Software version : V3.4.0
Document version : V3.4.0
Original instructions(English)

Programming manual (V3.4.0)

DOOSAN

Table of Contents

● Preface

This manual is composed of ten chapters. Chapter 2 through 9 describe DRL commands common to whole series robot, chapter 10 describes DRL commands that apply only to A-series robot.

The contents of this manual are current as of the date this manual was written, and product-related information may be modified without prior notification to the user.

For more information on the revised manual, please visit our Robot LAB website. (https://robotlab.doosanrobotics.com/)

● Copyright

1 DRL Basic Syntex

1.1 Basic Syntax

> **Caution**
> The syntax of DRL is the same as the syntax of Python which means that DRL does not include all the syntax and features of Python.
> DRL only supports the information described in this manual.

1.1.1 Indent

Features

This function is used to separate each code block. An error occurs if the indentation is not complied.

- Indentation is used to separate each code block.
- For Indentation, 2 spaces, 4 space or tab character can be used.
- For a block, same type of indentation should be used

Example

```
Code block 1
[TAB] Code block 2

Example)
if x == 3:
y += 1

# No Error
def fnSum(x,y):
[space4]sum = x + y
[space4]return sum

# No Error
def fnSubtract(x,y):
[TAB]diff = x - y
[TAB]return diff


# Indentation error
def fnProduct(x,y):
[space4]product = x * y
[TAB]return product
```

1.1.2 Comment

Features

This function is used to provide an additional description of the code. The comments do not affect the source code since they are excluded from code processing.

A statement following "#" is recognized as a comment. A block that begins with ’’’ and ends with ’’’ is recognized as a comment.

- Comment is used to provide an additional description of the code. The comments do not affect the source code since they are excluded from code processing
- A statement following "#" is recognized as a comment.
- A block that begins with ’’’ and ends with ’’’ is recognized as a comment

Example

```
# Comment example 1
'''
Comment example 2
'''
```

1.2 Variable

1.2.1 Variable name

Features

- Variable is used to express the data value and can consist of letters, numbers, and underscores (_). The first character cannot be a number.
- Letters are case sensitive.
- An error occurs if the variable name is the same as a reserved word or interpreter internal function name.

To avoid this, use the function name as using the prefix "var _", if possible.

> **Caution**
> Never use the following reserved words as variable names or function names.

and	**assert**	**break**	**class**	**continue**
def	**del**	**elif**	**else**	**except**

exec	**finally**	**for**	**from**	**global**
if	**import**	**in**	**is**	**lambda**
list	**not**	**open**	**or**	**os**
pass	**print**	**raise**	**return**	**select**
try	**while**	**with**	**yield**	

Example

```
friend = 10
Friend = 1
_myFriend = None

pass = 10 # Syntax error
movej = 0 # movej is a DRL instruction name and should not be used as a variable.
```

1.2.2 Numeric value

Features

DRL provides numeric types such as int, float, complex number and so on.

Example

```
10, 0x10, 0o10, 0b10
3.14, 314e-2
x = 3-4j

int_value = 10
hexa_value = 0x10
octa_value = 0o10
binary_value = 0b10
double_value = 3.14
double value = 314e-2
complex_value = 3-4j
```

1.2.3 String

String

Features

All character strings are in Unicode.

- Escape characters
  \n: New line
  \t: Tab
  \r: Carrage return
  \0: Null string
  \\: back slash(\) in string
  \': single quote mark in string
  \": double quote mark in string
- String concatenation: "py"+ "tyon" → "python"
- String repeatition: "py"* 3 → "pypypy"
- String indexing: "python" [0] → "p"
- String slicing: "python" [1:4] → "yth"

Example

```
"string1"
'string2'

tp_log("st"+"ring")
    #expected result: string
tp_log("str"* 3)
    #expected result: strstrstr
tp_log("line1\nline2")
    #expected result: line1
    # line2
tp_log("\"string\"")
    #expected result: "string"
tp_log("str"[0])
    #expected result: s
tp_log("string"[1:3])
    #expected result: tr
```

+, *

Example

```
"Doosan"+ "Robotics" → "DoosanRobotics"
"Doo"* 3 → "DooDooDoo"
```

Indexing & slicing

Example

```
" Doosan" [0] → "D"
" Doosan" [1:4] → "oos"
```

1.2.4 list

Features

- The items in a list can be changed and ordered.
- A list can be indexed and sliced.
- append, insert, extend, and + operators
- count, remove, and sort operators

Example

```
colors = ["red", "green", "blue"]
tp_log(colors[0]+","+colors[1]+","+colors[2])
      #expected print result: red,green,blue

numbers = [1, 3, 5, 7, 9]
sum = 0
for number in numbers:
    sum += number
tp_log( str(sum) )
      #expected result: 25
```

1.2.5 tuple

Features

Tupule is similar to a list but is faster at processing since it is read-only.

Example

```
colors = ("red", "green", "blue")
numbers = (1, 3, 5, 7, 9)

def fnMinMax(numbers):
    numbers.sort()
    return (numbers[0], numbers[-1])
minmax = fnMinMax([4,1,2,9,5,7])
tp_log("Min Value= " + str(minmax[0]))
    #expected result: Min Value = 1
tp_log("Max Value= "+ str(minmax[1]))
    #expected result: Max Value = 9
```

1.2.6 dictionary

Features

Dictionary specifies the keys and values and lists the values.

Example

```
d = dict(a = 1, b = 3, c = 5)

colors = dict()
colors["cherry"] = "red"

ages = {'Kim':35, 'Lee':38, 'Chang':37}
tp_log("Ages of Kim = " + str(ages['Kim']))
#expected print result: Ages of Kim = 35
```

1.3 Function

1.3.1 Function Syntax

Features

- Declaration: A function begins with def and ends with colon (:).
- The beginning and ending of a function is specified by an indentation of the code.
- The interface and implementation are not separated. However, they must have been defined before they are used.
- An error occurs if the function name is the same as a reserved word or interpreter internal function name. To avoid this, use the function name as using the prefix "fn_", if possible.

> **Caution**
> Never use the following reserved words as variable names or function names.

and	**assert**	**break**	**class**	**continue**
def	**del**	**elif**	**else**	**except**
exec	**finally**	**for**	**from**	**global**
if	**import**	**in**	**is**	**lambda**
list	**not**	**open**	**or**	**os**
pass	**print**	**raise**	**return**	**select**
try	**while**	**with**	**yield**	

Sentence

def <function name> (parameter 1, parameter 2, … parameter N):

    <syntax> …

return <return value>

```
# Example
def fn_Times(a, b):
  return a * b

fn_Times (10, 10)
```

```
def fn_Times(a, b):
   return a * b

tp_log(str(fn_Times(10, 5)))
#expected result: 50

def movej():
  return 0      # movej should not be used as a function name as interpreter
        # internal function name
```

1.3.2 Scoping rule

Features

- If there is no value corresponding to the local variable name in a function, the name can be found based on the LGB rule.
    - Namespace: An area that contains the variable name
    - Local scope: A namespace and local domain inside a function
    - Global scope: Global domain outside a function
    - Built-in scope: The domain related to the contents defined by Python and an internal domain
    - LGB rule: The order of finding a variable name. local → global → built-in

Example

```
# Error: can't find simple_pi in circle_area
simple_pi = 3.14
def circle_area(r):
    return r*r*simple_pi

# simple_pi should be declared as global if it is used in circle_area_ok
def circle_area_ok(r):
    global simple_pi
    return r*r*simple_pi

tp_log(str(circle_area(3.0)))
#expected result: 28.26
```

1.3.3 Parameter mode

Features

DRL provides 3 types of parameter modes: Default parameter values, Keyword parameters and Arbitary parameters

Example

```
def fn_Times(a = 10, b = 20):
 return a * b

#Example - Default parameter value
tp_log(str(fn_Times(5)))
      #expected result: 100

#Example - Keyword parameter
tp_log(str(fn_Times(b=5)))
#expected result: 50
tp_log(str(fn_Times(a=5, b=5)))
#expected result: 25

#Example - arbitary parameter
def fn_myUnion(*args):
    for arg in args:
        tp_log(str(arg))
fn_myUnion("red", 1)
      #expected print result: red
      #                       1
```

Example - Default parameter value

```
def fn_Times(a = 10, b = 20)
    return a * b

fn_Times(5)
```

Example - Keyword parameter

```
def fn_Times(a = 10, b = 20)
    return a * b

fn_Times(a=5, b=5)
```

Example - Variable parameter

```
def fn_myUnion(*ar)
    …….

fn_myUnion("red", "white", "black")
```

1.4 Control Statement

1.4.1 pass

Features

The ‘pass’ is used when an operation is not executed.

Example

```
while True:
    pass #pass means empty statement, so while statement continues to run.
tp_log("This line never reached")
```

1.4.2 if

Features

‘if’ is a conditional statement. It can use "elif" and "else" according to whether the condition of the "if" syntax is true or false.

sentence

if <conditional statement>:

<syntax>

if <conditional statement 1>:

<Syntax 1>

elif <conditional statement 2>:

<Syntax 2>

else:

<Syntax 3>

Example - if, elif, else

```
numbers = [2,5,7]
for number in numbers:
```

```
      if number%2==0:
           tp_log(str(number) + " is even")
      else:
           tp_log (str(number) + " is odd")
#expected result:
#2 is even
#5 is odd
#7 is odd
```

1.4.3 while

Features

'while' is a conditional statement that repeats an operation according to whether the condition is true or false.

syntax

while <conditional statement>:

<syntax>

Example

```
sum = 0
cnt = 1
while cnt < 10:
    sum = sum+cnt
    cnt = cnt+1
tp_log("sum = " + str(sum))
#expected result:
#sum = 45
```

1.4.4 for

Features

'for' repeats an operation within the specified repeating range.

Syntax

for <item> in <sequential object S>:

<syntax>

Example

```
x=0
for i in range(0, 3): # i is 0 -> 1 -> 2
    x= x + 1

sum = 0
for i in range(0, 10):
    sum = sum + i
tp_log("sum = " + str(sum))
#expected result:
#sum = 45
```

1.4.5 break

Features

'break' is used to exit the internal block of a loop.

Example

```
x =0
while True:
    x = x + 1
    if x > 10:
        break

sum = 0;cnt = 0
while True:
    if cnt > 9:
        break
    sum = sum + cnt
    cnt = cnt+1
tp_log("sum = " + str(sum))
#expected print result:
#sum = 45
```

1.4.6 continue

Features

If 'continue' is used in a loop block, the loop stops further executing and returns to the beginning point of the loop.

Example

```
#<ex> 1
x=0
y=0
while True:
    x = x + 1
    if x > 10:
        continue
    y += 100

#<ex> 2
sum = 0
for i in range(0, 10):
    if i%2==0:
        continue
    sum = sum + i
tp_log("sum of odd numbers = " + str(sum))
#sum of odd numbers = 25
```

1.4.7 Else in a loop

Features

The "else" block is executed when the loop is executed until the end without being terminated by the "break" function in the middle of executing a loop.

Example

```
L = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }

for i in L:
    if i % 2 == 0:
        continue
else:
    tp_log("exit without break")
```

2 Motion-related Commands

2.1 Pos Creation

2.1.1 posj()

Definition

posj(J1=0, J2=0, J3=0, J4=0, J5=0, J6=0)

Features

This function designates the joint space angle in coordinate values.

Parameters

Parameter Name	Data Type	Default Value	Description
J1	float list posj	0	1-axis angle or angle list or posj
J2	float	0	2-axis angle
J3	float	0	3-axis angle
J4	float	0	4-axis angle
J5	float	0	5-axis angle
J6	float	0	6-axis angle

Return

Posj

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
q1 = posj()                          # q1=posj(0,0,0,0,0,0)

q2 = posj(0, 0, 90, 0, 90, 0)

q3 = posj([0, 30, 60, 0, 90, 0])     # q3=posj(0,30,60,0,90,0)
```

Related commands

- movej()(p. 58)
- amovej()(p. 99)
- movesj()(p. 78)
- amovesj()(p. 112)

2.1.2 posx()

Definition

posx(x=0, y=0, z=0, A=0, B=0, C=0, ori_type=None, sol=None, turn=None)

Features

This function designates the joint space angle in coordinate values.

Note

- From SW version V3.2, various orientation types are supported.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float list posx	0	x position or position list or posx
y	float	0	y position
z	float	0	z position
A	float	0	A orientation(z-direction rotation of reference coordinate system)
B	float	0	B orientation(y-direction rotation of A rotated coordinate system)
C	float	0	C orientation(z-direction rotation of A and B rotated coordinate system)
ori_type	int	None	orientation type • DR_ELR_ZYZ(if None): Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)
sol	int	None	solution space • None: Robot configuration is automatically selected - will be supported later • 0~7: Select robot configuration (reference function: ikin_norm)
turn	int	None	joint multi turn - Will be supported later • None: Joint multi turn is automatically selected.

Return

Value	Description
posx	task space point

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
movej([0,0,90,0,90,0], v=10, a=20)

x2 = posx(400, 300, 500, 0, 180, 0)          # Euler ZYZ

x3 = posx([350, 350, 450, 0, 180, 0])        # x3=posx(350, 350, 450, 0, 180, 0),
Euler ZYZ

x4 = posx(x2)                                   # x4=posx(400, 300, 500, 0, 180, 0),
Euler ZYZ

x5 = posx(400, 300, 500, 0, 180, 0, DR_ELR_ZYX) # Euler ZYX

x6 = posx([400, 300, 500, 0, 0, 0, 1], ori_type=DR_QUAT) # Quaternion, only list
input available

x6.ori_type = DR_ROTVEC                          # Transform to rot vec (setter
method)

x7 = posx(x6, ori_type=DR_FIX_XYZ)               # Constructor as the other
orientation type

movel(x2, v=100, a=200)
```

Related commands

- movel()(p. 63)
- movec()(p. 72)
- movejx()(p. 68)
- amovel()(p. 102)
- amovec()(p. 108)

- amovejx()(p. 105)

2.1.3 trans()

Definition

trans(pos, delta, ref, ref_out, ori_type_out)

Features

- pos (pose) defined based on the ref coordinate is moved/rotated by the amount equal to delta, and then a value converted based on the ref_out coordinate is returned.
- In case that the ref coordinate is the tool coordinate, this function retuns the value based on input parameter(pos)'s coordinate without ref_out coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx list (float[6])	-	posx or position list
delta	posx list (float[6])	-	posx or position list
ref	int	None	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate: user defined
ref_out	int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • user coordinate: user defined

ori_type_out	int	None	output orientation type • None: Follows the orientation type of input parameter(pos). • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx list (float[6])	task space point

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
p0 = posj(0,0,90,0,90,0)

movej(p0, v=30, a=30)


x1 = posx(200, 200, 200, 0, 180, 0)

delta = [100, 100, 100, 0, 0, 0]
```

```
x2 = trans(x1, delta, DR_BASE, DR_BASE)        # translation of x1 with delta in base
coordinates

x3 = trans(x1, delta, ori_type_out=DR_ELR_ZYX) # transform to euler zyx

x1_base = posx(500, 45, 700, 0, 180, 0)

x4 = trans(x1_base, [10, 0, 0, 0, 0, 0], DR_TOOL)

movel(x4, v=100, a=100, ref=DR_BASE)


uu1 = [1, 1, 0]

vv1 = [-1, 1, 0]

pos = posx(559, 34.5, 651.5, 0, 180.0, 0)

DR_userTC1 = set_user_cart_coord(uu1, vv1, pos)  #user defined coordinate system

x1_userTC1 = posx(30, 20, 100, 0, 180, 0)        #posx on user coordinate system

x9 = trans(x1_userTC1, [0, 0, 50, 0, 0, 0], DR_userTC1, DR_BASE)

movel(x9, v=100, a=100, ref=DR_BASE)
```

Related commands

- posx()(p. 28)
- addto()(p. 41)

2.1.4 posb()

Definition

posb(seg_type, posx1, posx2=None, radius=0)

Features

- Input parameters for constant-velocity blending motion (moveb and amoveb) with the Posb coordinates of each waypoint and the data of the unit path type (line or arc) define the unit segment object of the trajectory to be blended.
- Only posx1 is inputted if seg_type is a line (DR_LINE), and posx2 is also inputted if seg_type is a circle (DR_CIRCLE). Radius sets the blending radius with the continued segment.

Parameters

Parameter Name	Data Type	Default Value	Description
seg_type	Int	-	DR_LINE DR_CIRCLE
posx1	posx	-	1st task posx
posx2	posx	-	2nd task posx
radius	float	0	Blending radius [mm]

Return

Posb

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
q0 = posj(0, 0, 90, 0, 90, 0)

movej(q0,vel=30,acc=60)

x0 = posx(564, 34, 690, 0, 180, 0)

movel(x0, vel=200, acc=400)     # Moves to the start position.


x1 = posx(564, 200, 690, 0, 180, 0)

seg1 = posb(DR_LINE, x1, radius=40)

x2 = posx(564, 100, 590, 0, 180, 0)

x2c = posx(564, 200, 490, 0, 180, 0)
```

```
seg2 = posb(DR_CIRCLE, x2, x2c, radius=40)

x3 = posx(564, 300, 490, 0, 180, 0)

seg3 = posb(DR_LINE, x3, radius=40)

x4 = posx(564, 400, 590, 0, 180, 0)

x4c = posx(564, 300, 690, 0, 180, 0)

seg4 = posb(DR_CIRCLE, x4, x4c, radius=40)

x5 = posx(664, 300, 690, 0, 180, 0)

seg5 = posb(DR_LINE, x5, radius=40)

x6 = posx(564, 400, 690, 0, 180, 0)

x6c = posx(664, 500, 690, 0, 180, 0)

seg6 = posb(DR_CIRCLE, x6, x6c, radius=40)

x7 = posx(664, 400, 690, 0, 180, 0)

seg7 = posb(DR_LINE, x7, radius=40)

x8 = posx(664, 400, 590, 0, 180, 0)

x8c = posx(564, 400, 490, 0, 180, 0)

seg8 = posb(DR_CIRCLE, x8, x8c, radius=0)         # The last radius must be 0.
        # If not 0, it is processed as 0.

b_list = [seg1, seg2, seg3, seg4, seg5, seg6, seg7, seg8]

moveb(b_list, vel=200, acc=400)
```

Related commands

- posx()(p. 28)
- moveb()(p. 84)
- amoveb()(p. 118)

2.1.5 fkin()

Definition

fkin(pos, ref, ori_type)

Features

This function receives the input data of joint angles or equivalent forms (float[6]) in the joint space and returns the TCP (objects in the task space) based on the ref coordinate.

Parameters

Parameter Name	Data type	Default Value	Description
pos	posj list (float[6])	-	posj or position list
ref	int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate
ori_type	int	DR_ELR_ZYZ	orientation type • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	Task space point

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
q1 = posj(0, 0, 90, 0, 90, 0)

movej(q1,v=10,a=20)

q2 = posj(30, 0, 90, 0, 90, 0)

x2 = fkin(q2, DR_WORLD) # x2: Space coordinate at the edge of the robot (TCP)
corresponding to joint value q2

movel(x2,v=100,a=200,ref=DR_WORLD) # Linear motion to x2


x1 = fkin(q1, DR_WORLD, DR_ELR_XYZ) # x1: Space coordinate at the edge of the robot
(TCP) corresponding to joint value q2 (orientation type: euelr xyz)

movel(x1,v=100,a=200,ref=DR_WORLD) # Linear motion to x1
```

Related commands

- set_tcp()(p. 230)
- posj()(p. 27)
- posx()(p. 28)

2.1.6 ikin()

Definition

ikin(pos, sol_space, ref, ref_pos_opt, iter_threshold)

Features

This function returns the joint position corresponding to posx.sol, which is equivalent to the robot pose in the operating space, among 8 joint shapes. Joint position is returned according to closest joint position depend on option(ref_pos_opt). Through status of return value, you can check whether current robot pose is in wrist singularity or out of operating region. When DR_SOL_AUTO(255) is entered into posx.sol, it moves to robot

configuration that is the closest to the current configuration, (the smallest L2 norm in the joint space of axes 2-5) among the reachable joint combination types.

Note

- After SW version V2.9, if the accuracy of the robot is calibrated when using this function, the accuracy correction algorithm operates. The accuracy level can be adjusted according to the option (iter_threshold). Also, there may be a delay of up to 0.1 second depending on the robot posture.
- Refer to ikin_norm() command to find the inverse kinematics solution related to each robot model's well known nominal DH parameters before calibrated DH parameters.
- Please ensure the use of 'Lock' to prevent global variable collisions when updating the return value posj in ikin with Threads.

Caution

If the res_pos_opt parameter is not entered, only pos is returned. Runtime error may occur if ref, status = ikin(...) is used without entering the ref_pos_opt parameter.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx list (float[6])	-	posx or position list
sol_space	int	-	solution space • None: Not available for v3.x DRL Command (Recommend using posx.sol) • 0~7: Available for 2.x & 3.x DRL Command • DR_SOL_AUTO(255): Available for 2.x & 3.x DRL Command
ref	int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate
ref_pos_opt	int	0	Determine closest joint position depend on option among multi turn solutions • 0 : posj(0,0,0,0,0,0) is reference • 1 : current joint position is reference

iter_threshold	list (float[2])	0.005	If the accuracy has been corrected, the level of the accuracy correction algorithm The norm value of TCP position (X, Y, Z) [mm]
		0.01	If the accuracy has been corrected, the level of the accuracy correction algorithm The norm value of TCP orientation (A, B, C) [deg]

Solution Space w.r.t. Robot Configuration

Solution space	Binary	Shoulder	Elbow	Wrist
0	000	Lefty	Below	No Flip
1	001	Lefty	Below	Flip
2	010	Lefty	Above	No Flip
3	011	Lefty	Above	Flip
4	100	Righty	Below	No Flip
5	101	Righty	Below	Flip
6	110	Righty	Above	No Flip
7	111	Righty	Above	Flip
255	Auto Calculation (the smallest L2 norm in the joint space of axes 2-5)			

Return

Value	Description
posj	Joint space point
status	• 0 : Return joint position • 1 : Out of workspace • 2 : In wrist singularity region

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
## 2.x compatible code
x1 = posx(370.9, 719.7, 651.5, 90, -180, 0)
q1, status = ikin(x1, 2, DR_BASE, ref_pos_opt=0) # Joint angle q1 where the
coordinate of the robot edge is x1 (second of 8 cases), reference joint position :
posj(0,0,0,0,0,0)
# q1=posj(60.3, 81.0, -60.4, -0.0, 159.4, -29.7) (M1013, tcp=(0,0,0))
movej(q1,v=10,a=20)

## 3.2~ code
x1 = posx(370.9, 719.7, 651.5, 90, -180, 0, sol=2)
q1, status = ikin(x1, ref=DR_BASE, ref_pos_opt=0)  # Joint angle q1 where the
coordinate of the robot edge is x1 (second of 8 cases), reference joint position :
posj(0,0,0,0,0,0)
# q1=posj(60.3, 81.0, -60.4, -0.0, 159.4, -29.7) (M1013, tcp=(0,0,0))
movej(q1,v=10,a=20)
```

Related commands

- set_tcp()(p. 230)
- posj()(p. 27)
- posx()(p. 28)

2.1.7 addto()

Definition

addto(pos, add_val=None)

Features

This function creates a new posj object by adding add_val to each joint value of posj.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posj list (float[6])	-	posj or position list
add_val	list (float[6])	None	List of add values to be added to the position • No value is added if it is None or [].

Return

Value	Description
posj	Joint space point

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Example

```
q1 = posj(10, 20, 30, 40, 50, 60)
```

```
movej (q1, v=10, a=20)

q2 = addto(q1, [0, 0, 0, 0, 45, 0])

movej (q2, v=10, a=20) # The robot moves to the joint (10, 20, 30, 40, 95, 60).
q3 = addto(q2, [])

q4 = addto(q3)
```

Related commands

- posj()(p. 27)

2.1.8 ikin_norm()

Definition

ikin_norm(pos, sol_space, ref, ref_pos_opt)

Features

This function returns the joint position corresponding to posx.sol, which is equivalent to the robot pose in the operating space, among 8 joint shapes. Joint position is returned according to closest joint position depend on option(ref_pos_opt). Through status of return value, you can check whether current robot pose is in wrist singularity or out of operating region. When DR_SOL_AUTO(255) is entered into posx.sol, it moves to robot configuration that is the closest to the current configuration, (the smallest L2 norm in the joint space of axes 2-5) among the reachable joint combination types.

Note

- After SW version V2.9, It is possible to correct robot DH parameters for improving accuracy. Robots after SW version V2.10.1 are released with calibrated DH parameters for improving accuracy.
- This function returns inverse kinematics solution related to each robot model's well known nominal DH parameters before calibrated DH parameters.
- Refer to ikin() command to find the inverse kinematics solution related to calibrated DH parameters.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx	-	posx or position list

	list (float[6])		
sol_space	int	-	solution space • None: Not available for v3.x DRL Command (Recommend using posx.sol) • 0~7: Available for 2.x & 3.x DRL Command • DR_SOL_AUTO(255): Available for 2.x & 3.x DRL Command
ref	int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate
ref_pos_opt	int	0	Determine closest joint position depend on option among multi turn solutions • 0 : posj(0,0,0,0,0,0) is reference • 1 : current joint position is reference

Solution Space w.r.t. Robot Configuration

Solution space	Binary	Shoulder	Elbow	Wrist
0	000	Lefty	Below	No Flip
1	001	Lefty	Below	Flip
2	010	Lefty	Above	No Flip
3	011	Lefty	Above	Flip
4	100	Righty	Below	No Flip
5	101	Righty	Below	Flip
6	110	Righty	Above	No Flip
7	111	Righty	Above	Flip
255	Auto Calculation (the smallest L2 norm in the joint space of axes 2-5)			

Return

Value	Description
posj	Joint space point
status	• 0 : Return joint position • 1 : Out of workspace • 2 : In wrist singularity region

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
## 2.x compatible code
x1 = posx(370.9, 719.7, 651.5, 90, -180, 0)
q1, status = ikin_norm(x1, 2, DR_BASE, ref_pos_opt=0) # Joint angle q1 where the
coordinate of the robot edge is x1 (second of 8 cases), reference joint position :
posj(0,0,0,0,0,0)
# q1=posj(60.3, 81.0, -60.4, -0.0, 159.4, -29.7) (M1013, tcp=(0,0,0))
movej(q1,v=10,a=20)

## 3.2~ code
x1 = posx(370.9, 719.7, 651.5, 90, -180, 0, sol=2)
q1, status = ikin_norm(x1, ref=DR_BASE, ref_pos_opt=0) # Joint angle q1 where the
coordinate of the robot edge is x1 (second of 8 cases), reference joint position :
posj(0,0,0,0,0,0)
# q1=posj(60.3, 81.0, -60.4, -0.0, 159.4, -29.7) (M1013, tcp=(0,0,0))
movej(q1,v=10,a=20)
```

Related commands

- set_tcp()(p. 230)
- posj()(p. 27)
- posx()(p. 28)

2.1.9 get_projected_posx()

Definition

get_projected_posx(pos, rel, ref)

Features

- It converts the pos (pose) defined based on the ref coordinate system into a pos that can be expressed with P series robots using the same coordinate system and returns it.
- It returns the pos that has the minimum joint movement distance based on the current robot's position.
- In the pos, the position does not change, and it returns the coordinates that can be expressed with the minimum rotational transformation.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx list (float[6])	-	posx or position list
rel	int	DR_MV_MOD_ABS	Movement basis DR_MV_MOD_ABS : absolute DR_MV_MOD_REL : relative
ref	int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD: world coordinate • DR_TOOL : tool coordinate • user coordinate: user defined

Return

Value	Description
posx	Task space point
status	• 0 : Task point return fail • 1 : Task point return success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
q1 = posj(180, 0, 90, 0, 90, 0)
movej(q1, v=30, a=30)
x1 = posx(-670, 113, 573, 8, 178, 96)
x1_prj, status = get_projected_posx(x1, rel=DR_MV_MOD_ABS, ref=DR_BASE)
movel(x1_prj, v=1000, a=1000)
```

Related commands

- set_tcp()(p. 230)
- posj()(p. 27)
- posx()(p. 28)

2.2 Motion settings

2.2.1 set_velj()

Definition

set_velj(vel)

Features

This function sets the global velocity in joint motion (movej, movejx, amovej, or amovejx) after using this command. The default velocity is applied to the globally set vel if movej() is called without the explicit input of the velocity argument.

Parameters

Parameter Name	Data Type	Default Value	Description
vel	float	-	velocity (same to all axes) or velocity (to an axis)
	list (float[6])		

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
#1
Q1 = posj(0,0,90,0,90,0)
Q2 = posj(0,0,0,0,90,0)
```

```
movej(Q1, vel=10, acc=20)
set_velj(30) # The global joint velocity is set to 30 (deg/sec).
set_accj(60) # The global joint acceleration is set to 60 (deg/sec2). [See
set_accj().]
movej(Q2)    # The joint motion velocity to Q2 is 30 (deg/sec) which is the global
velocity.
movej(Q1, vel=20, acc=40)   # The joint motion velocity to Q1 is 20 (deg/sec) which
is the specified velocity.

#2
set_velj(20.5) # Decimal point input is possible.
set_velj([10, 10, 20, 20, 30, 10]) # The global velocity can be specified to each
axis.
```

Related commands

- set_accj()(p. 48)
- movej()(p. 58)
- movejx()(p. 68)
- movesj()(p. 78)
- amovej()(p. 99)
- amovejx()(p. 105)
- amovesj()(p. 112)
- set_tcp()(p. 230)

2.2.2 set_accj()

Definition

set_accj(acc)

Features

This function sets the global velocity in joint motion (movej, movejx, amovej, or amovejx) after using this command. The globally set acceleration is applied as the default acceleration if movej() is called without the explicit input of the acceleration argument.

Parameters

Parameter Name	Data Type	Default Value	Description
acc	float	-	acceleration (same to all axes) or acceleration (acceleration to an axis)
	list (float[6])		

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
#1
Q1 = posj(0,0,90,0,90,0)
Q2 = posj(0,0,0,0,90,0)
movej(Q1, vel=10, acc=20)
set_velj(30) # The global joint velocity is set to 30 (deg/sec). [See set_velj().]
set_accj(60) # The global joint acceleration is set to 60 (deg/sec2).
movej(Q2)    # The joint motion acceleration to Q2 is 60(deg/sec2) which is the
global acceleration.
movej(Q1, vel=20, acc=40) # The joint motion acceleration to Q1 is 40(deg/sec2) which
is the specified acceleration.

#2
set_accj(30.55)
set_accj([30, 40, 30, 30, 30, 10])
```

Related commands

- set_velj()(p. 47)
- movej()(p. 58)
- movejx()(p. 68)
- movesj()(p. 78)
- amovej()(p. 99)
- amovejx()(p. 105)
- amovesj()(p. 112)
- set_tcp()(p. 230)

2.2.3 set_velx()

Definition

set_velx(vel1, vel2, clamp)

Features

This function sets the velocity of the task space motion globally. The globally set velocity velx is applied as the default velocity if the task motion such as movel(), amovel(), movec(), movesx() is called without the explicit input of the velocity value. In the set value, vel1 and vel2 define the linear velocity and rotating velocity, respectively, of TCP. If the value of vel2 is None, it will be calculated proportionally to the linear velocity. The clamp setting can be configured as follows: When the mode is activated, the speed of all movements is limited based on the TCP's linear velocity.

- Retain current mode: None
- Deactivate mode: DR_OFF
- Activate mode: DR_ON

Note

- In TCP speed clamping mode, if the safety parameter's TCP speed is lower than the desired speed, the safety parameter's TCP speed will be applied. In reduce mode, the speed will follow the limit set by the reduce mode.
- If the clamp is set to "None" at the start of the program, the TCP speed clamping mode will remain inactive.

Parameters

Parameter Name	Data Type	Default Value	Description
vel1	float	-	velocity 1 (TCP linear velocity)
vel2	float	None	velocity 2 (TCP rotating velocity)
clamp	int	None	TCP speed clamping mode • Retain current mode: None • Deactivate mode: DR_OFF • Activate mode: DR_ON

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
# 0
set_velx(30)            # The global task velocity is set to 30 (mm/sec). The global
task angular velocity is automatically determined. Maintain the existing mode.
set_velx(30, 20)        # The global task velocity is set to 30(mm/sec) and 20(deg/
sec). Maintain the existing mode.
set_velx(30, 20, DR_ON) # The global task velocity is set to 30(mm/sec) and 20(deg/
sec). Activate the mode.

# 1
P0 = posj(0,0,90,0,90,0)
movej(Q1, 60, 30)
P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)
movel(P1, vel=10, acc=20)
set_velx(30, 20, DR_ON)   # The global task velocity is set to 30(mm/sec) and 20(deg/
sec). Activate the mode.
set_accx(60, 40)          # The global task acceleration is set to 60(mm/sec2) and
40(deg/sec2).
movel(P2)                 # The task motion velocity to P2 is 30(mm/sec) and 20(deg/
sec) which are the global velocity.
movel(P1, vel=20, acc=40) # The task motion velocity to P1 is 20(mm/sec) and 20(deg/
sec) which are the specified velocity.

# 2
set_velx(10.5, 19.4)      # Input with decimal points is supported.

# 3
set_velx(100, 20, DR_ON)  # The global task velocity is set to 100(mm/sec) and
20(deg/sec). Activate the mode.
Q1 = posj(0,0,90,0,90,0)
Q2 = posj(0,0,0,0,90,0)
movej(Q1, vel=60, acc=60) # The global task velocity is limited to 100 (mm/sec).
```

```
movej(Q2, vel=60, acc=60, velx = 50) # TCP speed is limited to 50 (mm/sec) as the
locally set velx takes priority.
```

Related commands

- set_accx()(p. 52)
- movel()(p. 63)
- movec()(p. 72)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- amovel()(p. 102)
- amovec()(p. 108)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- set_tcp()(p. 230)

2.2.4 set_accx()

Definition

set_accx(acc1, acc2)

Features

This function sets the acceleration of the task space motion globally. The globally set acceleration accx is applied as the default acceleration if the task motion such as movel(), amovel(), movec(), movesx() is called without the explicit input of the acceleration value. In the set value, acc1 and acc2 define the linear acceleration and rotating acceleration, respectively, of the TCP.

Note

- If the value of acc2 is None, it will be automatically calculated based on a predefined ratio to the linear acceleration.

Parameters

Parameter Name	Data Type	Default Value	Description
acc1	float	-	acceleration 1 (TCP linear acceleration)
acc2	float	None	acceleration 2 (TCP rotating acceleration)

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
# 0
# The global task acceleration is set to 60(mm/sec2).
# The global task angular acceleration is automatically calculated proportionally.
set_accx(60)

# The global task acceleration is set to 60(mm/sec2) and 40(deg/sec2).
set_accx(60,40)

# 1
P0 = posj(0,0,90,0,90,0)
movej(P0, vel=30, acc=30)
P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)
movel(P1, vel=10, acc=20)

set_velx(30, 20)  # The global task velocity is set to 30(mm/sec) and 20(deg/sec).
set_accx(60, 40)  # The global task acceleration is set to 60(mm/sec2) and 40(deg/
sec2).
movel(P2)                 # The task motion acceleration to P2 is 60(mm/sec2) and
40(deg/sec2) which is the global acceleration.
movel(P1, vel=20, acc=40) # The task motion acceleration to P1 is 40(mm/sec2) and
40(deg/sec2) which is the specified acceleration.
```

Related commands

- set_velx()(p. 50)
- movel()(p. 63)
- movec()(p. 72)
- movesx()(p. 81)
- moveb()(p. 84)

- move_spiral()(p. 89)
- amovel()(p. 102)
- amovec()(p. 108)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- set_tcp()(p. 230)

2.2.5 set_ref_coord()

Definition

set_ref_coord(coord)

Features

This function sets the reference coordinate system.

Parameter

Parameter Name	Data Type	Default Value	Description
coord	int	-	Reference coordinate system • DR_BASE: Base Coordinate • DR_WORLD: World Coordinate • DR_TOOL: Tool Coordinate • user Coordinate: User defined

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
p0 = posj(0,0,90,0,90,0)
movej(p0, v=30, a=30)
x1 = posx(370.9, 419.7, 651.5, 90,-180,0)
movel(x1, v=100, a=100) # Base Coordinate basis
uu1 = [-1, 1, 0]  # x-axis vector of the user coordinate system (base coordinate
basis)
vv1 = [1, 1, 0]   # y-axis vector of the user coordinate system (base coordinate
basis)
pos = posx(370.9, -419.7, 651.5, 0, 0, 0) # Origin point of the user coordinate
system
DR_USER1 = set_user_cart_coord(uu1, vv1, pos)   # Sets the user coordinate system.
set_ref_coord(DR_USER1)           # Sets DR_USER1 of the user coordinate system to
the global coordinate system.
movel([0,0,0,0,0,0],v=100,a=100)  # The global coordinate system is used if the
reference coordinate system is not specified.
                                  # Moves to the origin point and direction of the
DR_USER1 coordinate system.
movel([0,200,0,0,0,0],v=100,a=100) # Moves to the (0,200,0) point of the DR_USER1
coordinate system.
```

Related commands

- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- set_tcp()(p. 230)

2.2.6 set_auto_acceleration_mode()

Definiton

set_auto_acceleration_mode(mode, ratio)

Features

Check motion command's input velocity, acceleration values before operating motion command. If allowable torque is exceeded, this function adjusts motion velocity, acceleration automatically. Input value mode sets activation or not. Input value ratio sets auto correction ratio.

Note

1. Robot model related to this function : M, H series
2. Motion command related to this function : movej, movejx, movel, movec, moveb
3. If you don't set input value ratio, ratio is set default value 1.1. In terms of tact time, velocity and acceleration are adjusted automatically with 10% improved performance.(recommended ratio is 1.0 for stable operating)
4. When set above acceptable weight of robot model, this function is activated automatically(DR_ON, ratio=1.0).
5. Refer to the table below for models that allow weight settings exceeding the permissible load for each robot model.
    a. The tool center of gravity follows the existing maximum allowed weight in the robot model's Payload diagram. (No separate Payload diagram is required because the acceleration is automatically adjusted)
    b. ex) M1013, 12kg tool attached: refer to 10kg tool payload c.o.g. graph
    c. Payload diagram: User manual / PART 3. Installation Manual / Product Introduction / Robot Specifications / Max. Payload within operating space

Model	Allowed Weight [kg]	Extended Weight in this mode [kg]
M0617	6	8
M1013	10	12
M1509	15	17

Caution

1. Changing this function's activation or not, ratio is possible only using this command. After terminating program, setting values(mode, ratio) are maintained.

2. When set above acceptable weight of robot model, It's impossible to deactivate this function through set_auto_acceleration_mode() command. But, setting values(mode, ratio) are maintained.
3. In some cases, It is possible to reduce tact time by setting ratio greater than 1.0. But sections that allowable torque is exceeded may occur.
4. When passing blending sections during motion, It may occur that allowable torque is exceeded.
5. For unsupported robot models, if a command is used, it will be ignored, and the code will proceed. Please exercise caution when using this feature.

Parameters

Parameter Name	Data Type	Default Value	Description
mode	int	DR_ON	• DR_OFF(0) : Deactivate mode • DR_ON(1) : Activate mode
ratio	int	1.1	• Changing auto correction ratio within 0 < ratio <=1.2

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_auto_acceleration_mode(DR_ON,1.0) # Activate function, ratio=1.0

set_velj(60.0)
set_accj(100.0)
set_velx(250.0, 80.625)
set_accx(1000.0, 322.5)

home = posj(0,0,0,0,0,0)
p1 = posj(56.27, 42.77, 30.08, 0, 107.15, -56.27)
p2 = posx(501.69, 813.53, 51.47, 56.27, 180, -56.27)
p3 = posx(501.69, 813.53, 651.47, 56.27, -180, -56.27)

movej(home)
movej(p1,v=200, a=100)
movel(p2, v=1000, a=1000)
movel(p3, v=1000, a=1000)
```

Related commands

- movej()(p. 58)
- movejx()(p. 68)
- movel()(p. 63)
- moveb()(p. 84)
- movec()(p. 72)
- set_tool()(p. 232)
- set_tcp()(p. 230)

2.3 Synchronous Motion

2.3.1 movej()

Definition

movej(pos, vel, acc, time, radius, mod, ra, ref, velx)

Features

The robot moves to the target joint position (pos) from the current joint position.

- Case1: The data type of input parameter(pos) is posj ==> movej
- Case2: The data type of input parameter(pos) is posx ==> movejx (v2.x DRL Command)

Note

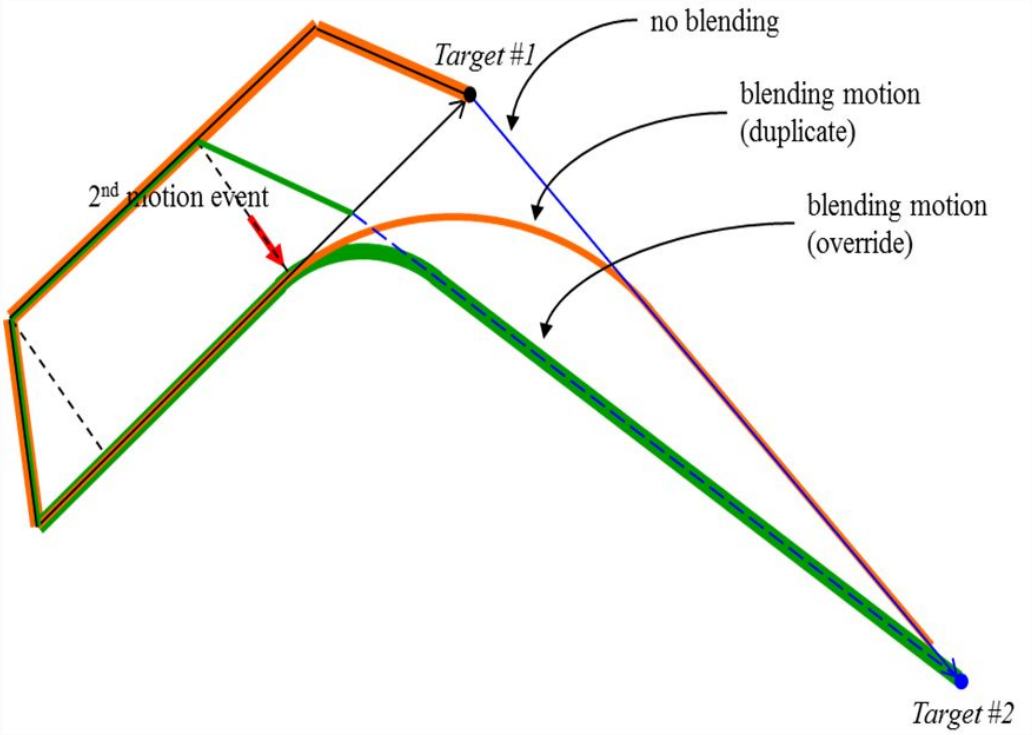
- From SW version V3.2, movejx function is supported in movej with new posx type.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posj	-	posj or joint angle list or posx
	list (float[6])		
	posx		
vel (v)	float	None	velocity (same to all axes) or velocity (to an axis)
	list (float[6])	None	
acc (a)	float	None	acceleration (same to all axes) or acceleration (acceleration to an axis)
	list (float[6])	None	
time (t)	float	None	Reach time [sec]
radius (r)	float	None	Radius for blending
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
ref	int	None	Reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • User coordinate: User defined
velx	float	None	TCP speed limiting

Note

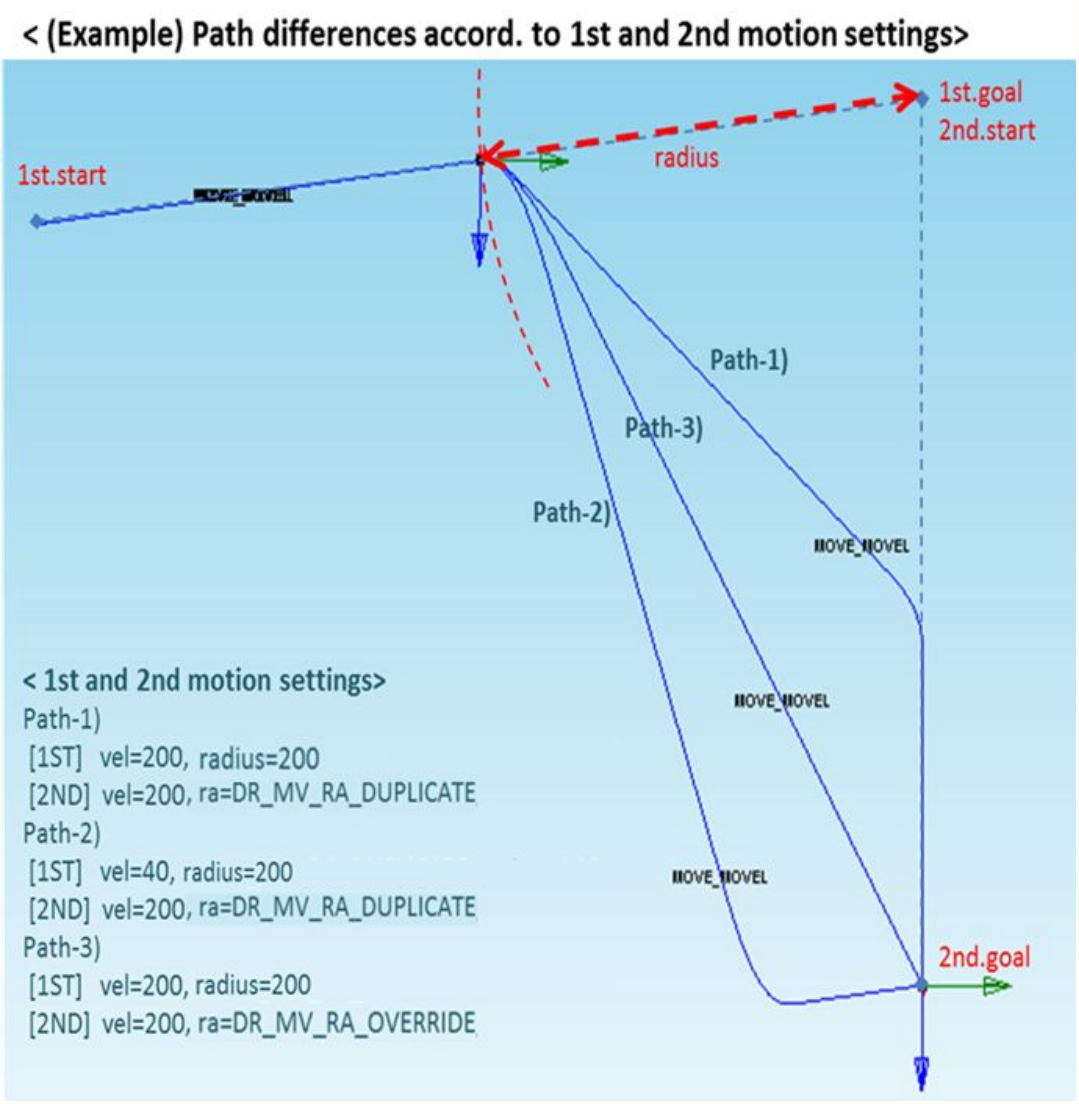
- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the radius is None, it is set to the blending radius in the blending section and 0 otherwise.
- If a new motion (following motion) is executed before the motion being executed (previous motion) is completed, the previous motion and the following motion are smoothly connected (Motion Blending). It is possible to set the option ra, which can determine whether to maintain or cancel the target of the previous motion, for the following motion. (Maintain: ra=DR_MV_RA_DUPLICATE / Cancel: ra=DR_MV_RA_OVERRIDE) For example, in the figure below, if the following motion is executed at the “2nd motion event” point of a previous motion with the target, “Target#1,” and if the option ra=DR_MV_RA_DUPLICATE is set for the following motion, the motion will follow the orange trajectory, as the motion maintains the target of the previous motion, and if option ra=DR_MV_RA_OVERRIDE is set for the following motion, the motion will follow the green trajectory, as it cancels the target of the previous motion.



- _global_velx is applied if velx is None and clamp is DR_ON in set_velx. (The initial value of _global_velx is 0.0 and can be set by set_velx.)

Caution

If the following motion is blended with the conditions of ra=DR_MV_RA_DUPLICATE and radius>0, the preceding motion can be terminated when the following motion is terminated while the remaining motion time determined by the remaining distance, velocity, and acceleration of the preceding motion is greater than the motion time of the following motion. Refer to the following image for more information.



- In versions below SW V2.8, if the blending radius exceeds 1/2 of the total moving distance, the motion is not operated because it affects the motion after blending, and the running task program is terminated when a blending error occurs
- In SW V2.8 or later, if the blending radius exceeds 1/2 of the total moving distance, the blending radius size is automatically changed to 1/2 of the total moving distance, and the change history can be checked in the information log message.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
Q1 = posj(0,0,90,0,90,0)
Q2 = posj(0,0,0,0,90,0)

# Moves to the Q1 joint angle at the velocity of 10(deg/sec) and acceleration of
20(deg/sec2).
movej(Q1, vel=10, acc=20)

# Moves to the Q2 joint angle with a reach time of 5 sec.
movej(Q2, time=5)

# Moves to the Q1 joint angle and executes the next motion when the robot is 200mm
away from the Q1 position.
movej(Q1, v=30, a=60, r=200)

# Immediately terminates the previous motion and blends it to move to the Q2 joint
angle.
movej(Q2, v=30, a=60, ra= DR_MV_RA_OVERRIDE)

# Adjust TCP speed not to exceed 100mm/s and move to Q1 joint angle
```

```
movej(Q1, v=60, a=60, velx=100)

## movejx function
P1 = posx(400,500,800,0,180,0, sol=2)
movej(P1, v=30, a=60)
```

Related commands

- posj()(p. 27)
- set_velj()(p. 47)
- set_accj()(p. 48)
- amovej()(p. 99)

2.3.2 movel()

Definition

movel(pos, vel, acc, time, radius, ref, mod, ra, app_type)

Features

The robot moves along the straight line to the target position (pos) within the task space.

Parameters

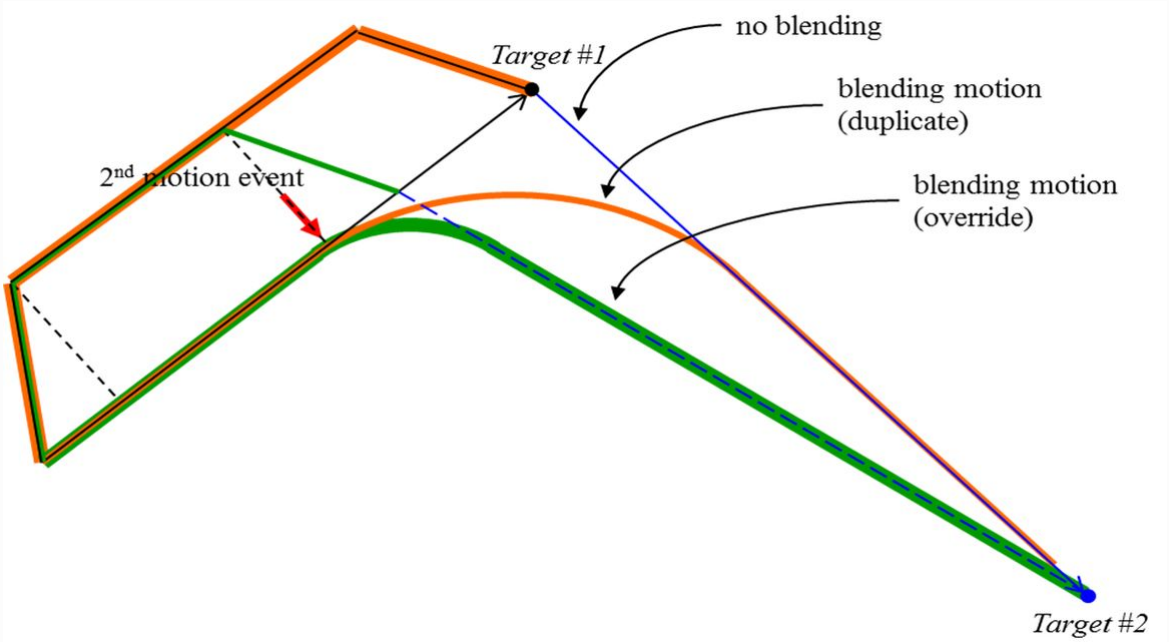
Parameter Name	Data Type	Default Value	Description
pos	posx	-	posx or position list
	list (float[6])		
vel (v)	float	None	velocity or velocity1, velocity2
	list (float[2])	None	
acc (a)	float	None	acceleration or acceleration1, acceleration2
	list (float[2])	None	
time (t)	float	None	Reach time [sec] • If the time is specified, values are processed based on time, ignoring vel and acc.

radius (r)	float	None	Radius for blending
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
app_type	int	DR_MV_APP_NONE	Application mode • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related

Note

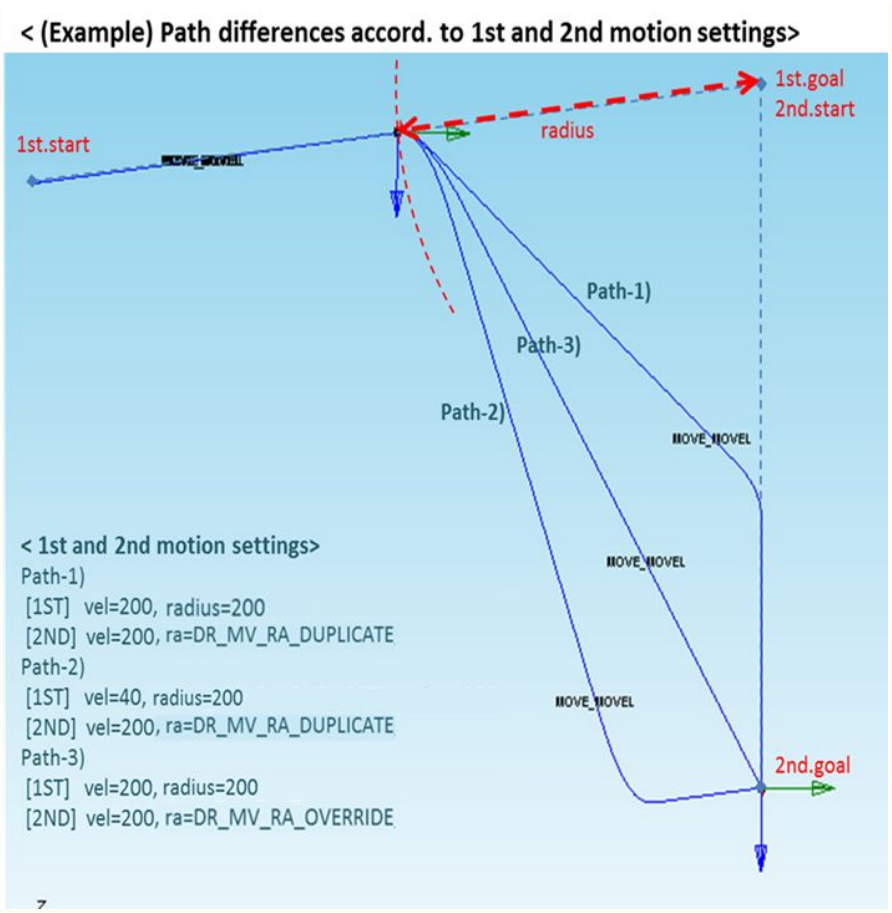
- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the radius is None, it is set to the blending radius in the blending section and 0 otherwise.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If ‘app_type’ is ‘DR_MV_APP_WELD’, parameter ‘vel’ is internally replaced by the speed setting entered in app_weld_set_weld_cond(), not the input value of ‘vel’.

- If a new motion (following motion) is executed before the motion being executed (previous motion) is completed, the previous motion and the following motion are smoothly connected (Motion Blending). It is possible to set the option ra, which can determine whether to maintain or cancel the target of the previous motion, for the following motion. (Maintain: ra=DR_MV_RA_DUPLICATE / Cancel: ra=DR_MV_RA_OVERRIDE) For example, in the figure below, if the following motion is executed at the “2nd motion event” point of a previous motion with the target, “Target#1,” and if the option ra=DR_MV_RA_DUPLICATE is set for the following motion, the motion will follow the orange trajectory, as the motion maintains the target of the previous motion, and if option ra=DR_MV_RA_OVERRIDE is set for the following motion, the motion will follow the green trajectory, as it cancels the target of the previous motion.



Caution

If the following motion is blended with the conditions of ra=DR_MV_RA_DUPLICATE and radius>0, the preceding motion can be terminated when the following motion is terminated while the remaining motion time determined by the remaining distance, velocity, and acceleration of the preceding motion is greater than the motion time of the following motion. Refer to the following image for more information.



- In versions below SW V2.8, if the blending radius exceeds 1/2 of the total moving distance, the motion is not operated because it affects the motion after blending, and the running task program is terminated when a blending error occurs
- In SW V2.8 or later, if the blending radius exceeds 1/2 of the total moving distance, the blending radius size is automatically changed to 1/2 of the total moving distance, and the change history can be checked in the information log message.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0, v=30, a=30)
P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)
P3 = posx(30,30,30,0,0,0)
movel(P1, vel=30, acc=100)
    # Moves to the P1 position with a velocity of 30(mm/sec) and acceleration of
100(mm/sec2).
movel(P2, time=5)
    # Moves to the P2 position with a reach time of 5 sec.
movel(P3, time=5, ref=DR_TOOL, mod=DR_MV_MOD_REL)
    # Moves the robot from the start position to the relative position of P3 in the
tool coordinate system
    # with a reach time of 5 sec.
movel(P2, time=5, r=10)
    # Moves the robot to the P2 position with a reach time of 5 seconds,
    # and the next motion is executed when the distance from the P2 position is 10mm.
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- amovel()(p. 102)

2.3.3 movejx()

Definition

movejx(pos, vel, acc, time, radius, ref, mod, ra, sol, velx)

Features

The robot moves to the target position (pos) within the joint space.

Since the target position is inputted as a posx form in the task space, it moves in the same way as movel. However, since this robot motion is performed in the joint space, it does not guarantee a linear path to the target position. In addition, one of 8 types of joint combination (robot configurations) corresponding to the task space coordinate system (posx) must be specified in sol (solution space). When DR_SOL_AUTO(255) is entered into sol, it moves to robot configuration that is the closest to the current configuration, (the smallest L2 norm in the joint space of axes 2-5) among the eight joint combination types.

> **Note**
> From SW version V3.2, movejx is only supported as 2.x compatible form, and the functionality for the new posx type is supported in movej.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx	-	posx or position list
	list (float[6])		
vel (v)	float	None	velocity (same to all axes) or velocity (to an axis)
	list (float[6])	None	
acc (a)	float	None	acceleration (same to all axes) or acceleration (acceleration to an axis)
	list (float[6])	None	
time (t)	float	None	Reach time [sec]
radius (r)	float	None	Radius for blending

Parameter Name	Data Type	Default Value	Description
ref	int	None	Reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • User coordinate: User defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
sol	int	0	Solution space • refer to Table. Robot Configuration
velx	float	None	TCP speed limiting

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the radius is None, it is set to the blending radius in the blending section and 0 otherwise.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- Using the blending in the preceding motion generates an error in the case of input with relative motion (mod=DR_MV_MOD_REL), and it is recommended to blend using movej() or movel().
- Refer to the description of movej() and movel() for blending according to option ra and vel/acc.
- _global_velx is applied if velx is None and clamp is DR_ON in set_velx. (The initial value of _global_velx is 0.0 and can be set by set_velx.)

Caution

- In versions below SW V2.8, if the blending radius exceeds 1/2 of the total moving distance, the motion is not operated because it affects the motion after blending, and the running task program is terminated when a blending error occurs
- In SW V2.8 or later, if the blending radius exceeds 1/2 of the total moving distance, the blending radius size is automatically changed to 1/2 of the total moving distance, and the change history can be checked in the information log message.

Robot configuration (shape vs. solution space)

Solution space	Binary	Shoulder	Elbow	Wrist
0	000	Lefty	Below	No Flip
1	001	Lefty	Below	Flip
2	010	Lefty	Above	No Flip
3	011	Lefty	Above	Flip
4	100	Righty	Below	No Flip
5	101	Righty	Below	Flip
6	110	Righty	Above	No Flip
7	111	Righty	Above	Flip
DR_SOL_AUTO(255)	Auto Calculation (the smallest L2 norm in the joint space of axes 2-5)			

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P0 = posj(0,0,90,0,90,0)

movej(P0, v=30, a=30)

P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)

movel(P2, vel=100, acc=200)       # Linear movement to P2

X_tmp, sol_init = get_current_posx()   # Obtains the current solution space from the
P2 position

movejx(P1, vel=30, acc=60, sol=sol_init)
# Moves to the joint angle with a velocity and acceleration of 30(deg/sec) and
60(deg/sec2), respectively,
# when the TCP edge is the P1 position (maintaining the solution space in the last P2
position)

movejx(P2, time=5, sol=2)
# Moves to the joint angle with a reach time of 5 sec when the TCP edge is at the P2
position
# (forcefully sets a solution space to 2)

movejx(P1, vel=[10, 20, 30, 40, 50, 60], acc=[20, 20, 30, 30, 40, 40], radius=100,
sol=2)
# Moves the robot to the joint angle when the TCP edge is at the P1 position,
# and the next motion is executed when the distance from the P2 position is 100mm.

movejx(P2, v=30, a=60, ra= DR_MV_RA_OVERRIDE, sol=2)
# Immediately terminates the last motion and blends it to move to the joint angle
```

```
# when the TCP edge is at the P2 position.

movejx(P1, v=60, a=60, sol=255, velx=100)
# Adjust the TCP speed so that it does not exceed 100 mm/s, and move to the joint
angle
# when the TCP edge is at the P1 position.
```

Related commands

- posx()(p. 28)
- set_velj()(p. 47)
- set_accj()(p. 48)
- get_current_posx()(p. 166)
- amovejx()(p. 105)

2.3.4 movec()

Definition

movec(pos, pos2, vel, acc, time, radius, ref, mod, angle, ra, ori, app_type)

Features

The robot moves along an arc to the target pos (pos2) via a waypoint (pos1) or to a specified angle from the current position in the task space.

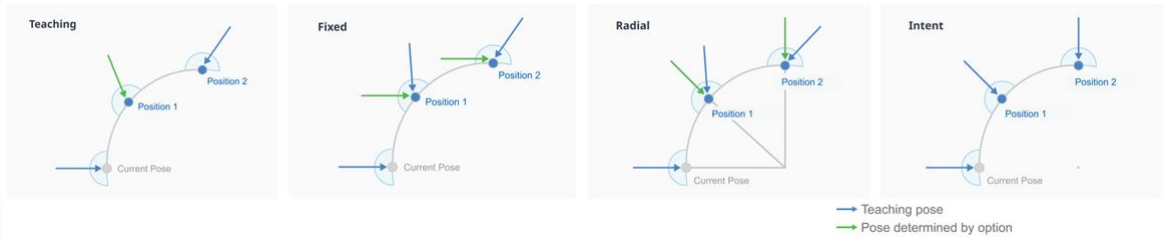
Parameters

Parameter Name	Data Type	Default Value	Description
pos	posj	-	posx or position list
	list (float[6])		
pos2	posx	-	posx or position list
	list (float[6]		
vel (v)	float	None	velocity or velocity1, velocity2
	list (float[2])	None	

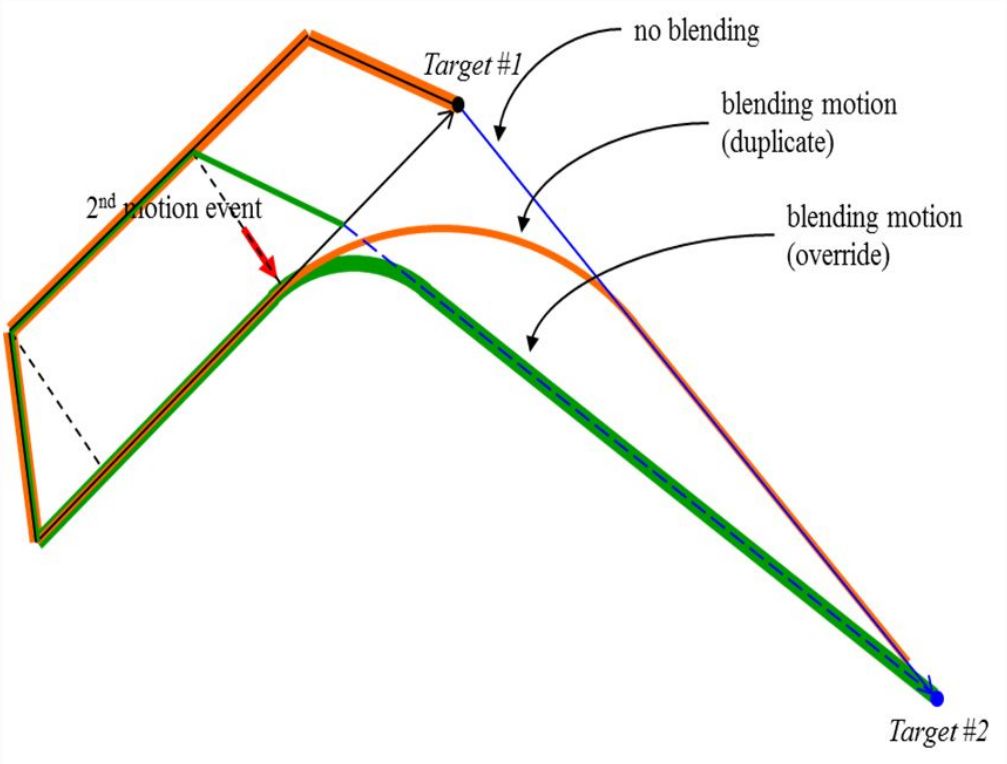
acc (a)	float	None	acceleration or acceleration1, acceleration2
	list (float[2])	None	
time (t)	float	None	Reach time [sec]
radius (r)	float	None	Radius for blending
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
angle (an)	float	None	angle or angle1, angle2
	list (float[2])		
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
ori	int	DR_MV_ORI_TEACH	Orientation mode • DR_MV_ORI_TEACH: orientation changes continuously from the initial to the final taught value • DR_MV_ORI_FIXED: orientation holds with the initial orientation • DR_MV_ORI_RADIAL: orientation changes radially from the initial. • DR_MV_ORI_INTENT: orientation changes according to the value set by the user.
app_type	int	DR_MV_APP_NONE	Application mode • DR_MV_APP_NONE: application related • DR_MV_APP_WELD: Welding application related

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius, angle:an)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the radius is None, it is set to the blending radius in the blending section and 0 otherwise.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If the mod is DR_MV_MOD_REL, pos1 and pos2 are defined in the relative coordinate system of the previous pos. (pos1 is the relative coordinate from the starting point while pos2 is the relative coordinate from pos1.)
- If the angle is None, it is set to 0.
- If only one angle is entered, the angle applied will be the total rotation angle of the circular path.
- If two angle values are inputted, angle1 refers to the total rotating angle moving at a constant velocity on the circular path while angle2 refers to the rotating angle in the rotating section for acceleration and deceleration. In that case, the total moving angle angle1 + 2 X angle2 moves along the circular path.
- If ‘app_type’ is ‘DR_MV_APP_WELD’, parameter ‘vel’ is internally replaced by the speed setting entered in app_weld_set_weld_cond(), not the input value of ‘vel’.
- ‘ori’ (orientation mode) is defined as below.
    a. DR_MV_ORI_TEACH(orientation based on teaching) : It moves while changing the current pose to the teaching pose of Pose 2, proportionate to the movement distance. The orientation of the taught pose, ‘pose 1’ is ignored.
    b. DR_MV_ORI_FIXED(fixed orientation) : Move along the path while maintaining the initial orientation up to the taught pose, ‘pose2’.
    c. DR_MV_ORI_RADIAL(orientation constrained radially) : Move along the path while maintaining radial orientation at the initial pose to the ‘pose 2’.
    d. DR_MV_ORI_INTENT (orientation set by user): Moves from the current posture to pose 2 through the taught posture of pose 1. At this time, movement from the current posture to pose 1 and movement from pose 1 to pose 2 are moved by the shortest rotation distance.



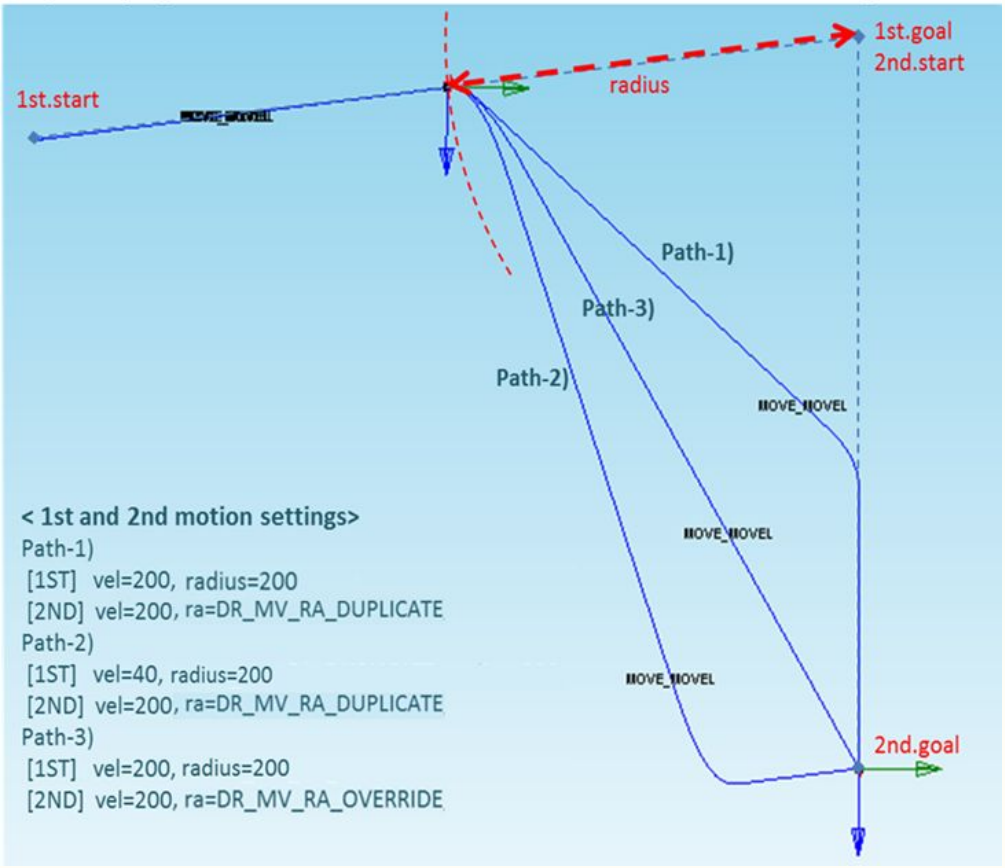
- If a new motion (following motion) is executed before the motion being executed (previous motion) is completed, the previous motion and the following motion are smoothly connected (Motion Blending). It is possible to set the option ra, which can determine whether to maintain or cancel the target of the previous motion, for the following motion. (Maintain: ra=DR_MV_RA_DUPLICATE / Cancel: ra=DR_MV_RA_OVERRIDE) For example, in the figure below, if the following motion is executed at the “2nd motion event” point of a previous motion with the target, “Target#1,” and if the option ra=DR_MV_RA_DUPLICATE is set for the following motion, the motion will follow the orange trajectory, as the motion maintains the target of the previous motion, and if option ra=DR_MV_RA_OVERRIDE is set for the following motion, the motion will follow the green trajectory, as it cancels the target of the previous motion.



Caution

If the following motion is blended with the conditions of ra=DR_MV_RA_DUPLICATE and radius>0, the preceding motion can be terminated when the following motion is terminated while the remaining motion time determined by the remaining distance, velocity, and acceleration of the preceding motion is greater than the motion time of the following motion. Refer to the following image for more information.

< (Example) Path differences accord. to 1st and 2nd motion settings>



- In versions below SW V2.8, if the blending radius exceeds 1/2 of the total moving distance, the motion is not operated because it affects the motion after blending, and the running task program is terminated when a blending error occurs
- In SW V2.8 or later, if the blending radius exceeds 1/2 of the total moving distance, the blending radius size is automatically changed to 1/2 of the total moving distance, and the change history can be checked in the information log message.
- For the **DR_MV_ORI_INTENT** option, the rotational speed is dependent on translational speed. If the translation speed is high, the rotational speed will also increase.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#1
P0 = posj(0,0,90,0,90,0)
movej(P0)
set_velx(30,20) # Set the global task velocity to 30(mm/sec) and 20(deg/sec).
set_accx(60,40) # Set the global task acceleration to 60(mm/sec2) and 40(deg/sec2).

P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)
P3 = posx(100, 300, 700, 45, 0, 0)
P4 = posx(500, 400, 800, 45, 45, 0)

movec(P1, P2, vel=30)
# Moves to P2 with a velocity of 30(mm/sec) and global acceleration of 60(mm/sec2)
# via P1 along the arc trajectory.
movej(P0)
movec(P3, P4, vel=30, acc=60)
# Moves to P4 with a velocity of 30(mm/sec) and acceleration of 60(mm/sec2).
# via P3 along the arc trajectory
movej(P0).
movec(P2, P1, time=5)
# Moves with a global velocity of 30(mm/sec) and acceleration of 60(mm/sec2).
# to P1 along the arc trajectory via P2 at the 5-second point.
movec(P3, P4, time=3, radius=100)
# Moves along the arc trajectory to P4 via P3 with a reach time of 3 seconds
# and then executes the next motion at a distance of 100mm from the P4 position.
movec(P2, P1, ra=DR_MV_RA_OVERRIDE)
# Immediately terminates the last motion and blends it to move to the P1 position.
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- amovec()(p. 108)

2.3.5 movesj()

Definition

movesj(pos_list, vel, acc, time, mod)

Features

The robot moves along a spline curve path that connects the current position to the target position (the last waypoint in pos_list) via the waypoints of the joint space input in pos_list.

The input velocity/acceleration means the maximum velocity/acceleration in the path, and the acceleration and deceleration during the motion are determined according to the position of the waypoint.

Parameters

Parameter Name	Data Type	Default Value	Description
pos_list	list (posj)	-	posj list
vel (v)	float	None	velocity(same to all axes) or velocity(to an axis)
	list (float[6])		
acc (a)	float	None	acceleration (same to all axes) or acceleration (acceleration to an axis)
	list (float[6])		
time (t)	float	None	Reach time [sec]

mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the mod is DR_MV_MOD_REL, each pos in the pos_list is defined in the relative coordinate of the previous pos. (If pos_list=[q1, q2, ...,q(n-1), q(n)], q1 is the relative angle of the starting point while q(n) is the relative coordinate of q(n-1).)
- This function does not support online blending of previous and subsequent motions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#CASE 1) Absolute angle input (mod= DR_MV_MOD_ABS)

q0 = posj(0,0,0,0,0,0)
movej(q0, vel=30, acc=60)   # Moves in joint motion to the initial position (q0).
q1 = posj(10, -10, 20, -30, 10, 20) # Defines the posj variable (joint angle) q1.
q2 = posj(25, 0, 10, -50, 20, 40)
q3 = posj(50, 50, 50, 50, 50, 50)
q4 = posj(30, 10, 30, -20, 10, 60)
q5 = posj(20, 20, 40, 20, 0, 90)

qlist = [q1, q2, q3, q4, q5]    # Defines the list (qlist) which is a set of q1-q5 as
the waypoints.

movesj(qlist, vel=30, acc=100)
    # Moves the spline curve that connects the waypoints defined in the qlist.
    # with a maximum velocity of 30(mm/sec) and maximum acceleration of 100(mm/sec2)

#CASE 2) Relative angle input (mod= DR_MV_MOD_REL)
q0 = posj(0,0,0,0,0,0)
movej(q0, vel=30, acc=60)          # Moves in joint motion to the initial position
(q0).
dq1 = posj(10, -10, 20, -30, 10, 20)         # Defines dq1 (q1=q0+dq1) as the
relative joint angle of q0
dq2 = posj(15, 10, -10, -20, 10, 20)         # Defines dq2 (q2=q1+dq2) as the
relative joint angle of q1
dq3 = posj(25, 50, 40, 100, 30, 10)          # Defines dq3 (q3=q2+dq3) as the
relative joint angle of q2
dq4 = posj(-20, -40, -20, -70, -40, 10)      # Defines dq4 (q4=q3+dq4) as the
relative joint angle of q3
dq5 = posj(-10, 10, 10, 40, -10, 30)         # Defines dq5 (q5=q4+dq5) as the
relative joint angle of q4

dqlist = [dq1, dq2, dq3, dq4, dq5]
    # Defines the list (dqlist) which is a set of q1-q5 as the relative waypoints.

movesj(dqlist, vel=30, acc=100, mod= DR_MV_MOD_REL )
    # Moves the spline curve that connects the relative waypoints defined in the
dqlist
    # with a maximum velocity of 30(mm/sec) and maximum acceleration of 100(mm/sec2)
(same motion as CASE-1).
```

Related commands

- posj()(p. 27)
- set_velj()(p. 47)
- set_accj()(p. 48)
- amovesj()(p. 112)

2.3.6 movesx()

Definition

movesx(pos_list, vel, acc, ref, mod, vel_opt)

Features

The robot moves along a spline curve path that connects the current position to the target position (the last waypoint in pos_list) via the waypoints of the task space input in pos_list.

The input velocity/acceleration means the maximum velocity/acceleration in the path and the constant velocity motion is performed with the input velocity according to the condition if the option for the constant speed motion is selected.

Parameters

Parameter Name	Data Type	Default Value	Description
pos_list	list (posx)	-	posx list
vel (v)	float list (float[2])	None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None	acceleration or acceleration1, acceleration2
time (t)	float	None	Reach time [sec]
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative

vel_opt	int	DR_MVS_VEL_NONE	Velocity option • DR_MVS_VEL_NONE: None • DR_MVS_VEL_CONST: Constant velocity

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If the mod is DR_MV_MOD_REL, each pos in the pos_list is defined in the relative coordinate of the previous pos. (If pos_list=[p1, p2, ...,p(n-1), p(n)], p1 is the relative angle of the starting point while p(n) is the relative coordinate of p(n-1).)
- This function does not support online blending of previous and subsequent motions.

Caution

The constant velocity motion according to the distance and velocity between the waypoints cannot be used if the "vel_opt= DR_MVS_VEL_CONST" option (constant velocity option) is selected, and the motion is automatically switched to the variable velocity motion (vel_opt= DR_MVS_VEL_NONE) in that case.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#CASE 1) Absolute coordinate input (mod= DR_MV_MOD_ABS)
P0 = posj(0,0,90,0,90,0)
movej(P0, v=30, a=30)
x0 = posx(600, 43, 500, 0, 180, 0)  # Defines the posx variable (space coordinate/
pose) x0.
movel(x0, vel=100, acc=200) # Linear movement to the initial position x0
x1 = posx(600, 600, 600, 0, 175, 0) # Defines the posx variable (space coordinate/
pose) x1.
x2 = posx(600, 750, 600, 0, 175, 0)
x3 = posx(150, 600, 450, 0, 175, 0)
x4 = posx(-300, 300, 300, 0, 175, 0)
x5 = posx(-200, 700, 500, 0, 175, 0)
x6 = posx(600, 600, 400, 0, 175, 0)

xlist = [x1, x2, x3, x5, x6]    # Defines the list (xlist) which is a set of x1-x6 as
the waypoints.

movesx(xlist, vel=[100, 30], acc=[200, 60], vel_opt=DR_MVS_VEL_NONE)
    # Moves the spline curve that connects the waypoints defined in the xlist
    # with a maximum velocity of 100, 30(mm/sec, deg/sec) and maximum acceleration of
200(mm/sec2) and
    # 60(deg/sec2).
movesx(xlist, vel=[100, 30], acc=[200, 60], time=5, vel_opt=DR_MVS_VEL_CONST)
    # Moves the spline curve that connects the waypoints defined in the xlist
    # with a constant velocity of 100, 30(mm/sec, deg/sec).

#CASE 2) Relative coordinate input (mod= DR_MV_MOD_REL)
P0 = posj(0,0,90,0,90,0)
movej(P0)
```

```
x0 = posx(600, 43, 500, 0, 180, 0)  # Defines the posx variable (space coordinate/
pose) x0.
movel(x0, vel=100, acc=200) # Linear movement to the initial position x0
dx1 = posx(0, 557, 100, 0, -5, 0)
    # Definition of relative coordinate dx1 to x0 (Homogeneous transformation of dx1
based in x1= x0)
dx2 = posx(0, 150, 0, 0, 0, 0)
    # Definition of relative coordinate dx2 to x1 (Homogeneous transformation of dx2
based in x2= x1)
dx3 = posx(-450, -150, -150, 0, 0, 0)
    # Definition of relative coordinate dx3 to x2 (Homogeneous transformation of dx3
based in x3= x2)
dx4 = posx(-450, -300, -150, 0, 0, 0)
    # Definition of relative coordinate dx4 to x3 (Homogeneous transformation of dx4
based in x4= x3)
dx5 = posx(100, 400, 200, 0, 0, 0)
    # Definition of relative coordinate dx5 to x4 (Homogeneous transformation of dx5
based in x5= x4)
dx6 = posx(800, -100, -100, 0, 0, 0)
    # Definition of relative coordinate dx6 to x5 (Homogeneous transformation of dx6
based in x6= x5)

dxlist = [dx1, dx2, dx3, dx4, dx5, dx6]
    # Defines the list (dxlist) which is a set of dx1-dx6 as the waypoints.

movesx(dxlist, vel=[100, 30], acc=[200, 60], mod= DR_MV_MOD_REL,
vel_opt=DR_MVS_VEL_NONE)
    # Moves the spline curve that connects the waypoints defined in the dxlist
    # with a maximum velocity of 100, 30 (mm/sec, deg/sec)
    # and maximum acceleration of 200(mm/sec2), and 60(deg/sec2) (same motion as
CASE-1).
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- amovesx()(p. 115)

2.3.7 moveb()

Definition

moveb(pos_list, vel, acc, time, ref, mod, app_type)

Features

This function takes a list that has one or more path segments (line or circle) as arguments and moves at a constant velocity by blending each segment into the specified radius. Here, the radius can be set through posb.

Parameters

Parameter Name	Data Type	Default Value	Description
pos_list	list (posb)	-	posb list
vel (v)	float list (float[2])	None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None	acceleration or acceleration1, acceleration2
time (t)	float	None	Reach time [sec] * If the time is specified, values are processed based on time, ignoring vel and acc.
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: User defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
app_type	int	DR_MV_APP_NONE	Application mode • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)

- Up to 50 arguments can be entered in posb_list.
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If the mod is DR_MV_MOD_REL, each pos in the posb_list is defined in the relative coordinate of the previous pos.
- If 'app_type' is 'DR_MV_APP_WELD', parameter 'vel' is internally replaced by the speed setting entered in app_weld_set_weld_cond(), not the input value of 'vel'.

Caution

- A user input error is generated if the blending radius in posb is 0.
- A user input error is generated due to the duplicated input of Line if contiguous Line-Line segments have the same direction.
- A user input error is generated to prevent a sudden acceleration if the blending condition causes a rapid change in direction.
- This function does not support online blending of previous and subsequent motions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Init Pose @ Jx1
Jx1 = posj(45,0,90,0,90,45)          # initial joint position
X0 = posx(370, 420, 650, 0, 180, 0)   # initial task position
```

```
# CASE 1) ABSOLUTE
# Absolute Goal Poses
X1 =  posx(370, 670, 650, 0, 180, 0)
X1a = posx(370, 670, 400, 0, 180, 0)
X1a2= posx(370, 545, 400, 0, 180, 0)
X1b = posx(370, 595, 400, 0, 180, 0)
X1b2= posx(370, 670, 400, 0, 180, 0)
X1c = posx(370, 420, 150, 0, 180, 0)
X1c2= posx(370, 545, 150, 0, 180, 0)
X1d = posx(370, 670, 275, 0, 180, 0)
X1d2= posx(370, 795, 150, 0, 180, 0)

seg11 = posb(DR_LINE, X1, radius=20)
seg12 = posb(DR_CIRCLE, X1a, X1a2, radius=20)
seg14 = posb(DR_LINE, X1b2, radius=20)
seg15 = posb(DR_CIRCLE, X1c, X1c2, radius=20)
seg16 = posb(DR_CIRCLE, X1d, X1d2, radius=20)
b_list1 = [seg11, seg12, seg14, seg15, seg16]
# The blending radius of the last waypoint (seg16) is ignored.

movej(Jx1, vel=30, acc=60, mod=DR_MV_MOD_ABS)
# Joint motion to the initial angle (Jx1)
movel(X0, vel=150, acc=250, ref=DR_BASE, mod=DR_MV_MOD_ABS)
# Line motion to the initial position (X0)
moveb(b_list1, vel=150, acc=250, ref=DR_BASE, mod=DR_MV_MOD_ABS)
    # Moves the robot from the current position through a trajectory consisting of
seg11(LINE), seg12(CIRCLE), seg14(LINE),
    # seg15(CIRCLE), and seg16(CIRCLE) with a constant velocity of 150(mm/sec) with
the exception of accelerating and decelerating sections.
```

```
    # (The final point is X1d2.) Blending to the next segment begins
    # when the distance of 20mm from the end point (X1, X1a2, X1b2, X1c2, and X1d2)
of each segment
    # is reached.
```

```
# CASE 2) RELATIVE
# Relative Goal Poses
dX1 =  posx(0, 250, 0, 0, 0, 0)
dX1a = posx(0, 0, -150, 0, 0, 0)
dX1a2= posx(0, -125, 0, 0, 0, 0)
dX1b = posx(0, 50, 0, 0, 0, 0)
dX1b2= posx(0, 75, 0, 0, 0, 0)
dX1c = posx(0, -250, -250, 0, 0, 0)
dX1c2= posx(0, 125, 0, 0, 0, 0)
dX1d = posx(0, 125, 125, 0, 0, 0)
dX1d2= posx(0, 125, -125, 0, 0, 0)

dseg11 = posb(DR_LINE, dX1, radius=20)
dseg12 = posb(DR_CIRCLE, dX1a, dX1a2, radius=20)
dseg14 = posb(DR_LINE, dX1b2, radius=20)
dseg15 = posb(DR_CIRCLE, dX1c, dX1c2, radius=20)
dseg16 = posb(DR_CIRCLE, dX1d, dX1d2, radius=20)
db_list1 = [dseg11, dseg12, dseg14, dseg15, dseg16]
# The blending radius of the last waypoint (dseg16) is ignored.

movej(Jx1, vel=30, acc=60, mod=DR_MV_MOD_ABS)
# Joint motion to the initial angle (Jx1)
movel(X0, vel=150, acc=250, ref=DR_BASE, mod=DR_MV_MOD_ABS)
# Line motion to the initial position (X0)
moveb(b_list1, vel=150, acc=250, ref=DR_BASE, mod=DR_MV_MOD_ABS)
    # Moves the robot from the current position through a trajectory consisting of
dseg11(LINE), dseg12(CIRCLE), dseg14(LINE),
    # dseg15(CIRCLE), and dseg16(CIRCLE) with a constant velocity of 150(mm/sec) with
the exception of accelerating and decelerating sections. (The final point is X1d2.)
    # Blending to the next segment begins when the distance of 20mm from the end
point (X1, X1a2, X1b2, X1c2, and X1d2) of each segment is reached. (The path is the
same as CASE#1.)
```

Related commands

- posb()(p. 33)
- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- amoveb()(p. 118)

2.3.8 move_spiral()

Features

In the specified coordinate system (ref), it performs the motion of a conical spiral trajectory that moves outward in the radial direction (rad_dir) with the current position as the center of the spiral, or moves inward in the radial direction with the target point as the center of the spiral. The target point can be defined by X, Y, Z coordinate values or by the maximum radius (rmax) and the axial distance traveled (lmax). It follows a spiral trajectory that rotates on the specified axis (axis) in a plane perpendicular to it, rotating the specified number of revolutions (rev).

Parameters

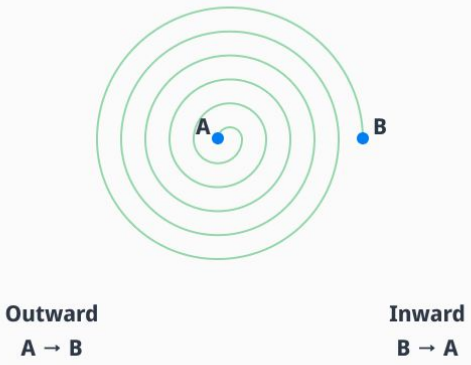
Parameter Name	Data Type	Default Value	Range	Description
rev	float	10	rev > 0	Total number of revolutions
rmax	float	None	rmax > 0	Final spiral radius [mm]
lmax	float	0		Distance moved in the axis direction [mm]
vel (v)	float	None		velocity
acc (a)	float	None		acceleration
time (t)	float	None	time ≥ 0	Total execution time <sec>
axis	int	DR_AXIS_Z	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis
ref	int	DR_TOOL	-	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate : user defined
pos	posx	None		posx or position list (X,Y,Z)

Parameter Name	Data Type	Default Value	Range	Description
	list(float[3]) list(float[6])			
mod	int	DR_MV_MOD_ABS		Movement basis • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative
rad_dir	int	DR_SPIRAL_OUTWARD		Radial direction • DR_SPIRAL_OUTWARD: toward outward • DR_SPIRAL_INWARD: toward inward
rot_dir	int	DR_ROT_FORWARD		Rotation direction • DR_ROT_FORWARD : Forward rotation (+) • DR_ROT_REVERSE : Reverse rotation (-)
radius (r)	float	None		Radius for blending
ra	int	DR_MV_RA_DUPLICATE		Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override

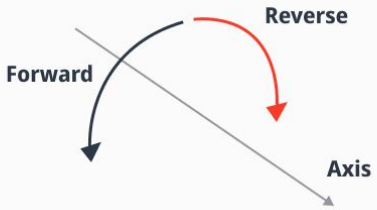
Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- rev refers to the total number of revolutions of the spiral motion.
- Rmax refers to the maximum radius of the spiral motion.
- Lmax refers to the parallel distance in the axis direction during the motion. A negative value means the parallel distance in the –axis direction.
- Vel refers to the moving velocity of the spiral motion.
- The first value of _global_velx (parallel velocity) is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- acc refers to the moving acceleration of the spiral motion.
- The first value of _global_accx (parallel acceleration) is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.

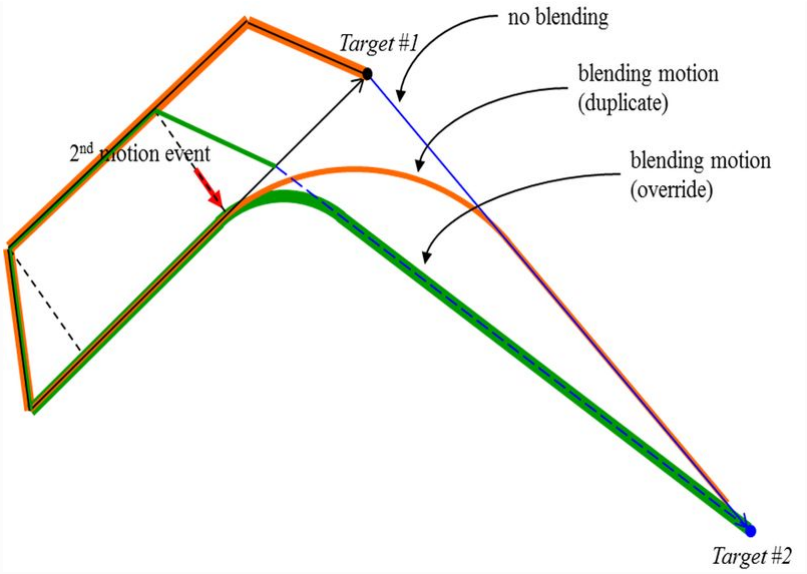
- The axis defines the axis that is perpendicular to the surface defined by the spiral motion.
- Ref refers to the reference coordinate system defined by the spiral motion.
- rad_dir refers to the radial direction of spiral



- rot_dir refers to the rotation direction of spiral along axis.



- If the radius is None, it is set to the blending radius in the blending section and 0 otherwise.
- If a new motion (following motion) is executed before the motion being executed (previous motion) is completed, the previous motion and the following motion are smoothly connected (Motion Blending). It is possible to set the option ra, which can determine whether to maintain or cancel the target of the previous motion, for the following motion. (Maintain: ra=DR_MV_RA_DUPLICATE / Cancel: ra=DR_MV_RA_OVERRIDE) For example, in the figure below, if the following motion is executed at the “2nd motion event” point of a previous motion with the target, “Target#1,” and if the option ra=DR_MV_RA_DUPLICATE is set for the following motion, the motion will follow the orange trajectory, as the motion maintains the target of the previous motion, and if option ra=DR_MV_RA_OVERRIDE is set for the following motion, the motion will follow the green trajectory, as it cancels the target of the previous motion.



- For the P series, it can only be used when the Flange Z-axis is aligned with the Base Z-axis.

Caution

An error can be generated to ensure safe motion if the rotating acceleration calculated by the spiral path is too great. In this case, reduce the vel, acc, or increase time value.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Model: M1013

J00 = posj(0,0,90,0,90,0)
P1 = [559,34.5,651.5]
P2 = [579, 65, 641.5]

D1 = [20,0,10]
D2 = [-20,0,-10]

movej(J00,vel=100,acc=100) # Move to Initial Poisition

# Max Radius with Blending
move_spiral(rev=5,rmax=20.0,lmax=-10,v=40,a=40,axis=DR_AXIS_Z,ref=DR_TOOL,rad_dir=DR_
SPIRAL_OUTWARD, rot_dir=DR_ROT_FORWARD, r=20)
move_spiral(rev=5,rmax=20.0,lmax=-10,v=40,a=40,axis=DR_AXIS_Z,ref=DR_TOOL,rad_dir=DR_
SPIRAL_INWARD, rot_dir=DR_ROT_FORWARD)

# Coordinate values with Blending
move_spiral(rev=5,pos=P2,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_dir=DR_SPIRAL_OUTWA
RD, rot_dir=DR_ROT_FORWARD, r=20)
move_spiral(rev=5,pos=P1,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_dir=DR_SPIRAL_INWAR
D, rot_dir=DR_ROT_FORWARD)

# Relative motion with Blending
move_spiral(rev=5,pos=D1,mod=DR_MV_MOD_REL,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_d
ir=DR_SPIRAL_OUTWARD, rot_dir=DR_ROT_FORWARD, r=20)
move_spiral(rev=5,pos=D2,mod=DR_MV_MOD_REL,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_d
ir=DR_SPIRAL_INWARD, rot_dir=DR_ROT_FORWARD)
```

Related commands

- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- amove_spiral()(p. 122)

2.3.9 move_periodic()

Definition

move_periodic(amp, period, atime, repeat, ref)

Features

This function performs the cyclic motion based on the sine function of each axis (parallel and rotation) of the reference coordinate (ref) input as a relative motion that begins at the current position. The attributes of the motion on each axis are determined by the amplitude and period, and the acceleration/deceleration time and the total motion time are set by the interval and repetition count.

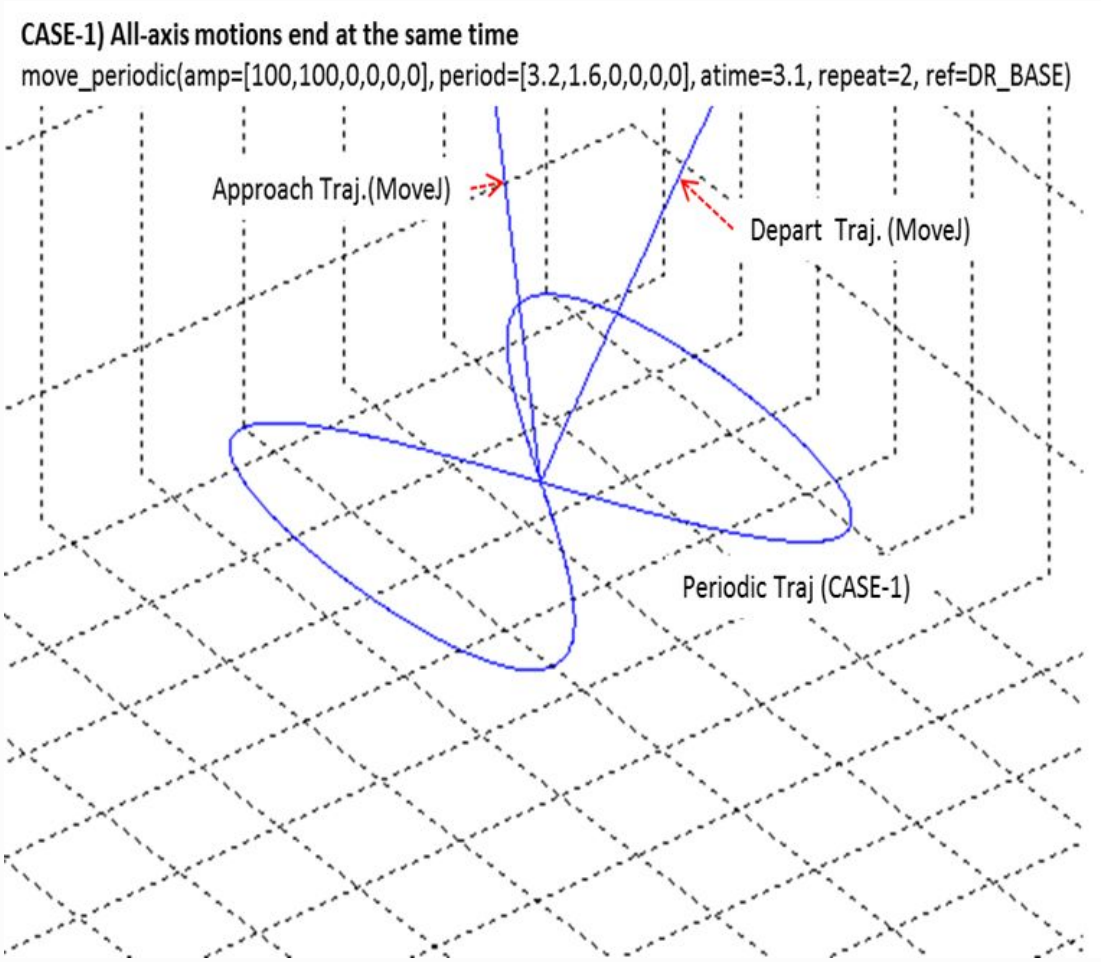
Parameters

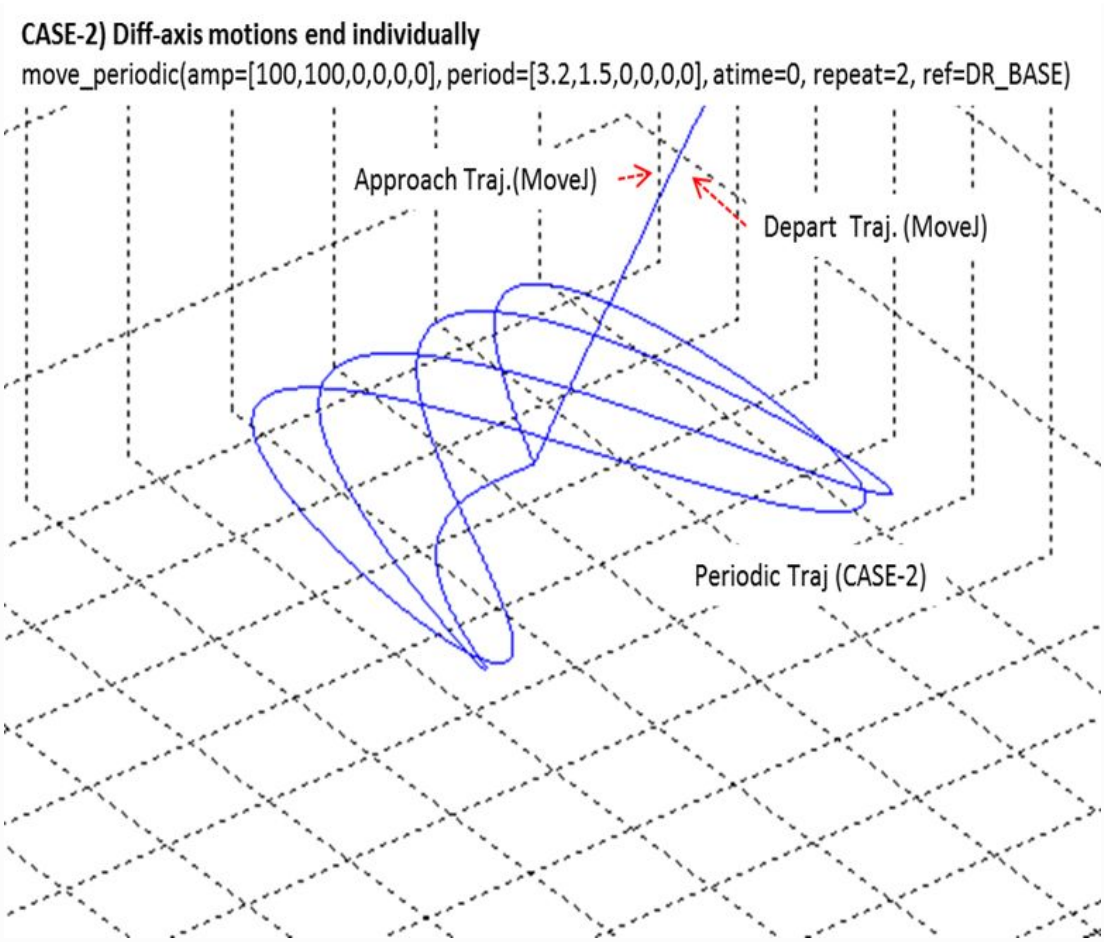
Parameter Name	Data Type	Default Value	Range	Description
amp	list (float[6])	-	0≤amp	Amplitude(motion between -amp and +amp) [mm] or [deg]
period	float or list (float[6])		0≤period	period(time for 1 cycle)[sec]
atime	float	0.0	0≤atime	Acc-, dec- time [sec]
repeat	int	1	> 0	Repetition count
ref	int	DR_TOOL	-	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate : user defined

Note

- Amp refers to the amplitude. The input is a list of 6 elements which are the amp values for the axes (x, y, z, rx, ry, and rz). The amp input on the axis that does not have a motion must be 0.
- Period refers to the time needed to complete a motion in the direction, the amplitude. The input is a list of 6 elements which are the periods for the axes (x, y, z, rx, ry, and rz).
- Atime refers to the acceleration and deceleration time at the beginning and end of the periodic motion. The largest of the inputted acceleration/deceleration times and maximum period*1/4 is applied. An error is generated when the inputted acceleration/deceleration time exceeds 1/2 of the total motion time.
- Repeat refers to the number of repetitions of the axis (reference axis) that has the largest period value and determines the total motion time. The number of repetitions for each of the remaining axes is determined automatically according to the motion time.

- If the motion terminates normally, the motions for the remaining axes can be terminated before the reference axis's motion terminates so that the end position matches the starting position. The deceleration section will deviate from the previous path if the motions of all axes are not terminated at the same time. Refer to the following image for more information.





- Ref refers to the reference coordinate system of the repeated motion.
- If a maximum velocity error is generated during a motion, adjust the amplification and period using the following formula.
  Max. velocity = Amplification(amp)*2*pi(3.14)/Period(period) (i.e., Max. velocity=62.83mm/sec if amp=10mm and period=1 sec)
- This function does not support online blending of previous and subsequent motions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0)

#1
move_periodic(amp =[10,0,0,0,30,0], period=1.0, atime=0.2, repeat=5, ref=DR_TOOL)
    # Repeats the x-axis (10mm amp and 1 sec. period) motion and rotating y-axis
(30deg amp and 1 sec. period) motion in the tool coordinate system
    # totally, repeat the motion 5 times.

#2
move_periodic(amp =[10,0,20,0,0.5,0], period=[1,0,1.5,0,0,0], atime=0.5, repeat=3,
ref=DR_BASE)
    # Repeats the x-axis (10mm amp and 1 sec. period) motion and z-axis (20mm amp and
1.5 sec. period) motion in the base coordinate system
    # 3 times. The rotating y-axis motion is not performed since its period is "0".
    # The total motion time is about 5.5 sec. (1.5 sec. * 3 times + 1 sec. for
acceleration/deceleration) since the period of the x-axis motion is greater.
    # The x-axis motion is repeated 4.5 times.
```

Related commands

- set_ref_coord()(p. 54)
- amove_periodic()(p. 126)

2.3.10 move_home()

Definition

move_home(target)

Features

Homing is performed by moving to the joint motion to the mechanical or user defined home position. According to the input parameter [target], it moves to the mechanical home defined in the system or the home set by the user.

Caution

- When using the Move command after homing is complete, it is recommended to use the wait command after homing.

Parameters

Parameter Name	Data Type	Default Value	Range	Description
target	int	-		Tartget of home position • DR_HOME_TARGET_MECHANIC : Mechanical home, joint angle (0,0,0,0,0,0) • DR_HOME_TARGET_USER : user home.

Note

- Homing motion is divided into two steps and performed sequentially.
  a. Move to the homing position at the speed specified in the system.
  b. Finding the home position precisely
- Safety should be ensured so that there is no danger of collision in the vicinity of homing operation.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
move_home(DR_HOME_TARGET_USER)      # Go to the user home

P0 = posj(0,0,90,0,90,0)
movej(P0)
```

2.4 Asynchronous Motion

2.4.1 amovej()

Definition

amovej(pos, vel, acc, time, mod, ra, ref, velx)

Features

The asynchronous movej motion operates in the same way as movej except that it does not have the radius parameter for blending. The command is the asynchronous motion command, and the next command is executed at the same time the motion begins.

- Case1: Data type of Input parameter(pos) is posj ==> movej
- Case2: Data type of Input parameter(pos) is posx ==> movejx (v2.x DRL Command)

Note

- movej(pos): The next command is executed after the robot starts from the current position and reaches (stops at) pos.
- amovej(pos): The next command is executed regardless of whether the robot starts from the current position and reaches (stops at) pos.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posj list (float[6])	-	posj or joint angle list
vel (v)	float list (float[6])	None	velocity (same to all axes) or velocity (to an axis)
acc (a)	float list (float[6])	None	acceleration (same to all axes) or acceleration (acceleration to an axis)
time (t)	float	None	Reach time [sec]
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • User coordinate: User defined
velx	float	None	TCP speed limiting

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)

- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- Refer to the description of the movej() motion for the path of blending according to option ra and vel/acc.
- _global_velx is applied if velx is None and clamp is DR_ON in set_velx. (The initial value of _global_velx is 0.0 and can be set by set_velx.)

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. The robot moves to q1 and stops the motion 3 seconds after it begins the
motion at q0 and then moves to q99
q0 = posj(0, 0, 90, 0, 90, 0)
q1 = posj(0, 0, 0, 0, 90, 0)
q99 = posj(0, 0, 0, 0, 0, 0)
```

```
# Moves to q0 and performs the next command immediately after
amovej(q0, vel=10, acc=20)
wait(3) # Temporarily suspends the program execution for 3 seconds (while the motion
continues).

# Maintains the q0 motion (DUPLICATE blending if the ra argument is omitted) and
iterates to q1.
# Performs the next command immediately after the blending motion.
amovej(q1, vel=10, acc=20)
mwait(0) # Temporarily suspends the program execution until the motion is terminated.

# Joint motion to q99
movej (q99, vel=10, acc=20)
```

Related commands

- posj()(p. 27)
- set_velj()(p. 47)
- set_accj()(p. 48)
- mwait()(p. 129)
- movej()(p. 58)

2.4.2 amovel()

Definition

amovel(pos, vel, acc, time, ref, mod, ra, app_type)

Features

The asynchronous movel motion operates in the same way as movel except that it does not have the radius parameter for blending. The command is the asynchronous motion command, and the next command is executed without waiting for the motion to terminate.

Note

- movel(pos): The next command is executed after the robot starts from the current position and reaches (stops at) pos.
- amovel(pos): The next command is executed regardless of whether the robot starts from the current position and reaches (stops at) pos.

Parameters

Parameter Name	Data Type	Default Value	Description

pos	posx list (float[6])	-	posx or position list
vel (v)	float list (float[2])	None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None	acceleration or acceleration1, acceleration2
time (t)	float	None	Reach time [sec] • If the time is specified, values are processed based on time, ignoring vel and acc.
ref	int	None	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate: User defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
app_type	int	DR_MV_APP_NONE	Reactive motion mode • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related

Note

- Abbreviated parameter names supported (v:vel, a:acc, t:time).
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)

- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- Refer to the description of the movej() motion for the path of the blending according to option ra and vel/acc.
- If 'app_type' is 'DR_MV_APP_WELD', parameter 'vel' is internally replaced by the speed setting entered in app_weld_set_weld_cond(), not the input value of 'vel'.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. D-Out 2 seconds after the motion starts with x1
```

```
j0 = posj(-148,-33,-54,180,92,32)
movej(j0, v=30, a=30)
x1 = posx(784, 543, 570, 0, 180, 0)
amovel (x1, vel=100, acc=200)   # Performs the next motion immediately after
beginning a motion with x1.
wait(2)                         # Temporarily suspends the program execution for 2
seconds (while the motion continues).
set_digital_output(1, 1)        # D-Out (no. 1 channel) ON
mwait(0)                        # Temporarily suspends the program execution until
the motion is terminated.
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- mwait()(p. 129)
- movel()(p. 63)

2.4.3 amovejx()

Definition

amovejx(pos, vel, acc, time, ref, mod, ra, sol, velx)

Features

The asynchronous movejx motion operates in the same way as movejx except that it does not have the radius parameter for blending. The command is the asynchronous motion command, and the next command is executed without waiting for the motion to terminate.

Note

- movejx(pos): The next command is executed after the robot starts from the current position and reaches (stops at) pos.
- amovejx(pos): The next command is executed regardless of whether the robot starts from the current position and reaches (stops at) pos.

Parameters

Parameter Name	Data Type	Default Value	Description

pos	posx	-	posx or position list
	list (float[6])		
vel (v)	float	None	velocity (same to all axes) or velocity (to an axis)
	list (float[6])	None	
acc (a)	float	None	acceleration (same to all axes) or acceleration (acceleration to an axis)
	list (float[6])	None	
time (t)	float	None	Reach time [sec]
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD : world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
sol	int	0	Solution space • Refer to Table. Robot Configuration (movejx)
velx	float	None	TCP speed limiting

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.

- If the time is None, it is set to 0.
- _g_coord is applied if the ref is None. The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.
- Refer to the description of the movej() motion for the path of the blending according to option ra and vel/acc.
- _global_velx is applied if velx is None and clamp is DR_ON in set_velx. (The initial value of _global_velx is 0.0 and can be set by set_velx.)

Caution

If relative motion is entered (mod=DR_MV_MOD_REL), the motion in progress cannot execute blending. Therefore it is recommended to execute blending with movej() or movel().

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. D-Out 2 seconds after the joint motion starts with x1
p0 = posj(-148,-33,-54,180,92,32)
movej(p0, v=30, a=30)

x1 = posx(784, 443, 770, 0, 180, 0)
```

```
# Performs the next motion immediately after beginning a joint motion with x1.
amovejx(x1, vel=100, acc=200, sol=1)

wait(2) # Temporarily suspends the program execution for 2 seconds (while the motion
continues).
set_digital_output(1, 1) # D-Out (no. 1 channel) ON
mwait(0) # Temporarily suspends the program execution until the motion is terminated.
```

Related commands

- posx()(p. 28)
- set_velj()(p. 47)
- set_accj()(p. 48)
- get_current_posx()(p. 166)
- mwait()(p. 129)
- movejx()(p. 68)

2.4.4 amovec()

Definition

amovec(pos, pos2, vel, acc, time, ref, mod, angle, ra, ori, app_type)

Features

The asynchronous movec motion operates in the same way as movec except that it does not have the radius parameter for blending. The command is the asynchronous motion command, and the next command is executed without waiting for the motion to terminate.

Note

- movec(pos1. pos2): The next command is executed after the robot starts from the current position and reaches (stops at) pos2.
- amovec(pos1. pos2): The next command is executed regardless of whether the robot starts from the current position and reaches (stops at) pos2.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx list (float[6])	-	posx or position list

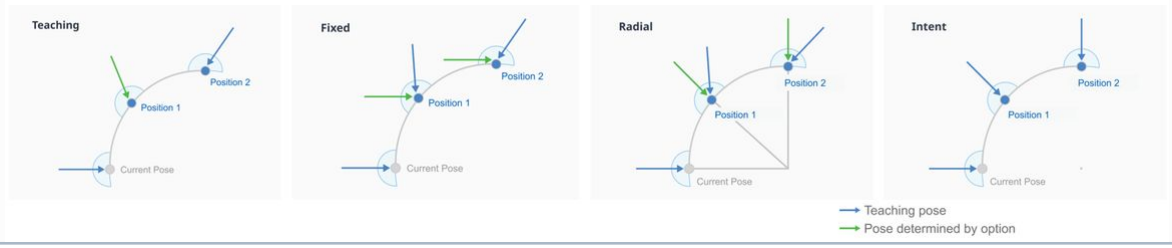
pos2	posx list (float[6]	-	posx or position list
vel (v)	float list (float[2])	None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None	acceleration or acceleration1, acceleration2
time (t)	float	None	Reach time [sec]
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
angle (an)	float list (float[2])	None	angle or angle1, angle2
ra	int	DR_MV_RA_DUPLICATE	Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override
ori	int	DR_MV_ORI_TEACH	Orientation mode • DR_MV_ORI_TEACH: orientation changes continuously from the initial to the final taught value • DR_MV_ORI_FIXED: orientation holds with the initial orientation • DR_MV_ORI_RADIAL: orientation changes radially from the initial. • DR_MV_ORI_INTENT: orientation changes according to the value set by the user.

app_type	int	DR_MV_APP_NONE	Application mode • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, angle:an)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If the mod is DR_MV_MOD_REL, pos1 and pos2 are defined in the relative coordinate system of the previous pos. (pos1 is the relative coordinate from the starting point while pos2 is the relative coordinate from pos1.)
- If the angle is None, it is set to 0.
- If only one angle is entered, the angle applied will be the total rotation angle of the circular path.
- If two angle values are inputted, angle1 refers to the total rotating angle moving at a constant velocity on the circular path while angle2 refers to the rotating angle in the rotating section for acceleration and deceleration. Here, the robot moves on the circular path at a total movement angle of angle1 + 2xangle2.
- Refer to the description of the movej() motion for the path of the blending according to option ra and vel/acc.
- If ‘app_type’ is ‘DR_MV_APP_WELD’, parameter ‘vel’ is internally replaced by the speed setting entered in app_weld_set_weld_cond(), not the input value of ‘vel’.
- ori’(orientation mode) is defined as below.
    a. DR_MV_ORI_TEACH(orientation based on teaching) : It moves while changing the current pose to the teaching pose of Pose 2, proportionate to the movement distance.The orientation of the taught pose, ‘pose 1’ is ignored.
    b. DR_MV_ORI_FIXED(fixed orientation) : Move along the path while maintaining the initial orientation up to the taught pose, ‘pose2’.
    c. DR_MV_ORI_RADIAL(orientation constrained radially) : Move along the path while maintaining radial orientation at the initial pose to the ‘pose 2

d. DR_MV_ORI_INTENT (orientation set by user): Moves from the current posture to pose 2 through the taught posture of pose 1. At this time, movement from the current posture to pose 1 and movement from pose 1 to pose 2 are moved by the shortest rotation distance.



Caution

For the **DR_MV_ORI_INTENT** option, the rotational speed is dependent on translational speed. If the translation speed is high, the rotational speed will also increase.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. D-Out 3 seconds after the arc motion through x1 and x2 begins
```

```
p0 = posj(-148,-33,-54,180,92,32)
movej(p0, v=30, a=30)
x1 = posx(784, 443, 770, 0, 180, 0)
amovejx (x1, vel=100, acc=200, sol=2) # Performs the next motion immediately after
beginning a joint motion with x1.
wait(2)              # Temporarily suspends the program execution for 2 seconds
(while the motion continues).
set_digital_output(1, 1)         # D-Out (no. 1 channel) ON
mwait(0)
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- mwait()(p. 129)
- movec()(p. 72)

2.4.5 amovesj()

Definition

amovesj(pos_list, vel, acc, time, mod)

Features

The asynchronous movesj motion operates in the same way as movesj() except for the asynchronous processing.

Generating a new command for the motion before the amovesj() motion results in an error for safety reasons. Therefore, the termination of the amovesj() motion must be confirmed using mwait() or check_motion() between amovesj() and the following motion command.

Note

- movesj(pos_list): The next command is executed after the robot starts from the current position and reaches (stops at) the end point of pos_list.
- amovesj(pos_list): The next command is executed regardless of whether the robot starts from the current position and reaches (stops at) the end point of pos_list.

Parameters

Parameter Name	Data Type	Default Value	Description

pos_list	list (posj)	-	posj list
vel (v)	float list (float[6])	None	velocity (same to all axes) or velocity (to an axis)
acc (a)	float list (float[6])	None	acceleration (same to all axes) or acceleration (acceleration to an axis)
time (t)	float	None	Reach time [sec]
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- If the mod is DR_MV_MOD_REL, each pos in the pos_list is defined in the relative coordinate of the previous pos. (If pos_list=[q1, q2, ...,q(n-1), q(n)], q1 is the relative angle of the starting point while q(n) is the relative coordinate of q(n-1).)
- This function does not support online blending of previous and subsequent motions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. D-Out 3 seconds after the spline motion through q1 - q5 begins
q0 = posj(0,0,0,0,0,0)
movej(q0, vel=30, acc=60)   # Moves in joint motion to the initial position (q0).
q1 = posj(10, -10, 20, -30, 10, 20)      # Defines the posj variable (joint angle)
q1.
q2 = posj(25, 0, 10, -50, 20, 40)
q3 = posj(50, 50, 50, 50, 50, 50)
q4 = posj(30, 10, 30, -20, 10, 60)
q5 = posj(20, 20, 40, 20, 0, 90)

qlist = [q1, q2, q3, q4, q5]
    # Defines the list (qlist) which is a set of waypoints q1-q5.

amovesj(qlist, vel=30, acc=100)
    # Moves the spline curve that connects the waypoints defined in the qlist.
    # with a maximum velocity of 30(mm/sec) and maximum acceleration of 100(mm/sec2).
    # Executes the next command.
wait(3)                                  # Temporarily suspends the program execution
for 3 seconds (while the motion continues).
set_digital_output(1, 1)                 # D-Out (no. 1 channel) ON
mwait(0)                                 # Temporarily suspends the program execution
until the motion is terminated.
```

Related commands

- posj()(p. 27)
- set_velj()(p. 47)
- set_accj()(p. 48)

- mwait()(p. 129)
- movesj()(p. 78)

2.4.6 amovesx()

Definition

amovesx(pos_list, vel, acc, time, ref, mod, vel_opt)

Features

The asynchronous movesx motion operates in the same way as movesx() except for the asynchronous processing.

Generating a new command for the motion before the amovesx() motion results in an error for safety reasons. Therefore, the termination of the amovesx() motion must be confirmed using mwait() or check_motion() between amovesx() and the following motion command.

Note

- movesx(pos_list): The next command is executed after the robot starts from the current position and reaches (stops at) the end point of pos_list.
- amovesx(pos_list): The next command is executed regardless of whether the robot starts from the current position and reaches (stops at) the end point of pos_list.

Parameters

Parameter Name	Data Type	Default Value	Description
pos_list	list (posx)	-	posx list
vel (v)	float list (float[2])	None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None	acceleration or acceleration1, acceleration2
time (t)	float	None	Reach time [sec]

ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: User defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
vel_opt	int	DR_MVS_VEL_NONE	Velocity option • DR_MVS_VEL_NONE: None • DR_MVS_VEL_CONST: Constant velocity

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If the mod is DR_MV_MOD_REL, each pos in the pos_list is defined in the relative coordinate of the previous pos. (If pos_list=[p1, p2, ...,p(n-1), p(n)], p1 is the relative angle of the starting point while p(n) is the relative coordinate of p(n-1).)
- This function does not support online blending of previous and subsequent motions.

Caution

The constant velocity motion according to the distance and velocity between the waypoints cannot be used if the "vel_opt= DR_MVS_VEL_CONST" option (constant velocity option) is selected, and the motion is automatically switched to the variable velocity motion (vel_opt= DR_MVS_VEL_NONE) in that case.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. D-Out 3 seconds after the spline motion through x1 - x6 begins
P0 = posj(0,0,90,0,90,0)
movej(P0)
x0 = posx(600, 43, 500, 0, 180, 0)            # Defines the posx variable (space
coordinate/pose) x0.
movel(x0, vel=100, acc=200)       # Linear movement to the initial position x0
x1 = posx(600, 600, 600, 0, 175, 0)           # Defines the posx variable (space
coordinate/pose) x1.
x2 = posx(600, 750, 600, 0, 175, 0)
x3 = posx(150, 600, 450, 0, 175, 0)
x4 = posx(-300, 300, 300, 0, 175, 0)
x5 = posx(-200, 700, 500, 0, 175, 0)
x6 = posx(600, 600, 400, 0, 175, 0)

xlist = [x1, x2, x3, x5, x6]             # Defines the list (xlist) which is a set of
x1-x6 as the waypoints.

amovesx(xlist, vel=[100, 30], acc=[200, 60], vel_opt=DR_MVS_VEL_NONE)
    # Moves the spline curve that connects the waypoints defined in the xlist
```

```
    # with a maximum velocity of 100, 30(mm/sec, deg/sec) and maximum acceleration of
200(mm/sec2) and
    # 60(deg/sec2). The next command is executed immediately after the motion starts.
wait(3)              # Temporarily suspends the program execution for 3 seconds
(while the motion continues).
set_digital_output(1, 1)            # D-Out (no. 1 channel) ON
mwait(0)                            # Temporarily suspends the program execution
until the motion is terminated.
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- mwait()(p. 129)
- movesx()(p. 81)

2.4.7 amoveb()

Definition

amoveb(pos_list, vel, acc, time, ref, mod, app_type)

Features

The asynchronous moveb motion operates in the same way as moveb() except for the asynchronous processing and executes the next line after the command is executed.

Generating a new command for the motion before the amoveb() motion results in an error for safety reasons. Therefore, the termination of the amoveb() motion must be confirmed using mwait() or check_motion() between amoveb() and the following motion command.

Note

- moveb(seg_list): The next command is executed after the robot starts from the current position and reaches (stops at) the end point of seg_list.
- amoveb(seg_list): The next command is executed regardless of whether the robot starts from the current position and reaches (stops at) the end point of seg_list.

Parameters

Parameter Name	Data Type	Default Value	Description

pos_list	list (posb)	-	posb list
vel (v)	float list (float[2])	None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None	acceleration or acceleration1, acceleration2
time (t)	float	None	Reach time [sec] • If the time is specified, values are processed based on time, ignoring vel and acc.
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: User defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
app_type	int	DR_MV_APP_NONE	Application mode • DR_MV_APP_NONE: No application related • DR_MV_APP_WELD: Welding application related

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- Up to 50 arguments can be entered in posb_list.
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accj is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If an argument is inputted to vel (e.g., vel=30), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=60), the input argument corresponds to the linear acceleration of the motion while the angular acceleration is determined proportionally to the linear acceleration.

- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- _g_coord is applied if the ref is None. (The initial value of _g_coord is DR_BASE, and it can be set by the set_ref_coord command.)
- If the mod is DR_MV_MOD_REL, each pos in the pos_list is defined in the relative coordinate of the previous pos.
- This function does not support online blending of previous and subsequent motions.
- If 'app_type' is 'DR_MV_APP_WELD', parameter 'vel' is internally replaced by the speed setting entered in app_weld_set_weld_cond(), not the input value of 'vel'.

Caution

- A user input error is generated if the blending radius in posb is 0.
- A user input error is generated due to the duplicated input of Line if contiguous Line-Line segments have the same direction.
- A user input error is generated to prevent a sudden acceleration if the blending condition causes a rapid change in direction.
- This function does not support online blending of previous and subsequent motions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. D-Out 3 seconds after the motion through the path of seg11 - seg16 begins
# Init Pose @ Jx1
Jx1 = posj(45,0,90,0,90,45)               # initial joint position
X0 = posx(370, 420, 650, 0, 180, 0)       # initial task position

# CASE#1) ABSOLUTE
# Absolute Goal Poses
X1 = posx(370, 670, 650, 0, 180, 0)
X1a = posx(370, 670, 400, 0, 180, 0)
X1a2= posx(370, 545, 400, 0, 180, 0)
X1b = posx(370, 595, 400, 0, 180, 0)
X1b2= posx(370, 670, 400, 0, 180, 0)
X1c = posx(370, 420, 150, 0, 180, 0)
X1c2= posx(370, 545, 150, 0, 180, 0)
X1d = posx(370, 670, 275, 0, 180, 0)
X1d2= posx(370, 795, 150, 0, 180, 0)

seg11 = posb(DR_LINE, X1, radius=20)
seg12 = posb(DR_CIRCLE, X1a, X1a2, radius=20)
seg14 = posb(DR_LINE, X1b2, radius=20)
seg15 = posb(DR_CIRCLE, X1c, X1c2, radius=20)
seg16 = posb(DR_CIRCLE, X1d, X1d2, radius=20)
b_list1 = [seg11, seg12, seg14, seg15, seg16]
    # The blending radius of the last waypoint (seg16) is ignored.
movej(Jx1, vel=30, acc=60, mod=DR_MV_MOD_ABS)
    # Joint motion to the initial angle (Jx1)
movel(X0, vel=150, acc=250, ref=DR_BASE, mod=DR_MV_MOD_ABS)
    # Line motion to the initial position (X0)
amoveb(b_list1, vel=150, acc=250, ref=DR_BASE, mod=DR_MV_MOD_ABS)
    # Moves the robot from the current position through a trajectory consisting of
seg11(LINE), seg12(CIRCLE), seg14(LINE),
    # seg15(CIRCLE), and seg16(CIRCLE) with a constant velocity of 150(mm/sec) with
the exception of accelerating and decelerating sections.
    # (The final point is X1d2.)
    # Blending to the next segment begins when the distance of 20mm from the end
point (X1, X1a2, X1b2, X1c2, and X1d2)
    # of each segment is reached.
wait(3)                                  # Temporarily suspends the program execution for
3 seconds (while the motion continues).
set_digital_output(1, 1)               # D-Out (no. 1 channel) ON
mwait(0)                               # Temporarily suspends the program execution
until the motion is terminated.
```

Related commands

- posb()(p. 33)
- set_velx()(p. 50)
- set_accx()(p. 52)

- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- mwait()(p. 129)
- moveb()(p. 84)

2.4.8 amove_spiral()

Features

The asynchronous move_spiral motion operates in the same way as move_spiral except that it does not have the radius parameter for blending. The command is the asynchronous motion command, and the next command is executed without waiting for the motion to terminate.

Note

- move_spiral: The next command is executed after the robot starts from the current position and reaches (stops at) the end point of the spiral trajectory.
- amove_spiral: The next command is executed after the robot starts from the current position and regardless of whether it reaches (stops at) the end point of the spiral trajectory.

Parameters

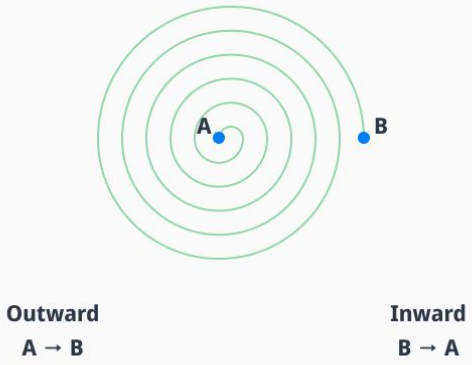
Parameter Name	Data Type	Default Value	Range	Description
rev	float	10	rev > 0	Total number of revolutions
rmax	float	None	rmax > 0	Final spiral radius [mm]
lmax	float	0		Distance moved in the axis direction [mm]
vel (v)	float	None		velocity
acc (a)	float	None		acceleration
time (t)	float	None	time ≥ 0	Total execution time <sec>
axis	int	DR_AXIS_Z	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis

Parameter Name	Data Type	Default Value	Range	Description
ref	int	DR_TOOL	-	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate : user defined
pos	posx list(float[3]) list(float[6])	None		posx or position list (X,Y,Z)
mod	int	DR_MV_MOD_ABS		Movement basis • DR_MV_MOD_ABS : Absolute • DR_MV_MOD_REL : Relative
rad_dir	int	DR_SPIRAL_OUTWARD		Radial direction • DR_SPIRAL_OUTWARD: toward outward • DR_SPIRAL_INWARD: toward inward
rot_dir	int	DR_ROT_FORWARD		Rotation direction • DR_ROT_FORWARD : Forward rotation (+) • DR_ROT_REVERSE : Reverse rotation (-)
ra	int	DR_MV_RA_DUPLICATE		Reactive motion mode • DR_MV_RA_DUPLICATE: duplicate • DR_MV_RA_OVERRIDE: override

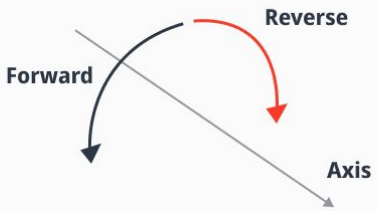
Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- rev refers to the total number of revolutions of the spiral motion.
- Rmax refers to the maximum radius of the spiral motion.
- Lmax refers to the parallel distance in the axis direction during the motion. A negative value means the parallel distance in the –axis direction.
- Vel refers to the moving velocity of the spiral motion.

- The first value of _global_velx (parallel velocity) is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- Acc refers to the moving acceleration of the spiral motion.
- The first value of _global_accx (parallel acceleration) is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)
- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the time is None, it is set to 0.
- The axis defines the axis that is perpendicular to the surface defined by the spiral motion.
- Ref refers to the reference coordinate system defined by the spiral motion.
- rad_dir refers to the radial direction of spiral



- rot_dir refers to the rotation direction of spiral along axis.



- Please refer to the description of the move_spiral() motion for the path of the blending according to option ra and vel/acc.
- For the P series, it can only be used when the Flange Z-axis is aligned with the Base Z-axis.

Caution

An error can be generated to ensure safe motion if the rotating acceleration calculated by the spiral path is too great. In this case, reduce the vel, acc, or increase time value.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Model: M1013

J00 = posj(0,0,90,0,90,0)
P1 = [559,34.5,651.5]
P2 = [579, 65, 641.5]

D1 = [20,0,10]
D2 = [-20,0,-10]

movej(J00,vel=100,acc=100) # Move to Initial Poisition

# Max Radius with Blending
amove_spiral(rev=5,rmax=20.0,lmax=-10,v=40,a=40,axis=DR_AXIS_Z,ref=DR_TOOL,rad_dir=DR
_SPIRAL_OUTWARD, rot_dir=DR_ROT_FORWARD)
wait(5.5)
amove_spiral(rev=5,rmax=20.0,lmax=-10,v=40,a=40,axis=DR_AXIS_Z,ref=DR_TOOL,rad_dir=DR
_SPIRAL_INWARD, rot_dir=DR_ROT_FORWARD)
mwait(0)

# Coordinate values with Blending
```

```
amove_spiral(rev=5,pos=P2,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_dir=DR_SPIRAL_OUTW
ARD, rot_dir=DR_ROT_FORWARD)
wait(10.2)
amove_spiral(rev=5,pos=P1,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_dir=DR_SPIRAL_INWA
RD, rot_dir=DR_ROT_FORWARD)
mwait(0)

# Relative motion with Blending
amove_spiral(rev=5,pos=D1,mod=DR_MV_MOD_REL,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_
dir=DR_SPIRAL_OUTWARD, rot_dir=DR_ROT_FORWARD)
wait(5.5)
amove_spiral(rev=5,pos=D2,mod=DR_MV_MOD_REL,v=40,a=40,axis=DR_AXIS_Z,ref=DR_BASE,rad_
dir=DR_SPIRAL_INWARD, rot_dir=DR_ROT_FORWARD)
mwait(0)
```

Related commands

- set_velx()(p. 50)
- set_accx()(p. 52)
- set_tcp()(p. 230)
- set_ref_coord()(p. 54)
- mwait()(p. 129)
- move_spiral()(p. 89)

2.4.9 amove_periodic()

Definition

amove_periodic(amp, period, atime, repeat, ref)

Features

The asynchronous move_periodic motion operates in the same way as move_periodic() except for the asynchronous processing and executes the next line after the command is executed.

Generating a new command for the motion before the amove_periodic() motion results in an error for safety reasons. Therefore, the termination of the amove_periodic() motion must be confirmed using mwait() or check_motion() between amove_periodic() and the following motion command.

This command performs a cyclic motion based on the sine function of each axis (parallel and rotation) of the reference coordinate (ref) input as a relative motion that begins at the current position. The attributes of the motion on each axis are determined by amp (amplitude) and period, and the acceleration/deceleration time and the total motion time are set by the interval and repetition count.

Note

- move_ periodic: Starting from the current position, reaching the end of the periodic trajectory, stopping, and then executing the following command

- amove_ periodic: Executes the next command immediately regardless of whether the end of the periodic trajectory is reached from the current position

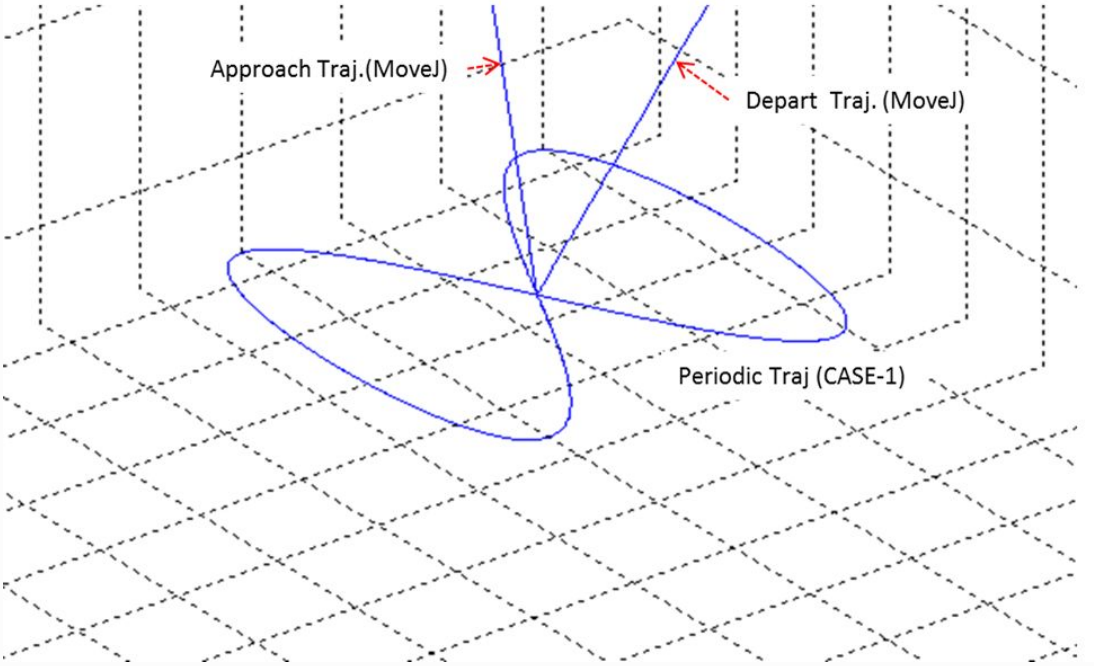
Parameters

Parameter Name	Data Type	Default Value	Range	Description
amp	list (float[6])	-	0≤amp	Amplitude (motion between -amp and +amp) [mm] or [deg]
period	float or list (float[6])		0≤period	Period (time for 1 cycle) [sec]
atime	float	0.0	0≤atime	Acc-, dec- time [sec]
repeat	int	1	> 0	Repetition count
ref	int	DR_TOOL	-	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate : user defined

Note

- Amp refers to the amplitude. The input is a list of 6 elements which are the amp values for the axes (x, y, z, rx, ry, and rz). The amp input on the axis that does not have a motion must be 0.
- Period refers to the time needed to complete a motion in the direction, the amplitude. The input is a list of 6 elements which are the periods for the axes (x, y, z, rx, ry, and rz).
- Atime refers to the acceleration and deceleration time at the beginning and end of the periodic motion. The largest of the inputted acceleration/deceleration times and maximum period*1/4 is applied. An error is generated when the inputted acceleration/deceleration time exceeds 1/2 of the total motion time.
- Repeat refers to the number of repetitions of the axis (reference axis) that has the largest period value and determines the total motion time. The number of repetitions for each of the remaining axes is determined automatically according to the motion time.
- If the motion terminates normally, the motions for the remaining axes can be terminated before the reference axis's motion terminates so that the end position matches the starting position. The deceleration section will deviate from the previous path if the motions of all axes are not terminated at the same time. Refer to the following image for more information.

CASE-1) All-axis motions end at the same time
move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.6,0,0,0,0], atime=3.1, repeat=2, ref=DR_BASE)



- Ref refers to the reference coordinate system of the repeated motion.
- If a maximum velocity error is generated during a motion, adjust the amplification and period using the following formula.
  Max. velocity = Amplification(amp)*2*pi(3.14)/Period(period) (i.e., Max. velocity=62.83mm/sec if amp=10mm and period=1 sec)
- This function does not support online blending of previous and subsequent motions.

Return

Value	Success
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
amove_periodic(amp =[10,0,0,0,0.5,0], period=1, atime=0.5, repeat=5, ref=DR_TOOL)
wait(1)
set_digital_output(1, 1)
mwait(0)
# Repeats the x-axis (10mm amp and 1 sec. period) motion and y rotating axis (0.5deg
amp and 1 sec. period) motion in the tool coordinate system
# 5 times.
# SET(1) the Digital_Output channel no. 1, 1 second after the periodic motion begins.
```

Related commands

- set_ref_coord()(p. 54)
- move_periodic()(p. 93)

2.5 Additional Functions

2.5.1 mwait()

Definition

mwait(time=0)

Features

This function sets the waiting time between the previous motion command and the motion command in the next line. The waiting time differs according to the time[sec] input.

Parameters

Parameter Name	Data Type	Default Value	Description
time	float	0	Waiting time after the motion ends [sec]

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Example 1. The robot moves to q1 and stops the motion 3 seconds after it begins the
motion at q0 and then moves to q99
q0 = posj(0, 0, 90, 0, 90, 0)
amovej (q0, vel=10, acc=20)       # Moves to q0 and performs the next command
immediately after
wait(3)                           # Temporarily suspends the program execution for 3
seconds (while the motion continues).
q1 = posj(0, 0, 0, 0, 90, 0)
amovej (q1, vel=10, acc=20)
    # Maintains the q0 motion (DUPLICATE blending if the ra argument is omitted) and
iterates to q1.
```

```
    # Performs the next command immediately after the blending motion.
mwait(0)                          # Temporarily suspends the program execution until
the motion is terminated.
q99 = posj(0, 0, 0, 0, 0, 0)
movej (q99, vel=10, acc=20)       # Joint motion to q99.
```

Related commands

- wait()(p. 359)
- amovesj()(p. 112)
- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.5.2 begin_blend()

Definition

begin_blend(radius=0)

Features

This function begins the blending section. The sync motion commands (movej, movel, movec, movejx) with blending section radius execute blending using the radius set as the default argument. There is no actual blending effect if the radius is 0. Moreover, if a blending radius that is different from the set radius is needed, the blending radius can be changed as an exception by specifying the blending radius to the motion argument.

Parameters

Parameter Name	Data Type	Default Value	Description
radius	float	0	Radius for blending

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
#1
begin_blend(radius=30)
    # The motion commands with the following radius option sets
    # the blending section to 30mm.
Q1 = posj(0,0,90,0,90,0)
Q2 = posj(0,0,0,0,90,0)
movej(Q1, vel=10, acc=20)
    # Moves to the Q1 joint angle and is set to execute the next motion
    # when the global distance from the Q1 space position is 30mm.
movej(Q2, time=5)
    # Moves to the Q2 joint angle after the blend while maintaining the last motion
(motion iteration).
    # It is set to execute the next motion
    # when the global distance from the Q2 space position is 30mm.
movej(Q1, v=30, a=60, r=200)
    # Moves to the Q1 joint angle after the blending while maintaining the last
motion (motion iteration).
    # It is set to execute the next motion
    # when the distance from the Q1 space position is 200mm (the global setting is
not applied).
movej(Q2, v=30, a=60, ra= DR_MV_RA_OVERRIDE)
    # Immediately terminates the last motion and blends it to move to the Q2 joint
angle.

end_blend()      # Ends the batch setting of the blending sections.
```

Related commands

- end_blend()(p. 133)
- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)

2.5.3 end_blend()

Features

This function ends the blending section. It means that the validity of the blending section that began with begin_blend() ends.

Return

Value	Description
0	Success

Example

```
#1
begin_blend(radius=30)
    # The motion commands that have the following radius option set the blending
section to 30mm.
Q1 = posj(0,0,90,0,90,0)
Q2 = posj(0,0,0,0,90,0)
movej(Q1, vel=10, acc=20)
    # Moves to the Q1 joint angle and is set to execute the next motion
    # when the global distance from the Q1 space position is 30mm.
movej(Q2, time=5)
    # Immediately terminates the last motion and blends it to move to the Q2 joint
angle (motion iteration).
    # It is set to execute the next motion
    # when the global distance from the Q2 space position is 30mm.
movej(Q1, v=30, a=60, r=200)
    # Immediately terminates the last motion and blends it to move to the Q1 joint
angle (motion iteration).
    # It is set to execute the next motion
    # when the distance from the Q1 space position is 200mm (the global setting is
not applied).
movej(Q2, v=30, a=60, ra= DR_MV_RA_OVERRIDE)
    # Immediately terminates the last motion and blends it to move to the Q2 joint
angle.
end_blend()          # Ends the batch setting of the blending sections.
```

Related commands

- begin_blend()(p. 131)
- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)

- movec()(p. 72)

2.5.4 check_motion()

Features

This function checks the status of the currently active motion.

Return

Value	Description
0	DR_STATE_IDLE (no motion in action)
1	DR_STATE_INIT (motion being calculated)
2	DR_STATE_BUSY (motion in operation)

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#1. The next motion (q99) is executed when an asynchronous motion to q0 begins
decelerating
q0 = posj(0, 0, 90, 0, 90, 0)
q99 = posj(0, 0, 0, 0, 0, 0)
amovej (q0, vel=10, acc=20)          # Executes the next command immediately after the
motion to q0.
while True:
if check_motion()==0:                # A motion is completed.
   amovej (q99, vel=10, acc=20)      # Joint motion to q99.
   break
if check_motion()==2:                # In motion
   pass
mwait(0)                             # Temporarily suspends the program execution
until the motion is terminated.
```

Related commands

- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)
- movesj()(p. 78)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- amovej()(p. 99)
- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.5.5 stop()

Definition

stop(st_mode)

Features

This function stops the currently active motion. stop time is determined by the ‘st_mode’ and robot position does not deviate from the in-progress path.

This command is only used to stop the robot from operating and will not cause stop the program. To stop a program from running, use additionally exit() function. Values DR_QSTOP_STO and DR_QSTOP respond to Stop Category 1 (torque off after maximum deceleration) and 2 (maximum deceleration), but they are not linked with motions, such as torque off, after stopping. DR_SSTOP deceleration time is about 1.5 times longer that the maximum deceleration time. In the case of DR_HOLD, stop immediately with no deceleration time.

Parameters

Parameter Name	Data Type	Default Value	Description
st_mode	int	-	stop mode • DR_QSTOP_STO: Quick stop (Stop Category 1 without STO(Safe Torque Off) • DR_QSTOP: Quick stop (Stop Category 2) • DR_SSTOP: Soft Stop • DR_HOLD: HOLE stop

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#1. The motion is terminated with a soft stop 2 seconds after moving to x1
p0 = posj(-148,-33,-54,180,92,32)
movej(p0, v=30, a=30)
x1 = posx(784, 543, 570, 0, 180, 0)
```

```
amovel (x1, vel=100, acc=200)   # Executes the next command immediately after the
motion with x1.
wait(2)                         # Temporarily suspends the program for 2 seconds.
stop(DR_SSTOP)                  # Stops the motion with a soft stop.
```

Related commands

- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)
- movesj()(p. 78)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- amovej()(p. 99)
- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.5.6 change_operation_speed()

Definition

change_operation_speed(speed)

Features

This function adjusts the operation velocity. The argument is the relative velocity in a percentage of the currently set velocity and has a value from 1 to 100. Therefore, a value of 50 means that the velocity is reduced to 50% of the currently set velocity.

Parameters

Parameter Name	Data Type	Default Value	Description
speed	int	-	operation speed(10~100)

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
change_operation_speed(10)
change_operation_speed(100)
#1. Motion with velocity specified to q0 and 20% of the specified velocity
q0 = posj(0, 0, 90, 0, 90, 0)
movej (q0, vel=10, acc=20)        # Moves to q0 at a velocity of 10mm/sec
change_operation_velocity(10)     # The velocities of all following motions executed
are 10% of the specified velocity.
q1 = posj(0, 0, 0, 0, 90, 0)
movej (q1, vel=10, acc=20)       # Moves to q1 at a velocity of 10% of 10mm/sec.
change_operation_speed(100)      # The velocities of all following motions executed
are 100% of the specified velocity.
movej (q0, vel=10, acc=20)       # Moves to q0 at a velocity 100% of 10mm/sec
```

Related commands

- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)

- movec()(p. 72)
- movesj()(p. 78)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- amovej()(p. 99)
- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.5.7 wait_manual_guide()

Features

This function enables the user to perform hand guiding (changing the position of the robot by pressing the Direct Teach button in the cockpit or the TP) during the execution of the program. The user executes the next command in one of the following two ways after hand guiding is completed (unless the program is terminated, it will wait at the command until one of the following is executed after the user performs hand guiding).

1. The user presses the "OK" or "Finish" button on the "Hand Guiding Execution" popup window generated from the TP.
2. A signal is applied to the digital input channel specified for "Manual guide release" in the safety I/O settings.

The current TCP position and the TCP position of the hand guided robot must be in the collaborative workspace in order to execute this command properly. Run the command after specifying the hand guiding area as the collaborative workspace and enabling it. An error is generated, and the program is terminated to ensure worker safety if the current position or hand guiding deviates from the collaborative workspace.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Sets up the collaborative workspace before executing the program.
# Surface 1: Point-1(1000,1000), Point-2(0,0)
# Surface 2: Point-1(1000,-1000), Point-2(0,0)
# Activation of domain 1 - Point(1000,0)

j00 = posj(0,0,90,0,90,0)
movej(j00,vel=20,acc=40)  # Enters the collaborative workspace.
wait_manual_guide()       # Direct teaching until the "Finish" button on the popup
window generated by the TP is pressed.
pos1 = get_current_posx() # Stores the directly taught point in pos1.
dposa = posx(0,0,-100,0,0,0)
movel(dposa, vel=300, acc=600, ref=DR_TOOL)
        # Retract 100 mm in the tool-Z direction from the taught position.
```

Related commands

- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)
- movesj()(p. 78)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- amovej()(p. 99)

- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.5.8 wait_nudge()

Features

This function enables users to resume the execution of the program through the user's nudge input (applying external force to the robot) when the execution of the program is paused. When the external force greater than the force threshold, it will proceed to the following command after the resume time, where the force threshold and the resume time are set at the collaborative workspace setting menu. This command can be used as an interlock during the program.

However, if the robot's configuration is in the singularity area, or if the force is applied continuously after the nudge input, warning will be occurred for safety.

For this function is allowed to execute within the collaborative workspace, please set the collaborative workspace, activate it, and assure the TCP position is in this workspace when this command is performed in advance.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Sets up the collaborative workspace before executing the program.
# Surface 1: Point-1(1000,1000), Point-2(0,0)
# Surface 2: Point-1(1000,-1000), Point-2(0,0)
# Activation of domain 1 - Point(1000,0)

j00 = posj(0,0,90,0,90,0)
movej(j00,vel=20,acc=40) # Enters the collaborative workspace.
wait_nudge()             # Wait for applying external force exceeding the threshold
to the robot
dposa = posx(0,0,-100,0,0,0)
movel(dposa, vel=300, acc=600, ref=DR_TOOL)
        # Retract 100 mm in the tool-Z direction from the taught position
```

Related commands

- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)
- movesj()(p. 78)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- amovej()(p. 99)
- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.5.9 enable_alter_motion()

Definition

enable_alter_motion(n,mode,ref,limit_dPOS,limit_dPOS_per)

Features

enable_alter_motion() and alter_motion() functions enable altering of the motion trajectory.

This function sets the configurations for altering function and allows the input quantity of alter_motion() to be applied to motion trajectory. The unit cycle time of generating altered motion is 10msec. Cycle time(n*10msec) can be adjusted using the input parameter n.

This function provides two modes: Accumulation mode and Increment mode. Input quantity of alter_motion() can be applied to motion trajectory in two ways: as an accumulated value or as an incremental value.

- In Accumulation mode, the input quantity means the absolute altering amount from current motion trajectory.
- In Increment mode, the input quantity means the incremental value from the previous absolute altering amount.

The reference coordinate can be changed through input parameter ref. Limitations for accumulation amount and increment amount can be set using input parameters limit_dPOS (accumulated limit) and limit_dPOS_per(incremental input limit during one cycle). The actual altering amount is constrained by these limits.

Parameters

Parameter Name	Data Type	Default Value	Description
n	int	None	Cycle time number
mode	int	None	Mode • DR_DPOS : accumulation amount • DR_DVEL : increment amount
ref	int	None	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • DR_TOOL: tool coordinate • user coordinate: user defined

Parameter Name	Data Type	Default Value	Description
limit_dPOS	list(float[2])	None	First value : limitation of position[mm] Second value : limitation of orientation[deg]
limit_dPOS_per	list(float[2])	None	First value : limitation of position[mm] Second value : limitation of orientation[deg]

Note

- _global_ref is applied if ref is None
- Accumulation amount or increment amount will not be limited if limit_dPOS or limit_dPOS_per is None.
- alter_motion() can be executed only in user thread.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
def alter_thread():
    global dir_a, wait_time
    alter_motion(dX) #dX : amount of alter motion

    if dX[0] > 9.9:
        dir_a = -10
    elif dX[0] < -9.9:
        dir_a = 10

    dX[0] = dX[0] + dir_a
```

```
    dX[3] = dX[3] + dir_a

    wait(wait_time)

dX = [10,0,0,10,0,0]

# Starting from V3.4, various orientation types are supported depending on the
ori_type setting in posx():
# dX = posx([10, 0, 0, 0, 0, 10], ori_type=DR_ELR_ZYZ, sol=None, turn=None)

J00 = posj(30,45,45,0,90,0)
movej(J00,vel=100,acc=100)

X1,sol = get_current_posx(DR_BASE)

J01 = posj(-30,45, 45,0,90,0)
movej(J01,vel=100,acc=100)

X2,sol = get_current_posx(DR_BASE)

# Alter motion is executed during the cycle time
cycle_time = 0.6

n_period = int(cycle_time * 1000 / 50)
wait_time = cycle_time

# cycle time:(n_period * 10)msec, mode:accumulate, reference coordination:base
coordination
# Lmitation of accumulation amount :50mm, 90deg
# Limitation of increment amount :60mm, 60deg
# Limit Alarm can occur if the cycle time is too short compare to the amount of alter
motion
enable_alter_motion(n=n_period,mode=DR_DPOS, ref=DR_BASE, limit_dPOS=[50,90],
limit_dPOS_per=[60,60])

th_id = thread_run(alter_thread, loop=True)

movel(pos=X1,vel=10,acc=100)
movel(pos=X2,vel=10,acc=100)

thread_stop(th_id)
disable_alter_motion() # deactivates alter motion
```

Related commands

- alter_motion()(p. 146)
- disable_alter_motion()(p. 148)

2.5.10 alter_motion()

Definition

alter_motion(pos)

Features

This function applies altering amount of motion trajectory when the alter function is activated. The meaning of the input values is defined in the description of enable_alter_motion().

Caution

- alter_motion() can be executed only in user thread.
- alter_motion() sets the orientation type of posx to DR_FIX_XYZ if it is None.

Note

- alter_motion() can be executed only in user thread.
- Alter motion can be adjusted through setting value limit_dPOS or limit_dPOS_per in enable_alter_motion function.
- Starting from SW version V3.4, alter_motion() supports various orientation types when pos is entered as posx(). If pos is entered in the format of list(float[6]), it operates as Fixed XYZ.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	list (float[6]) posx	-	position list posx (if ori_type is None, then ori_type is DR_FIX_XYZ)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
def alter_thread():
    global dir_a, wait_time
    alter_motion(dX) #dX : amount of alter motion

    if dX[0] > 9.9:
        dir_a = -10
    elif dX[0] < -9.9:
        dir_a = 10

    dX[0] = dX[0] + dir_a
    dX[3] = dX[3] + dir_a

    wait(wait_time)

dX = [10,0,0,10,0,0]

# Starting from V3.4, various orientation types are supported depending on the
ori_type setting in posx():
# dX = posx([10, 0, 0, 0, 0, 10], ori_type=DR_ELR_ZYZ, sol=None, turn=None)

J00 = posj(30,45,45,0,90,0)
movej(J00,vel=100,acc=100)

X1,sol = get_current_posx(DR_BASE)

J01 = posj(-30,45, 45,0,90,0)
movej(J01,vel=100,acc=100)

X2,sol = get_current_posx(DR_BASE)

# Alter motion is executed during the cycle time
cycle_time = 0.6

n_period = int(cycle_time * 1000 / 50)
wait_time = cycle_time

# cycle time:(n_period * 10)msec, mode:accumulate, reference coordination:base
coordination
# Lmitation of accumulation amount :50mm, 90deg
# Limitation of increment amount :60mm, 60deg
```

```
# Limit Alarm can occur if the cycle time is too short compare to the amount of alter
motion
enable_alter_motion(n=n_period,mode=DR_DPOS, ref=DR_BASE, limit_dPOS=[50,90],
limit_dPOS_per=[60,60])

th_id = thread_run(alter_thread, loop=True)

movel(pos=X1,vel=10,acc=100)
movel(pos=X2,vel=10,acc=100)

thread_stop(th_id)
disable_alter_motion() # deactivates alter motion
```

Related commands

- enable_alter_motion()(p. 143)
- disable_alter_motion()(p. 148)

2.5.11 disable_alter_motion()

Features

This function deactivates alter motion.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
def alter_thread():
    global dir_a, wait_time
    alter_motion(dX) #dX : amount of alter motion
```

```
    if dX[0] > 9.9:
        dir_a = -10
    elif dX[0] < -9.9:
        dir_a = 10

    dX[0] = dX[0] + dir_a
    dX[3] = dX[3] + dir_a

    wait(wait_time)

dX = [10,0,0,10,0,0]

# Starting from V3.4, various orientation types are supported depending on the
ori_type setting in posx():
# dX = posx([10, 0, 0, 0, 0, 10], ori_type=DR_ELR_ZYZ, sol=None, turn=None)

J00 = posj(30,45,45,0,90,0)
movej(J00,vel=100,acc=100)

X1,sol = get_current_posx(DR_BASE)

J01 = posj(-30,45, 45,0,90,0)
movej(J01,vel=100,acc=100)

X2,sol = get_current_posx(DR_BASE)

# Alter motion is executed during the cycle time
cycle_time = 0.6

n_period = int(cycle_time * 1000 / 50)
wait_time = cycle_time

# cycle time:(n_period * 10)msec, mode:accumulate, reference coordination:base
coordination
# Lmitation of accumulation amount :50mm, 90deg
# Limitation of increment amount :60mm, 60deg
# Limit Alarm can occur if the cycle time is too short compare to the amount of alter
motion
enable_alter_motion(n=n_period,mode=DR_DPOS, ref=DR_BASE, limit_dPOS=[50,90],
limit_dPOS_per=[60,60])

th_id = thread_run(alter_thread, loop=True)

movel(pos=X1,vel=10,acc=100)
movel(pos=X2,vel=10,acc=100)

thread_stop(th_id)
disable_alter_motion() # deactivates alter motion
```

Related commands

- enable_alter_motion()(p. 143)

- alter_motion()(p. 146)

2.5.12 check_robot_mastering()

Features

Checks whether the current robot is in a state that requires mastering or not.

Robot states that require mastering
When the controller power is forcibly turned off in Servo On state
When Servo On again after STO error occurs

Parameters

None

Return

Value	Description
0	mastering completed
1	mastering need

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ret = check_robot_mastering()
if ret == 1:
  move_home()
```

Related commands

- move_home()(p. 98)

2.5.13 motion_pause()

Features

This function pauses the motion being performed. Only the motion currently being performed is paused, and other functions that operate asynchronously are not affected.Pause the motion being performed. Only the motion currently being performed is paused, and other functions that operate asynchronously are not affected.

Parameters

None

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# pre-setting
set_velj(30)
set_accj(30)
set_velx(250)
set_accx(250)

# pause motion
movej([0, 0, 0, 0, 0, 0])
amovej([0, 0, 90, 0, 90, 0])
wait(1)
motion_pause()
wait(1)
motion_resume()
wait(5)

# pause motion in compliance control
movej(posj(0, 0, 90, 0, 90, 0))
task_compliance_ctrl()
amovel(posx(0, 0, -300, 0, 0, 0))
wait(1)
motion_pause()
wait(10)
motion_resume()
wait(5)
release_compliance_ctrl()

# pause force control
movej(posj(0, 0, 90, 0, 90, 0))
task_compliance_ctrl()
fd = [0, 0, -5, 0, 0, 0]
fctrl_dir= [0, 0, 1, 0, 0, 0]
set_desired_force(fd, dir=fctrl_dir)
wait(2)
motion_pause()
wait(2)
motion_resume()
wait(2)
release_force()
wait(0.5)
release_compliance_ctrl()
```

Related commands

- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)
- movesj()(p. 78)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- amovej()(p. 99)
- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.5.14 motion_resume()

Features

This function resumes paused motion.

Parameters

None

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# pre-setting
set_velj(30)
set_accj(30)
set_velx(250)
set_accx(250)

# pause motion
movej([0, 0, 0, 0, 0, 0])
amovej([0, 0, 90, 0, 90, 0])
wait(1)
motion_pause()
wait(1)
motion_resume()
wait(5)

# pause motion in compliance control
movej(posj(0, 0, 90, 0, 90, 0))
task_compliance_ctrl()
amovel(posx(0, 0, -300, 0, 0, 0))
wait(1)
motion_pause()
wait(10)
motion_resume()
wait(5)
release_compliance_ctrl()

# pause force control
movej(posj(0, 0, 90, 0, 90, 0))
task_compliance_ctrl()
fd = [0, 0, -5, 0, 0, 0]
```

```
fctrl_dir= [0, 0, 1, 0, 0, 0]
set_desired_force(fd, dir=fctrl_dir)
wait(2)
motion_pause()
wait(2)
motion_resume()
wait(2)
release_force()
wait(0.5)
release_compliance_ctrl()
```

Related commands

- movej()(p. 58)
- movel()(p. 63)
- movejx()(p. 68)
- movec()(p. 72)
- movesj()(p. 78)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- move_periodic()(p. 93)
- amovej()(p. 99)
- amovel()(p. 102)
- amovejx()(p. 105)
- amovec()(p. 108)
- amovesj()(p. 112)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)
- amove_periodic()(p. 126)

2.6 Servo Motion

2.6.1 servoj()

Definition

servoj(pos, vel, acc, time, mod)

Features

The command is the asynchronous motion command, and the next command is executed at the same time the motion begins. When the mode is set to Override mode, it follows the most recent target joint position in a motion within the maximum speed and acceleration among the continuously delivered commands. When the

mode is set to Queue mode, it can store up to 100 previous commands and sequentially follow the path. When 100 waypoints have been stored in Queue mode, the command will not return until reaching the next waypoint. If an Override command is input during Queue mode, it ignores the previously stored waypoints and follows the most recent target joint position.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posj list (float[6])	-	posj or joint angle list
vel (v)	float list (float[6])	None	maximum velocity (same to all axes) or maximum velocity (to an axis) [deg/s]
acc (a)	float list (float[6])	None	maximum acceleration (same to all axes) or maximum acceleration (acceleration to an axis) [deg/s^2]
time (t)	float	None	reach time [sec]
mod	int	DR_SERVO_OVERRIDE	Waypoint mode • DR_SERVO_OVERRIDE: Recent command • DR_SERVO_QUEUE: Queue mode (Keep waypoints)

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velj is applied if vel is None. (The initial value of _global_velj is 0.0 and can be set by set_velj.)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- After time is set, If reach time can't be keep because of condition of maximum velocity and acceleration, the reach time is adjusted automatically and notice through information message.

Caution

- Currently, it is not linked with the speed control function of the speed slide bar.
- Currently, it is not linked with check_motion(), change_operation_speed() functions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_velj(30)
set_accj(60)

Xt=posj(0, 0, 0, 0, 0, 0)
servoj(Xt)

Xt=posj(0, 0, 0, 0, 0, 0)
target = posx(90, 0, 120, -50, 50, -90)
del_t = [3, 0.1, 3, -3, 3, -3]

while 1:
    for i in range(0,6):
        Xt[i] = Xt[i] + del_t[i]

        if del_t[i] > 0:
            if Xt[i] > target[i]:
                Xt[i] = target[i]
        else:
            if Xt[i] < target[i]:
                Xt[i] = target[i]

    servoj(Xt)
```

Related commands

- posj()(p. 27)
- set_velj()(p. 47)
- set_accj()(p. 48)
- mwait()(p. 129)

2.6.2 speedl()

Definition

speedl(vel, acc, time)

Features

The command is the asynchronous motion command, and the next command is executed at the same time the motion begins. That motion follows the most recent target task velocity that is continuously delivered, within maximum acceleration.

Parameters

Parameter Name	Data Type	Default Value	Description
vel	list (float[6])	-	target joint velocity [deg/s]
acc (a)	float	None	maximum acceleration (same to all axes) or maximum acceleration (acceleration to an axis) [deg/s^2]
	list (float[6])		
time (t)	float	None	reach time [sec]

Note

- Abbreviated parameter names are supported. (a:acc, t:time)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accj.)
- After time is set, If reach time can't be keep because of condition of maximum and acceleration, the reach time is adjusted automatically and notice through information message.
- If you want to stop normally during movement, input vel as [0,0,0,0,0,0] or use the stop command.

Caution

- Currently, it is not linked with the speed control function of the speed slide bar.

- Currently, it is not linked with the DR_VAR_VEL option among the singularity options. When set with the DR_VAR_VEL option, it is automatically changed to DR_AVOID option and notice through information message.
- Currently, it is not linked with the force/compliance control function.
- Currently, it is not linked with check_motion(), change_operation_speed() functions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
movej(posj(0,0,90,0,90,0), v=30, a=60)
x = get_desired_posx()

v_y = 250
v = 5
at_max = 500
ar_max = 60
go_plus = True
while True:
    xd = get_desired_posx()
    if go_plus:
        speedl([v, v_y, v, 30, 0, 30], [at_max, ar_max])
        if xd[1] > x[1] + 200:
            go_plus = False
    else:
        speedl([-v, -v_y, -v, -30, -0, -30], [at_max, ar_max])
```

```
        if xd[1] < x[1] - 200:
            go_plus = True
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- mwait()(p. 129)
- stop()(p. 135)

2.6.3 speedj()

Definition

speedj(vel, acc, time)

Features

The command is the asynchronous motion command, and the next command is executed at the same time the motion begins. That motion follows the most recent target joint velocity that is continuously delivered, within maximum acceleration.

Parameters

Parameter Name	Data Type	Default Value	Description
vel	list (float[6])	-	target joint velocity [deg/s]
acc (a)	float	None	maximum acceleration (same to all axes) or maximum acceleration (acceleration to an axis) [deg/s^2]
	list (float[6])		
time (t)	float	None	reach time [sec]

Note

- Abbreviated parameter names are supported. (a:acc, t:time)
- _global_accj is applied if acc is None. (The initial value of _global_accj is 0.0 and can be set by set_accj.)
- After time is set, If reach time can't be keep because of condition of maximum and acceleration, the reach time is adjusted automatically and notice through information message.
- If you want to stop normally during movement, input vel as [0,0,0,0,0,0] or use the stop command.

Caution

- Currently, it is not linked with the speed control function of the speed slide bar.
- Currently, it is not linked with check_motion(), change_operation_speed() functions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
movej(posj(0,0,90,0,90,0), v=30, a=60)

v1 = 30
go_plus = True
while True:
    q = get_desired_posj()
    if go_plus:
        speedj([v1, 5, 5, 5, 5, 5], a=60)
        if q[0] > 90:
            go_plus = False
    else:
        speedj([-v1, -5, -5, -5, -5, -5], a=60)
        if q[0] < -90:
            go_plus = True
```

Related commands

- posj()(p. 27)
- set_velj()(p. 47)
- set_accj()(p. 48)
- mwait()(p. 129)
- stop()(p. 135)

2.6.4 servol()

Definition

servol(pos, vel, acc, time)

Features

The command is the asynchronous motion command, and the next command is executed at the same time the motion begins. That motion follows the most recent target task position that is continuously delivered, within maximum velocity, acceleration.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx list (float[6])	-	posx or position list
vel (v)	float list (float[2])	None	maximum velocity[mm/s] or maximum velocity[mm/s], maximum velocity[deg/s]
acc (a)	float list (float[2])	None	maximum acceleration[mm/s^2] or maximum acceleration[mm/s^2], maximum acceleration[deg/s^2]
time (t)	float	None	reach time [sec]

Note

- Abbreviated parameter names are supported. (v:vel, a:acc, t:time)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velj.)

- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accj.)
- After time is set, If reach time can't be keep because of condition of maximum velocity and acceleration, the reach time is adjusted automatically and notice through information message.

Caution

- Currently, it is not linked with the speed control function of the speed slide bar.
- Currently, it is not linked with the DR_VAR_VEL option among the singularity options. When set with the DR_VAR_VEL option, it is automatically changed to DR_AVOID option and notice through information message.
- Currently, it is not linked with the force/compliance control function.
- Currently, it is not linked with check_motion(), change_operation_speed() functions.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_velj(30)
set_accj(60)
set_velx(50)
set_accx(100)

movej(posj(0,0,90,0,90,0))
```

```
Xt=posx(368, 34.5, 442.5, 50.26, -180, 50.26)
servol(Xt,v=[100, 100], a=[200,300])

Xt=posx(368, 34.5, 442.5, 50.26, -180, 50.26)
target =posx(368, 34.5, 200, 50.26, -180, 50.26)
del_t = [0, 0, -3, 0, 3, 0]

while 1:
    for i in range(0,6):
        Xt[i] = Xt[i] + del_t[i]

        if del_t[i] > 0:
            if Xt[i] > target[i]:
                Xt[i] = target[i]
        else:
            if Xt[i] < target[i]:
                Xt[i] = target[i]
    servol(Xt,v=[100, 100], a=[200,300])
```

Related commands

- posx()(p. 28)
- set_velx()(p. 50)
- set_accx()(p. 52)
- mwait()(p. 129)

3 Auxiliary Control Commands

3.1 Robot Current Value

3.1.1 get_current_posj()

Features

This function returns the current joint angle.

Return

Value	Description
posj	Joint angle

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
q1 = get_current_posj()
```

Related commands

- get_desired_posj()(p. 174)

3.1.2 get_current_velj()

Features

This function returns the current joint velocity.

Return

Value	Description
float[6]	Joint speed

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
velj1 = get_current_velj()
```

Related commands

- get_desired_velj()(p. 175)

3.1.3 get_current_posx()

Definition

get_current_posx(ref, ori_type)

Features

Returns the pose and solution space of the current task coordinate. The pose is based on the ref coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
ref	Int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • user coordinate: User defined

Parameter Name	Data Type	Default Value	Description
ori_type	int	DR_ELR_ZYZ	orientation type • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Note

- ref: DR_BASE (base coordinate)/user coordinate (globally declared user coordinate)
- DR_BASE is applied when ref is omitted.

Return

Value	Description
Posx	Task space point
Int	Solution space (0 ~ 7)

Solution Space w.r.t. Robot Configuration

Solution space	Binary	Shoulder	Elbow	Wrist
0	000	Lefty	Below	No Flip
1	001	Lefty	Below	Flip
2	010	Lefty	Above	No Flip
3	011	Lefty	Above	Flip
4	100	Righty	Below	No Flip
5	101	Righty	Below	Flip
6	110	Righty	Above	No Flip

Solution space	Binary	Shoulder	Elbow	Wrist
7	111	Righty	Above	Flip

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
x1, sol = get_current_posx() #x1 w.r.t. DR_BASE

x1_wld, sol = get_current_posx(ref=DR_WORLD) #x1 w.r.t. DR_WORLD

DR_USR1=set_user_cart_coord(x1, x2, x3, pos)
set_ref_coord(DR_USR1)

x1, sol = get_current_posx(DR_USR1) #x1 w.r.t. DR_USR1

x2, sol = get_current_posx(DR_USR1, DR_FIX_XYZ) #x2 w.r.t. DR_USR1 (orientation type:
fixed xyz)
```

Related commands

- get_desired_posx()(p. 176)

3.1.4 get_current_tool_flange_posx()

Definition

get_current_tool_flange_posx(ref, ori_type)

Features

Returns the current tool flange pose based on the reference coordinate (ref). The tool flange will return to tcp=(0,0,0,0,0,0).

Parameters

Parameter Name	Data Type	Default Value	Description
ref	int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate
ori_type	int	DR_ELR_ZYZ	orientation type • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	Pose of tool flange

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
x1 = get_current_tool_flange_posx()                      #x1 : Flange pose base on the
base coordinate(default value)
x2 = get_current_tool_flange_posx(DR_BASE)               #x2 : Flange pose based on
the base coordinate
x3 = get_current_tool_flange_posx(DR_WORLD)              #x3 : Flange pose based on
the world coordinate
x4 = get_current_tool_flange_posx(ori_type=DR_FIX_XYZ)  #x4 : Flange pose base on the
base coordinate(default value), orientation type: fixed xyz
```

3.1.5 get_current_velx()

Definition

get_current_velx(ref)

Features

This function returns the current tool velocity based on the ref coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
ref	Int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate

Return

Value	Description
float[6]	Tool velocity

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
velx1 = get_current_velx()
# velx1 : velocity based on the base coordinate(default value)
velx2 = get_current_velx(DR_BASE)
# velx2 (=velx1) : velocity based on the base coordinate
velx3 = get_current_velx(DR_WORLD)
#velx3 : velocity based on the world coordinate
```

Related commands

- get_desired_velx()(p. 177)

3.1.6 get_current_rotm()

Definition

get_current_rotm(ref)

Features

This function returns the direction and matrix of the current tool based on the ref coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
ref	Int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate

Return

Value	Description
float[3][3]	Rotation matrix

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
rotm1 = get_current_rotm(DR_WORLD) # rotm1 : rotation matrix(3x3) based on the world coordinate
# The result value is stored in a 3x3 matrix
```

$$rotm1 = \begin{bmatrix} rotm1[0][0] & rotm1[0][1] & rotm1[0][2] \\ rotm1[1][0] & rotm1[1][1] & rotm1[1][2] \\ rotm1[2][0] & rotm1[2][1] & rotm1[2][2] \end{bmatrix}$$

3.1.7 get_joint_torque()

Features

This function returns the sensor torque value of the current joint.

Return

Value	Description
float[6]	JTS torque value

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
j_trq1 = get_joint_torque()
```

Related commands

- get_external_torque()(p. 172)
- get_tool_force(ref)(p. 173)

3.1.8 get_external_torque()

Features

This function returns the torque value generated by the external force on each current joint.

Return

Value	Description
float[6]	Torque value generated by an external force

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
trq_ext=get_external_torque()
```

Related commands

- get_joint_torque()(p. 172)
- get_tool_force()(p. 173)

3.1.9 get_tool_force()

Definition

get_tool_force(ref)

Features

This function returns the external force applied to the current tool based on the ref coordinate. The force and [1)]moment are based on the reference coordinate.

[1)]Before V2.8 software version, moment is based on the tool coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
ref	Int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate

Return

Value	Description
float[6]	External force applied to the tool

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
force_ext = get_tool_force(DR_WORLD) # force_ext: external force of the tool based on
the world coordinate
```

Related commands

- get_joint_torque()(p. 172)
- get_external_torque()(p. 172)

3.2 Robot Target Value

3.2.1 get_desired_posj()

Features

This function returns the current target joint angle. It cannot be used in the movel, movec, movesx, moveb, move_spiral, or move_periodic command.

Return

Value	Description
posj	Joint angle

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_INVALID)	Invalid command

Example

```
jp1 = get_desired_posj()
```

Related commands

- get_current_posj()(p. 165)

3.2.2 get_desired_velj()

Features

This function returns the current target joint velocity. It cannot be used in the movel, movec, movesx, moveb, move_spiral, or move_periodic command.

Return

Value	Description
float[6]	Target joint velocity

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_INVALID)	Invalid command

Example

```
velj1 = get_desired_velj()
```

Related commands

- get_current_velj()(p. 165)

3.2.3 get_desired_posx()

Definition

get_desired_posx(ref, ori_type)

Features

This function returns the target pose of the current tool. The pose is based on the ref coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
ref	Int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • user coordinate: User defined
ori_type	int	DR_ELR_ZYZ	orientation type • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Note

- ref: DR_BASE (base coordinate)/user coordinate (globally declared user coordinate)

- DR_BASE is applied when ref is omitted.

Return

Value	Description
float[6]	Tool velocity

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
x1 = get_desired_posx() #x1 w.r.t. DR_BASE
x2 = posx(100, 0, 0, 0, 0, 0)
x3 = posx(0, 0, 20, 20, 20, 20)
pos = x3
DR_USR1=set_user_cart_coord(x1, x2, x3, pos)
set_ref_coord(DR_USR1)

xa = get_desired_posx(DR_USR1) #xa w.r.t. DR_USR1
xb = get_desired_posx(DR_WORLD, DR_ELR_ZYX) #xb w.r.t. DR_WORLD, orientation type:
euler zyx
```

Related commands

- get_current_posx()(p. 166)

3.2.4 get_desired_velx()

Definition

get_desired_velx(ref)

Features

This function returns the target velocity of the current tool based on the ref coordinate. It cannot be used in the movej, movejx, or movesj command.

Parameters

Parameter Name	Data Type	Default Value	Description
ref	Int	DR_BASE	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate

Return

Value	Description
float[6]	Tool velocity

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_INVALID)	Invalid command

Example

```
vel_x1 = get_desired_velx()
# vel_x1 : desired velocity of the tool based on the base coordinate(default value)
vel_x2 = get_desired_velx(DR_BASE)
# vel_x2 : desired velocity of the tool based on the base coordinate
vel_x3 = get_desired_velx(DR_WORLD)
# vel_x3 : desired velocity of the tool based on the world
```

Related commands

- get_current_velx()(p. 170)

3.3 Control State Value

3.3.1 get_control_mode()

Features

This function returns the current control mode.

Return

Value	Description
int	Control mode 3 : Position control mode 4 : Torque control mode

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
mode = get_control_mode()
```

3.3.2 get_control_space()

Features

This function returns the current control space.

Return

Value	Description
int	Control mode 1 : Joint space control 2 : Task space control

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
x1 = get_control_space()
```

3.3.3 get_current_solution_space()

Features

This function returns the current solution space value.

Return

Value	Description
int	Solution space (0 ~ 7)

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
sol = get_current_solution_space()
```

Related commands

- get_solution_space(pos)(p. 181)

3.3.4 get_solution_space()

Definition

get_solution_space(pos)

Features

This function obtains the solution space value for the entered pos(posj).

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posj	-	posj or position list
	list (float[6])		

Return

Value	Description
0 ~ 7	Solution space

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
q1 = posj(0, 0, 0, 0, 0, 0)
sol1 = get_solution_space(q1)
sol2 = get_solution_space([10, 20, 30, 40, 50, 60])
```

Related commands

- get_current_solution_space()(p. 180)

3.3.5 get_orientation_error()

Definition

get_orientation_error(xd, xc, axis)

Features

This function returns the orientation error value between the arbitrary poses xd and xc of the axis.

Parameters

Parameter Name	Data Type	Default Value	Description
xd	posx list (float[6])	-	posx or position list
xc	posx list (float[6])	-	posx or position list

Parameter Name	Data Type	Default Value	Description
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis

Return

Value	Description
float	Orientation error value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
xd = posx(0, 0, 0, 0, 0, 0)
xc = posx(10, 20, 30, 40, 50, 60)
diff = get_orientation_error(xd, xc, DR_AXIS_X)
```

Related commands

- get_current_rotm()(p. 171)

3.4 Robot Safety Setup Parameters

3.4.1 get_cockpit()

Features

Returns the cockpit information of the current safety setup parameters.

Return

Value	Data Types	Description
ret	class.config_cockpit	About cockpit settings

Class

class.config_cockpit

Field	Data Types	Description
obj	list	Object information for that class
enable	int	0 : Disable 1 : Enable
button	int[2]	0 : Direct Instruction 1 : TCP-Z 2 : TCP-XY 3 : Orietation Only 4 : Position Only
recovery_teach	int	0 : Disable 1 : Enable

Example

```
ret = get_cockpit()
```

```
print(ret.button) ## list of button state
```

3.4.2 get_collision_sensitivity()

Features

Returns the collision sensitivity value of the current safety settings parameter.

Return

Value	Data Types	Description
ret	float	Conflict sensitivity information (0-100)

Example

```
def print_tp(s):
    tp_log(str(s))

ret = get_collision_sensitivity()

### collision sensitivity
print_tp(ret)
```

3.4.3 get_configurable_io()

Features

Returns the currently set general IO settings information.

Return

Value	Data Types	Description
ret	class.config_configurable_io	About general IO settings

Class

class.config_configuarable_io

Field	Data Types	Description

obj	list	Object information for that class
input	int[32]	About Digital Input Settings
output	int[32]	About Digital Output Settings

Example

```
ret = get_configurable_io()
print(ret.input)
print(ret.output)
```

3.4.4 get_current_tcp()

Features

Returns the TCP information of the currently set safety setting parameter.

Return

Value	Data Types	Description
ret	class.config_tcp_symbol	TCP Setup Infomation

Class

class.config_tcp_symbol

Field	Data Types	Description
obj	list	Object information for that class
symbol	string	TCP Name
tcp	class.config_tcp	TCP Information

class.config_tcp

Field	Data Types	Description

obj	list	Object information for that class
offset	float[6]	Target location

Example

```
ret = get_current_tcp()
print(ret.symbol) ## tcp symbol
print(ret.tcp.offset) ## tcp target pos list
```

3.4.5 get_current_tool_shape()

Features

Returns the currently set tool geometry information among the current safety setup parameters.

Return

Value	Data Types	Description
ret	class.config_tool_shape	Tool Geometry Settings Information

Class

class.config_tool_shape

Field	Data Types	Description
obj	list	Object information for this class
validity	int[5]	Validate(0, 1)
shape	class.safety_object[5]	Detailed geometry information

class.safety_object

Field	Data Types	Description
obj	list	Object information for this class

target_ref	int	Unused variable
object_type	int	Safety Object Types 0: sphere 1: capsule 2: cuboid
object	class.safety_object_data	About each solid shape

Note
The safety_object_data class is a class organized as a union union in C++ and returns a different class depending on the object_type.

class.safety_object_data

object_type	Field	Data Types	Description
-	obj	list	Object information for this class
0	sphere	class.safety_object_sphere	sphere-shaped objects
1	capsule	class.safety_object_capsule	Capsule-Shaped Objects
2	cuboid	class.safety_object_cuboid	Cuboid Shape Objects

Note
The previous name 'cube' is changed into 'cuboid'.
Can use either 'cube' or 'cuboid' based on the current version of the manual **but usage of 'cube' will be deprecated. 'cube' is not recommended**

class.safety_object_sphere

Field	Data Types	Description
obj	list	Object information for this class

radius	float	Radius
target_pos	class.point_3d	3D point information (x, y, z)

class.point_3d

Field	Data Types	Description
obj	list	Object information for this class
x	float	x-axis
y	float	y-axis
z	float	z-axis

class.safety_object_capsule

Field	Data Types	Description
obj	list	Object information for this class
radius	float	Radius
target_pos	class.point_3d[2]	3D point information (x, y, z)

class.safety_object_cuboid

Field	Data Types	Description
obj	list	Object information for this class
target_pos	class.point_3d[2]	3D point information (x, y, z)

class.point_2d

Field	Data Types	Description

obj	list	Object information for this class
x	float	x-axis
y	float	y-axis

Example

```
tool = get_current_tool_shape()

tp_log("---------------- example start --------------------")

validity = tool.validity # get tool validity
indices = [i for i, x in enumerate(validity) if x] # make list of activated part
indexs

tp_log("part validity: {}".format(validity))
for n in indices:
    tp_log("> part{} datas".format(n))
    tool_shape = tool.shape[n]

    obj_type = tool_shape.object_type # get tool object type

    if obj_type == 0: # obj type is Sphere
        r = tool_shape.object.sphere.radius
        target_pos = tool_shape.object.sphere.target_pos
        target_pos = [ target_pos.x, target_pos.y, target_pos.z]
        tp_log("type: sphere | radious: {:.2f} pos: {}".format(r, target_pos))

    elif obj_type == 1: # obj type is Capsule
        r = tool_shape.object.capsule.radius
        target_pos = tool_shape.object.capsule.target_pos
        target_pos = [[ target_pos[j].x, target_pos[j].y, target_pos[j].z] for j in
range(2)]
        tp_log("type: capsule | radious: {:.2f} pos: {}".format(r, target_pos))

    elif obj_type == 2: # obj type is Cuboid
        # the previous name 'cube' is changed into 'cuboid'
        # can use either 'cube' or 'cuboid' based on the current version of the
manual but usage of 'cube' will be deprecated. 'cube' is not recommended
        # target_pos = tool_shape.object.cube.target_pos
        target_pos = tool_shape.object.cuboid.target_pos
        target_pos = [[ target_pos[j].x, target_pos[j].y, target_pos[j].z] for j in
range(2)]
        tp_log("type: cuboid | pos: {}".format(target_pos))
tp_log("---------------- example end ----------------------")
```

3.4.6 get_current_tool()

Features

Returns the currently set tool information among the current safety setting parameters.

Return

Value	Data Types	Description
ret	class.config_tool_symbol	Tool settup information

Class

class.config_tool_symbol

Field	Data Types	Description
obj	list	Object information for that class
symbol	string	Tool name
tool	class.config_tool	Tool information

class.config_tool

Field	Data Types	Description
obj	list	Object information for that class
weight	float	Tool Mass
xyz	float[3]	Center of gravity information
inertia	float[6]	Inertia Information

Example

```
ret = get_current_tool()
```

```
print(ret.symbol) ## tool symbol
print(ret.tool.weight) ## tool weight
```

3.4.7 get_general_range()

Features

Returns the TCP/Robot limit value of the current safety settings parameter.

Parameters

None

Return

Value	Data Types	Description
ret	class.config_general_range	About setting TCP/robot restrictions

Class

class.config_general_range

Field	Data Types	Description
obj	list	Object information for that class
normal	class.general_range	TCP/Robot restrictions in normal mode
reduced	class.general_range	TCP/Robot Restrictions in Slow Mode

class.general_range

Field	Data Types	Description
obj	list	Object information for that class
max_force	float	Maximum external force limit
max_power	float	Maximum power limit

max_speed	float	Maximum TCP Rate Limit
max_momentum	float	Maximum momentum limit

Example

```
def print_tp(s):
    tp_log(str(s))

ret = get_general_range()

### normal mode
print_tp(ret.normal.max_force)
print_tp(ret.normal.max_power)
print_tp(ret.normal.max_speed)
print_tp(ret.normal.max_momentum)

### reduced mode
print_tp(ret.reduced.max_force)
print_tp(ret.reduced.max_power)
print_tp(ret.reduced.max_speed)
print_tp(ret.reduced.max_momentum)
```

3.4.8 get_idle_off()

Features

Returns the SERVO OFF idle information set during the current safety setup parameter.

Return

Value	Data Types	Description
ret	class.config_idle_off	About setting the SERVO OFF idle time

Class

class.config_idle_off

Field	Data Types	Description
obj	list	Object information for that class

func_enable	int	0 : Disable 1 : Enable
elapse_time	float	Idle time

Example

```
ret = get_idle_off()
print(ret.elapse_time) ## elapse time of idle off config
```

3.4.9 get_install_pose()

Features

Returns information about the currently set installation posture among the safety setup parameters.

Return

Value	Data Types	Description
ret	class.config_install_pose	About setting the installation stance

Class

class.config_install_pose

Field	Data Types	Description
obj	list	About objects of t
gradient	float	Skew information
rotation	float	Rotation information

Example

```
ret = get_install_pose()
print(ret.gradient) #### gradient of the install pose
```

```
print(ret.rotation) #### rotation of the install pose
```

3.4.10 get_io_speed_ratio()

Features

Returns the currently set Safe IO Rate Deceleration Rate information.

Return

Value	Data Types	Description
ret	float	Safe IO slowdown rate (0% to 100%)

Example

```
ret = get_io_speed_ratio()
print(ret)
```

3.4.11 get_joint_range()

Features

Returns the joint angle limit value of the current safety setup parameter.

Return

Value	Data Types	Description
ret	class.config_joint_range	About setting joint angle limits

Class

class.config_joint_range

Field	Data Types	Description
obj	list	About objects of that class
normal	class.joint_range	Joint angle limits in normal mode

reduced	class.joint_range	Joint Angle Limits in Decelerated Mode

class.joint_range

Field	Data Types	Description
obj	list	About objects of that class
max_vel	float[6]	Maximum Speed Limits for Each Axis
max_range	float[6]	Maximum Range Limits for Each Axis
min_range	float[6]	Minimum Range Limit for Each Axis

Example

```
def print_tp(s):
    tp_log(str(s))

ret = get_joint_range()

### normal mode
print_tp(ret.normal.max_vel)
print_tp(ret.normal.max_range)
print_tp(ret.normal.min_range)

### reduced mode
print_tp(ret.reduced.max_vel)
print_tp(ret.reduced.max_range)
print_tp(ret.reduced.min_range)
```

3.4.12 get_modbus_data_list()

Features

Returns all currently set MODBUS information.

Return

Value	Data Types	Description
ret	class.modbus_data_list	All set MODBUS data

Class

class.modbus_data_list

Field	Data Types	Description
obj	list	Object information for that class
count	int	Number of registered signals
reg	class.modbus_data[100]	Registered MODBUS information (up to 100)

Note
The modbus_data class is a class organized in C++'s union union structure, which returns a different class depending on the type.

class.modbus_data

type	Field	Data Types	Description
	obj	list	Object information for that class
	type	str	tool name
0	tcp	class.write_modbus_tcp_data	modbus tcp register information
1	rtu	class.write_modbus_rtu_data	modbus rtu register information

class.write_modbus_tcp_data

Field	Data Types	Description
obj	list	Object information for that class
symbol	str	signal name
ip	str	ip address

port	int	port number
slave_id	int	(0~255) slave id
reg_type	int	register type 0 : discrete input 1 : Coil 2 : input register 3 : holding register
reg_index	int	Register number
reg_value	int	Register value

class.write_modbus_rtu_data

Field	Data Types	Description
obj	list	Object information for that class
symbol	str	signal name
port	str	ip address
slave_id	int	port number
baudrate	int	(0~255) slave id
byte_size	int	Number of data bits
parity	str	Parity checks
stop_bit	int	stop bit 수
reg_type	int	Register type 0 : discrete input 1 : Coil 2 : input register 3 : holding register

reg_value	int	Register value

Example

```
ret = get_modbus_data_list()
```

3.4.13 get_nudge()

Features

Returns the nudge information of the currently set safety setting parameter.

Return

Value	Data Types	Description
ret	class.config_nudge	About nudge settings

Class

class.config_nudge

Field	Data Types	Description
obj	list	Object information for this class
enable	int	0: Disabled 1: Enabled
input_force	float	Input force (N)
delay_time	float	Delay time

Example

```
ret = get_nudge()
print(ret.input_force) ## force for nudge
```

3.4.14 get_safety_data_version()

Features

Returns the current safety settings parameter version information.

Return

Value	Data Types	Description
ret	int	Safety parameter version information 1: Safety settings data version as of v2.12.1

Example

```
ret = get_safety_data_version()
```

3.4.15 get_safety_function()

Features

Returns the Safe Stop Mode value of the current Safety Settings parameter.

Return

Value	Data Types	Description
ret	int[15]	Returns the stop mode for each index (see example for status per index) 0 : STO 2 : SS1 3 : SS2 4 : RS1

Example

```
# 0. Emergency Stop
# 1. Protective Stop
# 2. StandStill Monitoring
```

```
# 3. Joint Angle Monitoring
# 4. Joint Speed Monitoring
# 5. Joint Torque Monitoring
# 6. Collision Detection - Standalone Workspace
# 7. Collision Detection - Collaborative Workspace
# 8. TCP Position Monitoring
# 9. TCP Orientation Monitoring
# 10. TCP Speed Monitoring
# 11. TCP Force Monitoring - Standalone Workspace
# 12. TCP Force Monitoring - Collaborative Workspace
# 13. Momentum Monitoring
# 14. Power Mon.

ret = get_safety_function()
```

3.4.16 get_safety_io()

Features

Returns the currently set safety IO information among the current safety setting parameters.

Parameters

None

Return

Value	Data Types	Description
ret	class.config_safety_io	safety IO setup information

Class

class.config_safety_io

Field	Data Types	Description
obj	list	Object information for that class

input	int[16]	safety input information(0~15) 0: Not Used 1: Protective Stop (STO) (L) 2: Emergency Stop (L) 3: Protective Stop (L) 4: Reduced Speed (L) 5: 3-pos Enable Switch (L to H & H) 6: Hand-guiding Enable Switch (L to H & H) 7: HGC End & Task Resume (L to H) 8: Protective Stop (Auto Reset & Resume) (L) 9: Safe Zone Dynamic Enable (H) 10: Remote Control Mode (H) 11: Emergency Stop (No Loopback) (L) 12: Error Reset & Resume (L to H) 13: Interlock Reset (L to H) 14: Protective Stop (SS1) (L) 15: Protective Stop (SS2) (L) 16: Safe Zone Dynamic Enable (L)

output	int[16]	safety output information(0~15) 0: Not Used 1: Safe Torque Off (STO) (L) 2: Safe Operating Stop (SOS) (L) 4: Normal Speed Status (L) 5: Reduced Speed Status (L) 6: Auto Mode (L) 7: Manual Mode (L) 8: Standalone Zone (L) 9: Collaborative Zone (L) 10: High Priority Zone(L) 11: Tool Orientation Limited Zone (L) 12: Emergency Stop (L) 13: Emergency Stop (excl. No Loopback Input) (L) 15: Designated Zone (L) 17: Remote Mode (L) 18: Interlock Reset Needed(L)

Example

```
ret = get_safety_io()
print(ret.input) ### input list of safety io
print(ret.output) ### output list of safety io
```

3.4.17 get_safety_zone_cnt()

Features

Returns the currently set number of safe zones.

Return

Value	Data Types	Description
ret	int	Number of safety zones set

Example

```
ret = get_safety_zone_cnt()
print(ret)
```

3.4.18 get_safety_zone_list()

Features

Returns all current safe zone information.

Return

Value	Data Types	Description
ret	class.config_safety_zone_list	All safe zone information (up to 20)

Class

class.config_safety_zone_list

Field	Data Types	Description
obj	list	Object information for that class
id	int	Safe zone UUID
alias	string	safe area name
type	int	safe zone type. 0:space limit, 1:local zone
property	safety_zone_property_data	safety zone property data
shape	safety_zone_shape	safety zone shape

Note

The safety_zone_property_data class is a class structured as a union union in C++ and returns a different class depending on the type.

class.safety_zone_property_data

type	Field	Data Types	Description
	obj	list	Object information for that class
0	space_limit_zone	class.safety_zone_property_space_limit	space limit zone setup information
1	local_zone	class.safety_zone_property_local_zone	local zone setup information

class.safety_zone_property_space_limit

Field	Data Types	Description
obj	list	Object information for that class
inspection_type	int	0: body 1: tcp
joint_range_override	class.local_zone_property_joint_range	override joint range
dynamic_zone_enable	int	Dynamic zone enable option 0 : not used 1~8 : safety input channel
inside_zone_detection	int	Inside zone detection option 0 : not used 1~8 : safety input channel

class.local_zone_property_joint_range

Field	Data Types	Description
obj	list	Object information for that class

override	int[6]	0: no override 1: override global property
min_range	float[6]	Minimum Range
max_range	float[6]	Maximum Range

class.safety_zone_property_local_zone

Field	Data Types	Description
obj	list	Object information for that class
joint_range_override	class.local_zone_property_joint_range	Override Joint Angle Limits
joint_speed_override	class.local_zone_property_joint_speed	Override Joint Velocity Limits
tcp_force_override	class.local_zone_property_tcp_force	Override TCP External Force Limits
tcp_power_override	class.local_zone_property_tcp_power	Override Power Limits
tcp_speed_override	class.local_zone_property_tcp_speed	Override TCP Velocity Limits
tcp_momentum_override	class.local_zone_property_tcp_momentum	Override Momentum Limits
collision_override	class.local_zone_property_collision	Override Collision Sensitivity
speed_rate	class.local_zone_property_speed_rate	Override Speed Deceleration Rate
collision_violation_stop_mode_override	class.local_zone_property_collision_stopmode	Override Collision Violation Stopping Mode
force_violation_stop_mode_override	class.local_zone_property_tcpslf_stopmode	Override External Force Violation Stopping Mode

tool_orientation_limit_override	class.local_zone_property_tool_orientation	Tool Orientation Limits
dynamic_zone_enable	int	Safety Input-Based Dynamic Activation 0 : not used 1~8 : safety input channel
led_override	int	LED Color Change Information 0 : Not Used 1 : Green 2 : Yellow
nudge_enable	int	Nudge activation information 0: disable 1: enable
allow_ress_safe_work	int	Allow less safe work 0: not allowed 1: allowed
override_reduce	int	Setting to ignore deceleration mode 0: Do not ignore 1: Ignore
inside_zone_detection	int	Whether to enable Designated Zone detection 0 : not used 1 ~ 8 : Safety IO output channel 9 ~ 24 : General IO output channels (each means General IO 1~16)
collaborative_zone	int	Enable or disable co-working space 0 : Not Used 1 : Used

class.local_zone_property_joint_speed

Field	Data Types	Description

obj	list	Object information for that class
override	int[6]	0: no override 1: override global property
speed	float[6]	Speed information

class.local_zone_property_tcp_force

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property
force	float	Force information

class.local_zone_property_tcp_power

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property
power	float	Mechanical power information

class.local_zone_property_tcp_speed

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property

speed	float	TCP speed information

class.local_zone_property_tcp_momentum

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property
momentum	float	Momentum information

class.local_zone_property_collision

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property
sensitivity	float	Conflict sensitivity information

class.local_zone_property_speed_rate

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property
speed_rate	float	Deceleration Rate Information

class.local_zone_property_collision_stopmode

Field	Data Types	Description

obj	list	Object information for that class
override	int	0: no override 1: override global property
stop_mode	int	0: STO 2: SS1 3: SS2 4: RS1

class.local_zone_property_tcpslf_stopmode

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property
stop_mode	int	Stop Mode Information for TCP SLF 0 : STO 2 : SS1 3 : SS2 4 : RS1

class.local_zone_property_tool_orientation

Field	Data Types	Description
obj	list	Object information for that class
override	int	0: no override 1: override global property
direction	float[3]	Direction
angle	float	Angle

class.safety_zone_shape

Field	Data Types	Description
obj	list	Object information for that class
coordinate	int	Coordinate 0 : base 2 : world
shape_type	int	Shape type 0 : Sphere 1 : Cylinder 2 : Cuboid 3 : Tilted Cuboid 4 : Multi-Plane
shape_data	class.SAFETY_ZONE_SHAPE_DATA	Safe zone shape data
margin	float	Margin Min value: 0 Max value: 9999
valid_space	int	Valid space 0: Internal 1: outside

Note
The safety_zone_shape_data class is a class structured as a union union in C++ and returns a different class depending on the type.

class.safety_zone_shape_data

shape_type	Field	Data Types	Description

	obj	list	Object information for that class
0	sphere	class.safety_zone_shape_sphere	Sphere shape DataSources
1	cylinder	class.safety_zone_shape_cylinder	Cylinder shape Data
2	cuboid	class.safety_zone_shape_cuboid	Cuboid shape Data
3	obb	class.safety_zone_shape_tilted_cuboid	Tilted Cuboid shape Data
4	multiplane	class.safety_zone_shape_multi_plane	Multifaceted Column shape Data

class.safety_zone_shape_sphere

Field	Data Types	Description
obj	list	Object information for that class
center	class.point_3d	Centers information
radius	float	Radius

class.safety_zone_shape_cylinder

Field	Data Types	Description
obj	list	Object information for that class
center	class.point_2d	Centers information
radius	float	Radius
zlolimit	float	Z Axis Lower Limit
zuplimit	float	Z Axis Upper Limit

class.safety_zone_shape_cuboid

Field	Data Types	Description
obj	list	Object information for that class
xlolimit	float	x Axis Lower Limit
xuplimit	float	x Axis Upper Limit
ylolimit	float	y Axis Lower Limit
yuplimit	float	y Axis Upper Limit
zlolimit	float	z Axis Lower Limit
zuplimit	float	z Axis Upper Limit

class.safety_zone_shape_tilted_cuboid

Field	Data Types	Description
obj	list	Object information for that class
origin	class.point_3d	Centers information
u	class.point_3d	u vector
v	class.point_3d	v vector
w	class.point_3d	w vector

class.safety_zone_shape_multi_plane

Field	Data Types	Description
obj	list	Object information for that class
valid_plane	int[6]	Valid plane information

plane	class.line[6]	Plane information
zlolimit	float	z Axis Lower Limit
zuplimit	float	z Axis Upper Limit
space_point	class.point_2d	Selection Space

class.point_3d

Field	Data Types	Description
obj	list	Object information for this class
x	float	x-axis
y	float	y-axis
z	float	z-axis

class.line

Field	Data Types	Description
obj	list	Object information for this class
from_point	class.point_2d	Start point of the Plane
to_point	class.point_2d	End point of the Plane

class.point_2d

Field	Data Types	Description
obj	list	Object information for this class
x	float	x-axis

y	float	y-axis

Example

```
ret = get_safety_zone_list()
```

3.4.19 get_tcp_list()

Features

Returns all currently registered TCP information for the current safety setting parameter.

Return

Value	Data Types	Description
ret	class.config_tcp_list	All registered TCP information

Class

class.config_tcp_list

Field	Data Types	Description
obj	list	Object information for that class
count	int	Number of registered TCPs
tcp_list	class.config_tcp_symbol[50]	Registered TCP information (up to 50)

class.config_tcp_symbol

Field	Data Types	Description
obj	list	Object information for that class
symbol	string	TCP name

tcp	class.config_tcp	TCP details

class.config_tcp

Field	Data Types	Description
obj	list	Object information for that class
offset	float[6]	Target location

Example

```
ret = get_tcp_list()

for i in range(0, ret.count):
    print(ret.tcp_list[i].symbol)
    print(ret.tcp_list[i].tcp.offset)
```

3.4.20 get_tcp_symbol()

Features

Returns the currently active TCP name.

Return

Value	Data Types	Description
ret	str	Enabled TCP name

Example

```
ret = get_tcp_symbol()
```

3.4.21 get_tool_list()

Features

Returns all currently registered tool information for the current safety setting parameter.

Return

Value	Data Types	Description
ret	class.config_tool_list	Information about all registered tools

Class

class.config_tool_list

Field	Data Types	Description
obj	list	Object information for that class
count	int	Number of registered tools
tool_list	class.config_tool_symbol[50]	Registered tool information (up to 50)

class.config_tool_symbol

Field	Data Types	Description
obj	list	Object information for that class
symbol	string	Tool name
tool	class.config_tool	Tool details

class.config_tool

Field	Data Types	Description
obj	list	Object information for that class
weight	float	Tool Mass
xyz	float[3]	Center of gravity information
inertia	float[6]	Inertia Information

Example

```
ret = get_tool_list()

for i in range(0, ret.count):
    print(ret.tool_list[i].symbol)
    print(ret.tool_list[i].tool.weight)
```

3.4.22 get_tool_shape_list()

Features

Returns all currently registered tool shape information for the current safety setting parameter.

Return

Value	Data Types	Description
ret	class.config_tool_shape_list	Information about all registered tools shape

Class

class.config_tool_shape_list

Field	Data Types	Description
obj	list	Object information for that class
count	int	Number of registered tools
tool_shape_list	class.config_tool_shape_symbol[50]	Registered tool shape information (up to 50)

class.config_tool_shape_symbol

Field	Data Types	Description
obj	list	Object information for that class

symbol	string	Tool name
tool_shape	class.config_shape_tool	Tool details

class.config_tool_shape

Field	Data Types	Description
obj	list	Object information for that class
validity	int[5]	Validate (0, 1)
shape	class.safety_object[5]	tool shape, shape information

class.safety_object

Field	Data Types	Description
obj	list	Object information for that class
target_ref	int	Unused variable
object_type	int	Tool shape type 0: Sphere 1: Capsule 2: Cuboid
object	class.safety_object_data	Tool shape type details 0: class.safety_object_sphere 1: class.safety_object_capsule 2: class.safety_object_cuboid

Note

The previous name 'cube' is changed into 'cuboid'.

Can use either 'cube' or 'cuboid' based on the current version of the manual **but usage of 'cube' will be deprecated. 'cube' is not recommended**

class.safety_object_sphere

Field	Data Types	Description
obj	list	Object information for that class
radius	float	radius
target_pos	class.point_3d	Tool point * Elements the class.point_3d - X point (float) - Y point (float) - Z point (float)

class.safety_object_capsule

Field	Data Types	Description
obj	list	Object information for this class
radius	float	Radius
target_pos	class.point_3d[2]	Tool point * Elements the class.point_3d - X point (float) - Y point (float) - Z point (float)

class.safety_object_cuboid

Field	Data Types	Description
obj	list	Object information for that class

target_pos	class.point_3d[2]	Tool point * Elements the class.point_3d - X point (float) - Y point (float) - Z point (float)

Example

```
tools = get_tool_shape_list()

tp_log("---------------- example start --------------------")
tp_log("tool count: {}".format( tools.count))

for i in range( tools.count ): # loop for tool count

    tool = tools.tool_shape_list[i]

    name = tool.symbol # get tool name
    validity = tool.tool_shape.validity # get tool validity
    indices = [i for i, x in enumerate(validity) if x] # make list of activated part
indexs

    tp_log("-------------------------------------------------")
    tp_log("tool{:d} name: {} validity: {}".format(i, name, validity))
    for n in indices:
        tp_log("> tool{:d} part{} datas".format(i, n))
        tool_shape = tool.tool_shape.shape[n]

        obj_type = tool_shape.object_type # get tool object type

        if obj_type == 0: # obj type is Sphere
            r = tool_shape.object.sphere.radius
            target_pos = tool_shape.object.sphere.target_pos

            target_pos = [ target_pos.x, target_pos.y, target_pos.z]
            tp_log("type: sphere | radious: {:.2f} pos: {}".format(r, target_pos))

        elif obj_type == 1: # obj type is Capsule
            r = tool_shape.object.capsule.radius
            target_pos = tool_shape.object.capsule.target_pos

            target_pos = [[ target_pos[j].x, target_pos[j].y, target_pos[j].z] for j
in range(2)]
            tp_log("type: capsule | radious: {:.2f} pos: {}".format(r, target_pos))

        elif obj_type == 2: # obj type is Cuboid
            # the previous name 'cube' is changed into 'cuboid'
            # can use either 'cube' or 'cuboid' based on the current version of the
manual but usage of 'cube' will be deprecated. 'cube' is not recommended
```

```
            target_pos = tool_shape.object.cuboid.target_pos

            target_pos = [[ target_pos[j].x, target_pos[j].y, target_pos[j].z] for j
in range(2)]
            tp_log("type: cuboid | pos: {}".format(target_pos))

tp_log("---------------- example end ----------------------")
```

3.4.23 get_tool_shape_symbol()

Features

Returns the name of the currently active tool shape.

Return

Value	Data Types	Description
ret	str	Active tool shape name

Example

```
ret = get_tool_shape_symbol()
```

3.4.24 get_tool_symbol()

Features

Returns the name of the currently active tool.

Return

Value	Data Types	Description
ret	str	Activated tool name

Example

```
ret = get_tool_symbol()
```

3.4.25 get_user_coord_cnt()

Features

Returns the number of currently set user coordinate systems.

Return

Value	Data Types	Description
ret	int	Number of user coordinate systems set up

Example

```
ret = get_user_coord_cnt()
```

3.4.26 get_user_coord()

Features

Returns the currently set user coordinate system information.

Return

Value	Data Types	Description
ret	class.config_user_coordinate	All user coordinate system information

Class

class.config_user_coordinate

Field	Data Types	Description
obj	list	Object information for that class
target_ref	int	Reference Coordinate System base : 0 world : 2

target_pos	float[6]	Target coordinates
user_id	int	User id

Example

```
ret = get_user_coord()
print(ret.target_ref)
print(ret.user_id)
```

3.4.27 get_world_coord()

Features

Returns the currently set world coordinate system information.

Return

Value	Data Types	Description
ret	class.config_world_coordinate	About setting up the world coordinate system

Class

class.config_world_coordinate

Field	Data Types	Description
obj	list	Object information for that class
type	int	world2base: 0 base2ref: 1 world2ref: 2
pos	float[6]	Target location

Example

```
ret = get_world_coord()
```

```
print(ret.type)
print(ret.pos)
```

4 Other Settings Command

4.1 Tool/Workpiece Settings

4.1.1 get_workpiece_weight()

Features

This function measures and returns the weight of the workpiece.

> **Caution**
> In the Non-FTS A model, this function cannot be used.

Return

Value	Description
Positive value	Measured weight
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
weight = get_workpiece_weight()
```

Related commands

- reset_workpiece_weight()(p. 227)

4.1.2 reset_workpiece_weight()

Features

This function initializes the weight data of the material to initialize the algorithm before measuring the weight of the material.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
reset_workpiece_weight()
```

Related commands

- get_workpiece_weight() (p. 226)

4.1.3 set_workpiece_weight()

Definition

set_workpiece_weight(weight=0.0, cog=[0.0,0.0,0.0], cog_ref=DR_CUR_TCP, add_up=DR_REPLACE, start_time=None, transition_time=None)

Features

In addition to the tool weight/center of gravity at the end of the robot, set the weight/center of gravity of the work piece and other information. The weight and center of gravity of the entire payload is reflected by combining the set tool weight/center of gravity and the work piece's weight/center of gravity. It can be used in applications where the type of workpiece is frequently varied or the weight needs to be dynamically changed.

Caution

- When changing the set tool weight, the workpiece weight is initialized to 0.

Parameters

Parameter Name	Data Type	Default Value	Description
weight	int	0	Weight [kg]
cog	list(float[3])	[0, 0, 0]	Center of gravity of the workpiece (x, y, z) [mm]
cog_ref	int	DR_CUR_TCP	Reference coordinates of center of gravity position, DR_CUR_TCP : TCP coordinates, DR_FLANGE : FLANGE coordinates,
add_up	int	DR_REPLACE	DR_REPLACE(0): Replace workpiece DR_ADD(1): Add workpiece DR_REMOVE(2): Remove workpiece
start_time	float	None	Starting time of changing workpiece weight [sec]
transition_time	float	None	Transition time of changing workpiece weight [sec]

Note

- Workpiece weight cannot exceed the maximum payload weight (margin 10%) for each model. The same is true for the weight of total payload.
- The length (x, y, z) of the center of gravity of the workpiece cannot exceed 1000 mm. The same is true for the length of the center of gravity of the entire payload.
- It is possible to change the weight of the workpiece after the set time through start_time.
- Transition_time allows you to gradually change the weight of the workpiece through transition_time.

- When using the set_tool and set_workpiece_weight functions in succession, you must use the wait(transition_time) function as much as transition_time between them. Otherwise, there may be errors in the weight change.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Replace 1 kgf workpiece weight at the origin of the current TCP frame
set_workpiece_weight(1)
# Add 0.5 kgf workpiece weight at the origin of the current TCP frame. In total,
1.5kgf
set_workpiece_weight(0.5, add_up=DR_ADD)
# Remove 0.1 kgf workpiece weight at [0, 0, 10] in the current TCP frame. In total,
1.4kgf
set_workpiece_weight(0.1, [0, 0, 10], add_up=DR_REMOVE)
# Remove 0.1 kgf workpiece weight at [0, 0, 10] in the flange frame. In total, 1.3kgf
set_workpiece_weight(0.1, [0, 0, 10], DR_FLANGE, DR_REMOVE)
# Reset. And the weight transition is being occurred after 0.1 sec for 0.2 sec
set_workiece_weight(0, start_time=0.1, transition_time=0.2)
```

Related Commands

- set_tool()(p. 232)

4.1.4 set_tcp()

Definition

set_tcp(name)

Features

his function calls the name of the TCP registered in the Teach Pendant and sets it as the current TCP.

Parameter

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the TCP registered in the TP.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
set_tcp("tcp1")  # The TCP data registered as tcp1 in the TP is called and set to the
current TCP value.
P1 = posx(400,500,800,0,180,0)
movel(P1, vel=10, acc=20) # Moves the recognized center of the tool to the P1
position.
```

Related commands

- fkin()(p. 35)
- ikin()(p. 37)
- set_velx()(p. 50)
- movel()(p. 63)
- movec()(p. 72)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- amovel()(p. 102)
- amovec()(p. 108)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)

4.1.5 set_tool_shape()

Definition

set_tool_shape(name)

Features

This function activates the tool shape information of the entered name among the tool shape information registered in the Teach Pendant.

Parameters

Parameter Name	Data Type	Default Value	Description
name	string	-	Tool name registered in the Teach Pendant

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_tool_shape("tool_shape1")  # Activate the geometry of "tool_shape1".
```

Related commands

- set_tcp()(p. 230)

4.1.6 set_tool()

Definition

set_tool(name, start_time, transition_time)

Features

Teach Pendant>Workcell Manager>Activate the Tool Weight workcell item with the entered name from among the Tool Weight workcell items registered in the robot.

Tool weight can be changed after setting time(start_time) and during setting time(transition_time).

Parameters

Parameter Name	Data Type	Default Value	Description
name	string	-	The name of the tool weight registered in the Workcell Manager.
start_time	float	None	Tool weight is changed after setting time
transition_time	float	None	Tool weight is changed during setting time

Note

- When using the set_tool and set_workpiece_weight functions in succession, you must use the wait(transition_time) function as much as transition_time between them. Otherwise, there may be errors in the weight change.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_tool ("tool1",start_time=1,transition_time=2)
```

```
# After 1s and then aActivate the information of "tool1" registered in TP during 2s.
```

Related commands

- set_tcp()(p. 230)

4.1.7 set_tool_digital_output_type()

정의

set_tool_digital_output_type(port, output_type=None)

Features

Set of activated tool digital output's type in the Teach Pendant.

Parameters

Parameter Name	Data Type	Default Value	Description
port	integer	-	The port number of Tool Digital Output
output_type	integer	None	The type of Tool Digital Output • 0: PNP • 1: NPN

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_tool_digital_output_type (port=1, output_type=DR_OUTPUT_TYPE_PNP)
set_tool_digital_output_type (port=2, output_type=DR_OUTPUT_TYPE_NPN)
```

4.1.8 set_tool_digital_output_level()

Definition

set_tool_digital_output_level(lv=None)

Features

Set of activated tool digital output's level in the Teach Pendant.

Parameters

Parameter Name	Data Type	Default Value	Description
lv	integer	12	The level of Tool Digital Output • 0: 0v • 12: 12v • 24: 24v

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_tool_digital_output_type (ln=24)
```

4.1.9 get_tool_analog_input()

Definition

get_tool_analog_input(ch)

Features

Get of activated tool analog input's value in the Teach Pendant.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	integer	-	The channel number of Tool Analog Input • Range: 1-4

Return

Value	Description
(Current mode) 4.0 ~ 20.0 mA (Voltage mode) 0.0 ~ 10.0 V	Success

Value	Description
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
channel = 1
val = get_tool_analog_input (ch=channel)
if val < 0:
    tp_log("get_tool_analog_input error: " + str(val))
else:
    tp_log("get_tool_analog_input ch[" + str(channel) + "]: " + str(val))
```

4.1.10 set_mode_tool_analog_input()

Definition

set_mode_tool_analog_input(ch, mod)

Features

Set of activated tool analog input's mode in the Teach Pendant.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	integer	-	The channel number of Tool Analog Input • Range: 1-4
mod	integer	-	The mode of Tool Analog Input 0: Current mode 1: Voltage mode

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
TOOL_CURRENT = 0
TOOL_VOLTAGE = 1
channel = 1
mode = TOOL_VOLTAGE

val = set_mode_tool_analog_input (ch=channel, mod=mode)
if val < 0:
```

```
    tp_log("set_mode_tool_analog_input error: " + str(val))
else:
    val = get_tool_analog_input(ch=channel)
    if val < 0:
        tp_log("get_tool_analog_input error: " + str(val))
    else:
        tp_log("get_tool_analog_input ch[" + str(channel) + "]: " + str(val))
        if mode == TOOL_VOLTAGE:
            tp_log(" V")
        elif mode == TOOL_CURRENT:
            tp_log(" mA")
```

4.2 Control Mode Settings

4.2.1 set_singularity_handling()

Definition

set_singularity_handling(mode)

Features

Allows the user to select a response policy when a path deviation occurs due to a singularity in task motion. The mode can be set as follows

- Automatic avoidance mode(Default) : DR_AVOID
- Path first mode : DR_TASK_STOP
- Variable velocity mode : DR_VAR_VEL

The default setting is automatic avoidance mode, which reduces instability caused by singularity, but reduces path tracking accuracy. In case of path first setting, if there is possibility of instability due to singularity, a warning message is output after deceleration and then the corresponding task is terminated. In case of variable velocity mode setting, TCP velocity would be changed in singular region to reduce instability and maintain path tracking accuracy.

Parameters

Parameter Name	Data Type	Default Value	Description
mode	int	DR_AVOID	DR_AVOID : Automatic avoidance mode DR_TASK_STOP : Deceleration/ Warning/ Task termination DR_VAR_VEL : Variable velocity mode

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)
P3 = posx(400,500,200,0,180,0)
set_singularity_handling (DR_AVOID) # Automatic avoidance mode for singularity
movel(P1, vel=10, acc=20)
set_velx(30)
set_accx(60)
set_singularity_handling(DR_TASK_STOP) # Task motion path first
movel(P2)
set_singularity_handling(DR_VAR_VEL) # Variable velocity mode for singularity
movel(P3)
```

Related commands

- movel()(p. 63)
- movec()(p. 72)
- movesx()(p. 81)
- moveb()(p. 84)
- move_spiral()(p. 89)
- amovel()(p. 102)
- amovec()(p. 108)

- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)

4.2.2 set_singular_handling_force()

Definition

set_singular_handling_force(mode)

Features

The program is terminated by default through error processing when compliance or force control are used within the singularity area. It is possible to ignore error processing within the singularity area by changing the Mode setting.

- Error Processing : DR_SINGULARITY_ERROR
- Ignore Error Processing : DR_SINGULARITY_IGNORE

> **Caution**
> Compliance and force control within the singularity area are not recommended. The force estimate in a particular direction can be inaccurate.

Parameters

Parameter Name	Data Type	Default Value	Description
mode	int	DR_SINGULARITY_ERROR	DR_SINGULARITY_ERROR : Error processing DR_SINGULARITY_IGNORE : Ignore error processing

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_ref_coord(DR_BASE)
P0 = posj(0, 0, 90, 0, 90, 0)
movej(P0,vel=30,acc=60)

#Ignoring error when entering singularity
set_singular_handling_force(DR_SINGULARITY_IGNORE)

task_compliance_ctrl()
set_stiffnessx([500, 500, 500, 100, 100, 100], time=0.5)
fd = [0, 0, 30, 0, 0, 0]
fctrl_dir= [0, 0, 1, 0, 0, 0]
set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL)
release_compliance_ctrl()
```

Related Commands

- task_compliance_ctrl()(p. 250)
- set_stiffnessx()(p. 251)
- set_desired_force()(p. 253)
- release_compliance_ctrl()(p. 249)

4.2.3 set_palletizing_mode()

Definition

set_palletizing_mode(mode)

Features

During palletizing application motion, path tracking and velocity can be maintained around the wrist singularity point using this function. there is no instability in wrist singular region when B in motion command is set to 0deg or 180deg.

- Deactivate mode : DR_OFF
- Activate mode : DR_ON

Caution

- Setting tool orientation, B must be set to 0deg or 180deg when setting tcp information. If this condition don't be satisfied, Error is occurred when using this function.
- Normally velocity don't be changed in this mode. But if current joint velocity exceed allowable max joint velocity, velocity can be reduced automatically.
- In case of H series model, Rx, Ry moment control is restricted and external moment value of each Rx, Ry direction is 0.

Parameters

Parameter Name	Data Type	Default Value	Description
mode	int	DR_ON	DR_OFF : Deactivate mode DR_ON : Activate mode

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_singularity_handling(DR_VAR_VEL)

movej(posj(0,0,90,0,90,0),vel=30,acc=60)

set_palletizing_mode(DR_ON)
movel(posx(559,34.5,-400,45,180,45),vel=500,acc=1000)
set_palletizing_mode(DR_OFF)
```

Related Commands

- set_singularity_handling()(p. 239)

4.2.4 set_motion_end()

Definition

set_motion_end(mode)

Features

This command sets whether to operate the function to check the stop status of the robot after motion is completed. Stop time between consecutive motions decreases if it set to deactivate mode (DR_CHECK_OFF) and can be used for purposes of decreasing the overall work time. It is recommended to set it to DR_CHECK_ON when the tool is heavy and an accurate stop position is required for motion commands driven with high acceleration.

Caution

- It is not possible to change the mode, during the blending movements between consecutive motions without stopping.
- In the case of continuous motion that does not require a stop state, using motion blending is more effective in reducing tact time.
- After the program is finished, it is initialized to the default value again

Parameters

Parameter Name	Data Type	Default Value	Description
mode	int	DR_CHECK_ON	DR_CHECK_OFF(0) : Deactivate mode DR_CHECK_ON(1) : Activate mode

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_motion_end(DR_CHECK_OFF)

movej(posj(0,0,90,0,90,0) ,vel=30,acc=60,mod = DR_MV_MOD_ABS )
while 1:
    movej(posj(0,0,10,0,10,0) ,vel=30,acc=60,,mod = DR_MV_MOD_REL )
    movej(posj(0,0,-10,0,-10,0) ,vel=30,acc=60,,mod = DR_MV_MOD_REL )
```

Related Commands

- movel()(p. 63)
- movec()(p. 72)
- movesx()(p. 81)

- moveb()(p. 84)
- move_spiral()(p. 89)
- amovel()(p. 102)
- amovec()(p. 108)
- amovesx()(p. 115)
- amoveb()(p. 118)
- amove_spiral()(p. 122)

4.3 LED Settings

4.3.1 set_state_led_color()

Definition

set_state_led_color(r,g,b)

Features

While the task is running, the robot LED will be set to the color specified by the user.

Parameters

Parameter Name	Data Type	Default Value	Description
r	int	0	1: Red color on 0: Red color off
g	int	0	1: Green color on 0: Green color off
b	int	0	1: Blue color on 0: Blue color off

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
#white color
set_state_led_color(1,1,1)

#cyan color
set_state_led_color(0,1,1)

#magenta color
set_state_led_color(1,0,1)

#blue color
set_state_led_color(0,0,1)

#yellow color
set_state_led_color(1,1,0)

#green color
set_state_led_color(0,1,0)

#red color
set_state_led_color(1,0,0)

#led off
set_state_led_color(0,0,0)
```

4.3.2 set_state_led_off()

Features

While the task is running, the robot LED is turned off.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
#led off
set_state_led_off()
```

4.3.3 state_led_reset()

Features

During task play, when `set_state_led_color()` or `set_state_led_off()` is used in DRL, the LED state will be restored afterwards.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
#reset led state
state_led_reset()
```

5 Force/Stiffness Control and Other User-Friendly Features

5.1 Force/Compliance Control

5.1.1 release_compliance_ctrl()

Features

This function terminates compliance control and begins position control at the current position.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
task_compliance_ctrl()
set_stiffnessx([100, 100, 300, 100, 100, 100])
release_compliance_ctrl()
```

Related commands

- task_compliance_ctrl()(p. 250)
- set_stiffnessx()(p. 251)

5.1.2 task_compliance_ctrl()

Definition

task_compliance_ctrl(stx, time)

Features

This function begins task compliance control based on the preset reference coordinate system.

> **Caution**
> In non-FTS models (A0509, A0912, A0509F, A0912F, E0509), compliance control is only possible in the translation direction, and the control error may be large.

> **Caution**
> If the command is used in a simulation environment without a robot, it may not operate normally.

Parameters

Parameter Name	Data Type	Default Value	Description
stx	float[6]	[3000, 3000, 3000, 200, 200, 200]	Three translational stiffnesses Three rotational stiffnesses
time	float	0	Stiffness varying time [sec] Range: 0 - 1.0 * Linear transition during the specified time

> **Caution**
> In the Non-FTS A & E model, the data type of the stx parameter can be either float[6] or float[3] (rotational stiffness is automatically set to the default value)

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
task_compliance_ctrl()      # Begins with the default stiffness
set_stiffnessx([500, 500, 500, 100, 100, 100], time=0.5)
# Switches to the user-defined stiffness for 0.5 sec.
release_compliance_ctrl()

task_compliance_ctrl([500, 500, 500, 100, 100, 100])
# Begins with the user-defined stiffness.
release_compliance_ctrl()
```

Related commands

- set_stiffnessx()(p. 251)
- release_compliance_ctrl()(p. 249)

5.1.3 set_stiffnessx()

Definition

set_stiffnessx(stx, time)

Features

This function sets the stiffness value based on the global coordinate (refer to set_ref_coord()). The stiffness linearly changes for a given time from the current stiffness or default value to STX.The user-defined ranges of the translational stiffness and rotational stiffness are 0-20000N/m and 0-400Nm/rad, respectively.

User-defined ranges of stiffness

- M/H/A/P Series: Translation(0~20000N/m), Rotation(0~1000Nm/rad)
- Non-FTS A/E Series: Translation(0~20000N/m)

Caution
In the Non-FTS A & E model, the data type of the stx parameter can be either float[6] or float[3] (rotational stiffness is automatically set to the default value)

Caution
If the command is used in a simulation environment without a robot, it may not operate normally.

Parameters

Parameter Name	Data Type	Default Value	Description
stx	float[6]	[500, 500, 500, 100, 100, 100]	Three translational stiffnesses Three rotational stiffnesses
time	float	0	Stiffness varying time [sec] Range: 0 - 1.0 * Linear transition during the specified time

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_ref_coord(DR_WORLD)    # Global coordinate is the world coordinate
x0 = posx(0, 0, 90, 0, 90, 0)
movej(x0)
task_compliance_ctrl()
stx = [1, 2, 3, 4, 5, 6]
set_stiffnessx(stx)  # Set the stiffness value based on the global coordinate(world
coordinate)
release_compliance_ctrl()
```

Related commands

- task_compliance_ctrl()(p. 250)
- release_compliance_ctrl()(p. 249)

5.1.4 set_desired_force()

Definition

set_desired_force(fd, dir, time, mod)

Features

This function define the s target force, direction, translation time, and mode for force control based on the global coordinate.

Caution
In non-FTS models (A0509, A0912, A0509F, A0912F, E0509), force control is only possible in the translation direction, and the control error may be large.

Caution
If the command is used in a simulation environment without a robot, it may not operate normally.

Parameters

Parameter Name	Data Type	Default Value	Description
fd	float[6]	[0, 0, 0, 0, 0, 0]	Three translational target forces Three rotational target moments

Parameter Name	Data Type	Default Value	Description
dir	int[6]	[0, 0, 0, 0, 0, 0]	Force control in the corresponding direction if 1 Control compliance of corresponding direction if value is 0
time	float	0	Transition time of target force to take effect [sec] Range: 0 - 1.0
mod	int	DR_FC_MOD_ABS	DR_FC_MOD_ABS: Force control with absolute value DR_FC_MOD_REL: force control with relative value to initial state (the instance when this function is called)

Caution
In the Non-FTS A & E model, the data types of fd and dir parameters can be either float[6] or float[3]. (Rotational parameters are automatically set to the default values)

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Note

- The value of external force refers to the sensor measurement at terminating the force control (control mode transition to compliance control) by the command release_force().
  Therefore, the variation in external force can occur if the option mod=DR_FC_MOD_REL is applied.
- Tool weight and external force value refer to the sensor measurement regardless of the setting for 'mod'

Caution

To retain the accuracy in force control, it is recommended to start force control with setting mod=DR_FC_MOD_REL near the contact point.

Example

```
# Example # 1
# Executed in the global coordinate(tool coordinate)
# Zero force control in the z-axis direction of the tool, moment control in the z-
axis direction of the tool, and compliance control in the other directions
# Force control with the relative value to the sensor measurement at starting the
force control

set_ref_coord(DR_TOOL)
x0 = posx(0, 0, 90, 0, 90, 0)
movej(x0)
task_compliance_ctrl(stx=[500, 500, 500, 100, 100, 100])
fd = [0, 0, 0, 0, 0, 10]
fctrl_dir= [0, 0, 1, 0, 0, 1]
set_desired_force(fd, dir=fctrl_dir, mod=DR_FC_MOD_REL)

# Example #2
# 1. Move to initial posj: [J1, J2, J3, J4, J5, J6] = [0, 0, 90, 0, 90, 0]
# 2. Approach to the position to start force control: move -100mm along Base-z
direction
# 3. Start force control : apply -20N force along Base-z direction
# 4. Force & compliance control after detecting external force : while maintaining
-20N force along Base-z direction (force control), move 200mm along Base-y direction.
# 5. Retract 150mm in Base-z direction and move to initial posj

# 1. Move to initial posj
q0 = posj(0.0, 0.0, 90.0, 0.0, 90.0, 0.0)
set_velj(30.0)
set_accj(60.0)
movej(q0)

# 2. Approach to the position to start force control
set_velx(75.0)
set_accx(100.0)
delta_approach = [0.0, 0.0, -100.0, 0.0, 0.0, 0.0]
movel(delta_approach, mod=DR_MV_MOD_REL)
```

```
# 3. Start force control (apply -20N force along Base-z direction)
k_d = [3000.0, 3000.0, 3000.0, 200.0, 200.0, 200.0]
task_compliance_ctrl(k_d)
force_desired = 20.0
f_d = [0.0, 0.0, -force_desired, 0.0, 0.0, 0.0]
f_dir = [0, 0, 1, 0, 0, 0]
set_desired_force(f_d, f_dir)

# 4. Force & compliance control after detecting external force
force_check = 20.0
force_condition = check_force_condition(DR_AXIS_Z, max=force_check)
while (force_condition):
    force_condition = check_force_condition(DR_AXIS_Z, max=force_check)
    if force_condition == 0:
        break
delta_motion = [0.0, 200.0, 0.0, 0.0, 0.0, 0.0]
movel(delta_motion, mod=DR_MV_MOD_REL)

# 5. Retract 150mm in Base-z direction and move to initial posj
release_force()
wait(0.5)
delta_retract = [0.0, 0.0, 150.0, 0.0, 0.0, 0.0]
release_compliance_ctrl()
movel(delta_retract, mod=DR_MV_MOD_REL)
movej(q0)
```

Related commands

- release_force()(p. 256)
- task_compliance_ctrl()(p. 250)
- set_stiffnessx()(p. 251)
- release_compliance_ctrl()(p. 249)

5.1.5 release_force()

Definition

release_force(time=0)

Features

This function reduces the force control target value to 0 through the time value and returns the task space to adaptive control.

Parameters

Parameter Name	Data Type	Default Value	Description
time	float	0	Time needed to reduce the force Range: 0 - 1.0

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
j0 = posj(0, 0, 90, 0, 90, 0)
x0 = posx(0, 0, 0, 0, 0, 0)
x1 = posx(0, 500, 700, 0, 180, 0)
x2 = posx(300, 100, 700, 0, 180, 0)
x3 = posx(300, 100, 500, 0, 180, 0)
set_velx(100,20)
set_accx(100,20)
movej(j0, vel=10, acc=10)
movel(x2)
task_compliance_ctrl(stx = [500, 500, 500, 100, 100, 100])
fd = [0, 0, 0, 0, 0, 10]
fctrl_dir= [0, 0, 1, 0, 0, 1]
set_desired_force(fd, dir=fctrl_dir, time=1.0)
movel(x3, v=10)
release_force(0.5)
release_compliance_ctrl()
```

Related commands

- set_desired_force()(p. 253)
- task_compliance_ctrl()(p. 250)
- set_stiffnessx()(p. 251)
- release_compliance_ctrl()(p. 249)

5.1.6 get_force_control_state()

Features

It monitors the state of compliance and force control.

> **Note**
> For the Non-FTS A/E series, during the compliance and force control state, the rotation directions (rx, ry, rz) operate under the default compliance control settings.

Return

Value	Description
singularity	Risk : 0 < 1 < 2 0 : Safe section 1 : Stage 1 risk section of the singularity area 2 : Stage 2 risk section of the singularity area
mode	Information of 6 modes x, y, z, rx, ry and rz (sequential) 0 : Compliance control 1 : Force control 2 : None
Stx	6 in the order of x, y, z, rx, ry and rz Information of set target stiffness
fd	6 in the order of x, y, z, rx, ry and rz Information of set target force

Value	Description
ref	Information of set target force 0 : Base coordinate 1 : Tool coordinate 2 : World coordinate 101 ~ 120 : User coordinate

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
set_ref_coord(DR_BASE)
P0 = posj(0, 0, 90, 0, 90, 0)
movej(P0,vel=30,acc=60)

task_compliance_ctrl()
set_stiffnessx([500, 500, 500, 100, 100, 100], time=0.5)

while True:
    [singularity, mod, stx, fd, ref]=get_force_control_state()
tp_log("s={0}, m={1}, k={2}, f={3}, r={4}".format(singularity,mod,stx,fd,ref))
    wait(0.5)
release_compliance_ctrl()
```

Related Commands

- set_singular_handling_force()(p. 241)
- task_compliance_ctrl()(p. 250)
- set_stiffnessx()(p. 251)
- set_desired_force()(p. 253)
- release_compliance_ctrl()(p. 249)

5.1.7 set_damping_factor()

Definition

set_damping_factor(damping_factor, time)

Features

In force control, This function sets the damping factor based on the global coordinate (refer to set_ref_coord())

Caution
In non-FTS models (A0509, A0912, A0509F, A0912F, E0509), compliance control is only possible in the translation direction, and the control error may be large.

Parameters

Parameter Name	Data Type	Default Value	Description
damping_factor	float[6]	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	(Translation x, y, z), (Rotation x, y, z) Range: 0.1 - 2.0
time	float	0	Damping factor varying time [sec] Range: 0 - 1.0 * Linear transition during the specified time

Caution
In the Non-FTS A & E model, the data type of the damping_factorparameter can be either float[6] or float[3] (rotational parameter is automatically set to the default value)

Note
- When the damping factor is set to the default value (1.0), it operates with the default damping value set according to the robot model.
- When the damping factor is set, a value proportional to the default damping value is applied during force control.
- The damping factor is applied only during force control. It is applied as a default value for compliance control.
- This function is supported only for the M series. It cannot be used for A series and H series.

Warning
- If the damping factor is set to less than 1.0, the robot may vibrate depending on the contact surface. Please use it carefully.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# 1. Move to initial posj
q0 = posj(0.0, 0.0, 90.0, 0.0, 90.0, 0.0)
set_velj(30.0)
set_accj(60.0)
movej(q0)

# 2. Start force control (apply -20N force along Base-z direction)
k_d = [3000.0, 3000.0, 3000.0, 200.0, 200.0, 200.0]
task_compliance_ctrl(k_d)
force_desired = 20.0
f_d = [0.0, 0.0, -force_desired, 0.0, 0.0, 0.0]
f_dir = [0, 0, 1, 0, 0, 0]
d_f = [1.0, 1.0, 0.8, 1.0, 1.0, 1.0]
set_desired_force(f_d, f_dir)
set_damping_factor(d_f)
wait(2.0)
release_force()
wait(0.5)
release_compliance_ctrl()
```

Related commands

- set_force_factor()(p. 262)
- set_desired_force()(p. 253)
- release_force()(p. 256)

5.1.8 set_force_factor()

Definition

set_force_factor(force_factor, time)

Features

In force control, This function sets the force factor based on the global coordinate (refer to set_ref_coord())

Caution
In non-FTS models (A0509, A0912, A0509F, A0912F, E0509), compliance control is only possible in the translation direction, and the control error may be large.

Parameters

Parameter Name	Data Type	Default Value	Description
force_factor	float[6]	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	(Translation x, y, z), (Rotation x, y, z) Range: 0.1 - 2.0
time	float	0	Force factor varying time [sec] Range: 0 - 1.0 * Linear transition during the specified time

Caution
In the Non-FTS A & E model, the data type of the force_factor parameter can be either float[6] or float[3] (rotational parameter is automatically set to the default value)

Note
- When the force factor is set to the default value (1.0), it operates with the default damping value set according to the model.
- Setting the force factor below 1.0 reduces the setting time to follow the target force value, but may increase vibration.

- Setting the force factor above 1.0 may reduce vibration but increase the setting time to follow the target force value.
- The force factor setting is applied only during force control. It is applied as a default value for compliance control.
- This function is supported only for the M series. It cannot be used for A series and H series.

Warning

- If the force factor is set to less than 1.0, the robot may vibrate depending on the contact surface. Please use it carefully.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# 1. Move to initial posj
q0 = posj(0.0, 0.0, 90.0, 0.0, 90.0, 0.0)
set_velj(30.0)
set_accj(60.0)
movej(q0)

# 2. Start force control (apply -20N force along Base-z direction)
k_d = [3000.0, 3000.0, 3000.0, 200.0, 200.0, 200.0]
task_compliance_ctrl(k_d)
force_desired = 20.0
f_d = [0.0, 0.0, -force_desired, 0.0, 0.0, 0.0]
```

```
f_dir = [0, 0, 1, 0, 0, 0]
d_f = [1.0, 1.0, 0.8, 1.0, 1.0, 1.0]
f_f = [1.0, 1.0, 1.2, 1.0, 1.0, 1.0]
set_desired_force(f_d, f_dir)
set_damping_factor(d_f)
set_force_factor(f_f)
wait(2.0)
release_force()
wait(0.5)
release_compliance_ctrl()
```

Related commands

- set_damping_factor()(p. 260)
- set_desired_force()(p. 253)
- release_force()(p. 256)

5.1.9 set_external_force_reset()

Definition

set_external_force_reset(mode, offset)

Features

This function resets the external force sensor mounted on the robot. This can improve the accuracy of force/compliance control and start force/compliance control even when there is an external force due to contact. Since reset is performed based on the external force sensor mounted on the robot, the offset value is set based on the TCP force/torque for the As series, and the offset value is set based on the joint torque for other models.

Parameters

Parameter Name	Data Type	Default Value	Description
mode	int	1	Reset mode • 0: Change offset to [0.0, 0.0, 0.0, 0.0, 0.0, 0.0] • 1: Reset external force sensor. Set current measured external force as offset. (default) • 2: Set to the entered offset value offset

Parameter Name	Data Type	Default Value	Description
offset	float[6]	[0.0, 0.0, 0.0, 0.0, 0.0, 0.0]	Offset of external force sensor • As series: TCP force/torque (order: Fx, Fy, Fz, Tx, Ty, Tz), (unit: N, Nm) • Other series: Joint torque (order: J1~ 6), (unit: Nm)

Note

- You can reset the external force error using the set_external_force_reset() command regardless of whether the force/compliance control is in operation.
- The effect of the set_external_force_reset() command does not affect the robot's safety functions (collision detection, TCP SLF, etc.).
- Even if the tool weight setting is incorrect, you can reset the external force by using the set_external_force_reset() command.
- If force/compliance control cannot be started due to external force caused by contact, you can start force/compliance control by initializing the external force using the set_external_force_reset() command.
- After using the set_external_force_reset() command, errors may accumulate again when the robot moves, so for precise force control, use the set_external_force_reset command at a location as close as possible before contact.
- It affects the force status of the check_force_condition() command, and also affects the output values obtained through the get_external_torque() and get_tool_force() commands.
- This feature is useful for M/H/P/As series equipped with additional sensors such as JTS and FTS. Caution is required when using A/E series that use motor current sensor (CS).

Warning

- Please keep in mind that when using the set_external_force_reset() command when there is an external force due to contact, the offset includes the contact force, so the actual external force measurement is distorted.
- Also, if you use the set_external_force_reset() command repeatedly when there is contact, the problem may overlap, so be careful not to use it repeatedly.
- To avoid the above cases, please use the set_external_force_reset() command in a non-contact state if possible.

Return

Value	Description
offset	Current offset of external force sensor • As series: TCP force/torque (order: Fx, Fy, Fz, Tx, Ty, Tz), (unit: N, Nm) • Other series: Joint torque (order: J1~ 6), (unit: Nm)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Example
# 1. Move to starting position: [J1, J2, J3, J4, J5, J6] = [0, 0, 90, 0, 90, 0]
# 2. Start compliance control: Reset external force before starting compliance
control(Added wait(2.0) because if vibration remains after motion ends, it can cause
errors during external force reset.).
# 3. Approaching the target location of the force control
# 4. Start force control: Reset external force before starting force control to
improve force control accuracy.
# 5. Retract after force/compliance control ends: After compliance control ends,
restore external force offset to 0.

# 1. Move to starting position
q0 = posj(0.0, 0.0, 90.0, 0.0, 90.0, 0.0)
set_velj(30.0)
set_accj(60.0)
movej(q0)

# 2. Start compliance control
wait(2.0)
set_external_force_reset() # Reset external force
k_d = [3000.0, 3000.0, 3000.0, 200.0, 200.0, 200.0]
task_compliance_ctrl(k_d)
```

```
wait(2.0)

# 3. Approaching the target location of the force control
set_velx(75.0)
set_accx(100.0)
delta_approach = [0.0, 0.0, -100.0, 0.0, 0.0, 0.0]
movel(delta_approach, mod=DR_MV_MOD_REL)

# 4. Start force control
wait(2.0)
set_external_force_reset() # Reset external force
force_desired = 20.0
f_d = [0.0, 0.0, -force_desired, 0.0, 0.0, 0.0]
f_dir = [0, 0, 1, 0, 0, 0]
set_desired_force(f_d, f_dir)
wait(10.0)

# 5. Retract after force/compliance control ends
release_force(0.5)
delta_retract = [0.0, 0.0, 150.0, 0.0, 0.0, 0.0]
movel(delta_retract, mod=DR_MV_MOD_REL)
release_compliance_ctrl() # Restore external force offset to 0.
set_external_force_reset(0)
movej(q0)
```

Related commands

- task_compliance_ctrl()(p. 250)
- set_desired_force()(p. 253)
- check_force_condition()(p. 289)
- get_external_torque()(p. 172)
- get_tool_force()(p. 173)

5.2 User-friendly Functions

5.2.1 parallel_axis()

- parallel_axis(x1, x2, x3, axis, ref)(p. 267)
- parallel_axis(vect, axis, ref)(p. 269)

parallel_axis(x1, x2, x3, axis, ref)

Features

This function matches the normal vector of the plane consists of points(x1, x2, x3) based on the ref coordinate(refer to get_normal(x1, x2, x3)) and the designated axis of the tool frame. The current position is maintained as the TCP position of the robot.

Parameters

Parameter Name	Data Type	Default Value	Description
x1	posx list (float[6])	-	posx or position list
x2	posx list (float[6])	-	posx or position list
x3	posx list (float[6])	-	posx or position list
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis
ref	int	DR_BASE	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • user coordinate : user difined

Note

In the case of the P series,

- The z values of Input x1, x2 and x3 must all be the same.
- Input axis can only use DR_AXIS_Z, and input ref can only use DR_BASE.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
x0 = posj(0, 0, 90, 0, 90, 0)
movej(x0, 60, 30)
x1 = posx(0, 500, 700, 30, 0, 90)
x2 = posx(500, 0, 700, 0, 0, 45)
x3 = posx(300, 100, 500, 45, 0, 45)
parallel_axis(x1, x2, x3, DR_AXIS_X, DR_WORLD)
# match the tool x axis and the normal vector of the plane consists of
points(x1,x2,x3) # based on the world coordinate

#In the case of the P-series
x0 = posj(0, 0, 90, 0, 45, 0)
movej(x0, 60, 30)
x1 = posx(0, 500, 700, 30, 0, 90)
x2 = posx(500, 0, 700, 0, 0, 45)
x3 = posx(300, 100, 700, 45, 0, 45)
parallel_axis(x1, x2, x3, DR_AXIS_Z, DR_BASE)
```

Related commands

- get_normal()(p. 396)
- align_axis()(p. 271)

parallel_axis(vect, axis, ref)

Features

This function matches the given vect direction based on the ref coordinate and the designated axis of the tool frame. The current position is maintained as the TCP position of the robot.

Parameters

Parameter Name	Data Type	Default Value	Description
vect	list (float[3])	-	vector
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis
ref	int	DR_BASE	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • user coordinate: user defined

Note

In the case of the P series,

- Input vect can only be applied to vectors about the z-axis ([0,0,±z]). (z != 0)
- Input axis can only use DR_AXIS_Z, and input ref can only use DR_BASE.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
x0 = posj(0, 0, 90, 0, 90, 0)
movej(x0, v=60, a=30)
parallel_axis([1000, 700, 300], DR_AXIS_X, DR_WORLD)
# match the tool x axis and the vector([1000,700,300]) based on the world coordinate

#In the case of the P-series
x0 = posj(0, 0, 90, 0, 45, 0)
movej(x0, v=60, a=30)
parallel_axis([0, 0, 300], DR_AXIS_Z, DR_BASE)
```

Related commands

- movej()(p. 58)
- align_axis()(p. 271)

5.2.2 align_axis()

- align_axis(x1, x2, x3, pos, axis, ref)(p. 271)
- align_axis(vect, pos, axis, ref)(p. 273)

align_axis(x1, x2, x3, pos, axis, ref)

Features

This function matches the normal vector of the plane consists of points(x1, x2, x3) based on the ref coordinate(refer to get_normal(x1, x2, x3)) and the designated axis of the tool frame. The robot TCP moves to the pos position.

Parameters

Parameter Name	Data Type	Default Value	Description
x1	posx list (float[6])	-	posx or position list
x2	posx list (float[6])	-	posx or position list

Parameter Name	Data Type	Default Value	Description
x3	posx list (float[6])	-	posx or position list
pos	posx list (float[6])	-	posx or position list
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis
ref	int	DR_BASE	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • user coordinate: user defined

Note

In the case of the P series,

- The z values of Input x1, x2 and x3 must all be the same.
- Input axis can only use DR_AXIS_Z, and input ref can only use DR_BASE.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
p0 = posj(0,0,45,0,90,0)
movej(p0, v=30, a=30)

x1 = posx(0, 500, 700, 30, 0, 0)
x2 = posx(500, 0, 700, 0, 0, 0)
x3 = posx(300, 100, 500, 0, 0, 0)
pos = posx(400, 400, 500, 0, 0, 0)
align_axis(x1, x2, x3, pos, DR_AXIS_X, DR_BASE)
# match the tool x axis and the normal vector in the plane consists of points(x1, x2,
# x3) based on the base coordinate

#In the case of the P-series
p0 = posj(0,0,45,0,90,0)
movej(p0, v=30, a=30)

x1 = posx(0, 500, 700, 30, 0, 0)
x2 = posx(500, 0, 700, 0, 0, 0)
x3 = posx(300, 100, 700, 0, 0, 0)
pos = posx(400, 400, 600, 0, 0, 0)
align_axis(x1, x2, x3, pos, DR_AXIS_Z, DR_BASE)
```

Related commands

- movej()(p. 58)
- get_normal()(p. 396)
- parallel_axis()(p. 267)

align_axis(vect, pos, axis, ref)

Features

This function matches the given vect direction based on the ref coordinate and the designated axis of the tool frame. The robot TCP moves to the pos position.

Parameters

Parameter Name	Data Type	Default Value	Description
vect	list (float[3])	-	vector
pos	posx list (float[6])	-	posx or position list
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis
ref	int	DR_BASE	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate • user coordinate: user defined

Note

In the case of the P series,

- Input vect can only be applied to vectors about the z-axis ([0,0,±z]). (z != 0)
- Input axis can only use DR_AXIS_Z, and input ref can only use DR_BASE.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
p0 = posj(0,0,45,0,90,0)
movej(p0, v=30, a=30)

vect = [10, 20, 30]
pos = posx(100, 500, 700, 45, 0, 0)
align_axis(vect, pos, DR_AXIS_X)

#In the case of the P-series
p0 = posj(0,0,45,0,90,0)
movej(p0, v=30, a=30)

vect = [0,0,-10]
pos = posx(100, 500, 700, 45, 0, 0)
align_axis(vect, pos, DR_AXIS_Z, DR_BASE)
```

Related commands

- movej()(p. 58)
- parallel_axis()(p. 267)

5.2.3 is_done_bolt_tightening()

Definition

is_done_bolt_tightening(m=0, timeout=0, axis=None)

Features

This function monitors the tightening torque of the tool and returns True if the set torque (m) is reached within the given time and False if the given time has passed.

Parameters

Parameter Name	Data Type	Default Value	Description
m	float	0	Target torque
timeout	float	0	Monitoring duration [sec]

Parameter Name	Data Type	Default Value	Description
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
p0 = posj(0,0,90,0,90,0)
movej(p0, v=30, a=30)

task_compliance_ctrl()
xd = posx(559, 34.5, 651.5, 0, 180.0, 60)
amovel(xd, vel=50, acc=50) # Bolt tightening motion

res = is_done_bolt_tightening(10, 5, DR_AXIS_Z)
    # Returns True if the tightening torque of 10Nm is reached within 5 seconds.
    # Returns False otherwise.
if res==True:
  # some action comes here for the case that bolt tightening is done
  x=1
else:
  # some action comes here for the case that it fails
  x=2
```

Related commands

- movej()(p. 58)
- amovel()(p. 102)

5.2.4 calc_coord()

Definition

calc_coord(x1, x2, x3, x4, ref, mod, ori_type_out)

Features

This function returns a new user cartesian coordinate system by using up to 4 input poses ([x1]~[x4]), input mode [mod] and the reference coordinate system [ref]. The input mode is only valid when the number of input robot poses is 2.

- In the case that the number of input poses is 1, the coordinate system is calculated using the position and orientation of x1.
- In the case that the number of input poses is 2 and the input mode is 0, X-axis is defined by the direction from x1 to x2, and Z-axis is defined by the projection of the current Tool-Z direction onto the plane orthogonal to the x-axis. The origin is the position of x1.
- In the case that the number of input poses is 2 and the input mode is 1, X-axis is defined by the direction from x1 to x2, and Z-axis is defined by the projection of the z direction of x1 onto the plane orthogonal to the X-axis. The origin is the position of x1.
- In the case that the number of input poses is 3, X-axis is defined by the direction from x1 to x2. If a vector v is the direction from x1 to x3, Z-axis is defined by the cross product of X-axis and v (X-axis cross v). The origin is the position of x1.
- In the case that the number of input poses is 4, the definition of axes is identical to the case that the number of input poses is 3, but the origin is the position of x4

Parameters

Parameter Name	Data Type	Default Value	Description
x1, x2, x3, x4	posx list (float[6])	-	Posx or position list
ref	int	-	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate

Parameter Name	Data Type	Default Value	Description
mod	int	-	input mode (only valid when the number of input poses is 2) • 0: defining z-axis based on the current Tool-z direction • 1: defining z-axis based on the z direction of x1
ori_type_out	int	None	orientation type • None: Follows the orientation type of pos input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	Successful coordinate calculation Position information of the calculated coordinate

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
pos1 = posx(500, 30, 500, 0, 0, 0)
pos2 = posx(400, 30, 500, 0, 0, 0)
```

```
pos3 = posx(500, 30, 600, 45, 180, 45)
pos4 = posx(500, -30, 600, 0, 180, 0)
pose_user1 = calc_coord(pos1, ref=DR_BASE, mod=0, ori_type_out=DR_ELER_ZYX)
pose_user21 = calc_coord(pos1, pos2, ref=DR_WORLD, mod=0)
%% Define z-axis based on the Tool-z direction.
pose_user22 = calc_coord(pos1, pos2, ref=DR_BASE, mod=1)
%% Define z-axis based on the z direction of pos1
pose_user3 = calc_coord(pos1, pos2, pos3, ref=DR_BASE, mod=0)
pose_user4 = calc_coord(pos1, pos2, pos3, pos4, ref=DR_WORLD, mod=0)
ucart1 = set_user_cart_coord(pose_user1, ref=DR_BASE)
ucart2 = set_user_cart_coord(pose_user21, ref=DR_WORLD)
```

Related commands

- set_user_cart_coord()(p. 279)

5.2.5 set_user_cart_coord()

set_user_cart_coord(pos, ref)

Features

This function set a new user cartesian coordinate system using input pose [pos] and reference coordinate system[ref]. Up to 100 user coordinate systems can be set including the coordinate systems set within Workcell Item. Since the coordinate system set by this function is removed when the program is terminated, setting new coordinate systems within Workcell Item is recommended for maintaining the coordinate information.

Parameters

Parameter Name	Data Type	Default Value	Description
pos	posx list (float[6])	-	coordinate information (position and orientation)
ref	int	-	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate

Return

Value	Description
Positive integer	Successful coordinate setting Set coordinate ID (101 - 200)
-1	Failed coordinate setting

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
pos1 = posx(10, 20, 30, 0, 0, 0)
pos2 = posx(30, 50, 70, 45, 180, 45)
user_id1 = set_user_cart_coord(pos1, ref=DR_BASE)
user_id2 = set_user_cart_coord(pos2, ref=DR_WORLD)
```

Related commands

- set_ref_coord()(p. 54)

set_user_cart_coord(x1, x2, x3, pos, ref)

Features

This function sets a new user cartesian coordinate system using [x1], [x2], and [x3] based on ref coordinate system[ref]. Creates a user coordinate system with ux, uy, and uz as the vector for each axis and origin position is the position of [pos] based on [ref]. [1)]ux is defined as the unit vector of x1x2 , uz is defined as the unit vector defined by the cross product of x1x2 and x1x3 (x1x2 cross x1x3). uy is can be determined by right hand rule (uz cross ux). Up to 100 user coordinate systems can be set including the coordinate systems set within Workcell

Item. Since the coordinate system set by this function is removed when the program is terminated, setting new coordinate systems within Workcell Item is recommended for maintaining the coordinate information.

1. In software versions lower than M2.0.2, ux is used as the unit vector of x2x1

Parameters

Parameter Name	Data Type	Default Value	Description
x1	Posx list (float[6])	-	posx or position list
x2	Posx list (float[6])	-	posx or position list
x3	Posx list (float[6])	-	posx or position list
pos	Posx list (float[6])	-	posx or position list
ref	int	DR_BASE	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate

Return

Value	Description
Positive integer	Successful coordinate setting Set coordinate ID (101 - 200)
-1	Failed coordinate setting

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
x1 = posx(0, 500, 700, 0, 0, 0) # Ignores the Euler angle.
x2 = posx(500, 0, 700, 0, 0, 0)
x3 = posx(300, 100, 500, 0, 0, 0)
x4 = posx(300, 110, 510, 0, 0, 0)
pos = posx(10, 20, 30, 0, 0, 0)
user_tc1 = set_user_cart_coord(x1, x2, x3, pos, ref=DR_BASE)
user_tc2 = set_user_cart_coord(x2, x3, x4, pos, ref=DR_WORLD)
```

Related commands

- set_ref_coord()(p. 54)

set_user_cart_coord(u1, v1, pos, ref)

Features

This function sets a new user cartesian coordinate system using [u1] and [v1] based on [ref] coordinate system. The origin position the position of [pos] based on the [ref] coordinate while the direction of x-axis and y-axis bases are given in the vectors u1 and v1, respectively. Other directions are determined by u1 cross v1. If u1 and v1 are not orthogonal, v1', that is perpendicular to u1 on the surface spanned by u1 and v1, is set as the vector in the y-axis direction. Up to 100 user coordinate systems can be set including the coordinate systems set within Workcell Item. Since the coordinate system set by this function is removed when the program is terminated, setting new coordinate systems within Workcell Item is recommended for maintaining the coordinate information.

Parameters

Parameter Name	Data Type	Default Value	Description
u1	float[3]	-	X-axis unit vector
v1	float[3]	-	Y-axis unit vector

Parameter Name	Data Type	Default Value	Description
pos	posx list (float[6])	-	posx or position list
ref	int	DR_BASE	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate

Return

Value	Description
Positive integer	Successful coordinate setting Set coordinate ID (101 - 200)
-1	Failed coordinate setting

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
u1 = [1, 1, 0]
v1 = [-1, 1, 0]
pos = posx(10, 20, 30, 0, 0, 0)
user_tc1 = set_user_cart_coord(u1, v1, pos)
user_tc2 = set_user_cart_coord(u1, v1, pos, ref=DR_WORLD)
```

Related commands

- set_ref_coord()(p. 54)

5.2.6 overwrite_user_cart_coord()

Definition

overwrite_user_cart_coord(id, pos, ref, apply_mod)

Features

This function changes the pose and reference coordinate system of the requested user coordinate system [id] with the [pos] and [ref], respectively.

Parameters

Parameter Name	Data Type	Default Value	Description
id	int	-	coordinate ID
pos	posx list (float[6])	-	posx or position list
ref	int	DR_BASE	reference coordinate • DR_BASE: base coordinate • DR_WORLD: world coordinate
apply_mod	int	DR_TEMPORARY	DR_TEMPORARY(0) : Valid only during program execution DR_PERMANENT(1) : Valid even after program ends

Note
If apply_mod is 0, the user coordinate system is changed and maintained only when the program is executed, and if it is 1, the user coordinate system information stored in the host controller itself is changed.

Return

Value	Description
Positive integer	Successful coordinate setting Set coordinate ID (101 - 200)
-1	Failed coordinate setting

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
pose_user1 = posx(30, 40, 50, 0, 0, 0)
id_user = set_user_cart_coord(pose_user1, ref=DR_BASE)
pose_user2 = posx(100, 150, 200, 45, 180, 0)
overwrite_user_cart_coord(id_user, pose_user2, ref=DR_BASE)
pose_user3 = posx(100, 150, 200, 45, 180, 0)
overwrite_user_cart_coord(id_user, pose_user3, ref=DR_BASE, apply_mod=DR_PERMANENT)
```

Related commands

- set_user_cart_coord()(p. 279)

5.2.7 get_user_cart_coord()

Definition

get_user_cart_coord(id, ori_type)

Features

This function returns the pose and reference coordinate system of the requested user coordinate system [id].

Parameters

Parameter Name	Data Type	Default Value	Description
id	int	-	coordinate ID

Parameter Name	Data Type	Default Value	Description
ori_type	int	DR_ELR_ZYZ	orientation type • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	Position and orientation information of the coordinate to get
ref	Reference coordinate of the coordinate to get

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
pose_user1 = posx(10, 20, 30, 0, 0, 0)
id_user = set_user_cart_coord(pose_user1, ref=DR_BASE)
pose, ref = get_user_cart_coord(id_user)
pose_quat, ref = get_user_cart_coord(id_user, DR_QUAT)
```

Related commands

- set_user_cart_coord()(p. 279)

5.2.8 check_position_condition()

Definition

check_position_condition(axis, min, max, ref, mod, pos)

Features

This function checks the status of the given position. This condition can be repeated with the while or if statement. Axis and pos of input paramets are based on the ref coordinate.

In case of ref=DR_TOOL, pos should be defined in BASE coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis
min	float	DR_COND_NONE	Minimum value
max	float	DR_COND_NONE	Maximum value
ref	int	None	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate: User defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute • DR_MV_MOD_REL: Relative
pos	posx list (float[6])	-	posx or position list

Note

- The absolution position is used if the mod is DR_MV_MOD_ABS.
- The pos position is used if the mod is DR_MV_MOD_REL.
- Pos is meaningful only if the mod is DR_MV_MOD_REL.

Return

Value	Description
True	The condition is True.
False	The condition is False.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
CON1= check_position_condition(DR_AXIS_X, min=-5, max=0, ref=DR_WORLD)
CON2= check_position_condition(DR_AXIS_Y, max=700)
CON3= check_position_condition(DR_AXIS_Z, min=-10, max=-5) # -10≤z≤-5
CON4= check_position_condition(DR_AXIS_Z, min=30) # 30≤z

CON5= check_position_condition(DR_AXIS_Z,min=-10,max=-5, ref=DR_BASE) # -10≤z≤-5

CON6= check_position_condition(DR_AXIS_Z,min=-10,max=-5, mod=DR_MV_MOD_ABS) #
-10≤z≤-5

posx1 = posx(400, 500, 800, 0, 180,0)
CON7= check_position_condition(DR_AXIS_Z,min=-10,max=-5,mod = DR_MV_MOD_REL,
pos=posx1) # posx1_(z)-10≤z≤ posx1_(z)-5
```

Related commands

- check_force_condition()(p. 289)
- check_orientation_condition()(p. 290)
- set_ref_coord()(p. 54)

5.2.9 check_force_condition()

Definition

check_force_condition(axis, min, max, ref)

Features

This function checks the status of the given force. It disregards the force direction and only compares the sizes. This condition can be repeated with the while or if statement. Measuring the force and [1)]moment, axis is based on the ref coordinate.

[1)]Before V2.8 software version, measuring the moment, axis is based on the tool coordinate.

Parameters

Parameter Name	Data Type	Default Value	Description
axis	int	-	axis • DR_AXIS_X: x-axis • DR_AXIS_Y: y-axis • DR_AXIS_Z: z-axis • DR_AXIS_A: x-axis rotation • DR_AXIS_B: y-axis rotation • DR_AXIS_C: z-axis rotation
min	float	DR_COND_NONE	Minimum value (min ≥ 0)
max	float	DR_COND_NONE	Maximum value (max ≥ 0)
ref	int	None	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate: User defined

Return

Value	Description
True	The condition is True.
False	The condition is False.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
fcon1 = check_force_condition(DR_AXIS_Z, min=5, max=10, ref=DR_WORLD)           # 5
≤f_z≤10

while (fcon1):
  fcon2 = check_force_condition(DR_AXIS_C, min=30)              # 30≤m_z
  pcon1 = check_position_condition(DR_AXIS_X, min=0, max=0.1)   # 0≤x≤0.1

  if (fcon2 and pcon1):
    break
```

Related commands

- check_position_condition()(p. 287)
- check_orientation_condition()(p. 290)
- set_ref_coord()(p. 54)

5.2.10 check_orientation_condition()

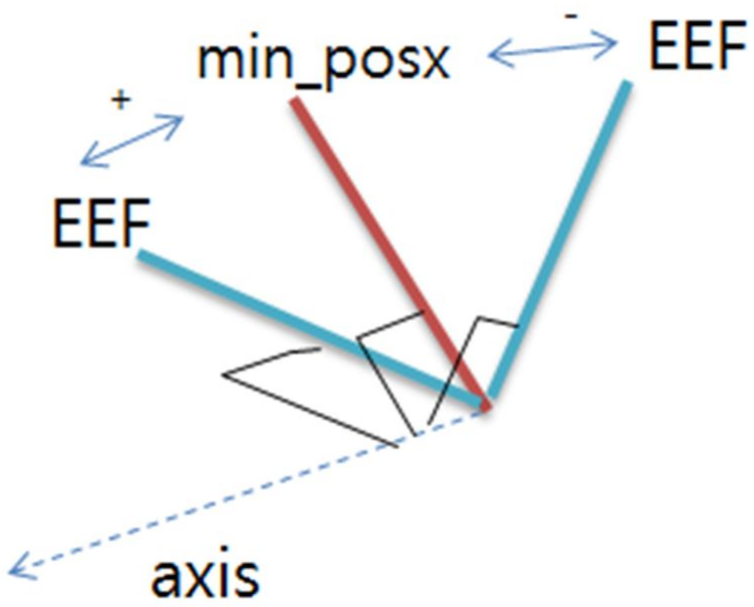
- check_orientation_condition(axis, min, max, ref, mod)(p. 291)
- check_orientation_condition(axis, min, max, ref, mod, pos)(p. 293)

check_orientation_condition(axis, min, max, ref, mod)

Features

This function checks the difference between the current pose and the specified pose of the robot end effector. It returns the difference between the current pose and the specified pose in rad with the algorithm that transforms it to a rotation matrix using the "AngleAxis" technique. It returns True if the difference is positive (+) and False if the difference is negative (-). It is used to check if the difference between the current pose and the rotating angle range is + or -. For example, the function can use the direct teaching position to check if the difference from the current position is + or - and then create the condition for the orientation limit. This condition can be repeated with the while or if statement

- Setting Min only: True if the difference is + and False if -
- Setting Min and Max: True if the difference from min is - while the difference from max is + and False otherwise
- Setting Max only: True if the maximum difference is + and False otherwise



Parameters

Parameter Name	Data Type	Default Value	Description
axis	int	-	axis • DR_AXIS_A: x-axis rotation • DR_AXIS_B: y-axis rotation • DR_AXIS_C: z-axis rotation

Parameter Name	Data Type	Default Value	Description
min	posx list (float[6])	-	posx or position list
max	posx list (float[6])	-	posx or position list
ref	int	None	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • user coordinate: User defined
mod	int	DR_MV_MOD_ABS	Movement basis • DR_MV_MOD_ABS: Absolute

Return

Value	Description
True	The condition is True.
False	The condition is False.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
posx1 = posx(400,500,800,0,180,30)
posx2 = posx(400,500,500,0,180,60)
```

```
CON1= check_orientation_condition(DR_AXIS_C, min=posx1, max=posx2)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 40)
# CON1=True since posx1 Rz=30 < posxc Rz=40 < posx2 Rz=60

CON2= check_orientation_condition(DR_AXIS_C, min=posx1)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 15)
# CON2=False since posx1 Rz=30 > posxc Rz=15

CON3= check_orientation_condition(DR_AXIS_C, max=posx2)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 75)
# CON2=False since posx1 Rz=75 > posxc Rz=60
```

Related commands

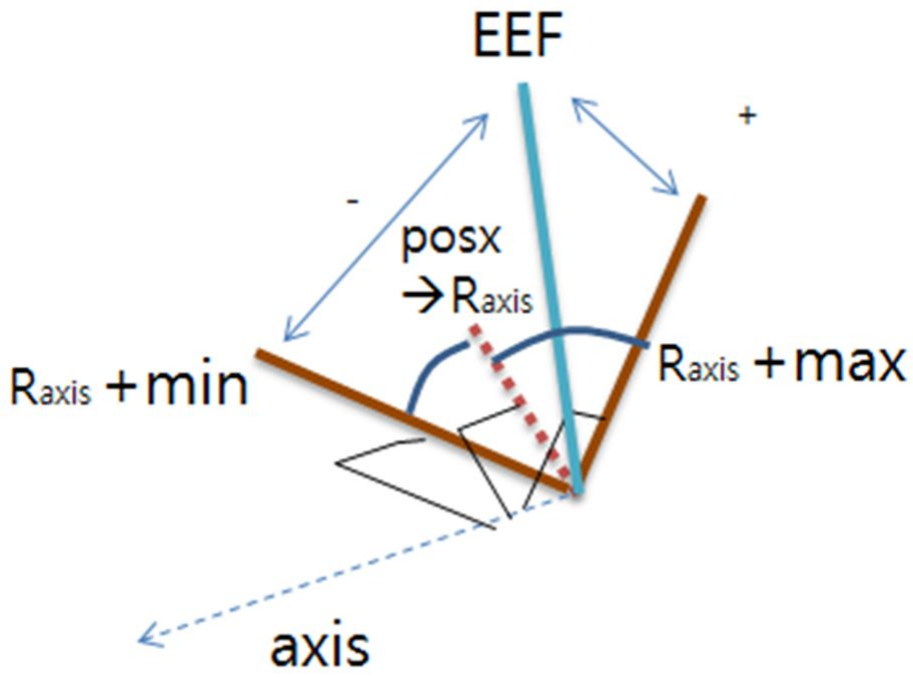
- check_position_condition()(p. 287)
- check_force_condition()(p. 289)
- set_ref_coord()(p. 54)

check_orientation_condition(axis, min, max, ref, mod, pos)

Features

This function checks the difference between the current pose and the rotating angle range of the robot end effector. It returns the difference (in rad) between the current pose and the rotating angle range with the algorithm that transforms it to a rotation matrix using the "AngleAxis" technique. It returns True if the difference is positive (+) and False if the difference is negative (-). It is used to check if the difference between the current pose and the rotating angle range is + or -. For example, the function can be used to set the rotating angle range to min and max at any reference position, and then determine the orientation limit by checking if the difference from the current position is + or -. This condition can be repeated with the while or if statement

- Setting Min only: True if the difference is + and False if -
- Setting Min and Max: True if the difference from min is - while the difference from max is + and False if the opposite.
- Setting Max only: True if the maximum difference is + and False otherwise



Note
Range of rotating angle: Refers to the relative angle range (min, max) basded on the set axis from the given position. The reference coordinate is defined according to the given position based on ref.

Parameters

Parameter Name	Data Type	Default Value	Description
axis	int	-	axis • DR_AXIS_X: x-axis rotation • DR_AXIS_Y: y-axis rotation • DR_AXIS_Z: z-axis rotation
min	float	DR_COND_NONE	Minimum value
max	float	DR_COND_NONE	Maximum value
ref	int	None	reference coordinate • DR_BASE : base coordinate • DR_WORLD : world coordinate • user coordinate: User defined

Parameter Name	Data Type	Default Value	Description
mod	int	DR_MV_MOD_REL	Movement basis • DR_MV_MOD_REL: Relative
pos	posx list (float[6])	-	posx or position list

Return

Value	Description
True	The condition is True.
False	The condition is False.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
posx1 = posx(400,500,800,0,180,15)
CON1= check_orientation_condition(DR_AXIS_C, min=-5, mod=DR_MV_MOD_REL, pos=posx1,
DR_WORLD)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 40)
# CON1=True since posx1 Rz=15 - (min=5) < posxc Rz=40

CON1= check_orientation_condition(DR_AXIS_C, max=5, mod=DR_MV_MOD_REL, pos=posx1)
# If the current task coordinate posxc = posx(400, 500, 500, 0, 180, 40)
# CON1=False since posxc Rz=40 > posx1 Rz=15 + (max=5)
```

Related commands

- check_position_condition()(p. 287)
- check_force_condition()(p. 289)
- set_ref_coord()(p. 54)

5.2.11 coord_transform()

Definition

coord_transform(pose_in, ref_in, ref_out, ori_type_out)

Features

This function transforms given task position expressed in reference coordinate, 'ref_in' to task position expressed in reference coordinate, 'ref_out'. It returns transformed task position. It supports calculation of coordinate transformation for the following cases.

- (ref_in) world reference coordinate → (ref_out) world reference coordinate
- (ref_in) world reference coordinate → (ref_out) base reference coordinate
- (ref_in) world reference coordinate → (ref_out) tool reference coordinate
- (ref_in) world reference coordinate → (ref_out) user reference coordinate
- (ref_in) base reference coordinate → (ref_out) base reference coordinate
- (ref_in) base reference coordinate → (ref_out) tool reference coordinate
- (ref_in) base reference coordinate → (ref_out) user reference coordinate
- (ref_in) tool reference coordinate → (ref_out) world reference coordinate
- (ref_in) tool reference coordinate → (ref_out) base reference coordinate
- (ref_in) tool reference coordinate → (ref_out) tool reference coordinate
- (ref_in) tool reference coordinate → (ref_out) user reference coordinate
- (ref_in) user reference coordinate → (ref_out) world reference coordinate
- (ref_in) user reference coordinate → (ref_out) base reference coordinate
- (ref_in) user reference coordinate → (ref_out) tool reference coordinate
- (ref_in) user reference coordinate → (ref_out) user reference coordinate

Parameters

Parameter Name	Data Type	Default Value	Description
pose_in	posx	-	posx

Parameter Name	Data Type	Default Value	Description
ref_in	float	DR_COND_NONE	reference coordinate before transformation • DR_BASE : base coordinate • DR_WORLD : world coordinate • DR_TOOL : tool coordinate • user coordinate: User defined
ref_out	float	DR_COND_NONE	reference coordinate after transformation • DR_BASE : base coordinate • DR_WORLD world coordinate • DR_TOOL : tool coordinate • user coordinate: User defined
ori_type_out	int	None	orientation type • None: Follows the orientation type of pos_in input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
pos	posx

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
base_pos = posx(400,500,800,0,180,15)
# If task position based on base reference coordinate base_pos =
posx(400,500,800,0,180,15)

tool_pos = coord_transform(base_pos, DR_BASE, DR_TOOL)
# Transform task position(base_pos) expressed in base reference coordinate to task
position expressed in tool reference coordinate
# This command returns task position expressed in tool reference coordinate and the
result value is stored in tool_pos

tool_pos_zyx = coord_transform(base_pos, DR_BASE, DR_TOOL, DR_ELR_ZYX)
# Orientation type of tool_pos_zyx: euler zyx
```

Related commands

- set_user_cart_coord()(p. 279)
- get_current_posx()(p. 166)
- get_desired_posx()(p. 176)
- set_ref_coord()(p. 54)

5.2.12 get_pattern_point()

Definition

get_pattern_point(pos1, pos2, pos3, pos4, index, pattern, row, column, stack, stack_offset, point_offset, ori_type_out=None)

Features

This function calculates the pallet point for the index that fits the given pattern using the given 4 points. Only square and rectangular flat pallets are available. When teaching the 4 points of the pallet, please teach after fixing the Orientation.

Parameters

Parameter Name	Data Type	Default Value	Description
pos1	posx list (float[6])	-	posx or position list
pos2	posx list (float[6])	-	posx or position list
pos3	posx list (float[6])	-	posx or position list
pos4	posx list (float[6])	-	posx or position list
index	int	1	0 ~ [(row X column) – 1]
pattern	Int	0	Normal Pallet -> 0: Snake, 1: Zigzag Rhombus Pallet -> 2: Snake, 3: Zigzag
row	Int	1	count (index of row)
column	Int	1	count (index of column)
stack	int	1	Pallet stack count
stack_offset	float	0.0	Pallet stack height
point_offset	float[3]	None	Teach point fine-tuning (Translation direction)

Parameter Name	Data Type	Default Value	Description
ori_type_out	int	None	output orientation type • None: Follows the orientation type of pos1 input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
pos	posx

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
base_pos = posx(400,500,800,0,180,15)
# If task position based on base reference coordinate base_pos =
posx(400,500,800,0,180,15)

tool_pos = coord_transform(base_pos, DR_BASE, DR_TOOL)
# Transform task position(base_pos) expressed in base reference coordinate to task
position expressed in tool reference coordinate
# This command returns task position expressed in tool reference coordinate and the
result value is stored in tool_pos
```

Related commands

- posx()(p. 28)

5.2.13 get_cockpit_input()

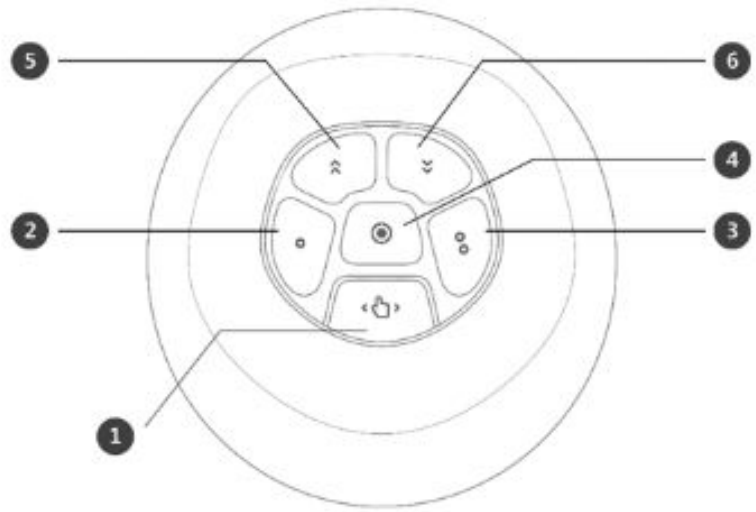
Definition

get_cockpit_input(index)

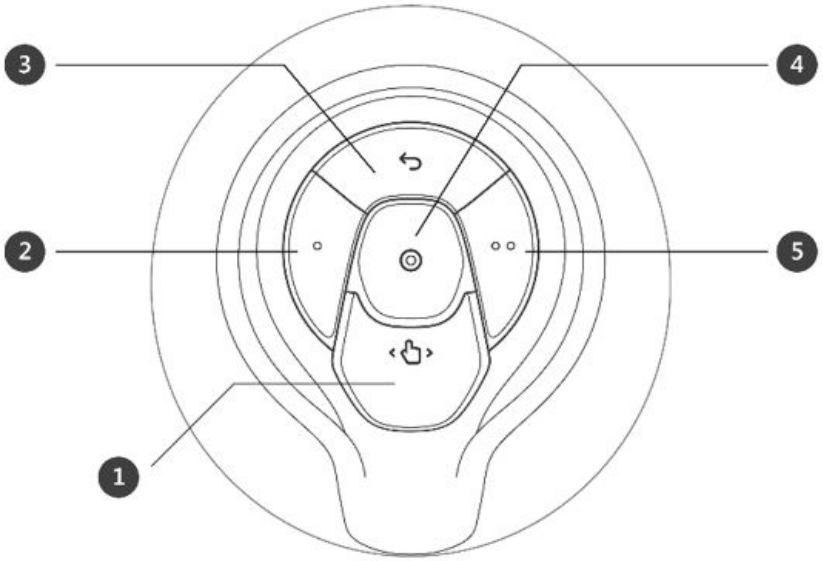
Features

You can check the status of the Cockpit button.

(6 Button)



(5 Button)



Parameter

Parameter Name	Data Type	Default Value	Description
index	int	None	Notifies the press/release status of a designated button among 5 (or 6) Cockpit buttons

Return

Value	Description
ret	0: Release the button 1: Press the button

Example

```
get_cockpit_input(1)
```

6 System Commands

6.1 IO Related

6.1.1 set_digital_output()

set_digital_output(index, val =None)

Features

This function sends a signal at the digital contact point of the controller. A value saved in the digital output register is output as a digital signal.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	I/O contact number mounted on the controller • Val argument existing: A number between 1 and 16 • No val argument: 1 ~ 16 , -1 ~ -16 (A positive number means ON while a negative number means OFF.)
val	int	-	I/O value • ON: 1 • OFF: 0

Note
If val is omitted, positive numbers become ON and negative numbers become OFF depending on the sign (+/-) of the index.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_digital_output(1, ON)           # No. 1 contact ON
set_digital_output(16, OFF) # No. 16 contact OFF
set_digital_output(3)       #No. 3 contact ON (A positive number means ON if the
argument val is omitted.)
set_digital_output(-3)      #No. 3 contact OFF (A negative number means OFF if the
argument val is omitted.)
```

set_digital_output(index, val=None, time=None, val2=None)

Features

This function sends a signal at the digital contact point of the controller. A value saved in the digital output register is output as a digital signal. After sending out the specified signal for the set time, the next signal is sent out.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	I/O contact number mounted on the controller • Val argument existing: A number between 1 and 16 • No val argument: 1 ~ 16 , -1 ~ -16 (A positive number means ON while a negative number means OFF.)

Parameter Name	Data Type	Default Value	Description
val	int	-	I/O value • ON: 1 • OFF: 0
time	float	-	Time(0.01 ~ 3,000,000)
val2	int	-	I/O value • ON: 1 • OFF: 0

Note
If val is omitted, the positive number becomes ON and the negative number OFF according to the sign of the argument index.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_digital_output(1, ON, 2.0, OFF) # No. 1 contact ON, OFF after 2 seconds
set_digital_output(5, OFF, 0.5, OFF)    # No. 16 contact OFF, ON after 0.5 seconds
```

6.1.2 set_digital_outputs()

set_digital_outputs(bit_list)

Features

This function sends a signal to multiple digital output contact points of the controller.

The digital signals of the contact points defined in bit_list are output at one.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_list	list (int)	-	List of multiple output contacts • The positive contact number outputs ON: 1~16 • The negative contact number outputs OFF: -1~-16

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_digital_outputs(bit_list=[1,2,3,4,5,6,7,8])  # Contact number 1-8 ON
set_digital_outputs([-1,-2,-3,-4,-5,-6,-7,-8])    # Contact number 1-8 OFF
set_digital_outputs([1,-2,3])                # Contact no. 1 ON, no. 2 OFF, and no. 3
ON
set_digital_outputs([4,-9,-12])              # Contact no. 4 ON, no. 9 OFF, and no. 12
OFF
```

set_digital_outputs(bit_start, bit_end, val)

Features

This function sends multiple signals at once from the digital output start contact point (bit_start) to the end contact point (bit_end) of the controller.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_start	int	-	Beginning contact number for output signal (1~16)
bit_end	int	-	Ending contact number for output signal (1~16)
val	int	-	Output value

Note

- Bit_end must be a larger number than bit_start.
- Val is the value of the combination of bits where bit_start =LSB and bit_end=MSB.
  Ex) bit_start =1, bit_end=4, val=0b1010 # No. 4=ON, no. 3=OFF, no. 2=ON, and no. 1=OFF

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Outputs contact 1=ON, contact 2=ON, contact 3=OFF, and contact 4=OFF.
set_digital_outputs(bit_start=1, bit_end=4, val=0b0011) # 0b means a binary number.

# Outputs contact 3=ON and contact 4=OFF.
set_digital_outputs(bit_start=3, bit_end=4, val=0b01) # 0b means a binary number.

# Outputs the ON signal from contacts 1 through 8.
set_digital_outputs(1, 8, 0xff)                         # 0x means a hexadecimal
number.
```

6.1.3 get_digital_input()

Definition

get_digital_input(index)

Features

This function reads the signals from digital contact points of the controller and reads the digital input contact value.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	A number 1 - 16 which means the contact number of I/O mounted on the controller.

Return

Value	Description
1	ON
0	OFF
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
in1 = get_digital_input(1)  # Reads the no. 1 contact
```

```
in8 = get_digital_input(8)  # Reads the no. 8 contact
```

6.1.4 get_digital_inputs()

get_digital_inputs(bit_list)

Features

This function reads the signals from multiple digital contact points of the controller. The digital signals of the contact points defined in bit_list are input at one.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_list	list (int)	-	List of contact points to read A number 1-16 which means the I/O contact number mounted on the controller.

Return

Value	Description
int (>=0)	Multiple contacts to be read at once (the value of the combination of the bit list where bit_start =LSB and bit_end=MSB)
Negative number	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# input contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
res = get_digital_inputs(bit_list=[1,2,3,4])
# res expected value = 0b1100 (binary number), 12 (decimal number), or 0x0C
(hexadecimal number)

# input contacts: No. 5=ON, No. 6=ON, No. 7=OFF, and No. 8=ON
res = get_digital_inputs([5,6,7,8])
# res expected value = 0b1011 (binary number), 11 (decimal number), or 0x0B
(hexadecimal number)
```

get_digital_inputs(bit_start, bit_end)

Features

This function reads multiple signals at once from the digital input start contact point (start_index) to the end contact point (end_index) of the controller.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_start	int	-	Beginning contact number for input signals (1-16)
bit_end	int	-	Ending contact number for input signals (1-16)

Note
Bit_end must be a larger number than bit_start.

Return

Value	Description
int (>=0)	Multiple contacts to be read at once Value of the combination of bits where bit_start =LSB and bit_end=MSB.
Negative number	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# input contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
res = get_digital_inputs(bit_start=1, bit_end=4)
#res expected value = 0b1100 (binary number), 12 (decimal number), or 0x0C
(hexadecimal number)
```

6.1.5 wait_digital_input()

Definition

wait_digital_input(index, val, timeout=None)

Features

This function waits until the signal value of the digital input register of the controller becomes val (ON or OFF). The waiting time can be changed with a timeout setting. The waiting time ends, and the result is returned if the waiting time has passed. This function waits indefinitely if the timeout is not set.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	A number 1 - 16 which means the I/O index mounted on the controller.
val	int	-	I/O value • ON : 1 • OFF : 0
timeout	float	-	Waiting time (sec) This function waits indefinitely if the timeout is not set.

Return

Value	Description
0	Success
-1	Failed (time-out)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
wait_digital_input(1, ON) # Indefinite wait until the no. 1 contact becomes ON
wait_digital_input(2, OFF) # Indefinite wait until the no. 2 contact becomes OFF
res = wait_digital_input(1, ON, 3) # Wait for up to 3 seconds until the no. 1 contact
becomes ON
    # Waiting is terminated and res = 0 if the no. 1 contact becomes ON within 3
seconds.
    # Waiting is terminated and res = -1 if the no. 1 contact does not become ON
within 3 seconds.
```

6.1.6 set_tool_digital_output()

set_tool_digital_output(index, val=None)

Features

This function sends the signal of the robot tool from the digital contact point.

> **Caution**
> Depending on the robot model or flange board version, the range of the index parameter may change.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	I/O contact number mounted on the robot arm • Val argument existing: A number between 1 and 6 • No val argument: 1 ~ 6 , -1 ~ -6 (A positive number means ON while a negative number means OFF.)
val	int	-	I/O value: The value to output

> **Note**
> If val is omitted, positive numbers become ON and negative numbers become OFF depending on the sign (+/-) of the index.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_tool_digital_output(1, ON)  # Sets the no. 1 contact of the robot arm ON
set_tool_digital_output(6, OFF) # Sets the no. 6 contact of the robot arm OFF
set_tool_digital_output(3       #No. 3 contact ON (A positive number means ON if the
argument val is omitted.)
set_tool_digital_output(-3)     #No. 3 contact OFF (A negative number means OFF if
the argument val is omitted.)
```

set_tool_digital_output(index, val=None, time=None, val2=None)

Features

This function sends the signal of the robot tool from the digital contact point. After sending out the specified signal for the set time, the next signal is sent out.

Caution
Depending on the robot model or flange board version, the range of the index parameter may change.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	I/O contact number mounted on the robot arm • Val argument existing: A number between 1 and 6 • No val argument: 1 ~ 6 , -1 ~ -6 (A positive number means ON while a negative number means OFF.)
val	int	-	I/O value • ON: 1 • OFF: 0
time	float	-	Time(0.01 ~ 3,000,000)
val2	int	-	I/O value • ON: 1 • OFF: 0

Note
If val is omitted, the positive number becomes ON and the negative number OFF according to the sign of the argument index.

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_tool_digital_output(1, ON, 2.0, OFF)    # Sets the no. 1 contact of the robot arm
ON, OFF after 2 seconds
set_tool_digital_output(5, OFF, 0.5, ON)    # Sets the no. 5 contact of the robot arm
OFF, ON after 0.5 seconds
```

6.1.7 set_tool_digital_outputs()

set_tool_digital_outputs(bit_list)

Features

This function sends the signal of the robot tool from the digital contact point. The digital signals of the contact points defined in bit_list are output at one.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_list	list (int)	-	List of multiple output contacts • The positive contact number outputs ON: 1~6 • The negative contact number outputs OFF: -1~-6

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_tool_digital_outputs(bit_list=[1,2,3,4,5,6])    # Sets the contacts 1-6 ON
set_tool_digital_outputs([-1,-2,-3,-4,-5,-6])       # Sets the contacts 1-6 OFF
set_digital_outputs([1,-2,3])            # Contact no. 1 ON, no. 2 OFF, and no. 3 ON
```

set_tool_digital_outputs(bit_start, bit_end, val)

Features

This function sends the signal of the robot tool from the digital contact point. The multiple signals from the first contact point (bit_start) to the last contact point (bit_end) are output at one.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_start	int	-	Beginning contact number for output signal (1~6)
bit_end	int	-	Ending contact number for output signal (1~6)
val	int	-	Output value

Note

- Bit_end must be a larger number than bit_start.

- Val is the value of the combination of bits where bit_start =LSB and bit_end=MSB.
  Ex) bit_start =1, bit_end=4, val=0b1010 # No. 4=ON, no. 3=OFF, no. 2=ON, and no. 1=OFF

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Outputs contact 1=ON, contact 2=ON, contact 3=OFF, and contact 4=OFF.
set_tool_digital_outputs(bit_start=1, bit_end=4, val=0b0011) # 0b means a binary
number.

# Outputs contact 3=ON and contact 4=OFF.
set_tool_digital_outputs(bit_start=3, bit_end=4, val=0b01) # 0b means a binary
number.

# Outputs the ON signal from contacts 1 through 8.
set_tool_digital_outputs(1, 8, 0xff)                       # 0x means a hexadecimal
number.
```

6.1.8 get_tool_digital_input()

Definition

get_tool_digital_input(index)

Features

This function reads the signal of the robot tool from the digital contact point.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	I/O contact number (1-6) mounted on the robot tool

Return

Value	Description
1	ON
0	OFF
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
get_tool_digital_input(1)       # Reads the no. 1 contact of tool I/O
get_tool_digital_input(6)       # Reads the no. 6 contact of tool I/O
```

6.1.9 get_tool_digital_inputs()

get_tool_digital_inputs(bit_list)

Features

This function reads the signal of the robot tool from the digital contact point.
The digital signals of the contact points defined in bit_list are input at one.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_list	list (int)	-	List of contact points to read • (I/O contact numbers (1-6) mounted on the robot arm)

Return

Value	Description
int (>=0)	Multiple contacts to be read at once (the value of the combination of the bit list where bit_start =LSB and bit_end=MSB)
Negative number	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# input contacts: No. 1=OFF, No. 2=OFF, and No. 3=ON
res = get_tool_digital_inputs(bit_list=[1,2,3]) # Reads the contacts 1, 2, and 3 at
once.
#res expected value = 0b100 (binary number), 4 (decimal number), or 0x04 (hexadecimal
number)

# input contacts: No. 4=ON, No. 5=ON, and No. 6=OFF
res = get_tool_digital_inputs([4,5,6])
#res expected value = 0b011 (binary number), 3 (decimal number), or 0x03 (hexadecimal
number)
```

get_tool_digital_inputs(bit_start, bit_end)

Features

This function reads the signal of the robot tool from the digital contact point. The multiple signals from the first contact point (start_index) to the last contact point (end_index) are input at one.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_start	int	-	Beginning contact number for input signals (1~6)
bit_end	int	-	Ending contact number for input signals (1~6)

Return

Value	Description
int (>=0)	Multiple contacts to be read at once Value of the combination of bits where bit_start =LSB and bit_end=MSB.
Negative number	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# input contacts: No. 1=OFF, No. 2=OFF, and No. 3=ON
res = get_tool_digital_inputs(bit_start=1, bit_end=3)
#res expected value = 0b100 (binary number), 4 (decimal number), or 0x04 (hexadecimal number)

# input contacts: No. 4=ON, No. 5=ON, and No. 6=OFF
res = get_tool_digital_inputs(4, 6)
#res expected value = 0b011 (binary number), 3 (decimal number), or 0x03 (hexadecimal number)
```

6.1.10 wait_tool_digital_input()

Definition

wait_tool_digital_input(index, val, timeout=None)

Features

This function waits until the digital input signal value of the robot tool becomes val (ON or OFF). The waiting time can be changed with a timeout setting. The waiting time ends, and the result is returned if the waiting time has passed. This function waits indefinitely if the timeout is not set.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	A number in 1 - 6 which means the I/O index mounted on the robot arm
val	int	-	I/O value • ON : 1 • OFF : 0
timeout	float	-	Waiting time (sec) This function waits indefinitely if the timeout is not set.

Return

Value	Description
0	Success
-1	Failed (time-out)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
wait_tool_digital_input(1, ON)  # Indefinite wait until the no. 1 contact becomes ON
wait_tool_digital_input(2, OFF) # Indefinite wait until the no. 2 contact becomes OFF

res = wait_tool_digital_input(1, ON, 3) # Wait for up to 3 seconds until the no. 1
contact becomes ON
    # Waiting is terminated and res = 0 if the no. 1 contact becomes ON within 3
seconds.
    # Waiting is terminated and res = -1 if the no. 1 contact does not become ON
within 3 seconds.
```

6.1.11 set_mode_analog_output()

Definition

set_mode_analog_output(ch, mod)

Features

This function sets the channel mode of the controller analog output.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	int	-	• 1 : channel 1 • 2 : channel 2
mod	int	-	analog io mode • DR_ANALOG_CURRENT: Current mode • DR_ANALOG_VOLTAGE: Voltage mode

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Sets analog_output channel 1 to the current mode.
set_mode_analog_output(ch=1, mod=DR_ANALOG_CURRENT)

# Sets analog_output channel 2 to the voltage mode.
set_mode_analog_output(ch=2, mod=DR_ANALOG_VOLTAGE)
```

6.1.12 set_mode_analog_input()

Definition

set_mode_analog_input(ch, mod)

Features

This function sets the channel mode of the controller analog input.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	int	-	• 1 : channel 1 • 2 : channel 2
mod	int	-	analog io mode • DR_ANALOG_CURRENT: Current mode • DR_ANALOG_VOLTAGE: Voltage mode

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Sets analog_input channel 1 to the current mode.
set_mode_analog_input(ch=1, mod=DR_ANALOG_CURRENT)

# Sets analog_input channel 2 to the voltage mode.
```

```
set_mode_analog_input(ch=2, mod=DR_ANALOG_VOLTAGE)
```

6.1.13 set_analog_output()

Definition

set_analog_output(ch, val)

Features

This function outputs the channel value corresponding to the controller analog output.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	int	-	• 1 : channel 1 • 2 : channel 2
val	float	-	analog output value • Current mode: 4.0~20.0 [mA] • Voltage mode: 0~10.0 [V]

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_mode_analog_output(ch=1, mod=DR_ANALOG_CURRENT) #out ch1=current mode
set_mode_analog_output(ch=2, mod=DR_ANALOG_VOLTAGE) #out ch1=voltage mode

set_analog_output(ch=1, val=5.2)   # Outputs 5.2 mA to channel 1
set_analog_output(ch=2, val=10.0)  # Outputs 10V to channel 2
```

6.1.14 get_analog_input()

Definition

get_analog_input(ch)

Features

This function reads the channel value corresponding to the controller analog input.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	int	-	• 1 : channel 1 • 2 : channel 2

Return

Value	Description
float	The analog input value of the specified channel • Current mode: 4.0~20.0 [mA] • Voltage mode: 0~10.0 [V]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_mode_analog_input(ch=1, mod=DR_ANALOG_CURRENT)  #input ch1=current mode
set_mode_analog_input(ch=2, mod=DR_ANALOG_VOLTAGE)  #input ch2=voltage mode

Cur = get_analog_input(1)  # Reads the analog input current value of channel 1
Vol = get_analog_input(2)  # Reads the analog input voltage value of channel 2
```

6.1.15 set_output()

Definition

set_output(port_type, index, val=None, time=None, val2=None)

Features

This function sends a signal at the digital / analog contact point of the controller / flange.

> **Caution**
> Depending on the current flange board version and robot model, the parameters may change. Please familiarize yourself with the manual.

Parameters

Parameter Name	Data Type	Default Value	Description
port_type	int	-	type of contact point • DR_CONTROLLER_DIGITAL : 0 • DR_FLANGE_DIGITAL : 1 • DR_CONTROLLER_ANALOG : 2

index	int	-	I/O contact number mounted on the controller(It also serves as a Channel for the Analog output)
			• when the port_type is DR_CONTROLLER_DIGITAL
			  • Val argument existing: A number between 1 and 16
			  • No val argument: 1 ~ 16 , -1 ~ -16(A positive number means ON while a negative number means OFF.)
			• when the port_type is DR_FLANGE_DIGITAL
			  • when the model of robot is M/H with old flange
			  • Val argument existing: A number between 1 and 6
			  • No val argument: 1 ~ 6 , -1 ~ -6(A positive number means ON while a negative number means OFF.)
			  • when the model of robot is A
			  • Val argument existing: A number between 1 and 2
			  • No val argument: 1 ~ 2 , -1 ~ -2(A positive number means ON while a negative number means OFF.)
			  • when the model of robot is M/H with new flange
			  • Val argument existing: A number between 1 and 4
			  • No val argument: 1 ~ 4, -1 ~ -4(A positive number means ON while a negative number means OFF.)
			• when the port_type is DR_CONTROLLER_ANALOG
			  • 1 ~ 2(channel)

val	float	None	I/O value
			• when the port_type is 0 or 1(digital)
			  • ON : 1
			  • OFF : 0
			• when the port_type is 2(analog)
			  • current mode : 4.00 ~ 20.00(mA)
			  • voltage mode : 0.00 ~ 10.00(V)

time	float	None	Time information (available only if the port_type is 0, 1) • 0.01 ~ 3,000,000(sec)

Parameter Name	Data Type	Default Value	Description
val2	int	None	I/O value • ON : 1 • OFF : 0

Note
If val is omitted, positive numbers become ON and negative numbers become OFF depending on the sign (+/-) of the index.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_output(1, -1) ## set_tool_digital_output(-1)
set_output(0, 2, OFF, 3, ON) ## set_digital_output(2, OFF, 3, ON)
set_mode_analog_output(ch=2, mod=DR_ANALOG_VOLTAGE) #out ch1=voltage mode
set_output(2, 1, 10.0) ## set_analog_output(2, 10)
```

6.1.16 get_input()

Definition

get_input(port_type, index)

Features

This function sends a signal at the digital / analog contact point of the controller / flange.

> **Caution**
> Depending on the current flange board version and robot model, the parameters may change. Please familiarize yourself with the manual.

Parameters

Parameter Name	Data Type	Default Value	Description
port_type	int	-	type of contact point • DR_CONTROLLER_DIGITAL : 0 • DR_FLANGE_DIGITAL : 1 • DR_CONTROLLER_ANALOG : 2 • DR_CONTROLLER_FLANGE : 3

Parameter Name	Data Type	Default Value	Description

index	int	-	I/O contact number mounted on the controller(It also serves as a Chnnel for the Analog output)
			• when the port_type is DR_CONTROLLER_DIGITAL
			  • Val argument existing: A number between 1 and 16
			  • No val argument: 1 ~ 16 , -1 ~ -16(A positive number means ON while a negative number means OFF.)
			• when the port_type is DR_FLANGE_DIGITAL
			  • when the model of robot is M/H with old flange
			  • Val argument existing: A number between 1 and 6
			  • No val argument: 1 ~ 6 , -1 ~ -6(A positive number means ON while a negative number means OFF.)
			  • when the model of robot is A
			  • Val argument existing: A number between 1 and 2
			  • No val argument: 1 ~ 2 , -1 ~ -2(A positive number means ON while a negative number means OFF.)
			  • when the model of robot is M/H with new flange
			  • Val argument existing: A number between 1 and 4
			  • No val argument: 1 ~ 4, -1 ~ -4(A positive number means ON while a negative number means OFF.)
			• when the port_type is DR_CONTROLLER_ANALOG
			  • 1 ~ 2(channel)
			• when the port_type is DR_FLANGE_ANALOG(Only available on new flanges)
			  • 1 ~ 2(A model)
			  • 1 ~ 4(M/H model)

Note

If val is omitted, positive numbers become ON and negative numbers become OFF depending on the sign (+/-) of the index.

Return

Value	Description
1	(Digital) ON

Value	Description
0	(Digital) OFF
float	(Analog) Current mode : 4.0 ~ 20.0(mA) Voltage mode : 0.0 ~ 10.0(V)
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
res1 = get_input(0, 1) ## get_digital_input(1)
res2 = get_input(2, 1) ## get_analog_input(1)
res3 = get_input(3, 1) ## get_tool_analog_input(1)
```

6.1.17 wait_input()

Definition

wait_input(port_type, index, val, timeout=None, condition=None)

Features

It waits until the signal value received from the digital/analog contact of the controller/flange becomes val. The wait time can be changed with the timeout setting, and the result is returned as soon as the wait state ends after the specified amount of time has elapsed. However, if timeout is not set, it will wait indefinitely.

Caution

Depending on the current flange board version and robot model, the parameters may change. Please familiarize yourself with the manual.

Parameters

Parameter Name	Data Type	Default Value	Description
port_type	int	-	type of contact point • DR_CONTROLLER_DIGITAL : 0 • DR_FLANGE_DIGITAL : 1 • DR_CONTROLLER_ANALOG : 2 • DR_FLANGE_ANALOG : 3

Parameter Name	Data Type	Default Value	Description

index	int	-	I/O contact number mounted on the controller(It also serves as a Chnnel for the Analog output)
			• when the port_type is DR_CONTROLLER_DIGITAL
			  • Val argument existing: A number between 1 and 16
			  • No val argument: 1 ~ 16 , -1 ~ -16(A positive number means ON while a negative number means OFF.)
			• when the port_type is DR_FLANGE_DIGITAL
			  • when the model of robot is M/H with old flange
			  • Val argument existing: A number between 1 and 6
			  • No val argument: 1 ~ 6 , -1 ~ -6(A positive number means ON while a negative number means OFF.)
			  • when the model of robot is A
			  • Val argument existing: A number between 1 and 2
			  • No val argument: 1 ~ 2 , -1 ~ -2(A positive number means ON while a negative number means OFF.)
			  • when the model of robot is M/H with new flange
			  • Val argument existing: A number between 1 and 4
			  • No val argument: 1 ~ 4, -1 ~ -4(A positive number means ON while a negative number means OFF.)
			• when the port_type is DR_CONTROLLER_ANALOG
			  • 1 ~ 2(channel)
			• when the port_type is DR_FLANGE_ANALOG(Only available on new flanges)
			  • 1 ~ 2(A model)
			  • 1 ~ 4(M/H model)

val	float	-	I/O value
			• when the port_type is 0 or 1(digital)
			  • ON : 1
			  • OFF : 0
			• when the port_type is 2(analog)
			  • current mode : 4.00 ~ 20.00(mA)
			  • voltage mode : 0.00 ~ 10.00(V)

timeout	float	None	waiting time [sec] If not set, wait indefinitely

Parameter Name	Data Type	Default Value	Description

condition	int	None	Analog value comparison condition
			• DR_ANALOG_CONDITION_UPPER: 0
			  • Wait until the received analog input is greater than or equal to the val parameter.
			• DR_ANALOG_CONDITION_LOWER : 1
			  • Wait until the received analog input is less than or equal to the val parameter

Note
If val is omitted, positive numbers become ON and negative numbers become OFF depending on the sign (+/-) of the index.

Return

Value	Description
0	Success
-1	Failed(time-out)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
wait_input(0, 1, ON) # wait_digital_input(1, ON)
wait_input(1, 2, ON) # wait_tool_digital_input(2, ON)
wait_input(2, 1, 4.0, DR_ANALOG_CONDITION_LOWER) # wait_analog_input(1,
DR_ANALOG_CONDITION_LOWER, 4.0)
```

6.1.18 wait_analog_input()

Definition

wait_analog_input(ch, condition, val, timeout=None)

Features

It waits until the signal value of the analog input channel of the controller becomes val (float value). The wait time can be changed with the timeout setting, and the result is returned as soon as the wait state ends after the specified amount of time has elapsed. However, if timeout is not set, it will wait indefinitely.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	int	-	1 : channel 1 2 : channel 2

condition	int	None	Analog value comparison condition
			• DR_ANALOG_CONDITION_UPPER: 0
			  • Wait until the received analog input is greater than or equal to the val parameter.
			• DR_ANALOG_CONDITION_LOWER : 1
			  • Wait until the received analog input is less than or equal to the val parameter

val	float	-	• current mode : 4.00 ~ 20.00(mA) • voltage mode : 0.00 ~ 10.00(V)
timeout	float	None	waiting time [sec] If not set, wait indefinitely

Return

Value	Description
0	Success
-1	Failed(time-out)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
wait_analog_input(1, DR_ANALOG_CONDITION_UPPER, 5.0, 4) ## Wait until analog data
received through ch 1 is equal to or greater than 5.0
```

6.1.19 wait_tool_analog_input()

Definition

wait_tool_analog_input(ch, condition, val, timeout=None)

Features

This function sends a signal at the digital contact point of the controller. A value saved in the digital output register is output as a digital signal.

> **Caution**
> This command is only available for new flanges.

Parameters

Parameter Name	Data Type	Default Value	Description
ch	int	-	1 : channel 1 2 : channel 2 3 : channel 3 (M/H series) 4 : channel 4 (M/H series)

Parameter Name	Data Type	Default Value	Description

condition	int	None	Analog value comparison condition
			• DR_ANALOG_CONDITION_UPPER: 0
			  • Wait until the received analog input is greater than or equal to the val parameter.
			• DR_ANALOG_CONDITION_LOWER : 1
			  • Wait until the received analog input is less than or equal to the val parameter

val	float	-	• current mode : 4.00 ~ 20.00(mA) • voltage mode : 0.00 ~ 10.00(V)
timeout	float	None	waiting time [sec] If not set, wait indefinitely

Return

Value	Description
0	Success
-1	Failed (time-out)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
wait_tool_analog_input(1, DR_ANALOG_CONDITION_UPPER, 5.0, 4) ## Wait until analog
data received through ch 1 is equal to or greater than 5.0
```

6.1.20 get_digital_output()

Definition

get_digital_output(index)

Features

This function reads the signals from digital contact points of the controller and reads the digital output contact value.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	A number 1 - 16 which means the contact number of I/O mounted on the controller.

Return

Value	Description
1	ON
0	OFF
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
out1 = get_digital_output(1) #Reads the no. 1 contact
out8 = get_digital_output(8) #Reads the no. 8 contact
```

6.1.21 get_digital_outputs()

get_digital_outputs(bit_list)

Features

This function reads the signals from multiple digital contact points of the controller. The digital signals of the contact points defined in bit_list are input at one.

Parameters

Parameter Name	Data Type	Default Value	Description
index	list (int)	-	List of contact points to read A number 1-16 which means the I/O contact number mounted on the controller.

Return

Value	Description
int (>=0)	Multiple contacts to be read at once (the value of the combination of the bit list where bit_start =LSB and bit_end=MSB)
Negative number	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# output contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
res = get_digital_outputs(bit_list=[1,2,3,4])
#res 기대값 = 0b1100 (binary number), 12 (decimal number), or 0x0C (hexadecimal
number)

# output contacts: No. 5=ON, No. 6=ON, No. 7=OFF, and No. 8=ON
res = get_digital_outputs([5,6,7,8])
#res expected value = 0b1011 (binary number), 11 (decimal number), or 0x0B
(hexadecimal number)
```

get_digital_outputs(bit_start, bit_end)

Features

This function reads multiple signals at once from the digital output start contact point (start_index) to the end contact point (end_index) of the controller.

Parameters

Parameter Name	Data Type	Default Value	Description
bit_start	int	-	Beginning contact number for output signals (1-16) (1~16)

Parameter Name	Data Type	Default Value	Description
bit_end	int	-	Ending contact number for output signals (1~16)

Note
Bit_end must be a larger number than bit_start.

Return

값	Description
int (>=0)	Multiple contacts to be read at once Value of the combination of bits where bit_start =LSB and bit_end=MSB.
Negative number	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# output contacts: No. 1=OFF, No. 2=OFF, No. 3=ON, and No. 4=ON
res = get_digital_outputs(bit_start=1, bit_end=4)
#res expected value = 0b1100 (binary number), 12 (decimal number), or 0x0C
(hexadecimal number)
```

6.2 TP Interface

6.2.1 tp_popup()

Definition

tp_popup(message, pm_type=DR_PM_MESSAGE, button_type=0)

Features

This function provides a message to users through the Teach Pendant. The higher level controller receives the string and displays it in the popup window, and the window must be closed by a user's confirmation.

Parameters

Parameter Name	Data Type	Default Value	Description
message	string	-	Message provided to the user • Messages are limited to within 256 bytes. • It is recommended that the text be concise. For long text, some content is omitted with an ellipsis (…). • Formatting-related code such as newline (\n) or carriage return (\r) is not allowed.
pm_type	int	DR_PM_MESSAGE	Message type • DR_PM_MESSAGE • DR_PM_WARNING • DR_PM_ALARM
button_type	int	0	button type of TP pop message • 0 : show Stop & Resume button • 1 : show Stop button

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
tp_popup("move done", DR_PM_MESSAGE)
tp_popup("Error!! ", DR_PM_ALARM)
a=1
b=2
c=3
tp_popup("a={0}, b={1}, c={2}".format(a,b,c) ,DR_PM_MESSAGE)
tp_popup("critical error!! ", DR_PM_ALARM, 1)
```

6.2.2 tp_log()

Definition

tp_log(message)

Features

This function records the user-written log to the Teach Pendant.

Caution

- Excessive use of TP_LOG or SET commands within loops may cause CPU overload.

Parameters

Parameter Name	Data Type	Default Value	Description
message	string	-	Log message • Messages are limited to within 256 bytes. • It is recommended that the text be concise. For long text, some content is omitted with an ellipsis (…). • Formatting-related code such as newline (\n) or carriage return (\r) is not allowed.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
tp_log("movej() is complete!")
```

6.2.3 tp_get_user_input()

Definition

tp_get_user_input(message, input_type)

Features

This function receives the user input data through the Teach Pendant.

Parameters

Parameter Name	Data Type	Default Value	Description
message	string	-	Character string message to be displayed on the TP user input window • Messages are limited to within 256 bytes. • It is recommended that the text be concise. For long text, some content is omitted with an ellipsis (…). • Formatting-related code such as newline (\n) or carriage return (\r) is not allowed.
input_type	int	-	TP user input message type • DR_VAR_INT: Integer type • DR_VAR_FLOAT: Real number type • DR_VAR_STR: Character string • DR_VAR_BOOL: Boolean

Return

Value	Description
User input data	User input data received from the TP

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
q1 = posj(10, 10, 10, 10, 10, 10)
q2 = posj(20, 20, 20, 20, 20, 20)
q3 = posj(30, 30, 30, 30, 30, 30)
q4 = posj(40, 40, 40, 40, 40, 40)
q5 = posj(50, 50, 50, 50, 50, 50)
q6 = posj(60, 60, 60, 60, 60, 60)

int_y= tp_get_user_input("message1", input_type= DR_VAR_INT)
if int_y==1:    # Moves to q1 if the TP user input is 1.
  movej(q1, vel=30, acc=30)
else:       # Moves to q2 if the TP user input is not 1.
  movej(q2, vel=30, acc=30)

float_y= tp_get_user_input("message2", input_type= DR_VAR_FLOAT)
if float_y==3.14:   # Moves to q3 if the TP user input is 3.14.
  movej(q3, vel=30, acc=30)
else:           # Moves to q4 if the TP user input is not 3.14.
  movej(q4, vel=30, acc=30)

str_y= tp_get_user_input("message3", input_type= DR_VAR_STR)
if str_y=="a":      # Moves to q5 if the TP user input is "a".
  movej(q5, vel=30, acc=30)
else:           # Moves to q6 if the TP user input is not "a".
  movej(q6, vel=30, acc=30)

bool_y= tp_get_user_input("message3", input_type= DR_VAR_BOOL)
if bool_y==True:        # Moves to q5 if the TP user input is "True or 1".
  movej(q5, vel=30, acc=30)
else:                   # Moves to q6 if the TP user input is "False or 0"
  movej(q6, vel=30, acc=30)
```

6.3 Thread

6.3.1 thread_run()

Definition

thread_run(th_func_name, loop=False)

Features

This function creates and executes a thread. The features executed by the thread are determined by the functions specified in th_func_name.

> **Note**
> The following constraints are applied when using the thread command.
> - Up to 4 threads can be used.
> - The following motion command cannot be used to move the robot in the thread.
>   - movej, amovej, movejx, amovejx, movel, amovel, movec, amovec, movesj, amovesj,
>   - movesx, amovesx, moveb, amoveb, move_spiral, amove_spiral,
>   - move_periodic, amove_periodic, move_home
> - The thread commands do not operate normally when the loop=Ture during thread_run and the block is an indefinite loop within the thread function. (The thread is normally stopped when the stop command is executed through the TP.)

Parameters

Parameter Name	Data Type	Default Value	Description
th_func_name	callable	-	Name of the function run by the thread
loop	bool	False	Flag indicates whether the thread will be repeated • True: Repeated calling of th_func_name (interval 0.01second) • False: One-time calling of th_func_name

Return

Value	Description
int	Registered thread ID

Value	Description
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#----- Thread ---------------------------------------
def fn_th_func():
    if check_motion()==0:   # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)

#----- Main routine ---------------------------------
th_id = thread_run(fn_th_func, loop=True) # Thread run

while 1:
    # do something…
    wait(0.1)
```

6.3.2 thread_stop()

Definition

thread_stop(th_id)

Features

This function terminates a thread.

The program is automatically terminated when the DRL program is terminated even if the thread_stop() command is not used.

Parameters

Parameter Name	Data Type	Default Value	Description
th_id	int	-	Thread ID to stop

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
def fn_th_func():
    if check_motion()==0:   # No motion in action
        set_digital_output(1, OFF)
```

```
    else:
        set_digital_output(1, ON)
#----- Main routine ----------------------------------------
th_id = thread_run(fn_th_func, loop=True)

# do something…
thread_stop(th_id) # Stops the thread.
```

6.3.3 thread_pause()

Definition

thread_pause(th_id)

Features

This function temporarily suspends a thread.

Parameters

Parameter Name	Data Type	Default Value	Description
th_id	int	-	Thread ID to suspend

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
def fn_th_func():
    if check_motion()==0:   # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)
#----- Main routine ---------------------------------------
th_id = thread_run(fn_th_func, loop=True)

# do something…

thread_pause(th_id) # Suspends the thread.
```

6.3.4 thread_resume()

Definition

thread_resume(th_id)

Features

This function resumes a temporarily suspended thread.

Parameters

Parameter Name	Data Type	Default Value	Description
th_id	int	-	Suspended thread ID to be resumed

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
def fn_th_func():
    if check_motion()==0:   # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)

#----- Main routine ---------------------------------------
th_id = thread_run(fn_th_func, loop=True)

# do something…
thread_pause(th_id) # Suspends the thread.

# do something…
thread_resume(th_id) # Resumes the suspended thread.
```

6.3.5 thread_state()

Definition

thread_state(th_id)

Features

This function checks the status of a thread.

Parameters

Parameter Name	Data Type	Default Value	Description
th_id	int	-	Thread ID to check the status

Return

Value	Description
1	RUN (TH_STATE_RUN)
2	PAUSE (TH_STATE_PAUSE)
3	STOP (TH_STATE_STOP)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Example

```
def fn_th_func():
    if check_motion()==0:   # No motion in action
        set_digital_output(1, OFF)
    else:
        set_digital_output(1, ON)

th_id = thread_run(fn_th_func, loop=True)
state1 = thread_state(th_id)

thread_pause(th_id)
state2 = thread_state(th_id)
```

6.3.6 Integrated example - Thread

This example explains how to use the thread.

Example 1: Thread example

```
#----- thread 1: client comm. ---------------------
def fn_th_client():
  global g_sock
  global g_cmd
  res, rx_data = client_socket_read(g_sock)
  if res > 0:
    g_cmd = rx_data.decode() #decode: Converts byte type into a string.
  else: # Communication error
    client_socket_close(g_sock)
    exit()  # Terminates the program.
  wait(0.1)
  return 0

#----- thread 2: check IO -------------------------
def fn_th_check_io():
  if get_digital_input(1) == ON:
    exit()  # Terminates the program.
  wait(0.1)
  return 0

#----- main ---------------------------------------
g_sock = client_socket_open("192.168.137.2", 20002) # Connects to the server.
g_cmd = ""

g_th_id1 = thread_run(th_client, loop=True)   # Runs the th_client thread.
g_th_id2 = thread_run(th_check_io, loop=True) # Runs the th_check_io thread.

p1 = posj(0, 0, 90, 0, 90, 0)
p2 = posj(10, 0, 90, 0, 90, 0)
p3 = posj(20, 0, 90, 0, 90, 0)

while 1:
  if g_cmd == "a":
    g_cmd = ""
    movej(p1,vel=100,acc=100)
    client_socket_write(g_sock, b"end")
  if g_cmd == "b":
    g_cmd = ""
    movej(p2,vel=100,acc=100)
    client_socket_write(g_sock, b"end")
  if g_cmd == "c":
    g_cmd = ""
    movej(p3,vel=100,acc=100)
    client_socket_write(g_sock, b"end")
```

```
wait(0.1)
```

th_client thread: Converts the data received from the server into a string and saves it in g_cmd.

th_check_io thread: Checks the state of contact no. 1 and terminates the program if it is ON.

main : Connects to the server.

1. 2 threads run: th_client and th_check_io
2. If "a" is received from the server, it moves to p1 and sends "end" to the servers.
3. If "b" is received from the server, it moves to p2 and sends "end" to the servers.
4. If "c" is received from the server, it moves to p3 and sends "end" to the servers.

6.4 Others

6.4.1 wait()

Definition

wait(second)

Features

This function waits for the specified time.

Parameters

Parameter Name	Data Type	Default Value	Description
second	Float	-	Time [sec]

Return

Value	Description
0	Success
Negative value	Error

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
wait(1.3)  # Waits for 1.3 seconds.

while 1:  # Checks contact no. 1 every 0.1 second.
  if get_digital_input(1) == ON:
    set_digital_output(1, ON)
  wait(0.1)
```

6.4.2 exit()

Features

This function terminates the currently running program.

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Example

```
exit()
```

6.4.3 sub_program_run()

Definition

sub_program_run(name)

Features

It executes a subprogram saved as a separate file.

Parameters

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of subprogram

Return

Value	Description
module	Module object of executed subprogram

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Note

- The first line of the subprogram must have the phrase "from DRCF import *".
- When programming with a teaching pendant, this phrase is automatically inserted.
- If the global variable names of the main program and subprograms are the same, they operate as different variables. Variables cannot be referenced by each other.
- If you need to share variables between the main program and subprograms, use system variables.

- System variables are set through the teaching pendant. Please refer to the user manual for detailed usage.

Example

```
# subprogramA and subprogramB must be created and saved in advance.

<subProgramA.drl>
from DRCF import *
movej([0,0,90,0,90,0], vel=30, acc=30)

<subProgramB.drl>
from DRCF import *
movej[(10,0,90,0,90,0), vel=30, acc=30)

<main program>
while True:
    var_select = tp_get_user_input("Select File", DR_VAR_INT)
    if var_select == 0:
       sub_program_run("subProgramA")  # execute subProgramA
    elif var_select == 1:
       sub_program_run("subProgramB")  # execute subProgramB
```

6.4.4 drl_report_line()

Definition

drl_report_line(option)

Features

This command is used to turn ON / OFF the execution line display function when the DRL script is running. When the run line display function is turned OFF, the time required to execute the run line display function is reduced, which significantly speeds up the execution of the DRL.

Caution

The following features do not operate in the section where the execution line display function is turned OFF.

- Execution time display by line
- Variable monitoring
- System Variable Update
- Step by Step in Debug mode
- Brake Point in Debug mode

Parameters

Parameter Name	Data Type	Default Value	Description
option	Int	-	Whether to display the DRL execution line ON(1), OFF(0)

Return

Value	Description
None	-

Example

```
x=0
y=0

drl_report_line(OFF)     # Execution line display function OFF
while x < 1000:          # Execution line not displayed (speed up execution)
  x += 1                 # Execution line not displayed (speed up execution)
  drl_report_line(ON)    # Execution line display function ON
x=0                      # Execution line shown
y=0                      # Execution line shown
```

6.4.5 set_fm()

Definition

set_fm(key, value)

Features

This command is used when interworking is required for information on variables (global variables, system variables, etc.) created when the program is executed, in addition to the system information already defined and linked with KT Smart Factory.

> **Caution**
> Please note that this function will not work if the linkage information is not set in the KT Smart Factory menu in the Setup menu.
> The KT Smart Factory menu only appears when setting up KT-specific licenses.

Parameters

Parameter Name	Data Type	Default Value	Description
key	string	-	Data Name
value	int float string	-	Interlocking Data Variable Possible data types • Integer data • Real data • string data

Return

Value	Description
None	-

Example

```
count = 0

movej(posj(0, 0, 90, 0,90,0), vel=30, acc=30)
while True:
    movej(posj(0, 0, -90, 0,90,0), vel=30, acc=30)
    movej(posj(0, 0, 90, 0,90,0), vel=30, acc=30)
    count = count + 1
    set_fm("TotalCount", count)
```

6.4.6 get_robot_model()

Features

This is a command to read the model name of the robot.

Return

Value	Description
model name	Returns the model name in String type. "M1013", "M0617", "M0609", "M1509", "A0307", "A0307S", "A0509", "A0509S", "A0912", "A0912S" "H2515", "H2017"

Example

```
model = get_robot_model()

if model =="M1013":
    set_velj(30)
else:
    set_velj(50)
```

6.4.7 get_robot_serial_num()

Features

This is a command to read the serial number of the robot.

Return

Value	Description
serial_number	Returns the serial number in String type. Serial number: 6-character string consisting of numbers and English characters

Example

```
serial_num = get_robot_serial_num()
```

6.4.8 check_robot_jts()

Features

This is a command to check whether the robot is equipped with a joint torque sensor.

Return

Value	Description
0	Success
Negative value	Error

Example

```
if check_robot_jts() != True:
  movej([0,0,90,0,90,0], 60, 30)
else:
  movej([0,0,0,0,0,0], 60, 30)
```

6.4.9 check_robot_fts()

Features

This is a command to check whether the robot is equipped with a force torque sensor.

Return

Value	Description
0	Success
Negative value	Error

Example

```
if check_robot_fts() != True:
    movej([0, 0, 90, 0, 90, 0], 60, 30)
else:
```

```
movej([0, 0, 0, 0, 0, 0], 60, 30)
```

6.4.10 start_timer()

Features

This is a command to measure the execution time of the simulation program of the controller. When used with the end_timer() command, it returns the execution time of the script between the two functions.

> **Caution**
> This function is for measuring motion execution time in Windows environment and Linux environment. When measuring in Real mode in an emulator (virtual controller) environment, incorrect values may be returned.

Return

Value	Description
0	Success
Negative value	Error

Example

```
start_timer()
wait(1)
t= end_timer()
tp_log("tttt={0} sec".format(t))
```

Related Commands

- end_timer()(p. 367)

6.4.11 end_timer()

Features

This is a command to measure the execution time of the simulation program of the controller. When used with the start_timer() command, it returns the execution time of the script between the two functions.

> **Caution**

This function is for measuring motion execution time in Windows environment and Linux environment. When measuring in Real mode in an emulator (virtual controller) environment, incorrect values may be returned.

Return

Value	Description
float	Measured time information (process execution time)

Example

```
start_timer()
wait(1)
t= end_timer()
tp_log("tttt={0} sec".format(t))
```

Related Commands

- start_timer()(p. 367)

6.4.12 message_to_dp()

Definition

message_to_dp(event_name, strdata)

Features

After providing a message to the user, a response message is returned when a response message is received from the user.

Parameters

Parameter Name	Data Type	Default Value	Description
event_name	string	-	Eventname to provide to the user • event_name is limited to 256 bytes.
strdata	string	-	Data to be provided to users • strdata is limited to 3000 bytes.

Return

Value	Description
string	Event name entered when calling Dart-API (sendUserEventResponse)
string	strdata entered when calling Dart-API (sendUserEventResponse)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
tp_log('message send!')
event, data = message_to_dp('wait', 'true')
tp_log('message get!')
```

6.4.13 send_load_module()

Definition

send_load_module(package, sub_type, active)

Features

Loads the Framework Module and returns the UniqueID of the loaded Module.

Parameters

Parameter Name	Data Type	Default Value	Description
package	str	-	Package name of module to load
sub_type	int	-	Sub type of module to load 0 : None 1 : TCP/IP 2 : Serial 4 : Modbus Slave 5 : Force Compliance 7 : Modbus Master 10 : Other
active	int	-	Whether to activate after loading 0 : no active 1 : active (Used only when sub_type is Force Compliance)

Return

Value	Description
int	Sub type of loaded module
int	status value 0 : success Value other than 0 : Error code
int	UniqueID of loaded module
str	Package name of the loaded module

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
sub_type, status, id, name =
 send_load_module("com.dart.module.default.admittancecontrol", 5, 1)
```

6.4.14 send_unload_module()

Definition

send_unload_module(unique_id)

Features

Unload the Framework Module.

Parameters

Parameter Name	Data Type	Default Value	Description
unique_id	int	-	UniqueID of module to unload

Return

Value	Description
int	status value 0 : success Value other than 0 : Error code
int	UniqueID of unloaded module

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
sub_type, status, id, name =
 send_load_module("com.dart.module.default.admittancecontrol", 5, 1)
send_unload_module(id)
```

7 Mathematical Function

7.1 Basic Function

7.1.1 ceil(x)

Features

This function returns the smallest integer value of integers equal to or larger than x. It truncates up to the integer.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Value	Description
rounded integer	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.1.2 floor(x)

Features

This function returns the largest integer value of integers equal to or smaller than x. It rounds down to the nearest one.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Value	Description
rounded integer	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.1.3 pow(x, y)

Features

Return x raised to the power of y.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	
y	float	-	

Return

Value	Description
x raised to the power y	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.1.4 sqrt(x)

Features

This function returns the square root of x.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Value	Description
the square root of x	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occured.

7.1.5 log(x, b)

Features

This function returns the log of x with base b.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-
b	float	-	base, e (natural logarithm)

Return

Value	Description
the logarithm of f to the base of b.	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occured

7.1.6 d2r(x)

Features

This function returns the x degrees value to radians.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	The angle in degrees

Return

Value	Description
The angle in radians	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.1.7 r2d(x)

Features

This function returns the x radians value to degrees.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float		The angle in radians

Return

Value	Description
The angle in degrees	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occured

7.1.8 random()

Features

This function returns a random number between 0 and 1.

Return

Value	Description
random number	Random number between 0 and 1 (float)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.2 Trigonometric functions

7.2.1 sin(x)

Features

This function returns the sine value of x radians.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Value	Description
the sine of x	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.2.2 cos(x)

Features

This function returns the sine value of x radians.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Value	Description
the cosine of x	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.2.3 tan(x)

Features

This function returns the tangent value of x radians.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Value	Description
the tangent of x	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.2.4 asin(x)

Features

This function returns the arc sine value of x radians.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	

Return

Value	Description
the arc sine of x	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.2.5 acos(x)

Features

This function returns the arc cosine value of x radians.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Value	Description
the arc cosine of x	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.2.6 atan(x)

Features

This function returns the arc tangent value of x radians.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float	-	-

Return

Return	Description
the arc tangent of x	-

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.2.7 atan2(y, x)

Features

This function returns the arc tangent value of y/x radians.

Parameters

Parameter Name	Data Type	Default Value	Description
y	float	-	-
x	float	-	-

Return

Value	Description
the arc tangent of y/x	The result is between -pi and pi

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.3 Linear algebra

7.3.1 norm(x)

Features

This function returnesthe L2 norm of x.

Parameters

Parameter Name	Data Type	Default Value	Description
x	float[3]	-	Point coordinate (x, y, z)

Return

Value	Description
float	Size of the point coordinate vector

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

7.3.2 rotx(angle)

Features

This function returns a rotation matrix that rotates by the angle value along the x-axis.

Parameters

Parameter Name	Data Type	Default Value	Description
angle	float	0	Rotating angle [deg]

Return

Value	Description
float[3][3]	Rotation matrix

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rotm = rotx(30)
```

7.3.3 roty(angle)

Features

This function returns a rotation matrix that rotates by the angle value along the y-axis.

Parameters

Parameter Name	Data type	Default Value	Description
angle	float	0	Rotating angle [deg]

Return

Value	Description
float[3][3]	Rotation matrix

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rotm = roty(30)
```

7.3.4 rotz(angle)

Features

This function returns a rotation matrix that rotates by the angle value along the z-axis.

Parameters

Parameter Name	Data Type	Default Value	Description
angle	float	0	Rotating angle [deg]

Return

Value	Description
float[3][3]	Rotation matrix

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rotm = rotz(30)
```

7.3.5 rotm2eul(rotm)

Features

This function receives a rotation matrix and returns the Euler angle (zyz order) to degrees. Of the Euler angle (rx, ry, rz) returned as a result, ry is always a positive number.

Parameters

Parameter Name	Data Type	Default Value	Description
rotm	Float[3][3]	-	Rotation matrix

Return

Value	Description
float[3]	ZYZ Euler angle [deg]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rotm = [[1,0,0],[0,0.87,-0.5],[0,0.5,0.87]]
eul = rotm2eul(rotm)
```

7.3.6 rotm2rotvec(rotm)

Features

This function receives a rotation matrix and returns the rotation vector (angle/axis representation).

Parameters

Parameter Name	Data Type	Default Value	Description
rotm	float[3][3]	-	Rotation matrix

Return

Value	Description
float[3]	rotation vector [rad]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rotm = [[1,0,0],[0,0.87,-0.5],[0,0.5,0.87]]
rotvec = rotm2rotvec(rotm)
```

7.3.7 eul2rotm([alpha,beta,gamma])

Features

This function transforms a Euler angle (zyz order) to a rotation matrix.

Parameters

Parameter Name	Data Type	Default Value	Description
eul	float[3]	[0 0 0]	Euler angle (zyz) [deg]

Return

Value	Description
float[3][3]	Rotation matrix

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
eul = [90, 90, 0]
rotm = eul2rotm (eul)
```

7.3.8 eul2rotvec([alpha,beta,gamma])

Features

This function transforms a Euler angle (zyz order) to a rotation vector.

Parameters

Parameter Name	Data Type	Default Value	Description
eul	float[3]	[0 0 0]	Euler angle (zyz) [deg]

Return

Value	Description
float[3]	rotation vector [rad]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
eul = [90, 90, 0]
rotvec = eul2rotvec (eul)
```

7.3.9 eul2rpy([alpha,beta,gamma])

Features

This function receives Euler angle zyz as a degree value and returns the degree value of Euler angle z(=Yaw)y(=Pitch)x(=Roll).

Parameters

Parameter Name	Data Type	Default Value	Description
eul	float[3]	[0, 0, 0] [deg]	Euler angle zyz [deg] ※ Input in order of alpha (A), beta (B), gamma (C)

Return

Value	Description
float[3]	Euler angle (zyx) [deg] ※ Return in order of yaw (Rz), pitch (Ry), roll (Rx)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
eul = [10, 20, 30]
rpy = eul2rpy(eul)
```

7.3.10 rpy2eul([yaw,pitch,roll])

Features

This function receives Euler anlge z(=Yaw)y(=Pitch)x(=Roll) as a degree value and returns the degree value of Euler angle zyz.

Parameters

Parameter Name	Data Type	Default Value	Description
rpy	float[3]	[0 0 0]	Euler angle (zyx) [deg] ※ Input in order of yaw (Rz), pitch (Ry), roll (Rx)

Return

Value	Description
float[3]	Euler angle (zyz) [deg] ※ Return in order of alpha(A), beta(B), gamma(C)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rpy = [10, 20, 30]
eul = rpy2eul(rpy)
```

7.3.11 rotvec2eul([rx,ry,rz])

Features

This function transforms a rotation vector to a Euler angle (zyz).

Parameters

Parameter Name	Data Type	Default Value	Description
rotvec	float[3]	-	rotation vector [rad]

Return

Value	Description
float[3]	ZYZ Euler angle [deg]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rotvec = [0, 0, 0.7854]
eul = rotvec2eul(rotvec) # eul=[45,0,0]
```

7.3.12 rotvec2rotm([rx,ry,rz])

Features

This function transforms a rotation vector to a rotation matrix.

Parameters

Parameter Name	Data Type	Default Value	Description
rotvec	float[3]	-	rotation vector [rad]

Return

Value	Description
float[3][3]	Rotation matrix

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
rotm = rotvec2rotm([0.7854,0,0])
```

7.3.13 htrans()

Definition

htrans(posx1,posx2,ori_type_out)

Features

This function returns the pose corresponding to T1*T2 assuming that the homogeneous transformation matrices obtained from posx1 and posx2 are T1 and T2, respectively.

$$H_1H_2 = \begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_2 & r_2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_1R_2 & r_1 + R_1r_2 \\ 0 & 1 \end{bmatrix}$$

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]
posx2	posx list (float[6])	-	posx or position list [mm, deg]
ori_type_out	int	None	output orientation type • None: Follows the orientation type of posx1 input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	[mm, deg]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
posx = htrans(posx1,posx2)
posx_quat = htrans(posx1, posx2, DR_QUAT)
```

7.3.14 get_intermediate_pose()

Definition

get_intermediate_pose(posx1,posx2,alpha,ori_type_out)

Features

This function returns posx located at alpha of the linear transition from posx1 to posx2. It returns posx1 if alpha is 0, the median value of two poses if alpha is 0.5, and posx2 if alpha is 1.

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]
posx2	posx list (float[6])	-	posx or position list [mm, deg]
alpha	float	-	0.0 ≤ alpha ≤ 1.0
ori_type_out	int	None	output orientation type • None: Follows the orientation type of posx1 input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	[mm, deg]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
alpha = 0.5
posx = get_intermediate_pose(posx1,posx2,alpha)
```

7.3.15 get_distance()

Definition

get_distance(posx1, posx2)

Features

This function returns the distance between two pose positions in [mm].

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm]
posx2	posx list (float[6])	-	posx or position list [mm]

Return

Value	Description
float	[mm]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
dis_posx = get_distance(posx1, posx2)
```

7.3.16 get_normal()

Definition

get_normal(x1, x2, x3)

Features

This function returns the normal vector of a surface consisting of three points (posx) in the task space. This direction is clockwise.

Parameters

Parameter Name	Data Type	Default Value	Description
x1	posx list (float[6])	-	posx or position list
x2	posx list (float[6])	-	posx or position list

Parameter Name	Data Type	Default Value	Description
x3	posx list (float[6])	-	posx or position list

Return

Value	Description
float[3]	normal vector

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
x1 = posx(0, 500, 700, 30, 0, 90)
x2 = posx(500, 0, 700, 0, 0, 45)
x3 = posx(300, 100, 500, 45, 0, 45)

vect = get_normal(x1, x2, x3)
```

7.3.17 add_pose()

Definition

add_pose(posx1,posx2,ori_type_out)

Features

This function obtains the sum of two poses.

$$\text{add\_pose}(\begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} R_2 & r_2 \\ 0 & 1 \end{bmatrix}) \Rightarrow \begin{bmatrix} R_1 R_2 & r_1 + r_2 \\ 0 & 1 \end{bmatrix}$$

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]
posx2	posx list (float[6])	-	posx or position list [mm, deg]
ori_type_out	int	None	output orientation type • None: Follows the orientation type of posx1 input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	[mm, deg]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
add_posx = add_pose(posx1, posx2)
add_posx_zyx = add_pose(posx1, posx2, DR_ELR_ZYX)
```

7.3.18 subtract_pose()

Definition

subtract_pose(posx1,posx2,ori_type_out)

Features

This function obtains the difference between two poses.

$$\text{subtract\_pose}(\begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} R_2 & r_2 \\ 0 & 1 \end{bmatrix}) \Rightarrow \begin{bmatrix} R_2^T R_1 & r_1 - r_2 \\ 0 & 1 \end{bmatrix}$$

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]
posx2	posx list (float[6])	-	posx or position list [mm, deg]
ori_type_out	int	None	output orientation type • None: Follows the orientation type of posx1 input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	[mm, deg]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
subtract_posx = subtract_pose(posx1, posx2)
subtract_posx_zyx = subtract_pose(posx1, posx2, DR_ELR_ZYX)
```

7.3.19 inverse_pose()

Definition

inverse_pose(posx1,ori_type_out)

Features

This function returns the posx value that represents the inverse of posx.

$$\text{inv\_pose}\cdot\left(\begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}\right)=\begin{bmatrix} R_1 & r_1 \\ 0 & 1 \end{bmatrix}^{-1}=\begin{bmatrix} R_1^T & -R_1^T r_1 \\ 0 & 1 \end{bmatrix}$$

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]

Parameter Name	Data Type	Default Value	Description
ori_type_out	int	None	output orientation type • None: Follows the orientation type of posx1 input parameter • DR_ELR_ZYZ: Euler Angles(z-y'-z'', in degrees) • DR_ELR_ZYX: Euler Angles(z-y'-x'', in degrees) • DR_ELR_XYZ: Euler Angles(x-y'-z'', in degrees) • DR_FIX_XYZ: Fixed Angles(x-y-z, in degrees) • DR_ROTVEC: 3D rotation vector (angle/axis representation, in degrees) • DR_QUAT: unit quaternion(x, y, z, w)

Return

Value	Description
posx	[mm, deg]

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
inv_posx = inverse_pose(posx1)
```

7.3.20 dot_pose()

Definition

dot_pose(posx1, posx2)

Features

This function obtains the inner product of the translation component when two poses are given.

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]
posx2	posx list (float[6])	-	posx or position list [mm, deg]

Return

Value	Description
float	Inner product of two poses

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
res= dot_pose(posx1, posx2)
```

7.3.21 cross_pose()

Definition

cross_pose(posx1, posx2)

Features

This function obtains the outer product of the translation component when two poses are given.

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]
posx2	posx list (float[6])	-	posx or position list [mm, deg]

Return

Value	Description
float[3]	Outer product of two poses.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
posx2 = [200, 50, 100, 90, 30, 150]
res= cross_pose(posx1, posx2)
```

7.3.22 unit_pose()

Definition

unit_pose(posx1)

Features

This function obtains the unit vector of the given posx translation component.

Parameters

Parameter Name	Data Type	Default Value	Description
posx1	posx list (float[6])	-	posx or position list [mm, deg]

Return

Value	Description
float[3]	Unit vector of the given posx

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
posx1 = [100, 20, 300, 90, 0, 180]
res = unit_pose(posx1)
```

8 External Communication Commands

8.1 Serial

8.1.1 serial_open()

Definition

serial_open(port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

Features

This function opens a serial communication port.

Parameters

Parameter Name	Data Type	Default Value	Description
port	string	None	• D-SUB(9 pin) Connection : “COM” • USB to Serial Connection : “COM_USB”
baudrate	int	115200	Baud rate 2400, 4800, 9600, 19200, 38400, 57600, 115200
bytesize	int	8	Number of data bits • DR_FIVEBITS: 5 • DR_SIXBITS: 6 • DR_SEVENBITS: 7 • DR_EIGHTBITS: 8
parity	str	"N"	Parity checking • DR_PARITY_NONE: "N" • DR_PARITY_EVEN: "E" • DR_PARITY_ODD: "O" • DR_PARITY_MARK: "M" • DR_PARITY_SPACE: "S"

Parameter Name	Data Type	Default Value	Description
stopbits	int	1	Number of stop bits • DR_STOPBITS_ONE =1 • DR_STOPBITS_ONE_POINT_FIVE = 1.5 • DR_STOPBITS_TWO = 2

Return

Value	Description
serial.Serial instance	Successful connection

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	Serial.SerialException error occurred

Example

```
# When connected to serial port D-SUB (9 pin)
ser = serial_open(port="COM", baudrate=115200, bytesize=DR_EIGHTBITS,
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

res = serial_write(ser, b"123ABC")

serial_close(ser)


# When a USB to serial device is connected to a USB port
ser = serial_open(port="COM_USB", baudrate=115200, bytesize=DR_EIGHTBITS,
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

res = serial_write(ser, b"123ABC")

serial_close(ser)
```

8.1.2 serial_close()

Definition

serial_close(ser)

Features

This function closes a serial communication port.

Parameters

Parameter Name	Data Type	Default Value	Description
ser	serial.Serial	-	Serial instance

Return

Value	Description
0	Successful closing of a serial port

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
ser = serial_open(port="COM", baudrate=115200, bytesize=DR_EIGHTBITS,
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

res = serial_write(ser, b"123456789")

serial_close(ser)
```

8.1.3 serial_state()

Definition

serial_state(ser)

Features

This function returns the status of a serial communication port.

Parameters

Parameter Name	Data Type	Default Value	Description
ser	serial.Serial	-	Serial instance

Return

Value	Description
1	Serial port opened
0	Serial port closed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
ser = serial_open(port="COM", baudrate=115200, bytesize=DR_EIGHTBITS,
        parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

state = serial_state(ser)

serial_close(ser)
```

8.1.4 serial_set_inter_byte_timeout()

Definition

serial_set_inter_byte_timeout(ser, timeout=None)

Features

This function sets the timeout between the bytes (inter-byte) when reading and writing to the port.

Parameters

Parameter Name	Data Type	Default Value	Description
ser	serial.Serial	-	Serial instance
timeout	float	None	Timeout between bytes during reading or writing • Continued processing of data that was processed before the timeout • None: inter-byte timeout not specified

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
ser = serial_open(port="COM", baudrate=115200, bytesize=DR_EIGHTBITS,
        parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

res = serial_set_inter_byte_timeout(ser, 0.1)
```

```
serial_close(ser)
```

8.1.5 serial_write()

Definition

serial_write(ser, tx_data)

Features

This function writes the data (tx_data) to a serial port.

Parameters

Parameter Name	Data Type	Default Value	Description
ser	serial.Serial	-	Serial instance
tx_data	byte	-	Data to be transmitted • The data type must be a byte. • Refer to the example below.

Return

Value	Description
0	Success
-1	The port is not open.
-2	serial.SerialException error occurred

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
ser = serial_open(port="COM", baudrate=115200, bytesize=DR_EIGHTBITS,
          parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

serial_write(ser, b"123456789") # b means the byte type.

# Convert string to byte
msg = “abcd”                # msg is a string variable
serial_write(ser, msg.encode()) # encode() converts string type to byte type

serial_close(ser)
```

8.1.6 serial_read()

Definition

serial_read(ser, length=-1, timeout=-1)

Features

This function reads the data from a serial port.

Parameters

Parameter Name	Data Type	Default Value	Description
ser	serial.Serial	-	Serial instance
length	int	-1	Number of bytes to read • -1: Not specified (The number of bytes to read is not specified) • n(>=0): The specified number of byte is read.
timeout	int float	-1	Read waiting time • -1: Indefinite wait • n(>0): n seconds

Return

Value(res, rx_data)	Description

res	n	Number of bytes of the received data
	-1	The port is not open.
	-2	serial.SerialException error occurred
rx_data		Number of bytes read (byte type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Example

```
ser = serial_open(port="COM", baudrate=115200, bytesize=DR_EIGHTBITS,
          parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)

res, rx_data = serial_read(ser)
#Wait indefinitely until data is received

res, rx_data = serial_read(ser, timeout=3)
#Wait until data is received, set a 3 seconds timeout
# If received within 3 seconds, the read data is returned immediately
# Return the value read so far after 3 seconds have elapsed

res, rx_data = serial_read(ser, length=100)
# Wait indefinitely until reading 100 bytes

res, rx_data = serial_read(ser, length=100, timeout=3)
#Wait until reading 100byte, set 3 seconds timeout
# If 100 bytes are received within 3 seconds, the read data is returned immediately.
# Return the value read so far after 3 seconds have elapsed

# Convert the received byte type to string type
rx_msg = rx_data.decode()
# rx_data is a byte type and decode() is used to convert it to a string type.
# For example, if rx_data = b”abcd”, then rx_msg=”abcd”.

res, rx_data = serial_close(ser)
```

8.1.7 serial_get_count()

Features

This function reads the number of devices connected to USB to Serial.

Return

Value (port_info, device_name)	Description
count	Number of connected serial ports

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Example

```
count = serial_get_count() # read number of connected serial ports

for i in range(count):
port_info, device_name = serial_get_info(i+1)
tp_popup("i={}, port ={}, dev ={}".format(i, port_info, device_name))
```

8.1.8 serial_get_info()

Definition

serial_get_info(id)

Features

This function reads the port information and device name of the connected USB to Serial.

Parameters

Parameter Name	Data Type	Default Value	Description
id	int	1	ID of "USB to Serial" to read (1-10)

Return

Value (port_info, device_name)	Description
port_info	Port information (NULL means no device is connected)
device_name	Device name (NULL means no device is connected)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Example

```
port_info, device_name = serial_get_info(1) #1 connected device information
#port_info = "COM_USB"
ser = serial_open(port=port_info, baudrate=115200, bytesize=DR_EIGHTBITS,
          parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)
```

8.1.9 Integrated Example - Serial

This is an example for performing a self-loop-back test on RXD (#2 pin) and

TXD (#3 pin) are connected with the serial port.

Example 1 : Self-loop-back test example

```
# serial port open
# if D-SUB (9pin) is connected: port="COM"
# if USB is connected with USB to Serial: port="COM_USB"
```

```
ser = serial_open(port="COM_USB", baudrate=115200, bytesize=DR_EIGHTBITS,
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)
wait(1)

# SEND DATA : "123ABC"
res = serial_write(ser, b"123ABC") # b means byte type
wait(1)

# READ DATA
res, rx_data = serial_read(ser)
# RXD and TXD are H/W connected res=6 (byte) rx_data = b"123ABC" are recevied

tp_popup("res ={0}, rx_data={1}".format(res, rx_data))

# close corresponding serial port
serial_close(ser)
```

Received data is collected as is and the result is outputted as a TP pop-up message.

If executed properly, it outputs a result of res=6 rx_data = b'123ABC'.

Example 2 : Various packet transmission examples

Transmission packet: "MEAS_START" +data1[4byte]+data2[4byte]

data1: Converts integer to 4 bytes ex) 1 → 00000001

data2: Converts integer to 4 bytes ex) 2 → 00000002

ex) In case data1=1, data2=2: "MEAS_START"+00000001+00000002

Actual Packet: 4D4541535F53544152540000000100000002

Received packet: res=18, rx_data="MEAS_START"+00000001+00000002

Extract rxd1 : Convert 10th to 14th byte into integer

Extract rxd2 : Convert 14th to 18th byte into integer

```
ser = serial_open(port="COM_USB", baudrate=115200, bytesize=DR_EIGHTBITS,
parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE)
wait(1)

send_data = b"MEAS_START" # b means byte type
data1 =1
data2 =2
send_data += (data1).to_bytes(4, byteorder='big')
send_data += (data2).to_bytes(4, byteorder='big')

# SEND DATA
res = serial_write(ser, send_data)
wait(1)
```

```
# READ DATA
# RXD, TXD are connected by H/W, so send_data is received as it is
res, rx_data = serial_read(ser)

tp_popup("res ={0}, rx_data={1}".format(res, rx_data))

rxd1 = int.from_bytes(rx_data[10:10+4], byteorder='big', signed=True)
rxd2 = int.from_bytes(rx_data[14:14+4], byteorder='big', signed=True)

tp_popup("res={0}, rxd1={1}, rxd2={2}".format(res, rxd1, rxd2))

#Close the serial port
serial_close(ser)
```

Connect the USB to serial device to the USB port and send byte type send_data.

Since RXD(2pin) and TXD(3pin) are connected to receive the transmitted data as it is,

res = 18, rx_data has the same packet as send_data.

Extract rxd1 : Convert 10th to 14th byte into integer

Extract rxd2 : Convert 14th to 18th byte into integer

The end result will be res=18, rxd1=1, rxd2=2

8.2 Flange I/O

8.2.1 flange_serial_open()

Definition

flange_serial_open(baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits = DR_STOPBITS_ONE)

Features

A command used for opening the Pseudo Flange Serial communication port.

After calling this function, a minimum delay of 100 milliseconds is required before the flange serial operates normally.

The characteristics of pseudo flange serial communication are different from normal serial communication. Therefore, handshaking communication is recommended. (e.g., modbus RTU) Due to the internal buffer size limit (255bytes) and internal delay, overflow may occur when used in sensors, etc.

(However, it is not supported by the new Flange I/O (M/H Series 2024.03.22 & A/E Series 2024.04.11 or later) and can be set in the Flange I/O tab of the Robot Parameter module.)

Parameters

Parameter Name	Data Type	Default Value	Description
baudrate	int	115200	Baud rate 2400, 4800, 9600, 19200, 38400, 57600, 115200, 1000000 etc
bytesize	int	8	Number of data bits • DR_FIVEBITS : 5 • DR_SIXBITS : 6 • DR_SEVENBITS : 7 • DR_EIGHTBITS : 8
parity	str	"N"	Parity checking • DR_PARITY_NONE: "N" • DR_PARITY_EVEN: "E" • DR_PARITY_ODD: "O"
stopbits	int	1	Number of stop bits • DR_STOPBITS_ONE =1 • DR_STOPBITS_TWO = 2

Return

Value	Description
0	Successful connection
Negative value	Fail to connection

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
flange_serial_open(baudrate=115200, bytesize=DR_EIGHTBITS,parity=DR_PARITY_NONE,
stopbits = DR_STOPBITS_ONE)
wait(0.1)
```

8.2.2 flange_serial_read()

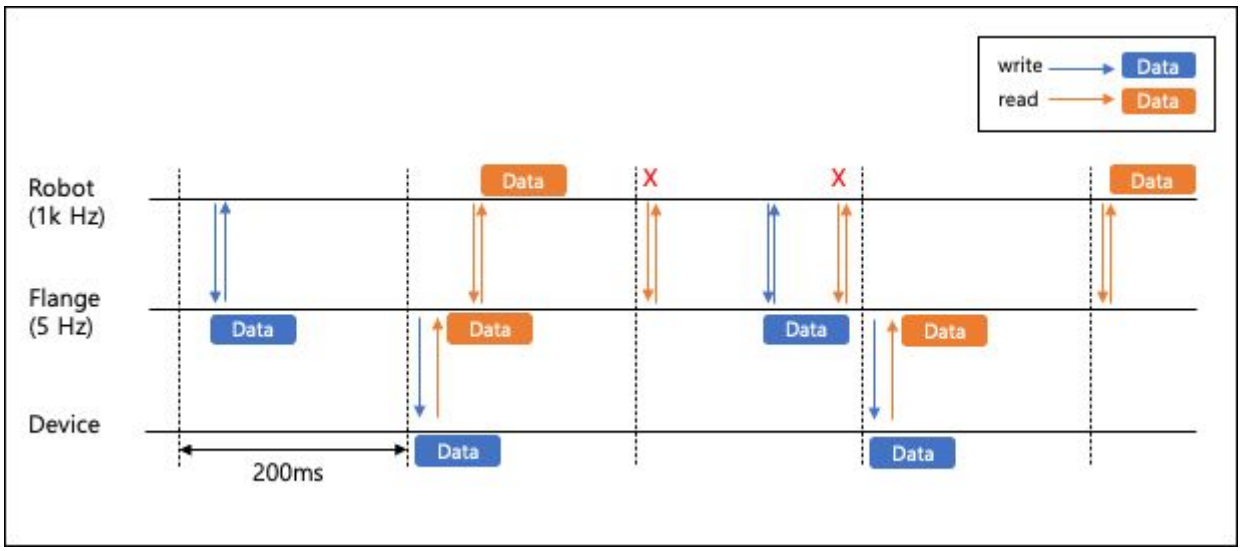
Definition

flange_serial_read(timeout=None, port=1)

Features

This function reads the data from a flange serial port.

When using the **Modbus RTU** communication method, a minimum delay of **250 milliseconds** is required after a write operation before receiving a read response. Instead of repeatedly calling `flange_serial_read()` or using the `wait()` function for the delay, you may add a timeout parameter to the function to handle the wait internally. However, the `timeout` value should be set slightly higher (e.g., 1 second) to ensure successful data reception within the minimum required time.



Parameters

Parameter Name	Data Type	Default Value	Description
timeout	float int	None	Read waiting time
port	int	1	Port number to read X1 Port: 1 X2 Port: 2 (Not available for A Series)

Return

Value	Description
res	Number of bytes of the received data -1 : time out -2 : overflow
rx_data	Number of bytes read (byte type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Sample 1. Wait using wait
# 2F-85 Gripper - Close the Gripper at full speed and full force
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x09\x00\x00\xFF\xFF\xFF"))
wait(0.25)
res, data = flange_serial_read(1,1)
```

```
# Sample 2. Wait using timeout
# 2F-85 Gripper - Close the Gripper at full speed and full force
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x09\x00\x00\xFF\xFF\xFF"))
res, data = flange_serial_read(1,1)
```

8.2.3 flange_serial_write()

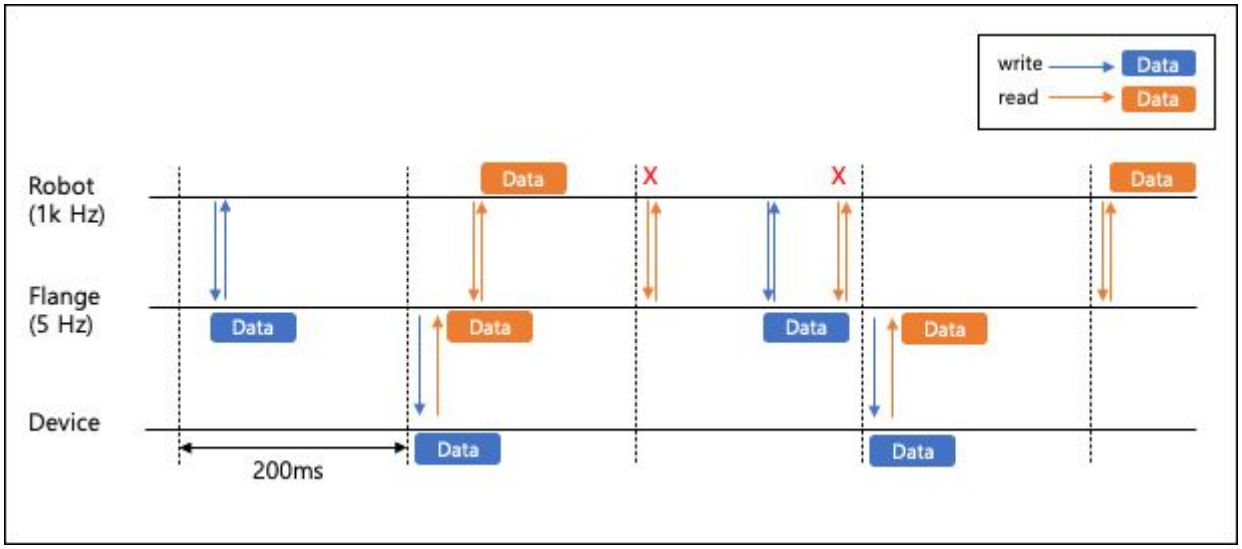
Definition

flange_serial_write(tx_data, port=1)

Features

This function records the data (data) to a flange serial port.

After invoking this function, you must wait at least 200 milliseconds before invoking it again.
In DRL, use the `wait(0.2)` command on the following line to enforce the delay.



Parameters

Parameter Name	Data Type	Default Value	Description
tx_data	byte	-	Data to be transmitted (max 32byte) • The data type must be a byte. • Refer to the example below.

Parameter Name	Data Type	Default Value	Description
port	int	1	Port number to write X1 Port: 1 X2 Port: 2 (Not available for A Series)

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# 2F-85 Gripper Examples
# Activation Request(clear Act)
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x00\x00\x00\x00\x
00\x00"))
wait(0.2)
# Activation Request(set Act)
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x01\x00\x00\x00\x
00\x00"))
wait(0.2)
```

8.2.4 flange_serial_close()

Features

This function closes a flange serial communication port.

(However, it is not supported by the new Flange I/O (M/H Series 2024.03.22 & A/E Series 2024.04.11 or later) and can be set in the Flange I/O tab of the Robot Parameter module.)

Return

Value	Description
0	Success

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

8.2.5 Integrated Example - Serial(Flange I/O)

Example 1) Controls a Robotiq 2F using Pseudo Flange Serial

```
# Sample 2F-85 Gripper
flange_serial_open(baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE,
stopbits = DR_STOPBITS_ONE)
wait(0.1)

# Step1-1:Activation Request(clear Act)
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x00\x00\x00\x00\x
00\x00"))
wait(0.25)
res, data = flange_serial_read()

# Step1-2:Activation Request(set Act)
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x01\x00\x00\x00\x
00\x00"))
```

```
wait(0.25)
res, data = flange_serial_read()

# Step 2: Read Gripper status until the activation is completed
flange_serial_write(modbus_send_make(b"\x09\x03\x07\xD0\x00\x01"))
wait(0.25)
res, data = flange_serial_read()

# Step 3: Move the robot to the pick-up location
wait(1)

# Step 4: Close the Gripper at full speed and full force
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x09\x00\x00\xFF\x
FF\xFF"))
wait(0.25)
res, data = flange_serial_read()

# Step 5: Read Gripper status until the grasp is completed
flange_serial_write(modbus_send_make(b"\x09\x03\x07\xD0\x00\x03"))
wait(0.25)
res, data = flange_serial_read()

# Step 6: Move the robot to the release location
wait(1)

# Step 7: Open the Gripper at full speed and full force
flange_serial_write(modbus_send_make(b"\x09\x10\x03\xE8\x00\x03\x06\x09\x00\x00\x00\x
FF\xFF"))
wait(0.25)
res, data = flange_serial_read()

# Step 8: Read Gripper status until the opening is completed
flange_serial_write(modbus_send_make(b"\x09\x03\x07\xD0\x00\x03"))
wait(0.25)
res, data = flange_serial_read()

flange_serial_close()
```

Example 2) Controls a onRobot RC6 V2 using New Flange Serial

```
# Sample RC6 V2 Gripper
# select port (1,2)
# If your robot is A, you can select 1 only
port = 1

flange_serial_write(modbus_send_make(b"\x41\x06\x00\x00\x01\x2c"), port)
res, data = flange_serial_read(1,port )
wait(0.2)

flange_serial_write(modbus_send_make(b"\x41\x06\x00\x01\x03\xe8"), port )
```

```
res, data = flange_serial_read(1,1)
wait(0.2)

flange_serial_write(modbus_send_make(b"\x41\x06\x00\x02\x00\x01"), port )
res, data = flange_serial_read(1,port )
wait(0.2)

flange_serial_write(modbus_send_make(b"\x41\x06\x00\x00\x01\x2c"), port )
res, data = flange_serial_read(1,port )
wait(0.2)

flange_serial_write(modbus_send_make(b"\x41\x06\x00\x01\x00\x00"), port )
res, data = flange_serial_read(1,port )
wait(0.2)

flange_serial_write(modbus_send_make(b"\x41\x06\x00\x02\x00\x01"), port )
res, data = flange_serial_read(1,port )
wait(0.2)
```

8.3 Tcp/Client

8.3.1 client_socket_open()

Definition

client_socket_open(ip, port)

Features

This function creates a socket and attempts to connect it to a server (ip, port).

It returns the connected socket when the client is connected.

Parameters

Parameter Name	Data Type	Default Value	Description
ip	str	-	Server IP address: (E.g.) “192.168.137.200”
port	int	-	Server Port number (e.g.) 20002

Return

Value	Description
socket.socket instance	Successful connection

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	socket.error occurred during a connection

Example

```
sock = client_socket_open("192.168.137.200", 20002)
# An indefinite connection is attempted to the server (ip="192.168.137.200",
port=20002).
# The connected socket is returned if the connection is successful.
# The data is read, written, and closed using the returned socket as shown below.

client_socket_write(sock, b"123abc")   # Sends data to the server (b represents the
byte type).
res, rx_data = client_socket_read(sock) # Receives the data from the server.
client_socket_close(sock)              # Closes the connection to the server.
```

8.3.2 client_socket_close()

Definition

client_socket_close(sock)

Features

This function terminates communication with the server. To reconnect to the server, the socket must be closed with client_socket_close(sock) and reopened.

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from client_socket_open()

Return

Value	Description
0	Successful disconnection

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_RUNTIME)	socket.error occurred during disconnection

Example

```
sock = client_socket_open("192.168.137.200", 20002)
# An indefinite connection is attempted to the server (ip="192.168.137.200",
port=20002).

# do something…

client_socket_close(sock)       # Closes the connection to the server.
```

8.3.3 client_socket_state()

Definition

client_socket_state(sock)

Features

This function returns the socket connection status.To know the connection status with the server, check the return value of client_socket_read or client_socket_write (see Example 2).

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from client_socket_open()

Return

Value	Description
1	Socket normal state
0	Socket abnormal state

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
sock = client_socket_open("192.168.137.200", 20002)

state = client_socket_state(sock) # Reads the socket state.

client_socket_close(sock)
```

Example 2

```
sock = client_socket_open("192.168.137.200", 20002)
res, rx_data =client_socket_read(sock)
tp_log("[RX] res={0}, rx_data ={1}".format(res, rx_data))
if (res < 0):
   tp_log("[RX] server disconnect") #When the server connection is disconnected
   client_socket_close(sock)
   exit()

client_socket_close(sock)
```

8.3.4 client_socket_read()

Definition

client_socket_read(sock, length=-1, timeout=-1)

Features

This function receives data from the server.

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from client_socket_open()
length	int	-1	Number of bytes of the received data • -1 : Not specified (The number of bytes to read is not specified.) • n(>=0) : The specified number of bytes is read.
timeout	int float	-1	Waiting time for receipt • -1 : Indefinite wait • n(>0) : n seconds

Return

Value (res, rx_data)		Description
res	>0	Number of bytes of the received data
	-1	The server is not connected.
	-2	Socket.error occurred during data reception
	-3	Timeout during data reception
rx_data		Received data (byte type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Example

```
sock = client_socket_open("192.168.137.200", 20002)

res, rx_data = client_socket_read(sock)   # Indefinite wait until the data is
received
# Reads all received data since the length is omitted.
# Waits indefinitely until the data is received since timeout is omitted.
# (res = size of received data, rx_data=received data) is returned when the data is
received.

res, rx_data = client_socket_read(sock, timeout=3) # Waits for up to 3 seconds until
the data is received.
# (res = size of received data, rx_data=received data) is returned if the data is
received within 3 seconds.
# (res = -3, rx_data=None) is returned if the data is not received within 3 seconds.

res, rx_data = client_socket_read(sock, length=64)   # Reads 64 bytes of the received
data.

res, rx_data = client_socket_read(sock, length=64, timeout=3)
# Reads 64 bytes of the received data within the 3-second timeout.

rx_msg = rx_data.decode() # rx_data is a byte type and can be converted to a string
type
                          # using decode().
                          # For example, if rx_data = b"abcd",
                          # rx_msg= "abcd".
client_socket_close(sock)
```

8.3.5 client_socket_write()

Definition

client_socket_write(sock, tx_data)

Features

This function transmits data to the server.

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from client_socket_open()
tx_data	byte	-	Data to be transmitted • The data type must be a byte. • Refer to the example below.

Return

Value	Description
0	Success
-1	The server is not connected.
-2	Server is disconnected, or socket.error occurred during a data transfer

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
sock = client_socket_open("192.168.137.200", 20002)

client_socket_write(sock, b"1234abcd") # b means the byte type.

msg = "abcd" # msg is a string variable.
client_socket_write(sock, msg.encode()) # encode() converts a string type to a byte
type.
```

```
client_socket_close(sock)
```

8.3.6 Integrated example (Tcp/Client)

Assume that server IP = 192.168.137,200 and open port =20002

and that the received packets are sent to the client as they are (mirroring).

Example 1 : Example of a default TCP client

```
# Assume server IP = 192.168.137,200 and open port =20002.
g_sock = client_socket_open("192.168.137.200", 20002)

tp_popup("connect O.K!",DR_PM_MESSAGE)
while 1:
 client_socket_write(g_sock, b"abcd") # The string "abcd" is sent in a byte type.
 wait(0.1)
 res, rx_data = client_socket_read(g_sock) # Waits for the data from the server.
 tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
 wait(0.1)
```

The example connects to the server and sends the string "abcd". (b converts the string to a byte type.)

The message received from the server is output to the TP.

res = 4 and rx_data=b"abcd" since the server transmits the received data as is.

Example 2 : Examples of a packet transfer

Transmission packet: “MEAS_START" +data1[4byte]+data2[4byte]

- data1: Conversion of the integer to 4 byte. ex) 1 → 00000001
- data2: Conversion of the integer to 4 byte. ex) 2 → 00000002

ex) data1=1 and data2=2: “MEAS_START"+00000001+00000002

- Actual packet: 4D4541535F535441525400000001000000002
- Received packet: res=18, rx_data=“MEAS_START"+00000001+00000002
    - rxd1 extraction: Conversion of 10th - 14th bytes to an integer
    - rxd2 extraction: Conversion of 14th - 18th bytes to an integer

```
g_sock = client_socket_open("192.168.137.100", 20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

send_data = b"MEAS_START"
data1 =1
data2 =2
```

```
send_data += (data1).to_bytes(4, byteorder='big')
send_data += (data2).to_bytes(4, byteorder='big')

client_socket_write(g_sock, send_data)

wait(0.1)

res, rx_data = client_socket_read(g_sock)
tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)

rxd1 = int.from_bytes(rx_data[10:10+4], byteorder='big', signed=True)
rxd2 = int.from_bytes(rx_data[14:14+4], byteorder='big', signed=True)

tp_popup("res={0}, rxd1={1}, rxd2={2}".format(res, rxd1, rxd2), DR_PM_MESSAGE)

client_socket_close(g_sock)
```

The example connects to the server and sends a byte type send_data.

res = 18 and rx_data=send_data since the server transmits the received data as is.

- rxd1 extraction: Conversion of 10th - 14th bytes to an integer
- rxd2 extraction: Conversion of 14th - 18th bytes to an integer

The final result is res=18, rxd1=1, and rxd2=2.

Example 3 : Reconnection

```
def fn_reconnect():
 global g_sock
 client_socket_close(g_sock)
 g_sock = client_socket_open("192.168.137.200", 20002)
 return

g_sock = client_socket_open("192.168.137.200", 20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

client_socket_write(g_sock, b"abcd")
wait(0.1)

while 1:
 res, rx_data = client_socket_read(g_sock)
 if res < 0:
     fn_reconnect()
 else:
     tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
 wait(0.1)
```

The example checks the return value of the client_socket_read() command.

A negative value is returned if the connection to the server is terminated or there is a communication problem.

The function reconnect() is called to attempt a reconnection if a negative value is returned.

Note that the opened socket is closed when a reconnection is attempted.

8.4 Tcp/Server

8.4.1 server_socket_open()

Definition

server_socket_open(port)

Features

The robot controller creates a server socket and waits for the connection to the client. Returns the connected socket when the client is connected.

Parameters

Parameter Name	Data Type	Default Value	Description
port	int	-	Port number to open

Return

Value	Description
socket.socket instance	Successful connection

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	socket.error occurred during a connection

Example

```
sock = server_socket_open(20002)
# Opens the port 20002 and waits until the client connects.
# The connected socket is returned if the connection is successful.
# The data is read, written, and closed using the returned socket as shown below.

server_socket_write(sock, b"123abc") # Sends data to the client (b represents the
byte type).
res, rx_data = server_socket_read(sock) # Receives the data from the client.

server_socket_close(sock) # Closes the connection to the client.
```

8.4.2 server_socket_close()

Definition

server_socket_close(sock)

Features

This function terminates communication with the client. To reconnect to the client, the socket must be closed with server_socket_close(sock) and reopened.

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from server_socket_open() socket instance

Return

Value	Description
0	Successful disconnection

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_RUNTIME)	socket.error occurred during disconnection

Example

```
sock = server_socket_open(20002)
# Opens the port 20002 and waits until the client connects.

# do something…

server_socket_close(sock) ) # Closes the connection to the client.
```

8.4.3 server_socket_state()

Definition

server_socket_state(sock)

Features

This function returns the socket status.

To know the connection status with the client, check the return value of server_socket_read or server_socket_write (see Example 2).

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from server_socket_open() socket instance

Return

Value	Description
1	Socket normal state
0	Socket abnormal state

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example 1

```
sock = server_socket_open(20002)

state = server_socket_state(sock) # Reads the socket state.

server_socket_close(sock)
```

Example 2

```
sock = server_socket_open(20002)

res, rx_data =server_socket_read(sock)
tp_log("[RX] res={0}, rx_data ={1}".format(res, rx_data))
if (res < 0):   #When the client connection is disconnected
   tp_log("[RX] client disconnect")
   server_socket_close(sock)
   exit()

server_socket_close(sock)
```

8.4.4 server_socket_read()

Definition

server_socket_read(sock, length=-1, timeout=-1)

Features

This function reads data from the client.

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from server_socket_open() socket instance
length	int	-1	Number of bytes of the received data • -1 : Not specified (The number of bytes to read is not specified.) • n(>=0) : The specified number of bytes is read.
timeout	int float	-1	Waiting time for receipt • -1 : Indefinite wait • n(>0) : n seconds

Return

Value (res, rx_data)		Description
res	0	Number of bytes of the received data
	-1	The client is not connected.
	-2	socket.error occurred during data reception
	-3	Timeout during data reception
rx_data		Received data (byte type)

Example

```
sock = server_socket_open(20002)

res, rx_data = server_socket_read(sock)   # Indefinite wait until the data is
received
# Reads all received data since the length is omitted.
```

```
# Waits indefinitely until the data is received since timeout is omitted.
# (res = size of received data, rx_data=received data) is returned when the data is
received.

res, rx_data = server_socket_read(sock, timeout=3) # Waits for up to 3 seconds until
the data is received.
# (res = size of received data, rx_data=received data) is returned if the data is
received within 3 seconds.
# (res = -3, rx_data=None) is returned if the data is not received within 3 seconds.

res, rx_data = server_socket_read(sock, length=64)   # Reads 64 bytes of the received
data.

res, rx_data = server_socket_read(sock, length=64, timeout=3)
# Reads 64 bytes of the received data within the 3-second timeout.

rx_msg = rx_data.decode() # rx_data is a byte type and can be converted to a string
type
                          # using decode().
                          # For example, if rx_data = b"abcd",
                          # rx_msg= "abcd".

server_socket_close(sock)
```

8.4.5 server_socket_write()

Definition

server_socket_write(sock, tx_data)

Features

This function transmits data to the client.

Parameters

Parameter Name	Data Type	Default Value	Description
sock	socket.socket	-	Socket instance returned from server_socket_open() socket instance
tx_data	byte	-	Data to be transmitted • The data type must be a byte. • Refer to the example below.

Return

Value	Description
0	Success
-1	The client is not connected.
-2	client is disconnected, or socket.error occurred during a data transfer

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Example

```
sock = server_socket_open(20002)

server_socket_write(sock, b"1234abcd") # b means the byte type.

msg = "abcd"    # msg is a string variable.
server_socket_write(sock, msg.encode()) # encode() converts a string type to a byte
type.

server_socket_close(sock)
```

8.4.6 Integrated example - Tcp/Server

The example assumes that the client connects to the controller with IP = 192,168,137.100 and port = 20002

and that the received packets are sent to the server as they are (mirroring).

Example 1 : Default TCP server example

```
g_sock = server_socket_open(20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

while 1:
 server_socket_write(g_sock, b"abcd") # The string "abcd" is sent in a byte type.
```

```
wait(0.1)
res, rx_data = server_socket_read(g_sock) # Waits for the data from the server.
tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
wait(0.1)
```

The example opens the port 20002 and waits until the client connects.

It connects to the client and sends the string "abcd".

The message received from the client is output to the TP.

res = 4 and rx_data=b"abcd" since the client transmits the received data as is.

Example 2 : Examples of a packet transfer

Transmission packet: “MEAS_START" +data1[4byte]+data2[4byte]

data1: Conversion of the integer to 4 byte. ex) 1 → 00000001

data2: Conversion of the integer to 4 byte. ex) 2 → 00000002

ex) data1=1 and data2=2: “MEAS_START"+00000001+00000002

Actual packet: 4D4541535F535441525400000001000000002

Received packet: res=18, rx_data=“MEAS_START"+00000001+00000002

rxd1 extraction: Conversion of 10th - 14th bytes to an integer

rxd2 extraction: Conversion of 14th - 18th bytes to an integer

```
g_sock = server_socket_open(20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

send_data = b"MEAS_START"
data1 =1
data2 =2
send_data += (data1).to_bytes(4, byteorder='big')
send_data += (data2).to_bytes(4, byteorder='big')

server_socket_write(g_sock, send_data)

wait(0.1)

res, rx_data = server_socket_read(g_sock)
tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)

rxd1 = int.from_bytes(rx_data[10:10+4], byteorder='big', signed=True)
rxd2 = int.from_bytes(rx_data[14:14+4], byteorder='big', signed=True)

tp_popup("res={0}, rxd1={1}, rxd2={2}".format(res, rxd1, rxd2), DR_PM_MESSAGE)
```

```
server_socket_close(g_sock)
```

The example sends the byte type send_data.

res = 18 and rx_data=send_data since the client transmits the received data as is.

rxd1 extraction: Conversion of 10th - 14th bytes to an integer

rxd2 extraction: Conversion of 14th - 18th bytes to an integer

The final result is res=18, rxd1=1, and rxd2=2.

Example 3 : Reconnection

```
def fn_reopen():
 global g_sock
 server_socket_close(g_sock)
 g_sock = server_socket_open(20002)
 return

g_sock = server_socket_open(20002)
tp_popup("connect O.K!",DR_PM_MESSAGE)

server_socket_write(g_sock, b"abcd")
wait(0.1)

while 1:
 res, rx_data = server_socket_read(g_sock)
 if res < 0:
     fn_reopen()
 else:
     tp_popup("res={0}, rx_data ={1}".format(res, rx_data), DR_PM_MESSAGE)
 wait(0.1)
```

The example checks the return value of the server_socket_read() command.

A negative value is returned if the connection to the client is terminated or there is a communication problem.

The function reopen() is called to wait for the client connection if a negative value is returned.

Note that the opened socket is closed when a reconnection is attempted.

8.5 Modbus

8.5.1 add_modbus_signal()

Definition

add_modbus_signal(ip, port, name, reg_type, index, value=0, slaveid=255)

Features

This function registers the ModbusTCP signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

Parameters

Parameter Name	Data Type	Default Value	Description
ip	string	-	IP address of the ModbusTCP module
port	int	-	Port number of the ModbusTCP module
name	string	-	Modbus signal name
reg_type	int	-	Modbus signal type • DR_MODBUS_DIG_INPUT • DR_MODBUS_DIG_OUTPUT • DR_MODBUS_REG_INPUT • DR_MODBUS_REG_OUTPUT
index	int	-	Modbus signal index
value	int	0	Output when the type is DR_COIL or DR_HOLDING_REGISTER (ignored otherwise)
slaveid	int	255	• Slave ID of the ModbusTCP module (0 or 1-247 or 255) 0 : Broadcase address 255 : Default value for ModbusTCP

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# An example of connecting two Modbus IO and allocating the contacts
# Modbus IO 1 : IP address 192.168.137.254, port 502
Input Register Address 0 ~ 3, signal name “input1”~”input4”
Holding Register Addreess 0 ~ 3, signal name “output1”~”output4”
# Modbus IO 2 : IP address 192.168.137.253, port 502
Input Register Address 0 ~ 3, signal name “input1”~”input4”
Holding Register Addreess 0 ~ 3, signal name “output1”~”output4”

# set < Modbus IO 1 > “input1”~”input4”, “output1”~”output4”
add_modbus_signal(ip="192.168.137.254",port=502, name="input1",
reg_type=DR_INPUT_REGISTER, index=0, slaveid=255)
add_modbus_signal(ip="192.168.137.254",port=502, name="input2",
reg_type=DR_INPUT_REGISTER, index=1, slaveid=255)
add_modbus_signal(ip="192.168.137.254",port=502, name="input3",
reg_type=DR_INPUT_REGISTER, index=2, slaveid=255)
add_modbus_signal(ip="192.168.137.254",port=502, name="input4",
reg_type=DR_INPUT_REGISTER, index=3, slaveid=255)

add_modbus_signal(ip="192.168.137.254",port=502, name="output1",
reg_type=DR_HOLDING_REGISTER, index=0, value=0, slaveid=255)
add_modbus_signal(ip="192.168.137.254",port=502, name=" output2",
reg_type=DR_HOLDING_REGISTER, index=1, value=0, slaveid=255)
```

```
add_modbus_signal(ip="192.168.137.254",port=502, name=" output3",
reg_type=DR_HOLDING_REGISTER, index=2, value=0, slaveid=255)
add_modbus_signal(ip="192.168.137.254",port=502, name=" output4",
reg_type=DR_HOLDING_REGISTER, index=3, value=0, slaveid=255)

# set < Modbus IO 1 > “input5”~”input8”, “output5”~”output8”
add_modbus_signal(ip="192.168.137.253",port=502, name="input5",
reg_type= DR_INPUT_REGISTER, index=0, slaveid=255)
add_modbus_signal(ip="192.168.137.253",port=502, name=" input6",
reg_type= DR_INPUT_REGISTER, index=1, slaveid=255)
add_modbus_signal(ip="192.168.137.253",port=502, name=" input7",
reg_type= DR_INPUT_REGISTER, index=2, slaveid=255)
add_modbus_signal(ip="192.168.137.253",port=502, name=" input8",
reg_type= DR_INPUT_REGISTER, index=3, slaveid=255)

add_modbus_signal(ip="192.168.137.253",port=502, name=" output5",
reg_type=DR_HOLDING_REGISTER, index=0, value=0, slaveid=255)
add_modbus_signal(ip="192.168.137.253",port=502, name=" output6",
reg_type=DR_HOLDING_REGISTER, index=1, value=0, slaveid=255)
add_modbus_signal(ip="192.168.137.253",port=502, name=" output7",
reg_type=DR_HOLDING_REGISTER, index=2, value=0, slaveid=255)
add_modbus_signal(ip="192.168.137.253",port=502, name=" output8",
reg_type=DR_HOLDING_REGISTER, index=3, value=0, slaveid=255)
```

8.5.2 add_modbus_rtu_signal()

Definition

add_modbus_rtu_signal(slaveid=1, port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name, reg_type, index, value=0)

Features

This function registers the ModbusRTU signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

Parameters

Parameter Name	Data Type	Default Value	Description
slaveid	int	1	Slave ID of the ModbusRTU (0 or 1-247) 0 : Broadcase address
port	string	None	E.g.) “COM”, “COM_USB”, “/dev/ttyUSB0”

Parameter Name	Data Type	Default Value	Description
baudrate	int	115200	Baud rate 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
bytesize	int	8	Number of data bits • DR_EIGHTBITS: 8
parity	string	"N"	Parity checking • DR_PARITY_NONE: "N" • DR_PARITY_EVEN: "E" • DR_PARITY_ODD: "O"
stopbits	int	1	Number of stop bits • DR_STOPBITS_ONE =1 • DR_STOPBITS_TWO = 2
name	string	-	Modbus signal name
reg_type	int	-	Modbus signal type • DR_DISCRETE_INPUT =DR_MODBUS_DIG_INPUT • DR_COIL =DR_MODBUS_DIG_OUTPUT • DR_INPUT_REGISTER =DR_MODBUS_REG_INPUT • DR_HOLDING_REGISTER =DR_MODBUS_REG_OUTPUT
index	int	-	Modbus signal index
value	int	0	Output when the type is DR_COIL or DR_HOLDING_REGISTER (ignored otherwise)

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
add_modbus_rtu_signal(slaveid=1, port=port_info, baudrate=115200,
bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name='di1',
reg_type=DR_INPUT_REGISTER, index=0)
add_modbus_rtu_signal(slaveid=1, port=port_info, baudrate=115200,
bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name='do1',
reg_type=DR_HOLDING_REGISTER, index=0, value=12345)
```

8.5.3 add_modbus_signal_multi()

Definition

add_modbus_signal_multi(ip, port, slaveid=255, name=None, reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=1)

Features

This function registers the ModbusTCP FC15 & FC16 multiple signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

Note
Initial value setting function is not supported.

Parameters

Parameter Name	Data Type	Default Value	Description
ip	string	-	IP address of the ModbusTCP module
port	int	-	Port number of the ModbusTCP module
slaveid	int	255	• Slave ID of the ModbusTCP module (0 or 1-247 or 255) 0 : Broadcase address 255 : Default value for ModbusTCP
name	string	None	Modbus signal name
reg_type	int	DR_HOLDING_REGISTER	Modbus signal type • DR_COIL =DR_MODBUS_DIG_OUTPUT • DR_HOLDING_REGISTER =DR_MODBUS_REG_OUTPUT
start_address	int	0	Start address of Modbus multiple signal
cnt	int	1	Count of Modbus multiple signal

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
add_modbus_signal_multi(ip="192.168.137.200", port=502, slaveid=255, name="multi",
reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=5)
```

8.5.4 add_modbus_rtu_signal_multi()

Definition

add_modbus_rtu_signal_multi(slaveid=1, port=None, baudrate=115200, bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name=None, reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=1)

Features

This function registers the ModbusRTU FC15 & FC16 multiple signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

Note
Initial value setting function is not supported.

Parameters

Parameter Name	Data Type	Default Value	Description
slaveid	int	1	Slave ID of the ModbusRTU (0 or 1-247) 0 : Broadcase address
port	string	None	E.g.) “COM”, “COM_USB”, “/dev/ttyUSB0”
baudrate	int	115200	Baud rate 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200

Parameter Name	Data Type	Default Value	Description
bytesize	int	8	Number of data bits • DR_EIGHTBITS: 8
parity	string	"N"	Parity checking • DR_PARITY_NONE: "N" • DR_PARITY_EVEN: "E" • DR_PARITY_ODD: "O"
stopbits	int	1	Number of stop bits • DR_STOPBITS_ONE =1 • DR_STOPBITS_TWO = 2
name	string	-	Modbus signal name
reg_type	int	DR_HOLDING_REGISTER	Modbus signal type • DR_COIL =DR_MODBUS_DIG_OUTPUT • DR_HOLDING_REGISTER =DR_MODBUS_REG_OUTPUT
start_address	int	0	Start address of Modbus multiple signal
cnt	int	1	Count of Modbus multiple signal

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
add_modbus_rtu_signal_multi(slaveid=1, port="COM", baudrate=115200,
bytesize=DR_EIGHTBITS, parity=DR_PARITY_NONE, stopbits=DR_STOPBITS_ONE, name="multi",
reg_type=DR_HOLDING_REGISTER, start_address=0, cnt=5)
```

8.5.5 del_modbus_signal()

Definition

del_modbus_signal(name)

Features

This function deletes the registered Modbus signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

Parameters

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the registered Modbus signal

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Use the following command when the Modbus IO signals are registered as "input1" and
"output1"
# and this signal registration is to be deleted. .
del_modbus_signal("input1")          # Deletes the registered " input1" contact
del_modbus_signal("output1")         # Deletes the registered " output1" contact
```

8.5.6 del_modbus_signal_multi()

Definition

del_modbus_signal_multi(name)

Features

This function deletes the registered Modbus multiple signal. The Modbus I/O must be set in the Teach Pendant I/O set-up menu. Use this command only for testing if it is difficult to use the Teach Pendant. The Modbus menu is disabled in the Teach Pendant if it is set using this command.

Parameters

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the registered Modbus multiple signal

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Use the following command when the Modbus multiple signal is registered as
"multi1"(cnt=5)
# and this signal registration is to be deleted. .
del_modbus_signal_multi("multi1")    # Deletes the registered "multi1" contact
```

8.5.7 set_modbus_output()

Definition

set_modbus_output(iobus, value)

Features

This function sends the signal to an external Modbus system.

Function Code 05 Write Single Coil Register

Function Code 06 Write Signle Holding Register

Parameters

Parameter Name	Data Type	Default Value	Description
iobus	string	-	modbus name (set in the TP)
value	int	-	Value for Modbus coil register. • ON : 1 • OFF : 0
			Value for Modbus holding register

Note
When registered as a multiple signal, it is available by adding address value index to signal name.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Modbus Coil Registers are registered as "do1" and "do2".
```

```
set_modbus_output("do1", ON)
set_modbus_output("do2", OFF)

#Modbus Holding Registers are registered as "reg1" and "reg2".
set_modbus_output("reg1", 10)
set_modbus_output("reg2", 24)

#Modbus multiple signal is registered as “multi”(start address=10, cnt=2).
#“multi_10” & “multi_11” abailable
set_modbus_output("multi_10", 24)
set_modbus_output("multi_11", 65535)
```

8.5.8 set_modbus_outputs()

Definition

set_modbus_outputs(iobus_list, val_list)

Features

This function sends multiple signals to the Modbus Slave unit.

The maximum number of outputs is 32.

Parameters

Parameter Name	Data Type	Default Value	Description
iobus	string	-	Modbus name (set in the TP)
val_list	int	-	I/O output value list

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Modbus digital I/O output contact "d1" OFF, "d2" ON, and "d3" ON
set_modbus_outputs(iobus_list=[ "d1", "d2", "d3",], val_list=[0,1,1])

# Modbus digital I/O output contact "d3" OFF and "d4" ON
set_modbus_outputs(["d3", "d4"], [0,1])
```

8.5.9 set_modbus_output_multi()

Definition

set_modbus_output_multi(iobus, val_list)

Features

This function sends the signal to an external Modbus system.

Function Code 15 Write Multiple Coil Register

Function Code 16 Write Multiple Holding Register

Parameters

Parameter Name	Data Type	Default Value	Description
iobus	string	-	Modbus multiple signal name (set in the TP)
val_list	list		Value list of modbus multiple signal

Note

An error occurs if the number of signals registered in the multiple signal name does not match the number of elements in the output value list.

Return

Value	Description
0	Success
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Modbus Coil Registers are registered as "do1"(cnt=5), "do2"(cnt=3)
set_modbus_output_multi("do1", [ON, OFF, ON, ON, ON])
set_modbus_output_multi("do2", [ON, ON, ON])

#Modbus Holding Registers are registered as "reg1"(cnt=5), "reg2"(cnt=3)
set_modbus_output_multi("reg1", [10, 101, 12345, 777, 555])
set_modbus_output_multi("reg2", [24, 25, 26])
```

8.5.10 get_modbus_input()

Definition

get_modbus_input(iobus)

Features

This function reads the signal from the Modbus Slave unit.

Parameters

Parameter Name	Data Type	Default Value	Description
iobus	string	-	Modbus name (set in the TP)

Note
When registered as a multiple signal, it is available by adding address value index to signal name.

Return

Value	Description
0 or 1 or value	Modbus Descrete / Coil Register: ON or OFF Modbus Input / Holding Register : Register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Modbus digital I/O is connected, and the signals are registered as "di1" and "di2".
get_modbus_input("di1")
get_modbus_input("di2")

#Modbus analog I/O is connected, and the signals are registered as "reg1" and "reg2".
get_modbus_input("reg1")
```

```
get_modbus_input("reg2")

#Modbus multiple signal is registered as “multi”(start address=10, cnt=2).
#“multi_10” & “multi_11” abailable
get_modbus_ input ("multi_10")
get_modbus_ input ("multi_11")
```

8.5.11 get_modbus_inputs()

Definition

get_modbus_inputs(iobus_list)

Features

This function reads multiple signals from the Modbus Slave unit.

Parameters

Parameter Name	Data Type	Default Value	Description
iobus_list	list(string)	-	Modbus input name list (set in the TP) signal type can be used only in the following cases • DR_MODBUS_DIG_INPUT • DR_MODBUS_DIG_OUTPUT

Return

Value	Description
int (>=0)	Multiple signals to be read at once (the value of the combination of iobus_list where the first value is LSB and the last value is MSB)
Negative number	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Modbus digital I/O input signal "di1" =OFF, "di2"=ON, and "di3" =ON
res = get_modbus_inputs(iobus_list=["di1", "di2", "di3"])
      #res expected value = 0b110 (binary number), 6 (decimal number), or 0x06
(hexadecimal number)

# Modbus digital I I/O input signal "di4" OFF and "di5" ON
res = get_modbus_inputs(["di4", "di5"])
      #res expected value = 0b10 (binary number), 2 (decimal number), or 0x02
(hexadecimal number)
```

8.5.12 get_modbus_inputs_list()

Definition

get_modbus_inputs_list(iobus_list)

Features

It is the command for reading multiple register type open signals from an external Modbus Slave unit.

Parameter

Parameter Name	Data Type	Default Value	Description
iobus_list	list(string)	-	Modbus input name list (configured at TP) Available only with the following signal types • DR_MODBUS_REG_INPUT • DR_MODBUS_REG_OUTPUT

Return

Value	Description
res	Number values read
val_list	List of multiple signal values read simultaneously

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
# Modbus Register I/O "Holding1" is 1234, "Input1" is 567,
# and "Holding2" is 9876
res, val_list = get_modbus_inputs_list(iobus_list=[ "Holding1", "Input1",
"Holding2"])
                #res expected value = 3
                #val_list expected value = [1234, 567, 9876]
```

8.5.13 get_modbus_input_multi()

Definition

get_modbus_input_multi(iobus)

Features

This function reads the signal from the Modbus Slave unit.

Parameters

Parameter Name	Data Type	Default Value	Description
iobus	string	-	Modbus multi signal name (set in the TP)

Return

Value	Description
list	List of values corresponding to the number of signals

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#Modbus multiple signal is registered as “multi”(cnt=2).
get_modbus_input("multi")   # return = [10, 101]
```

8.5.14 wait_modbus_input()

Definition

wait_modbus_input(iobus, val, timeout=None)

Features

This function waits until the specified signal value of the Modbus digital I/O becomes val (ON or OFF). The waiting time can be changed with a timeout setting. The waiting time ends, and the result is returned if the waiting time has passed. This function waits indefinitely if the timeout is not set.

Parameters

Parameter Name	Data Type	Default Value	Description
iobus	string	-	Modbus name (set in the TP)
val	int	-	Modbus digital I/O • ON : 1 • OFF : 0
			Value for Modbus analog I/O
timeout	float	-	Waiting time (sec) This function waits indefinitely if the timeout is not set.

Note
When registered as a multiple signal, it is available by adding address value index to signal name.

Return

Value	Description
0	Success
-1	Failed (time-out)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
wait_modbus_input("CIN0", ON) # Indefinite wait until the "CIN0" signal becomes ON
wait_modbus_input("CIN0", OFF) # Indefinite wait until the "CIN0" signal becomes OFF

res = wait_modbus_input("CIN0", ON, 3) # Wait for up to 3 seconds until the "CIN0"
signal becomes ON
   # res = 0 if the "CIN0" signal becomes ON within 3 seconds.
   # res = -1 if the "CIN0" signal does not become ON within 3 seconds.

#Modbus multiple signal is registered as “multi”(start address=10, cnt=2).
#“multi_10” & “multi_11” abailable
wait_modbus_input("multi_10")
wait_modbus_input("multi_11")
```

8.5.15 set_modbus_slave()

Definition

set_modbus_slave(address, val)

Features

It is used to export values to the General Purpose Register area of the Modbus TCP Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	int	-	Address value of GPR area (128~255)
val	int	-	2byte value (0~65535)

Return

Value	Description
0	Success
Negative value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_modbus_slave(128, 0)
set_modbus_slave(255, 65535)
```

8.5.16 get_modbus_slave()

Definition

get_modbus_slave(address)

Features

It is used to import values by approaching the General Purpose Register area of the Modbus TCP Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	int	-	Address value of the GPR area to read (128~255)

Return

Value	Description
value	Corresponding register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
value1 = get_modbus_slave(128)
value2 = get_modbus_slave(255)
```

8.5.17 modbus_crc16()

Definition

modbus_crc16(data)

Features

When using the Modbus protocol, this command reduces the load on Modbus CRC16 calculations.

Parameters

Parameter Name	Data Type	Default Value	Description
data	byte	-	Modbus data for CRC16 calculations

Return

Value	Description
crchigh	High byte of CRC16 calculation result
crclow	Low byte of CRC16 calculation result

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
data = b"\x01\x02\x03\x04\x05\x06"
crchigh, crclow = modbus_crc16(data)
#crchigh = 186(DEC), BA(HEX)
#crclow = 221(DEC), DD(HEX)
```

8.5.18 modbus_send_make()

Definition

modbus_send_make(send_data)

Features

When using the Modbus protocol, this command provides the result data including the Modbus CRC16 result for the send data.

Parameters

Parameter Name	Data Type	Default Value	Description
send_data	byte	-	Transfer data requiring CRC calcuation

Return

Value	Description
result_data	Data in which transmission data and CRC16 value are conbined

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
senddata = b"\x01\x02\x03\x04\x05\x06"
resultdata = modbus_send_make(senddata)
#resultdata = b'\x01\x02\x03\x04\x05\x06\xba\xdd’
```

8.5.19 modbus_recv_check()

Definition

modbus_recv_check(recv_data)

Features

When using Modbus protocol, this command to check data integrity using CRC16 value for receive data.

Parameters

Parameter Name	Data Type	Default Value	Description
recv_data	byte	-	raw modbus data

Return

Value	Description
res	True/False

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
#recvdata = b"\x01\x02\x03\x04\x05\x06\xba\xdd"
res = modbus_recv_check(recvdata)
```

```
#recv = True
```

8.5.20 modbus_unsigned_to_signed()

Definition

modbus_unsigned_to_signed(unsigned_data)

Features

When using Modbus protocol, this is a command to convert 2 bytes unsigned data into signed data.

Parameters

Parameter Name	Data Type	Default Value	Description
unsigned_data	int	-	2byte unsigned data(0~65535)

Return

Value	Description
signed_data	2byte signed data(-32769 ~ 32767)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
unsigned_data = 40000
```

```
signed_data = modbus_unsigned_to_signed(unsigned_data)
```

8.6 Industrial Ethernet (EtherNet/IP,PROFINET)

8.6.1 set_output_register_bit()

Definition

set_output_register_bit(address, val)

Features

It is used to export values to the Output Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Output Bit GPR area in Industrial Ethernet Slave(0-63)
val	int	-	ON : 1 OFF : 0

Return

Value	Description
0	Success
Negative value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_output_ register_bit (0, ON)
set_output_ register_bit (63, OFF)
```

8.6.2 set_output_register_int()

Definition

set_output_register_int(address, val)

Features

It is used to export values to the Output Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Output Int GPR area in Industrial Ethernet Slave(0-23)
val	int	-	int value (4byte)

Return

Value	Description
0	Success
Negative value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_output _register_int (0, 0x00FF00FF)
set_output _register_int (23, 65535)
```

8.6.3 set_output_register_float()

Definition

set_output_register_float(address, val)

Features

It is used to export values to the Output Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Output Float GPR area in Industrial Ethernet Slave(0-23)
val	float	-	float value (4byte)

Return

Value	Description
0	Success
Negative value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
set_output _register_float (0, 4.5)
set_output _register_float (23, 2.3)
```

8.6.4 get_output_register_bit()

Definition

get_output_register_bit(address)

Features

It is used to import values to the Output Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Output Bit GPR area in Industrial Ethernet Slave(0-63)

Return

Value	Description
value	Corresponding register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
a = get_output _register_bit (0)
b = get_output _register_bit (63)
```

8.6.5 get_output_register_int()

Definition

get_output_register_int(address)

Features

It is used to import values to the Output Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Output Int GPR area in Industrial Ethernet Slave(0-23)

Return

Value	Description
value	Corresponding register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
a = get_output _register_int (0)
b = get_output_register_int(23)
```

8.6.6 get_output_register_float()

Definition

get_output_register_float(address)

Features

It is used to import values to the Output Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Output Float GPR area in Industrial Ethernet Slave(0-23)

Return

Value	Description
value	Corresponding register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
a = get_output _register_float (0)
```

```
b = get_output _register_float (63)
```

8.6.7 get_input_register_bit()

Definition

get_input_register_bit(address)

Features

It is used to import values to the Input Bit General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Input Bit GPR area in Industrial Ethernet Slave(0-63)

Return

Value	Description
value	Corresponding register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
a = get_input _register_bit (0)
b = get_input_register_bit (63)
```

8.6.8 get_input_register_int()

Definition

get_input_register_int(address)

Features

It is used to import values to the Input Int General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Input Int GPR area in Industrial Ethernet Slave(0-23)

Return

Value	Description
value	Corresponding register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred

Exception	Description
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
a = get_input _register_int (0)
b = get_input_register_int(23)
```

8.6.9 get_input_register_float()

Definition

get_input_register_float(address)

Features

It is used to import values to the Input Float General Purpose Register area of the Industrial Ethernet(EtherNet/IP, PROFINET) Slave.

Parameters

Parameter Name	Data Type	Default Value	Description
address	unsigned short	-	Address value of Input Float GPR area in Industrial Ethernet Slave(0-23)

Return

Value	Description
value	Corresponding register value

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
a = get_input _register_float (0)
b = get_input _register_float (63)
```

8.7 FOCAS

8.7.1 focas_connect()

Definition

focas_connect(ip, port, timeout)

Features

This command is used to connect to the Machine Center Controller. When connected normally, a handle value greater than 0 is returned.
The connectable controllers are as follows.

- FANUC Series 30i/31i/32i/35i-MODEL B
- FANUC Series 31i-MODEL B5
- FANUC Series Power Motion i-MODEL A
- FANUC Series 0i-MODEL D/F

Parameters

Parameter Name	Data Type	Default Value	Description
ip	str	-	Server IP address: (e.g.) “192.168.137.200”
port	int	-	Server Port number (e.g.) 8193
timeout	int	-	timeout setting (infinite for 0:, unit: s)

Return

Value	Description
errorCode	Error code (refer to focas_get_error_str)
Handle	Success (return handle value)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
```

8.7.2 focas_disconnect()

Definition

focas_disconnect(handle)

Features

This command is used to disconnect from the Machine Center Controller.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS

Return

Value	Description
errorCode	0 : Success Value other than 0: error (see focas_get_error_str)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
ErrCode = focas_disconnect(hMachineCenter)
```

8.7.3 focas_pmc_read_bit()

Definition

focas_pmc_read_bit(handle, addr_type, start_num, bit_offset)

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is bit.

> **Caution**
> Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS
addr_type	str		G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive
start_num	int		Start Address Number(0~9999)
bit_offset	int		Bit Offset(0~7)

Return

Value	Description
errorCode	0: Success (communication is canceled normally) Value other than 0: error (see focas_get_error_str)
pmc_data	PMC data (bit type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect(“10.10.0.95”, 8193, 10)
ErrCode, PMC_Data = focas_pmc_read_bit(hMachineCenter,”R”, 3500, 0)
ErrCode = focas_disconnect(hMachineCenter)
```

8.7.4 focas_pmc_read_char()

Definition

focas_pmc_read_char(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is char (1Byte).

Caution
Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS
addr_type	str		G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive
start_num	int		Start Address Number(0~9999)
read_count	int		Read from Start Address Number Number of chars (max. 5)

Return

Value	Description
errorCode	0: Success (communication is canceled normally) Value other than 0: error (see focas_get_error_str)
pmc_data	PMC data (char type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
ErrCode, PMC_Data = focas_pmc_read_char(hMachineCenter,"R",100,3)
ErrCode = focas_disconnect(hMachineCenter)
```

8.7.5 focas_pmc_read_word()

Definition

focas_pmc_read_word(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is word (2Byte).

> **Caution**
> Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS

Parameter Name	Data Type	Default Value	Description
addr_type	str		G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive
start_num	int		Start Address Number(0~9999)
read_count	int		Read from Start Address Number Number of words (max. 5)

Return

Value	Description
errorCode	0: Success (communication is canceled normally) Value other than 0: error (see focas_get_error_str)
pmc_data	PMC data (word type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
ErrCode, PMC_Data = focas_pmc_read_word(hMachineCenter,"R",3500,3)
ErrCode = focas_disconnect(hMachineCenter)
```

8.7.6 focas_pmc_read_long()

Definition

focas_pmc_read_long(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller. It is used when the data return type is long (4Byte).

> **Caution**
> Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS

Parameter Name	Data Type	Default Value	Description
addr_type	str		G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive
start_num	int		Start Address Number(0~9999)
read_count	int		Read from Start Address Number Number of longs (max. 5)

Return

Value	Description
errorCode	0: Success (communication is canceled normally) Value other than 0: error (see focas_get_error_str)
pmc_data	PMC data (long type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
ErrCode, PMC_Data = focas_pmc_read_long(hMachineCenter,"R",3500,3)
ErrCode = focas_disconnect(hMachineCenter)
```

8.7.7 focas_pmc_read_float()

Definition

focas_pmc_read_float(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller.

It is used when the data return type is float (4Byte, 32-bit-floatring-point-type).

> **Caution**
> Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS

Parameter Name	Data Type	Default Value	Description
addr_type	str		G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive
start_num	int		Start Address Number(0~9999)
read_count	int		Read from Start Address Number Number of floats (max. 5)

Return

Value	Description
errorCode	0: Success (communication is canceled normally) Value other than 0: error (see focas_get_error_str)
pmc_data	PMC data (float type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
ErrCode, PMC_Data = focas_pmc_read_float(hMachineCenter,"D",10,3)
ErrCode = focas_disconnect(hMachineCenter)
```

8.7.8 focas_pmc_read_double()

Definition

focas_pmc_read_double(handle, addr_type, start_num, read_count)

Features

This command is used to read PMC Data of Machine Center Controller.

It is used when the data return type is double (8Byte, 64-bit-floatring-point-type).

> **Caution**
> Before driving the DRL, be sure to check the PMC Signal Map of the controller before driving.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS

Parameter Name	Data Type	Default Value	Description
addr_type	str		G (Output signal from PMC to CNC) F (Input signal to PMC from CNC) Y (Output signal to PMC from machine) X (Input signal from PMC to machine) A (Message display) R (Internal relay) T (Timer) K (Keep relay) C (Counter) D (Data table) M (Input signal from other PMC path) N (Output signal to other PMC path) E (Extra relay) Z (System relay) • Case insensitive
start_num	int		Start Address Number(0~9999)
read_count	int		Read from Start Address Number Number of doubles (max. 5)

Return

Value	Description
errorCode	0: Success (communication is canceled normally) Value other than 0: error (see focas_get_error_str)
pmc_data	PMC data (double type)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred

Exception	Description
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
ErrCode, PMC_Data = focas_pmc_read_double(hMachineCenter,"D",10,3)
ErrCode = focas_disconnect(hMachineCenter)
```

8.7.9 focas_get_error_str()

Definition

focas_get_error_str(handle, errorCode)

Features

It is used to analyze the errorCode returned when using the Focas Library related function. When entering an error code, the cause of the related error is returned as a string.

Parameters

Parameter Name	Data Type	Default Value	Description
handle	int	-	Communication specific control constant value required for use of FOCAS
errorCode	int	-	Error code returned after FOCAS related DRL execution

Return

Value	Description
errorCodeString	Details of the error code (string)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

ErrorCode Details

Exception	Description
.-17	The following message is displayed depending on the detailed cause. "The send data is larger than the maximum transfer unit" "The sending data size is illegal" "The number of the recieved packet is 0" "The mark of the protocol is incorrect in the recieved packet header" "The packet type flag is incorrect in the recieved packet header" "The flag of the direction is incorrect in the recieved packet header" "Illegal received data size" "Communication error in the Ethernet Board " "protocol error"

-16	The following message is displayed depending on the detailed cause. "Error of socket API function ") "Error of connect API function") "Error of send API function" "Error of recv API function" "Error of select API function" "Error of setsockopt API function" "Error of gethostbyname API function" "Timeout error of send API function" "Timeout error of recv API function" "Error of Winsock API is occurred in other process" "EOF (end of file) detected " "socket error"
-15	"DLL not exist error"
-14	"error in APi library inital valiefile"
-13	"low temperature alarm of intellig0ent terminal"
-12	"hight temperature alarm of intelligent terminal"
-11	"bus error"
-10	"system error"
-9	"hssb communication error"
-8	"library handle error"
-7	"CNC/PMC version missmatch"
-6	"abnormal error"
-5	"system error"
-4	"shared RAM parity error"
-3	"emm386 or mmcsys install error"

-2	"reset or stop occured error"
-1	"busy error"
0	"no problem"
1	"command prepare error or pmc not exist"
2	"data block length error"
3	"data number error or address range error"
4	"data attribute error or data type error"
5	"data error"
6	"no option error"
7	"write protect error"
8	"memory overflow error"
9	"cnc parameter not correct error"
10	"buffer error"
11	"path error"
12	"cnc mode error"
13	"execution rejected error"
14	"data server error"
15	"alarm has been occurred"
16	"CNC is not running"
17	"protection data error"
18	"error generated by PMC"

19	"PMC handle error"
20	"overwrite stop in program read"
21	"reset interrupt in program read"
-100	"Library opening failed."
-101	"The maximum number of machine tools that can be connected has been exceeded."
-102	"The handle is not connected"

Example

```
ErrCode, hMachineCenter = focas_connect("10.10.0.95", 8193, 10)
if ErrCode != 0 :
   ErrorString = focas_get_error_str(hMachineCenter, ErrCode)
MC_Command = [-178.12, 11.478]
focas_pmc_write_double(hMachineCenter, "G", 3100, MC_Command, 2)
focas_disconnect(hMachineCenter)
```

9 Application Commands

9.1 External Encoder Setting Commands

9.1.1 set_extenc_polarity()

Definition

set_extenc_polarity(channel, polarity_A, polarity_B, polarity_Z, polarity_S)

Features

It configures the polarity of phase A, B and the trigger method of phase S, Z of the corresponding encoder channel.

Parameters

Parameter Name	Data Type	Default Value	Description
channel	int	1	Encoder channel (1, 2) 1: Channel 1 2: Channel 2
polarity_A	int	0	Polarity of phase A (0: Phase A, 1: /Phase A)
polarity_B	int	0	Polarity of phase B (0: Phase B, 1: /Phase B)
polarity_Z	int	0	Trigger method of Phase Z (0: Falling edge, 1: Rising edge)
polarity_S	int	0	Trigger method of Phase S (0: Falling edge, 1: Rising edge)

Return

Value	Description
N/A	Not Used

Exception

Exception	Description
N/A	Not Used

Example

```
set_extenc_polarity(1, 0, 1, 0, 1)
# External Encoder channel 1 is set to phase A, /phase B, phase Z (falling edge),
phase S (rising edge)
```

Related commands

- set_extenc_mode()(p. 500)

9.1.2 set_extenc_mode()

Definition

set_extenc_mode(channel, mode_AB, pulse_AZ, mode_Z, mode_S, inverse_cnt)

Features

It configures the operation mode of phase A, B, Z and S of the corresponding encoder channel.

[1)] Compared to versions prior to V2.7.0, Integrated mode_S parameter option - 1: Strobe Signal à Encoder Count Clear (conveyor tracking available with a single option of Encoder Count Clear), 2: works for Encoder Count Clear (for compatibility to prior version)

Parameters

Parameter Name	Data Type	Default Value	Description
channel	int	1	Encoder channel (1, 2) 1: Channel 1 2: Channel 2

Parameter Name	Data Type	Default Value	Description
mode_AB	int	0	Use of phase AB Mode (0 ~ 4) 0: Not Used 1: Phase A Quadrature use Phase B Quadrature use 2: Phase A Count Phase B Direction use 3: Phase A Up Count use Phase B Not Used 4: Phase A Down Count use Phase B Not Used
pulse_AZ	int	0	Pulse A Count per Pulse Z (0 ~ 100000)
mode_Z	int	0	Phase Z Use Mode (0 ~ 1) 0: Not Used 1: A/B Count Error Compensation 2: Encoder Count Clear
mode_S	int	0	Phase S Use Mode (0 ~ 1) 0: Not Used 1: Encoder Count Clear
inverse_cnt	int	0	Encoder Count Direction Reverse Status 0: Forward 1: Reverse

Return

Value	Description
N/A	Not Used

Exception

Exception	Description
N/A	Not Used

Example

```
set_extenc_mode(1, 2, 20000, 1, 1, 0)
# External Encoder channel 1 operation mode is set as follows
# Phase A Count, Phase B Direction Use
# Pulse A Count per Z Pulse is 20000
# Phase Z error count accumulate compensation mode use, phase S use
# Encoder Count direction is set to forward
```

Related commands

- set_extenc_polarity()(p. 499)

9.1.3 get_extenc_count()

Definition

get_extenc_count(channel)

Features

Get the count value of the corresponding encoder channel.

Parameters

Parameter Name	Data Type	Default Value	Description
channel	int	1	Encoder channel (1, 2) 1: Channel 1 2: Channel 2

Return

Value	Description
count	Current encoder count value of corresponding channel

Exception

Exception	Description
N/A	Not Used

Example

```
enc_cnt = get_extenc_count(1)
# External Encoder channel 1 current count value calculation
```

Related commands

- set_extenc_polarity()(p. 499)
- set_extenc_mode()(p. 500)
- clear_extenc_count()(p. 503)

9.1.4 clear_extenc_count()

Definition

clear_extenc_count(channel)

Features

Reset counter value of the corresponding encoder channel to 0.

Parameters

Parameter Name	Data Type	Default Value	Description
channel	int	1	Encoder channel (1, 2) 1: Channel 1 2: Channel 2

Return

Value	Description
N/A	Not Used

Exception

Exception	Description
N/A	Not Used

Example

```
clear_extenc_count(1)
# External Encoder channel 1 count value reset to 0
```

Related commands

- get_extenc_count()(p. 502)

9.2 Conveyor Tracking

9.2.1 set_conveyor_ex()

Definition

set_conveyor_ex(name="", conv_type=0, encoder_channel=1, triggering_mute_time=0.0, count_per_dist=5000, conv_coord=posx(0,0,0,0,0,0), ref=DR_BASE, conv_speed=100.0, speed_filter_size=500, min_dist=0.0, max_dist=1000.0 ...)

Features

Configures the conveyor and obtains Conveyor ID to allow the Conveyor Tracking Application to start. After the command is executed, it monitors workpieces triggered in the configured conveyor until the program ends. It can be used when you need to set parameters manually if is unavailable to configure conveyor information through UI.

[1] Added default value for all arguments compared to versions prior to M2.4.0

[2] Added 'ref' argument compared to versions prior to M2.4.0 (world coordinates available)

[3] Removed 'obj_offset_coord' argument compared to versions prior to M2.4.0, The 'obj_offset_coord' argument is changed to input only in get_conveyor_obj() function.

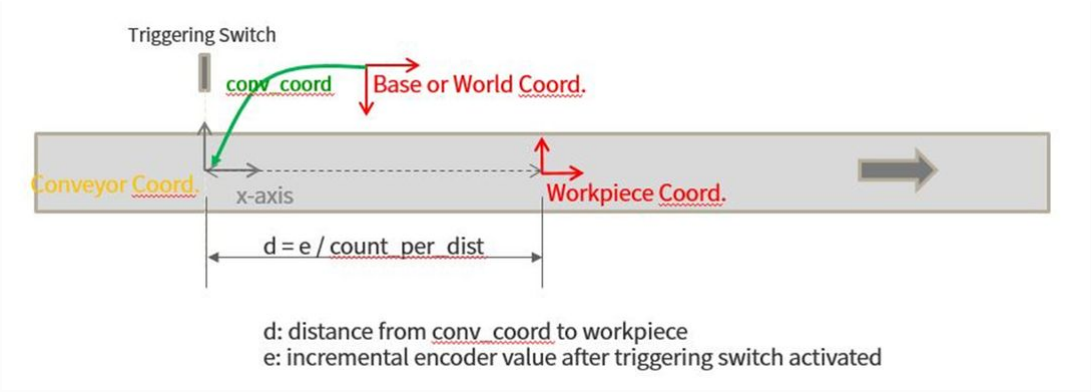
Parameters

Parameter Name	Data Type	Default Value	Description
name	string	""	Conveyor name
conv_type	int	0	Conveyor type(0: Linear, 1: Circular)
encoder_channel	int	1	External encoder channel (1, 2)
triggering_mute_time	float	0.0	It is the time (s) triggering (encoder reset, start workpiece tracking) is not performed when a triggering signal is received immediately after triggering.
count_per_dist	int	5000	Encoder count converted value per length (Linear: count/m, Circular: count/rad)
conv_coord	posx list (float[6])	posx(0,0,0,0,0,0)	Fixed conveyor coordinates (based on Base/World coordinates, mm, ˚)
ref	int	DR_BASE	Reference coordinates of conveyor coordinates (DR_BASE: Base, DR_WORLD: World)
conv_speed	float	100.0	Conveyor nominal velocity (Linear: mm/s, Circular: ˚/s)

Parameter Name	Data Type	Default Value	Description
speed_filter_size	int	500	Moving Average Filter Size during conveyor velocity filtering
min_dist	float	0.0	Minimum conveyor work length (based on Triggering Switch, Linear: mm, Circular: ˚)
max_dist	float	1000.0	Maximum conveyor work length (based on Triggering Switch, Linear: mm, Circular: ˚)
watch_window	float	100.0	Conveyor work standby monitoring length (based on minimum work length, Linear: mm, Circular: ˚)
out_tracking_dist	float	10.0	Conveyor tracking release buffer section length (based on maximum work length, Linear: mm, Circular: ˚)

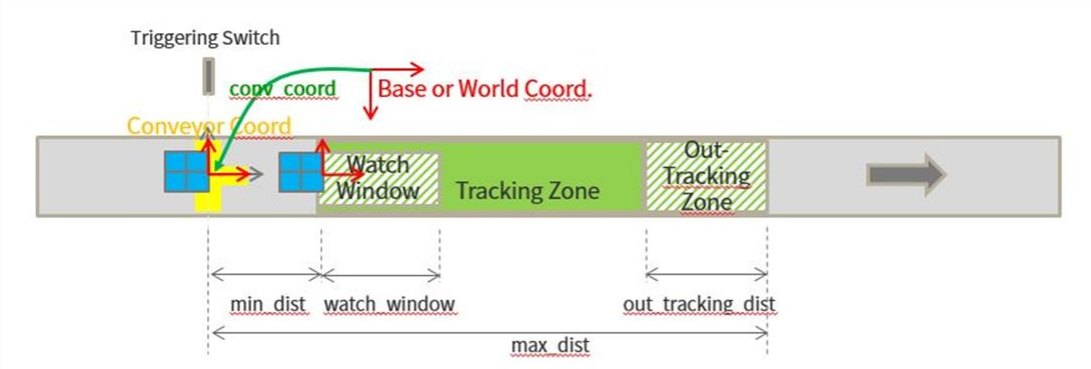
Note

- Currently conv_type argument does not support Circular Conveyors!
- All workpieces that pass the Triggering Switch are monitored until they reach max_dist after set_conveyor() or set_conveyor_ex() function execution and before the program ends.
- However, if triggering_mute_time is configured, and if the Triggering Switch activates during the corresponding time after the previous workpiece is detected, it is not included on the monitoring list. It is used when noise is present in the Triggering Switch or when the workpiece needs to be removed for a certain amount of time.
- conv_coord is a coordinate system fixed to the conveyor relative to the base or world coordinate system. **Here, the x-axis of conv_coord represents the direction the conveyor flows.** From the moment the conveyor workpiece activates the triggering switch, the increased encoder value can be converted to the length of the workpiece travel by using the count_per_dist argument, and extending this length in the x-axis direction of the conv_coord will position the workpiece relative to the reference coordinate system.



[Conveyor/Item Coordinate]

- conv_speed is the moving speed of the conveyor. It is used to give Info only if the conveyor speed sensed by the encoder exceeds 200% of this speed. Therefore, if the measurement is not possible through the TP UI, enter the approximate value.
- Speed_filter_size is the size of the moving-average filter used to estimate the conveyor speed from the encoder. The larger the size, the more the noise can be canceled, but the tracking accuracy may deteriorate during acceleration and deceleration.
- The area on top of the conveyor is categorized into Watch Window, Tracking Zone and Out-Tracking Zone.
- Watch Window is the area that determines whether workpieces within the area are available for the job when obtaining workpiece coordinates for tracking. When the get_conveyor_obj() function is loaded, if a workpiece is not present within this area, the function is not returned, and if a workpiece is present within this area, it returns workpiece coordinates according to get_conveyor_obj() function options (FIFO, LIFO).
- The Tracking Zone is the area that performs Conveyor Tracking.
- The Out-Tracking Zone is the area where the robot automatically ends tracking after it determines that the robot has exited the work space of the robot or the work space specified by the user during continuous tracking.
- These three areas are defined with the four lengths (min_dist, max_dist, watch_window, out_tracking_dist) as shown below.



[Conveyor Area and Length]

Return

Value	Description
Conveyor ID	Returns Conveyor ID if conveyor setting is successful
None	Conveyor setting failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
CONV1 = set_conveyor_ex(name='conveyor_1',
conv_type=0, # linear
encoder_channel=1, triggering_mute_time=0.0,
count_per_dist=5000, # 5000 count/mm)
conv_coord=posx(500, 100, 500, 0, -90, 0), ref=DR_BASE,
conv_speed=100.0, # conveyor speed: 100 mm/s,
speed_filter_size=500, # moving avg. filter size: 500 ms
min_dist=100, max_dist=1000, watch_window=200, out_tracking_dist=10)
```

Related commands

- get_conveyor_obj()(p. 509)
- tracking_conveyor()(p. 513)
- untracking_conveyor()(p. 515)

9.2.2 get_conveyor_obj()

Definition

get_conveyor_obj(conv_id, timeout=None, container_type=DR_FIFO, obj_offset_coord=None)

Features

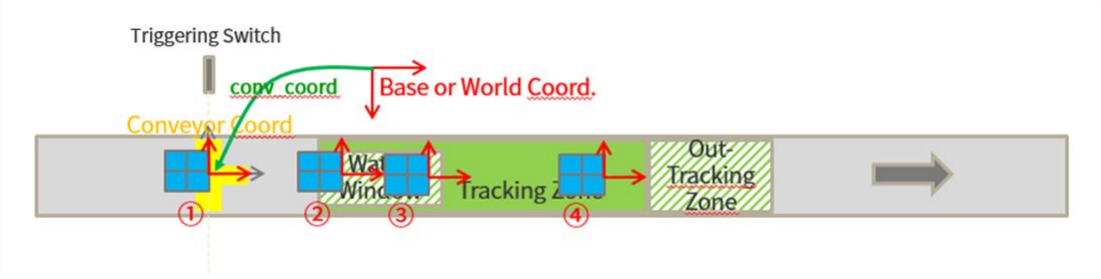
It returns the workpiece coordinate ID available for the job from the corresponding conveyor. When a function is called, it returns the workpiece present in the Watch Zone one by one according to the container rule.

Parameters

Parameter Name	Data Type	Default Value	Description
conv_id	int	-	Conveyor ID
timeout	float	None	If there is no workpiece to return during this timeout, it ends standby and return the function
container_type	int	DR_FIFO	Workpiece container type (DR_FIFO: first-in/first-out, DR_LIFO: last-in/last-out
obj_offset_coord	posx	None	Workpiece coordinates (mm, °) based on conveyor lock coordinates
	list(float[6])		

Note

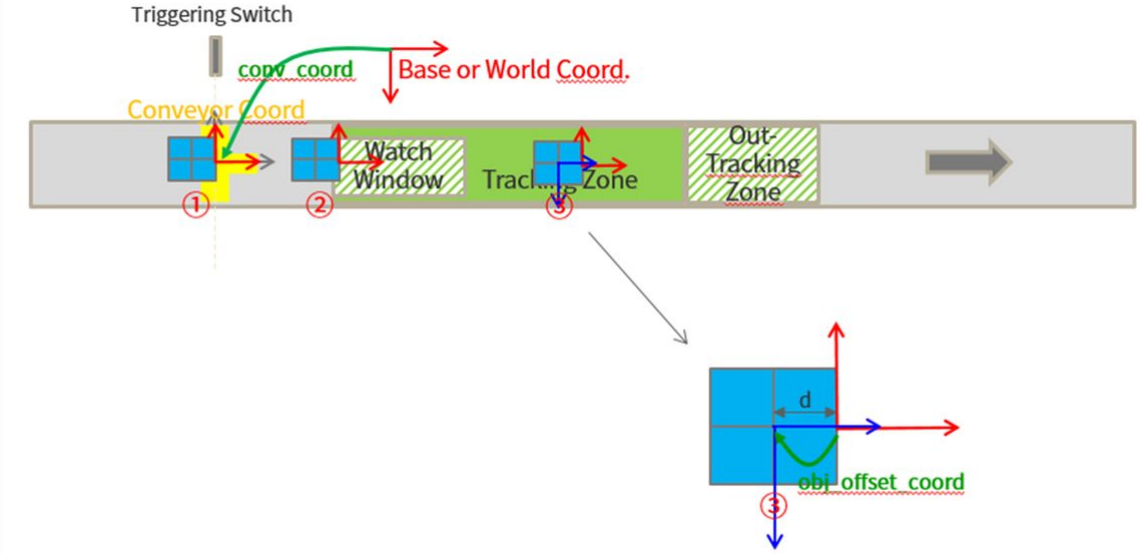
1. When calling this function, it returns the coordinates ID of each workpiece in the Watch Window according to the container rule. For example, if you call get_conveyor_obj() function when the workpieces are places as shown below, the workpiece ② and ③ in the Watch Window will be candidates. At this time, if the container_type is set to DR_FIFO the corrdinates ID of ③ that entered the Watch Window first. If it is set to DR_LIFO, it returns the coordinates ID of ② that entered the Watch Window later. If there is no workpieces in the Watch Window at the time of the function call, it waits until the time set in the timeout parameter and return the id if the workpiece comes in.



[Workpiece Coordinate ID Return Rule Description]

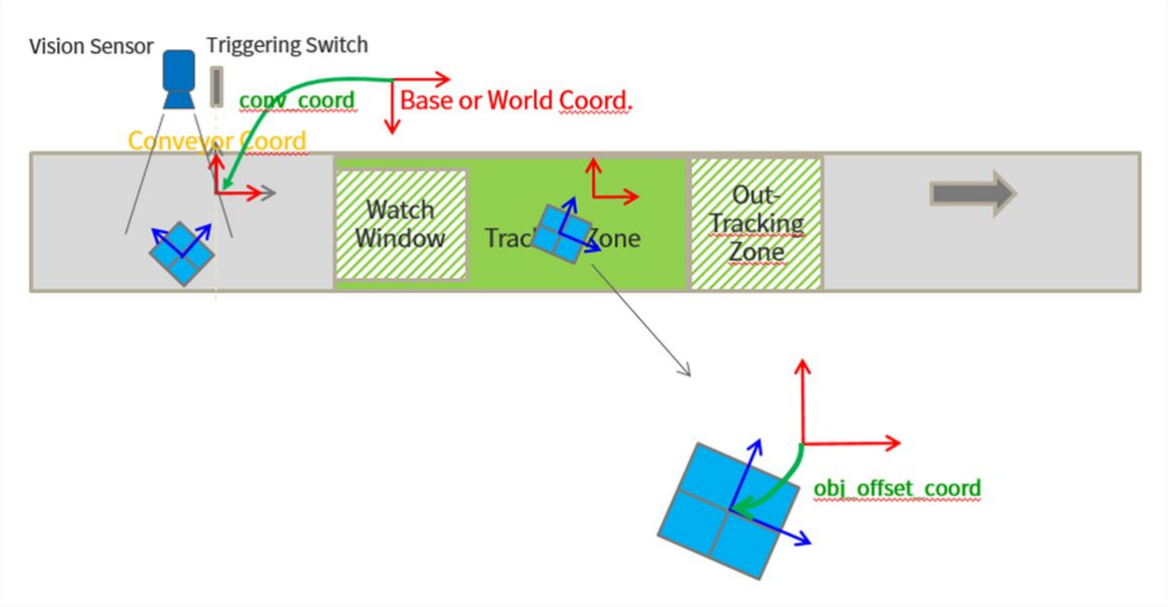
2. obj_offset_coord is used when you want to apply offset to the workpiece coordinates. It is usually used for easy input of a teaching point or when you want to dynamically change the position and orientation of the workpiece coordinates in conjunction with an external sensor (ex. Vision sensor).

In the case shown below, the workpiece coordinates are created on the right side of the workpiece and the orientation is different from the base or world coord. At this time, if you want to position the workpiece coordinates at the center of workpiece and make the orientation to be same with the ones of base or world coordinates, you can apply it as obj_offset_coord = posx (-d, 0, 0, -90, 0, 0). It is not necessary to acquire a teaching point through this TP UI, but it could utilize this method if you need to use drl only or enter the teaching point directly.



[obj_offset_coord use case 1]

Next, if the workpiece changes its position in a direction that is independent of the conveying direction, or the orientation of the workpiece changes as shown below, the encoder signal alone cannot determine the position / orientation of the workpiece. In this case, you need to detect them using external vision sensor. After detecting this value, you can input the position/orientation change detected as obj_offset_coord dynamically and the workpiece coordinates are created accordingly.



[obj_offset_coord use case 2: using vision sensor]

Return

Value	Description
int	CONV_COORD. Conveyor user coordinate ID (201~230)
Negative integer	If no workpiece is present even after the timeout expires

Note
If no workpiece to return is present, no function is returned until the timeout time expires. If the timeout time expires but no workpiece is present, it returns -1. However, if a timeout time is not entered, it doesn't return continuously.

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
## One object in a cycle
CONV1 = set_conveyor('conveyor1')

movel(posx(100, 100, 50, 0, 0, 0), ref=DR_BASE) # waiting position
while True:
CONV_COORD_1 = get_conveyor_obj(CONV1)
tracking_conveyor(CONV1)

# synched motion
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)
movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
set_digital_output(DO_GRIPPER, 1)
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

untracking_conveyor(CONV1)

movel(posx(100, 100, 50, 0, 0, 0), ref=DR_BASE) # waiting position

## Multi objects in a cycle
CONV1 = set_conveyor('conveyor1')

while True:
CONV_COORD_1 = get_conveyor_obj(CONV1)
tracking_conveyor(CONV1)

# fist object
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)
movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
set_digital_output(DO_GRIPPER, 1)
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

# second object
CONV_COORD_2 = get_conveyor_obj(CONV1, time_out=10)
if CONV_COORD_2 > 0: # -1 if no objects available during time_out
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_2)
movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_2)
set_digital_output(DO_GRIPPER, 1)
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_2)

# first object if you need
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

untracking_conveyor(CONV1)

movel(posx(100, 100, 50, 0, 0, 0), ref=DR_BASE)
```

Related commands

- tracking_conveyor()(p. 513)
- untracking_conveyor()(p. 515)

9.2.3 tracking_conveyor()

Definition

tracking_conveyor(conv_id, time=0.3)

Features

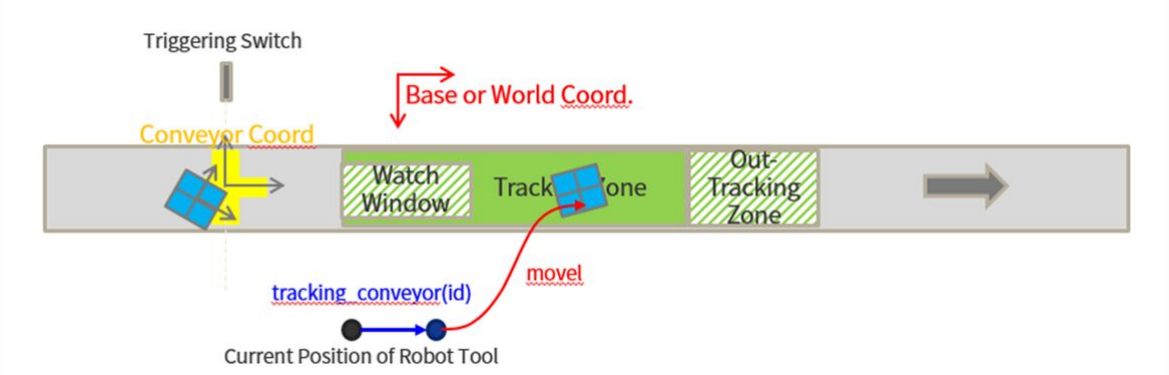
The robot starts Conveyor Tracking.

Parameters

Parameter Name	Data Type	Default Value	Description
conv_id	int	-	Conveyor ID
time	float	0.3	Acceleration time(sec) to start Tracking

Note

If the tracking_conveyor command is given, the conveyor starts tracking from the current position of robot. You can call task motion right after tracking_conveyor function for reducing tack time, although transient error can occur during acceleration time.



[tracking sequence]

Return

Value	Description
0	Conveyor Tracking success
Negative integer	If the robot is expected to exit the robot work space during acceleration

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
CONV1 = set_conveyor('conveyor1')

while True:
CONV_COORD_1 = get_conveyor_obj(CONV1)

tracking_conveyor(CONV1) # start moving to track conveyor

# task on conveyor
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)
movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
set_digital_output(DO_GRIPPER, 1)
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

untracking_conveyor(CONV1)
obj_count = obj_count + 1
```

Related commands

- untracking_conveyor()(p. 515)

9.2.4 untracking_conveyor()

Definition

untracking_conveyor(conv_id, time=0.3)

Features

The robot moves as its velocity goes to 0 and finish Conveyor Tracking.

Parameters

Parameter Name	Data Type	Default Value	Description
conv_id	int	-	Conveyor ID
time	float	0.3	Deceleration time (sec) to end Tracking

Note

- If a time value is shorter than the robot's maximum deceleration speed, the robot ignores the entered value and decelerates using the maximum deceleration speed.
- To reduce tack-time, deceleration motion is blended with task motion after untracking_conveyor if it is called. (However, Joint motion cannot be called during deceleration time.)

Return

Value	Description
0	Finishing Conveyor Tracking success
Negative integer	If the robot is expected to exit the robot work space during deceleration

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C Extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
CONV1 = set_conveyor('conveyor1')

while True:
CONV_COORD_1 = get_conveyor_obj(CONV1)
tracking_conveyor(CONV1)

# task on conveyor
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)
movel(posx(0,0, 0, 0, 0, 0), ref=CONV_COORD_1)
set_digital_output(DO_GRIPPER, 1)
movel(posx(0,0, 50, 0, 0, 0), ref=CONV_COORD_1)

untracking_conveyor(CONV1, 0.1)
```

Related commands

- tracking_conveyor()(p. 513)

9.3 Welding

It is not supported on the P-series.

9.3.1 app_weld_enable_digital()

Features

This enables the communication interface welding function. Only supports EtherNet/IP interface.

Return

Value	Description
0	Enable Welding Success

Value	Description
Negative Value	Enable Welding Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_digital()
app_weld_disable_digital()
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)...
- app_weld_set_interface_eip_m2r_other()(p. 543)
- app_weld_set_weld_cond_digital()(p. 547)
- app_weld_adj_welding_cond_digital()(p. 551)
- app_weld_disable_digital()(p. 518)
- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)

9.3.2 app_weld_disable_digital()

Features

This disables the communication interface welding function.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_digital()
app_weld_disable_digital()
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)

- app_weld_set_interface_eip_m2r_other()(p. 543)...
- app_weld_enable_digital()(p. 516)
- app_weld_set_weld_cond_digital()(p. 547)
- app_weld_adj_welding_cond_digital()(p. 551)
- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)

9.3.3 app_weld_set_interface_eip_r2m_process()

Definition

app_weld_set_interface_eip_r2m_process(welding_start=[0,0,0,0,0,0,0,0], robot_ready=[0,0,0,0,0,0,0,0], error_reset=[0,0,0,0,0,0,0,0])

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the link signal interface between the robot controller and welder used for welding in the communication data sent to the welder from the robot controller. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.
app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

Parameters

Parameter Name	Data Type	Default Value	Description
welding_start	Refer to the table below	Refer to the table below	Start Weld Command (specification for each welder)
robot_ready			Robot Status (specification for each welder)
error_reset			Reset Welder Error (specification for each welder)

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7
		0	Valid Data Size Value (bit)

Parameter Name	Data Type	Default Value	Description
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Communication Interface Setting Example (EWM Welder)

Data Type (on/off: 0): Item selected On/Off

Ewm Welder Data Sheet

Byte no.	Bit no.	Function/description	Bit assignment
0	4	Start signal welding process	0 switched off 1 switched on

Specification Entry Method

Item	Setting Value
Usage Status	1 (Used)
Data Type	0 (on/off)
Data Digits	0 (1)
Communication Data Point (byte)	0
Communication Data Point (bit)	4
Data Size	0 (1 bit, disable Low)
Valid Data Size	1 (1 bit)
Minimum Data Value	0
Maximum Data Value	1

Data Type (Select: 1): If data with an integer of 1 is selected

Ewm Welder Data Sheet

Byte no.	Bit no.	Function/description	Bit assignment
3	0-7	Selection JOB	Range 1-255

Specification Entry Method

Item	Setting Value
Usage Status	1 (Used)
Data Type	1 (Select)
Data Digits	0 (1)
Communication Data Point (byte)	3
Communication Data Point (bit)	0
Data Size	4 (8-bit)
Valid Data Size	8 (8 bit)
Minimum Data Value	0
Maximum Data Value	255

Data Type (Value: 2): If a real number value is entered

Ewm Welder Data Sheet

Byte no.	Bit no.	Function/description	Bit assignment
6	0-15	Welding voltage(current actual value)	0 to 0x7FFF (High-Byte first) equivalent to 0.0V to 100.0V

Specification Entry Method

Item	Setting Value
Usage Status	1 (Used)
Data Type	2 (Value)
Data Digits	1 (0.1)
Communication Data Point (byte)	6
Communication Data Point (bit)	0
Data Size	6 (16-bit)
Valid Data Size	15 (15 bit)
Minimum Data Value	0.0 (V)
Maximum Data Value	100.0 (V)

Note
If the data type is 2 (Value), a valid data size (0x7FFF → 15 bit), minimum data value (0.0V) and maximum data value (100.0V) must be entered.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error

Exception	Description
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_r2m_process(welding_start=[1,0,0,0,4,0,1,0,0],
                                       robot_ready=[1,0,0,0,5,0,1,0,0],
                                       error_reset=[1,0,0,1,4,0,1,0,0])
```

Related commands

- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)

9.3.4 app_weld_set_interface_eip_r2m_mode()

Definition

app_weld_set_interface_eip_r2m_mode(welding_mode=[0,0,0,0,0,0,0,0,0], s_2t=[0,0,0,0,0,0,0,0,0], pulse_mode=[0,0,0,0,0,0,0,0,0],wm_opt1=[0,0,0,0,0,0,0,0,0])

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to the welding mode in the communication data sent to the welder from the robot controller. Required modes can be additionally added with the option item (wm_opt1). Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.
app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

Parameters

Parameter Name	Data Type	Default Value	Description
welding_mode	Refer to the table below	Refer to the table below	Welding Mode (specification for each welder)
s_2t			Latched/Non-latched Mode (specification for each welder)
pulse_mode			Pulse Mode (specification for each welder)
wm_opt1			Option Mode (specification for each welder)

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7

Parameter Name	Data Type	Default Value	Description
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7
		0	Valid Data Size Value (bit)
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Note
For examples of data (0~2) interface settings, refer to the app_weld_set_interface_eip_r2m_process() section.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error

Exception	Description
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_r2m_mode(welding_mode=[1,1,0,0,0,2,2,0,3],
                                    s_2t=[0,0,0,0,0,0,0,0,0],
                                    pulse_mode=[1,1,0,0,2,0,1,0,1],
                                    wm_opt1=[0,0,0,0,0,0,0,0,0])
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)

9.3.5 app_weld_set_interface_eip_r2m_test()

Definition

app_weld_set_interface_eip_r2m_test(gas_test=[0,0,0,0,0,0,0,0,0], inching_plus=[0,0,0,0,0,0,0,0,0], inching_minus=[0,0,0,0,0,0,0,0,0], blow_out_torch=[0,0,0,0,0,0,0,0,0], simulation=[0,0,0,0,0,0,0,0,0], ts_opt1=[0,0,0,0,0,0,0,0,0], ...)

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to the test signal setting in the communication data sent to the welder from the robot controller. Additional functions related to the test signal can be added with the option item (ts_opt1, ts_opt2). Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.
app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

Parameters

Parameter Name	Data Type	Default Value	Description
gas_test	Refer to the table below	Refer to the table below	Gas Test Signal (specification for each welder)
inching_plus			Forward Inching Signal (specification for each welder)
inching_minus			Reverse Inching Signal (specification for each welder)
blow_out_torch			Torch Cleaning Signal (specification for each welder)
simulation			Mock Welding Signal (specification for each welder)
ts_opt1			Option Signal (specification for each welder)
ts_opt2			Option Signal (specification for each welder)

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)

Parameter Name	Data Type	Default Value	Description
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7
		0	Valid Data Size Value (bit)
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Note
For examples of data (0~2) interface settings, refer to the app_weld_set_interface_eip_r2m_process() section.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_r2m_test(gas_test=[1,0,0,0,6,0,1,0,0],
                                    inching_plus=[1,0,0,1,0,0,1,0,0],
                                    inching_minus=[1,0,0,1,2,0,1,0,0],
                                    blow_out_torch=[1,0,0,0,7,0,1,0,0],
                                    simulation=[0,0,0,1,7,0,1,0,0],
                                    ts_opt1=[0,0,0,0,0,0,0,0,0],
                                    ts_opt2=[0,0,0,0,0,0,0,0,0])
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)

9.3.6 app_weld_set_interface_eip_r2m_condition()

Definition

app_weld_set_interface_eip_r2m_condition(job_num=[0,0,0,0,0,0,0,0,0], synergic_id=[0,0,0,0,0,0,0,0,0], r_wire_feed_speed=[0,0,0,0,0,0,0,0,0], voltage_correct=[0,0,0,0,0,0,0,0.0,0.0], dynamic_correct=[0,0,0,0,0,0,0,0,0])

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to the welding condition setting in the communication data sent to the welder from the robot controller. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.
app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

Parameters

Parameter Name	Data Type	Default Value	Description
job_num	Refer to the table below	Refer to the table below	JOB Number (specification for each welder)
synergic_id			SYNERGIC Number (specification for each welder)
r_wire_feed_speed			Wire Feeding Speed Correction (specification for each welder)
voltage_correct			Voltage Correction (specification for each welder)
dynamic_correct			Dynamic Correction (specification for each welder)

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1

Parameter Name	Data Type	Default Value	Description
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7
		0	Valid Data Size Value (bit)
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Note
For examples of data (0~2) interface settings, refer to the app_weld_set_interface_eip_r2m_process() section.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_r2m_condition(job_num=[1,1,0,3,0,4,8,0,255],
                                         synergic_id=[1,1,0,2,0,3,4,0,15],
                                         r_wire_feed_speed=[1,2,1,6,0,6,15,0.0,25.0],
                                         voltage_correct=[1,2,1,8,0,6,15,-10.0,10.0],
                                         dynamic_correct=[1,2,0,10,0,6,15,-40,40])
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process(])(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)

9.3.7 app_weld_set_interface_eip_r2m_option()

Definition

app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0,0],...)

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. Required functions other than basic setting items (app_weld_set_interface_eip_r2m_process(), app_weld_set_interface_eip_r2m_mode(), app_weld_set_interface_eip_r2m_test(), app_weld_set_interface_eip_r2m_condition()) in the communication data sent to the welder from the robot controller can be set with the corresponding command. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.
app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

Parameters

Parameter Name	Data Type	Default Value	Description
opt1	Refer to the table below	Refer to the table below	Option Item (specification for each welder)
opt2			
opt3			
opt4			
opt5			
opt6			
opt7			
opt8			

opt9			
opt10			
opt11			
opt12			
opt13			
opt14			
opt15			

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7

		0	Valid Data Size Value (bit)
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Note
For examples of data (0~2) interface settings, refer to the app_weld_set_interface_eip_r2m_process() section.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_r2m_option(opt1=[0,0,0,0,0,0,0,0,0,0],
                                      opt2=[0,0,0,0,0,0,0,0,0,0],
                                      opt3=[0,0,0,0,0,0,0,0,0,0],
                                      opt4=[0,0,0,0,0,0,0,0,0,0],
                                      opt5=[0,0,0,0,0,0,0,0,0,0],
                                      opt6=[0,0,0,0,0,0,0,0,0,0],
                                      opt7=[0,0,0,0,0,0,0,0,0,0],
                                      opt8=[0,0,0,0,0,0,0,0,0,0],
                                      opt9=[0,0,0,0,0,0,0,0,0,0],
```

```
                                        opt10=[0,0,0,0,0,0,0,0,0],
                                        opt11=[0,0,0,0,0,0,0,0,0],
                                        opt12=[0,0,0,0,0,0,0,0,0],
                                        opt13=[0,0,0,0,0,0,0,0,0],
                                        opt14=[0,0,0,0,0,0,0,0,0],
                                        opt15=[0,0,0,0,0,0,0,0,0])
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)

9.3.8 app_weld_set_interface_eip_m2r_process()

Definition

app_weld_set_interface_eip_m2r_process(current_flow=[0,0,0,0,0,0,0,0,0], process_active=[0,0,0,0,0,0,0,0,0], main_current=[0,0,0,0,0,0,0,0,0], machine_ready=[0,0,0,0,0,0,0,0,0], comm_ready=[0,0,0,0,0,0,0,0,0])

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. The link signal interface between the controller and welder in the communication data sent to the welder from the robot controller can be set. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

1. To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.

app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

2. Start robot motion links with the current_flow signal from the welder but is linked with the corresponding signal when the main_current item is set.

3. End robot motion links with the current_flow signal from the welder but is linked with the corresponding signal when the process_active item is set.

Parameters

Parameter Name	Data Type	Default Value	Description
current_flow	Refer to the table below	Refer to the table below	Welding Current Generated (specification for each welder)
process_active			Welding Process Activated (specification for each welder)
main_current			Regular Welding Current Generated (specification for each welder)
machine_ready			Welding Standby (specification for each welder)
comm_ready			Communication Standby (specification for each welder)

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7

Parameter Name	Data Type	Default Value	Description
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7
		0	Valid Data Size Value (bit)
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Note
For examples of data (0~2) interface settings, refer to the app_weld_set_interface_eip_r2m_process() section.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_m2r_process(current_flow=[1,0,0,0,0,0,1,0,0],
                                       process_active=[1,0,0,0,6,0,1,0,0],
                                       main_current=[1,0,0,0,5,0,1,0,0],
                                       machine_ready=[0,0,0,0,0,0,0,0,0],
                                       comm_ready=[0,0,0,0,0,0,0,0,0])
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)

9.3.9 app_weld_set_interface_eip_m2r_monitoring()

Definition

app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0,0], welding_current=[0,0,0,0,0,0,0,0,0], wire_feed_speed=[0,0,0,0,0,0,0,0,0], wire_stick=[0,0,0,0,0,0,0,0,0], error=[0,0,0,0,0,0,0,0,0] ...)

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. This sets the interface related to monitoring of the welding machine's status setting in the communication data sent to the welder from the robot controller. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.
app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

Parameters

Parameter Name	Data Type	Default Value	Description
welding_voltage	Refer to the table below	Refer to the table below	Actual Welding Voltage (specification for each welder)
welding_current			Actual Welding Current (specification for each welder)
wire_feed_speed			Actual Wire Feeding Speed (specification for each welder)
wire_stick			Check Wire Stick Status (specification for each welder)
error			Check Error Status (specification for each welder)
error_num			Check Error Number (specification for each welder)

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)

Parameter Name	Data Type	Default Value	Description
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7
		0	Valid Data Size Value (bit)
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Note
For examples of data (0~2) interface settings, refer to the app_weld_set_interface_eip_r2m_process() section.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_m2r_monitoring(welding_voltage=[0,0,0,0,0,0,0,0,0,0],
                                          welding_current=[0,0,0,0,0,0,0,0,0,0],
                                          wire_feed_speed=[0,0,0,0,0,0,0,0,0,0],
                                          wire_stick=[0,0,0,0,0,0,0,0,0,0],
                                          error=[0,0,0,0,0,0,0,0,0,0],
                                          error_num=[0,0,0,0,0,0,0,0,0,0])
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_other()(p. 543)

9.3.10 app_weld_set_interface_eip_m2r_other()

Definition

app_weld_set_interface_eip_m2r_other(opt1=[0,0,0,0,0,0,0,0,0,0], opt2=[0,0,0,0,0,0,0,0,0,0], opt3=[0,0,0,0,0,0,0,0,0,0], opt4=[0,0,0,0,0,0,0,0,0,0], opt5=[0,0,0,0,0,0,0,0,0,0], opt6=[0,0,0,0,0,0,0,0,0,0], opt7=[0,0,0,0,0,0,0,0,0,0],...)

Features

This sets the communication interface setting to use welders that support EtherNet/IP communication. Required functions other than basic setting items (app_weld_set_interface_eip_m2r_process(), app_weld_set_interface_eip_m2r_monitoring(), app_weld_set_interface_eip_m2r_other(), app_weld_set_interface_eip_r2m_condition()) in the communication data sent to the robot controller from the welder can be set with the corresponding command. Enter the setting values below along with details based on the communication signal data sheet of the corresponding welder.

Note

To ensure proper welding using an EtherNet/IP remote control welder, all of 8 interface commands must be set up.
app_weld_set_interface_eip_r2m_process(),
app_weld_set_interface_eip_r2m_mode(),
app_weld_set_interface_eip_r2m_test(),
app_weld_set_interface_eip_r2m_condition(),
app_weld_set_interface_eip_r2m_option(),
app_weld_set_interface_eip_m2r_process(),
app_weld_set_interface_eip_m2r_monitoring(),
app_weld_set_interface_eip_m2r_other()

Parameters

Parameter Name	Data Type	Default Value	Description
opt1	Specification		
opt2	Specification		
opt3	Specification		
opt4	Specification		
opt5	Specification		
opt6	Specification		
opt7	Specification		
opt8	Specification		
opt9	Specification		

Parameter Name	Data Type	Default Value	Description
opt10	Specification		

The data type, default value and description are identical to the below

Parameter Name	Data Type	Default Value	Description
	list(int[7])	0	Not Used: 0 Used: 1
		0	Data Type (on/off: 0, Select: 1, Value: 2)
		0	Data Digits (1: 0, 0.1: 1, 0.01: 2)
		0	Communication Data Point (byte): 0~255
		0	Communication Data Point (bit): 0~7
		0	Data Size 1-bit(disable Low): 0 1-bit(disable High): 1 2-bit: 2 4-bit: 3 8-bit(byte): 4 15-bit: 5 16-bit(short): 6 32-bit(int): 7
		0	Valid Data Size Value (bit)
	list(float[2])	0	Minimum Data Value
		0	Maximum Data Value

Note

For examples of data (0~2) interface settings, refer to the app_weld_set_interface_eip_r2m_process() section.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_set_interface_eip_m2r_other(opt1=[1,2,1,12,0,6,15,0.0,25.5],
                                     opt2=[1,0,0,0,1,0,1,0,0],
                                     opt3=[0,0,0,0,0,0,0,0,0],
                                     opt4=[0,0,0,0,0,0,0,0,0],
                                     opt5=[0,0,0,0,0,0,0,0,0],
                                     opt6=[0,0,0,0,0,0,0,0,0],
                                     opt7=[0,0,0,0,0,0,0,0,0],
                                     opt8=[0,0,0,0,0,0,0,0,0],
                                     opt9=[0,0,0,0,0,0,0,0,0],
                                     opt10=[0,0,0,0,0,0,0,0,0])
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)

- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)

9.3.11 app_weld_set_weld_cond_digital()

Definition

app_weld_set_weld_cond_digital(flag_dry_run=0, vel_target=0, vel_min=0, vel_max=0, welding_mode=0, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0, ts_opt1=0, ts_opt2=0,...)

Features

This sets the welding condition of the remote control welders. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_digital()) and Disable Welding (app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. Items that can be set as welding conditions are welders corresponding to the following commands (app_weld_set_interface_eip_r2m_mode(), app_weld_set_interface_eip_r2m_condition(), app_weld_set_interface_eip_r2m_option()) and items that completed the communication interface setting.

Only one welding condition is allowed in a single welding section. This welding condition can be adjusted during welding with the app_weld_adj_welding_cond_dgital() command while the voltage correction/dynamic correction/feeding speed/speed (and weaving offset) can be also adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_digital().

Note

1. Voltage Correction: Adjusts the length of the ark.
2. Dynamic Correction: Adjusts the ark property.

Parameters

Parameter Name	Data Type	Default Value	Description
flag_dry_run	int	0	Dry-Run Mode Actual Welding (0) Dry-Run (1): Motion/Weaving/Offset only
vel_target	float	0	Target Speed (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)

Parameter Name	Data Type	Default Value	Description
vel_min	float	0	Minimum Target Speed Correction (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)
vel_max	float	0	Maximum Target Speed Correction (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)
welding_mode	Int	0	Welding Mode Setting
s_2t	Int	0	2T, 2T Special Setting
pulse_mode	Int	0	Pulse Mode Setting
wm_opt1	Int	0	Welding Mode Option 1 Setting
simulation	int	0	Simulation Mode Setting
ts_opt1	Int	0	Test Signal Option 1 Setting
ts_opt2	Int	0	Test Signal Option 2 Setting
Job_num	Int	0	Job Number Setting
synergic_id	Int	0	Synergic ID Setting
r_wire_feed_speed	float	0	Wire Feeding Speed Setting
voltage_correct	float	0	Voltage Correction Setting
dynamic_correct	float	0	Dynamic Correction Setting
r_opt1	float	0	Option 1 Setting
r_opt2	float	0	Option 2 Setting
r_opt3	float	0	Option 3 Setting

Parameter Name	Data Type	Default Value	Description
r_opt4	float	0	Option 4 Setting
r_opt5	float	0	Option 5 Setting
r_opt6	float	0	Option 6 Setting
r_opt7	float	0	Option 7 Setting
r_opt8	float	0	Option 8 Setting
r_opt9	float	0	Option 9 Setting
r_opt10	float	0	Option 10 Setting
r_opt11	float	0	Option 11 Setting
r_opt12	float	0	Option 12 Setting
r_opt13	float	0	Option 13 Setting
r_opt14	float	0	Option 14 Setting
r_opt15	float	0	Option 15 Setting

Return

Value	Description
0	Setting Success
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error

Exception	Description
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_digital()
app_weld_set_weld_cond_digital(flag_dry_run=0, vel_target=16, vel_min=0.00,
vel_max=16.67, welding_mode=3, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0,
ts_opt1=0, ts_opt2=0, job_num=4, synergic_id=0, r_wire_feed_speed=10,
voltage_correct=0, dynamic_correct=0, r_opt1=0, r_opt2=0, r_opt3=0, r_opt4=0,
r_opt5=0, r_opt6=0, r_opt7=0, r_opt8=0, r_opt9=0, r_opt10=0, r_opt11=0, r_opt12=0,
r_opt13=0, r_opt14=0, r_opt15=0)
#Welding Speed=60 mm/sec (=1 cm/min), Welding Mode=3, Job Number=4, Wire Feeding
Speed=10 m/min

app_weld_disable_digital()
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)
- app_weld_enable_digital()(p. 516)
- app_weld_adj_welding_cond_digital()(p. 551)
- app_weld_disable_digital()(p. 518)
- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)

9.3.12 app_weld_adj_welding_cond_digital()

Definition

app_weld_adj_welding_cond_digital(flag_reset=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None, dynamic_cor=None, voltage_cor=None, job_number=None, synergic_id=None)

Features

This adjusts the welding condition and weaving condition during welding with a remote control welder. It is used to change the welding condition of each section in a series of sections before calling the motion command (movel(), movec(), moveb(), movesx()). If an adjustment factor is entered using this command, the corresponding welding condition and weaving condition are adjusted, and real-time adjustment of the welding/weaving condition from the welding monitoring information screen of the TP becomes unavailable. Execute flag_reset=1 to reset the adjusted condition to the main condition set using app_weld_set_weld_cond_digital() and app_weld_weave_cond_trapezoidal(). Setting flag_reset=1 will reset to the final condition adjusted in real-time using the TP (the weaving width ratio (wv_width_ratio), which cannot be adjusted in real-time, is changed to 1), and the welding condition can be adjusted in real-time from the TP.

Parameters

Parameter Name	Data Type	Default Value	Description
flag_reset	int	0	0: Apply Adjustment 1 : Default Target (app_weld_set_weld_cond_digital()) Apply Value
f_target	float	-	Feeding Speed (m/min)
vel_target	float	-	Target Speed (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)
wv_offset	float[2]	-	Weaving Coordinate-Y Direction Offset (mm)
		-	Weaving Coordinate-Z Direction Offset (mm)
wv_width_ratio	float	-	Changed Weaving Width/Set Weaving Width Ratio (0-2)
dynamic_cor	float	-	Dynamic Correction

Parameter Name	Data Type	Default Value	Description
voltage_cor	float	-	Voltage Correction
job_number	float	-	Job Number
synergic_id	float	-	Synergic ID

Note

Conditions which do not designate a value in factors vel_target/wv_offset/wv_width_ratio/ dynamic_cor/voltage_cor/job_number/synergic_id will maintain the current condition (including conditions with real-time adjustments), so only set factors that require adjustment. However, in the case of wv_offset, even if an adjustment is made only to the Y direction or Z direction, values for both sequences must be entered.

Return

Value	Description
0	Setting Success
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
movej(posj(0,0,90,0,90,0),v=30,a=60)

pt1= posx(559, 434.5, 651.5, 45, 180, 45)
pt2= posx(559, 434.5, 151.5, 45, 180, 45)
pt3= posx(559, 0.0, 151.5, 45, 180, 45)

app_weld_enable_digital()

app_weld_set_weld_cond_digital(flag_dry_run=0, vel_target=16, vel_min=0.00,
vel_max=16.67, welding_mode=3, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0,
ts_opt1=0, ts_opt2=0, job_num=4, synergic_id=0, r_wire_feed_speed=15,
voltage_correct=0, dynamic_correct=0, r_opt1=0, r_opt2=0, r_opt3=0, r_opt4=0,
r_opt5=0, r_opt6=0, r_opt7=0, r_opt8=0, r_opt9=0, r_opt10=0, r_opt11=0, r_opt12=0,
r_opt13=0, r_opt14=0, r_opt15=0)

movel(pt1, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)
app_weld_adj_welding_cond_digital(flag_reset=0, f_target=10, vel_target=16,
wv_offset=[20,30], wv_width_ratio=0.5,
dynamic_cor=0, voltage_cor=0, job_number=5, synergic_id=4)
movel(pt2, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)
app_weld_adj_welding_cond_digital(flag_reset=1)
movel(pt3, v=5, a=5, app_type=DR_MV_APP_WELD)
# Start Point → pt1: Apply Initial Welding Condition Setting (Job Number: 4, Synergic
ID: 0, Feeding Speed: 15 m/min)
# pt1 → pt2: Apply Correction Condition (Job Number: 5, Synergic ID: 4, Feeding
Speed: 15 m/min)
# pt2 → pt3: Apply Initial Setting Apply Initial Welding Condition Setting (Job
Number: 4, Synergic ID: 0, Feeding Speed: 15 m/min)

app_weld_disable_digital()
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)
- app_weld_adj_welding_cond_digital()(p. 551)
- app_weld_set_weld_cond_digital()(p. 547)
- app_weld_adj_welding_cond_digital()(p. 551)
- app_weld_disable_digital()(p. 518)

- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)
- movel()(p. 63)
- amovel()(p. 102)
- movec()(p. 72)
- amovec()(p. 108)
- moveb()(p. 84)
- amoveb()(p. 118)
- movesx()(p. 81)
- amovesx()(p. 115)

9.3.13 app_weld_get_welding_cond_digital()

Features

This monitors the welding status during welding with a remote control welder. Items available for monitoring are voltage correction/dynamic correction/feeding speed/speed/weaving offset/error number/error status/wire tip fusion/option and measured voltage/current/feeding speed and welding status. Signals other than the basic monitoring items can be added. The corresponding signal must be preset using the interface app_weld_set_interface_eip_m2r_other(). In addition, it is possible to check the fail status using the welding status factor.

Return

Value	Description
voltage_cor	Current Target Voltage Correction (V) (target with adjustment applied)
dynamic_cor	Current Target Dynamic Correction (target with adjustment applied)
f_target	Current Target Feeding Speed (m/min) (target with adjustment applied)
vel_target	Current Target Speed (mm/sec) (target with adjustment applied) • Take note that the unit is different from that of the monitoring output unit (cm/min) of the teaching pendant
v_meas	Current Measured Voltage (V)
c_meas	Current Measured Current (A)

Value	Description
wv_offset[2]	Current Target Offset (Y and Z directions, mm) (target with adjustment applied)
status	Non-weld:0, Weld (Normal): 1, Weld (Abnormal): 9, Dry-run: 99
f_meas	Current Measured Feeding Speed (mm/sec)
error_num	Error Number
wire_stick	Wire Tip Fusion Status (0: Normal, 1: Abnormal)
error	Error Status (0: Normal, 1: Abnormal)
option1	Option 1 Information
option2	Option 2 Information
option3	Option 3 Information
option4	Option 4 Information
option5	Option 5 Information
option6	Option 6 Information
option7	Option 7 Information
option8	Option 8 Information
option9	Option 9 Information
option10	Option 10 Information
current_flow	Current Flow Status (0: Normal, 1: Abnormal)
process_active	Process Activation Status (0: Normal, 1: Abnormal)
machine_ready	Welding Preparation Status (0: Normal, 1: Abnormal)

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
movej(posj(0,0,90,0,90,0),v=30,a=60)

pt1= posx(559, 434.5, 651.5, 45, 180, 45)
pt2= posx(559, 434.5, 151.5, 45, 180, 45)
pt3= posx(559, 0.0, 151.5, 45, 180, 45)

app_weld_enable_digital()

app_weld_set_weld_cond_digital(flag_dry_run=1, vel_target=16, vel_min=0.00,
vel_max=16.67, welding_mode=3, s_2t=0, pulse_mode=0, wm_opt1=0, simulation=0,
ts_opt1=0, ts_opt2=0, job_num=4, synergic_id=0, r_wire_feed_speed=15,
voltage_correct=0, dynamic_correct=0, r_opt1=0, r_opt2=0, r_opt3=0, r_opt4=0,
r_opt5=0, r_opt6=0, r_opt7=0, r_opt8=0, r_opt9=0, r_opt10=0, r_opt11=0, r_opt12=0,
r_opt13=0, r_opt14=0, r_opt15=0)

movel(pt1, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)
app_weld_adj_welding_cond_digital(flag_reset=0, f_target=10, vel_target=16,
wv_offset=[20,30], wv_width_ratio=0.5,
dynamic_cor=0, voltage_cor=0, job_number=5, synergic_id=4)
movel(pt2, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)
app_weld_adj_welding_cond_digital(flag_reset=1)
amovel(pt3, v=5, a=5, app_type=DR_MV_APP_WELD)

[voltage_cor, dynamic_cor, f_target, vel_target, v_meas, c_meas, wv_offset, status,
f_meas, error_num, wire_stick,
error, opt1, opt2, opt3, opt4, opt5, opt6, opt7, opt8, opt9, opt10, current_flow,
process_active, machine_ready]=app_weld_get_welding_cond_digital();

while True:
   if status == 9:
```

```
        tp_popup("welding error!! ", DR_PM_ALARM, 1)
    # An alarm is generated if abnormal welding occurs (status=9)
     else :
        if check_motion()==0:
            break

app_weld_disable_digital()
```

Related commands

- app_weld_set_interface_eip_r2m_process()(p. 519)
- app_weld_set_interface_eip_r2m_mode()(p. 524)
- app_weld_set_interface_eip_r2m_test()(p. 527)
- app_weld_set_interface_eip_r2m_condition()(p. 530)
- app_weld_set_interface_eip_r2m_option()(p. 533)
- app_weld_set_interface_eip_m2r_process()(p. 537)
- app_weld_set_interface_eip_m2r_monitoring()(p. 540)
- app_weld_set_interface_eip_m2r_other()(p. 543)
- app_weld_get_welding_cond_digital()(p. 554)
- app_weld_set_weld_cond_digital()(p. 547)
- app_weld_adj_welding_cond_digital()(p. 551)
- app_weld_disable_digital()(p. 518)
- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)
- movel()(p. 63)
- amovel()(p. 102)
- movec()(p. 72)
- amovec()(p. 108)
- moveb()(p. 84)
- amoveb()(p. 118)
- movesx()(p. 81)
- amovesx()(p. 115)

9.3.14 app_weld_enable_analog()

Definition

app_weld_enable_analog(ch_v_out=[1,0], spec_v_out=[0,0,0,0], ch_f_out=[2,0], spec_f_out=[0,0,0,0], ch_v_in=[1,0], spec_v_in=[0,0,0,0], ch_c_in=[2,0], spec_c_in=[0,0,0,0],ch_arc_on=1,ch_gas_on=2,ch_inching_fwd=3,ch_inching_bwd=4, ...)

Features

This enables the analog welding function. It enters the connection and environment information of the available welding machine's analog I/O and digital signal output as input factors.

The target welding machine must support an analog interface so it can receive target current and target voltage inputs from the analog output channel of the connected controller. Set the channel number (1 or 2) and output mode (current/voltage) of the physically connected analog channel in ch_v_out and ch_f_out. The analog input/output range of the controller is 0-10 V for voltage mode and 4-20 mA for current mode. Make sure to set the mode and output range of each channel to be compatible with the input specification and range of the welding machine. For example, if the target input range of the welding machine is 0-10 V, it is ideal to set the output channel of the controller as voltage mode (0-10 V output range). Another example would be if the input channel specification of the welding machine is 2-15 V. In this case, set the analog channel current mode (4-20 mA output range) on the corresponding controller and connect a 75 ohm resistance to output a 3-15 V voltage. (In this case, the 2-3 V range, which cannot be set using the controller, cannot set a target.) It is recommended that the welding machine be set with as large an input range as possible.

Set the maximum and minimum range of the controller analog output in spec_v_out and spec_f_out.

First item of spec_v_out/spec_f_out = WO_min

Second item of spec_v_out/spec_f_out = CO_min

Third item of spec_v_out/spec_f_out = WO_max

Fourth item of spec_v_out/spec_f_out = CO_max

Where, WO_min and WO_max are the minimum and maximum output of the welder, and CO_min and CO_max are the controller analog output corresponding to WO_min and WO_max.

Note

The welding current varies according to the wire feeding speed, basic material, material/type/stick-out of the welding wire, and welding voltage, and this must be monitored with the welding machine or a separate current sensor.

In order to monitor the voltage/current measurements during welding, it is necessary to connect an analog output welding machine or a separate sensor. Set the analog input channel number and input mode of the corresponding controller in ch_v_in and ch_c_in.

Set the maximum and minimum input range of sensor measured in spec_v_in and spec_f_in.

First item of spec_v_in/spec_c_in = SO_min

Second item of spec_v_in/spec_c_in = CI_min

Third item of spec_v_in/spec_c_in = SO_max

Fourth item of spec_v_in/spec_c_in = CI_max

Where, SO_min and SO_max are the minimum and maximum sensor, and CI_min and CI_max are the controller input corresponding to SO_min and SO_max.

Set the channel numbers for ARC-ON/OFF (gas output signal - start/end), GAS-ON/OFF (gas output signal - start/end), INCHING-Forward-ON/OFF (forward wire feed signal - start/end), INCHING-Backward-ON/OFF (backward wire feed signal - start/end), and BlowOut-ON/OFF (torch cleaning gas output signal - start/end), which connect to the welding machine using digital contact method. In the case of signal outputs other than the ARC-ON/OFF signal, enter them selectively, depending on whether the welding machine supports the corresponding function.

Parameters

Parameter Name	Data Type	Default Value	Description
ch_v_out	list(int[2])	1	Target Voltage Analog Output Channel (1-2) If no designation is made: 0
		0	0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V)
spec_v_out	list(float[4])	0	Welder Output Voltage (V) Minimum (a)
		0	Controller Output corresponding to (a)
		0	Welder Output Voltage (V) Maximum (b)
		0	Controller Output corresponding to (b)
ch_f_out	list(int[2])	2	Feeding Speed Command Analog Output Channel (1-2) If no designation is made: 0
		0	0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V)
spec_f_out	list(float[4])	0	Feeding Speed (m/min) Minimum (c)
		0	Controller Output corresponding to (c)
		0	Feeding Speed (m/min) Maximum (d)
		0	Controller Output corresponding to (d)

Parameter Name	Data Type	Default Value	Description
ch_v_in	list(int[2])	1	Voltage Sensor Analog Input Channel (1-2) If no sensor is present: 0
		0	0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V)
spec_v_in	list(float[4])	0	Voltage Sensor Input (V) Minimum (e)
		0	Controller Input corresponding to (e)
		0	Voltage Sensor Input (V) Maximum (f)
		0	Controller Input corresponding to (f)
ch_c_in	list(int[2])	2	Current Sensor Analog Input Channel (1-2) If no sensor is present: 0
		0	0: Current Mode (4~20 mA) 1: Voltage Mode (0~10 V)
spec_c_in	list(float[4])	0	Current Sensor Input (A) Minimum (g)
		0	Controller Input corresponding to (g)
		0	Current Sensor Input (A) Maximum (h)
		0	Controller Input corresponding to (h)
ch_arc_on	int	1	Welding Output Digital Output Channel (1-16)
ch_gas_on	int	2	Protective Gas Digital Output Channel (1~16) If no connection is present: 0
ch_inching_fwd	int	3	Welding Wire Forward Stick-Out Digital Output Channel (1-16) If no connection is present: 0

Parameter Name	Data Type	Default Value	Description
ch_inching_bwd	int	4	Welding Wire Reverse Stick-Out Digital Output Channel (1-16) If no connection is present: 0
ch_blow_out	int	5	Torch Cleaning Gas Output Digital Channel (1~16) If no connection is present: 0

Return

Value	Description
0	Enable Welding Success
Negative Value	Enable Welding Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)

# Voltage Output (Channel 1, Voltage Mode), Welder Voltage Specification (Min/
Max)=(0-300)
# Feeding Speed Output (Channel 2, Voltage Mode), Feeding Speed Specification (Min/
Max)=(0-40
```

```
# Voltage Sensing (Channel 1, Voltage Mode), Sensor Measurement Specification (Min/
Max)=(0-300)
# Current Sensing (Channel 2, Voltage Mode), Sensor Specification (Min/Max)=(0-40)
# Start Welding Signal (Channel 1), Gas Output Signal (Channel 2), Wire Forward
Stick-Out Signal (Channel 3),
# Wire Reverse Stick-Out Signal (Channel 4), Torch Cleaning Gas Output Signal
(Channel 5)
app_weld_disable_analog()
```

Related commands

- app_weld_disable_analog()(p. 562)
- app_weld_set_weld_cond_analog()(p. 563)
- app_weld_adj_welding_cond_analog()(p. 567)
- app_weld_get_welding_cond_analog()(p. 570)
- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)

9.3.15 app_weld_disable_analog()

Features

This disables the analog welding function.

Return

Value	Description
0	Success
Negative Value	Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)

app_weld_disable_analog()
```

Related commands

- app_weld_enable_analog()(p. 557)
- app_weld_set_weld_cond_analog()(p. 563)
- app_weld_adj_welding_cond_analog()(p. 567)
- app_weld_get_welding_cond_analog()(p. 570)
- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)

9.3.16 app_weld_set_weld_cond_analog()

Definition

app_weld_set_weld_cond_analog(flag_dry_run=0, v_target=0, f_target=0, vel_target=0, vel_min=0, vel_max=0, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5])

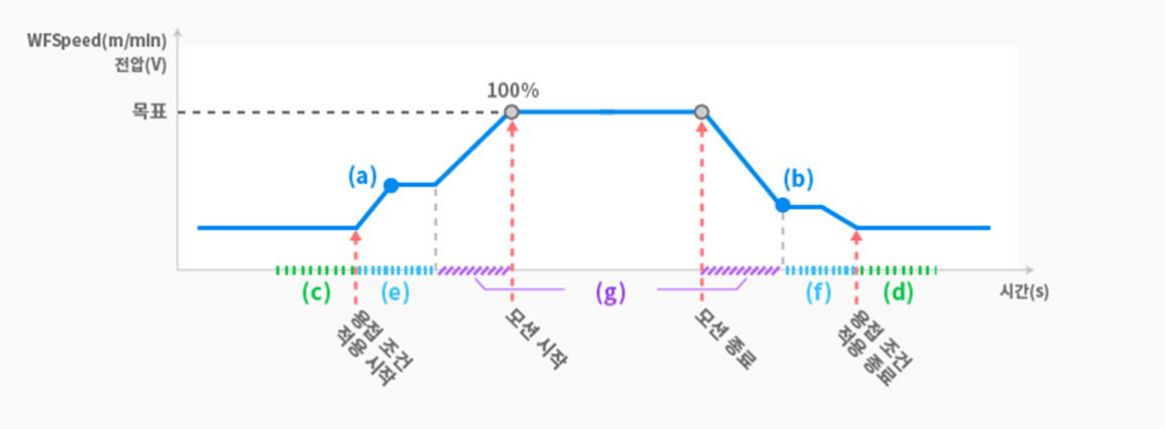
Features

This sets the analog welding condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()) commands, and any operations starting at a point outside the welding section will generate an error. The Welding Parameter (weld_proc_param) of the welding condition displays detailed conditions, including gas during start/end welding and condition maintenance time. Refer to the figure below for the values to enter. Only one welding condition is allowed in a single welding section. This welding condition can be adjusted during welding with the

app_weld_adj_welding_cond_analog() command while the voltage/feeding speed/speed (and weaving offset) can be also adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()).

Note

The welding current varies according to the wire feeding speed, basic material, material/type/stick-out of the welding wire, and welding voltage, and this must be monitored with the welding machine or a separate current sensor.



(a)Rsf/Rsv (b)Rff/Rfv (c)Tss (d)Tsf (e)Tas (f)Taf (g)Twc

Parameters

Parameter Name	Data Type	Default Value	Description
flag_dry_run	int	0	Dry-Run Mode Actual Welding (0) Dry-Run (1): Motion/Weaving/Offset only
v_target	float	0	Target Voltage (V)
f_target	float	0	Target Feeding Speed (m/min)
vel_target	float	0	Target Speed (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)

Parameter Name	Data Type	Default Value	Description
vel_min	float	0	Minimum Target Speed Correction (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)
vel_max	float	0	Maximum Target Speed Correction (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)
weld_proc_param	list(float[9])	0.2	Rsf (Feeding Speed Start Condition/Target Condition Ratio) (0< Rsf <= 1)
		0.2	Rsv (Voltage Start Condition/Target Condition Ratio) (0< Rsv <= 1)
		0.5	Tss (Protective Gas Discharge Time Before Welding, sec) (0<= Tss)
		0.5	Tas (Start Welding Condition Maintenance Time, sec) (0<= Twc)
		0.5	Twc (Change Welding Condition Time, sec) (0<= Twc)
		0.2	Rff (Feeding Speed End Condition/Target Condition Ratio) (0< Rff <= 1)
		0.2	Rfv (Voltage End Condition/Target Condition Ratio) (0< Rfv <= 1)
		0.5	Taf (End Welding Condition Maintenance Time, sec) (0<= Taf)
		0.5	Tsf (Protective Gas Discharge Time After Welding, sec) (0<= Tsf)

Return

Value	Description
0	Setting Success
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=24, f_target=20,
vel_target=60, vel_min=10,
vel_max=100, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5])
# Target Voltage/Feeding Speed = 24 V, 20 m/min, Welding Speed=60 mm/sec (=1 cm/min),
Actual Welding, Use Default Welding Parameter
app_weld_disable_analog()
```

Related commands

- app_weld_enable_analog()(p. 557)
- app_weld_adj_welding_cond_analog()(p. 567)
- app_weld_get_welding_cond_analog()(p. 570)
- app_weld_weave_cond_trapezoidal()(p. 575)

- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)

9.3.17 app_weld_adj_welding_cond_analog()

Definition

app_weld_adj_welding_cond_analog(flag_reset=0, v_target=None, f_target=None, vel_target=None, wv_offset=None, wv_width_ratio=None)

Features

This adjusts the welding condition and weaving condition during analog welding. It is used to change the welding condition of each section in a series of sections before calling the motion command (movel(), movec(), moveb(), movesx()). If an adjustment factor is entered using this command, the corresponding welding condition and weaving condition are adjusted, and real-time adjustment of the welding/weaving condition from the welding monitoring information screen of the TP becomes unavailable. Execute flag_reset=1 to reset the adjusted condition to the main condition set using app_weld_set_weld_cond_analog() and app_weld_weave_cond_trapezoidal(). Setting flag_reset=1 will reset to the final condition adjusted in real-time using the TP (the weaving width ratio (wv_width_ratio), which cannot be adjusted in real-time, is changed to 1), and the welding condition can be adjusted in real-time from the TP.

Parameters

Parameter Name	Data Type	Default Value	Description
flag_reset	int	0	0: Apply Adjustment 1: Default Target (app_weld_set_weld_cond_analog()) Apply Value
v_target	float	-	Target Voltage (V)
f_target	float	-	Feeding Speed (m/min))
vel_target	float	-	Target Speed (mm/sec) • Take note that the unit is different from that of the teaching pendant input (cm/min)
wv_offset	float[2]	-	Weaving Coordinate-Y Direction Offset (mm)

Parameter Name	Data Type	Default Value	Description
		-	Weaving Coordinate-Z Direction Offset (mm)
wv_width_ratio	float	-	Changed Weaving Width/Set Weaving Width Ratio (0-2)

Note
Conditions which do not designate a value in factors v_target/f_target/vel_target/wv_offset/ wv_width_ratio will maintain the current condition (including conditions with real-time adjustments), so only set factors that require adjustment. However, in the case of wv_offset, even if an adjustment is made only to the Y direction or Z direction, values for both sequences must be entered.

Return

Value	Description
0	Setting Success
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
movej(posj(0,0,90,0,90,0),v=30,a=60)

pt1= posx(559, 434.5, 651.5, 45, 180, 45)
```

```
pt2= posx(559, 434.5, 151.5, 45, 180, 45)
pt3= posx(559, 0.0, 151.5, 45, 180, 45)

app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3,
ch_inching_bwd=4, ch_blow_out=5)

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=24, f_target=20,
vel_target=60, vel_min=10,
vel_max=100, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5])

movel(pt1, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)

app_weld_adj_welding_cond_analog(flag_reset=0, v_target=20, f_target=10,
vel_target=30, wv_offset=[20,10], wv_width_ratio=0.5)
movel(pt2, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)
app_weld_adj_welding_cond_analog(flag_reset=1)
movel(pt3, v=5, a=5, app_type=DR_MV_APP_WELD)
# Start Point → pt1: Apply Origin Welding Condition (24V, 20 m/min)
# pt1 → pt2: Apply Adjusted Condition (20 V, 10 m/min)
# pt2 → pt3: Apply Origin Condition (24 V, 20 m/min)

app_weld_disable_analog()
```

Related commands

- app_weld_enable_analog()(p. 557)
- app_weld_set_weld_cond_analog()(p. 563)
- app_weld_get_welding_cond_analog()(p. 570)
- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)
- movel()(p. 63)
- amovel()(p. 102)
- movec()(p. 72)
- amovec()(p. 108)
- moveb()(p. 84)
- amoveb()(p. 118)
- movesx()(p. 81)
- amovesx()(p. 115)

9.3.18 app_weld_get_welding_cond_analog()

Features

This monitors the welding status during analog welding. Values available for monitoring are the current target voltage/current/speed/weaving offset/digital output signal and measured voltage/current and welding status. If measured voltage/current is not set (if ch_v_in and ch_c_in are not set during app_weld_enable()), the output of the corresponding values are set equal to the target voltage (v_target)/feeding speed (f_target). In addition, it is possible to check the fail status using the welding status factor.

Return

Value	Description
v_target	Current Target Voltage (V) (target with adjustment applied)
f_target	Current Target Feeding Speed (m/min) or Target Current (A) (target with adjustment applied)
vel_target	Current Target Speed (mm/sec) (target with adjustment applied) * Take note that the unit is different from that of the monitoring output unit (cm/min) of the teaching pendant
v_meas	Current Measured Voltage (V)
c_meas	Current Measured Current (A)
wv_offset[2]	Current Target Offset (Y and Z directions, mm) (target with adjustment applied)
sig_out[4]	Digital Output Signal (arc_on, gas_on, inching_fwd, inching_bwd)
status	Non-weld:0, Weld (Normal): 1, Weld (Abnormal): 9, Dry-run: 99

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
movej(posj(0,0,90,0,90,0),v=30,a=60)

pt1= posx(559, 434.5, 651.5, 45, 180, 45)
pt2= posx(559, 434.5, 151.5, 45, 180, 45)
pt3= posx(559, 0.0, 151.5, 45, 180, 45)

app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3,
ch_inching_bwd=4, ch_blow_out=5)

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=24, f_target=20,
vel_target=60, vel_min=10,
vel_max=100, weld_proc_param=[0.2,0.2,0.5,0.5,0.5,0.2,0.2,0.5,0.5])

movel(pt1, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)
app_weld_adj_welding_cond_analog(flag_reset=0, v_target=20, f_target=10,
vel_target=30, wv_offset=[20,10], wv_width_ratio=0.5)
movel(pt2, v=5, a=5, r=30, app_type=DR_MV_APP_WELD)
app_weld_adj_welding_cond_analog(flag_reset=1)
amovel(pt3, v=5, a=5, app_type=DR_MV_APP_WELD)

while True:
  Vt, Ft, velt, Vm, Cm, Off, Dout, status = app_weld_get_welding_cond_analog()
    if status == 9:
      tp_popup("welding error!! ", DR_PM_ALARM, 1)
    # An alarm is generated if abnormal welding occurs (status=9)
    else :
      if check_motion()==0:
        break

app_weld_disable_analog()
```

Related commands

- app_weld_enable_analog()(p. 557)
- app_weld_set_weld_cond_analog()(p. 563)
- app_weld_adj_welding_cond_analog()(p. 567)

- app_weld_weave_cond_trapezoidal()(p. 575)
- app_weld_weave_cond_zigzag()(p. 578)
- app_weld_weave_cond_circular()(p. 580)
- app_weld_weave_cond_sinusoidal()(p. 582)

9.3.19 app_weld_get_extreme_point()

Definition

app_weld_get_extreme_point(time=50)

Features

Sends the delay in acquiring the outermost angle information during the weld weaving motion from the parent controller to the child controller.

Parameters

Parameter Name	Data Type	Default Value	Description
time	float	50(ms)	Delay time(ms)

Return

Value	Description
left	Weaving Coordinate System Returns Value on Reach Before Delay of Outermost Angle Point Left of Proceeding Direction • On Reach : 1 • Not reached : 0
right	Returns a value when the outermost angle point to the right of the weaving coordinate system progression is reached before the delay time. • On Reach : 1 • Not reached : 0

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error

Exception	Description
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
left, right = app_weld_get_extreme_point(time=50)
```

9.3.20 app_weld_adj_motion_offset()

Definition

app_weld_adj_motion_offset(offset=[0.0, 0.0])

Features

A function to control Y and Z coordinates in real time when the device's welding seem tracking function is in operation.

Parameters

Parameter Name	Data Type	Default Value	Description
offset	float[2]	0.0, 0.0	offset[0]: Y coordinate's offset offset[1]: Z coordinate's offset

Return

Value	Description
0	Setting success
Minus value	Setting fail

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
def arc_sensing_model(left, right, time):

## Edit your own seam tracking algorithm in this function

# This example is assumed as below
# 1. Seam tracking algorithm is in welding machine
# 2. Use digital IO communication between robot controller and welding machine

    if left == 1:
        set_digital_output(15, ON)
    elif left == 0:
        set_digital_output(15, OFF)

    if right == 1:
        set_digital_output(16, ON)
    elif right == 0:
        set_digital_output(16, OFF)

    y_offset = 0
    z_offset = 0
    if get_digital_input(9) == OFF:
        y_offset = 10
        z_offset = 0
    if get_digital_input(10) == ON:
        y_offset = -10
        z_offset = 0
    if get_digital_input(11) == ON:
        y_offset = 0
        z_offset = 10
    if get_digital_input(12) == ON:
        y_offset = 0
```

```
        z_offset = -10

    offset=[y_offset, z_offset]

    return offset

def seam_tracking():
    Vt, Ct, Ft, velt, Vm, Cm, Off, Dout, status = app_weld_get_welding_cond_analog()
    if status == 99:
        left, right = app_weld_get_extreme_point(time=90)
        offset=arc_sensing_model(left, right, time)
        app_weld_adj_motion_offset(offset)

set_velj(30)
set_accj(60)
set_velx(30)
set_accx(30)

movej(posj(0,0,90,0,90,0))

app_weld_enable_analog(ch_v_out=[0,0], spec_v_out=[0,0,0,0], ch_f_out =[0,0],
spec_f_out =[0,0,0,0], ch_v_in =[0,0], spec_v_in =[0,0,0,0], ch_c_in =[0,0],
spec_c_in=[0,0,0,0], ch_arc_on=0, ch_gas_on=0, ch_inching_fwd=0, ch_inching_bwd=0,
ch_blow_out=0)

th_id = thread_run(seam_tracking, loop=True) #Activate Saem Tracking Function

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=0, f_target=0, vel_target=10,
vel_min=0,
vel_max=100, weld_proc_param=[0,0,0,0,0,0,0,0,0,0])

app_weld_weave_cond_zigzag(wv_offset=[-10,0], wv_ang=0, wv_param=[10,1])
# zigzag weaving pattern, offset=0,0 tilt angle=0, weaving width=10(mm), weaving
period=0.5(sec)

app_weld_adj_welding_cond_analog(flag_reset=1)
movel(posx(619.50, -50, 982.80, 0, -180, 0), app_type=DR_MV_APP_WELD)

thread_stop(th_id) #Deactivate Seam Tracking Function

app_weld_disable_analog()
```

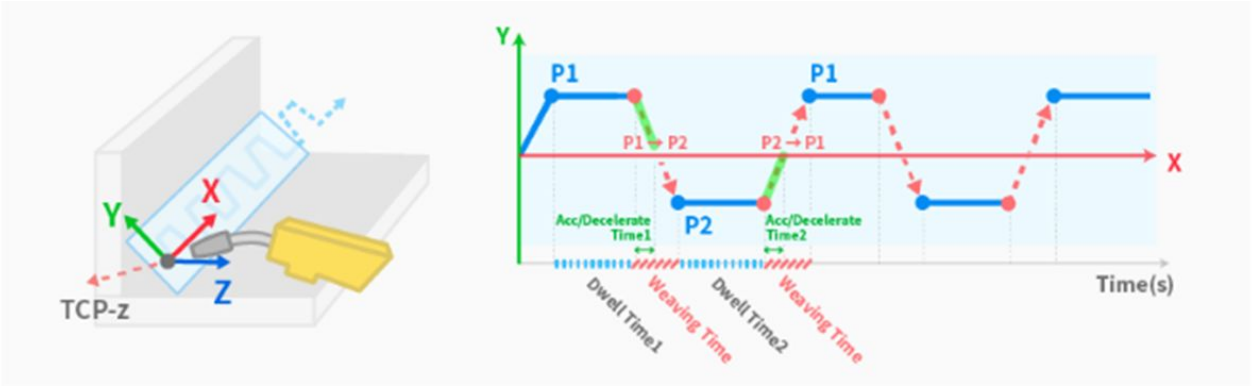
9.3.21 app_weld_weave_cond_trapezoidal()

Definition

app_weld_weave_cond_trapezoidal(wv_offset=[0,0], wv_ang=0, wv_param=[0,1.5,0,-1.5,0.3,0.1,0.3,0.3,0.1,0.3])

Features

This sets the trapezoidal weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()/ app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog()/ app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()/ app_weld_set_weld_cond_digital()).



Parameters

Parameter Name	Data Type	Default Value	Description
wv_offset	float[2]	0	Weaving Coordinate-Y Direction Offset (mm)
		0	Weaving Coordinate-Z Direction Offset (mm)
wv_ang	float	0	Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg)
wv_param	list(float[10])	0	Weaving Point 1-x (mm)
		1.5	Weaving Point 1-y (mm)

Parameter Name	Data Type	Default Value	Description
		0	Weaving Point 2-x (mm)
		-1.5	Weaving Point 2-y (mm)
		0.3	Weaving Point 1→2 hrs (sec)
		0.1	Weaving Point 1→2 Dec/Acc Time (sec)
		0.3	Weaving Point 1 Dwell Time (sec)
		0.3	Weaving Point 2→1 hr (sec)
		0.1	Weaving Point 2→1 Dec/Acc Time (sec)
		0.3	Weaving Point 2 Dwell Time (sec)

Return

Value	Description
0	Setting Success
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=200, f_target=150,
vel_target=10, vel_min=10,
vel_max=100, weld_proc_param=[0.5,0.3,2,1,0.7,0.4,0.7,0.6,1.5])

app_weld_weave_cond_trapezoidal(wv_offset=[0,0], wv_ang=0,
wv_param=[0,5,0,-5,0.7,0.2,0.5,0.7,0.2,0.5])

# Trapezoidal Weaving Pattern, Offset=0,0, Tilt Angle=0, Weaving Point 1=(0,5),
Weaving Point 2=(0,-5), Weaving Time=0.7 (sec) (same in both directions), Weaving
Dec/Acc Time=0.2 (sec) (same in both directions), Weaving Point 1 Dwell Time=0.5 sec,
Weaving Point 2 Dwell Time=0.5 sec
app_weld_disable_analog ()
```

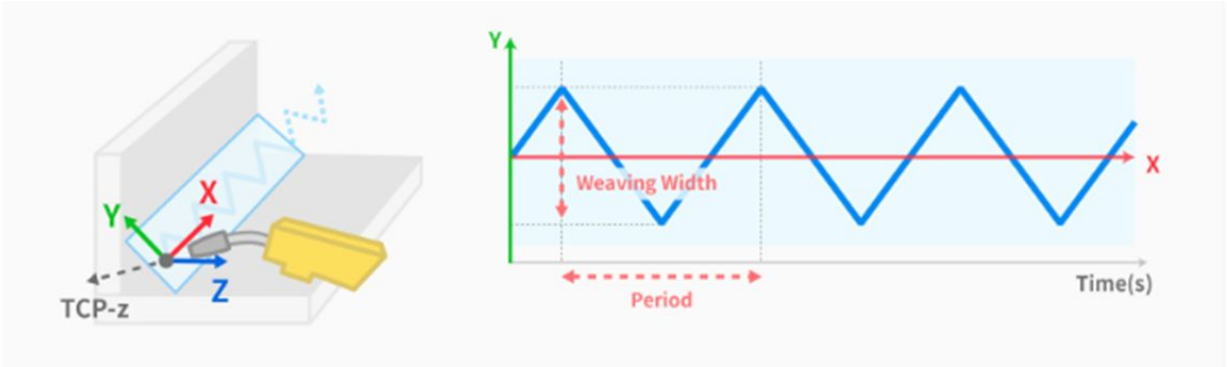
9.3.22 app_weld_weave_cond_zigzag()

Definition

app_weld_weave_cond_zigzag(wv_offset=[0,0], wv_ang=0, wv_param=[3,0.6])

Features

This sets the zigzag weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()/ app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog()/ app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()/ app_weld_set_weld_cond_digital()).



Parameters

Parameter Name	Data Type	Default Value	Description
wv_offset	float[2]	0	Weaving Coordinate-Y Direction Offset (mm)
		0	Weaving Coordinate-Z Direction Offset (mm)
wv_ang	float	0	Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg)
wv_param	list(float[2])	3	Weaving Width (mm)
		0.6	Weaving Period (sec)

Return

Value	Description
0	Setting Success
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=200, f_target=150,
vel_target=10, vel_min=10,
vel_max=100, weld_proc_param=[0.5,0.3,2,1,0.7,0.4,0.7,0.6,1.5])

app_weld_weave_cond_zigzag(wv_offset=[0,0], wv_ang=0, wv_param=[10,0.5])
# Zigzag Weaving Pattern, Offset=0,0 Tilt Angle=0, Weaving Width=10 (mm), Weaving
Period=0.5 (sec)
app_weld_disable_analog()
```

9.3.23 app_weld_weave_cond_circular()

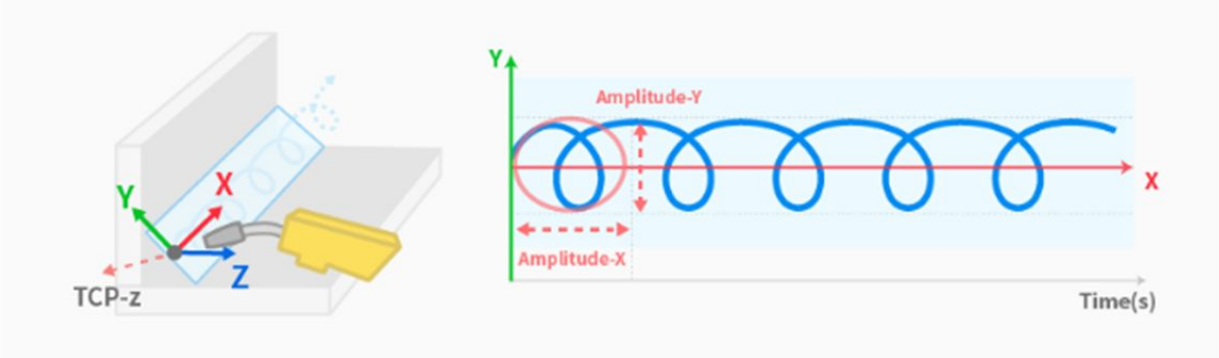
Definition

app_weld_weave_cond_circular(wv_offset=[0,0], wv_ang=0, wv_param=[3,3,0.3,0.3])

Features

This sets the circular weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()/ app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog()/ app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET

using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()/ app_weld_set_weld_cond_digital()).



Parameters

Parameter Name	Data Type	Default Value	Description
wv_offset	float[2]	0	Weaving Coordinate-Y Direction Offset (mm)
		0	Weaving Coordinate-Z Direction Offset (mm)
wv_ang	float	0	Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg)
wv_param	list(float[4])	3	X Direction Weaving Width (mm)
		3	Y Direction Weaving Width (mm)
		0.3	X Direction Weaving Period (sec)
		0.3	Y Direction Weaving Period (sec)

Return

Value	Description
0	Setting Success
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=200, f_target=150,
vel_target=10, vel_min=10,
vel_max=100, weld_proc_param=[0.5,0.3,2,1,0.7,0.4,0.7,0.6,1.5])

app_weld_weave_cond_circular(wv_offset=[0,0], wv_ang=0, wv_param=[3,3,0.3,0.3])
# Circular Weaving Pattern, Offset=0,0 Tilt Angle=0, X Direction Weaving Width=3
(mm), Y Direction Weaving=3 (mm), X Direction Weaving Period=0.3 (s), Y Direction
Weaving Period=0.3 (s)
app_weld_disable_analog()
```

9.3.24 app_weld_weave_cond_sinusoidal()

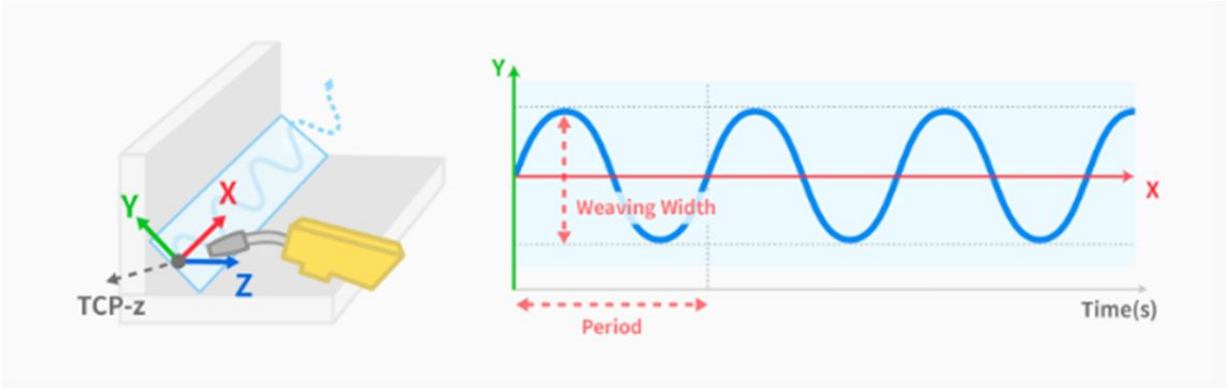
Definition

app_weld_weave_cond_sinusoidal(wv_offset=[0,0], wv_ang=0, wv_param=[3,0.6])

Features

This sets the sinusoidal weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()/ app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will

generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog()/ app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()/ app_weld_set_weld_cond_digital()).



Parameters

Parameter Name	Data Type	Default Value	Description
wv_offset	float[2]	0	Weaving Coordinate-Y Direction Offset (mm)
		0	Weaving Coordinate-Z Direction Offset (mm)
wv_ang	float	0	Weaving Coordinate-Weaving Plane Tile Angle centering on X-Axis (deg)
wv_param	list(float[2])	3	Weaving Width (mm)
		0.6	Weaving Period (sec)

Return

Value	Description
0	Setting Success

Value	Description
Negative value	Setting Failure

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)

app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=200, f_target=150,
vel_target=10, vel_min=10,
vel_max=100, weld_proc_param=[0.5,0.3,2,1,0.7,0.4,0.7,0.6,1.5])

app_weld_weave_cond_sinusoidal(wv_offset=[0,0], wv_ang=0, wv_param=[10,0.5])
# Sinusoidal Weaving Pattern, Offset=0,0 Tilt Angle=0, Weaving Width=10 (mm), Weaving
Period=0.5 (s)
app_weld_disable_analog()
```

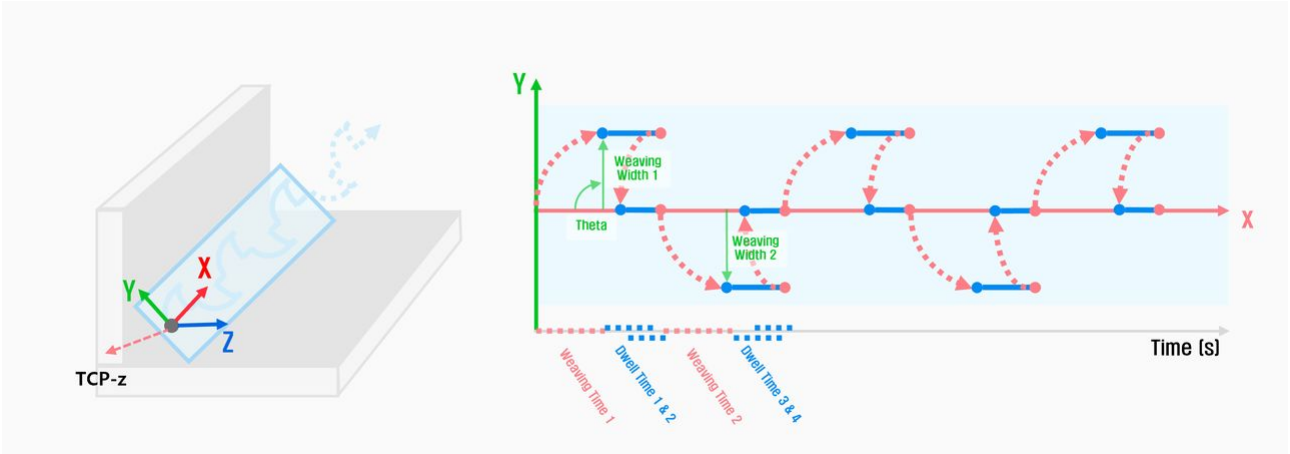
9.3.25 app_weld_weave_cond_cwave()

Definition

app_weld_weave_cond_cwave(wv_offset=[0,0], wv_ang=0, wv_param=[45, 10, 10, 0.5, 0.5, 0.2, 0.2, 0.2, 0.2])

Features

This sets the c-wave weaving condition. It is only valid within the welding section defined with the Enable Welding (app_weld_enable_analog()) and Disable Welding (app_weld_disable_analog()/ app_weld_disable_digital()) commands, and any operations starting at a point outside the welding section will generate an error. The weaving condition is defined by the weaving coordinates, which are defined with the TCP direction from the weaving x-axis as the weaving z-axis, and the vector-multiplied (cross product) direction as the weaving y-axis with the welding path direction as the weaving x-axis. Refer to the figure below for the coordinates and weaving setting factor. Only one weaving condition is allowed in a single welding section. The offset or weaving width can be adjusted during welding with the app_weld_adj_welding_cond_analog()/ app_weld_set_weld_cond_digital() command or the voltage/current/speed and the offset can be adjusted from the Welding Condition Adjustment popup of the teaching pendant. However, from the teaching pendant, welding condition adjustments are only available if the welding condition adjustment status is set to RESET using a command (the welding condition setting is designated with app_weld_set_weld_cond_analog()/ app_weld_set_weld_cond_digital()).



Parameter

Parameter Name	Data Type	Default Value	Description
wv_offset	float[2]	[0, 0]	[0] Weave Coordinate System - Y Offset (mm) [1] Weave coordinate system-z offset (mm)
wv_angle	float	0	Rotation angle of the weave plane about the weave coordinate system-x axis (deg)

Parameter Name	Data Type	Default Value	Description
wv_param	list(float[9])	[45, 10, 10, 0.5, 0.5, 0.2, 0.2, 0.2, 0.2]	[0] Weaving Theta (deg) [1] Weaving Width1 (mm) [2] Weaving Width2 (mm) [3] Weaving Time1 (sec) [4] Weaving Time2 (sec) [5] Dwell Time1 (sec) [6] Dwell Time 2 (sec) [7] Dwell Time 3 (sec) [8] Dwell Time 4 (sec)

Return

Value	Description
0	success
negative value	fail

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_enable_analog(ch_v_out=[1,1], spec_v_out=[0,0,300,10], ch_f_out =[2,1],
spec_f_out =[0,0,40,10], ch_v_in =[1,1], spec_v_in =[0,0,300,10], ch_c_in =[2,1],
spec_c_in=[0,0,40,10], ch_arc_on=1, ch_gas_on=2, ch_inching_fwd=3, ch_inching_bwd=4,
ch_blow_out=5)
```

```
app_weld_set_weld_cond_analog(flag_dry_run=1, v_target=200, f_target=150,
vel_target=10, vel_min=10,
vel_max=100, weld_proc_param=[0.5,0.3,2,1,0.7,0.4,0.7,0.6,1.5])

app_weld_weave_cond_cwave(wv_offset=[0,0], wv_ang=0, wv_param=[45, 10, 10, 0.5, 0.5,
0.2, 0.2, 0.2, 0.2])
# C-wave Weaving Pattern, Offset=0,0 Tilt Angle=0, Weaving Theta=45(deg), Weave
Width=10(mm), Weaving Time=0.5(sec), Dwell Time=0.2(sec)
app_weld_disable_analog()
```

9.3.26 app_weld_signal_contact_status()

Definition

app_weld_signal_contact_status(ch_arc=[0,1], ch_gas=[0,1], ch_wire=[0,1], ch_mcn=[0,1], timeout=0.1)

Features

It monitors the arc, gas, and wire signals during welding and raises an error handling popup if there is an abnormality.

Parameter

Parameter Name	Data Type	Default Value	Description
ch_arc	list(int[2])	[0,1]	[0] Arc Status Digital Input Channel (1-16), (default: 0) [1] Contact status (A contact:1, B contact: 2), (default : 1)
ch_gas	list(int[2])	[0,1]	[0] Gas status Digital input channel (1-16), (default: 0) [1] Contact status (A contact: 1, B contact: 2), (default : 1)
ch_wire	list(int[2])	[0,1]	[0] Wire Status Digital Input Channel (1~16), (default: 0) [1] Contact status (A contact: 1, B contact: 2), (default : 1)
ch_mcn	list(int[2])	[0,1]	[0] Machine Status's Digital Input Channel (1~16), (default: 0) [1] Contact status (A contact: 1, B contact: 2), (default: 1)
timeout	float	1.0	The maximum waiting time about Arc-Wait (default: 1.0 sec)

Return

Value	Description
0	success
negative value	fail

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data error
DR_Error (DR_ERROR_VALUE)	Invalid parameter value
DR_Error (DR_ERROR_RUNTIME)	C Extension module error
DR_Error (DR_ERROR_STOP)	Program terminated forcibly

Example

```
app_weld_signal_contact_status(ch_arc=[0,1], ch_gas=[0,1], ch_wire=[0,1],
ch_mcn=[0,1], timeout=1.0)
```

10 A-Series Command

10.1 Controller

10.1.1 get_function_input()

Definition

get_function_input(index)

Features

This function reads a state of the function button from the process button device.

Parameters

Parameter Name	Data Type	Default Value	Description
index	int	-	It is the index of the function button mounted on the process button to be read. It is available 1 to 4.

Return

Value	Description
1	ON
0	OFF
Negative value	Failed

Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

Example

```
in1 = get_function_input(1)           # Reads the no. 1 function button input
in8 = get_function_input(4)           # Reads the no. 4 function button input
```