Struktury danych języka RAPID, RMiM

Janusz Jakubiak

Opis zadania

Wymagania wstępne

- Zagadnienia z ćwiczenia „Robot przemysłowy ABB IRB1400 – Wprowadzenie" i wymagania przedstawione tamże.
- Przygotować algorytmy realizacji zadań podanych w dalszej części instrukcji.
- Zapoznać się z przedstawionymi w RAPID Reference Manual (`http://rab.ict.pwr.wroc.pl/irb1400/datasys_rev1.pdf`) opisami używanych typów danych oraz instrukcji i funkcji (patrz dalsza część instrukcji)

Pytania i zadania przed rozpoczęciem ćwiczenia

- Co reprezentują i z jakich elementów składają się struktury `pose` i `robtarget`?
- W jaki sposób, mając dane pozycje 2 wierzchołków kwadratu, wyznaczyć pozostałe 2 (wzory, algorytm)?
- W jaki sposób, mając dany wektor na płaszczyźnie (XY), wyznaczyć wektor obrócony względem niego o zadany kąt (wzory, algorytm, funkcje języka RAPID) ?

Cele ćwiczenia

Poznanie i wykorzystanie w praktyce złożonych struktur danych języka RAPID. Wykorzystanie w programach względnych przesunięć i obrotów.

Przebieg ćwiczenia:

1. Zaprogramować ruch do dowolnego nazwanego punktu, zinterpretować składowe rekordu (typ `robtarget`).
2. Napisać procedurę, która dla dwóch zdefiniowanych wierzchołków z jednego boku kwadratu obliczy położenie pozostałych wierzchołków i narysuje pełny kwadrat. Robot powinien poruszać się 10-15cm nad wzorcem.Należy się upewnić, że oś narzędzia jest prostopadła do płaszczyzny wzorca.
3. Korzystając z procedury z poprzedniego punktu zrealizować rysowanie kwadratów obróconych względem siebie (według wzorca **??**). Skorzystać z dostępnych funkcji języa RAPID.
4. Wykorzystując definicję przemieszczenia programu (`DispSet`) narysować serię kwadratów obróconych względem siebie o zadany kąt (Por. rys. **??**)
5. Narysować jedną z szachownic z rys. **??**, następnie wykorzystać funkcję (`DefDFrame`) do wyliczenia przesunięcia i narysowania drugiej z szachownic

Podgląd wartości zmiennych z panelu operatora

Wartości każdej zmiennej (a także wszystkich pól składowych rekordów) można przejrzeć z poziomu panelu operatora. W tym celu należy z menu wybrać typ danych `View→Data Types`, a następnie otworzyć okno zmiennych `View→Data`.

Definiowanie procedur i funkcji

Język RAPID umożliwia tworzenie podprogramów: procedur i funkcji[1].

Edycja nowego programu rozpoczyna się zawsze w procedurze `main`. Aby przejść do innego podprogramu należy z menu wybrać `View→Routines`, a następnie wybrać podprogram, który chcemy modyfikować, lub wybierając `New` utworzyć nowy podprogram.

Po utworzenia nowego podprogramu należy zdefiniować jego nazwę, a następnie wybrać `Decl`, aby zdefiniować typ podprogramu (procedura/funkcja) oraz parametry wywołania (nazwa oraz typ), a w przypadku funkcji – także typ zwracanej wartości.

Wyrównanie osi narzędzia

Oprogramowanie robota umożliwia wyrównanie osi Z narzędzia z osią jednago z układów współrzędnych (świata, podstawy, lub obiektu). W tym celu należy w trybie `jog` należy wybrać z menu `Special→Align` i wykonać ruch joystickiem. Wyrównanie następuje w kierunku najbliższej z osi wskazanego układu współrzędnych.

Dostęp do pól zmiennych

Zmienne typu złożonego interpretowane są jak rekordy[2], z poziomu programu możemy zatem korzystać także z indywidualnych pól rekordów, np.
`p10.trans.x` pobiera z konfiguracji `p10` współrzędną x wektora przesunięcia (translacji)
`p10.extax.eax_b` oznacza kąt obrotu drugiej osi zewnętrznej

Elementy języka RAPID przydatne do realizacji zadania

Do realizacji ćwiczenia niezbędna jest znajomość procedur i funkcji wykorzystywanych w ćwiczeniu „Robot przemysłowy ABB IRB1400 – Wprowadzenie" (Instrukcja „Programowanie robota IRB1400"), przydatne będą standardowe funkcje sterujące przebiegiem programu `For`, `If`, podstawowe funkcje matematyczne `Sin`, `Cos`, `Sqrt` itp., a także dodatkowo:

Typy danych

`confdata`	Ustawienie osi robota (do kinematyki odwrotnej)
`extjoint`	Położenie osi zewnętrznych
`num`	Liczba/rejestr
`orient`	Orientacja
`pos`	Pozycja (x, y, z)
`pose`	Transformacja współrzędnych (tzw. ramka przesunięcia)
`robtarget`	Konfiguracja robota (w przestrzeni)
`wobjdata`	Dane obiektu
`zonedata`	Strefa dokładności

Instrukcje

`PDispSet (DispFrame)`	Włączenie przesunięcia programu o ustaloną ramkę przesunięcia
`PDispOn[\Rot][\ExeP]PPoint Tool[\WObj]`	Włączenie ustawionego przesunięcia programu
`PDispOff`	Wyłączenie przesunięcia programu

[1] tak jak w języku Pascal procedura odpowiada instrukcji (komendzie), zaś funkcja – wyrażeniu (zwraca wartość)
[2] odpowiednik `record` w Pascalu, lub `struct` w C

Funkcje

`CRobT`	odczyt bieżącej konfiguracji robota (wynik typu `robtarget`)
`PoseMult(Pose1 Pose2)`	iloczyn ramek przesunięcia (typu `pose`)
`PoseVect (Pose Pos)`	iloczyn ramki przesunięcia i wektora (typu `pos`)
`PoseInv(Pose1)`	odwrotność ramki przesunięcia (typu `pose`)
`VectMagn (Pos)`	długość wektora (typu `num`)
`DotProd (Vector1 Vector2)`	iloczyn skalarny wektorów (typu `num`)
`DefFrame(nP1 nP2 nP3 [\Origin])`	ramka przesunięcia wyliczona z 3 punktów `robtarget`(typu `pose`)
`DefDFrame (oP1 oP2 oP3 nP1 nP2 nP3)`	ramka przesunięcia pomiędzy ramkami starą: o (3 punkty) i nową n (3 punkty) (typu `pose`)
`ORobT`	Usunięcie przesunięcia programu (typu `robtarget`)
`OrientZYX (numX numY numX)`	Wylicza rotację na podstawie kątów Eulera ZYX (typu `orient`)

Materiały dodatkowe:

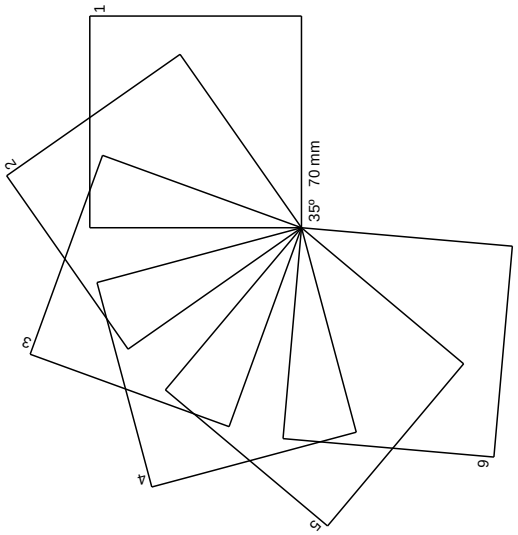
1. Instrukcja „Programowanie robota IRB1400" Paweł Ludwików, `http://rab.ict.pwr.wroc.pl/dydaktyka/instrukcje/irb1400-komendy.pdf`
2. Ćwiczenie „Robot przemysłowy ABB IRB1400 – Wprowadzenie" `http://rab.ict.pwr.wroc.pl/lab_010/Robotyka2/irb_I/`
3. Podstawy RAPID-a (fragmenty), Marek Duchinski

Przykład wyliczania przesunięcia obiektów: (źródło: forum ABB)

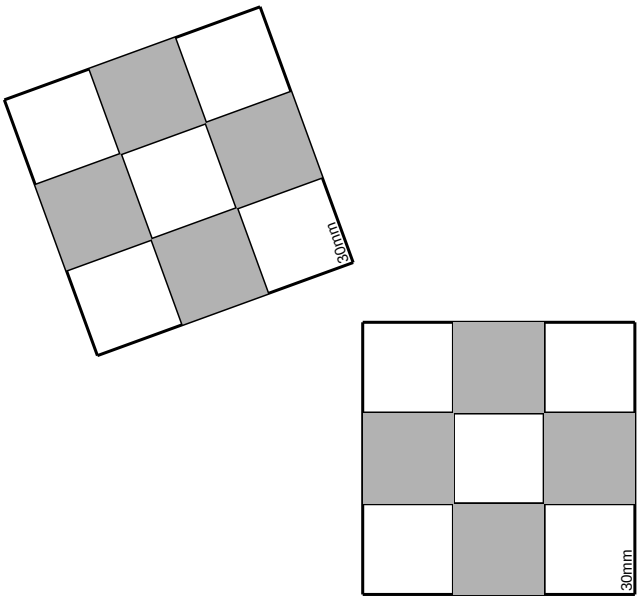
```
FUNC wobjdata createWobjCad(
    robtarget ptRob1,
    robtarget ptRob2,
    robtarget ptRob3,
    robtarget ptVehic1,
    robtarget ptVehic2,
    robtarget ptVehic3)

    VAR pose frameRob:=[[0,0,0],[1,0,0,0]];
    VAR pose frameVehic:=[[0,0,0],[1,0,0,0]];
    VAR wobjdata wobjTmp:= [FALSE, TRUE, "", [[0, 0, 0],[1, 0, 0, 0]],
                            [[0, 0, 0],[1, 0, 0, 0]]];

   !calculation vehicle frame position in world frame of the robot
    frameRob:=DefFrame(ptRob1,ptRob2,ptRob3);
    frameVehic:=DefFrame(ptVehic1,ptVehic2,ptVehic3);
    wobjTmp.uframe:=PoseMult(frameRob,PoseInv(frameVehic));
    RETURN wobjTmp;
  ENDFUNC
```



(a) Rotacja 35°



(b) Przemieszczenie

Rysunek 1: Wzorce