DYDAKTYKA ABB

CONST – dana stała; jej wartość może by zmieniana ręcznie, ale **nie można jej zmienić** podczas wykonywania programu

VAR – dana zmienna, której wartość może być zmieniana podczas wykonywania programu; po restarcie systemu zmienne typu **VAR wracają** do wartości początkowych, ustalonych w programie

PERS – zmienna trwała, której wartość może być zmieniana podczas wykonywania programu; po restarcie systemu zmienne typu **PERS zachowują** wartości zapamiętane przed restartem

wartość zdefiniowana

```
CONST num Volta:=1;
VAR num Integra:=2;
PERS num Ursula:=3;
```

zmiana zadana programowo

```
Integra:=200;
Ursula:=300;
```

wartość po zmianie

```
CONST num Volta:=1;
VAR num Integra:=200;
PERS num Ursula:=300;
```

wartość po resecie/restarcie programu

```
CONST num Volta:=1;
VAR num Integra:=2;
PERS num Ursula:=300;
```

DANE

VAR, PERS, CONST

Technical reference manuals Rapid instructions

ABB

Dane przechowywane przez system mogą być typu Stała, Const, Zmienna Var i „zmienna trwała” Pers.
W przykładzie pokazano jak zdefiniowane w systemie dane CONST, VAR i PERS zachowują się w przypadku zmiany ich definicji podczas uruchomienia programu.
Dane można dodawać przez Program data, (Dane programu) w ABB menu.
Dane typu num mogą być zdefiniowane np. jako stałe lub zmienne.

DYDAKTYKA ABB

```
12    VAR num Dix2;
13    PROC Path_10()
14        Dix2:=SignalDixi;
15
16        IF Dix2=1 THEN
17            GOTO wykonaj_procedure_Path_20;
18        ENDIF
19    MoveJ Target_20,v1000,fine,SchunkTool\WObj:=Workobject_1;
20    MoveJ Target_30,v1000,fine,SchunkTool\WObj:=Workobject_1;
21    MoveJ Target_40,v1000,fine,SchunkTool\WObj:=Workobject_1;
22    MoveJ Target_50,v1000,fine,SchunkTool\WObj:=Workobject_1;
23        wykonaj_procedure_Path_20:
24        Path_20;
25        MoveJ Target_20,v1000,fine,SchunkTool\WObj:=Workobject_1;
26    ENDPROC
27    PROC Path_20()
28    MoveJ Target_60,v1000,fine,SchunkTool\WObj:=Workobject_1;
29    MoveJ Target_70,v1000,fine,SchunkTool\WObj:=Workobject_1;
30    MoveJ Target_80,v1000,fine,SchunkTool\WObj:=Workobject_1;
31    MoveJ Target_90,v1000,fine,SchunkTool\WObj:=Workobject_1;
32    MoveJ Target_100,v1000,fine,SchunkTool\WObj:=Workobject_1;
33        MoveJ Target_60,v1000,fine,SchunkTool\WObj:=Workobject_1;
34    ENDPROC
```

„wykonaj_procedurę_Path_20:” etykieta / znacznik w programie

„GOTO” instrukcja przenosząca PP (Program Pointer – wskaźnik programu) w inne miejsce tej samej procedury.

INSTRUKCJA GOTO

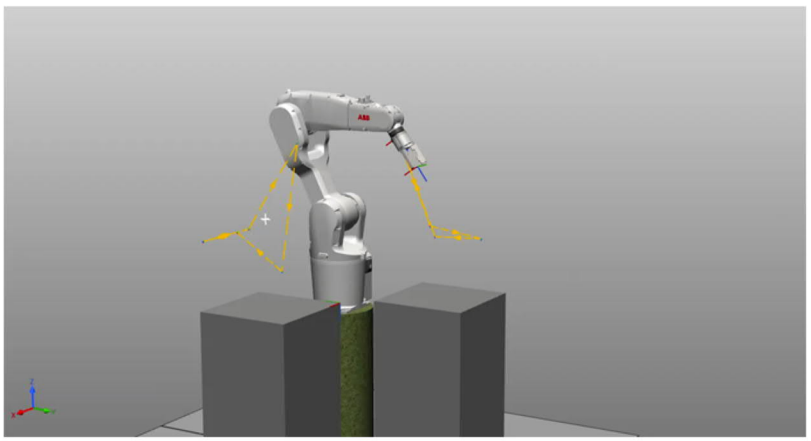
GOTO – przeniesienie PP w obrębie procedury

Technical reference manuals Rapid instructions

ABB

Przykład opisuje sytuację, w której w module utworzono zmienną numeryczną VAR num Dix2.
W procedurze Path_10 przypisano Dix2 wartość cyfrowego sygnału wejściowego o nazwie SignalDixi.
Następnie wstawiono warunek IF THEN (Jeśli zmienna Dix2 = 1 (tak naprawdę stanie się to jeśli sygnał wejściowy będzie miał wartość 1) to wykonaj skok to znacznika o nazwie „wykonaj_procedurę_Path_20”
Oznaczać to będzie, że Program Poitner ominie część kodu Rapid i przeskoczy do znacznika. W przypadku gdy wartość Dixi2 będzie inna niż 1 program będzie wykonany w całości.
Zastosowano tutaj również instrukcję Proccall, która umożliwia przechodzenie PP z jednej procedury do innej (w tym przypadku do Path_20).
Kod przedstawiony na rysunku jest kodem z programu RobotStudio.
Aby zdefiniować Dix2 :=SignalDixi należy najpierw taki sygnał np. wirtualny utworzyć w systemie robota.

DYDAKTYKA ABB



Inputs

SignalDixi

SignalDixi

```
12  ····VAR·num·Dix2;
13  ····PROC·Path_10()
14  ····Dix2:=SignalDixi;
15  ····
16  ····IF·Dix2=1·THEN
17  ····GOTO·wykonaj_procedure_Path_20;
18  ····ENDIF
19  ····MoveJ·Target_20,v1000,fine,SchunkTool\WObj:=Workobject_1;
20  ····MoveJ·Target_30,v1000,fine,SchunkTool\WObj:=Workobject_1;
21  ····MoveJ·Target_40,v1000,fine,SchunkTool\WObj:=Workobject_1;
22  ····MoveJ·Target_50,v1000,fine,SchunkTool\WObj:=Workobject_1;
23  ····wykonaj_procedure_Path_20:
24  ····Path_20;
25  ····MoveJ·Target_20,v1000,fine,SchunkTool\WObj:=Workobject_1;
26  ····ENDPROC
27  ····PROC·Path_20()
28  ····MoveJ·Target_60,v1000,fine,SchunkTool\WObj:=Workobject_1;
29  ····MoveJ·Target_70,v1000,fine,SchunkTool\WObj:=Workobject_1;
30  ····MoveJ·Target_80,v1000,fine,SchunkTool\WObj:=Workobject_1;
31  ····MoveJ·Target_90,v1000,fine,SchunkTool\WObj:=Workobject_1;
32  ····MoveJ·Target_100,v1000,fine,SchunkTool\WObj:=Workobject_1;
33  ····MoveJ·Target_60,v1000,fine,SchunkTool\WObj:=Workobject_1;
34  ····ENDPROC
```

SIGNALDIXI:=0

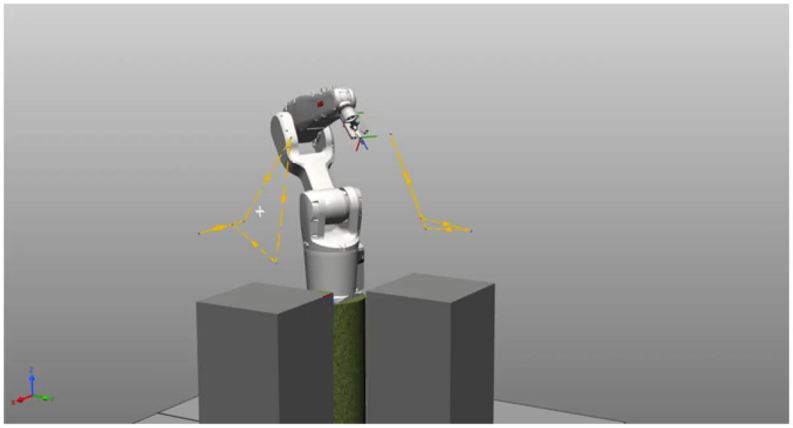
GOTO – przeniesienie PP w obrębie procedury

Technical reference manuals Rapid instructions

ABB

Przykład działania instrukcji

DYDAKTYKA ABB



```
12  ····VAR·num·Dix2;
13  ····PROC·Path_10()
14  ········Dix2:=SignalDixi;
15  ········
16  ········IF·Dix2=1·THEN
17  ············GOTO·wykonaj_procedure_Path_20;
18  ········ENDIF
19  ····MoveJ·Target_20,v1000,fine,SchunkTool\WObj:=Workobject_1;
20  ····MoveJ·Target_30,v1000,fine,SchunkTool\WObj:=Workobject_1;
21  ····MoveJ·Target_40,v1000,fine,SchunkTool\WObj:=Workobject_1;
22  ····MoveJ·Target_50,v1000,fine,SchunkTool\WObj:=Workobject_1;
23  ········wykonaj_procedure_Path_20:
24  ········Path_20;
25  ····MoveJ·Target_20,v1000,fine,SchunkTool\WObj:=Workobject_1;
26  ····ENDPROC
27  ····PROC·Path_20()
28  ····MoveJ·Target_60,v1000,fine,SchunkTool\WObj:=Workobject_1;
29  ····MoveJ·Target_70,v1000,fine,SchunkTool\WObj:=Workobject_1;
30  ····MoveJ·Target_80,v1000,fine,SchunkTool\WObj:=Workobject_1;
31  ····MoveJ·Target_90,v1000,fine,SchunkTool\WObj:=Workobject_1;
32  ····MoveJ·Target_100,v1000,fine,SchunkTool\WObj:=Workobject_1;
33  ····MoveJ·Target_60,v1000,fine,SchunkTool\WObj:=Workobject_1;
34  ····ENDPROC
```

Inputs

SignalDixi

SIGNALDIXI:=1

GOTO – przeniesienie PP w obrębie procedury

Technical reference manuals Rapid instructions

DYDAKTYKA ABB

```
MODULE testgoto
    VAR num rel1:=2;
    PROC gotowe()
        gornypoziom:
        WaitTime \Inpos,1;
        rel1:=1;
        IF rel1<2 THEN
            GOTO gornypoziom;
        ELSE WaitTime \Inpos, 2;
        ENDIF
    ENDPROC
ENDMODULE
```

„WaitTime" instrukcja zatrzymania programu i odliczania czasu, argument opcjonalny \Inpos określa, że robot odliczy wyznaczony czas bez zmiany pozycji

INSTRUKCJA WAIT

Wait – instrukcja zatrzymania programu

Technical reference manuals Rapid instructions

ABB

Instrukcja WaitTime służy do odliczania czasu, jeśli podany jest argument opcjonalny \Inpos, to robot zatrzymuje się w instrukcji ruchowej poprzedzającej instrukcję WaitTime i „odlicza" zadany czas (tutaj to 1 [s]).

DYDAKTYKA ABB

„WaitDO" instrukcja Wait Digital Output – kursor programu pozostanie w linii instrukcji do momentu ustawienia określonego sygnału wyjścia
„WaitSyncTask" instrukcja do synchronizacji tasków robota w określonych punktach. Wymaga „tasklist" i „syncident" – unikalnej nazwy punktu synchronizacji

„WaitDI" instrukcja Wait Digital Input – kursor programu pozostanie w linii instrukcji do momentu ustawienia określonego sygnału wejścia
Przykład:
WaitDI di4, 1;
Wykonywanie programu będzie kontynuowane gdy sygnał wejścia „di4" zostanie ustawiony
Przykład:
WaitDI grip_status, 0;
Wykonywanie programu będzie kontynuowane gdy sygnał wejścia grip_status zostanie zresetowany

INSTRUKCJA WAIT

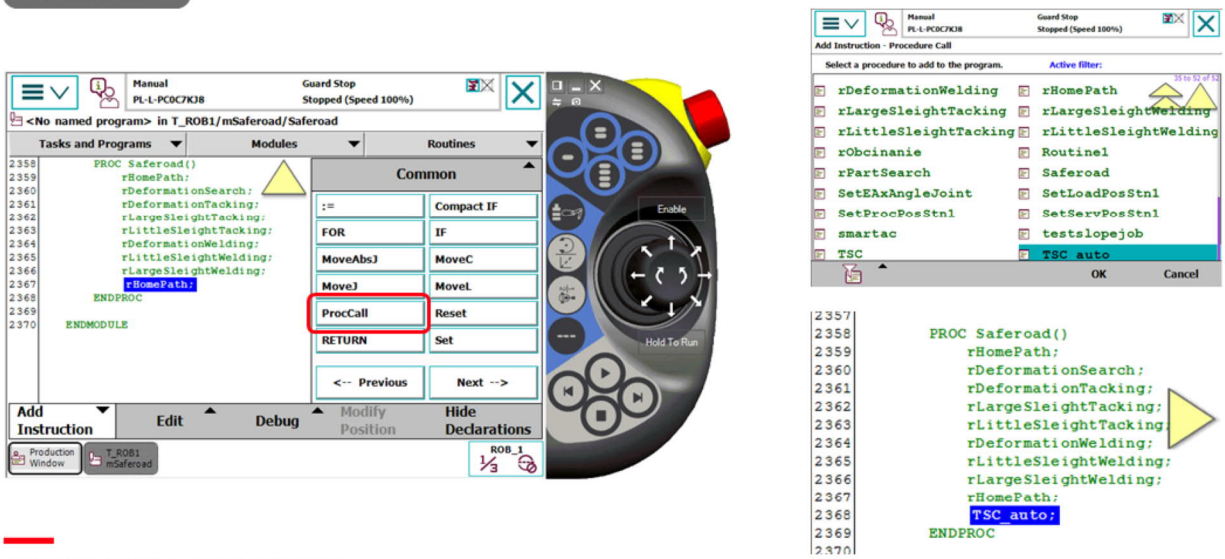
Wait – instrukcje zatrzymujące program

Technical reference manuals Rapid instructions

ABB

Pozostałe instrukcje Wait

DYDAKTYKA ABB



INSTRUKCJA PROCCALL – EDYTOR PROGRAMU

ProcCall – wywołanie kolejnej procedury

Technical reference manuals Rapid instructions

Instrukcja wywołująca inną procedurę. Jeśli w linii procedury Path_10 znajdzie się instrukcja Proccall procedury Path_20 to Program Pointer po dojściu do tej instrukcji przejdzie do procedury Path_20 i a po jej przejściu wróci do Path_10

DYDAKTYKA ABB

```
Search_Part Test_rygiel, pRygiel_7, pRygiel_8, v300, tWeldGun\WObj:=WobjLittleSleight;

VAR bool Test_rygiel:=FALSE;


IF Test_rygiel=TRUE THEN
        TPWrite "Rygiel bazy Litle Sleigh nie zostal zwolniony";
        Stop;

    ENDIF
```

DANE PROGRAMU

BOOL – dane programu FALSE / TRUE

Technical reference manuals Rapid instructions

ABB

Przykład danej typu BOOL która może mieć wartość typu fałsz / prawda (False/True)

DYDAKTYKA ABB

```
PROC TIMEMEASURE()
 VAR clock tmeasure;
      VAR num time;
      ClkReset tmeasure;
      ClkStart tmeasure;
MoveL RelTool(p10start,0,0,30), v1000, z50, pisak;
MoveL p10, v1000, z50, pisak;
AccSet 40, 40\FinePointRamp:=30;
MoveJ p10start, v1000, fine, pisak\WObj:=wobj0;
      clkstop tmeasure;
      time:=clkread(tmeasure);
      tpwrite ""\num:=time;
ENDPROC
```

DANE PROGRAMU

CLOCK – pomiar czasu

Technical reference manuals Rapid instructions

ABB

Przykład działania pomiaru czasu. Należy utworzyć zmienną numeryczną np. „time”, zmienną typu clock tutaj np. „tmeasure” Następnie podaje się instrukcje ClkReset, ClkStart a w momencie gdy chcemy zakończyć pomiar ClkStop.
Uwaga, aby szybciej znaleźć potrzebne instrukcje na panelu FlexPendant stosuj filtrowanie wg nazwy.
Aby na FlexPendant wprowadzić zapis time := clkread (tmeasure) należy najpierw dodać instrukcję „ := ” w menu Common a potem dodać jej lewą i prawą stronę! ”

DYDAKTYKA ABB

```
PROC square()
MoveJ [[469.64,-49.65,596.78],[0.0108005,-0.997788,-0.0645406,-0.0116651],[-1,0,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]], v1000, fine, flamaster;
MoveL [[378.98,83.85,250.00],[0.010766,-0.997792,-0.0644979,-0.0116168],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]], v1000, fine, flamaster;
hammer;
MoveL [[678.98,83.85,250.00],[0.010766,-0.997792,-0.0644979,-0.0116168],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],, v1000, fine, flamaster;
hammer;
MoveL [[678.98,183.85,250.00],[0.010766,-0.997792,-0.0644979,-0.0116168],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],, v1000, fine, flamaster;
hammer;
MoveL [[378.98,183.85,250.00],[0.010766,-0.997792,-0.0644979,-0.0116168],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]],, v1000, fine, flamaster;
hammer;
MoveL [[378.98,83.85,250.00],[0.010766,-0.997792,-0.0644979,-0.0116168],[0,0,-2,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]], v1000, fine, flamaster;
ENDPROC


PROC hammer()
VAR robtarget pdob:=[[469.64,-49.63,604.99],[0.010804,-0.997785,-0.0645981,-0.0116556],[-1,0,-3,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];

pdob := CRobT(\Tool:=flamaster\WObj:=kartka);
MoveL Offs(pdob,0,0,100), v1000, fine, flamaster\WObj:=kartka;
MoveL pdob, v1000, fine, flamaster\WObj:=kartka;
MoveL Offs(pdob,0,0,100), v1000, fine, flamaster\WObj:=kartka;
MoveL pdob, v1000, fine, flamaster\WObj:=kartka;

ENDPROC
```

DATA

CRobT – odczytanie pozycji

Technical reference manuals Rapid instructions

Przykład działania funkcji CRobT

DYDAKTYKA ABB

```
2   MODULE·Misha
3   ····PROC·mmm()
4   ····    MoveL·[[94.95,77.56,0.28],[0.0708146,0.0611092,-0.0644
5   ····    MoveL·[[310.83,104.80,-5.71],[0.0708152,0.0611095,-0.0
6   ····    MoveC·[[195.26,141.54,1.41],[0.0708127,0.0611101,-0.06
7   ····    MoveJ·[[210,116.61,-93.55],[0.0708805,0.061114,-0.0644
8   ····    MoveL·[[383.52,218.27,-234.62],[0.502425,-0.542301,-0.
9
10  ····ENDPROC
11  PROC·Forloopa()
12  ····FOR·i·FROM·1·TO·3·DO
13  ····    mmm;
14  ····ENDFOR
15  ENDPROC
16  ENDMODULE
```

DATA

FOR – powtórzenie

Technical reference manuals Rapid instructions

Powtórzenie wykonywania procedury „mmm” trzykrotnie.

Przykład działania instrukcji FOR

DYDAKTYKA ABB

```
11  PROC Forloopa()
12      FOR i FROM 1 TO 3 DO
13          mmm;
14          IF i=3 THEN
15              Routine1;
16          ENDIF
17      ENDFOR
18  ENDPROC
19  ENDMODULE
```

DATA

IF – warunek

Technical reference manuals Rapid instructions

Gdy i = 3 wykonana zostanie procedura o nazwie „Routine1”

IF - warunek

DYDAKTYKA ABB

```
MODULE Misha
    LOCAL VAR pose pose1:=[[0,0,50],[1,0,0,0]];
    PROC mmm()
        MoveJ [[94.95,77.56,0.28],[0.0708146,0.0611092,-0.0644544,-0.993527],[-1,-1,-
        MoveL [[310.83,104.80,-5.71],[0.0708152,0.0611095,-0.0644561,-0.993527],[0,-1
        MoveC [[195.26,141.54,1.41],[0.0708127,0.0611101,-0.0644567,-0.993527],[-1,-1
        MoveJ [[210,116.61,-93.55],[0.0708805,0.061114,-0.0644475,-0.993523],[-1,-1,-
        MoveL [[383.52,218.27,-234.62],[0.502425,-0.542301,-0.00760624,-0.673366],[-1
        PDispOff;

    ENDPROC
PROC Forloopa()
    FOR i FROM 1 TO 3 DO
        mmm;
        IF i=2 THEN
            PDispSet pose1;
        ENDIF
    ENDFOR
ENDPROC
ENDMODULE
```

PDispOff – anulowanie przesunięcia

DATA

PDispSet – przesunięcie o stałą wartość

Technical reference manuals Rapid instructions

Gdy i = 2 wykonana zostanie instrukcja przesunięcia o 50 mm współrzędnych osi z w procedurze mmm

PDispSet – przykład kodu.