Programowanie robota IRb-1400

Paweł Ludwików

6 kwietnia 2005 roku

Spis treści

1 Język RAPID

1.1 Przegląd instrukcji

`For` powtarzanie ciągu instrukcji zadaną ilość razy

`If` wyrażenie warunkowe

`komentarz` wstawienie komentarza w program

`MoveC` przesuwa efektor po łuku

`MoveJ` przesuwa efektor we współrzędnych przegubowych

`MoveL` przesuwa efektor liniowo.

`Reset` ustawienie wyjścia cyfrowego na zero

`Set` ustawienie wyjścia cyfrowego na jeden

`While` powtarzanie ciągu instrukcji

1.2 Opis instrukcji

1.2.1 For

For Wykonanie ciągu instrukcji zadaną ilość razy

`FOR` jest używane, jeśli jedna lub wiele instrukcji powinno być powtórzonych ustaloną liczbę razy.

Jeśli instrukcje powinny być powtarzane tak długo jak spełniony jest podany warunek, należy użyć instrukcji `WHILE`

Przykład

```
FOR i FROM 1 TO 10 DO
  proc1;
ENDFOR
```

Wykonuje procedurę `proc1` 10 razy.

Argumenty

`For licznik FROM początek TO koniec [STEP krok] DO ... ENDFOR`

`licznik` Typ danych: identyfikator

Nazwa zmiennej która będzie przechowywać aktualną wartość licznika pętli. Zmienna jest tworzona automatycznie i nazwa powinna być unikalna (nie zdeklarowana wcześniej w programie).

`początek` Typ danych: liczba

Początkowa wartość licznika (przeważnie wartość całkowita).

`koniec` Typ danych: liczba

Końcowa wartość licznika (przeważnie wartość całkowita).

`krok` Typ danych: liczba

Wartość o którą licznik jest zwiększany (lub zmniejszany) przy każdym obiegu pętli (przeważnie wartość całkowita).

Domyślnie przyjmowana na 1 (lub -1 jeśli wartość `początek` > `koniec`)

Przykład

```
FOR i FROM 10 TO 2 STEP -1 DO
  a{i}:=a{i-1};
ENDFOR
```

Powoduje przesunięcie wartości w tablicy tak że `a{10}:=a{9}, a{9}:=a{8},` itd.

Wykonanie programu

1. obliczenie wartości `początek`, `koniec` i `krok`,
2. do licznika wpisywana jest wartość początkowa,
3. następuje sprawdzenie, czy wartość licznika mieści się pomiędzy wartościami `początek` i `koniec`. Jeżeli tak nie jest, pęta `FOR` zostaje opuszczona i program kontynuuje wykonywanie instrukcji umieszczonej po `ENDFOR`,
4. wykonywane są instrukcje wewnątrz pętli,
5. licznik pętli jest zwiększany (zmniejszany) o wartość `krok`,
6. pętla jest powtarzana, zaczynając od punktu 3.

Ograniczenia

Licznik pętli jest definiowany lokalnie w pętli `FOR` i w związku z tym przesłania inne zmienne lub procedury z tą samą nazwą. Wartość licznika może jedynie być odczytywana (nie można przestawiać wartości) przez instrukcje wewnątrz pętli.

1.2.2 If

Jeśli warunek jest spełniony, wtedy ...; w przeciwnym przypadku ... IF jest używana, jeśli różne instrukcje powinny być wykonane w zależności czy warunek jest spełniony czy nie.

Przykłady

```
IF reg1>5 THEN
  Set  do1;
  Set  do2;
ENDIF
```

Sygnały do1 i do2 zostaną ustawione jeśli reg1 jest większy od 5.

```
IF reg1>5 THEN
  Set  do1;
  Set  do2;
ELSE
  Reset  do1;
  Reset  do2;
ENDIF
```

Sygnały do1 i do2 zostaną ustawione lub skasowane w zależności od tego czy reg1 jest większy od 5 czy nie.

Argumenty

```
IF warunek THEN ...
  {ELSIF warunek THEN ...}
[ELSE ...]
ENDIF
```

warunek Typ danych: bool

Warunek, który musi być spełniony aby wykonały się instrukcje pomiędzy IF a ELSE/ELSIF.

Przykład

```
IF licznik > 100 THEN
  licznik:=100;
ELSIF licznik < 0 THEN
  licznik:=0;
ELSE
  licznik:=licznik+1;
ENDIF
```

Wartość `licznik` jest zwiększona o 1. W przypadku, gdy jest poza granicami 0–100, zostanie ustawiony na odpowiedniej wartości skrajnej.

Wykonanie

Warunki są testowane w kolejności występowania, do momentu aż jeden zostanie spełniony. Programu przechodzi do wykonywania instrukcji związanych z tym warunkiem. Jeśli żaden z warunków nie był spełniony, program wykonuje fragment za instrukcją `ELSE`. Jeżeli więcej niż jeden warunek jest spełniony zostaną wykonane jedynie instrukcje związane z pierwszym z warunków.

1.2.3 komentarz

Komentarze są używane do uczynienia programu bardziej przejrzystym dla piszącego, nie mają wpływu na proces wykonywania programu

Przykład

```
!Przyjmij pozycje nad paleta
```

Komentarz jest wstawiony do programu, w celu ułatwienia zrozumienia wykonywanych operacji.

Argumenty

`! komentarz` Typ danych: ciąg znaków

Dowolny tekst

Wykonanie

Nic nie jest zmieniane przez tą instrukcję.

1.2.4 MoveC

MoveC Przemieszczenie efektora po łuku.

Instrukcja `MoveC` jest używana do przemieszczenia środka narzędzia (TCP *ang. Tool Center Point*) po wycinku okręgu do wskazanego punktu. W trakcie manewru orientacja jest niezmienna względem łuku.

Przykłady

```
MoveC p1, p2, v500, z30, tool2;
```

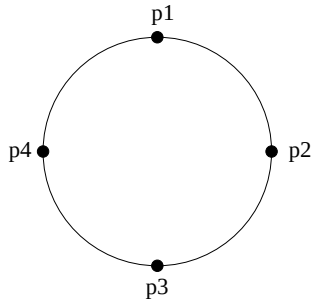
Środek narzędzia (TCP) `tool2` jest przesunięty do pozycji `p2` z prędkością `v500` i strefą tolerancji `z30`. Okrąg jest zadany przy użyciu punktu rozpoczęcia manewru, punktu pośredniego `p1` i końcowego `p2`.

```
MoveC *, *, v500 \T:=5, fine, grip3;
```

TCP narzędzia `grip3` jest przesunięty do pozycji zapamiętanej w instrukcji (oznaczonej przez drugi znak `*`). Punkt pośredni okręgu jest również zapisany w instrukcji (pierwszy znak `*`). Cały manewr potrwa 5 sekund.

```
MoveL p1, v500, fine, tool1;
MoveC p2, p3, v500, z20, tool1;
MoveC p4, p1, v500, fine, tool1;
```

Powyższe instrukcje spowodują zakreślenie pełnego okręgu (przy założeniu, że punkty są identyczne z tymi pokazanymi na rysunku 1)



Rysunek 1: Pełny okrąg złożony z dwóch instrukcji `MoveC`

Argumenty

```
MoveC [\Conc] CirPoint ToPoint Speed [\V] | [\T] Zone [\Z] Tool [\WObj] [\Corr]
```

`[\Conc]` (Concurrent) Typ danych: przełącznik

Następujące po sobie instrukcje zostaną wykonane jednocześnie. Dzięki temu można skrócić czas wykonywania instrukcji w przypadku, gdy synchronizacja z zewnętrznymi urządzeniami nie jest wymagana.

Maksymalna ilość po sobie występujących instrukcji przesunięć wykorzystujących `\Conc` jest ograniczona do 5. Jeśli owa sekcja zawiera `StorePath-RestoPath`, wówczas nie jest możliwe użycie `\Conc`.

`CirPoint` Typ danych: robtarget

Punkt pośredni okręgu. Jest to pozycja na okręgu pomiędzy punktem początkowym a końcowym. Aby osiągnąć dużą powtarzalność ruchu, punkt ten powinien leżeć mniej więcej w połowie odcinka pomiędzy początkiem i końcem. Jeśli jest położony zbyt blisko początku lub końca, robot może wyświetlić ostrzeżenie. Punkt można podać jako nazwaną pozycję (np. `p10`) albo zapisać bezpośrednio w instrukcji (wówczas zaznaczony jest jako `*`).

`ToPoint` Typ danych: robtarget

Punkt końcowy wycinka okręgu. Podaje się jako nazwaną pozycję lub definiuje bezpośrednio w instrukcji (oznaczony jako `*`).

`Speed` Typ danych: speeddata

Ustawienia prędkości TCP, prędkości zmiany orientacji oraz zewnętrznych osi.

`[\V]` (Velocity) Typ data: liczba

Dzięki temu argumentowi można bezpośrednio w instrukcji podać prędkość TCP wrażoną w mm/s.

`[\T]` (Time) Typ danych: liczba

Określa czas wykonywania ruchu w sekundach.

`Zone` Typ danych: zonedata

Ustawienia dokładności pozycjonowania.

`[\Z]` (Zone) Typ danych: liczba

Argument podawany celem bezpośredniego określenia dokładności położenia TCP. Długość krawędzi narożnej podawana jest w milimetrach.

`Tool` Typ danych: tooldata

Używane narzędzie. Środek narzędzia jest punktem, który osiąga zadane położenie.

`[\WObj]` (Work Object) Typ danych: wobjdata

Układ współrzędnych względem którego opisywane jest położenie robota. Ta opcja może być pominięta i wtedy przyjmowany jest globalny (world) układ współrzędnych.

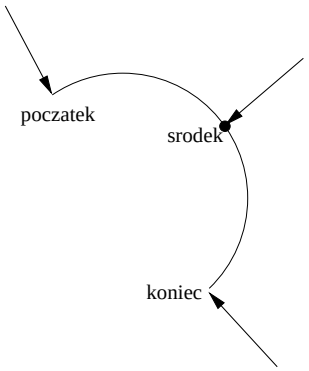
`[\Corr]` (Correction) Typ danych: przełącznik

Dane korekcyjne zapisane przez instrukcję `CorrWrite` zostaną dodane do ścieżki i punktu docelowego, jeśli argument jest obecny.

Wykonanie programu

Robot i zewnętrzne jednostki są przemieszczane do punktu docelowego w następujący sposób:

- środek narzędzia (TCP) narzędzia przesuwa się po wycinku okręgu ze stałą, zaprogramowaną prędkością,
- zmiana orientacji narzędzia odbywa się ze stałą prędkością począwszy od ustawienia przy rozpoczęciu manewru i kończąc na ustawieniu w punkcie docelowym,



Rysunek 2: Orientacja narzędzia podczas ruchu po okręgu

- reorientacja jest przeprowadzana względem kolistej ścieżki,
- orientacja w punkcie pośrednim nie jest sprawą krytyczną; używana jest jedynie do wyboru jednego z dwóch możliwych kierunków ruchu wzdłuż okręgu. Dokładność zmian orientacji podczas ruchu zależy jedynie od punktu początkowego i końcowego,
- nieskoordynowane osie zewnętrzne poruszają się ze stałymi prędkościami dobranymi automatycznie tak, aby osiągnąć punkty docelowe w tym samym czasie co robot. W tym przypadku pozycja pośrednia jest niewykorzystana.

Jeżeli niemożliwe jest utrzymanie zaprogramowanej prędkości reorientacji lub osi zewnętrznych, prędkość całego manewru zostaje obniżona.

Ograniczenia

Nie można wykonać zmiany trybu wykonywania programu (przód/tył lub na odwrót) wykonana gdy robot został zatrzymany podczas wykonywania manewru na kolistej ścieżce.

Punkt początkowy ruchu dla instrukcji `MoveC` (ani żadnej innej powodującej ruch po okręgu) nie powinien mieścić się pomiędzy punktem pośrednim a końcowym. Robot nie wykona manewru, jeśli tak się stanie.

1.2.5 MoveJ

MoveJ Przemieszczenie efektora we współrzędnych przegubowych.

`MoveJ` jest używana celem szybkiego przemieszczenia robota z jednego punktu do drugiego, kiedy ruch narzędzia nie musi być linią prostą. Robot i zewnętrzne osie poruszają się po nieliniowej ścieżce w kierunku celu. Prędkości ruchu są dobrane tak aby wszystkie osie osiągnęły położenie końcowe w tej samej chwili.

Przykłady

```
MoveJ p1, vmax, z30, tool2;
```

Narzędzie (TCP) `tool2` zostanie przesunięte wzdłuż nieliniowej ścieżki do położenia `p1`, z prędkością `vmax` oraz strefą tolerancji `z30`.

```
MoveJ *, vmax \T:=5, fine, grip3;
```

TCP narzędzia `grip3` zostanie przesunięty po nieliniowej ścieżce do punktu zapamiętanego w instrukcji (oznaczonego znakiem `*`). Ruch potrwa 5 sekund.

Argumenty

```
MoveJ [\Conc] ToPoint Speed [\V] | [\T] Zone [\Z] Tool [\WObj]
```

`[\Conc]` (Concurrent) Typ danych: przełącznik

Następujące po sobie instrukcje zostaną wykonane jednocześnie. Dzięki temu można skrócić czas wykonywania instrukcji w przypadku, gdy synchronizacja z zewnętrznymi urządzeniami nie jest wymagana.

Maksymalna ilość po sobie występujących instrukcji przesunięć wykorzystujących `\Conc` jest ograniczona do 5. Jeśli owa sekcja zawiera `StorePath-RestoPath`, wówczas nie jest możliwe użycie `\Conc`.

`ToPoint` Typ danych: robtarget

Punkt końcowy ruchu. Podaje się jako nazwaną pozycję lub definiuje bezpośrednio w instrukcji (oznaczony jako `*`).

`Speed` Typ danych: speeddata

Ustawienia prędkości TCP, prędkości zmiany orientacji oraz zewnętrznych osi.

`[\V]` (Velocity) Typ data: liczba

Dzięki temu argumentowi można bezpośrednio w instrukcji podać prędkość TCP wrażoną w mm/s.

`[\T]` (Time) Typ danych: liczba

Określa czas wykonywania ruchu w sekundach.

`Zone` Typ danych: zonedata

Ustawienia dokładności pozycjonowania.

`[\Z]` (Zone) Typ danych: liczba

Argument podawany celem bezpośredniego określenia dokładności położenia TCP. Długość krawędzi narożnej podawana jest w milimetrach.

`Tool` Typ danych: tooldata

Używane narzędzie. Środek narzędzia jest punktem, który osiąga zadane położenie.

`[\WObj]` (Work Object) Typ danych: wobjdata

Układ współrzędnych względem którego opisywane jest położenie robota. Ta opcja może być pominięta i wtedy przyjmowany jest globalny (world) układ współrzędnych.

Wykonanie programu

Środek narzędzia (TCP) jest przemieszczany w kierunku punktu docelowego metodą interpolacji kątów przegubowych. Oznacza to że każda oś porusza się ze stałą prędkością kątową oraz że wszystkie osie osiągają położenie końcowe w tej samej chwili. Uzyskana trajektoria jest nieliniowa.

Ogólnie, TCP porusza się z prędkością zbliżoną do zaprogramowanej (niezależnie czy zewnętrzne osie są skoordynowane). Reorientacja narzędzia następuje jednocześnie z ruchem postępowym TCP.

1.2.6 MoveL

`MoveL` Ruch liniowy robota

Instrukcja `MoveL` jest używana do przemieszczenia punktu środka narzędzia (TCP) liniowo do podanego celu. Można jej również użyć do przeorientowania narzędzia.

Przykłady

```
MoveL p1, v1000, z30, tool2;
```

Środek narzędzia `tool2` zostanie przesunięty liniowo do pozycji `p1` z prędkością `v1000` i strefą dokładności `z30`

```
MoveL *, v1000\T:=5, fine, grip3;
```

TCP narzędzia `grip3` zostanie przesunięty po liniowej ścieżce do punktu zapamiętanego w instrukcji (oznaczonego znakiem `*`). Ruch będzie trwał 5 sekund.

Argumenty

```
MoveL [\Conc] ToPoint Speed [\V] | [\T] Zone [\Z] Tool [\WObj] [\Corr]
```

`[\Conc]` (Concurrent) Typ danych: przełącznik

Następujące po sobie instrukcje zostaną wykonane jednocześnie. Dzięki temu można skrócić czas wykonywania instrukcji w przypadku, gdy synchronizacja z zewnętrznymi urządzeniami nie jest wymagana.

Maksymalna ilość po sobie występujących instrukcji przesunięć wykorzystujących `\Conc` jest ograniczona do 5. Jeśli owa sekcja zawiera `StorePath-RestoPath`, wówczas nie jest możliwe użycie `\Conc`.

`ToPoint` Typ danych: robtarget

Punkt końcowy ruchu. Podaje się jako nazwaną pozycję lub definiuje bezpośrednio w instrukcji (oznaczony jako `*`).

`Speed` Typ danych: speeddata

Ustawienia prędkości TCP, prędkości zmiany orientacji oraz zewnętrznych osi.

`[\V]` (Velocity) Typ data: liczba

Dzięki temu argumentowi można bezpośrednio w instrukcji podać prędkość TCP wrażoną w mm/s.

`[\T]` (Time) Typ danych: liczba

Określa czas wykonywania ruchu w sekundach.

`Zone` Typ danych: zonedata

Ustawienia dokładności pozycjonowania.

`[\Z]` (Zone) Typ danych: liczba

Argument podawany celem bezpośredniego określenia dokładności położenia TCP. Długość krawędzi narożnej podawana jest w milimetrach.

`Tool` Typ danych: tooldata

Używane narzędzie. Środek narzędzia jest punktem, który osiąga zadane położenie.

`[\WObj]` (Work Object) Typ danych: wobjdata

Układ współrzędnych względem którego opisywane jest położenie robota. Ta opcja może być pominięta i wtedy przyjmowany jest globalny (world) układ współrzędnych.

`[\Corr]` (Correction) Typ danych: przełącznik

Dane korekcyjne zapisane przez instrukcję `CorrWrite` zostaną dodane do ścieżki i punktu docelowego, jeśli argument jest obecny.

Wykonanie programu

Robot osiąga położenie docelowe w następujący sposób:

- TCP narzędzia jest przemieszczany liniowo ze stałą, zaprogramowaną prędkością,
- zmiana orientacji narzędzia następuje w równych interwałach wzdłuż ścieżki

1.2.7 Reset

`Reset` Ustawienia wyjścia cyfrowego w stan nieaktywny (na zero)

Przykład

`Reset do15;` Wartość sygnału `do15` zostanie ustawiona na 0.

Argumenty

`Reset sygnał`

`sygnał` Typ danych: signaldo nazwa sygnału, który ma być ustawiony na 0.

1.2.8 Set

`Set` Uaktywnienie wyjścia cyfrowego (ustawienie na jeden)

Przykład

`Set do15;` Wartość sygnału `do15` zostanie ustawiona na 1.

Argumenty

`Set sygnał`

`sygnał` Typ danych: signaldo nazwa sygnału, który ma być ustawiony na 1.

1.2.9 While

`WHILE` powoduje powtarzanie ciągu instrukcji tak długo jak warunek jest spełniony.

Jeśli liczba powtórzeń może być określona z góry, można również użyć do tego celu instrukcji `FOR`.

Przykład

```
WHILE reg1 < reg2 DO
  ...
  reg1 := reg1 + 1;
ENDWHILE
```

Powtarza instrukcje wewnątrz pętli `WHILE` tak długo jak `reg1` $<$ `reg2`.

Argumenty

`WHILE warunek DO ... ENDWHILE`

`warunek` Typ danych: bool

Warunek, który musi być spełniony aby wykonywały się instrukcje pomiędzy `WHILE` a `ENDWHILE`.

Wykonanie programu

1. wyliczenie wartości logicznej warunku. Jeśli warunek nie jest spełniony, wykonywanie pętli zostaje zakończone i program przechodzi do instrukcji następującej po `ENDWHILE`,
2. wykonanie instrukcji wewnątrz pętli,
3. powtórzenie całej pętli, rozpoczynając od punktu 1.

1.3 Opis funkcji

1.3.1 Offs

`Offs` Przesunięcie położenia

Funkcja `Offs` służy do przesunięcia współrzędnych punktu.

Przykłady

```
MoveL Offs(p2, 0, 0, 10), v1000, z50, tool1;
```

Narzędzie zostanie przemieszczone do położenia 10 mm w kierunku osi z od punktu `p2`

```
p1 := Offs (p1, 5, 10, 15);
```

Punkt `p1` zostanie przesunięty o 5 mm w kierunku osi x, 10 mm w kierunku y oraz 15 mm w kierunku z.

Argumenty

```
Offs(punkt, przesX, przesY, przesZ)
```

`punkt` Typ danych: robtarget

Punkt, względem którego będzie robione przesunięcie.

`przesX` Typ danych: liczba

Przesunięcie w kierunku osi X, podane w mm.

`przesY` Typ danych: liczba

Przesunięcie w kierunku osi Y, podane w mm.

`przesZ` Typ danych: liczba

Przesunięcie w kierunku osi Z, podane w mm.

2 Wprowadzanie zmian w programie

2.1 Wybór instrukcji lub argumentu

Przed dokonaniem zmiany w programie można wybrać całą instrukcję lub pojedynczy argument do edycji. Jeśli zachodzi potrzeba zmiany pojedynczego argumentu, często najłatwiej jest wpierw wybrać ów argument. Przy zmianie całej instrukcji należy zaznaczyć całą instrukcję. Czasami, np. przy dodawaniu nowej instrukcji, nie jest istotne czy wybrana jest instrukcja czy któryś z argumentów.

- Wybór całej instrukcji:

Aby przejść do	Naciśnij
poprzedniej instrukcji	KursorDoGóry
następnej instrukcji	KursorWDół
pierwszej instrukcji	Edit: Goto Top
ostatniej instrukcji	Edit: Goto Bottom
następnej strony	NastepnaStrona
poprzedniej strony	PoprzedniaStrona

  Jeśli kursor zostanie przesunięty do pierwszej linii w instrukcjach złożonych (IF, FOR, WHILE lub TEST), wszystkie instrukcje, włącznie z ostatnią linią (np. ENDIF) zostaną wybrane. Naciśnięcie KursorWDół spowoduje wybieranie kolejnych instrukcji wewnątrz instrukcji złożonych. Zakończenia bloku złożonego (np. ENDIF, ELSE) nie mogą być wybrane oddzielnie.

- Wybór kilku instrukcji

  – zaznacz pierwszą lub ostatnią instrukcję z grupy,

  – wybierz Edit: Mark,

  – zaznacz pozostałe instrukcje używając KursorDoGóry lub KursorWDół

  Wybór zostaje automatycznie dezaktywowany po użyciu Edit: Cut albo Edit: Copy. Można również anulować zaznaczenie wybierając Edit: Unmark.

- Wybór argumentu

  Celem wyboru argumentu:

  – naciśnij KursorWPrawo aby przesunąć kursor o jeden argument w prawo, lub KursorWLewo aby przesunąć kursor o jeden argument w lewo.

2.2 Modyfikacja położenia w instrukcji pozycjonowania

- przemieść robota do wymaganej pozycji,
- wybierz instrukcję, którą należy zmienić. Jeśli instrukcja zawiera więcej niż jedną pozycję, np. MoveC, wybierz argument do zmiany,
- wciśnij klawisz ModPos albo wybierz Edit: ModPos

Stare położenie zostanie zastąpione aktualnym położeniem robota. Uwaga: jeśli zmienione zostanie położenie nazwane (czyli takie bez „*”), wszystkie instrukcje odwołujące się do tej nazwy także ulegną zmianie.