Kuka Robot Language

Zapisywanie programu w plikach z rozszerzeniami:
SCR – kod programu
DAT – stałe dane programu

File	Directory	Meaning
$CONFIG.DAT	KRC:\R1\System	System data list with general configuration data
SPS.SUB		Submit file
BAS.SRC		Basic package for initialization etc.
SELECT.DAT SELECT.SRC		Distribution programs for the execution of macros, subprograms, etc.
VW.SRC		Standard VW routines
WEAV_DEF.SRC	KRC:\R1\System	Program for weave motions
$MACHINE.DAT	KRC:\R1\mada	System data list with system variables for adapting the controller and the robot
$OPERATE.SRC*1)		System file with program and robot status data
$ROBCOR.DAT		System data list with data for the dynamic model of the robot
MACHINE.UPG ROBCOR.UPG		Upgrade files
CELL.SRC	KRC:\R1\Folgen	Program for controlling robots via a central PLC
MAKROSAW.SRC	KRC:\R1\Makros	Block selection program
MAKROSPS.SRC		Program for the free-running PLC
VW_USER.DAT VW_USER.SRC VW_USER_R.DAT VW_USER_R.SRC VW_USER_S.DAT VW_USER_S.SRC	KRC:\R1\VW_USER	Present for reasons of compatibility with previous versions.*2) Expert programming modules
*1)File is present, but not visible or editable *2)Avoid using where possible!		

Pliki konfiguracyjne
$MACHINE.DAT
$CUSTOM.DAT
$ROBOTER.DAT
$CONFIG.DAT zmienne zadeklarowane w tym pliku widoczne są w każdym programie

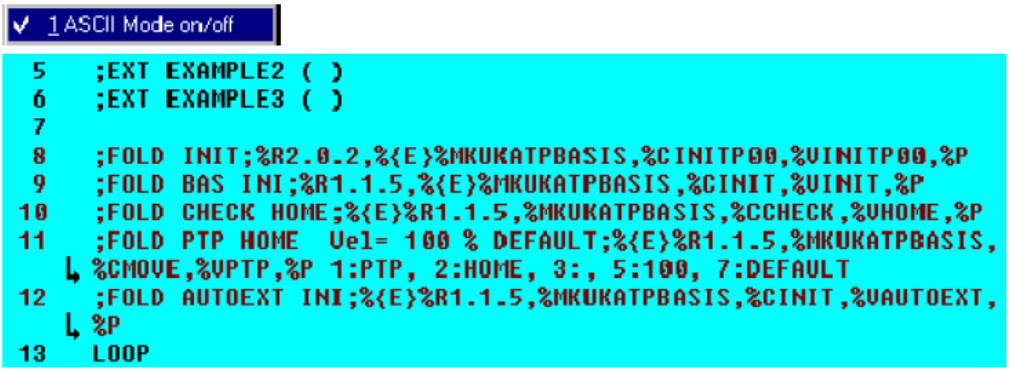
Progress.ini
GLOBAL_KEY=TRUE (włączenie możliwości stosowania słowa kluczowego GLOBAL)

$CUSTOM.DAT (\PROGRAM FILES\KRC\MADA\STEU) (predefiniowanie kanałów szeregowych – **SER_1** i **SER_2**)

Komentarz „ ; % ”

PTP P1 **;Move to starting point**
...
;--- Reset outputs ---
FOR I = 1 TO 16
$OUT[I] = FALSE
ENDFOR

Ukrywanie fragmentów programu

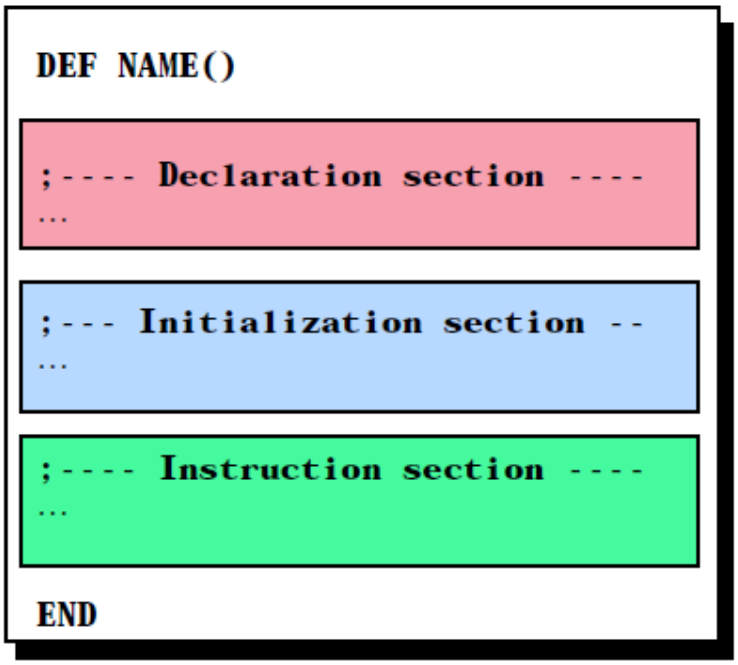


;FOLD RESET OUT
FOR I=1 TO 16
$OUT[I]=FALSE
ENDFOR
;ENDFOLD
RESET OUT

Tryby wykonywania programu

Run mode	Description
ISTEP	Incremental Step (single step) The program is executed step by step, i.e. with a STOP after each instruction (including blank lines). The program is executed without advance processing.
MSTEP	Motion Step (program step) The program is executed one motion instruction at a time, i.e. with a STOP before each motion instruction. The program is executed without advance processing.
GO	All instructions in the progam are executed up to the end of the program without a STOP.

Struktura programu



Zmienne:

- maks. 12(24) znaki
- znaki A-Z, 0-9, ”_”, ”$”
- nie mogą rozpoczynać się od liczby
- nie mogą być słowem kluczowym

Zasięg zmiennych

- jeżeli zadeklarowane w pliku SRC -> pomiędzy DEF i END
- jeżeli zadeklarowane w pliku DAT -> cały czas

Typy zmiennych

Data type	Integer	Real	Boolean	Character
Keyword	**INT**	**REAL**	**BOOL**	**CHAR**
Meaning	Integer	Floating-point number	Logic state	1 character
Range of values	$-2^{31}...2^{31}-1$	±1.1E-38... ±3.4E+38	TRUE, FALSE	ASCII character

Deklaracje tablic i macierzy

DECL INT OTTO[7]
DECL REAL MATRIX[7,3]
DECL BOOL ARRAY_3D[5,3,4]

Predefiniowane struktury

The following structures are predefined in the file $OPERATE.SRC:

```
STRUC AXIS    REAL  A1,A2,A3,A4,A5,A6
STRUC E6AXIS  REAL  A1,A2,A3,A4,A5,A6,E1,E2,E3,E4,E5,E6
STRUC FRAME   REAL  X,Y,Z,A,B,C
STRUC POS     REAL  X,Y,Z,A,B,C, INT S,T
STRUC E6POS   REAL  X,Y,Z,A,B,C,E1,E2,E3,E4,E5,E6, INT S,T
```

T

We	Bit 5	Bit 4	Bit3	Bit 2	Bit 1	Bit 0
0	A6 >= 0°	A5 >= 0°	A4 >= 0°	A3 >= 0°	A2 >= 0°	A1 >= 0°
1	A6 < 0°	A5 < 0 °	A4 < 0°	A3 < 0°	A2 < 0°	A1 < 0°

S

Wert	Bit 2	Bit 1	Bit 0
0	0° <= A5 < 180° A5 < -180°	A3 < φ (φ zależy od typu robota)	Obszar przy podłodze
1	-180° <= A5 < 0° A5 >= 180°	A3 >= φ (φ zależy od typu robota)	Obszar nad głową

DECL POS POSITION

POSITION.X = 34.4
POSITION.Y = -23.2
POSITION.Z = 100.0
POSITION.A = 90
POSITION.B = 29.5
POSITION.C = 3.5
POSITION.S = 2
POSITION.T = 6

POSITION={X 34.4,Y -23.2,Z 100.0,A 90,B 29.5,C 3.5,S 2,T 6}

POSITION={B 100.0,X 29.5,T 6}
POSITION={A 54.6,B -125.64,C 245.6}
POSITION={POS: X 230,Y 0.0,Z 342.5}

In the case of POS, E6POS, AXIS, E6AXIS and FRAME structures missing components are not altered. In all other aggregates, non-existing components are set to invalid.

Typ ENUM

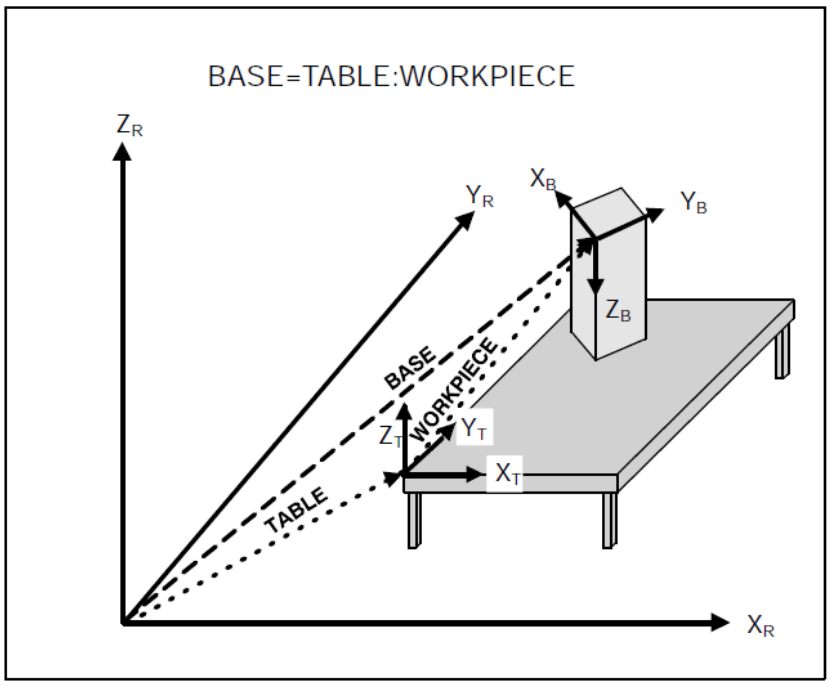
ENUM MODE_OP T1, T2, AUT, EX, INVALID

Zmienne systemowe rozpoczynają się od ”$”

Operacje

Arytmetyczne	+	-	/	*		
Logiczne	NOT	AND	OR	EXOR		
Bitowe	B_NOT	B_AND	B_OR	B_EXOR		
Relacyjne	==	<	>	<=	>=	<>
Geometryczne	:	(sumowanie wektorów FRAME i POS)				

Left operand (reference CS)	Operator	Right operand (target CS)	Result
POS	:	POS	POS
POS	:	FRAME	FRAME
FRAME	:	POS	POS
FRAME	:	FRAME	FRAME



Funkcje matematyczne

Descrip-tion	Function	Data type of argument	Range of values of argument	Data type of function	Range of values of result
Absolute value	**ABS(X)**	REAL	-≧...+≧	REAL	0...+≧
Square root	**SQRT(X)**	REAL	0...+≧	REAL	0...+≧
Sine	**SIN(X)**	REAL	-≧...+≧	REAL	-1...+1
Cosine	**COS(X)**	REAL	-≧...+≧	REAL	-1...+1
Tangent	**TAN(X)**	REAL	-≧...+≧*	REAL	-≧...+≧
Arc cosine	**ACOS(x)**	REAL	-1...+1	REAL	0_..180_
Arc tangent	**ATAN2(Y,X)**	REAL	-≧...+≧	REAL	-90_..+90_
* no odd multiple of 90_, i.e. X ¸ (2k-1)*90_, k∈8					

Zmienne systemowe:

TIMER

$TIMER[1]…$TIMER[10]

$TIMER_STOP[1]…$TIMER_STOP[10]

$TIMER_STOP[4] = FALSE ; uruchomienie timera

$TIMER_STOP[4] = TRUE ; zatrzymanie timera

For VW, the following $TIMER[1]...$TIMER[10] and $TIMER_STOP[1]...$TIMER_STOP[10] are available to the programmer. $TIMER[11]...$TIMER[16] and $TIMER_STOP[11]...$TIMER_STOP[16] are frequently assigned for KUKA technology packages.

Pozycja robota
$POS_ACT (current robot position)

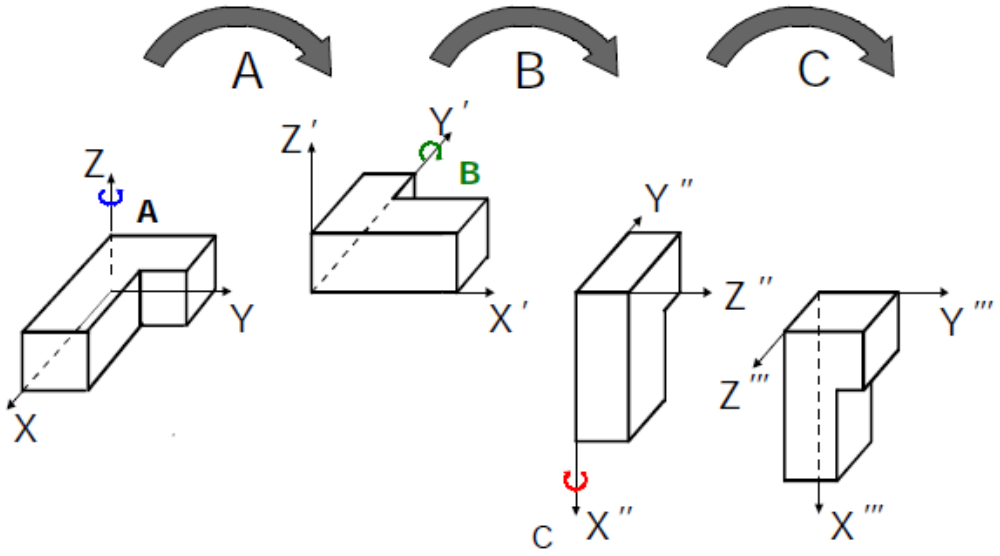
Flagi
$FLAG[1]...$FLAG[1024]

Flagi cykliczne
$CYCFLAG[1]...$CYCFLAG[32]
$CYCFLAG[10] = $IN[2] AND $IN[13]
Wyznaczane są, gdy któraś z wartości po prawej stronie ulegnie zmianie.

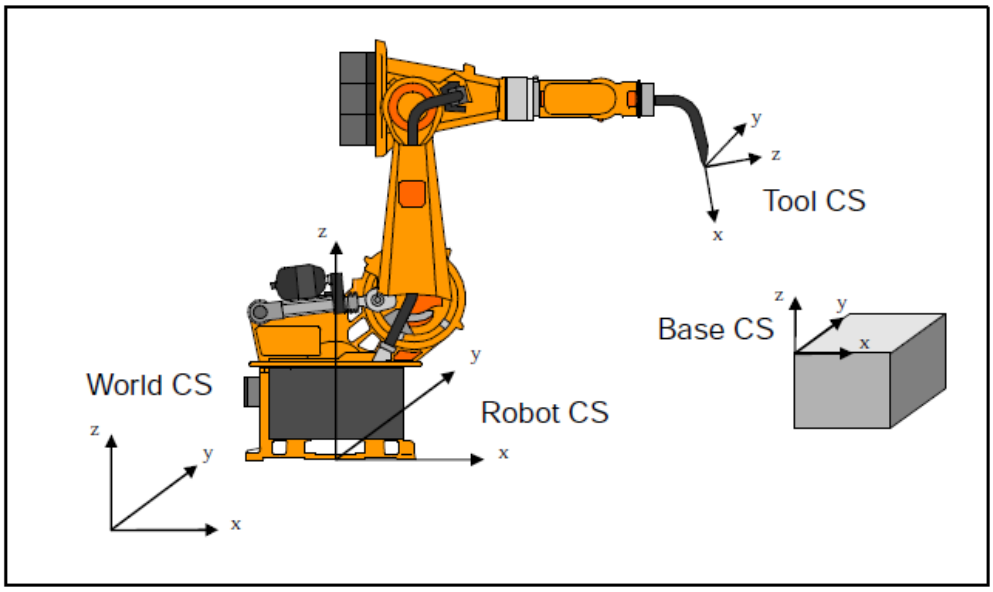
Układy współrzędnych

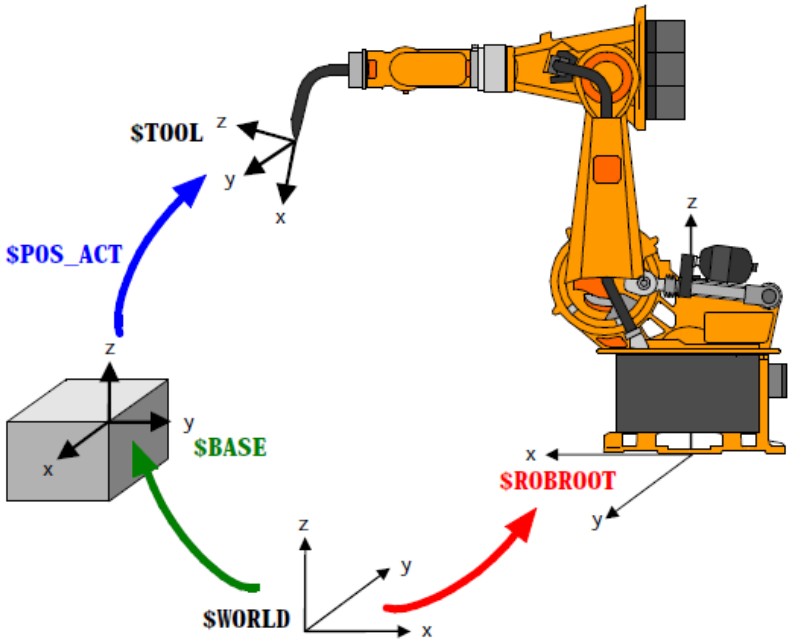
1. Rotation through angle A about the Z axis
2. Rotation through angle B about the new Y axis
3. Rotation through angle C about the new X axis

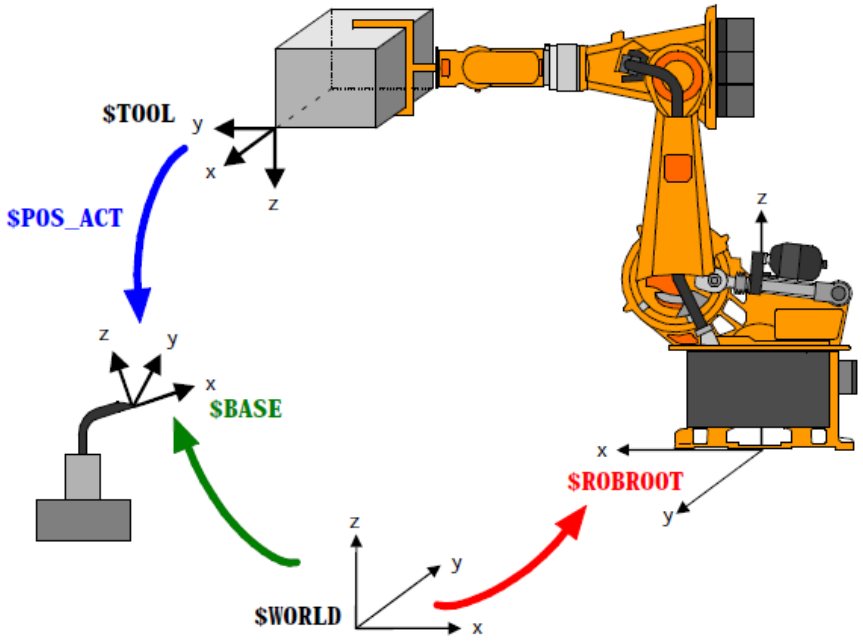
This rotation sequence corresponds to the well-known roll-pitch-yaw angles in the field of aviation. Angle C corresponds in this case to the roll, angle B to the pitch and angle A to the yaw.



Coordinate system	System variable	Status
World coordinate system	$WORLD	write-protected
Robot coordinate system	$ROBROOT	write-protected (can be changed in R1/$MASCHINE.DAT)
Tool coordinate system	$TOOL*	writable
Base (workpiece) coordinate system	$BASE*	writable
* In the case of gripper-related interpolation, $TOOL and $BASE are switched over (see below)		







PTP prędkości i przyspieszenia

$VEL_AXIS[*axis number***]**
$ACC_AXIS[*axis number***]**

Sterowanie z uwzględnieniem zjawisk dynamicznych

Dynamic robot model

The dynamic behavior of the robot mechanical system is shown as a system of equations making the following calculations possible:

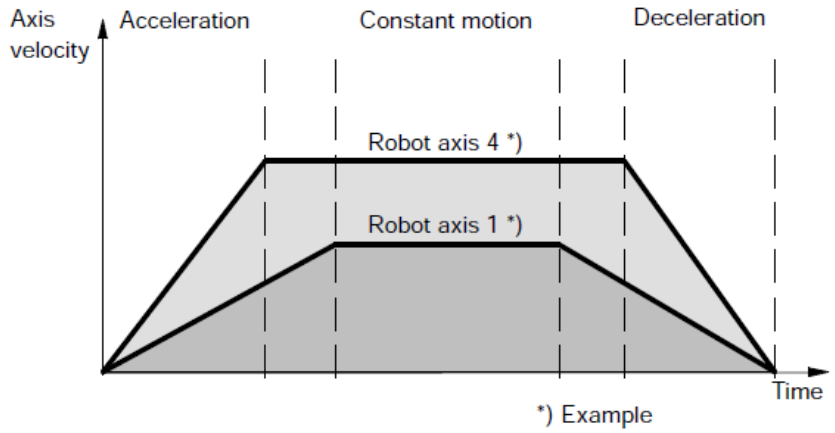
- **Torque on each individual drive** with all influences (friction, gravity, centrifugal forces, supporting forces, etc.), calculated from the torque values at the drive motor;
- **Motor-drive torque** (torque precontrol) for the current motion situation, calculated from the position, velocity and acceleration of the axes;
- **Max. permissible acceleration of all axes** with higher motion profile, determined from the positions of the axes and the maximum permissible motor and gear torque values.

The following parameters, among others, are taken into consideration:

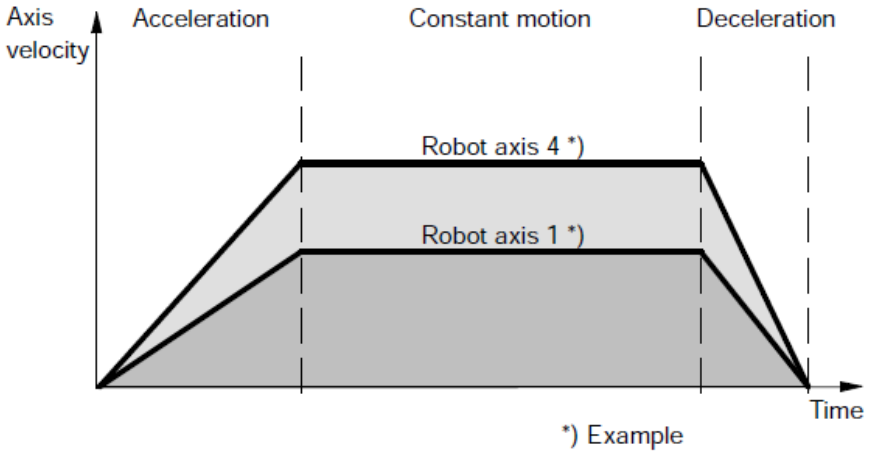
- Mass, center of gravity and moments of inertia of the mechanical components of the robot;
- Mass, center of gravity and moments of inertia of the load mounted (tool);
- Motor torque, gear torque, friction torque;
- Gravity, Coriolis force, product of inertia and moment at support;
- Position, velocity and acceleration of the axes.

Tryb synchroniczny PTP – ruch wszystkich osi rozpoczyna i kończy się w tym samym czasie.

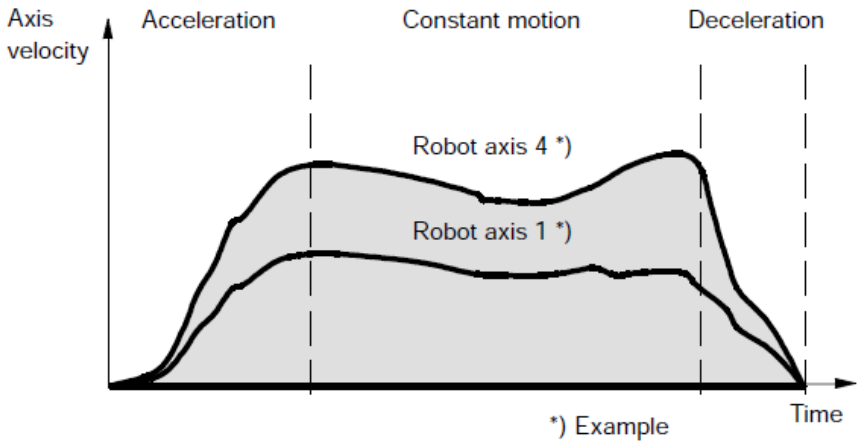
Programowany profil prędkości



Profil prędkości ze zmiennym przyspieszeniem (model-assisted)



Profil prędkości wyższego rzędu (model-assisted)



Uaktywnienie

Activation

is carried out in the data list $ROBCOR.DAT using the variables $ADAP_ACC and $OPT_MOVE.

1. Programmable velocity profile
   $ADAP_ACC=#NONE
   $OPT_MOVE=#NONE
2. Model-assisted velocity profile (acceleration adaptation)
   $ADAP_ACC=#STEP1
   $OPT_MOVE=#NONE
3. Model-assisted velocity profile (higher motion profile)
   $ADAP_ACC=#STEP1
   $OPT_MOVE=#STEP1

Ruch continuous path

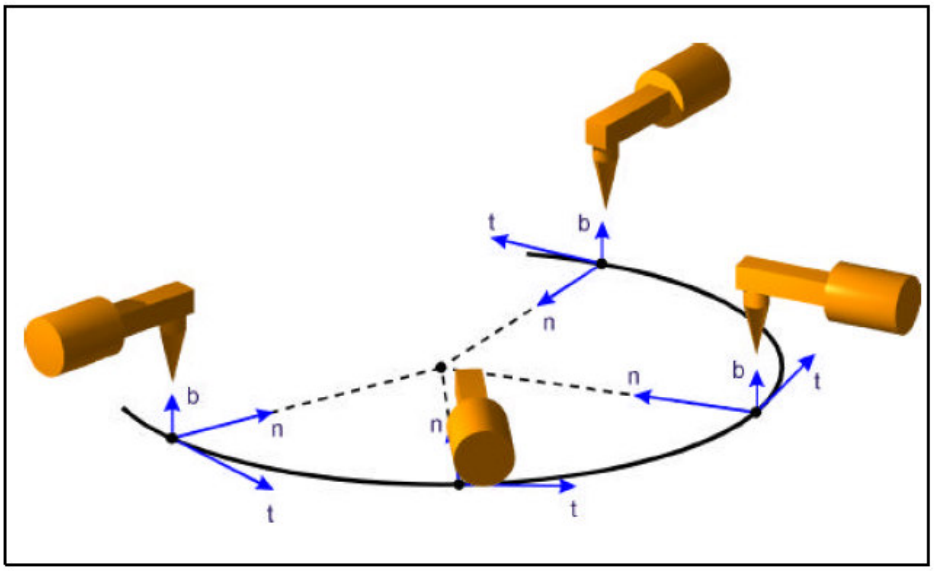
	Variable name	Data type	Unit	Function
Velocities	$VEL.CP	REAL	m/s	Path velocity
	$VEL.ORI1	REAL	°/s	Swivel velocity
	$VEL.ORI2	REAL	°/s	Rotational velocity
Accelerations	$ACC.CP	REAL	m/s^2	Path acceleration
	$ACC.ORI1	REAL	$°/s^2$	Swivel acceleration
	$ACC.ORI2	REAL	$°/s^2$	Rotational acceleration

Nowe współrzędne pojawiają się co 12ms

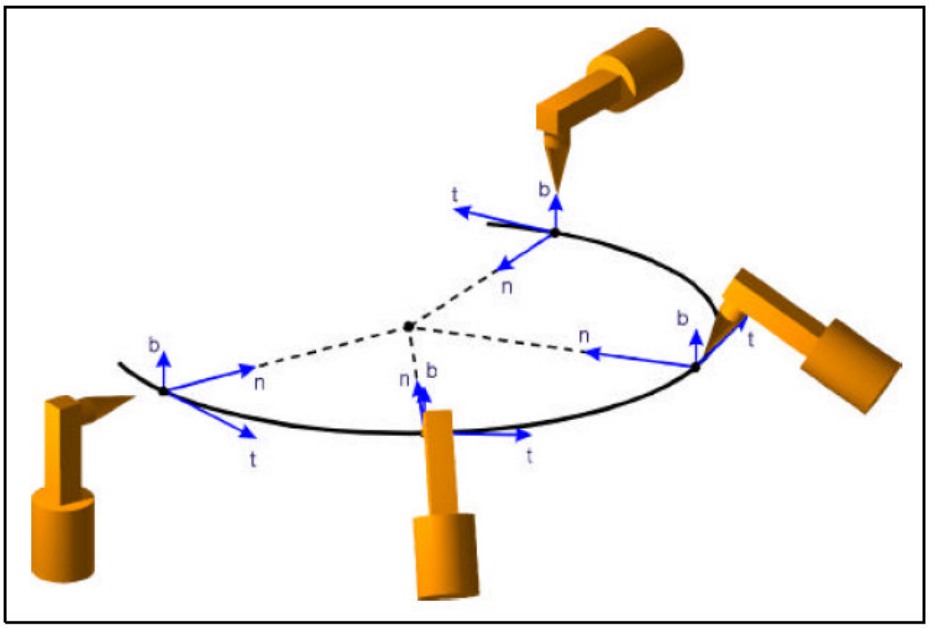
Określanie orientacji
Stała lub zmienna
$ORI_TYPE = #VAR
$ORI_TYPE = #CONSTANT ; taka jak w punkcie początkowym

Dla ruchu po okręgu
Stała lub zmienna

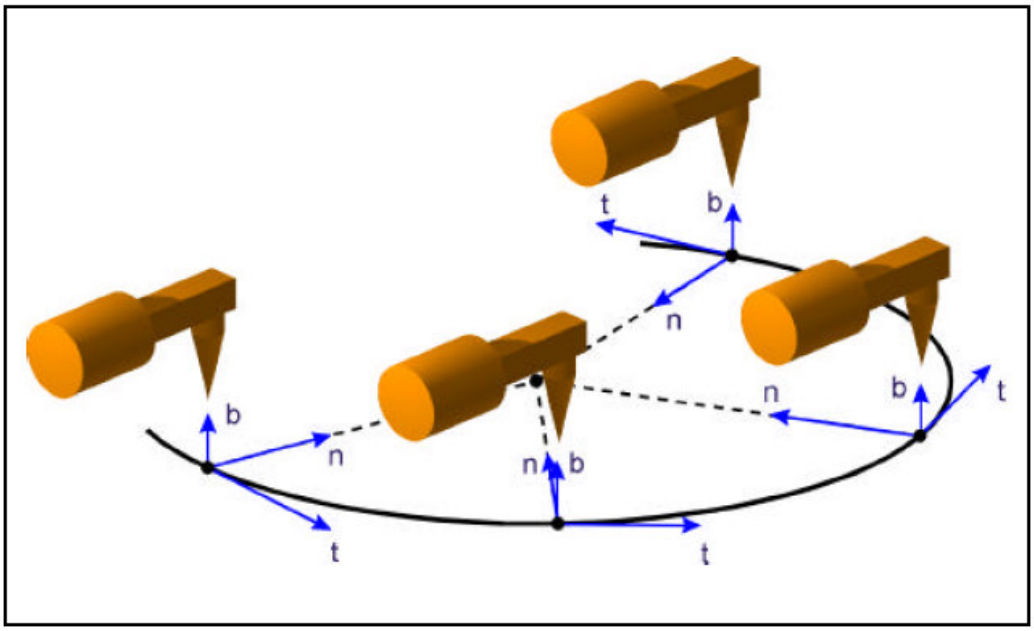
$CIRC_TYPE = #PATH ;path—related
$ORI_TYPE=#CONST



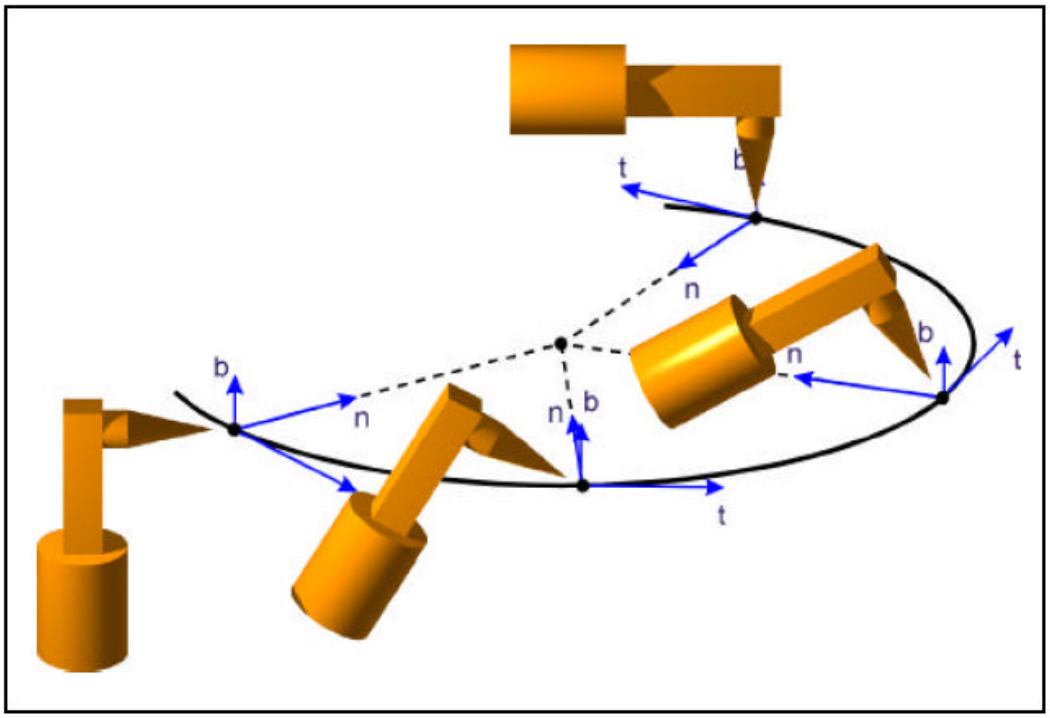
$CIRC_TYPE = #PATH ;path—related
$ORI_TYPE=#VAR



$CIRC_TYPE = #BASE ;space—related
$ORI_TYPE = #CONSTANT



$CIRC_TYPE = #BASE ;space—related
$ORI_TYPE = #VAR



Elementy struktury programu

GOTO MARK_1
...
MARK_1:

IF execution condition **THEN**
instructions
ELSE
instructions
ENDIF

SWITCH

DEF MAIN()
...
SIGNAL PROG_NR $IN[1] TO $IN[4]
;The desired program number is now stored in the
INT variable PROG_NO by the PLC
...
SWITCH PROG_NO
CASE 1 ;if PROG_NO=1
PART_1()
CASE 2 ;if PROG_NO=2
PART_2()
PART_2A()
CASE 3,4,5 ;if PROG_NO=3, 4 or 5
$OUT[3]=TRUE
PART_345()
DEFAULT ;if PROG_NO<>1,2,3,4 or 5
ERROR_UP()
ENDSWITCH
...
END

Pętle
FOR Counter **=** Start **TO** End **STEP** Increment
Instructions
ENDFOR

WHILE Execution condition
Instructions
ENDWHILE

REPEAT
Instructions
UNTIL Termination condition

LOOP
Instructions
ENDLOOP

```
DEF EXIT_PRO()
PTP HOME
LOOP ;Start of the endless loop
PTP POS_1
LIN POS_2
IF $IN[1] == TRUE THEN
EXIT ;Terminate when input 1 set
ENDIF
CIRC HELP_1,POS_3
PTP POS_4
ENDLOOP ;End of the endless loop
PTP HOME
END
```

Oczekiwanie

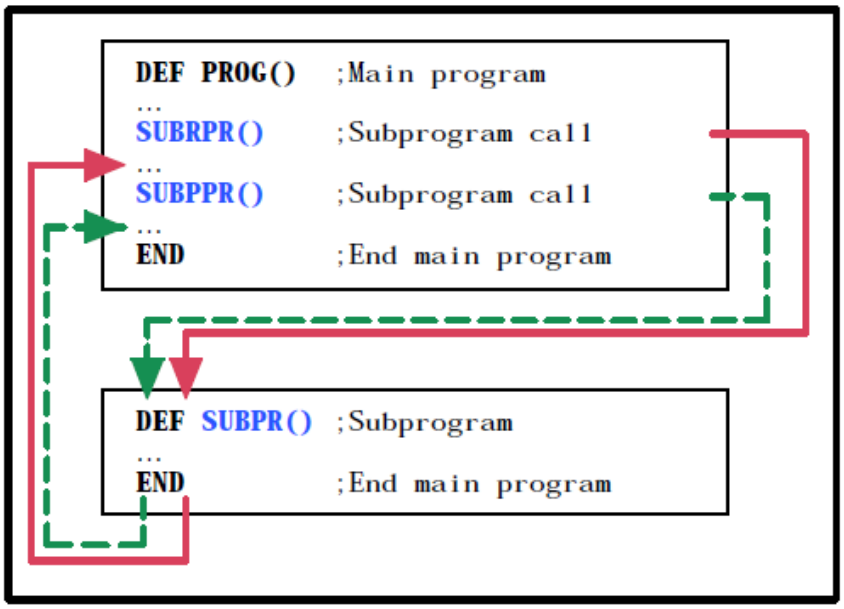
WAIT FOR condition
WAIT FOR $IN[14] ;waits until input 14 is TRUE
WAIT FOR BIT_1 == FALSE ;waits until variable BIT_1 = FALSE

WAIT SEC time
WAIT SEC 17.542
WAIT SEC TIME*4+1

Zatrzymanie programu

HALT

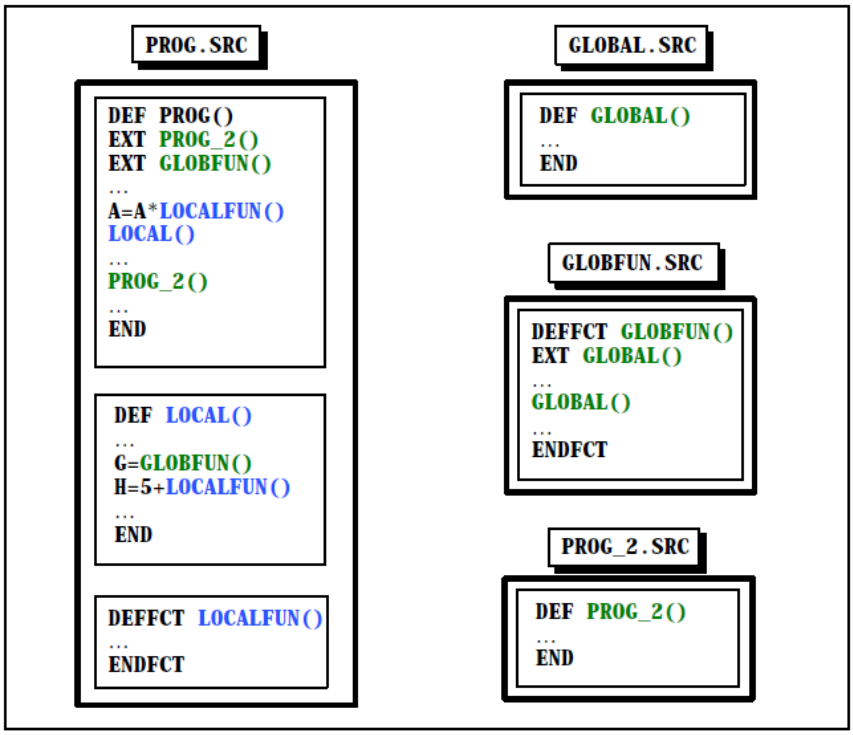
Podprogramy i funkcje



DEF SUBPROG()
...
END

DEFFCT INT FUNCTION()
...
RETURN(X)
ENDFCT

Podprogramy i funkcje lokalne i globalne
EXT PROG_3()
EXTFCT FUNCTION(REAL:IN)



Wymiana parametrów
IN przez wartość
OUT przez odwołanie
DEF CALCULATE(X:OUT,Y:IN,Z:IN,B)
EXTFCT REAL FUNCT1(REAL:IN,BOOL:OUT,REAL,CHAR:IN)

DEF ARRAY ( )
EXT BAS (BAS_COMMAND:IN,REAL:IN)
INT X[5] ;Array declaration
INT I
EXT DOUBLE (INT[]:OUT)
BAS (#INITMOV,0)
FOR I=1 TO 5
X[I]=I ;Initialize array X[]
ENDFOR ;X[1]=1,X[2]=2,X[3]=3,X[4]=4,x[5]=5
DOUBLE (X[]) ;Call subprogramm with array parameter
;X[1]=2,X[2]=4,X[3]=6,X[4]=8,X[5]=10
END

```
DEF DOUBLE(A[]:OUT)
INT A[] ;Renewed declaration of the array
INT I
FOR I=1 TO 5
A[I]=2*A[I] ;Doubling of the values in the array
ENDFOR
END
```

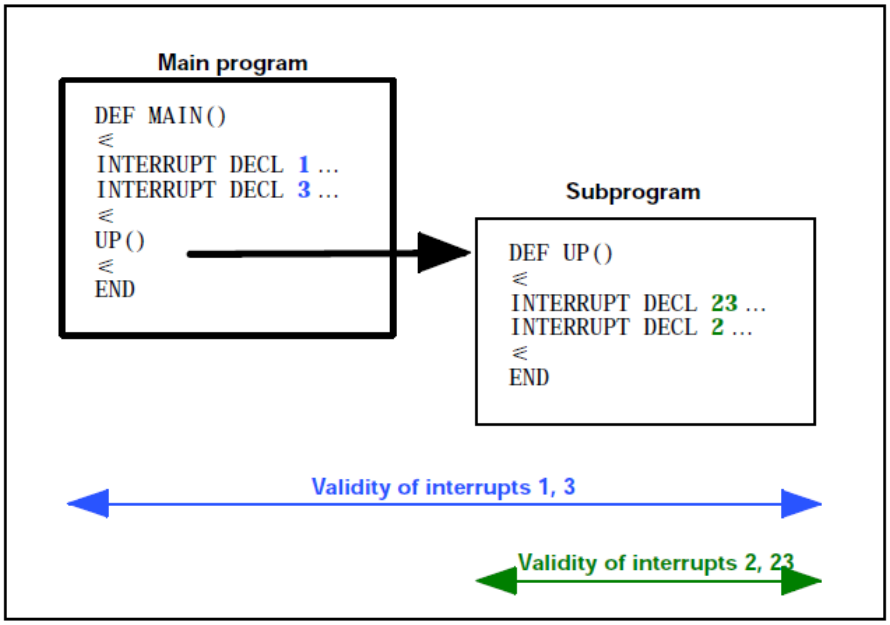
Przerwania

INTERRUPT DECL Priority **WHEN** Event **DO** Subprogram

For the meaning of the arguments:

Argument	Data type	Meaning
Priority	INT	Arithmetic expression specifying the priority of the interrupt. Priority levels 1...39 and 81...128 are available. A level 1 interrupt has the highest priority.
Event	BOOL	Logical expression defining the interrupt event. The following are permissible: S a boolean constant S a boolean variable S a signal name S a comparison
Subprogram		The name of the interrupt program to be executed when the event occurs.

```
INTERRUPT DECL 4 WHEN $IN[3]==TRUE DO UP1()
```



INTERRUPT ON 4
INTERRUPT ON
INTERRUPT OFF 4
INTERRUPT OFF

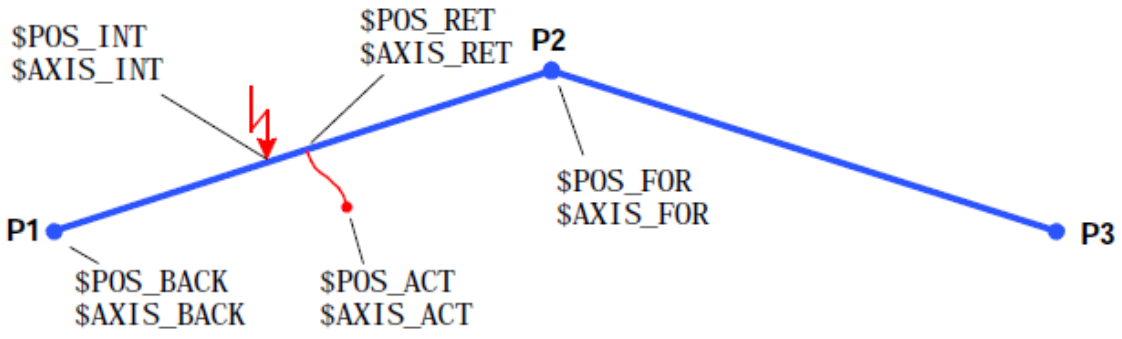
The preconditions for triggering an interrupt are:
- The interrupt must be declared (INTERRUPT DECL ...)
- The interrupt must be switched on (INTERRUPT ON)
- The interrupt must not be disabled
- The corresponding event must have occurred (edge--triggered)

Even if a program is stopped at a HALT statement, interrupts are still recognized and executed (caution: so are motion commands!).
After the instruction has been executed, the program is once again paused at the HALT statement.

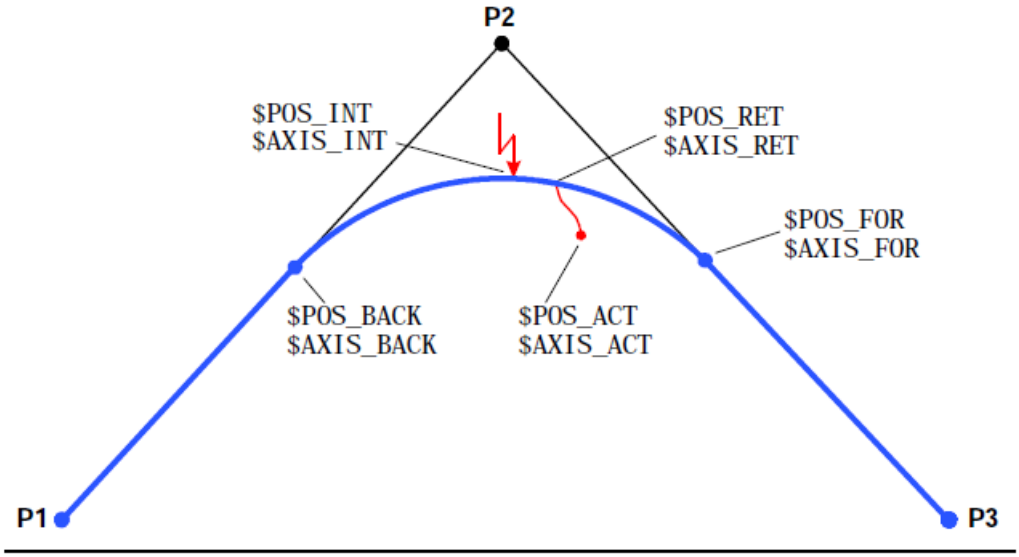
Zmienne dostępne pod przerwaniami

Joint (axis-specific)	Cartesian	Description
$AXIS_INT	$POS_INT	Position at which the interrupt was triggered
$AXIS_ACT	$POS_ACT	Current actual position
$AXIS_RET	$POS_RET	Position at which the robot left the path
$AXIS_BACK	$POS_BACK	Position of the start point of the path
$AXIS_FOR	$POS_FOR	Position of the destination point of the path

Interrupt system variables with exact positioning points



Interrupt system variables in the event of interrupt in an approximate positioning range



Zatrzymanie ruchu
BREAK

Zakończenie podprogramów przerwaniowych
RESUME

Importowanie zmiennych z innych plików
IMPORT INT OTTO_2 IS /R1/PROG_1 .. OTTO

PROG_1.SRC

```
DEF PROG_1 ( )
...
HALT
...
OTTO = 25
...
END
```

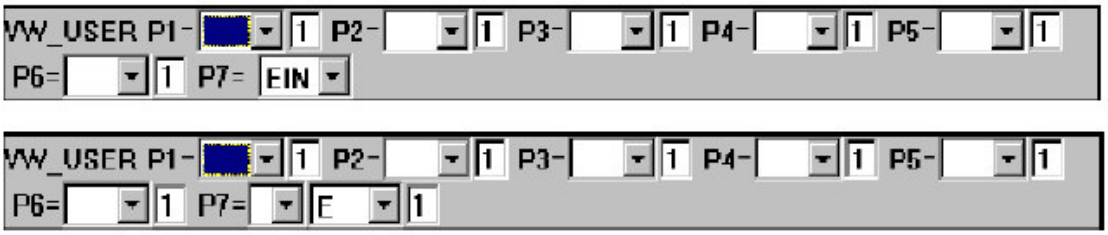
PROG_1.DAT

```
DEFDAT PROG_1 PUBLIC
INT OTTO = 0
ENDDAT
```

PROG_2.SRC

```
DEF PROG_2 ( )
IMPORT INT OTTO_2 IS /R1/PROG_1 .. OTTO
...
...
END
```

USER



Box name	Function	Range of values
P1...6=	arithmetic operand	i, bin, t, num
1	number	i (1...10), bin (1...10), t (1...10), num-999...9999
P7=	operator *1)	_, !
EIN, E	operand type	EIN (ON), AUS (OFF), E, A, M, F, T, S
1	operand number *1)	E (1...1024), A (1...1024), M (1...24), F (1...999), T (1...10), S (1...32)
*1) => not displayed with logic state "EIN (ON)", "AUS (OFF)"		

Słowa kluczowe

ANIN	Statement	Cyclic reading of the analog inputs.
ANOUT	Statement	Operator control of the analog output.
BRAKE	Statement	Braking of the robot motion in interrupt
CASE	Statement	Initiates a branch in the SWITCH statement
CCLOSE	Statement	Closing of channels.
CHANNEL	Declaration	Declaration of signal names for input and output channels.
CIRC	Statement	Circular motion.
CIRC_REL	Statement	Circular motion with relative target coordinates.
CONFIRM	Statement	Acknowledging of acknowledgement messages.
CONTINUE	Statement	Prevention of advance run stops.
COPEN	Statement	Opening an input/output channel.
CREAD	Statement	Reading of data from channels.
CWRITE	Statement	Writing of data to channels.
DECL	Declaration	Declaration of variables and arrays.
DEF	Definition	Declaration of programs and subprograms.
DEFAULT	Statement	Initiates the default branch in the SWITCH statement.
DEFDAT	Definition	Declaration of data lists.
DEFFCT	Definition	Declaration of functions.
DELAY	Parameter	Initiates the specification of the delay in the TRIGGER and ANOUT statements.
DIGIN	Statement	Cyclic reading in of digital inputs.
DISTANCE	Parameter	Initiates the specification of the switching point in the TRIGGER statement

DO	Statement	Initiates both the call of the interrupt routine in the INTERRUPT declaration and the call of a subprogram or an assignment of a value in the TRIGGER statement.
ELSE	Statement	Initiates the second statement branch in the IF statement.
END	Statement	End of a subprogram (see DEF).
ENDDAT	Statement	End of a data list (see DEFDAT).
ENDFCT	Statement	End of a function (see DEFFCT).
ENDFOR	Statement	Ends the FOR loop statement.
ENDIF	Statement	Ends the IF branching.
ENDLOOP	Statement	Ends the LOOP.
ENDSWITCH	Statement	Ends the SWITCH branches.
ENDWHILE	Statement	Ends the WHILE loop statement.
ENUM	Declaration	Declaration of enumeration types.
EXIT	Statement	Unconditional exit from loops.
EXT	Declaration	Declaration of external subprograms.
EXTFCT	Declaration	Declaration of external functions.
FOR	Statement	Counting loop or initiation of the WAIT statement condition.
GLOBAL	Declaration	Declaration of a global area of validity.
GOTO	Statement	Unconditional jump statement.
HALT	Statement	Neatly interrupt program execution and halt processing.
IF	Statement	Execution of statements depending on the result of a logical expression.
IMPORT	Declaration	Imports variables from data lists.
INTERRUPT	Statement	Definition of an interrupt function and its activation and deactivation.
IS	Statement	Initiates the source specifications in the IMPORT declaration.
LIN	Statement	Linear motion.
LIN_REL	Statement	Linear motion with relative coordinates.
LOOP	Statement	Endless loop.
MAXIMUM	Parameter	Keyword for the maximum value of analog outputs.
MINIMUM	Parameter	Keyword for the minimum value of analog outputs.
PRIO	Parameter	Initiates the specification of the priority when calling a subprogram in the TRIGGER statement.
PTP	Statement	Point--to--point motion.
PTP_REL	Statement	Point--to--point motion with relative coordinates.
PULSE	Statement	Activation of a pulse output.
REPEAT	Statement	Program loop that is always executed at least once (non--rejecting loop). The termination condition is checked at the end of the loop.
RESUME	Statement	Aborting of subprograms and interrupt routines.
RETURN	Statement	Return from functions and subprograms.
SEC	Statement	Initiates the specification of the wait time in the WAIT statement.
SIGNAL	Declaration	Declaration of signal names for input and output.
SREAD	Statement	Breaks a data set down into its constituent parts.
STRUC	Declaration	Declaration of structure types.
SWITCH	Statement	Choice between several statement branches.
SWRITE	Statement	Combination of data to form a data set.

THEN	Statement	Initiates the first statement branch in the IF statement.
TO	Statement	Separates initial and final values in the FOR statement and initiates the specification of the digital inputs/outputs in the SIGNAL declaration.
TRIGGER	Statement	Path--related triggering of a switching action synchronous to the robot motion.
UNTIL	Statement	Initiates the “end” inquiry in the REPEAT loop
WAIT	Statement	Wait for a continue condition or for a specified period of time.
WHEN	Statement	Initiates the logic expression in the INTERRUPT declaration and the specification of the path--related distance criterion in the TRIGGER statement.
WHILE	Statement	Program loop; termination condition is checked at the beginning of the loop (rejecting loop).