujemnym osi Z. Jeśli określona odległość jest ujemną wartością, robot przesuwa się w kierunku dodatnim osi Z.

**Objaśnienie**

W przypadku omawianych poleceń, pozycja narzędzia jest konfigurowana w pozycji danego ustawienia, a pozycja w określonej odległości od danego ustawienia w kierunku osi Z układu współrzędnych narzędzia.

**Przykład**

| | |
|---|---|
| JAPPRO place,100 | Przesuwa ruchem z interpolacją osiową do ustawienia 100 mm od ustawienia “place” w kierunku osi Z układu współrzędnych narzędzia. Ustawienie “place” jest określone w wartościach przekształcenia. |
| LAPPRO place, offset | Przesuwa ruchem z interpolacją liniową do ustawienia oddalonego od ustawienia “place”, określonego w wartościach przekształcenia, o odległość określoną przez zmienną “offset”, w kierunku osi Z układu współrzędnych narzędzia. |

**JDEPART distance** (odległość)
**LDEPART distance** (odległość)

**Funkcja**

Przesuwa robota do ustawienia oddalonego od bieżącego ustawienia o określoną odległość, wzdłuż osi Z układu współrzędnych narzędzia.
JDEPART: Przesuwa w ruchu z interpolacją osiową.
LDEPART : Przesuwa w ruchu z interpolacją liniową.

**Parametr**

Odległość
Określa w milimetrach odległość pomiędzy ustawieniem bieżącym a ustawieniem docelowym na osi kierunku Z układu współrzędnych narzędzia. Jeśli określona odległość jest wartością dodatnią, robot przesuwa się “back” (w tył) lub w kierunku ujemnym osi Z. Jeśli określona odległość jest wartością ujemną, robot przesuwa się “forward” (do przodu) lub w kierunku dodatnim osi Z.

**Przykład**

| | |
|---|---|
| JDEPART 80 | Narzędzie robota przesuwa się w tył o 80 mm w kierunku –Z układu współrzędnych narzędzia ruchem z interpolacją osiową. |
| LDEPART 2∗offset | Narzędzie robota przesuwa się w tył o wartość 2∗offset (przesunięcie) (200 mm jeśli offset = 100) w kierunku – Z układu współrzędnych narzędzia ruchem z interpolacją liniową. |

## HOME home pose number (numer pozycji domowej)

**Funkcja**

Przesuwa ruchem z interpolacją osiową do ustawienia określonego jako HOME lub HOME2.

**Parametr**

Home pose number (numer pozycji domowej)
Określa numer pozycji domowej (1 lub 2). Jeśli pominięto, przyjmowana jest wartość HOME 1.

**Objaśnienie**

Skonfigurowane mogą być dwie pozycje domowe (HOME 1 i HOME 2). Omawiana instrukcja przesuwa robota do jednej z pozycji domowych, ruchem z interpolacją osiową. Pozycja domowa powinna być określona przed użyciem polecenia/instrukcji SETHOME lub SET2HOME. Jeśli nie zdefiniowano pozycji domowej, przyjmuje się, że współrzędne wyjściowe/punkt początkowy stanowią pozycję wyjściową (wszystkie osie w 0°).

**Przykład**

| | |
|---|---|
| HOME | Przesuwa do pozycji domowej określonej poleceniem/instrukcją SETHOME w ruchu z interpolacją osiową. |
| HOME 2 | Przesuwa do pozycji domowej określonej poleceniem/instrukcją SET2HOME w ruchu z interpolacją osiową. |

## **DRIVE joint number, displacement, speed** (numer osi, przesunięcie, prędkość)

**Funkcja**

Przesuwa pojedynczą oś robota.

**Parametr**

1. Joint number (numer osi)
Określanie numeru osi, która ma być przesunięta. (W przypadku robotów wyposażonych w sześć osi, są one oznaczone numerami od 1 do 6, zaczynając od osi najbardziej oddalonej od kołnierza mocującego narzędzia.)

2. Displacement (przesunięcie)
Określa wielkość przesunięcia osi jako wartość dodatnią lub ujemną.

Wartość ta jest wyrażana w jednostkach określających także ustawienie osi; tj. jeśli oś jest osią obrotową, wartość ta jest wyrażana w stopniach (°), a jeśli oś jest osią przesuwną, wartość ta jest wyrażana w jednostce odległości (mm).

3. Speed - prędkość
Określenie prędkości przedmiotowego ruchu. Jak w przypadku prędkości zwykłego programu, wyrażana jest w procentach prędkości monitorowania. Jeśli nie określono, przyjmuje się wartość 100% prędkości monitorowania.

**Objaśnienie**

Omawiana instrukcja przesuwa tylko jedną oś.

Prędkość ruchu dla tej instrukcji stanowi kombinację prędkości określonej przez nią oraz prędkości monitorowania. Prędkość programu ustawiona w programie nie ma wpływu na tę instrukcję.

**Przykład**

| | |
|---|---|
| DRIVE 2,-10,75 | Przesuwa oś (joint) 2 (JT2) −10° z bieżącego ustawienia. Prędkość stanowi wartość 75% prędkości monitorowania. |

| | |
|---|---|
| **DRAW** | **X translation, Y translation, Z translation X rotation, Y rotation, Z rotation, speed** (ruch postępowy X, ruch postępowy Y, ruch postępowy Z, ruch obrotowy X, ruch obrotowy Y, ruch obrotowy Z, prędkość) |

| | |
|---|---|
| **TDRAW** | **X translation, Y translation, Z translation X rotation, Y rotation, Z rotation, speed**(ruch postępowy X, ruch postępowy Y, ruch postępowy Z, ruch obrotowy X, ruch obrotowy Y, ruch obrotowy Z, prędkość) |

**Funkcja**

Przesuwa robota z określoną prędkością ruchem liniowym z bieżącego ustawienia o określoną odległość w kierunku osi X, Y, Z i dokonuje obrotu o określonej wartości wokół każdej z osi. Instrukcja DRAW przesuwa robota w oparciu o współrzędne globalne, instrukcja TDRAW przesuwa robota w oparciu o współrzędne narzędzia.

**Parametr**

1. X translation (ruch postępowy X)
Określa w mm wielkość przesunięcia na osi X. Jeśli nie określono, wprowadzana jest wartość 0 mm.

2. Y translation (ruch postępowy Y)
Określa w mm wielkość przesunięcia na osi Y. Jeśli nie określono, wprowadzana jest wartość 0 mm.

3. Z translation (ruch postępowy Z)
Określa w mm wielkość przesunięcia na osi Z. Jeśli nie określono, wprowadzana jest wartość 0 mm.

4. X rotation (ruch obrotowy X)
Określa w st. wielkość obrotu wokół osi X. Dopuszczalny zakres jest mniejszy niż ±180°. Jeśli nie określono, wprowadzana jest wartość 0 st.

5. Y rotation (ruch obrotowy Y)
Określa w st. wielkość obrotu wokół osi Y. Dopuszczalny zakres jest mniejszy niż ±180°. Jeśli nie określono, wprowadzana jest wartość 0 st.

6. Z rotation (ruch obrotowy Z)
Określa w st. wielkość obrotu wokół osi Z. Dopuszczalny zakres jest mniejszy niż ±180°. Jeśli

nie określono, wprowadzana jest wartość 0 st.

7. Speed - prędkość
Określa prędkość w %, mm/s, mm/min, cm/min lub s. Jeśli nie określono, robot przesuwa się z prędkością programu.

**Objaśnienie**

Robot przesuwa się ruchem liniowym z bieżącego ustawienia do określonego ustawienia.

**Przykład**

| | |
|---|---|
| DRAW 50,,-30 | Przesuwa ruchem liniowym z bieżącego ustawienia o 50 mm w kierunku osi X axis i o –30 mm w kierunku osi Z współrzędnych globalnych. |

## ALIGN

**Funkcja**

Przesuwa oś Z układu współrzędnych narzędzia w taki sposób, aby była równoległa z najbliższą osią współrzędnych globalnych.

**Objaśnienie**

Przy każdym zastosowaniu, jeśli kierunek ruchu referencyjnego jest ustawiony zgodnie z kierunkiem narzędzia (tool) Z, DO ALIGN umożliwia dogodne wyrównanie kierunku narzędzia do współrzędnych globalnych, przed przeprowadzeniem uczenia danych ustawienia.

| | |
|---|---|
| **C1MOVE** | **pose variable name, clamp number** (nazwa zmiennej ustawienia, numer narzędzia) |
| **C2MOVE** | **pose variable name, clamp number** (nazwa zmiennej ustawienia, numer narzędzia) |

Opcja

**Funkcja**

Przesuwa robota do określonego ustawienia po drodze kołowej.

**Parametr**

1. Pose variable name (nazwa zmiennej ustawienia)

Określa punkt docelowy ruchu robota. (Dopuszczalne są wartości przekształcenia, złożone wartości przekształcenia, wartości przesunięcia osi lub funkcje informacji o ustawieniu.)

2. Clamp number (numer narzędzia)

Określa numer narzędzia, które ma być otwarte lub zamknięte w ustawieniu docelowym. Liczba dodatnia zamyka narzędzie, ujemna otwiera je. Możliwe jest skonfigurowanie dowolnego numeru narzędzia, aż do maksymalnego numeru ustawionego przy pomocy polecenia HSETCLAMP (lub funkcja pomocnicza 0605). Jeśli pominięto, narzędzie nie otwiera się lub nie zamyka.

**Objaśnienie**

Instrukcja C1MOVE przesuwa do punktu znajdującego się w połowie drogi na trajektorii koła, instrukcja C2MOVE przesuwa do końca trajektorii.

Aby przesuwać robota ruchem z interpolacją kołową, niezbędne jest wyuczenie trzech ustawień. Omawiane trzy ustawienia różnią się w instrukcjach C1MOVE i C2MOVE.

C1MOVE： 1. Ustawienie ostatniej instrukcji ruchu.

2. Ustawienie, które ma być użyte jako parametr instrukcji C1MOVE.
3. Ustawienie kolejnej instrukcji ruchu. (instrukcje C1MOVE lub C2MOVE)

C2MOVE： 1. Ustawienie ostatniej instrukcji ruchu C1MOVE.

2. Ustawienie instrukcji ruchu poprzedzającej instrukcję C1MOVE.
3. Ustawienie instrukcji ruchu C2MOVE.

**[UWAGA]**

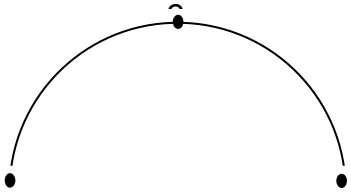
Przed instrukcją C1MOVE wymagane są następujące instrukcje ruchu:
ALIGN, C1MOVE, C2MOVE, DELAY, DRAW, TDRAW, DRIVE, HOME, JMOVE, JAPPRO, JDEPART, LMOVE, LAPPRO, LDEPART, STABLE, XMOVE

Po instrukcji C1MOVE musi nastąpić instrukcja C1MOVE lub C2MOVE.

Instrukcja C1MOVE musi poprzedzać instrukcję C2MOVE.

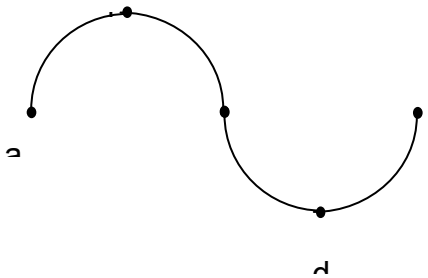
**Przykład**

```
JMOVE   c1
C1MOVE c2
C2MOVE c3
```

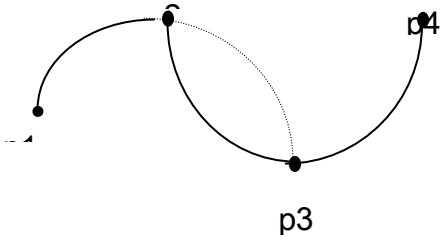


Robot przesuwa się ruchem z interpolacją osiową do c1, a następnie przesuwa się ruchem z interpolacją kołową po łuku utworzonym przez c1 c2 c3.

```
  JMOVE   #a
C1MOVE #b      } łuk a,b,c
C2MOVE #c
  C1MOVE #d    } łuk c,d,e
C2MOVE #e
```




```
  LMOVE   #p1    } łuk p1,p2,p3
C1MOVE #p2
  C1MOVE #p3   } łuk p2,p3,p4
C2MOVE #p4
```



## SPEED speed, rotational speed, ALWAYS

**Funkcja**

Określenie prędkości ruchu robota.

**Parametr**

1. Speed - prędkość

Określa prędkość programu. Zazwyczaj jest ona określana w procentach w zakresie od 0,01 do 100 (%). Prędkość absolutna może być skonfigurowana poprzez określenie prędkości w następujących jednostkach: MM/S i MM/MIN. Jednostka S (sekundy) określa czas ruchu. Jeśli jednostka ta jest pominięta, wielkość jest odczytywana jako procent (%).

2. Rotational speed - prędkość obrotowa（Opcja）

Określa prędkość obrotową pozycji narzędzia w ruchu z interpolacją liniową i kołową. Zazwyczaj jest ona określana w procentach w zakresie od 0,01 do 100 (%). Prędkość absolutna może być skonfigurowana poprzez określenie prędkości w następujących jednostkach: DEG/S i DEG/MIN (deg = st.). Jeśli jednostka ta jest pominięta, wielkość jest odczytywana jako procent (%). Jeśli parametr ten został pominięty, prędkość obrotowa jest ustawiana na 100%.

3. ALWAYS (zawsze)

Po wprowadzeniu tego parametru konfiguracja prędkości, w omawianej instrukcji, pozostaje ważna do wykonania kolejnej instrukcji SPEED. Jeśli nie wprowadzono, prędkość jest aktualna tylko dla kolejnej instrukcji ruchu.

**Objaśnienie**

Rzeczywista prędkość ruchu robota jest określana iloczynem prędkości monitorowania i prędkości ruchu skonfigurowanej przez omawianą instrukcję (Monitor speed (prędkość monitoringu)× Program speed (prędkość programu). Jednakże, w przypadkach, jak poniżej nie można zagwarantować pełnej prędkości.

1. Jeśli odległość pomiędzy dwoma wyuczonymi ustawieniami jest zbyt mała.
2. Jeśli wyuczono prędkość ruchu liniowego o wartość przekraczającą prędkość maksymalną osi.

Prędkość ruchu jest określana odmiennie w ruchu z interpolacją osiową i liniową. W ruchu z interpolacją osiową prędkość ruchu jest określana jako procent prędkości maksymalnej każdej osi. W ruchu liniowym prędkość ruchu jest określana jako procent prędkości maksymalnej

w punkcie początkowym układu współrzędnych narzędzia.

Gdy prędkość jest oznaczona w jednostce odległości na jednostkę czasu lub w sekundach, konfigurowana jest prędkość ruchu liniowego w punkcie początkowym układu współrzędnych narzędzia. Podczas ruchu z interpolacją osiową skonfiguruj prędkość w procentach. (Nawet jeśli prędkość ustawiono jako wartość absolutną lub czas ruchu, robot nie będzie się poruszał z tą prędkością. Prędkość natomiast będzie przetwarzana jako procent, jaki stanowi dana wartość w prędkości maksymalnej.)

Prędkość absolutna wyrażana w MM/ S i MM/MIN, a czas określający prędkość w S, opisują prędkość, jeśli prędkość monitorowania przyjmuje wartość 100 %. Jeśli prędkość monitorowania zmniejsza się, prędkości te również maleją (proporcjonalnie).

**[UWAGA]**

Nawet jeśli iloczyn prędkości programu i prędkości ustawionej przez polecenie SPEED (monitor speed - prędkość monitorowania) przekracza 100 %, rzeczywista prędkość monitorowania nie przekracza 100 %.
Nie można skonfigurować prędkości obrotowej bez włączonej opcji sterującej prędkością obrotową. Jeśli opcja ta jest wyłączona, występuje błąd.

**Przykład**

| | |
|---|---|
| SPEED 50 | Ustawia prędkość kolejnego ruchu na 50 %. |
| SPEED 100 | Ustawia prędkość kolejnego ruchu na 100 %. |
| SPEED 200 | Ustawia prędkość kolejnego ruchu na 100 % (prędkość przekraczająca 100 % jest odczytywana jako 100 %) |
| SPEED 20MM/S ALWAYS | Prędkość punktu początkowego (punktu centralnego narzędzia - TCP) jest ustawiana na 20 mm/sek., aż do zmiany jej przez inną instrukcję SPEED (jeśli prędkości monitorowania przyjmuje wartość 100 %). |
| SPEED 6000 MM/MIN | Ustawia prędkość następnego ruchu robota na 6000 mm / min. (Prędkość punktu początkowego układu współrzędnych narzędzia, jeśli prędkość monitorowania wynosi 100%). |
| SPEED 5 S | Ustawia prędkość następnego ruchu robota w taki sposób, że punkt docelowy jest osiągany w ciągu 5 sekund. (Prędkość punktu początkowego układu współrzędnych narzędzia, jeśli prędkość monitorowania wynosi 100%). |

| | |
|---|---|
| SPEED 100 MM/S, 10 DEG/S | Ustawia prędkość kolejnego ruchu. Pierwszeństwo mają te ustawienia prędkości, które wymagają dłuższego czasu do osiągnięcia punktu docelowego. |

## ACCURACY distance ALWAYS

**Funkcja**

Ustawia dokładność podczas określania ustawienia robota.

**Parametr**

1. Odległość

Określa w milimetrach odległość zakresu dokładności.

2. ALWAYS (zawsze)

Po wprowadzeniu tego parametru, konfiguracja dokładności pozostaje ważna do wykonania kolejnej instrukcji ACCURACY. Jeśli nie wprowadzono, konfiguracja dokładności jest aktualna tylko dla kolejnej instrukcji ruchu.

**Objaśnienie**

Po wprowadzeniu parametru ALWAYS wszystkie wykonywane ruchy są sterowane przez dokładność, skonfigurowaną przez omawianą instrukcję.

Wartość domyślna konfiguracji dokładności to 1 mm.

Istnieje ograniczenie stosowania konfiguracji dokładności, jako że kontrola dokładności w systemie AS uruchamiana jest dopiero po rozpoczęciu przez robota hamowania, podczas zbliżania się do wyuczonego ustawienia. (Patrz także 4.5.4 Relacje pomiędzy przełącznikiem (Switch) CP, a instrukcjami ACCURACY, ACCEL, DECEL.)

**[UWAGA]**

Jeśli dokładność jest ustawiona na 1 mm, robot konfiguruje ustawienia po każdej instrukcji ruchu, czyniąc przerwy pomiędzy odcinkami ruchu. Aby zapewnić ruch po ciągłej ścieżce ruchu (CP), ustaw dokładność na większy zakres.

Uważaj, aby nie ustawić zakresu dokładności na zbyt małe wartości. Może to spowodować brak zbieżności osi.

Dokładność skonfigurowana przez omawianą instrukcję nie dotyczy dokładności powtórzeń a jedynie ustawiania robota, z tego względu nie ustawiaj wartości 1 mm lub mniejszej.

**Przykład**

| | |
|---|---|
| ACCURACY 10 ALWAYS | Zakres dokładności jest ustawiony na10 mm dla wszystkich instrukcji ruchu następujących po tej instrukcji ruchu. |

## DECOMPOSE array variable name[element number] = pose variable name

**Funkcja**

Zapisuje każdy element wartości określonej zmiennej pozycji jako element zmiennej tablicowej (X, Y, Z, O, A, T dla wartości przekształcenia; JT1, JT2, JT3, JT4, JT5, JT6 dla wartości przesunięcia osi).

**Parametr**

1. Array variable name - nazwa zmiennej tablicowej
Określa nazwę zmiennej tablicowej, w której zapisane mają być wartości każdego elementu.

2. Element number (numer elementu)
Określa pierwszy element, w którym mają być zapisane elementy.

3. Pose variable name (nazwa zmiennej pozycji)
Określa nazwę zmiennej pozycji, z której ma być wyodrębniony każdy element (wartości przekształcenia, wartości przesunięcia osi).

**Objaśnienie**

Przydziela elementy określonej informacji o ustawieniu do elementów zmiennej tablicowej.

W przypadku wartości przekształcenia, przydzielanych jest sześć elementów każdej z wartości XYZOAT. W przypadku wartości przesunięcia osi, każdy z elementów jest przydzielany na podstawie wartości każdej osi ramienia robota.

**Przykład**

| | |
|---|---|
| DECOMPOSE X[0]=part | Przydziela elementy wartości przekształcenia “part” do pierwszych sześciu elementów w zmiennej tablicowej “x”. |
| DECOMPOSE angles[4]=#pick | Przydziela elementy wartości przesunięcia osi “#pick” do elementu numer 4 i do elementów następujących po nim w zmiennej tablicowej “angles”. |

Np., w powyższej instrukcji, jeśli wartości #pick są równe 10,20,30,40,50,60, to:

| | |
|---|---|
| angle[4]=10 (angle - kąt) | angle[7]=40 (angle - kąt) |
| angle[5]=20 (angle - kąt) | angle[8]=50 (angle - kąt) |
| angle[6]=30 (angle - kąt) | angle[9]=60 (angle - kąt) |

## 6.3 INSTRUKCJE STERUJĄCE NARZĘDZIEM

| | |
|---|---|
| OPEN | Wysyła sygnał otwarcia narzędzia, gdy zaczyna się kolejna instrukcja ruchu. |
| OPENI | Wysyła sygnał otwarcia narzędzia, gdy ukończona zostaje bieżąca instrukcja ruchu. |
| CLOSE | Wysyła sygnał zamknięcia narzędzia, gdy zaczyna się kolejna instrukcja ruchu. |
| CLOSEI | Wysyła sygnał zamknięcia narzędzia, gdy ukończona zostaje bieżąca instrukcja ruchu. |

**OPEN** **clamp number** (numer narzędzia)
**OPENI** **clamp number (numer narzędzia)**

**Funkcja**

Otwiera narzędzia robota (wysyła sygnał otwarcia narzędzia).

**Parametr**

Clamp number (numer narzędzia)
Określa numer narzędzia. Jeśli pominięto, przyjmowane jest 1.

**Objaśnienie**

Niniejsza instrukcja wysyła sygnały sterujące zaworem chwytaka pneumatycznego w celu otwarcia narzędzia.

W przypadku instrukcji OPEN, sygnał nie jest wysyłany, aż do rozpoczęcia kolejnego ruchu.

Poniżej przedstawiono taktowanie dla wysyłania sygnałów przy użyciu instrukcji OPENI:

1. Jeśli robot jest aktualnie w ruchu, sygnał jest wysyłany po ukończeniu tego ruchu. Jeśli robot porusza się po ciągłej ścieżce ruchu (CP motion), ciągła ścieżka ruchu jest zawieszana (BREAK).
2. Jeśli robot nie porusza się, sygnał jest natychmiast wysyłany do zaworu sterującego

**Przykład**

OPEN Sygnał otwarcia narzędzia jest wysyłany do zaworu sterującego narzędzia (clamp) 1, gdy robot zaczyna kolejny ruch.

OPENI 2 Sygnał otwarcia narzędzia jest wysyłany do zaworu sterującego narzędzia (clamp) 2, gdy robot kończy bieżący ruch.

| | |
|---|---|
| **CLOSE** | **clamp number**(numer narzędzia) |
| **CLOSEI** | **clamp number**(numer narzędzia) |

**Funkcja**

Zamyka narzędzia robota (wysyła sygnał zamknięcia narzędzia).

**Parametr**

Clamp number (numer narzędzia)

Określa numer narzędzia. Jeśli pominięto, przyjmowane jest 1.

**Objaśnienie**

Niniejsza instrukcja wysyła sygnały sterujące zaworem chwytaka pneumatycznego w celu zamknięcia narzędzia.

W przypadku instrukcji CLOSE, sygnał nie jest wysyłany, aż do rozpoczęcia kolejnego ruchu.

Poniżej przedstawiono taktowanie dla wysyłania sygnałów przy użyciu instrukcji CLOSEI:

1. Jeśli robot jest aktualnie w ruchu, sygnał jest wysyłany po ukończeniu tego ruchu. Jeśli robot porusza się po ciągłej ścieżce ruchu (CP motion), ciągła ścieżka ruchu jest zawieszana (BREAK).
2. Jeśli robot nie porusza się, sygnał jest natychmiast wysyłany do zaworu sterującego.

**Przykład**

| | |
|---|---|
| CLOSE 3 | Sygnał zamknięcia narzędzia jest wysyłany do zaworu sterującego narzędzia (clamp) 3, gdy robot zaczyna kolejny ruch. |
| CLOSEI | Sygnał zamknięcia narzędzia jest wysyłany do zaworu sterującego narzędzia (clamp) 2, gdy robot kończy bieżący ruch. |

## 6.5 INSTRUKCJE STERUJĄCE PROGRAMEM

| | |
|---|---|
| GOTO | Przechodzi do określonej etykiety (label). |
| IF | Przechodzi do określonej etykiety (label), gdy ustawiony jest warunek. |
| CALL | Rozgałęzia do podprogramu standardowego. |
| RETURN | Powraca do programu, który wywołał podprogram standardowy. |
| WAIT | Wprowadza wykonanie programu w stan oczekiwania (stand-by) do momentu ustawienia warunku. |
| TWAIT | Wprowadza wykonanie programu w stan oczekiwania (stand-by) do momentu upływu określonego czasu. |
| MVWAIT | Wprowadza wykonanie programu w stan oczekiwania (stand-by) do momentu uzyskania zadanej odległości lub czasu. |
| LOCK | Zmienia priorytety programów sterujących robotem. |
| PAUSE | Wstrzymuje wykonanie programu. |
| HALT | Zatrzymuje wykonanie programu. (Wznowienie nie jest możliwe.) |
| STOP | Zatrzymuje wykonanie cyklu. |
| SCALL | Rozgałęzia do podprogramu standardowego. |
| ONE | Wywołuje program po wystąpieniu błędu. |
| RETURNE | Wykonuje od kroku następującego po kroku, w którym wystąpił błąd. |

## GOTO label IF condition

**Funkcja**

Przechodzi do kroku programu, który jest oznaczony określoną etykietą (label).

**Parametr**

1. Label (etykieta)
Określa etykietę kroku programu, do którego należy przejść. Etykietą może być dowolna liczba całkowita w przedziale od 0 do 32767.

2. Condition (warunek)
Określa warunek dla przejścia (jump). Parametr oraz słowo kluczowe IF mogą być pominięte. Jeśli pominięto, program wykonuje przejście zawsze, gdy wykonywana jest instrukcja.

**Objaśnienie**

Przechodzi do kroku oznaczonego etykietą (label). Jeśli nie określono warunku, program wykonuje przejście, gdy ustawiony jest warunek. Jeśli warunek nie jest skonfigurowany, wykonanie jest kontynuowane od następnego kroku po niniejszej instrukcji.

Pamiętaj, że etykieta i numer kroku różnią się od siebie. Numery kroków są przydzielane przez system automatycznie do wszystkich kroków programu. Etykiety są przydzielane krokom programu w sposób celowy i są wprowadzane po numerze kroku.

Omawiana instrukcja działa identycznie do instrukcji IF GOTO, gdy ustawiony jest warunek.

**Przykład**

| | |
|---|---|
| GOTO 100 | Przechodzi do label 100, brak warunku. Jeśli krok oznaczony etykietą 100 nie istnieje, występuje błąd. |
| GOTO 200 IF n==3 | Jeśli zmienna “n” jest równa 3, program przechodzi do label 200. Jeśli nie, wykonywany jest krok następujący po tym kroku. |

## IF condition GOTO label

**Funkcja**

Przechodzi do kroku oznaczonego określoną etykietą (label), gdy ustawiony jest określony warunek.

**Parametr**

1. Condition (warunek)
Określa warunek w wyrażeniach, np.: n = = 0, n>3, m + n<0.

2. Label (etykieta)
Określa etykietę kroku, do którego należy przejść (nie numer kroku). Etykieta musi się znajdować w tym samym programie.

**Objaśnienie**

Program przechodzi do kroku oznaczonego określoną etykietą (label), gdy ustawiony jest określony warunek. Jeśli warunek nie jest spełniony, wykonywany jest krok następujący po tej instrukcji.

Jeśli określona etykieta nie istnieje, występuje błąd.

**Przykład**

| | |
|---|---|
| IF n>3 GOTO 100 | Jeśli wartość (liczba całkowita) zmiennej “n” jest większa od 3, program przechodzi do kroku oznaczonego etykietą (label) 100. Jeśli wartość zmiennej nie jest większa do 3, wykonywany jest krok następujący po niniejszym kroku. |
| IF flag GOTO 25 | Jeśli wartość (liczba całkowita) zmiennej “flag” jest różna od 0, program przechodzi do kroku oznaczonego etykietą (label) 25. Jeśli wartość zmiennej “flag” jest równa 0, wykonywany jest krok następujący po niniejszym kroku. Jest to równoznaczne z zapisem: IF flag<>0 GOTO 25. |

## CALL program name (nazwa programu)

**Funkcja**

Wstrzymuje wykonanie bieżącego programu i przechodzi do nowego programu (podprogramu standardowego). Po ukończeniu wykonania podprogramu standardowego, wykonywany jest ponownie oryginalny program od kroku następującego po instrukcji CALL.

**Parametr**

Program Name (nazwa programu)

Określa nazwę podprogramu standardowego, który ma być wykonany.

**Objaśnienie**

Niniejsza instrukcja wstrzymuje czasowo wykonanie bieżącego programu i przechodzi do pierwszego kroku określonego podprogramu standardowego.

**[UWAGA]**

Ten sam podprogram standardowy nie może być jednocześnie wywołany przy pomocy programu sterującego robotem i programu kontrolnego procesu działającego równolegle. Ponadto, podprogram standardowy nie może sam się wywołać.

W czasie, gdy wywoływane są podprogramy standardowe, można wstrzymać do 20 programów.

**Przykład**

CALL sub1 Przechodzi do podprogramu standardowego o nazwie “sub1”. Po wykonaniu instrukcji RETURN w “sub1”, wykonywany jest ponownie oryginalny program od kroku następującego po instrukcji CALL.

## RETURN

**Funkcja**

Kończy wykonanie podprogramu standardowego i powraca do kroku następującego po instrukcji CALL programu, który wywołał określony podprogram standardowy.

**Objaśnienie**

Omawiana instrukcja kończy wykonanie podprogramu standardowego i powraca do programu, który wywołał określony podprogram standardowy. Jeśli podprogram standardowy nie jest wywoływany przez inny program (np, gdy podprogram standardowy jest wykonywany przez polecenie EXECUTE), wykonanie programu jest kończone.

Po zakończeniu podprogramu standardowego, wykonywany jest dalej oryginalny program, nawet w przypadku braku instrukcji RETURN. Jednakże, instrukcja RETURN powinna być zapisana w ostatnim kroku podprogramu standardowego (lub w dowolnym miejscu, w którym podprogram standardowy ma być zakończony).

## WAIT condition (warunek)

**Funkcja**

Wprowadza program w stan oczekiwania do momentu ustawienia warunku.

**Parametr**

Condition (warunek)

Określa warunek stanu oczekiwania (stand-by). (wyrażenia w liczbach rzeczywistych)

**Objaśnienie**

Omawiana instrukcja wstrzymuje wykonanie programu do momentu ustawienia warunku. Polecenie CONTINUE NEXT wznawia wykonanie programu przed ustawieniem warunku (pomija wykonywaną instrukcję WAIT).

**Przykład**

| | |
|---|---|
| WAIT SIG (1001, – 1003) | Zatrzymuje wykonanie programu do momentu, gdy zewnętrzny sygnał wejściowy 1001 (WX1) jest włączony, a sygnał 1003(WX3) jest wyłączony. |
| WAIT TIMER(1)>10 | Zatrzymuje wykonanie programu do momentu, gdy wartość timer1 przekracza 10 (sekund). |
| WAIT n>100 | Zatrzymuje wykonanie programu do momentu, gdy wartość zmiennej “n” przekracza 100. (W niniejszym przypadku założono, iż zmienna “n” jest wartością podliczaną przez program kontrolny procesu działający równolegle lub przerwania programu.) |

## TWAIT time (czas)

**Funkcja**

Wstrzymuje wykonanie programu do momentu upływu określonego czasu.

**Parametr**

Time - czas

Określa w sekundach czas wstrzymania wykonania sygnału.

**Objaśnienie**

Omawiana instrukcja wstrzymuje wykonanie programu do momentu upływu określonego czasu.

Wykonywana instrukcja A TWAIT może być pominięta przy użyciu polecenia CONTINUE NEXT.

Instrukcja WAIT może być z identycznym skutkiem używana zamiast instrukcji TWAIT.

**Przykład**

| | |
|---|---|
| TWAIT 0.5 | Wprowadza oczekiwanie na 0,5 sekunndy. |
| TWAIT deltat | Wprowadza oczekiwanie do momentu upłynięcia zmiennej “deltat”. |

## 6.6 INSTRUKCJE STRUKTURY PROGRAMU

IF……THEN…ELSE……END

WHILE……DO……END

DO……UNTIL

FOR……END

CASE……OF……VALUE……ANY……END

SCASE……OF……SVALUE……ANY……END

**IF logical expression THEN** (JEŚLI wyrażenie logiczne TO)
**program instructions(1) - instrukcje programu**
**ELSE (ALBO)**
**program instructions(2) (instrukcje programu)**
**END (KONIEC)**

**Funkcja**

Wykonuje grupę kroków programu, zgodnie z wynikiem wyrażenia logicznego.

**Parametr**

1. Logical expression (wyrażenie logiczne)
Wyrażenie logiczne lub wyrażenie wartości rzeczywistych. Sprawdza czy wartość ta jest TRUE (prawdziwa) (nie 0), czy FALSE (fałszywa) (0).

2. Program instructions (1) (instrukcje programu)
Wprowadzone tutaj instrukcje programu są wykonywane, jeśli powyższe wyrażenie logiczne posiada wartość TRUE.

2. Program instructions (2) (instrukcje programu)
Wprowadzone tutaj instrukcje programu są wykonywane, jeśli powyższe wyrażenie logiczne posiada wartość FALSE.

**Objaśnienie**

Omawiana struktura przepływu sterowania wykonuje jedną z dwóch grup instrukcji zgodnie z wartością wyrażenia logicznego. Poniżej przedstawiono procedurę wykonania:

1. Oblicza wyrażenie logiczne i przechodzi do kroku (step) 4, jeśli otrzymana wartość to 0 (FALSE).
2. Oblicza wyrażenie logiczne i wykonuje instrukcje programu (1), jeśli otrzymana wartość to 1 (TRUE).
3. Przechodzi (jumps) do 5.
4. Jeśli istnieje instrukcja ELSE, wykonywane są program instructions (instrukcje programu) (2).
5. Kontynuuje wykonanie programu od kroku następującego po END.

1. Instrukcja ELSE i END muszą być wprowadzane w osobnych wierszach.

2. Struktura IF…THEN musi się kończyć instrukcją END.

[**UWAGA**]

**Przykład**

W poniższym programie, jeśli n jest większe od 5, prędkość programu (program speed) jest ustawiana na 10%, jeśli nie, na 20 %.

```
21      IF n>5 THEN
22              sp=10
23      ELSE
24              sp=20
25      END
26      SPEED sp ALWAYS
```

Poniższy program sprawdza najpierw wartość zmiennej “m”. Jeśli “m” jest różne od 0, program sprawdza zewnętrzny sygnał wejściowy 1001(WX1) i w zależności od statusu sygnału wyświetla różne komunikaty. W niniejszym przykładzie, zewnętrzna struktura IF nie posiada instrukcji ELSE.

```
71      IF m THEN
72              IF SIG(1001) THEN
73                      PRINT"Input signal is TRUE" (sygnał wejściowy posiada wartość
TRUE)
74              ELSE
75                      PRINT"Input signal is FALSE" (sygnał wejściowy posiada wartość
FALSE)
76              END
77      END
```

## WHILE condition DO (JEŚLI warunek WYKONAJ)
## program instructions (instrukcje programu)
## END (KONIEC)

#### Funkcja

Jeśli określony warunek posiada wartość TRUE (prawda), wykonywane są instrukcje programu. Jeśli warunek przyjmuje wartość FALSE (fałsz), instrukcja WHILE jest pomijana.

#### Parametr

1. Condition (warunek)

Wyrażenie logiczne lub wyrażenie wartości rzeczywistych. Sprawdza czy wartość ta jest TRUE (prawdziwa) (nie 0), czy FALSE (fałszywa) (0).

2. Program instructions (instrukcje programu)

Określa grupę instrukcji, które mają być wykonane, jeśli warunek przyjmuje wartość TRUE.

#### Objaśnienie

Omawiana struktura przepływu sterowania powtarza dane kroki programu, jeśli określony warunek przyjmuje wartość TRUE. Poniżej przedstawiono procedurę wykonania:

1. Oblicza wyrażenie logiczne i przechodzi do kroku (step) 4, jeśli otrzymana wartość to 0 (FALSE).
2. Oblicza wyrażenie logiczne i wykonuje instrukcje programu, jeśli otrzymana wartość to 1 (TRUE).
3. Przechodzi (jumps) do 1.
4. Kontynuuje wykonanie programu od kroku następującego po END.

**[UWAGA]**

W przeciwieństwie do struktury DO, jeśli warunek przyjmuje wartość FALSE, żaden z kroków programu w strukturze WHILE nie jest wykonywany.

Jeśli używana jest omawiana struktura, wartość warunku musi się w końcu zmienić z TRUE na FALSE.

#### Przykład

W poniższym przykładzie, monitorowane są sygnały wejściowe 1001 i 1002, a ruch robota jest zatrzymywany w oparciu o ich warunki. Jeśli oba sygnały pochodzące z podajników dwóch

części (parts) zmieniają swoją wartość na 0 (podajnik jest pusty), robot zatrzymuje się, a wykonanie jest kontynuowane od kroku następującego po instrukcji END (krok (step) 27 w niniejszym przykładzie).

Jeśli jeden z podajników w czasie uruchomienia struktury WHILE (external input signal (zewnętrzny sygnał wejściowy wył) OFF=0), nie jest wykonywany żaden krok struktury, a przetwarzanie jest przesuwane do kroku 27.

```
20      .
21      .
22      .
23      WHILE SIG(1001,1002) DO
24              CALL part1
25              CALL part2
26      END
27      .
28      .
29      .
30      .
```

## DO (WYKONAJ)
## program instructions (instrukcje programu)
## UNTIL logical expression (AŻ DO wyrażenie logiczne)

**Funkcja**

Tworzy pętlę wykonania (DO loop)

**Parametr**

1. Program instructions (instrukcje programu)

Instrukcje te są powtarzane dopóty, dopóki wyrażenie logiczne nie przyjmie wartości FALSE.

2. Logical expression (wyrażenie logiczne)

Wyrażenie logiczne lub wyrażenie wartości rzeczywistych. Jeśli wynik tego wyrażenia logicznego zmienia się na TRUE, wykonanie instrukcji programu w tej strukturze jest zatrzymywane.

**Objaśnienie**

Omawiana struktura przepływu sterowania wykonuje grupę instrukcji programu, jeśli podany warunek (wyrażenie logiczne) przyjmuje wartość FALSE.

Poniżej przedstawiono procedurę wykonania:

1. Wykonuje instrukcje programu.
2. Sprawdza wartość wyrażenia logicznego i jeśli wynik przyjmuje wartość FALSE, powtarzana jest procedura 1. Jeśli wynik przyjmuje wartość TRUE, następuje przejście do procedury 3.
3. Kontynuuje wykonanie programu od kroku następującego po instrukcji UNTIL.

Wykonywanie struktury DO jest kończone, gdy wartość wyrażenia logicznego zmienia się z FALSE na TRUE.

W przeciwieństwie do struktury WHILE, instrukcje programu w strukturze DO są wykonywane co najmniej jeden raz.

Instrukcje programu pomiędzy instrukcją DO, a UNTIL mogą być pominięte. Jeśli instrukcji brak, powtarzana jest ocena wyrażenia logicznego następującego po UNTIL. Jeśli wartość wyrażenia logicznego zmienia się na TRUE, kończone jest wykonanie pętli i wykonywany jest krok następujący po strukturze DO.

Struktura DO musi się kończyć instrukcją UNTIL.

**Przykład**

W przykładzie poniżej, struktura DO steruje następującym zadaniem: część jest podnoszona i przenoszona do bufora. Gdy bufor zostaje zapełniony, włączany jest binarny sygnał wejściowy “buffer.full”. Po włączeniu sygnału robot zatrzymuje się i rozpoczyna inną operację.

```
10      .
11      .
12      .
13      DO
14          CALL get.part
15          CALL put.part
16      UNTIL SIG(buffer.full)
17      .
18      .
19      .
```

**FOR loop variable = start val** [UWAGA] **nd value STEP step value** (DLA zmienna pętli = wartość początkowa DO wartość końcowa KROK wartość kroku)
**program instructions (instrukcje programu) END (KONIEC)**

**Funkcja**

Powtarza wykonanie programu.

**Parametr**

1. Loop variable (zmienna pętli)

Zmienna lub wartość rzeczywista. Zmienna ta stanowi pierwszą konfigurację wartości początkowej, po każdorazowym wykonaniu pętli dodawane jest 1.

2. Starting value (wartość początkowa)

Wartość rzeczywista lub wyrażenie. Określa pierwszą wartość zmiennej pętli.

3. End value (wartość końcowa)

Wartość rzeczywista lub wyrażenie. Wartość ta jest porównywana do aktualnej wartości zmiennej pętli i jeśli wartość zmiennej pętli osiąga tą wartość, program kończy pętlę.

4. Step value (wartość kroku)

Wartość rzeczywista lub wyrażenie mogą być pominięte. Po każdej pętli wartość ta jest dodawana do lub odejmowana od zmiennej pętli. Jeśli używasz instrukcji STEP, wprowadź ten parametr, chyba że zmienna pętli ma wzrosnąć o 1. Jeśli nie określono wartości kroku, do zmiennej pętli dodawane jest 1. W takim przypadku, instrukcja STEP również może być ominięta.

**Objaśnienie**

Omawiana struktura przepływu powtarza wykonanie instrukcji programu pomiędzy instrukcjami FOR i END. Zmienna pętli jest powiększana o określoną wartość kroku po każdorazowym wykonaniu pętli.

Poniżej przedstawiono procedurę wykonania:

1. Wartość początkowa jest przydzielana do zmiennej pętli.
2. Obliczana jest wartość końcowa i wartość kroku.
3. Porównywana jest zmienna pętli z wartością końcową.

a. Jeśli wartość kroku jest dodatnia, a zmienna pętli jest większa, niż wartość końcowa, następuje przejście do procedury 7.
b. Jeśli wartość kroku jest ujemna, a zmienna pętli jest mniejsza, niż wartość końcowa, następuje przejście do procedury 7.

W innych przypadkach, następuje przejście do procedury 4.

4. Wykonywane są instrukcje programu po instrukcji FOR.
5. Po osiągnięciu instrukcji END, wartość kroku jest dodawana do zmiennej pętli.
6. Następuje powrót do procedury 3.
7. Wykonywane są instrukcje programu po instrukcji END. (Wartość zmiennej pętli w momencie testu porównawczego powyższej procedury 3 nie zmienia się.)

**[UWAGA]**

Dla każdej instrukcji FOR musi istnieć instrukcja END.

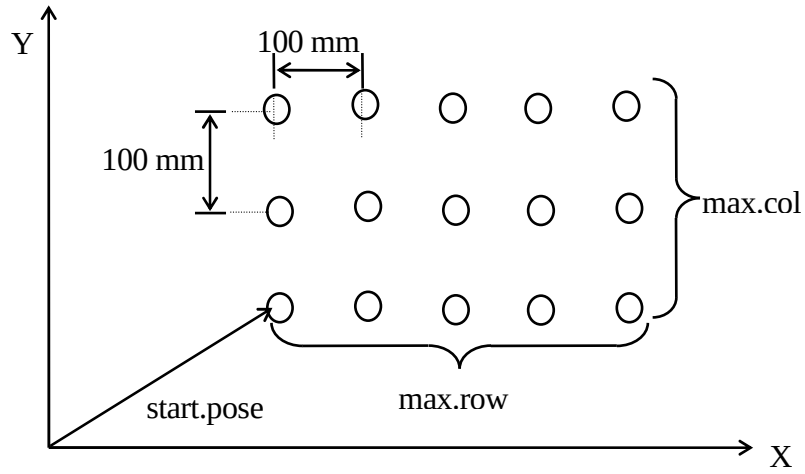
Pamiętaj, że jeśli podczas pierwszej kontroli zmienna pętli jest większa, niż wartość końcowa (lub mniejsza, jeśli wartość kroku jest ujemna), żadna z instrukcji programu pomiędzy FOR a END, nie jest wykonywana.

Wartość numeru pętli (zmienna pętli) nie może być zmieniana przez inne programowanie (operatory, wyrażenia, itp.) w pętli FOR.

**Przykład**

Podprogram standardowy “pick.place” podnosi część i umieszcza ją na “hole” (otwór). Części są umieszczane w sposób zaprezentowany na rysunku poniżej. (Paleta jest umieszczona równolegle do osi X i Y globalnego układu współrzędnych, a odległość pomiędzy częściami wynosi 100 mm.

```
FOR row = 1 TO max.row
POINT hole = SHIFT (start.pose BY (row-1)*100,0,0)
FOR col = 1 TO max.col
CALL pick.place
POINT hole = SHIFT(hole BY 0,100,0)
END
END
```




**CASE index variable OF (PRZYPADEK zmienna indeksu)**
**VALUE case number 1，……: (WARTOŚĆ numer przypadku 1)**
**program instructions (instrukcje programu)**
**VALUE case number 2，……: (WARTOŚĆ numer przypadku 1)**
**program instructions (instrukcje programu)**
**:**
**VALUE case number n，……: (WARTOŚĆ numer przypadku n)**
**program instructions (instrukcje programu)**
**ANY : (DOWOLNE)**
**program instructions (instrukcje programu)**
**END**

**Funkcja**

Wykonuje program zgodnie z określonym numerem przypadku.

**Parametr**

1. Index variable (zmienna indeksu)

Zmienna wartości rzeczywistej lub wyrażenie. Określa, która struktura CASE ma być wykonywana zgodnie z wartością tej zmiennej.

2. Program instructions (instrukcje programu)

Wykonuje instrukcje programu, gdy wartość zmiennej indeksu równa się jednej z wartości po instrukcji VALUE.

**Objaśnienie**

Omawiana struktura umożliwia programowi wybór spośród kilku grup instrukcji i przetwarzanie tej wybranej. Jest to potężne narzędzie języka AS zapewniające dogodną metodę dopuszczającą w ramach programu kilka alternatyw.

Poniżej przedstawiono procedurę wykonania:

1. Sprawdzanie wartości zmiennej indeksu po instrukcji CASE.
2. Kontrola kroków VALUE i znalezienie pierwszego kroku, który zawiera wartość równą wartość zmiennej indeksu.

3. Wykonanie instrukcji po tym kroku VALUE.
4. Przejście do instrukcji po instrukcji END.

Jeśli nie istnieje wartość pasująca do zmiennej indeksu, wykonywane są instrukcje programu po instrukcji ANY. Jeśli brak jest instrukcji ANY, żaden z kroków w strukturze CASE nie jest wykonywany.

**[UWAGA]**

Instrukcja ANY oraz jej instrukcje programu mogą być pominięte.

Instrukcja ANY w omawianej strukturze może być użyta tylko jeden raz. Instrukcja ta musi się znajdować na końcu struktury, jak zaprezentowano w poniższym przykładzie.

Dwukropek ":" po instrukcji wyrażenie ANY może być pominięte. Wprowadzając dwukropek, zawsze wstawiaj spację po ANY. ANY: bez spacji jest traktowane jako etykieta.

Zarówno instrukcja ANY, jak i END muszą być wprowadzane w osobnych wierszach.

**Przykład**

W poniższym przykładzie, jeśli wartość zmiennej rzeczywistej x jest ujemna, wykonanie programu jest zatrzymywane po wyświetleniu komunikatu. Jeśli wartość jest dodatnia, program jest przetwarzany, zgodnie z poniższymi 3 przypadkami:

1. jeśli wartość jest liczbą parzystą w przedziale od 0 do 10,
2. jeśli wartość jest liczbą nieparzystą w przedziale od 1 do 9,
3. jeśli wartość jest liczbą dodatnią inną, niż powyżej określone.

```
        IF x<0 GOTO 10
        CASE x OF
        VALUE 0,2,4,6,8,10:
        PRINT "The number x is EVEN" (liczba x jest liczbą PARZYSTĄ)
        VALUE 1,3,5,7,9:
        PRINT "The number x is ODD" (liczba x jest liczbą NIEPARZYSTĄ)
        ANY :
        PRINT "The number x is larger than 10" (liczba jest większa, niż 10)
        END
        STOP
10      PRINT "Stopping because of negative value" (zatrzymanie ze względu na wartość ujemną)
        STOP
```

```
SCASE   index variable   OF (PRZYPADEK CIĄGU zmienna indeksu)
SVALUE   string_1 , ……: (WARTOŚĆ CIĄGU ciąg_1)
program instructions (instrukcje programu)
SVALUE   string_2 , ……: (WARTOŚĆ CIĄGU ciąg_2)
program instructions (instrukcje programu)
:
SVALUE   string_n , ……: (WARTOŚĆ CIĄGU ciąg_n)
program instructions (instrukcje programu)
ANY : (DOWOLNE)
program instructions (instrukcje programu)
END (KONIEC)
```

Opcja

**Funkcja**

Wykonuje program zgodnie z określonym ciągiem znaków.

**Parametr**

1. Index variable (zmienna indeksu)

Określa zmienną ciągu znaków lub wyrażenie. Określa, która struktura SCASE ma być wykonywana zgodnie z ciągiem znaków tej zmiennej.

2. Program instructions (instrukcje programu)

Wykonuje instrukcje programu, gdy wartość ciągu zmiennej indeksu równa się jednej z wartości po instrukcji SVALUE.

**Objaśnienie**

W przeciwieństwie do opisanej wcześniej instrukcji CASE, warunek wykonania dla struktury SCASE jest konfigurowany jako ciąg znaków. Patrz także, CASE structure.

Jeśli nie istnieje ciąg pasujący do zmiennej indeksu, wykonywane są instrukcje programu po instrukcji ANY. Jeśli brak jest instrukcji ANY, żaden z kroków w strukturze SCASE nie jest wykonywany.

[**UWAGA**]

Instrukcja ANY oraz jej instrukcje programu mogą być pominięte.

Instrukcja ANY w omawianej strukturze może być użyta tylko jeden raz. Instrukcja musi się znajdować na końcu struktury.

Dwukropek ":" po instrukcji wyrażenie ANY może być pominięte. Wprowadzając dwukropek, zawsze wstawiaj spację po ANY. ANY: bez spacji jest traktowane jako etykieta.

Zarówno instrukcja ANY, jak i END muszą być wprowadzane w osobnych wierszach.

**Przykład**

W poniższym programie, jeśli zmienna ciągu znaków $str równa się ciągowi $a+"c", wykonywany jest program kontrolny procesu działający równolegle (PC). Jeśli zmienna ciągu znaków $str równa się ciągowi $a+"g", wykonywany jest program pg.

```
SCASE $ s t r OF
SVALUE $a+"c":
CALL pc
SVALUE $a+"g":
CALL pg
END
```

## 8.0 OPERATORY

W niniejszym rozdziale przedstawiono funkcje operatorów w języku AS. Operatory te są używane w połączeniu z poleceniami wprowadzanymi z ekranu (monitor commands) oraz instrukcjami programu (program instructions).

## 8.1 OPERATORY ARYTMETYCZNE

Operatory arytmetyczne są wykorzystywane do wykonywania ogólnych obliczeń matematycznych.

| Operator | Funkcja | Przykład |
|---|---|---|
| + | Dodawanie | i = i+1 |
| – | Odejmowanie | j= i–1 |
| * | Mnożenie | i = i*3 |
| / | Dzielenie | i = i/2 |
| MOD | Reszta | i= iMOD2 |
| ^ | Potęga | i = i^3 |

**Przykład**

i = i+1 Wartość i plus 1 jest przypisana do i; np. jeśli i równa się 5, do i zostanie przypisane 6, jako wynik wyrażenia i + 1.

i = i MOD 2 Jeśli i jest równe 5, operator wykonuje działanie 5 ÷ 2 i przypisuje i resztę 1.

i = i ^ 3 Wartość $i^3$ jest przypisywana do i. Jeśli i jest równe 2, po lewej stronie instrukcji do i przypisywane jest 8.

W przypadku dzielenia (/) i MOD (reszty), użycie 0 jako wartości położonej najbardziej na prawo skutkuje wystąpieniem błędu.

**Przykład** i = i/0
i = i MOD 0

## 8.2 OPERATORY RELACYJNE

Operatory relacyjne są używane z takimi instrukcjami jak IF i WAIT, w celu weryfikacji konfiguracji warunku.

| Operator | Funkcja | Przykład |
|---|---|---|
| < | TRUE (–1) (prawda), jeśli wartość po lewej jest mniejsza od wartości po prawej | i < j |
| > | TRUE (–1) (prawda), jeśli wartość po lewej jest większa od wartości po prawej | i > j |
| <= | TRUE (–1) (prawda), jeśli wartość po lewej jest równa wartości po prawej | i<= j |
| =< | Jak powyżej | i =< j |
| >= | TRUE (–1) (prawda), jeśli wartość po lewej jest większa lub równa wartości po prawej | i >= j |
| => | Jak powyżej | i=> j |
| == | TRUE (–1) (prawda), jeśli obie strony są równe | i == j |
| <> | TRUE (–1) (prawda), jeśli strony nie są sobie równe | i <> j |

**Przykład**

IF i<j GOTO 10 — Jeśli j jest większe od i, (tj. instrukcja i<j jest prawdziwa), program przechodzi do kroku oznaczonego etykietą (label) 10. Jeśli nie, wykonywany jest następny krok.

WAIT t==5 — Jeśli t przyjmuje wartość 5 (tj. t==5 jest prawdą), program przechodzi do następnego kroku, jeśli nie, wykonanie program jest zawieszane do momentu skonfigurowania warunku.

IF i+j>100 GOTO 20 — Jeśli i + j jest większe od 100 (tj. wyrażenie i + j > 100 jest prawdziwe), program przechodzi do kroku oznaczonego etykietą (label) 20. Jeśli nie, wykonywany jest następny krok.

IF $a =="abc" GOTO 20 — Jeśli $a jest równe “abc” (tj. $a == “abc” jest prawdziwe), program przechodzi do kroku oznaczonego etykietą (label) 20. Jeśli nie, wykonywany jest następny krok.

## 8.3 OPERATORY LOGICZNE

Operatory logiczne są używane w takich operacjach logicznych, jak 0+1=1, 1+1=1, 0+0=0 (logiczne OR (LUB)) lub 0×1=0, 1×1=1, 0×0=0 (logiczne AND (I)). W języku AS istnieją dwa rodzaje operatorów logicznych, operatory logiczne oraz binarne.

Operatory logiczne nie są używane w obliczaniu wartości liczbowych, a przy określaniu prawdy lub fałszu danej wartości lub wyrażenia. Jeśli wartość liczbowa jest 0, przyjmuje się, że jest fałszywa - FALSE (OFF). Natomiast, wszystkim wartościom różnym od zera przypisuje się prawdę - TRUE. Pamiętaj, że obliczenia, w których używany jest omawiany operator w przypadku −1 dają wartość TRUE (prawda).

| Operator | Funkcja | Przykład |
|---|---|---|
| AND (I) | Logiczne AND (I) | i AND j |
| OR (LUB) | Logiczne OR (LUB) | i OR j |
| XOR | Wyłączające logiczne OR (LUB) | i XOR j |
| NOT (NIE) | Logiczne uzupełnienie | NOT i |

**Przykład**

i AND j Wyznacza wartość logicznego AND pomiędzy i i j. Zmienne i i j są zazwyczaj wartościami logicznymi, ale mogą być także liczbami rzeczywistymi. W takim przypadku, wszystkim liczbom rzeczywistym różnym od 0 przypisuje się prawdę - ON (TRUE).

| i | j | Wynik |
|---|---|---|
| 0 | 0 | 0 (OFF) |
| 0 | not (nie) 0 | 0 (OFF) |
| not (nie) 0 | 0 | 0 (OFF) |
| not (nie) 0 | not (nie) 0 | −1 (ON) |

Wynik przyjmuje wartość ON (TRUE) tylko wtedy, gdy obie wartości są prawdziwe - ON (TRUE).

i OR j Wyznacza wartość logicznego OR pomiędzy i i j.

| i | j | Wynik |
|---|---|---|
| 0 | 0 | 0 (OFF) |
| 0 | not (nie) 0 | −1 (ON) |
| not (nie) 0 | 0 | −1 (ON) |
| not (nie) 0 | not (nie) 0 | −1 (ON) |

Wynik przyjmuje wartość ON (TRUE) tylko wtedy, gdy obie wartości są prawdziwe - ON (TRUE) (lub każda z nich).

i XOR j Wyznacza wartość logicznego OR pomiędzy i i j.

| i | j | Wynik |
|---|---|---|
| 0 | 0 | 0 (OFF) |
| 0 | not (nie) 0 | −1 (ON) |
| not (nie) 0 | 0 | −1 (ON) |
| not (nie) 0 | not (nie) 0 | 0 (OFF) |

Wynik przyjmuje wartość ON (TRUE) tylko wtedy, gdy jedna z wartości jest prawdziwa - ON (TRUE).

NOT i Wyznacza wartość logicznego uzupełnienia i.

| i | Wynik |
|---|---|
| 0 | −1 (ON) |
| not (nie) 0 | 0 (OFF) |

W systemie AS, logiczny status wartości lub wyrażenie może być wyrażany w następujący sposób:

True (prawda) not 0 (nie 0), ON, TRUE
False (fałsz): 0, OFF, FALSE

## 8.4 OPERATORY BINARNE

Binarne operatory logiczne wykonują operacje logiczne dla każdego odpowiedniego bita dwóch wartości liczbowych. Np., jeśli liczba składa się z 4 bitów, obliczane będą wartości: 0000, 0001, 0010, 0011, ……, 1111 (W systemie AS, wartości liczbowe składają się z 32 bitów).

| Wyrażenie binarne. | Wyrażenie dziesiętne |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| ⋮ | ⋮ |
| 1111 | 15 |

| Operator | Funkcja | Przykład |
|---|---|---|
| BOR | Binarne OR (LUB) | i BOR j |
| BAND | Binarne AND (I) | i BAND j |
| BXOR | Binarne XOR | i BXOR j |
| COM | Binarne uzupełnienie | COM i |

**Przykład**

i BOR j Jeśli i=5, j=9, wynik równa się 13.

i=5 0101
j=9 1001
1101 ⇒13

i BAND j Jeśli i=5, j=9, wynik równa się 1.

i=5 0101
j=9 1001
0001 ⇒ 1

i BXOR j Jeśli i=5, j=9, wynik równa się 12.

i=5 0101

j=9 1001
1100 ⇒ 12

COM i Jeśli i=5, wynik równa się –6.

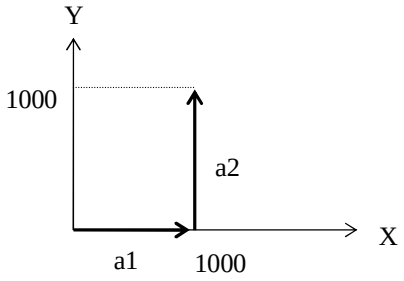
i=5 0… 0101
1… 1010 ⇒ –6

## 8.5 OPERATORY WARTOŚCI PRZEKSZTAŁCENIA

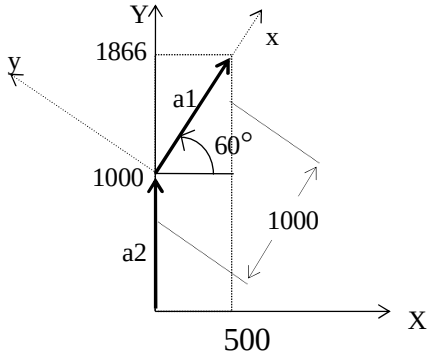
W systemie AS, operatory + i – są wykorzystywane do określania złożonych wartości przekształcenia (wartości XYZOAT). Pamiętaj jednak, że odmiennie niż w przypadku zwykłego dodawania lub odejmowania, prawo przemienności nie przypisuje prawdy (true) w przypadku operacji przekształcenia. Wyrażenia arytmetyczne " a + b" i "b + a" będą dawały identyczny wynik, natomiast "pose a + pose b" niekoniecznie będzie równe "pose b + pose a". Dzieje się tak, ponieważ w przypadku operacji przekształcenia brane są pod uwagę pozycje osi. Jak w poniższym przykładzie:

a1 = (1000, 0, 0, 0, 0, 0)
a2 = ( 0, 1000, 0, 60, 0, 0)

a1+a2 = (1000, 1000, 0, 60, 0, 0)



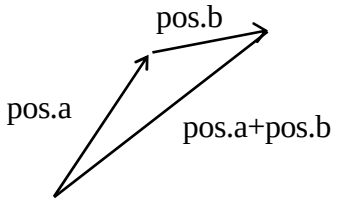

a2+a1 = (500, 1866, 0, 60, 0, 0)



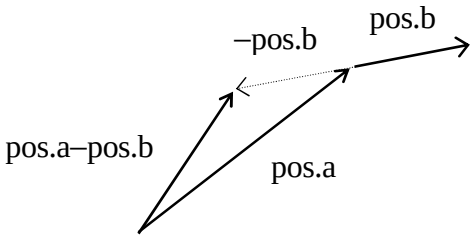
| Operator | Funkcja | Przykład |
|---|---|---|
| + | Dodawanie dwóch wartości przekształcenia | pos.a+pos.b |
| – | Odejmowanie dwóch wartości przekształcenia | pos.a–pos.b |

**Przykład**

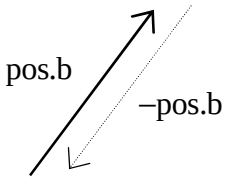
pos.a+pos.b




pos.a–pos.b



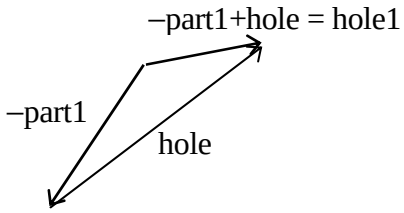

Operator przekształcenia "–" użyty z pojedynczą wartością (np. –pos.b) oznacza odwrotną wartość pos.b. Np., jeśli wartość przekształcenia pos.b określa ustawienie przedmiotu B w odniesieniu do przedmiotu A, to –pos.b określa ustawienie przedmiotu A w odniesieniu do przedmiotu B.



W poniższym przykładzie, “hole1” ma być określony w odniesieniu do “part1” (zdefiniowanej wcześniej). Można to uczynić przy pomocy złożonej wartości przekształcenia part1+hole.

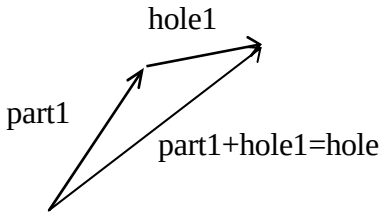
Przesuń krokowo robota do ustawienia, które ma być zdefiniowane jako “hole1” i wyucz go jako “hole”, przy pomocy polecenia HERE. Używając tego ustawienia (“hole”), można określić “hole1”.

POINT hole1 = – (part1)+hole




Innym sposobem określenia “hole1”, bez używania operatora “–”, jest napisanie “hole1” na lewo od wyrażenia w poleceniu POINT. Poniższe polecenie także określa “hole1”.

POINT part1+hole1 = hole



## 8.6 OPERATORY CIĄGU ZNAKÓW

| Operator | Funkcja | Przykład |
|---|---|---|
| + | Łączy dwa ciągi | $a = $b + $c |

**Przykład**

$a=$b+$c Łączy $b+$c i przypisuje ten ciąg do $a.
Jeśli $b=“abc”, $c=“123”, to $a równa się “abc123”.

## DISTANCE (transformation value, transformation value) (wartość przekształcenia, wartość przekształcenia)

**Funkcja**

Oblicza odległość pomiędzy dwoma ustawieniami, które są wyrażone w wartościach przekształcenia.

**Parametr**

Transformation value - wartość przekształcenia
Określa nazwy dwóch wartości przekształcenia definiujących odległość, która ma być obliczona.

**Objaśnienie**

Wskazuje w milimetrach odległość (distance) pomiędzy dwoma ustawieniami. (Ustawienia mogą być wprowadzone w dowolnym momencie)

**Przykład**

| | |
|---|---|
| k=DISTANCE(HERE,part) | Oblicza odległość pomiędzy bieżącym punktem centralnym narzędzia (TCP), a ustawieniem “part” i zamienia wynik na k. |

## #PPOINT (jt1, jt2, jt3, jt4, jt5, jt6)

**Funkcja**

Wskazuje określone wartości przesunięcia osi.

**Parametr**

jt1
jt2
jt3
:
jt6

Określa (w stopniach dla osi obrotowych) wartość każdego kąta.

Jeśli nie określono, przyjmowane jest 0.

**Objaśnienie**

Niniejsza funkcja wskazuje określone wartości przesunięcia osi. Oznaczają one przesunięcie każdej osi, od joint (osi) 1 do ostatniej (niekoniecznie sześć)

[**UWAGA**]

Funkcja #PPOINT jest przetwarzana wyłącznie w wartościach przesunięcia osi. Z tego względu na jej początku zawsze wprowadzaj "#".

**Przykład**

W poniższym programie, joint (oś) 2 i 3 sześcioosiowego robota są przesuwane o określoną wartość z bieżącego ustawienia.

```
HERE    #ref                    Zapisuje w pamięci bieżące ustawienie.(#ref)
DECOMPOSE x[0]=#ref             Rozkłada każdy komponent na elementy zmiennej
                                tablicowej x[0],……x[5].

JMOVE #PPOINT (X[0], x[1]+a, x[2]-a/2, x[3], x[4], x[5])
```

W przypadku obu omawianych instrukcji otrzymujemy ustawienie jak w powyższym programie, z tym wyjątkiem, że obie osie nie poruszają się w tym samym czasie.

```
DRIVE   2, a, 100               Przesuń jt2 o a°.
DRIVE 3, -a/2, 100              Przesuń jt3 o – a/2°.
```

## SHIFT ( transformation values BY X shift, Y shift, Z shift )

**Funkcja**

Wskazuje wartości przekształcenia ustawienia przesuniętego o odległość określoną dla każdej osi podstawowej (X,Y,Z) od ustawienia określonego parametrem “transformation values” (wartości przekształcenia).

**Parametr**

1. Transformation values - wartości przekształcenia
Określa ustawienie, które ma być przesunięte w wartościach przekształcenia.

2. X shift (przesunięcie), Y shift (przesunięcie), Z shift (przesunięcie)
Określa wartości, które mają być dodane do każdego elementu przekształcenia określonego powyżej. W przypadku pominięcia dowolnej wartości, przyjmowane jest 0.

**Objaśnienie**

Wartości X shift, Y shift, Z shift są dodawane do każdego elementu X, Y, Z danych wartości przekształcenia. Otrzymany wynik jest prezentowany w wartościach przekształcenia.

**Przykład**

Jeśli przekształcenie “x” przyjmuje wartości (200,150,100,10,20,30), to

```
POINT   y=SHIFT(x BY 5, -5, 10)
```

“x” jest przesuwane o określoną wartość do (205,145,110,10,20,30) i te wartości są przypisywane zmiennej “y”.

## 9.3 FUNKCJE MATEMATYCZNE

| | |
|---|---|
| ABS | Oblicza wartość bezwzględną wyrażenia liczbowego. |
| SQRT | Oblicza pierwiastek kwadratowy wyrażenia liczbowego. |
| PI | Oblicza stałe π. |
| SIN | Oblicza wartość sinus. |
| COS | Oblicza wartość cosinus. |
| ATAN2 | Oblicza wartość arcus tangens. |
| RANDOM | Wskazuje losowo wybrany numer. |

**ABS ( real value ) (wartość rzeczywista)**

**SQRT ( real value ) (wartość rzeczywista)**

**PI**

**SIN ( real value ) (wartość rzeczywista)**

**COS ( real value ) (wartość rzeczywista)**

**ATAN2 ( real value1, real value 2 ) (wartość rzeczywista)**

**RANDOM**

| Keyword (słowo kluczowe) | Funkcja | Przykład |
|---|---|---|
| ABS | Oblicza wartość bezwzględną wyrażenia liczbowego. | ABS(value) (wartość) |
| SQRT | Oblicza pierwiastek kwadratowy wyrażenia liczbowego. | SQRT(value) (wartość) |
| PI | Oblicza stałe π(3.1415……). | PI |
| SIN | Oblicza wartość sinusa danego kąta. | SIN (wartość) |
| COS | Oblicza wartość cosinusa danego kąta. | COS (wartość) |
| ATAN2 | Oblicza wartość kąta (w stopniach), którego tangens równa się v1/v2. | ATAN2(v1,v2) |
| RANDOM | Wyświetla losowo wybrany numer w zakresie od 0,0 do 1,0. | RANDOM |

**Przykład**

| | |
|---|---|
| x=ABS(y) | zamienia wartość \|y\| na x |
| x=SQRT(y) | zamienia pierwiastek kwadratowy \|y\| na x |
| en=2*PI*r | zamienia wynik 2πr na en |
| Z=(SIN(x) ^ 2)+(COS(y) ^ 2) | zamienia wynik $(\sin(x))^2+(\cos(y))^2$ na z |
| slope=ATAN2(rise,run) | zamienia wynik $\text{tangent}^{-1}$(rise/run) na slope |
| r=RANDOM*10 | zamienia numer losowego wyboru od 0 do 10 na r |

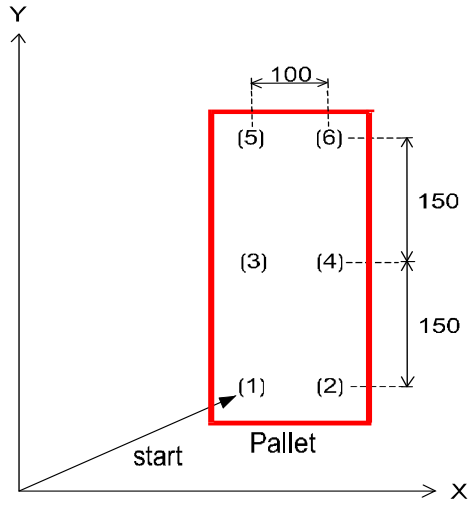
## 11.0 PRZYKŁADOWE PROGRAMY

W niniejszym rozdziale przedstawiono przykładowe programy wykorzystujące opisane wcześniej instrukcje i funkcje języka AS.

## 11.1 KONFIGURACJA POCZĄTKOWA PROGRAMÓW

Przeprowadzenie poniższych ustawień przed wykonaniem jakichkolwiek funkcji robota ułatwia programowanie.

· Przesuń robota do pozycji domowej (położenie i pozycja).
· Zdefiniuj dla każdego zadania niezbędne zmienne (np. w przypadku paletyzacji, ustaw numery części dla każdej palety).
· Zainicjuj licznik, flagę, itp.
· Ustaw dla zadania układ współrzędnych narzędzia.
· Ustaw dla zadania współrzędne globalne.




W powyższym przykładzie części są paletyzowane w kolejności od (1) do (6). W tym przypadku do konfiguracji początkowej powinien być użyty program jak poniżej. W tym przykładzie paleta jest ustawiona równolegle do współrzędnych globalnych robota.

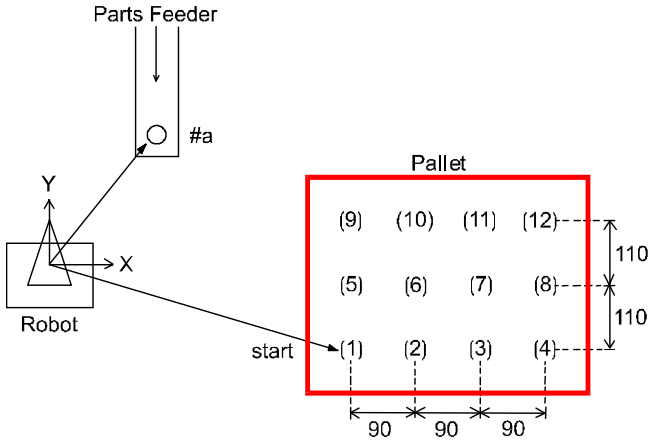
```
1  BASE   NULL ;określa układ współrzędnych globalnych robota (NULL)
2  TOOL   tool1          ;przekształcenie narzędzia (tool1)*
3  row.max=3             ;3 szeregi
4  col.max=2             ;2 kolumny
5  xs=100                ;ustawia odległość alokacji we współrzędnych X (ΔX=100mm).
6  ys=150                ;ustawia odległość alokacji we współrzędnych Y (ΔY=150mm).
7  POINT put=start       ;zastępuje wartość pose(1) - pozycji(1) wprowadzoną zmienną.
8  OPENI                 ;otwiera chwytak narzędzia.
9  HOME                  ;przesuwa do pozycji wyjściowej (home pose).**
```

Uwaga* Wartość przekształcenia narzędzia (tool 1) powinna być zdefiniowana przed rozpoczęciem przekształcania.

Uwaga ** Punkt początkowy (HOME) powinien być określony przed rozpoczęciem przesuwania.

## 11.2 PALETYZACJA

W niniejszym przykładzie części są zbierane z podajnika i umieszczane na paletach w trzech szeregach (w odległości 110 mm) i czterech kolumnach (w odległości 90 mm). W celu uproszczenia objaśnień, zarówno paleta, jak i części umieszczone na niej są ustawione równolegle do płaszczyzny XY współrzędnych globalnych robota. Pominięto także procedurę synchronizacji podajnika i robota z wykorzystaniem zewnętrznych sygnałów we/wy (instrukcje SWAIT, SIGNAL, itp.).




· Paleta jest ustawiona równolegle do płaszczyzny XY współrzędnych globalnych.
· Ustawienie #a (Podajnik części) i ustawienie “start” (gdzie umieszczona jest pierwsza część) muszą być określone przed wykonaniem programu.

Przykładowy program

```
        .PROGRAM palletize
;       konfiguracja początkowa (3 szeregi, 4 kolumny, ΔX=90, ΔY=110, itp.)
        row.max=3
        col.max=4
        xs=90
        ys=110
        SPEED 100 ALWAYS
        ACCURACY 100 ALWAYS
        POINT put=start
        OPENI
        ;
        ;       Start paletyzacji
        FOR row=1   TO   row.max
        FOR   col=1 TO col.max
        JAPPRO #a,100
        SPEED   30
        ACCURACY 1                      } Zbiera części z podajnika.
        LMOVE   #a
        CLOSEI
        LDEPART 200
        ;
        JAPPRO put, 200
        SPEED   30
        ACCURACY 1                      } Umieszcza części na palecie.
        LMOVE    put
        OPENI
        LDEPART 200
        ;
        ;       Oblicza ustawienia części w następnym szeregu.
        POINT put=SHIFT(put BY xs, 0,0)
        END
        ;
        ;       Oblicza ustawienia części w następnej kolumnie.
        POINT put=SHIFT(start by 0,ys*row, 0)
        END
        .END
```

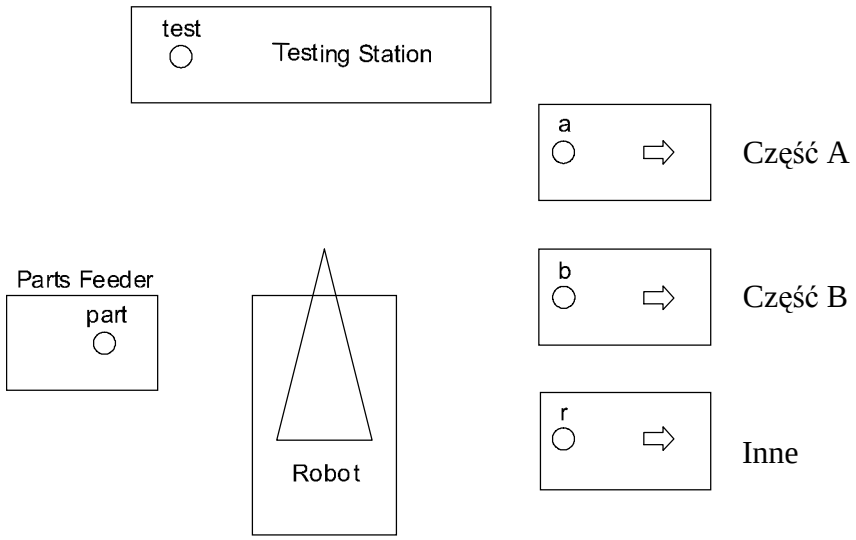
## 11.3 ZEWNĘTRZNA BLOKADA (INTERLOCKING)

W niniejszym przykładzie przedstawiono operację przeprowadzoną synchronicznie z urządzeniem zewnętrznym. Program ten wykorzystuje instrukcje: SIGNAL, IF, SWAIT, ONI, IGNORE.

1. W podajniku części (Parts Feeder) ustawiane są w kolejności losowej dwa rodzaje części (parts) - A i B. (Gotowy sygnał wejścia dla zestawu: IN1)
2. Robot zbiera części z podajnika i ustawia je na stacji badawczej (Testing Station). (Gotowy sygnał wyjścia dla zestawu:OUT1)
3. Na stacji badawczej części są klasyfikowane jako A, B oraz jako inne niż A lub B.
   Gotowy sygnał wejścia dla testu: IN2
   Klasyfikacja sygnału wejścia dla części: IN3, IN4
   (IN3, IN4) = (1, 0) : część A
   (IN3, IN4) = (0, 1) : część B
   (IN3, IN4) = (0, 0) lub (1, 1) : Inne
4. Robot umieszcza części zgodnie z klasyfikacją każdej z nich.

Jeśli w czasie, gdy robot zbiera część z podajnika i przenosi do stacji badawczej, pojawia się problem związany ze stacją badawczą, program jest natychmiast zatrzymywany i rozgałęziany do podprogramu standardowego do wykrywania i usuwania usterek. IN7 jest zewnętrznym sygnałem wejścia odpowiednim dla wystąpienia problemu. Sygnał IN6 jest wprowadzany po ukończeniu wykrywania i usuwania usterek, a robot wznawia wykonywanie natychmiast po wejściu sygnału.

Program służący do wykonywania powyższej operacji nosi nazwę MAIN, natomiast podprogram standardowy EMERGENCY.




Przykładowy program

```
; Określ zmienne
set.end  =1001          ;sygnał dla gotowego zestawu (IN1) nosi nazwę
set.end
test.end =1002          ;sygnał dla gotowego testu (IN2) nosi nazwę test.end
a.part   =1003          ;sygnał dla części A (IN3) nosi nazwę a.part
b.part   =1004          ;sygnał dla części B (IN4) nosi nazwę b.part
retry    =1006          ;sygnał dla rozwiązanego problemu (IN6) nosi nazwę
retry
fault=1007              ;sygnał dla problemu (IN7) nosi nazwę fault
test.start= 1           ;sygnał dla startu testu (IN2) nosi nazwę test.start

.PROGRAM main()
OPENI
10   JAPPRO   part,100
 ONI   fault CALL emergency    ;monitoruje pod kątem wystąpienia sygnału błędu i po
                               jego wykryciu przeskakuje do awaryjnego podprogramu
                               standardowego.
SWAIT set.end                  ;oczekuje na ustawienie części w podajniku.
LMOVEpart              ;przesuwa do części (Parts Feeder).
CLOSEI
LDEPART      100
JAPPRO       test,100          ;przenosi część do stacji badawczej (Testing Station).
LMOVEtest
BREAK
```

```
;
IGNORE          fault       ;zatrzymuje monitoring dla sygnału IN7 (fault - błąd).
SIGNAL          test.start          ;włącza sygnał test.start.
TWAIT 1.0
SWAIT test.end                      ;oczekuje do zakończenia testu.
JDEPART         100
SIGNAL          -test.start ;wyłącza sygnał test.start.
IF SIG(a.part,-b.part) GOTO 20      ;w przypadku części A, przeskakuje do label 20.
IF SIG(-a.part,b.part) GOTO 30      ;;w przypadku części B, przeskakuje do label 30.
POINT  n=r                          ;jeśli nie jest to A, ani B, przenosi część do r.
GOTO  40
20   POINT      n=a                 ;określa miejsce położenia części A.
GOTO  40
30   POINT      n=b                 ;określa miejsce położenia części B.
40   JAPPRO     n,100               ;przenosi część do jej ustawienia.
LMOVEn
OPENI
LDEPART         100
GOTO  10
.END

.PROGRAM emergency()
PRINT  "**ERROR**"                  ;wysyła komunikat błędu na terminal.
SWAIT retry                         ;oczekuje na rozwiązanie problemu.
ONI      fault CALL emergency       ;ponownie rozpoczyna monitoring pod kątem błędu.
RETURN                              ;powraca do głównego programu.
.END
```

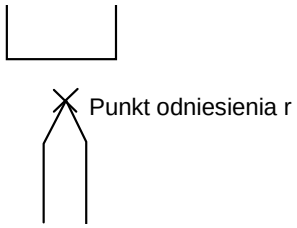
## 11.4 PRZEKSZATŁCENIA NARZĘDZIA

W sekcji tej wyjaśniono, w jaki sposób uzyskać wartości przekształcenia narzędzia i utworzyć, z ich wykorzystaniem, program.

### 11.4.1 WARTOŚCI PRZEKSZATŁCENIA NARZĘDZIA-1 (GDY ROZMIAR NARZĘDZIA NIE JEST ZNANY)

Gdy rozmiar narzędzia nie jest znany ze względu na jego nieporęczny kształt, wartości przekształcenia narzędzia można obliczyć w sposób przedstawiony poniżej. Oś Z współrzędnych globalnych jest ustawiona prostopadle do podłoża.

1. Wybierz przedmiot posiadający ostry punkt. Ustaw końcówkę przedmiotu w taki sposób, aby wskazywała prosto w górę. Punkt ten będzie stanowił punkt odniesienia (reference point) “r”.

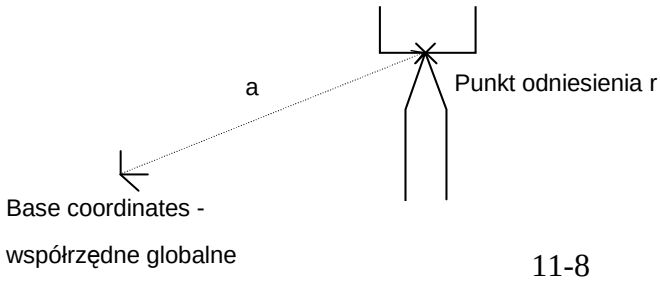



2. Przesuń robota krokowo w taki sposób, aby kołnierz robota był odwrócony w dół. Następnie wprowadź następujące polecenia w trybie odtwarzania:

```
>SPEED 10 ↵
>TOOL NULL ↵          ;ustawia współrzędne wyjściowe narzędzia.
>DO ALIGN  ↵           ;ustawia w linii oś narzędzia Z z osią podstawy Z.
```

3. Używając trybu podstawowego (Base Mode) przesuń krokowo robota w taki sposób, aby środek kołnierza był prostopadły w stosunku do punktu odniesienia. Następnie przesuń krokowo robota tylko wzdłuż współrzędnych globalnych X, Y, Z. Wprowadź jak poniżej, tak że wartości przekształcenia dla tego ustawienia są przypisane do zmiennej “a”:

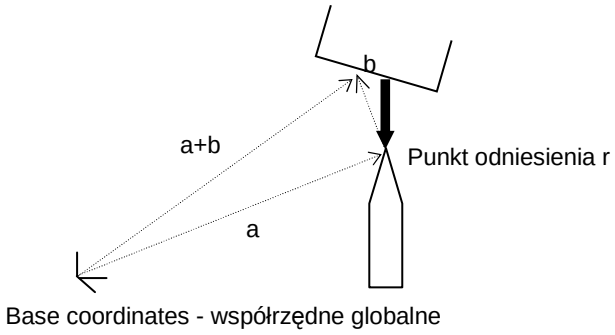
```
>HERE a ↵
```



4. Zamocuj narzędzie do kołnierza i przesuń krokowo punkt centralny narzędzia (TCP) do punktu odniesienia w taki sposób, aby oś Z nowego układu współrzędnych narzędzia była prostopadła do osi X i Y współrzędnych globalnych.

Wprowadź jak poniżej, aby wyuczyć wartości przekształcenia tego ustawienia jako wartości złożone “a+b”:

```
>HERE a+b ↵
```




5. Na podstawie tych wartości złożonych, wartości przekształcenia narzędzia mogą być odczytane jako “-b”. Wprowadź:

```
>POINT t=-b ↵
```

Przypisuje to wartość –b do zmiennej t .

6. Określ przekształcenie narzędzia jako t.

```
>TOOL t ↵
```

7. Aby sprawdzić, wprowadź jak poniżej:

```
>DO JMOVE r ↵
```

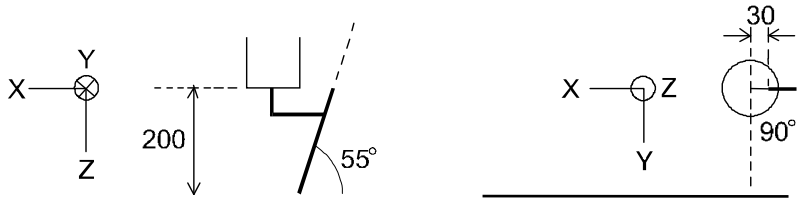
Końcówka narzędzia powinna przesunąć się do punktu odniesienia r.

Po określeniu przekształcenia narzędzia, aż do jego zmiany, wszystkie działania opierają się na nim.

### 11.4.2 WARTOŚCI PRZEKSZATŁCENIA NARZĘDZIA-2 (GDY ROZMIAR NARZĘDZIA JEST ZNANY)

Gdy rozmiar narzędzia jest znany, wartości przekształcenia narzędzia można uzyskać w sposób przedstawiony poniżej. Wartości określone w omawianej procedurze są zasadniczo

bardziej precyzyjne od tych uzyskanych w poprzedniej procedurze. (Patrz wyżej 11.4.1)




Osie XYZ, na rysunku powyżej, wyrażają współrzędne wyjściowe narzędzia. Poniższa procedura ustawia punkt początkowy układu współrzędnych narzędzia na dziobie palnika (torch tip) a oś Z w tym samym kierunku co palnik (torch).

(1)Określ zmienną przekształcenia narzędzia “torch”, używając polecenia POINT:

```
>POINT torch ↵
  X       Y       Z       O       A       T
  0       0       0       0       0       0

Change
>-30, 0, 200, 0, 35, 0 ↵
  X       Y       Z       O       A       T
-30       0     200       0      35       0
Change ↵
>
```

(2) Określ wartość przekształcenia narzędzia, używając zmiennej “torch”.

```
>TOOL torch ↵
```

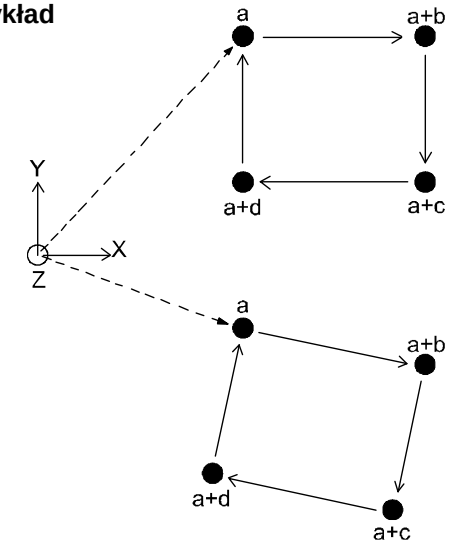
## 11.5 POZYCJE WZGLĘDNE

### 11.5.1 WYKORZYSTANIE USTAWIEŃ (POZYCJI) WZGLĘDNYCH

Ustawienie może być określone względem punktu odniesienia. Dokonując konfiguracji w ten sposób, relacja pomiędzy ustawieniem, a punktem odniesienia pozostaje stała, nawet po ponownym zdefiniowaniu punktu odniesienia.

Np., po wyuczeniu czterech rogów stołu, relacja pozycji pomiędzy robotem, a stołem zmienia się w zależności od ich położenia. Jednak dopóki kształt stołu pozostaje niezmieniony, relacje czterech rogów pozostają niezmienione. Dlatego też, jeśli jeden z rogów jest wyuczony jako punkt odniesienia dla określania relacji w pozycjonowaniu bezwzględnym pomiędzy robotem a stołem, a pozostałe rogi są wyuczone w odniesieniu do pierwszego rogu; jeśli stół zostanie przeniesiony, ponownie zdefiniowany będzie musiał być tylko punkt odniesienia.

**Przykład**




Uczenie

>HERE a ↵

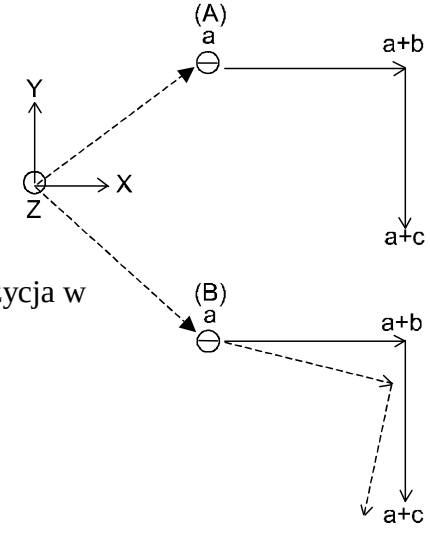
>HERE a+b ↵

>HERE a+c ↵

>HERE a+d ↵

Program

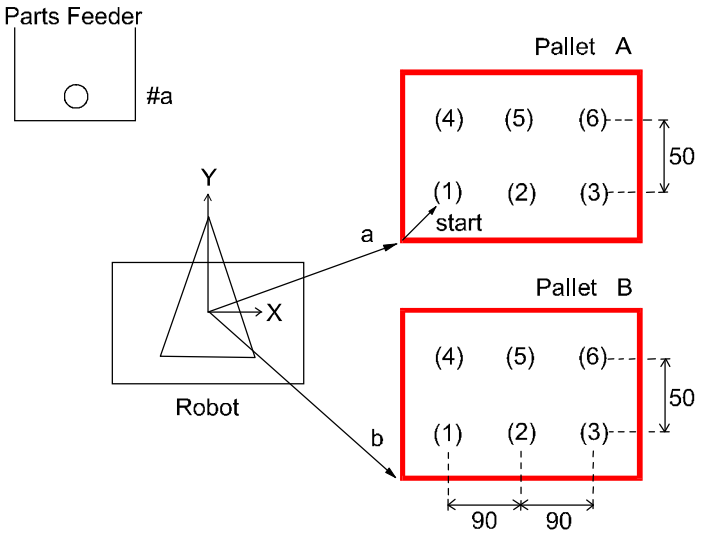
```
JMOVE   a
LMOVE   a+b
LMOVE   a+c
LMOVE   a+d
LMOVE   a
```




Na rysunku (A) po prawej punkt odniesienia a i złożone wartości przekształcenia innych rogów są wyuczone. Następnie na rysunku (B) punkt odniesienia jest określony ponownie. Jeśli pozycja kiści robota nie jest ustawiona ponownie (tj. pozostaje taka sama, jak pozycja w (A)), robot będzie się poruszał po trajektorii linii ciągłej. Jeśli robot ma poruszać się wzdłuż linii punktowej, niezbędne jest ponowne zdefiniowanie pozycji i położenia.

## 11.5.2 PRZYKŁAD PROGRAMU WYKORZYSTUJĄCEGO POZYCJE WZGLĘDNE

W tym przykładzie części są paletyzowane jak w poprzednim, z tym że wykorzystywane są dwie palety. Palety są umieszczane osobno, jednak relacja pomiędzy punktem odniesienia a miejscami, w których mają być położone palety są takie same na każdej z palet. Operacja ta ustawia części z podajnika (Parts Feeder) na palecie (Pallet) A. Po ustawieniu sześciu części, robot wykonuje to samo w odniesieniu do palety B. (Pominięto procedurę synchronizacji z podajnikiem części).




Ustawienia do wyuczenia

#a : ustawienie, w którym robot zbiera części z podajnika

a : ustawienie odniesienia na palecie A

b : ustawienie odniesienia na palecie B

start : ustawienie pierwszej części na palecie, w odniesieniu do punktu odniesienia

Przykładowy program

```
.PROGRAM    relative.test
;           Konfiguracja początkowa (2 szeregi, 3 kolumny, ΔX=90, ΔY=50, itp.)
row.max=2
col.max=3
xs=90
ys=50
OPENI
flg=0                                          ; flg=0 : Pallet A, flg=1 : Pallet B
POINT    pallet=a
;           start paletyzacji
10          POINT    put=start
FOR    ro w1 TO row.max
FOR    col=1 TO col.max
JAPPRO #a,100
LMOVE    #a                                    zbiera części z podajnika.
CLOSEI
LDEPART 100
POINT     put_pt=pallet+put
JAPPRO    put pt,200
LMOVE     put pt                               umieszcza części na palecie
OPENI
LDEPART 200

POINT put=SHIFT(put BY xs,0,0)          ;odnajduje położenie części w następnej
kolumnie.
END
POINT put=SHIFT(start BY 0,ys*row,0)    ;odnajduje położenie części w następnej kolumnie
END
IF    flg<>0 GOTO 30     ; przejście do procedury końcowej po zakończeniu palety B (flg=1)
flg=1
POINT pallet=b            ;określa ustawienie odniesienia palety B.
GOTO 10
30          TYPE "*** end ***"
STOP
.END
```

## 11.6 POZYCJE WZGLĘDNE WYKORZYSTUJĄCE FUNKCJĘ FRAME

W przykładzie z sekcji 11.5.1, podczas ponownego definiowania ustawienia odniesienia, pozycja kiści musi być poprawna. Jeśli wykorzystywana jest funkcja FRAME, nie jest to konieczne. Aby zdefiniować wartość przekształcenia układu współrzędnych, przeprowadź uczenie czterech punktów (b, c, d, e). Punkty b i c określają kierunek osi X, trzeci punkt d określa płaszczyznę XY, a punkt e punkt początkowy. Po przeprowadzeniu uczenia punktów, wprowadź następujące polecenia:

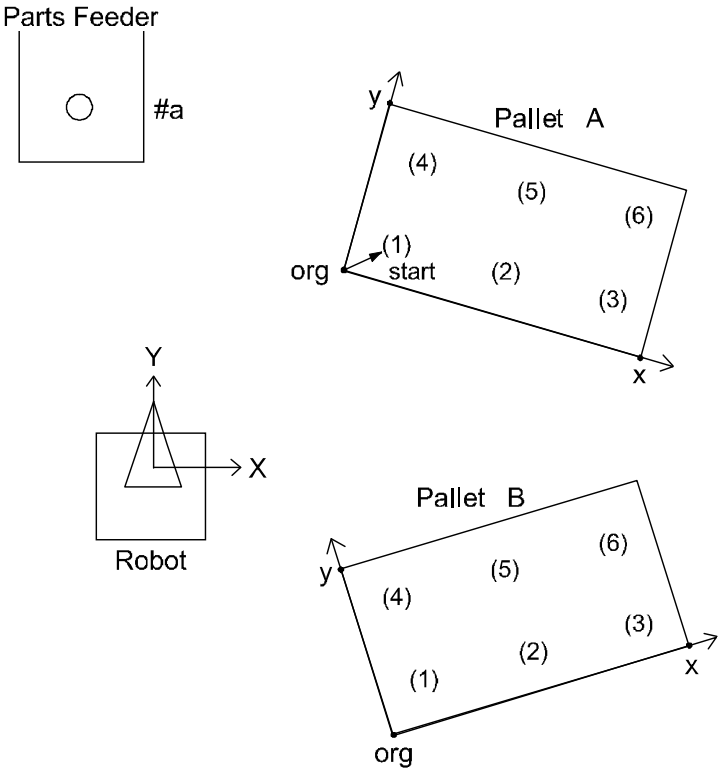
```
POINT a=FRAME(b,c,d,e)
```

Następnie, jako zmienna "a", definiowane są względne współrzędne. Wartości XYZ określają pozycję punktu początkowego współrzędnych względnych, natomiast wartości kątów OAT (O, A, T = Orientation, Azimuth, Tool) wskazują pozycję współrzędnych względnych.

Wszystkie ustawienia dokonane na podstawie współrzędnych względnych będą dalej wyrażane jako ustawienie a+□. Jeśli położenie palety zmienia się, ponownie przeprowadź uczenie b, c, d, e, aby zdefiniować a w identyczny sposób jak powyżej.

Współrzędne względne zdefiniowane przy pomocy funkcji FRAME są także nazywane współrzędnymi FRAME.

W poniższym programie przykładowym, operacja identyczna jak przedstawiona w sekcji 11.5.2 jest przeprowadzana z wykorzystaniem współrzędnych FRAME.




Pierwsza procedura służy paletyzacji części na palecie A. Przeprowadzane jest uczenie trzech narożników na palecie, jednego jako punktu początkowego, następnego jako punktu na osi X a kolejnego jako punktu na osi Y (patrz rysunek powyżej). Wykonaj poniższy program, aby przeprowadzić paletyzację na palecie A ( zauważ, że po określeniu punktów, pozostała część programu jest identyczna z poprzednim programem przykładowym). Aby wykonać paletyzację na palecie B, ponownie przeprowadź uczenie trzech narożników i wykonaj ten sam program. Współrzędne FRAME zostaną zdefiniowane ponownie, a części będą paletyzowane na palecie B i A.

Przykładowy program

```
.PROGRAM    frame.test
;               Konfiguracja początkowa (2 szeregi, 3 kolumny, ΔX=90, ΔY=50, itp.)
row.max=2
col.max=3
xs=90
yx=50
OPENI
;
POINT pallet=FRAME(org,x,y,org)             ;określa współrzędne FRAME palety
;                                       (3 punkty: dla punktu początkowego, dla osi X/Y)
POINT    put=start                      ;          Start paletyzacji
```

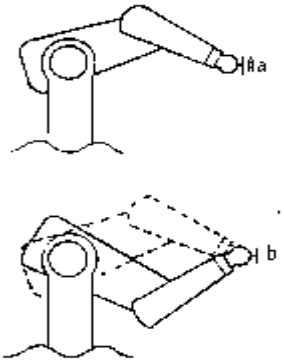
```
FOR row=1   TO   row.max
FOR   col=1 TO col.max
JAPPRO #a,100
LMOVE   #a
CLOSEI                                  Zbiera części z palety.
LDEPART 100
;
POINT   put_pt=pallet+put
JAPPRO put_pt,200
LMOVE   put_pt                          Umieszcza części na palecie.
OPENI
LDEPART 200
;
POINT put=SHIFT(put BY xs,0,0)          ;odnajduje położenie części w następnej kolumnie.
END
;
POINT    put=SHIFT(startBY   0,ys*row,0)    ;odnajduje położenie części w następnej
kolumnie.
END
STOP
.END
```

## 11.7 USTAWIANIE KONFIGURACJI RAMIENIA ROBOTA

Większość robotów posiada sześć osi i kiedy ustawienie jest uczone z wykorzystaniem wartości przesunięcia osi, podawana jest wartość przesunięcia każdej z sześciu osi, tak że ustawienie robota jest określane w sposób niepowtarzalny. Z drugiej strony, kiedy ustawienie jest określane z wykorzystaniem wartości przekształcenia, w niektórych przypadkach, w zależności od konfiguracji ramienia robota, więcej niż jeden zestaw wartości osi daje to samo ustawienie określone przez jedną wartość przekształcenia. W systemie AS robot, zasadniczo, zachowuje konfigurację poprzedniego działania i nie jest konieczna zmiana konfiguracji robota. Jednakże, w poniższych przypadkach, konfiguracja robota powinna być określona przez instrukcję konfiguracji:

1. Kiedy robot porusza się z punktu, posiadając niejasną konfigurację, do punktu wyuczonego przez wartości przekształcenia.
2. Gdy piąta oś (oś zgięta) przechodzi przez punkt początkowy (0°) w konfiguracji kiści SBS. (SBS: swivel - obrót, bend - zgięcie, swivel - obrót)

Np. na rysunku po prawej,
jeśli ustawienie #a jest określane konfiguracją ABOVE,
rezultatem instrukcji JMOVE b będzie
ABOVE (linia punktowa na rysunku), nawet jeśli ustawienie b jest jest początkowo określone BELOW.

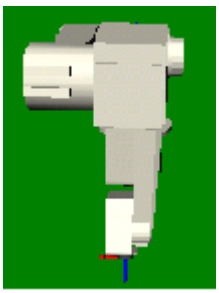



```
JMOVE   #a
JMOVE   b
```

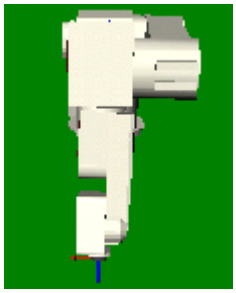
W ten sam sposób, jeśli #a jest określane UWRIST (JT5>0),
konfiguracja w ustawieniu b będzie UWRIST, niezależnie
od konfiguracji z momentu przeprowadzania uczenia ustawienia.

Aby rozwiązać ten problem, musisz zmienić konfigurację robota w czasie jego ruchu. Zrób to wykonując instrukcję konfiguracji po zakończeniu instrukcji ruchu osi w punkcie określonym wartościami przekształcenia (instrukcje ruchu z interpolacją osiową: JMOVE, JAPPRO, JDEPART, DRIVE itp.). Wymieniono tutaj sześć instrukcji konfiguracji, w oknie uwagi na następnych stronach przedstawiono także przykładowy program.

LEFTY, RIGHTY
Ustawiają pozycję pierwszych trzech osi (JT1, JT2, JT3) robota. LEFTY określa konfigurację ramienia przypominającą lewe ramię człowieka, RIGHTY prawe ramię człowieka.



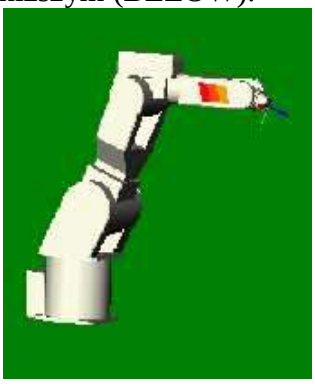
LEFTY



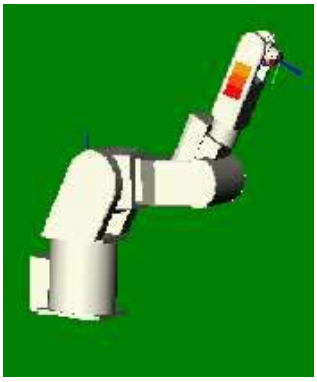
RIGHTY

ABOVE, BELOW

Ustawia robota w takiej konfiguracji, że trzecia oś (JT3) znajduje się w wyższym położeniu (ABOVE) lub w niższym (BELOW).



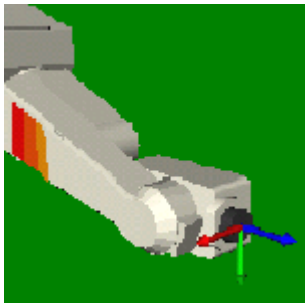
ABOVE



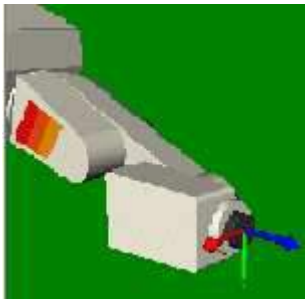
BELOW

UWRIST, DWRIST

Ustawia robota w takiej konfiguracji, że wartość piątej osi (JT5) jest dodatnia (UWRIST) lub ujemna (DWRIST) przy tej samej pozycji powierzchni kołnierza narzędzia.



UWRIST



DWRIST

[**UWAGA**]

1. Zasadniczo, instrukcja konfiguracji nie wpływa na wartości przesunięcia osi (ustawienia posiadające nazwy z #) i robot przesuwa się do wyuczonego położenia w wyuczonej pozycji. Jednakże, skutkuje to błędem podczas ruchu robota z interpolacją liniową pomiędzy ustawieniami, gdzie konfiguracja na początku różni się od konfiguracji punktu docelowego.

2. Robot nie reaguje w sposób natychmiastowy na instrukcję konfiguracji. Konfiguracja zmienia się podczas wykonywania ruchu z interpolacją osiową (JMOVE, JAPPRO, JDEPART, DRIVE, itp.).

3. W regularnych programach, konfiguracja nie musi być zmieniana za wyjątkiem, gdy wymaga tego określony cel. Instrukcje konfiguracji są wykorzystywane w następujących przypadkach:

(1) Gdy program nie zaczyna się od instrukcji ruchu przesuwającej robota do ustawienia określonego przez wartości przesunięcia osi, instrukcja konfiguracji powinna być zapisana na początku programu, aby określić pozycję robota.

(2) Gdy instrukcja JAPPRO jest używana w podany niżej sposób, należy wykorzystać instrukcję konfiguracji:

```
JMOVE   a
JAPPRO  #b,100
JMOVE   #b
```

Pozycja następująca po instrukcji ruchu "JAPPRO #b,100" może, zgodnie z konfiguracją poprzedniego ruchu, różnić się od pozycji w #b. Jeśli pozycja kiści (± kąta JT5) jest odmienna, wykonanie kolejnego kroku "JMOVE #b", może powodować znaczną rotację osi JT4 i JT6. Aby tego uniknąć, musisz wykonać uczenie ustawienia 100 mm powyżej ustawienia #b jako #bb i użyć instrukcji JMOVE.

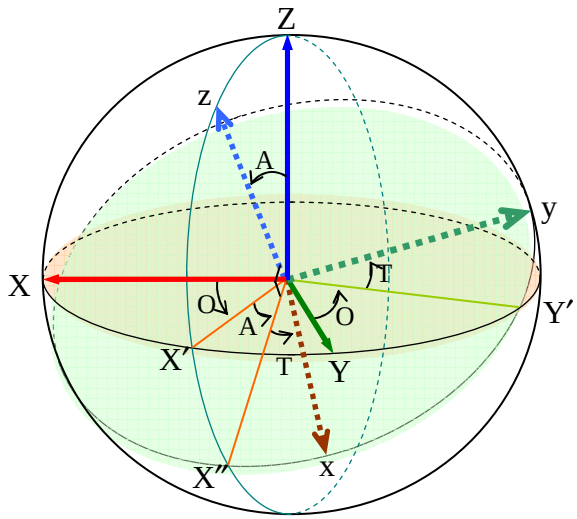
```
JMOVE   a
JMOVE   #bb
JMOVE   #b
   :
```

Innym sposobem na uniknięcie odchylenia na osiach JT4 i JT6 jest określenie kierunku kiści, z wykorzystaniem instrukcji konfiguracji. W tym przykładzie, w odniesieniu do konfiguracji w #b zakłada się wartość UWRIST (wartość osi JT5 jest dodatnia).

```
JMOVE   a
UWRIST
JAPPRO  #b,100
JMOVE   #b
   :
```

Pozycja kiści osi zmienia się na UWRIST po wykonaniu "JAPPRO #b, 100", unikając w ten sposób niepotrzebnej rotacji osi 4 i 6 podczas wykonania "JMOVE #b".

## ZAŁĄCZNIK 5 KĄTY EULERA (O, A, T = ORIENTATION, AZIMUTH, TOOL)




Pozycja układu współrzędnych Σ ( x,y,z ) w odniesieniu do układu współrzędnych globalnych Σ ( X,Y,Z ) jest zazwyczaj wyrażana przy pomocy kątów Eulera (O, A, T = Orientation, Azimuth, Tool). Jak zaprezentowano na powyższym rysunku, trzy kąty mogą być określone w następujący sposób. Na powyższym rysunku dwa układy współrzędnych

Σ ( x,y,z ) oraz Σ ( X,Y,Z ) posiadają wspólny punkt początkowy.

O: Kąt pomiędzy płaszczyzną Zz a XZ
A: Kąt pomiędzy płaszczyzną osią z a Z
T: Kąt pomiędzy osią x a osią X″
Oś X″ znajduje się na płaszczyźnie Zz, natomiast kąt pomiędzy tą osią a osią z wynosi 90°.

Omawiane trzy kąty stanowią równocześnie kąty ruchów obrotowych układu współrzędnych globalnych Σ(X,Y,Z) niezbędnych do osiągnięcia zbieżności z układem współrzędnych Σ(x,y,z). Aby uzyskać omawiany wynik, kolejność ruchów obrotowych nie może zostać zmieniona.

1. Ruch obrotowy O układu współrzędnych Σ(X,Y,Z) wokół osi Z. (Przesuwa Σ(X,Y,Z) do Σ(X′,Y′,Z).)
2. Ruch obrotowy A układu współrzędnych Σ(X′,Y′,Z) wokół ′ osi Y. (Przesuwa Σ(X′,Y′,Z) do Σ(X″,Y′,z).)
3. Ruch obrotowy T układu współrzędnych Σ(X″,Y′,z) wokół osi z. (Przesuwa Σ(X″,Y′,z) do Σ(x,y,z).)

Ponadto, omawiane zagadnienie może być rozpatrywane w kategoriach współrzędnych biegunowych. Jeśli punkt P, znajdujący się na osi z w odległości d od punktu początkowego, jest zapisany jako (d, A, O), to O i A w wartościach oznaczanych przez ten układ współrzędnych równa się opisanym powyżej O i A. Kierunek osi z jest wyrażany przez te dwie wartości.

Kontroler serii D
PROGRAMOWANIE W JĘZYKU AS

Sierpień 2002: 1 edycja
Styczeń 2007: 4-ta Edycja

Publikacja: KAWASAKI HEAVY INDUSTRIES, LTD.

90209-1017DED-PL