ROS

Doosan Robot

M0609 | M0617 | M1013 | M1509

ROS Programming Manual

DOOSAN

1. Installation Doosan ROS package 8
1.1 Overview 8
1.2 Prerequisitese 8
1.3 Installation 8

2. Operation mode 9
2.1 Virtual mode 9
2.2 Real mode 9

3. dsr_description 12
3.1 dsr_description <robot_model>.launch 12

4. dsr_moveit 14
4.1 dsr_moveit_config 14
4.2 dsr_control + moveit 16
4.3 MoveIt Commander 18

5. dsr_launcher 20
5.1 dsr_launcher 20

6. dsr_example 23
6.1 Single Robot 24
6.2 Multi Robot 27
6.3 Gripper 30
6.4 Mobile robot 33

7. dsr_msgs 36
7.1 Topic 36

7.1.1 RobotState.msg 36
7.1.2 RobotStop.msg 39
7.1.3 RobotError.msg 40
7.1.4 LogAlarm.msg 42
7.1.5 ModbusState.msg 44
7.1.6 JogMultiAxis.msg 45
7.2 Service/motion 46
7.2.1 MoveJoint.srv 46
7.2.2 MoveLine.srv 48
7.2.3 MoveJointx.srv 51
7.2.4 MoveCircle.srv 53
7.2.5 MoveSplineJoint.srv 56
7.2.6 MoveSplineTask.srv 58
7.2.7 MoveBlending.srv 60
7.2.8 MoveSpiral.srv 62
7.2.9 MovePeriodic.srv 64
7.2.10 MoveWait.srv 67
7.2.11 Jog.srv 68
7.2.12 multiJog.srv 69
7.2.13 MovePause.srv 70
7.2.14 MoveResume.srv 71
7.2.15 MoveStop.srv 72
7.2.16 Trans.srv 73
7.3 Service/system 75
7.3.1 GetRobotMode.srv 75
7.3.2 SetRobotMode.srv 76
7.3.3 GetRobotSystem.srv 77
7.3.4 SetRobotSystem.srv 78
7.3.5 GetRobotSpeedMode.srv 79
7.3.6 SetRobotSpeedMode.srv 80
7.3.7 SetSafeStopResetType.srv 81

7.3.8 GetCurrentPose.srv 82
7.3.9 GetCurrentSiolutionSpace.srv 83
7.3.10 GetJointTorque.srv 84
7.3.11 GetExternalTorque.srv 85
7.3.12 GetToolForce.srv 86
7.3.13 GetLastAlarm.srv 87
7.4 Service/tcp 89
7.4.1 ConfigCreateTcp.srv 89
7.4.2 ConfigDeleteTcp.srv 90
7.4.3 GetCurrentTcp.srv 91
7.4.4 SetCurrentTcp.srv 92
7.5 Service/tool 93
7.5.1 ConfigCreateTool.srv 93
7.5.2 ConfigDeleteTool.srv 94
7.5.3 GetCurrentTool.srv 95
7.5.4 SetCurrentTool.srv 96
7.6 Service/io 97
7.6.1 SetCtlBoxDigitalOutput.srv 97
7.6.2 GetCtlBoxDigitalInput.srv 98
7.6.3 SetToolDigitalOutput.srv 99
7.6.4 GetToolDigitalIntput.srv 100
7.6.5 SetCtlBoxAnalogOutputType.srv 101
7.6.6 SetCtlBoxAnalogInputType.srv 102
7.6.7 SetCtlBoxAnalogOutput.srv 103
7.6.8 GetCtlBoxAnalogInput.srv 104
7.7 Service/modbus 105
7.7.1 ConfigCreateModbus.srv 105
7.7.2 ConfigDeleteModbus.srv 107
7.7.3 SetModbusOutput.srv 108
7.7.4 GetModbuInput.srv 109
7.8 Service/drl 110

7.8.1 DrlStart.srv 110
7.8.2 DrlStop.srv 111
7.8.3 DrlPause.srv 112
7.8.4 DrlResume.srv 113
7.8.5 GetDrlState.srv 114
7.9 Service/gripper 115
7.9.1 SerialSendData.srv 115
7.9.2 RobotiqMove.srv 116

8. Motion-related Functions 117
8.1 posj(q1=0, q2=0, q3=0, q4=0, q5=0, q6=0) 117
8.2 posx(x=0, y=0, z=0, w=0, p=0, r=0) 118
8.3 trans(pos, delta, ref, ref_out) 119
8.4 posb(seg_type, posx1, posx2=None, radius=0) 121
8.5 set_velj(vel) 123
8.6 set_accj(acc) 124
8.7 set_velx(vel1, vel2) 125
8.8 set_velx(vel) 126
8.9 set_accx(acc1, acc2) 127
8.10 set_accx(acc) 128
8.11 set_tcp(name) 129
8.12 movej 130
8.13 movel 133
8.14 movejx 136
8.15 movec 139
8.16 movesj 143
8.17 movesx 146
8.18 moveb 150
8.19 move_spiral 154
8.20 move_periodic 157

8.21 amovej 160
8.22 amovel 163
8.23 amovejx 166
8.24 amovec 169
8.25 amovesj 172
8.26 amovesx 175
8.27 amoveb 178
8.28 amove_spiral 181
8.29 amove_periodic 184
8.30 mwait(time=0) 187
8.31 stop(st_mode) 189
8.32 change_operation_speed(speed) 191

9. Auxiliary Control Functions 193
9.1 get_current_posj() 193
9.2 get_current_posx() 194
9.3 get_current_solution_space() 195
9.4 get_joint_torque() 196
9.5 get_external_torque() 197
9.6 get_tool_force() 198

10. Other Settings and Safety-related Functions 199
10.1 set_tool(name) 199

11. System Functions 200
11.1 Robot Mode 200
11.1.1 set_robot_mode(robot_mode) 200
11.1.2 get_robot_mode() 201
11.2 IO 202
11.2.1 set_digital_output(index, val=None) 202

11.2.2 get_digital_input(index) 204
11.2.3 set_tool_digital_output(index, val=None) 205
11.2.4 get_tool_digital_input(index) 206
11.2.5 set_mode_analog_output(ch, mod) 207
11.2.6 set_mode_analog_input(ch, mod) 208
11.2.7 set_analog_output(ch, val) 209
11.2.8 get_analog_input(ch) 210

11.3 Modbus 211

11.3.1 add_modbus_signal (ip, port, name, reg_type, index, value=0) 211
11.3.2 del_modbus_signal (name) 214
11.3.3 set_modbus_output(iobus, val) 215
11.3.4 get_modbus_input(iobus) 216

1. Installation Doosan ROS package

1.1 Overview

- **Doosan robotics ROS package**

Doosan robotics ROS package is a metapackage for running Doosan cooperative robots on ROS URDF model is provided, simulation is possible through Rviz, Gazebo, and the real robot can be driven through moveIt or various examples.

1.2 Prerequisitese

- **System**

You must use an x86 system.

We recommend a workstation-class PC for the best simulation.

- **OS & Distro**

Ubuntu 16.04(32/64bit) + ROS kinetic

1.3 Installation

- **Install the source from github**

Download and build the source from Doosan robotics

Github : **https://github.com/doosan-robotics/doosan-robot**

- **How to Install**

```
### We recoomand the /home/<user>/catkin_ws/src
$ mkdir -p /home/<user>/catkin_ws/src
$ cd /home/<user>/catkin_ws/src
$ catkin_init_workspace
$ git clone https://github.com/doosan-robotics/doosan-robot
$ rosdep install --from-paths doosan-robot --ignore-src --rosdistro kinetic -r -y
$ catkin_make
$ source ./devel/setup.bash
# In this manual, workspace was used as '~/catkin_ws'
```

2. Operation mode

2.1 Virtual mode

▪ Feature

- If you are driving without a real robot, use virtual mode.
- Selecting virtual mode sets the mode argument to virtual when running the dsr_aluncher launch file. If you omit the argument, it defaults to virsual. .

  ex> roslaunch dsr_launcher single_robot_gazebo.launch **mode:=virtual**
- When ROS launches in virtual mode, the emulator(DRCF) runs automatically.

  - (DRCF) location: doosan-robot/common/bin/ DRCF
- One emulator is required for each robot.
- When controlling multiple robots, the emulator will automatically run as many as the number of robots and use different port.

2.2 Real mode

▪ Feature

- Use real mode to drive a real robot.
- In real mode operation, communication must be established with the real robot controller.
- The default IP of the robot controller is 192.168.127.100 and the port is 12345.
- Selecting arguments(**mode:=real host:=192.168.127.100 port:=12345**) to real mode when running the dsr_launcher launch file.

  ex> roslaunch dsr_launcher single_robot_gazebo.launch **mode:=real host:=192.168.127.100 port:=12345**

▪ Connect with Controller

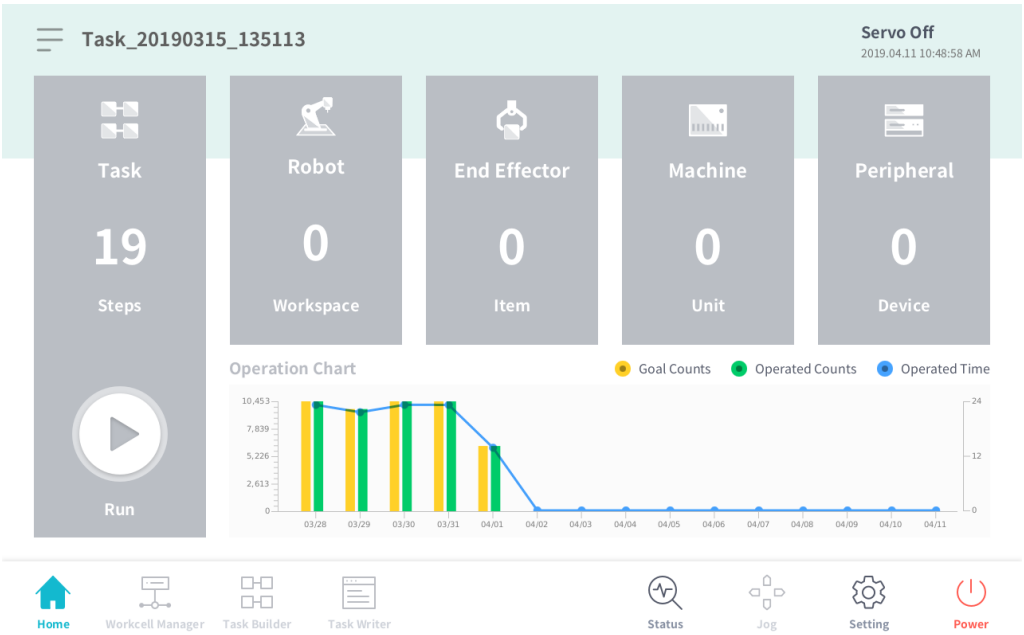


Figure 2.2 Teach Pandaunt Screen

- The user can set static IP in Setting -> Network of TP screen.

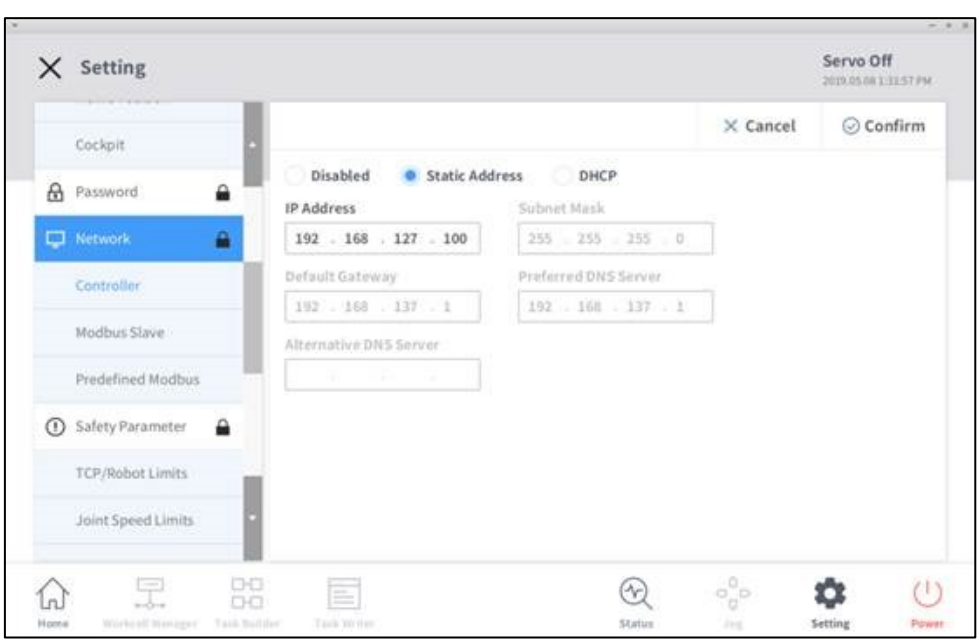


Figure 2.3 Check robot controller IP on TP

- Check the IP of the controller set in the Network tab, and set this IP in the ROS (host := ROBOT_IP)
- If the ROS control node is correctly executed, ROS has the robot control.
- If the TP transfer control, below pop-up message on the TP screen.

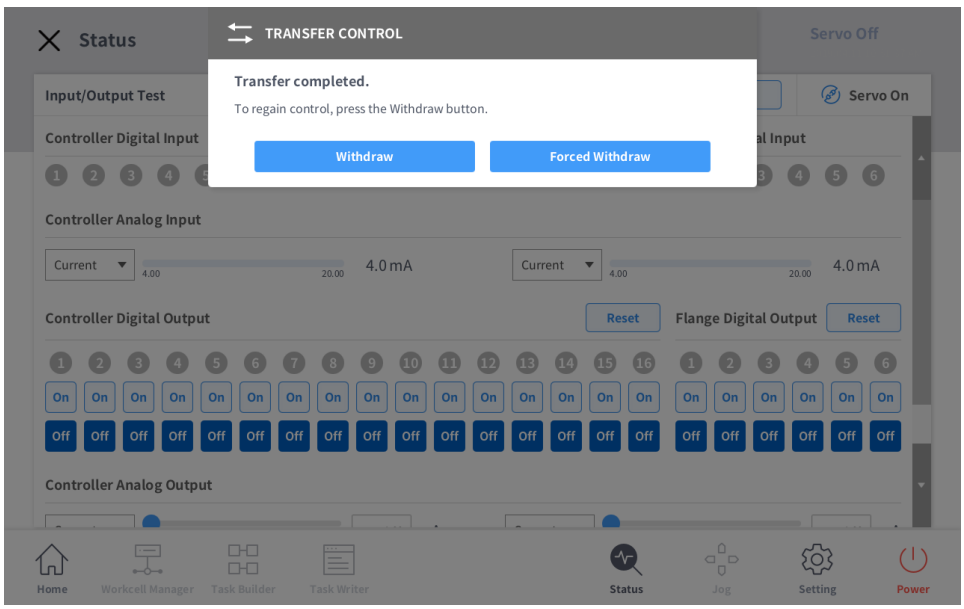


Figure 2.4 Transfer Control pop-up message

3. dsr_description

3.1 dsr_description <robot_model>.launch

- **Feature**
  - Launch the robot model on Rviz simulator and load Joint_state_publisher.
  - The robot can be moved by using Joint_state_publisher.

- **Parameter**

Parameter Name	Data Type	Default Value	Description
model	-	m1013	M-Series Robot model . m0609, m0617, m1013, m1509 A-Series Robot model . a0509
color	-	white	Robot Color . white or blue
gripper	-	none	using gripper or not . none : not use gripper . robotiq_2f : use robotiq 2finger gripper

- **Example**

```
$ roslaunch dsr_description m0609.launch
$ roslaunch dsr_description m1013.launch color:=blue # Change Color
$ roslaunch dsr_description m1509.launch gripper:=robotiq_2f # insert robotiq
gripper
$ roslaunch dsr_description m0617.launch color:=blue gripper:=robotiq_2f
$ roslaunch dsr_description a0509.launch # A-Series

Robot and Joint_state_publisher are loaded on Rviz simulator (Figure 3.1).
The robot can be moved by using Joint_state_publisher.
```

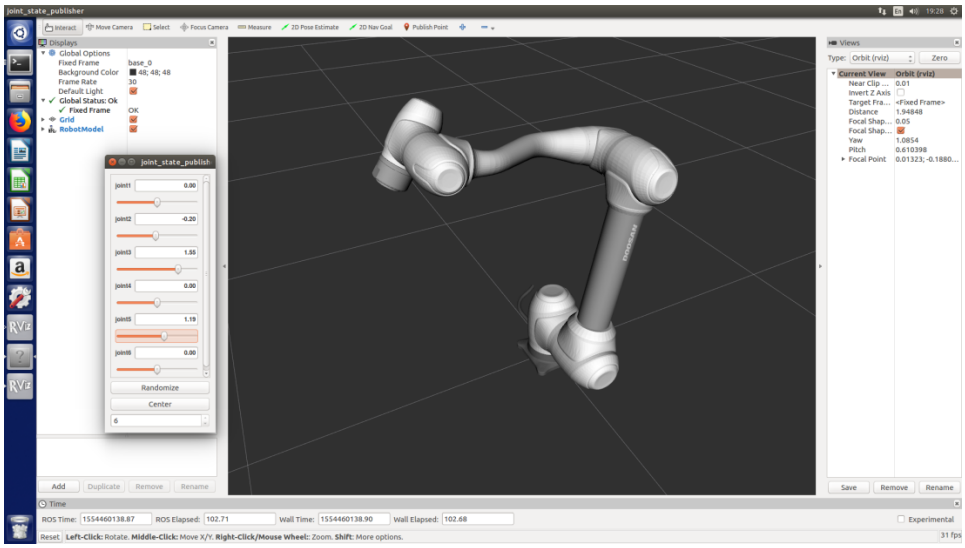


Figure 3.1 Robot on the Rviz simulator

4. dsr_moveit

4.1 dsr_moveit_config

▪ Feature

- Launch the robot model on Rviz simulator and operated by MoveIt.
- It only works in simulation mode.

▪ Parameter

Parameter Name	Data Type	Default Value	Description
color	-	white	Robot Color . white or blue

▪ Example

```
$ roslaunch moveit_config_m0609 m0609.launch
$ roslaunch moveit_config_m0617 m0617.launch
$ roslaunch moveit_config_m1013 m1013.launch color:=blue
$ roslaunch moveit_config_m1509 m1509.launch
$ roslaunch moveit_config_a0509 a0509.launch
```

Robot and Motion Planning Interface window are loaded on Rviz(Figure 4.1)
MotionPlanning allows the robot to run virtually.

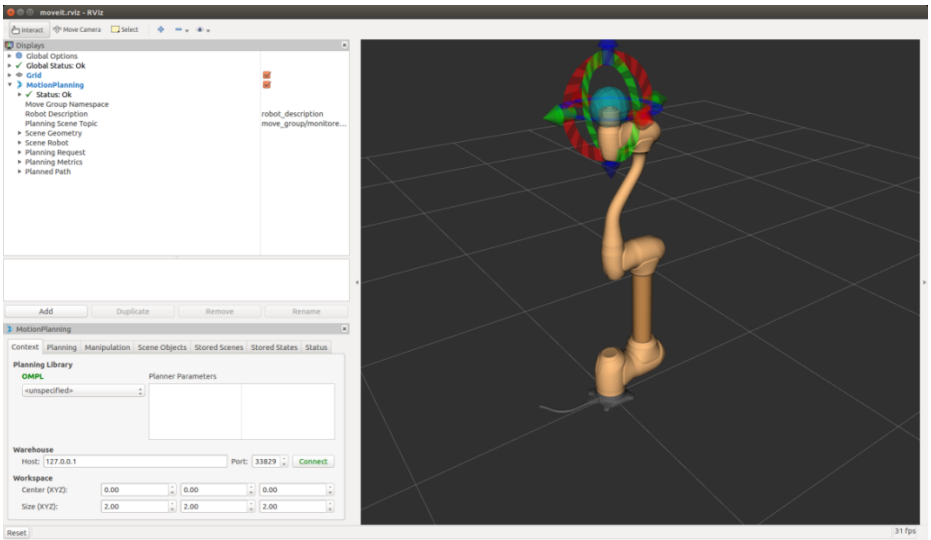


Figure 4.1 Rviz + MoveIt

4.2 dsr_control + moveit

- **Feature**
  - Launch the robot model on Rviz simulator and operated by MoveIt.
  - Working by connecting with emulator mode or real robot.
  - The emulator mode only works in virtual mode.

- **Parameter**

Paramater Name	Data Type	Default Value	Description
host	-	127.0.0.1	Robot Controller IP . Emulator : 127.0.0.1 . Real robot controller : 192.168.127.100
port	-	12345	port
mode	-	virtual	Robot operation mode - virtual : virtual mode - real : real mode
model	-	m1013	M-Series Robot model . m0609, m0617, m1013, m1509 A-Series Robot model . a0509
color	-	white	Robot color . white or blue
gripper	-	none	using gripper or not . none : not use gripper . robotiq_2f : use robotiq 2finger gripper

- **Example**

```
<virtual mode>
$ roslaunch dsr_control dsr_moveit.launch model:=m0609 mode:=virtual
$ roslaunch dsr_control dsr_moveit.launch model:=m0617
```

$ roslaunch dsr_control dsr_moveit.launch model:=m1013 **mode:=virtual** color:=blue

<real mode>

Robot controller IP defalut = 192.168.127.100, port = 12345

$ roslaunch dsr_control dsr_moveit.launch model:=m1509 host:=192.168.127.100 **mode:=real** color:=blue gripper:=robotiq_2f

$ roslaunch dsr_control dsr_moveit.launch model:=a0509 host:=192.168.127.10**0** **mode:=real**

The Robot and Motion Planning Interface window are loaded on Rviz(Figure 4.2). The MotionPlanning allows the robot to run in real environment.

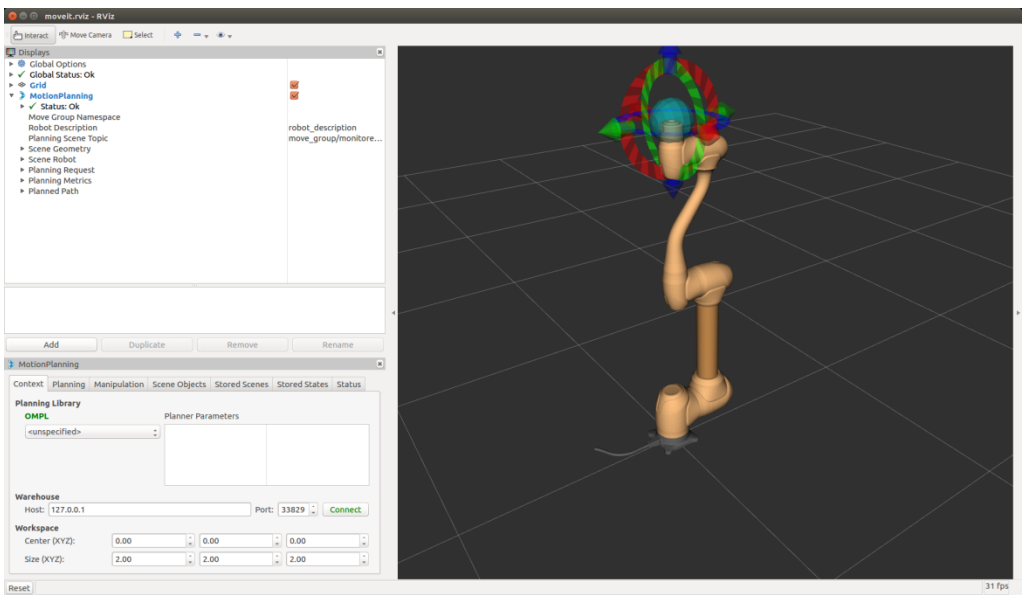


Figure 4.2 Rviz + dsr_control

4.3 MoveIt Commander

- **Feature**
  - Control the robot through Moveit Commander..

- **Install**
  - sudo apt-get insatll ros-kinetic-moveit-commander

- **Example**

<virtual mode>

$ roslaunch dsr_control dsr_moveit.launch model:=**m1013 mode:=virtual**

$ ROS_NAMESPACE=/dsr01**m1013** rosrun moveit_commander moveit_commander_cmdline.py robot_description:=/dsr01**m1013**/robot_description

> use arm
> goal0 = [0 0 0 0 0 0] # save the home position to variable "goal0"
> goal1 = [0 0 1.57 0 1.57 0] # save the target position to varialbe "goal1" / radian
> go goal1 # plan & excute (the robot is going to move target position)
> go goal0 # paln & excute (the robot is going to move home position)

<real mode>

Robot controller IP defalut = 192.168.127.100, port = 12345

$ roslaunch dsr_control dsr_moveit.launch model:=**a0509 host:=192.168.127.100 mode:=real**

$ ROS_NAMESPACE=/dsr01**a0509** rosrun moveit_commander moveit_commander_cmdline.py robot_description:=/dsr01**a0509**/robot_description

> use arm
> goal0 = [0 0 0 0 0 0] # save the home position to variable "goal0"
> goal1 = [0 0 1.57 0 1.57 0] # save the target position to varialbe "goal1" / radian

5. dsr_launcher

5.1 dsr_launcher

▪ Feature

- Configure various robot environment through dsr_launcher.
- The user can build Single Robot, Multi Robot, Gripper, mobile environment according to Paramater.
- Multi Robot configuration is an extension of Single Robot configuration. This is an example of 2 robots of Multi Robot configuration basically provided in this package. Please modify the dsr_launcher / multi-robot * .launch files to the appropriate configuration for your environment. (ns, host, port, model, etc. should be corrected correctly.)
- After loading dsr_launcher, run dsr_example for each environment with rosrun.

  (For details, see Chapter 6, dsr_example.)

▪ Paramater

Parameter Name	Datatype	Default Value	Description
ns	-	dsr01	ROBOT name space . single robot : dsr01 . multi robot: dsr01, dsr02, dsr03, dsr04 …
host	-	127.0.0.1	Robot controller IP . Emulator : 127.0.0.1 . Real robot controller : 192.168.127.100
port	-	12345	port
mode	-	virtual	Robot operation mode - virtual : virtual mode - real : real mode
model	-	m1013	M-Series Robot model . m0609, m0617, m1013, m1509 A-Series Robot model . a0509

Parameter Name	Datatype	Default Value	Description
color	-	white	Robot color . white or blue
gripper	-	none	using gripper or not . none : not use gripper . robotiq_2f : use robotiq 2finger gripper
mobile	-	none	using mobile robot or not . none : not use mobile robot . husky : use husky mobile robot

- **Examples**

<single robot>

- rviz, virtual mode, m1013(white)

$ roslaunch dsr_launcher single_robot_rviz.launch **mode:=virtual** model:=m1013

- gazebo, real mode, m1013(blue)

$ roslaunch dsr_launcher single_robot_gazebo.**launch host:=192.168.127.100 port:= 12345 mode:=real** color:=**bule**

- rviz+gazebo, real mode, m1013(white)

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch host:=192.168.127.100 port:= 12345 mode:=real

- + gripper

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch **gripper:=robotiq_2f**

- + mobile

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch gripper:=robotiq_2f **mobile:=husky**

<multi robot>

- rviz, virtual mode, m1013 x 2

$ roslaunch dsr_launcher multi_robot_rviz.launch

- gazebo, virtual mode, m1013 x 2

$ roslaunch dsr_launcher multi_robot_gazebo.launch

- rviz + gazebo, virtual mode, m1013 x 2

$ roslaunch dsr_launcher multi_robot_rviz_gazebo.launch

6. dsr_example

▪ Feature

- Provides an example of robot operation according to robot environment configured through dsr_launcher.

  (Please refer to Chapter 5, "dsr_launcher" for detailed robot environment configuration.)

- The example files was written by python.

  - Directory of .py files: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts

6.1 Single Robot

▪ Feature

- Provides an example of operating a Single Robot.

  (Please refer to Chapter 5, "dsr_launcher" for detailed robot environment configuration.)

- The example files was written in python

  - Directory of .py files: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts/simple

▪ Paramaters of dsr_launcher

Parameter Name	Datatype	Default Value	Description
ns	-	dsr01	ROBOT name space . single robot : dsr01 . multi robot: dsr01, dsr02, dsr03, dsr04 …
host	-	127.0.0.1	Robot controller IP . Emulator : 127.0.0.1 . Real robot controller : 192.168.127.100
port	-	12345	port
mode	-	virtual	Robot operation mode - virtual : virtual mode - real : real mode
model	-	m1013	M-Series Robot model . m0609, m0617, m1013, m1509 A-Series Robot model . a0509
color	-	white	Robot color . white or blue
gripper	-	none	using gripper or not . none : not use gripper . robotiq_2f : use robotiq 2finger gripper
mobile	-	none	using mobile robot or not

Parameter Name	Datatype	Default Value	Description
			. none : not use mobile robot . husky : use husky mobile robot

- **Example**

1. Robot controller default IP/Port

- IP : 192.168.127.100 , port = 12345

2. launch

Run dsr_launher for your desired configuration.

- single robot in rviz, virtual mode, m1013(white)

$ roslaunch dsr_launcher single_robot_rviz.launch mode:=virtual model:=m1013 color:=white

- single robot in gazebo, real mode, m1013(blue)

$ roslaunch dsr_launcher single_robot_gazebo.launch mode:=real host:=192.168.127.100 model:=m1013 color:=blue

- single robot in rviz + gazebo, virtual mode, m1013(white)

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013 color:=white

3. run application node

- Edit example files

. Open the example file you want to run and modify the ROBOT_ID and ROBOT_MODEL accordingly.

.. ex>

ROBOT_ID = "dsr01"

ROBOT_MODEL = "m1013"

$ rosrun dsr_example_py **single_robot_simple.py**

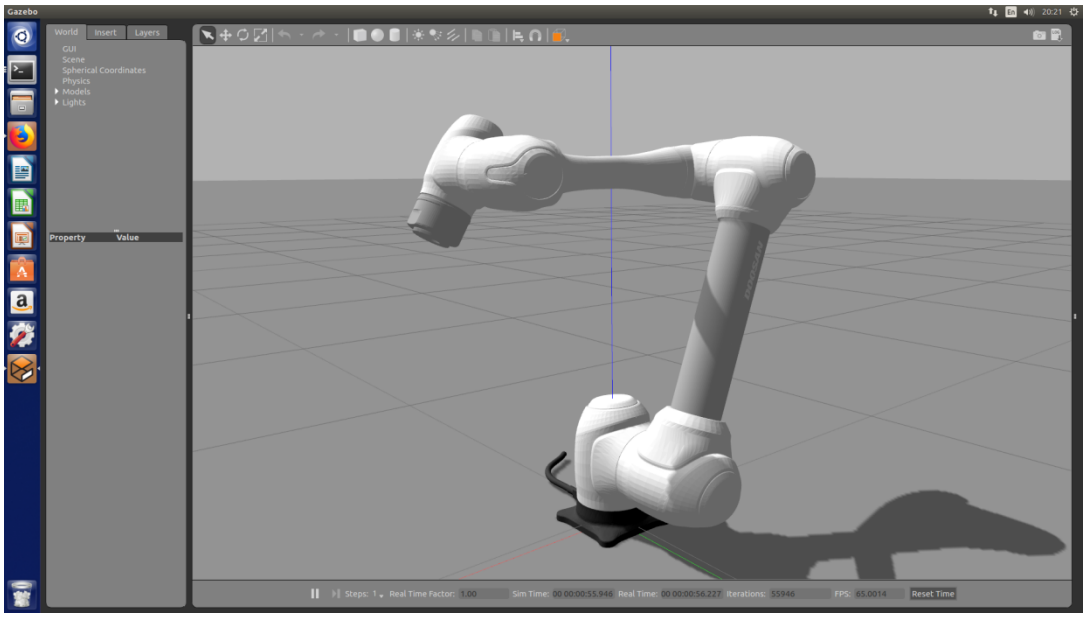


Figure 6.2 single robot

6.2 Multi Robot

▪ Feature

- Provides an example of driving Multi Robot.

  (Please refer to Chapter 5, "dsr_launcher" for detailed robot environment configuration.)

- The example files was written in python

  - Directory of .py files: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts/simple

▪ Paramaters of dsr_launcher

Parameter Name	Datatype	Default Value	Description
ns	-	dsr01	ROBOT name space . single robot : dsr01 . multi robot: dsr01, dsr02, dsr03, dsr04 …
host	-	127.0.0.1	Robot controller IP . Emulator : 127.0.0.1 . Real robot controller : 192.168.127.100
port	-	12345	port
mode	-	virtual	Robot operation mode - virtual : virtual mode - real : real mode
model	-	m1013	M-Series Robot model . m0609, m0617, m1013, m1509 A-Series Robot model . a0509
color	-	white	Robot color . white or blue
gripper	-	none	using gripper or not . none : not use gripper . robotiq_2f : use robotiq 2finger gripper
mobile	-	none	using mobile robot or not

Parameter Name	Datatype	Default Value	Description
			. none : not use mobile robot . husky : use husky mobile robot

- **Example**

1. Robot controller default IP/Port

- IP : 192.168.127.100 , port = 12345

- For multi robot, set the IP of each robot controller differently

1. launch

- edit launch file

. $ cd ~/catkin_ws/src/doosan-robot/dsr_launcher/launch

. Modify the multi_robot_*.launch file for each situation.

.. edit argument ns host port, model...

- multi robot in rviz

$ roslaunch **dsr_launcher multi_robot_rviz.launch** model:=m1013

- multi robot in gazebo

$ roslaunch **dsr_launcher multi_robot_gazebo.launch** color:=bule

- multi robot in rviz + gazebo

$ roslaunch **dsr_launcher multi_robot_rviz_gazebo.launch**

2. run application node

- Edit example files

. Open the example file you want to run and modify the robot_id and robot_model accordingly.

.. ex>

robot_id1 = "dsr01"; robot_model1 = "m1013"

robot_id2 = "dsr02"; robot_model2 = "m1013"

$ rosrun dsr_example_py **multi_robot_simple.py**

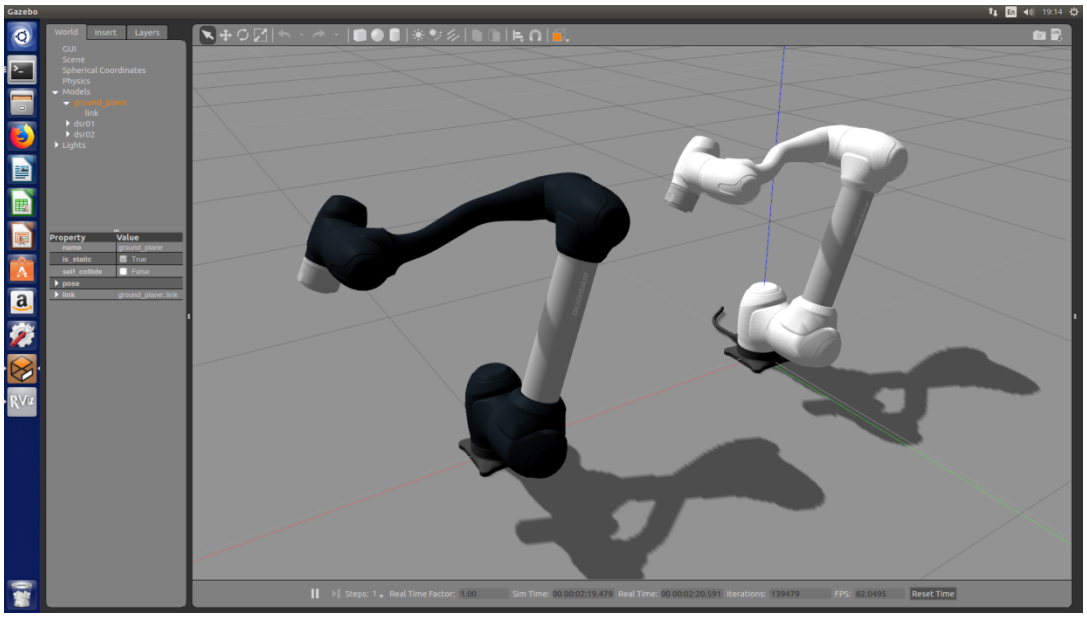


Figure 6.3 multi robot

6.3 Gripper

▪ Feature

- Provides an example of using gripper (robotiq_2f)

  (Please refer to Chapter 5, "dsr_launcher" for detailed robot environment configuration.)

- When running dsr_launcher, give argument (**gripper: = robotiq_2f)**.
- The example files was written in python

  - Directory of .py files: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts/gripper

▪ Praramaters of dsr_launcher

Parameter Name	Datatype	Default Value	Description
ns	-	dsr01	ROBOT name space . single robot : dsr01 . multi robot: dsr01, dsr02, dsr03, dsr04 …
host	-	127.0.0.1	Robot controller IP . Emulator : 127.0.0.1 . Real robot controller : 192.168.127.100
port	-	12345	port
mode	-	virtual	Robot operation mode - virtual : virtual mode - real : real mode
model	-	m1013	M-Series Robot model . m0609, m0617, m1013, m1509 A-Series Robot model . a0509
color	-	white	Robot color . white or blue
gripper	-	none	using gripper or not . none : not use gripper . robotiq_2f : use robotiq 2finger gripper

Parameter Name	Datatype	Default Value	Description
mobile	-	none	using mobile robot or not . none : not use mobile robot . husky : use husky mobile robot

- **Example**

1. Robot controller default IP/Port

- IP : 192.168.127.100 , port = 12345

2. launch : single robot + gripper

- single robot in rviz

$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 **gripper:=robotiq_2f**

- single robot in gazebo

$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 **gripper:=robotiq_2f**

- single robot in rviz + gazebo

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013 **gripper:=robotiq_2f**

3. run application node

- Edit example files

. Open the example file you want to run and modify the ROBOT_ID and ROBOT_MODEL accordingly.

.. ex>

ROBOT_ID = "dsr01"

ROBOT_MODEL = "m1013"

$ rosrun dsr_example_py **pick_and_place.py**

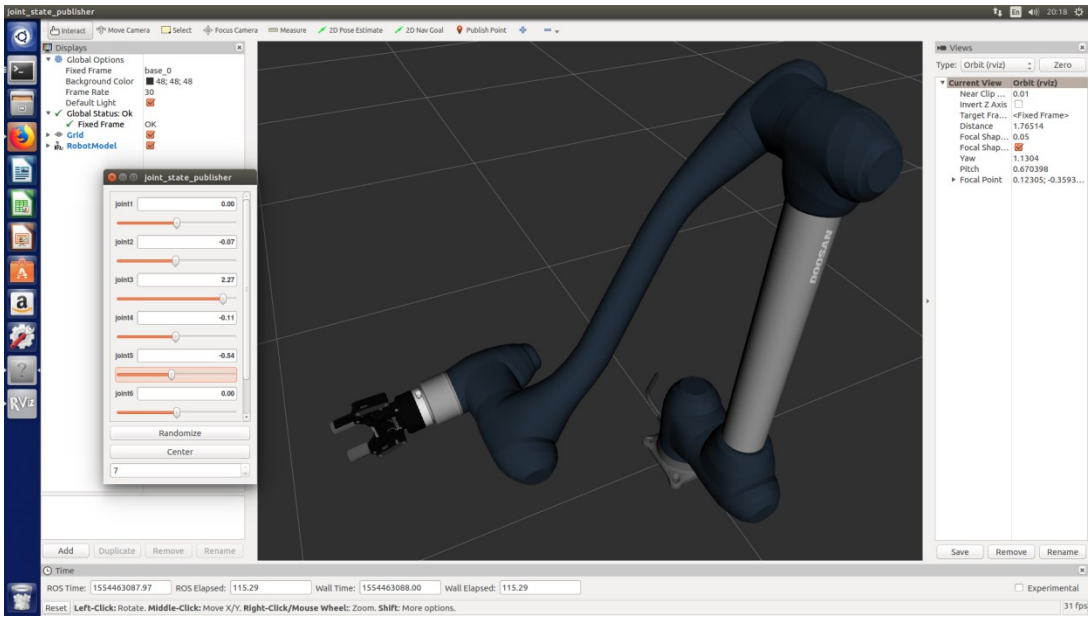


Figure 6.4 robot + gripper

6.4 Mobile robot

▪ Feature

- Provides mobile robot(huskey) examples.

  (Please refer to Chapter 5, "dsr_launcher" for detailed robot environment configuration.)

- When running dsr_launcher, give argument (**mobile:=husky**).
- The example files was written in python

  - Directory of .py files: ~/catkin_ws/src/doosan-robot/dsr_example/py/scripts/mobile

▪ Paramaters of dsr_launcher

Parameter Name	Datatype	Default Value	Description
ns	-	dsr01	ROBOT name space . single robot : dsr01 . multi robot: dsr01, dsr02, dsr03, dsr04 …
host	-	127.0.0.1	Robot controller IP . Emulator : 127.0.0.1 . Real robot controller : 192.168.127.100
port	-	12345	port
mode	-	virtual	Robot operation mode - virtual : virtual mode - real : real mode
model	-	m1013	M-Series Robot model . m0609, m0617, m1013, m1509 A-Series Robot model . a0509
color	-	white	Robot color . white or blue
gripper	-	none	using gripper or not . none : not use gripper . robotiq_2f : use robotiq 2finger gripper

Parameter Name	Datatype	Default Value	Description
mobile	-	none	using mobile robot or not . none : not use mobile robot . husky : use husky mobile robot

- **Example**

1. Robot controller default IP/Port

- IP : 192.168.127.100 , port = 12345

2. launch : single robot + mobile

- single robot in rviz

$ roslaunch dsr_launcher single_robot_rviz.launch model:=m1013 **mobile:=husky**

- single robot in gazebo

$ roslaunch dsr_launcher single_robot_gazebo.launch model:=m1013 **mobile:=husky**

- single robot in rviz + gazebo

$ roslaunch dsr_launcher single_robot_rviz_gazebo.launch model:=m1013 **mobile:=husky**

3. run application node

- Edit example files

. Open the example file you want to run and modify the ROBOT_ID and ROBOT_MODEL accordingly.

.. ex>

ROBOT_ID = "dsr01"

ROBOT_MODEL = "m1013"

$ rosrun dsr_example_py **single_robot_moblie.py**

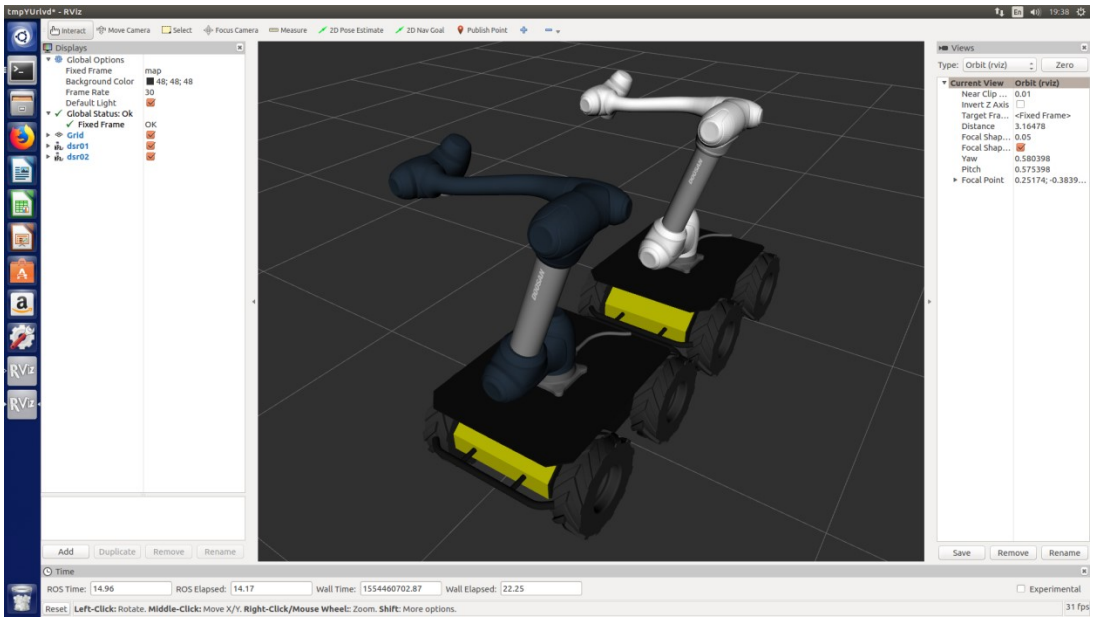


Figure 6.5 robot on mobile

7. dsr_msgs

7.1 Topic

7.1.1 RobotState.msg

- **Features**

  Topic message of robot state.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
robot_state	int32		Robot state : enum.ROBOT_STATE
robot_state_str	string		Robot state (string type)
current_posj	float64[6]		Current joint position of the robot
current_posx	float64[6]		Current task position of the robot
io_control_box	int32		Digital input status of robot controller
access_control	int32		Control permission status: enum.ACCESS.CONTROL
homming_completed	bool		Homming state of robot
tp_initialized	bool		TP initialization status
speed	int8		Current speed of robot
mastering_need	int8		Robot mastering required
drl_stopped	bool		DRL(Doosan Robot Languge) stop state
disconnected	bool		Communication connection status

enum.ROBOT_STATE

Num	name	Description
0	STATE_INITIALIZING	It is an initialization state for setting various parameters by automatically entering by TP application.
1	STATE_STANDBY	Operable Standby state.
2	STATE_MOVING	It is a command operation state that is automatically switched when the command is received after receiving the command in the command wait state. When the operation is completed, it is switched to the auto command waiting state.
3	STATE_SAFE_OFF	Robot stop mode due to function and operation error, Servo off state (motor and brake power are cut off after control stop)
4	STATE_TEACHING	Direct teaching state
5	STATE_SAFE_STOP	Robot stop mode due to function and operation error. Safe stop status (status in which only the control is stopped, program suspended status in the automatic mode)
6	STATE_EMERGENCY_STOP:	Emergency stop state
7	STATE_HOMMING	Homing mode state (state in which robot is aligned in hardware).
8	STATE_RECOVERY	When the robot is stopped due to an error such as exceeding the robot driving range, it is in the recovery mode state to move within the driving range.
9	eSTATE_SAFE_STOP2	Same as eSTATE_SAFE_STOP but a state that must be switched to recovery mode because the robot is out of range
10	STATE_SAFE_OFF2	Same as eSTATE_SAFE_OFF state, but must be in recovery mode due to exceeding the robot drive range
11	STATE_RESERVED1	reserved
12	STATE_RESERVED2	reserved

enum.ROBOT_STATE

Num	name	Description
0	MANAGE_ACCESS_CONTROL_FORCE_REQUEEST	Control forced release message transmission
1	MANAGE_ACCESS_CONTROL_REQUEST,	Send control request message
2	MANAGE_ACCESS_CONTROL_RESPONSE_YES	Send control authorization request acknowledge message
3	MANAGE_ACCESS_CONTROL_RESPONSE_NO	Send control permission request decline message

7.1.2 RobotStop.msg

- **Features**

Topic message of robot stop.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
stop_mode	int32		robot stop mode : Refer to enum.STOP _MODE.

enum.STOP_MODE

Num	name	Description
0	STOP_TYPE_QUICK_STO,	reserved
1	STOP_TYPE_QUICK	quick stop (motion trajectory maintenance)
2	STOP_TYPE_SLOW	slow stop (motion trajectory maintenance)
3	STOP_TYPE_HOLD	Emergency stop
	STOP_TYPE_EMERGENCY	Emergency stop

7.1.3 RobotError.msg

- **Features**

  Topic message of robot error.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
Level	int32	-	Log level : enum.LOG _LEVEL
Group	int32	-	Log group : enum.LOG _GROUP
Code	int32	-	error code
msg1	string	-	error msg 1
msg2	string	-	error msg 2
msg3	string	-	error msg 3

enum.LOG_LEVEL

Num	name	Description
0	LOG_LEVEL_RESERVED	reserved
1	LOG_LEVEL_SYSINFO	Informational messages about simple functions and operational errors
2	LOG_LEVEL_SYSWARN	Robot is stopped due to simple function and operation error.
3	LOG_LEVEL_SYSERROR	Robot is stopped due to safety issue or device error.

enum.LOG_GROUP

Num	name	Description
0	LOG_GROUP_RESERVED	reserved
1	LOG_GROUP_SYSTEMFMK	Robot Controller (framework)
2	eLOG_GROUP_MOTIONLIB,	Robot Controller (motion)

Num	name	Description
3	LOG_GROUP_SMARTTP	TP application (GUI)
4	LOG_GROUP_INVERTER	Inverter board
5	LOG_GROUP_SAFETYCONTROLLER	Safety board (Safety Controller)

7.1.4 LogAlarm.msg

- **Features**

  Topic message of log alarm

- **Parameters**

Parameter Name	Data Type	Default Value	Description
level	int32	-	Log level : enum.LOG _LEVEL
group	int32	-	Log group : enum.LOG _GROUP
index	int32	-	error code
param	string	-	messages[3]

enum.LOG_LEVEL

Num	name	Description
0	LOG_LEVEL_RESERVED	reserved
1	LOG_LEVEL_SYSINFO	Informational messages about simple functions and operational errors
2	LOG_LEVEL_SYSWARN	Robot is stopped due to simple function and operation error.
3	LOG_LEVEL_SYSERROR	Robot is stopped due to safety issue or device error.

enum.LOG_GROUP

Num	name	Description
0	LOG_GROUP_RESERVED	reserved
1	LOG_GROUP_SYSTEMFMK	Robot Controller (framework)
2	eLOG_GROUP_MOTIONLIB,	Robot Controller (motion)
3	LOG_GROUP_SMARTTP	TP application (GUI)
4	LOG_GROUP_INVERTER	Inverter board

Num	name	Description
5	LOG_GROUP_SAFETYCONTROLLER	Safety board (Safety Controller)

7.1.5 ModbusState.msg

- **Features**

  Topic message of Modbus State

- **Parameters**

Parameter Name	Data Type	Default Value	Description
level	string	-	Modbus Signal Name.
group	int32	-	Modbus Register Value (Unsigned : 0 ~ 65535)

7.1.6 JogMultiAxis.msg

Features

Topic message for multi-axis jog control

Multi-axis jog speed = (250mm/s)/√3 x [Unit vector] x speed[%]

⚠ **Caution**

This message is available with robot controller software version 2.50 or later.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
jog_axis	float64[6]	-	Unit vector orientation of task space [Tx, Ty, Tz, Rx, Ry, Rz] (-1.0 ~ 1.0)
move_reference	int8	-	0: MOVE_REFERENCE_BASE 1: MOVE_REFERENCE_TOOL: 2: MOVE_REFERENCE_WORLD
speed	float64		jog speed [%] (1~100)

7.2 Service/motion

7.2.1 MoveJoint.srv

- **Features**

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

- **Parameters**

Parameter Name	Data Type	Default Value	Description
pos	float64[6]	-	joint angle list
vel	float64	-	velocity
acc	float64	-	acceleration
time	float64	0.0	Reach time [sec]
radius	float64	0.0	Radius for blending
mode	int8	0	Movement basis • MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
blendType	int8	0	Reactive motion mode • BLENDING_SPEED_TYPE_DUPLICATE =0 • BLENDING_SPEED_TYPE_OVERRIDE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

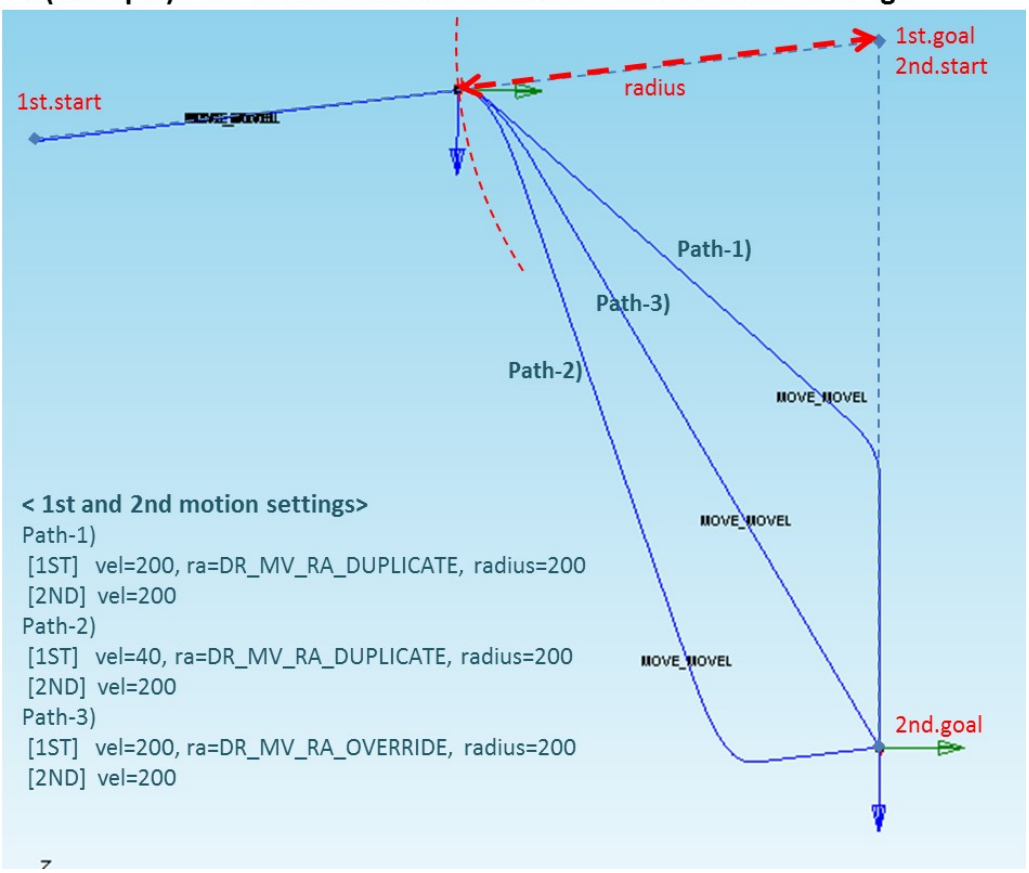
Note

· If the time is specified, values are processed based on time, ignoring vel and acc.

⚠ Caution

If the following motion is blended with the conditions of blendType =BLENDING_SPEED_TYPE_BUPLICATE 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>



Return

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.2 MoveLine.srv

- **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	float64[6]	-	position list
vel	float64[2]	-	linear velocity, angular velocity
acc	float64[2]	-	linear acceleration, angular acceleration
time	float64	0.0	Reach time [sec] * If the time is specified, values are processed based on time, ignoring vel and acc.
radius	float64	0.0	Radius for blending
ref	int8	0	reference coordinate • MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1 • MOVE_REFERENCE_WORLD=2
mode	int8	0	Movement basis • MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
blendType	int8	0	Reactive motion mode • BLENDING_SPEED_TYPE_DUPLICATE =0

Parameter Name	Data Type	Default Value	Description
			• BLENDING_SPEED_TYPE_OVERRIDE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

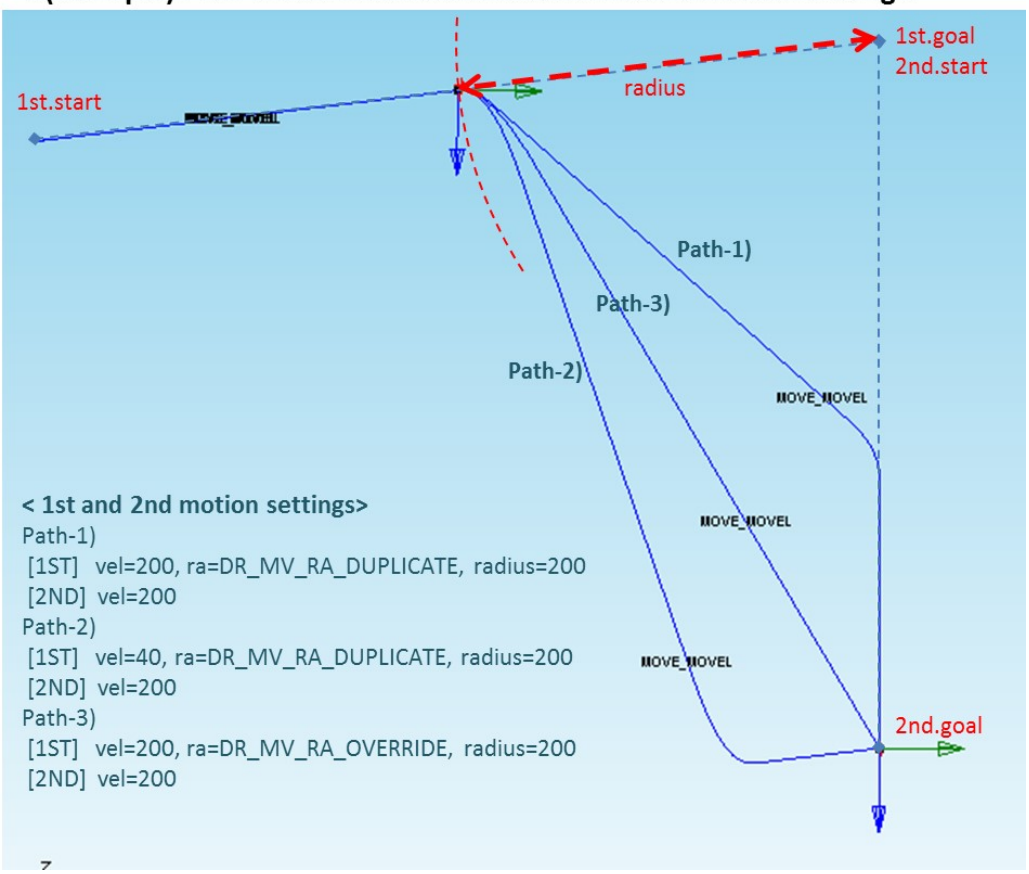
Note

- 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
- The ref argument DR_WORLD is only available in M2.40 or later versions.

Caution

If the following motion is blended with the conditions of blendType = BLENDING_SPEED_TYPE_DUPLICATEE 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>



- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.3 MoveJointx.srv

- **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).

- **Parameters**

Parameter Name	Data Type	Default Value	Description
pos	float64[6]	-	position list
vel	float64	-	velocity
acc	float64	-	acceleration
time	float64	0.0	Reach time [sec]
radius	float64	0.0	Radius for blending
ref	int8	0	reference coordinate • MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1 • MOVE_REFERENCE_WORLD=2
mode	int8	0	Movement basis • MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
blendType	int8	0	Reactive motion mode • BLENDING_SPEED_TYPE_DUPLICATE =0 • BLENDING_SPEED_TYPE_OVERRIDE =1
sol	int8	0	Solution space
syncType	int8	0	• SYNC = 0 • ASYNC = 1

Note

- If the time is specified, values are processed based on time, ignoring vel and acc.
- Using the blending in the preceding motion generates an error in the case of input with relative motion (eMoveMode = MOVE_MODE_ RELATIVE), and it is recommended to blend using MoveJoint or MoveLine
- The ref argument DR_WORLD is only available in M2.40 or later versions.
- Refer to the description of MoveJoint.srv and MoveLine.srv for blending according to option ra and vel/acc.

▪ 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

▪ Return

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.4 MoveCircle.srv

- **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	std_msgs/Float64MultiArray[]	-	target[2][6] • position list
vel	float64[2]	-	linear velocity, angular velocity
acc	float64[2]	-	linear acceleration, angular acceleration
time	float64	0.0	Reach time [sec]
radius	float64	0.0	Radius for blending
ref	int8	0	reference coordinate • MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1 • MOVE_REFERENCE_WORLD=2
mode	int8	0	Movement basis • MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
angle1	float64	0.0	angle1
angle2	float64	0.0	angle2
blendType	int8	0	Reactive motion mode • BLENDING_SPEED_TYPE_DUPLICATE =0 • BLENDING_SPEED_TYPE_OVERRIDE =1
syncType	int8	0	• SYNC = 0

Parameter Name	Data Type	Default Value	Description
			• ASYNC = 1

Note

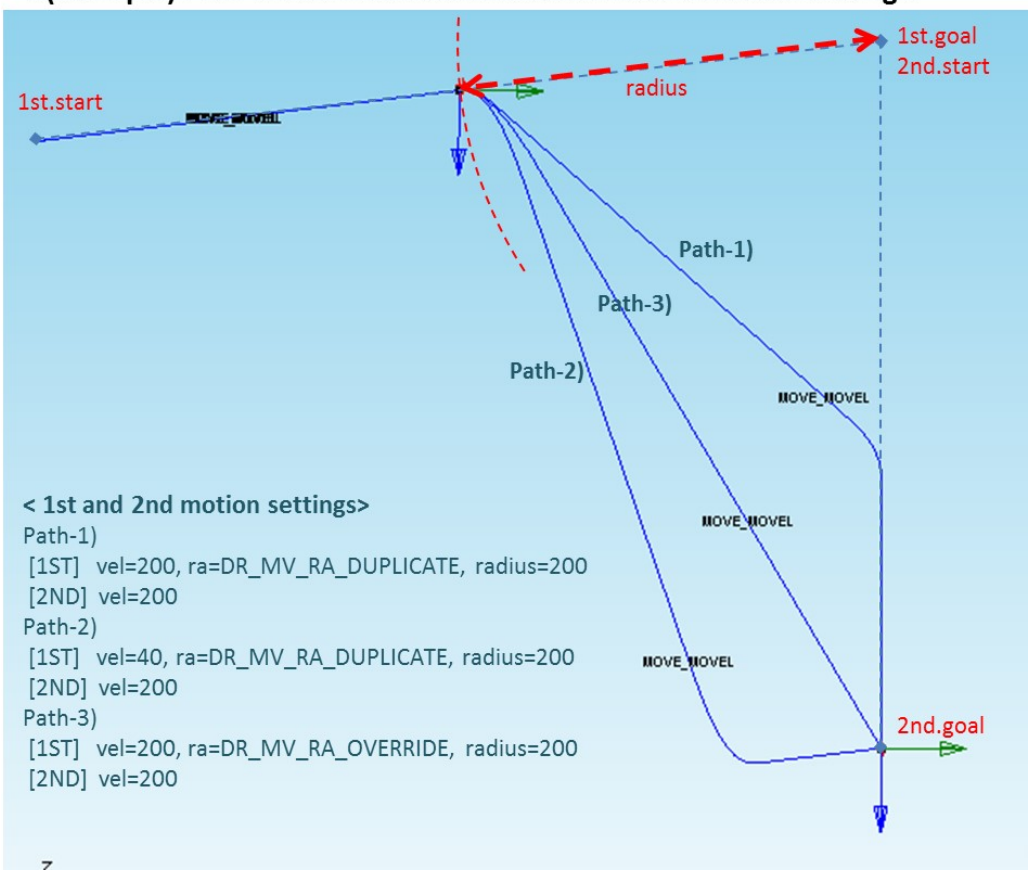
- If an argument is inputted to vel (e.g., vel=[30, 0]), the input argument corresponds to the linear velocity of the motion while the angular velocity is determined proportionally to the linear velocity.
- If an argument is inputted to acc (e.g., acc=[60, 0]), the input argument corresponds to the linear 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.
- The ref argument DR_WORLD is only available in M2.40 or later versions.
- If the mod is MOVE_MODE_RELATIVE L, pos[0] and pos[1] are defined in the relative coordinate system of the previous pos. (pos[0] is the relative coordinate from the starting point while pos[1] is the relative coordinate from pos[0].)
- If only one angle is inputted, the total rotated angle on the circular path is applied to the angle.
- 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.

Caution

If the following motion is blended with the conditions of blendType=

BLENDING_SPEED_TYPE_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>



- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.5 MoveSplineJoint.srv

- **Features**

The robot moves along a spline curve path that connects the current position to the target position (the last waypoint in position list) via the waypoints of the joint space input in position 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	std_msgs/Float64MultiArray[]	-	target pos [100][6] max = 100
posCnt	int8	-	Count of target pos
vel	float64	-	velocity
acc	float64	-	acceleration
time	float64	0.0	Reach time [sec]
mode	int8	0	Movement basis • MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

Note

- If the time is specified, values are processed based on time, ignoring vel and acc.
- If the mod is MOVE_MODE_RELATIVE, 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 service does not support online blending of previous and subsequent motions.

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.6 MoveSplineTask.srv

- **Features**

The robot moves along a spline curve path that connects the current position to the target position (the last waypoint in position 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	std_msgs/Float64MultiArray[]	-	target pos [100][6] max = 100
posCnt	int8	-	Count of target pos
vel	float64[2]	-	linear velocity, angular velocity
acc	float64[2]	-	linear acceleration, angular acceleration
time	float64	0.0	Reach time [sec]
ref	int8	0	reference coordinate • MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1 • MOVE_REFERENCE_WORLD=2
mode	int8	0	Movement basis • MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
opt	int8	0	Velocity option • SPLINE_VELOCITY_OPTION_DEFAULT=0 • SPLINE_VELOCITY_OPTION_CONST=1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

Note

- If an argument is inputted to vel (e.g., vel=[30, 0]), 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, 0]), 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.
- The ref argument DR_WORLD is only available in M2.40 or later versions.
- If the mod is MOVE_MODE_RELATIVE, each pos in the pos_list is defined in the relative coordinate of the previous pos. (If positiolist=[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 service 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 "opt= SPLINE_VELOCITY_OPTION_CONST" option (constant velocity option) is selected, and the motion is automatically switched to the variable velocity motion (opt= SPLINE_VELOCITY_OPTION_DEFAULT) in that case.

▪ Return

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.7 MoveBlending.srv

- **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	자료형	기본값	설명
pos	std_msgs/Float64MultiArray[]	-	posb list (pos1[6]:pos2[6]:type[1]:radius[1]) x 50(max)
posCnt	int8		Count of target pos
vel	float64[2]	-	linear velocity, angular velocity
acc	float64[2]	-	linear acceleration, angular acceleration
time	float64	0.0	Reach time [sec]
ref	int8	0	Reference coordinate • MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1 • MOVE_REFERENCE_WORLD=2
mode	int8	0	Movement basis • MOVE_MODE_ABSOLUTE =0 • MOVE_MODE_RELATIVE =1
syncType	int8	0	• SYNC = 0 • ASYNC = 1

Note

- If an argument is inputted to vel (e.g., vel=[30, 0]), 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, 0]), 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.
- The ref argument DR_WORLD is only available in M2.40 or later versions.
- If the mod is MOVE_MODE_RELATIVE, each pos in the posb_list is defined in the relative coordinate of the previous pos.

⚠ 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 service does not support online blending of previous and subsequent motions

▪ Return

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.8 MoveSpiral.srv

- **Features**

The radius increases in a radial direction and the robot moves in parallel with the rotating spiral motion in an axial direction. It moves the robot along the spiral trajectory on the surface that is perpendicular to the axis on the coordinate specified as ref and the linear trajectory in the axis direction.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
revolution	float64	-	Total number of revolutions [revolution]
maxRadius	float64		Final spiral radius [mm]
maxLength	float64		Distance moved in the axis direction [mm]
vel	float64[2]	-	linear velocity, angular velocity
acc	float64[2]	-	linear acceleration, angular acceleration
time	float64	0.0	Total execution time [sec]
taskAxis	int8	0	axis • TASK_AXIS_X = 0 • TASK_AXIS_Y = 1 • TASK_AXIS_Z = 2
ref	int8	0	reference coordinate • MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1 • MOVE_REFERENCE_WORLD=2
syncType	int8	0	• SYNC = 0 • ASYNC = 1

Note

- Revolution refers to the maximum radius 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
- If the time is specified, values are processed based on time, ignoring vel and acc.
- The ref argument DR_WORLD is only available in M2.40 or later versions.
- 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.
- This service does not support online blending of previous and subsequent motions.

⚠ 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 time value.
-

▪ Return

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.9 MovePeriodic.srv

- **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	Description
amp	float64[6]	-	Amplitude (motion between -amp and +amp) [mm] or [deg]
periodic	float64[6]	-	Period (time for 1 cycle) [sec]
acc	float64	-	Acceleration
time	float64	-	Acc-, dec- time [sec]
repeat	int8	-	Repetition count
ref	int8	0	reference coordinate • MOVE_REFERENCE_BASE =0 • MOVE_REFERENCE_TOOL=1 • MOVE_REFERENCE_WORLD=2
syncType	int8	0	• SYNC = 0 • ASYNC = 1

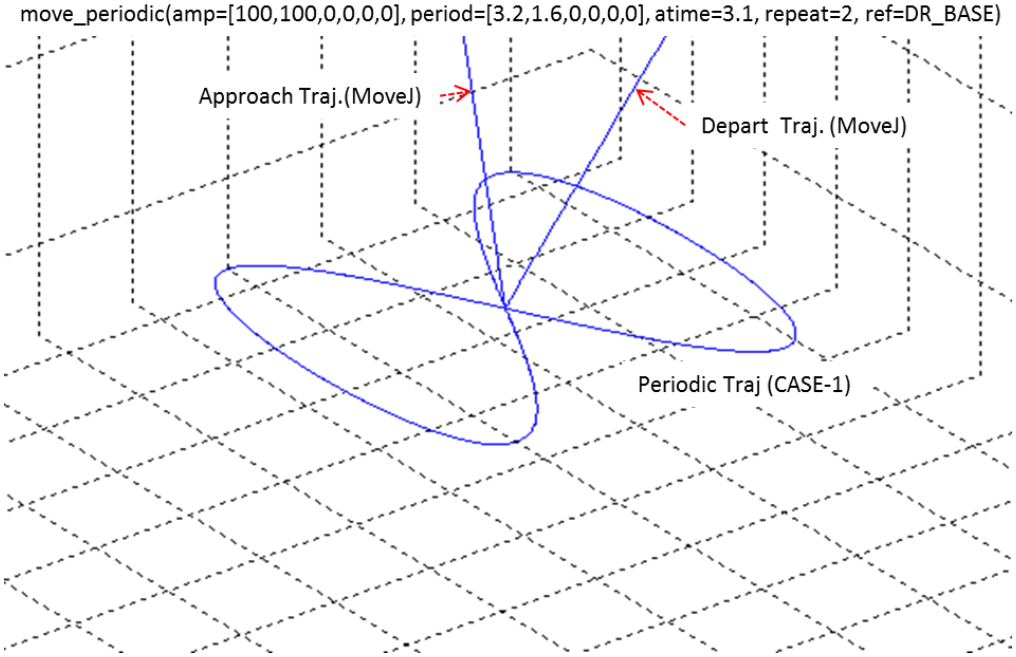
Note

- 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

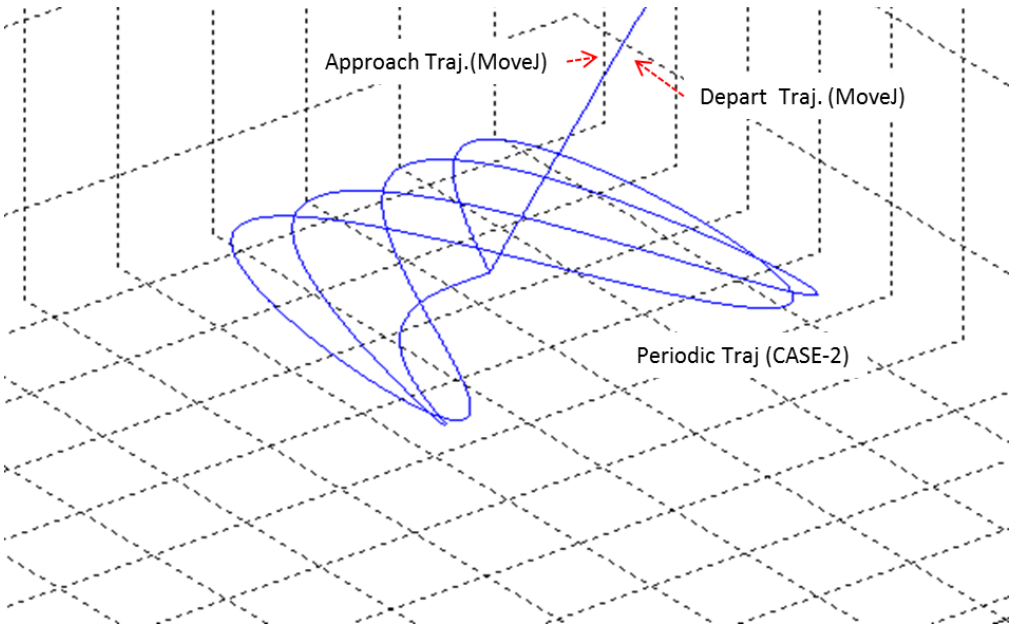
CASE-1) All-axis motions end at the same time

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)



CASE-2) Diff-axis motions end individually

move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.5,0,0,0,0], atime=0, repeat=2, ref=DR_BASE)



- ref refers to the reference coordinate system of the repeated motion.
- The ref argument DR_WORLD is only available in M2.40 or later versions.
- 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

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.10 MoveWait.srv

- **Features**

  This service sets the waiting time between the previous motion command and the motion command in the next line.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
N/A	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.11 Jog.srv

- **Features**

This service is for performing jog motion control for each axis of the robot.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
Jog_axis	int8	-	0 ~ 5 : JOINT 1 ~ 6 6 ~ 11: TASK 1 ~ 6 (X,Y,Z,rx,ry,rz)
move_reference	int8	-	0 : MOVE_REFERENCE_BASE 1 : MOVE_REFERENCE_TOOL
speed	float64	-	jog speed [%] : + forward , 0=stop, - backward

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.12 multiJog.srv

- **Features**

This is a service to perform jog control on multiple axes of robots in the robot controller.

Multi-axis jog speed = (250mm / s) / √3 x [unit vector] x speed [%]

⚠ **Caution**

This service is available with robot controller software version 2.50 or later.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
jog_axis	float64[6]	-	Unit vector direction of task space [Tx, Ty, Tz, Rx, Ry, Rz] (-1.0 ~ 1.0)
move_reference	int8	-	0: MOVE_REFERENCE_BASE 1: MOVE_REFERENCE_TOOL: 2: MOVE_REFERENCE_WORLD
speed	float64		jog speed [%] (1~100)

- **리턴**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.13 MovePause.srv

- **Features**

  It is a service to decelerate and pause the motion of the current robot.

  If no robot motion is in progress, it is ignored.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
N/A	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.14 MoveResume.srv

- **Features**

Service to resume the motion of a suspended robot.

If no robot path motion is in progress, it is ignored.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
N/A	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.15 MoveStop.srv

- **Features**

Service to stop robot motion.

It stops differently depending on the input parameters.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
Stop_mode	int32	-	0 : reserved 1 : quick stop (keep motion trajectory) 2 : slow stop (keep motion trajectory)

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.2.16 Trans.srv

▪ Features

- Input parameter(pos) based on the ref coordinate is translated/rotated as delta based on the same coordinate and this function returns the result that is converted to the value based on the ref_out coordinate.
- 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

Note

- The ref argument DR_WORLD is only available in M2.40 or later versions.
- The ref_out argument is only available in M2.40 or later versions.

▪ Return

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

7.3 Service/system

7.3.1 GetRobotMode.srv

- **Features**

  It is a service input for checking the current operation mode of the robot controller. The auto mode is a mode for automatically performing a series of operations (programs), and the manual mode is for performing a single operation such as jogging.

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
Robot_mode	int8	-	refer to enum.ROBOT_MODE.

- **enum.ROBOT_MODE**

Num	name	Description
0	ROBOT_MODE_MANUAL	Manual mode
1	ROBOT_MODE_AUTONOMOUS	Auto mode
2	ROBOT_MODE_MEASURE	Measure mode (Not currently supported)

7.3.2 SetRobotMode.srv

- **Features**

  This service is for setting the current operation mode of the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
robot_mode	int8	-	refer to enum.ROBOT_MODE

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

- **enum.ROBOT_MODE**

Num	name	Description
0	ROBOT_MODE_MANUAL	Manual mode
1	ROBOT_MODE_AUTONOMOUS	Auto mode
2	ROBOT_MODE_MEASURE	Measure mode (Not currently supported)

7.3.3 GetRobotSystem.srv

- **Features**

  It is a service input for confirming the current operation mode (virtual robot, actual robot) of the robot controller.

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
robot_system	int8	-	refer to enum.ROBOT_SYSTEM

- **enum.ROBOT_SYSTEM**

Num	name	Description
0	ROBOT_SYSTEM_REAL	Actual robot system
1	ROBOT_SYSTEM_VIRTUAL	virtual robot system

7.3.4 SetRobotSystem.srv

- **Features**

  This is a service for setting up the current robot system of the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
robot_system	int8	-	refer to enum.ROBOT_SYSTEM

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

- **enum.ROBOT_SYSTEM**

Num	name	Description
0	ROBOT_SYSTEM_REAL	Actual robot system
1	ROBOT_SYSTEM_VIRTUAL	virtual robot system

7.3.5 GetRobotSpeedMode.srv

- **Features**

  This service is used to check the current speed mode (normal mode, deceleration mode) from the robot controller.

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
Speed_mode	int8	-	refer to enum.SPEED_MODE

- **enum.SPEED_MODE**

Num	name	Description
0	SPEED_NORMAL_MODE	Normal speed mode
1	SPEED_REDUCED_MODE	deceleration speed mode

7.3.6 SetRobotSpeedMode.srv

- **Features**

  This service is used to set and change the currently operating speed mode of the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
speed_mode	int8	-	refer to enum.SPEED_MODE

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

- **enum.SPEED_MODE**

Num	name	Description
0	SPEED_NORMAL_MODE	Normal speed mode
1	SPEED_REDUCED_MODE	deceleration speed mode

7.3.7 SetSafeStopResetType.srv

- **Features**

This service is used to define a series of actions to be executed automatically after the state transition using the SetRobotMode service when the operation status information of the robot controller is SAFE_STOP.

If the robot operation mode is automatic, you can define and set whether to re-execute the program. In manual mode, this setting is ignored.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
reset_type	int8	-	refer to enum.SAFE_STOP_RESET_TYPE

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

- **enum.SAFE_STOP_RESET_TYPE**

Num	name	Description
0	SAFE_STOP_RESET_TYPE_DEFAULT	Simple state release (manual mode)
	SAFE_STOP_RESET_TYPE_PROGRAM_STOP	Stop program (autoc mode)
1	SAFE_STOP_RESET_TYPE_PROGRAM_RESUME	Restart the program (automatic mode)

7.3.8 GetCurrentPose.srv

- **Features**

This service is used to check the current position information of each axis of the robot according to the coordinate system (joint space or task space) in the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
space_type	int8	-	refer to enum.ROBOT_SPACE

- **Return**

Return Name	Data Type	Default Value	Description
pos	float64[6]	-	Robot position information

- **enum.ROBOT_SPACE**

Num	name	Description
0	ROBOT_SPACE_JOINT	Joint space
1	ROBOT_SPACE_TASK	task space

7.3.9 GetCurrentSiolutionSpace.srv

- **Features**

  It is a service to get the the current solution space value..

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
solution_space	int8	-	solution space (0~7)

- **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

7.3.10 GetJointTorque.srv

- **Features**

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

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
joint_torque	float64[6]	-	JTS torque value

7.3.11 GetExternalTorque.srv

- **Features**

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

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
ext_torque	float64[6]	-	Torque value generated by an external force

7.3.12 GetToolForce.srv

- **Features**

  This service returns the external force applied to the current tool based on the ref coordinate.

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
tool_force	float64[6]	-	External force applied to the tool

7.3.13 GetLastAlarm.srv

- **Features**

  This service is used to check the most recent log and alarm codes generated by the robot controller.

- **Parameters**

  None

- **Return**

Return Name	Data Type	Default Value	Description
log_alarm	LogAlarm.msg	-	refet to LogAlam.msg

- **LogAlam.msg**

Parameter Name	Data Type	Default Value	Description
level	int32	-	refet to enum.LOG _LEVEL
group	int32	-	refet to enum.LOG _GROUP
index	int32	-	error code
param	string[3]	-	param[3]

enum.LOG_LEVEL

Num	name	Description
0	LOG_LEVEL_RESERVED	reserved
1	LOG_LEVEL_SYSINFO	Informational messages about basic functions and operational errors
2	LOG_LEVEL_SYSWARN	Robot is stopped due to basic function and operation error
3	LOG_LEVEL_SYSERROR	Robot is stopped due to safety issue or device error

enum.LOG_GROUP

Num	name	Description
0	LOG_GROUP_RESERVED	reserved
1	LOG_GROUP_SYSTEMFMK	framework
2	eLOG_GROUP_MOTIONLIB,	Motion algorithm
3	LOG_GROUP_SMARTTP	TP program (GUI)
4	LOG_GROUP_INVERTER	Robot Inverter Board
5	LOG_GROUP_SAFETYCONTROLLER	Safety Controller

The log and alarm messages are passed through the predefined contents through the number, and the relevant parameters are sent together if necessary.
Please refer to log and alarm definition section for details.

7.4 Service/tcp

7.4.1 ConfigCreateTcp.srv

- **Features**

This service calls creating the name of the TCP information.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the TCP
pos	float64[6]	-	TCP infomation

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.4.2 ConfigDeleteTcp.srv

- **Features**

This service calls deleting the registered TCP information.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the TCP

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.4.3 GetCurrentTcp.srv

- **Features**

It is a service that fetches the currently set TCP information from the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
None	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
info	string	-	Name of the TCP

7.4.4 SetCurrentTcp.srv

- **Features**

It is a service to set the information about the currently set the TCP information

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the TCP

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.5 Service/tool

7.5.1 ConfigCreateTool.srv

- **Features**

It is a service to register tool information to be mounted on the robot end.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the TCP
weight	float		Weight of tool
cog	float64[3]		center of mass
inertia	float64[6]		inertia information

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.5.2 ConfigDeleteTool.srv

- **Features**

This service calls deleting the registered TOOL information.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the TCP

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.5.3 GetCurrentTool.srv

- **Features**

It is a service that fetches the currently set TOOL information from the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
None	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
info	string	-	Name of the TOOL

7.5.4 SetCurrentTool.srv

- **Features**

It is a service to set the information about the currently set the TOOL information

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Name of the TOOL

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.6 Service/io

7.6.1 SetCtlBoxDigitalOutput.srv

- **Features**

This service sends a signal at the digital contact point of the controller..

- **Parameters**

Parameter Name	Data Type	Default Value	Description
index	int8	-	I/O contact number mounted on the controller • Val argument existing: A number between 1 and 16
value	int8		I/O value • ON: 1 • OFF: 0

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.6.2 GetCtlBoxDigitalInput.srv

- **Features**

This function reads the signals from digital contact points of the controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
index	int8	-	A number 1 - 16 which means the contact number of I/O mounted on the controller.

- **Return**

Return Name	Data Type	Default Value	Description
value	bool	-	OFF =0, ON =1

7.6.3 SetToolDigitalOutput.srv

- **Features**

This function sends the signal of the robot tool from the digital contact point.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
index	int8	-	I/O contact number mounted on the robot arm • Val argument existing: A number between 1 and 6
value	int8		I/O value • ON: 1 • OFF: 0

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.6.4 GetToolDigitalIntput.srv

- **Features**

This function reads the signal of the robot tool from the digital contact point.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
index	int8	-	I/O contact number (1-6) mounted on the robot tool

- **Return**

Return Name	Data Type	Default Value	Description
value	bool	-	OFF =0, ON =1

7.6.5 SetCtlBoxAnalogOutputType.srv

- **Features**

This function sets the channel mode of the controller analog output..

- **Parameters**

Parameter Name	Data Type	Default Value	Description
channel	int8	-	• 1 : channel 1 • 2 : channel 2
mode	int8	-	analog io mode • current =0 • voltage =1

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.6.6 SetCtlBoxAnalogInputType.srv

- **Features**

This function sets the channel mode of the controller analog input.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
channel	int8	-	• 1 : channel 1 • 2 : channel 2
mode	int8	-	analog io mode • current =0 • voltage =1

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.6.7 SetCtlBoxAnalogOutput.srv

- **Features**

This function outputs the channel value corresponding to the controller analog output.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
channel	int8	-	• 1 : channel 1 • 2 : channel 2
value	float64	-	analog output value • Current mode: 4.0~20.0 [mA] • Voltage mode: 0~10.0 [V]

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.6.8 GetCtlBoxAnalogInput.srv

- **Features**

This function reads the channel value corresponding to the controller analog input.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
channel	int8	-	• 1 : channel 1 • 2 : channel 2

- **Return**

Return Name	Data Type	Default Value	Description
value	float	-	The analog input value of the specified channel • Current mode: 4.0~20.0 [mA] • Voltage mode: 0~10.0 [V]

7.7 Service/modbus

7.7.1 ConfigCreateModbus.srv

- **Features**

This function registers the 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	-	Modbus signal name
ip	string	-	IP address of the Modbus module
port	int8	-	Port number of the Modbus module
reg_type	int8	-	Modbus signal type • MODBUS_REGISTER_TYPE_DISCRETE_INPUTS • MODBUS_REGISTER_TYPE_COILS • MODBUS_REGISTER_TYPE_INPUT_REGISTER • MODBUS_REGISTER_TYPE_HOLDING_REGISTER
index	int8	-	Modbus signal의 index
value	int8	-	Output when the type is MODBUS_REGISTER_TYPE_COILS or MODBUS_REGISTER_TYPE_HOLDING_REGISTER (ignored otherwise)

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.7.2 ConfigDeleteModbus.srv

- **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**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.7.3 SetModbusOutput.srv

- **Features**

  This service sends the signal to an external Modbus system.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Modbus name
value	int32	-	Modbus digital I/O • ON : 1 • OFF : 0
			Value for Modbus analog I/O

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.7.4 GetModbuInput.srv

- **Features**

  This function reads the signal from the Modbus system.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
name	string	-	Modbus name

- **Return**

Return Name	Data Type	Default Value	Description
value	int32	-	ON or Off in the case of the Modbus digital I/O
			The register value in the case of the Modbus analog module

7.8 Service/drl

7.8.1 DrlStart.srv

- **Features**

This service is used to execute DRL script. (DRL is a Doosan robot language).

- **Parameters**

Parameter Name	Data Type	Default Value	Description
RobotSystem	int8	-	
Code	string	-	DRL code string

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

Note

- Robot operation status should be in STANDBY state (STATE_STANDBY) and should be used when robot mode is in auto mode.
- DRL programming should be done by referring to the Programming Manual.
-

7.8.2 DrlStop.srv

- **Features**

This service is used to stop the DRL program (task) currently running on the robot controller. Stops differently according to the eStopType received as an argument, and stops the motion of the current section

- **Parameters**

Parameter Name	Data Type	Default Value	Description
stop_mode	int8	-	drl stop mode • STOP_TYPE_QUICK_STO = 0 • STOP_TYPE_QUICK = 1 • STOP_TYPE_SLOW = 2 • STOP_TYPE_HOLD = STOP_TYPE_EMERGENCY = 3

- **Return**

Return Name	Data Type	Default Value	Description
Success	bool	-	True or False

7.8.3 DrlPause.srv

- **Features**

  This is a service to temporarily stop the DRL program (task) currently running on the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
None	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.8.4 DrlResume.srv

- **Features**

  This is a service to resume the currently paused DRL program (task) from the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
None	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.8.5 GetDrlState.srv

- **Features**

  This service reads the status of the DRL program

- **Parameters**

Parameter Name	Data Type	Default Value	Description
None	-	-	-

- **Return**

Return Name	Data Type	Default Value	Description
state	int8	-	state of DRL program . 0 : DRL_PROGRAM_STATE_PLAY . 1 : DRL_PROGRAM_STATE_STOP . 2 : DRL_PROGRAM_STATE_HOLD

7.9 Service/gripper

7.9.1 SerialSendData.srv

- **Features**

It controls the gripper through actual serial communication.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
data	string	-	String to send

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

7.9.2 RobotiqMove.srv

- **Features**

It is a service to control robotiq's gripper in simulator environment.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
width	float	-	Width of finger gripper : 0.0(close)~0.8(open)

- **Return**

Return Name	Data Type	Default Value	Description
success	bool	-	True or False

8. Motion-related Functions

8.1 posj(q1=0, q2=0, q3=0, q4=0, q5=0, q6=0)

- **Features**

  This function designates the joint space angle in coordinate values.

- **Parameters**

No.	Data Type	Default Value	Description
q1	float list posj	0	1-axis angle or angle list or posj
q2	float	0	2-axis angle
q3	float	0	3-axis angle
q4	float	0	4-axis angle
q5	float	0	5-axis angle
q6	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()/amovej()/movesj()/amovesj()

8.2 posx(x=0, y=0, z=0, w=0, p=0, r=0)

- **Features**

  This function designates the task space in coordinate values.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
x	float list posx	0	z position or position list or posx
y	float	0	y position
z	float	0	z position
w	float	0	w orientation (z-direction rotation of reference coordinate system)
p	float	0	p orientation (y-direction rotation of w rotated coordinate system)
r	float	0	r orientation (z-direction rotation of w and p rotated coordinate system)

- **Return**

  posx

- **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)
x3 = posx([350, 350, 450, 0, 180, 0])      #x3=posx(350, 350, 450, 0, 180, 0)
x4 = posx(x2)                              #x4=posx(400, 300, 500, 0, 180, 0)
movel(x2, v=100, a=200)
```

- **Related commands**

movel()/movec()/movejx()/amovel()/amovec()/amovejx()

8.3 trans(pos, delta, ref, ref_out)

▪ Features

- Input parameter(pos) based on the ref coordinate is translated/rotated as delta based on the same coordinate and this function returns the result that is converted to the value based on the ref_out coordinate.
- 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

Note

- The ref argument DR_WORLD is only available in M2.40 and later versions.
- The ref_out argument is only available in M2.40 and later versions.

Return

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

▪ Exception

Exception	Description

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)

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()/addto()

8.4 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()/moveb()/amoveb()

8.5 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
	list (float[6])		velocity (to an axis)

- **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_accx()/movej()/movejx()/movesj()amovej()/amovejx()/amovesj()

8.6 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
	list (float[6])		acceleration (acceleration to an axis)

- **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/sec^2).
movej(Q2)          # The joint motion acceleration to Q2 is 60(deg/sec^2) which is the global acceleration.
movej(Q1, vel=20, acc=40) # The joint motion acceleration to Q1 is 40(deg/sec^2) which is the specified
acceleration.
#2
set_accj(30.55)
set_accj([30, 40, 30, 30, 30, 10])
```

- **Related commands**

set_velj()/movej()/movejx()/movesj()/amovej()/amovejx()amovesj()

8.7 set_velx(vel1, vel2)

- **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, relatively, of TCP.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
vel1	float	-	velocity 1
vel2	float	-	velocity 2

- **Return**

Value	Description
0	Success

- **Exception**

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

- **Example**

```
<#1>
P0 = posj(0,0,90,0,90,0)
movej(P0)
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 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)       # Decimal point input is possible.
```

- **Related commands**

set_accx()/movel()/movec()/movesx()/moveb()/move_spiral()/amovel()/amovec()/ amovesx()/amoveb()/amove_spiral()

8.8 set_velx(vel)

- **Features**

  This function sets the linear velocity of the task space motion globally. The globally set velocity vel 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. The set value vel defines the linear velocity of the TCP while the rotating velocity of the TCP is determined proportionally to the linear velocity.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
vel	float	-	velocity

- **Return**

Value	Description
0	Success

- **Exception**

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

- **Example**

```
#1
p0 = posj(0,0,90,0,90,0)
movej(p0)

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)    # The global task velocity is set to 30 (mm/sec). The global task angular velocity is automatically determined.
set_accx(60)    # The global task acceleration is set to 60 (mm/sec^2). The global task angular acceleration is automatically determined.
movel(P2)       # The task motion linear velocity to P2 is 30(mm/sec) which is the global velocity.
movel(P1, vel=20, acc=40)       # The task motion linear velocity to P1 is 20(mm/sec) which is the specified velocity.
#2
set_velx(10.5)  # Decimal point input is possible.
```

- **Related commands**

set_accx()/movel()/movec()/movesx()/moveb()/move_spiral()/amovel()/amovec()/ amovesx()/amoveb()/amove_spiral()

8.9 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, relatively, of the TCP.

▪ Parameters

Parameter Name	Data Type	Default Value	Description
acc1	float	-	acceleration 1
acc2	float	-	acceleration 2

▪ Return

Value	Description
0	Success

▪ Exception

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

▪ Example

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
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/sec)
and 40(deg/sec2) which is the specified acceleration.
```

▪ Related commands

set_velx()/movel()/movec()movesx()/moveb()/move_spiral()/amovel()/amovec()/amovesx()/amoveb()/amove_spiral()

8.10 set_accx(acc)

- **Features**

  This function sets the linear acceleration of the task space motion globally. The globally set acceleration acc 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. The set value acc defines the linear acceleration of the TCP while the rotating acceleration of the TCP is determined proportionally to the linear acceleration.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
acc	float	-	acceleration

- **Return**

Value	Description
0	Success

- **Exception**

Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred

- **Example**

```
P0 = posj(0,0,90,0,90,0)
movej(P0)
P1 = posx(400,500,800,0,180,0)
P2 = posx(400,500,500,0,180,0)
movej(P0, vel=10, acc=20)
movel(P1, vel=10, acc=20)
set_velx(30)          # The global task velocity is set to 30 (mm/sec). The global task angular velocity is automatically determined.
set_accx(60)          # The global task acceleration is set to 60 (mm/sec^2). The global task angular acceleration is automatically determined.
movel(P2)             # The task motion linear acceleration to P2 is 60(mm/sec^2) which is the global acceleration.
movel(P1, vel=20, acc=40)     # The task motion linear acceleration to P1 is 40(mm/sec^2) which is the specified acceleration.
```

- **Related commands**

set_velx()/movel()/movec()movesx()/moveb()/move_spiral()/amovel()/amovec()/amovesx()/amoveb()/amove_spiral()

8.11 set_tcp(name)

- **Features**

This function calls the name of the TCP registered in the Teach Pendant and sets it as the current TCP.

- **Parameters**

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()/ikin()/movel()/movejx()/movec()/movesx()/moveb()/amovel()/amovejx()/amovec()/amovesx()/amoveb()

8.12 movej

- **Features**

The robot moves to the target joint position (pos) from the current 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 None	velocity (same to all axes) or velocity (to an axis)
acc (a)	float list (float[6])	None None	acceleration (same to all axes) or acceleration (acceleration to an axis)
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

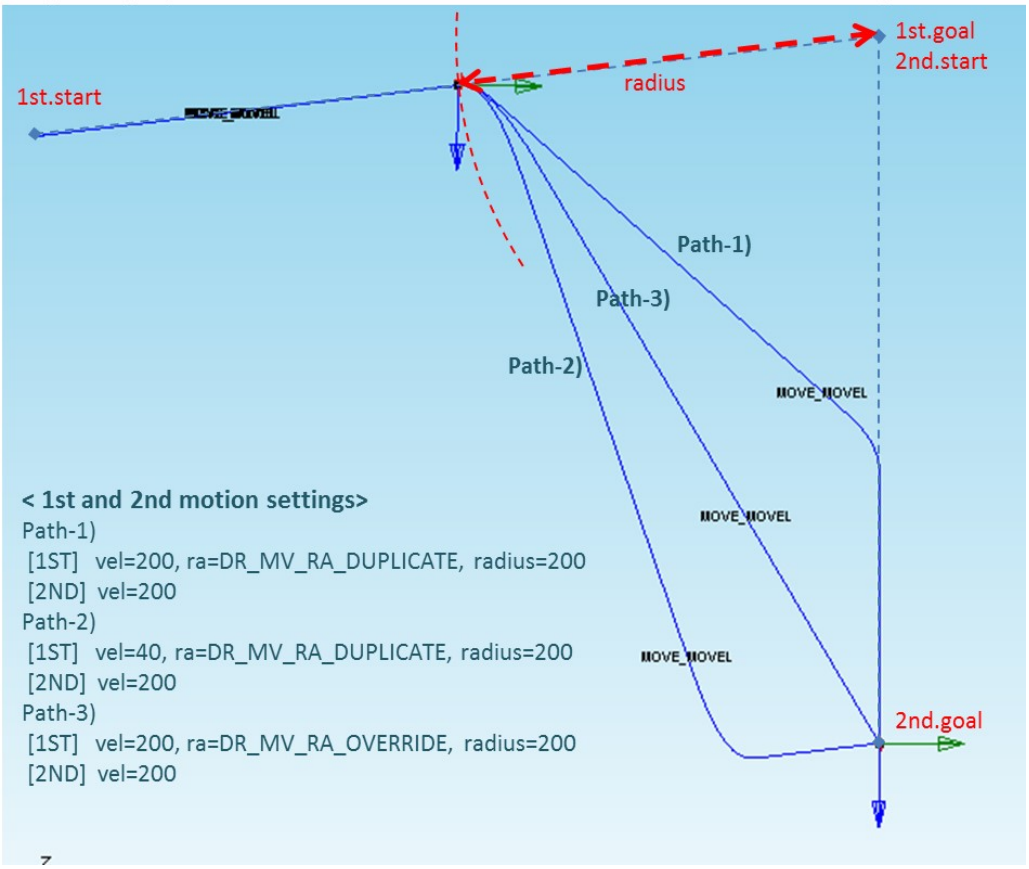
Note

- 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.

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>



- **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)
movej(Q1, vel=10, acc=20)
        # Moves to the Q1 joint angle at the velocity of 10(deg/sec) and acceleration of
        20(deg/sec2).
movej(Q2, time=5)
        # Moves to the Q2 joint angle with a reach time of 5 sec.
movej(Q1, v=30, a=60, r=200)
        # Moves to the Q1 joint angle and is set to execute the next motion
        # when the distance from the Q1 space position is 200mm.
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.
```

- **Related commands**

posj()/set_velj()/set_accj()/amovej()

8.13 movel

▪ 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 list (float[6])	-	posx or position list
vel (v)	float list (float[2])	None None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None 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.
radius (r)	float	None	Radius for blending
ref	int	None	reference coordinate ・ DR_BASE: base 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

Note

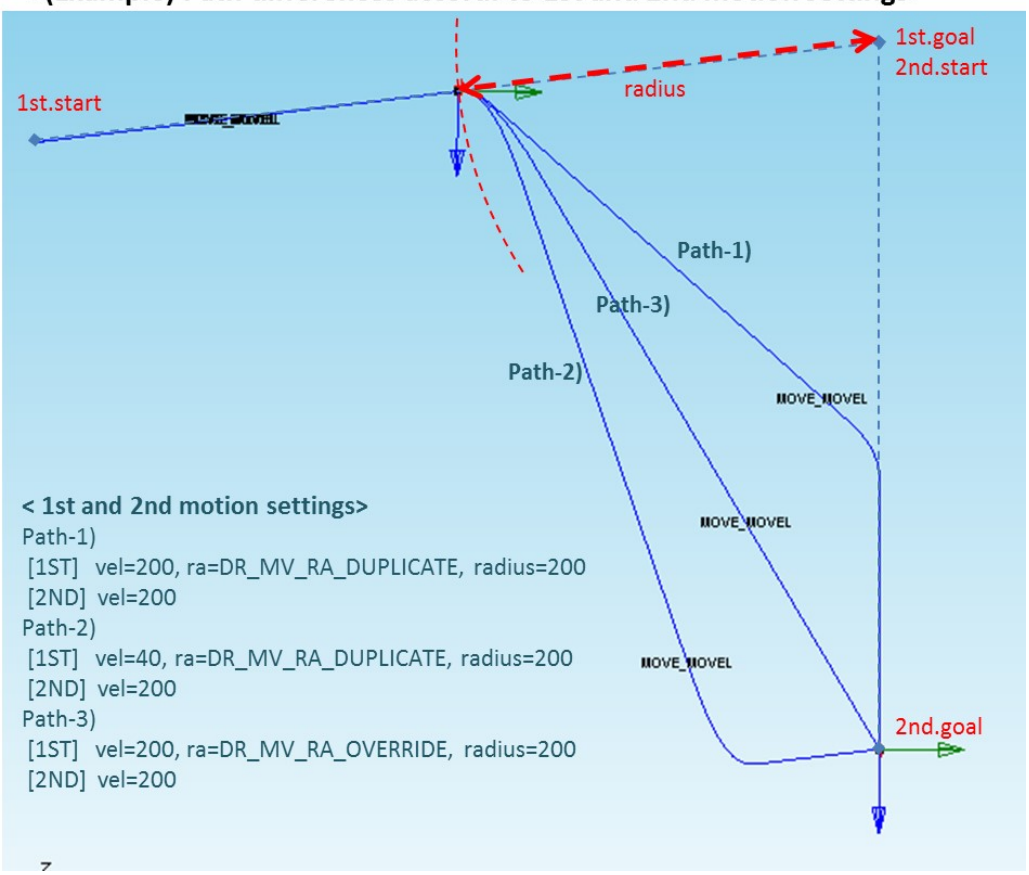
- Abbreviated parameter names are supported. (v:vel, a:acc, t:time, r:radius)
- _global_velx is applied if vel is None. (The initial value of _global_velx is 0.0 and can be set by set_velx.)
- _global_accx is applied if acc is None. (The initial value of _global_accx is 0.0 and can be set by set_accx.)

- 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.)

⚠ 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>



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()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/amovel()

8.14 movejx

▪ 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).

▪ Parameters

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

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.

▪ 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

▪ 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/sec^2), 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.
```

Related commands

posx()/set_velj()/set_accj()/get_current_posx()/amovejx()

8.15 movec

▪ 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 list (float[6])	-	posx or position list
pos2	posx list (float[6]		posx or position list
vel (v)	float list (float[2])	None None	velocity or velocity1, velocity2
acc (a)	float list (float[2])	None None	acceleration or acceleration1, acceleration2
time (t)	float	None	Reach time [sec]
radius (r)	float	None	Radius for blending
ref	int	None	reference coordinate · DR_BASE: base 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

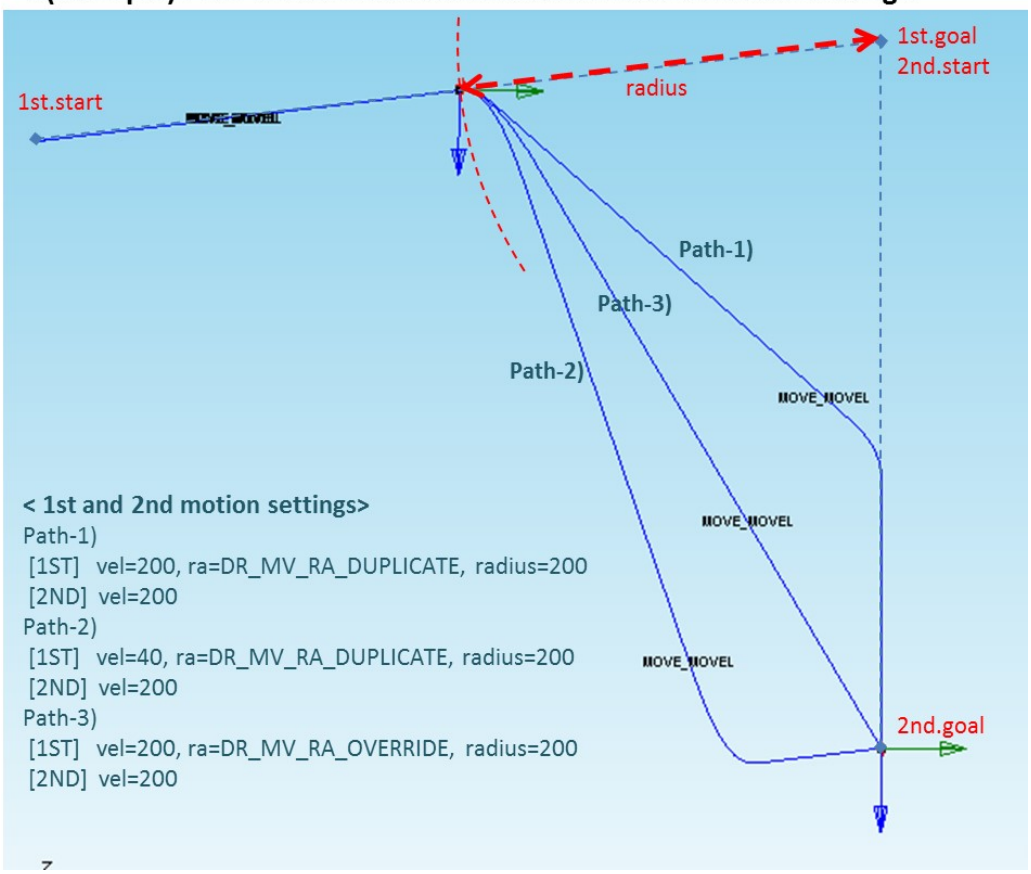
Note

- 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 inputted, the total rotated angle on the circular path is applied to the angle.
- 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.

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>



- **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/sec^2) and 40(deg/sec^2).

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/sec^2)
# 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/sec^2).
# 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/sec^2).
# 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()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/amovec()

8.16 movesj

▪ 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 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**

```
#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/sec^2)
```

```
#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()/set_velj()/set_accj()/amovesj()

8.17 movesx

▪ 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_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/sec^2) and
        # 60(deg/sec^2).
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/sec^2), and 60(deg/sec^2) (same motion
        as CASE-1).
```

- **Related commands**

posx()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/amovesx()

8.18 moveb

- **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_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

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.

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

Exception	Description
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()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/amoveb()

8.19 move_spiral

▪ Features

The radius increases in a radial direction and the robot moves in parallel with the rotating spiral motion in an axial direction. It moves the robot along the spiral trajectory on the surface that is perpendicular to the axis on the coordinate specified as ref and the linear trajectory in the axis direction.

▪ Parameters

Parameter Name	Data Type	Default Value	Range	Description
rev	float	10	rev > 0	Total number of revolutions
rmax	float	10	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_TOOL : tool coordinate ・ user coordinate: user defined

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.
- This function does not support online blending of previous and subsequent motions.

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 time value.

▪ Return

Value	Description
0	Success
Negative value	Error

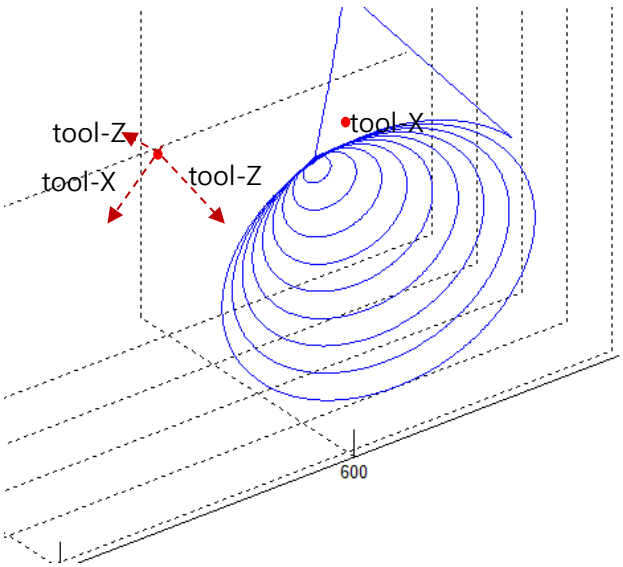
▪ Exception

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**

```
# hole search
# (A motion that completes 9.5 revolutions (rev) to the 50 mm radius (rmax) from 0 on the Tool-X/Y surface as the center of the rotation in the Tool-Z direction and the spiral trajectory that moves 50 mm (lmax) in the Tool-Z direction at the same time in 10 seconds from the initial position)

J00 = posj(0,0,90,0,60,0)
movej(J00,vel=30,acc=30)          # Joint movement to the initial pose
move_spiral(rev=9.5,rmax=20.0,lmax=50.0,time=20.0,axis=DR_AXIS_Z,ref=DR_TOOL)
```



- **Related commands**

set_velx()/set_accx()/set_tcp()/set_ref_coord()/amove_spiral()

8.20 move_periodic

▪ 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_TOOL : tool coordinate ・ user coordinate: user defined

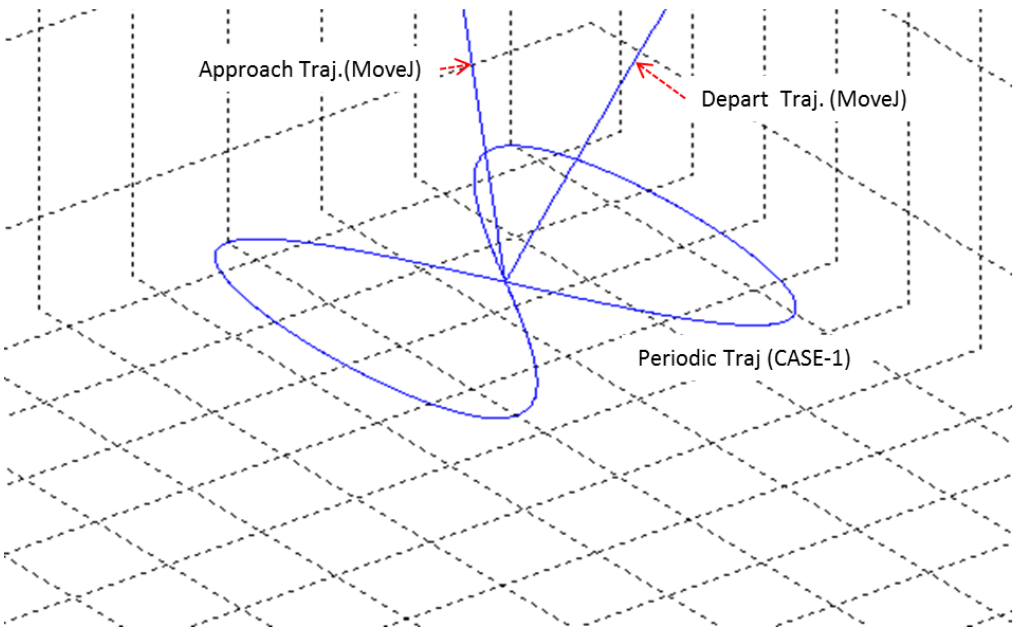
Note

- 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.

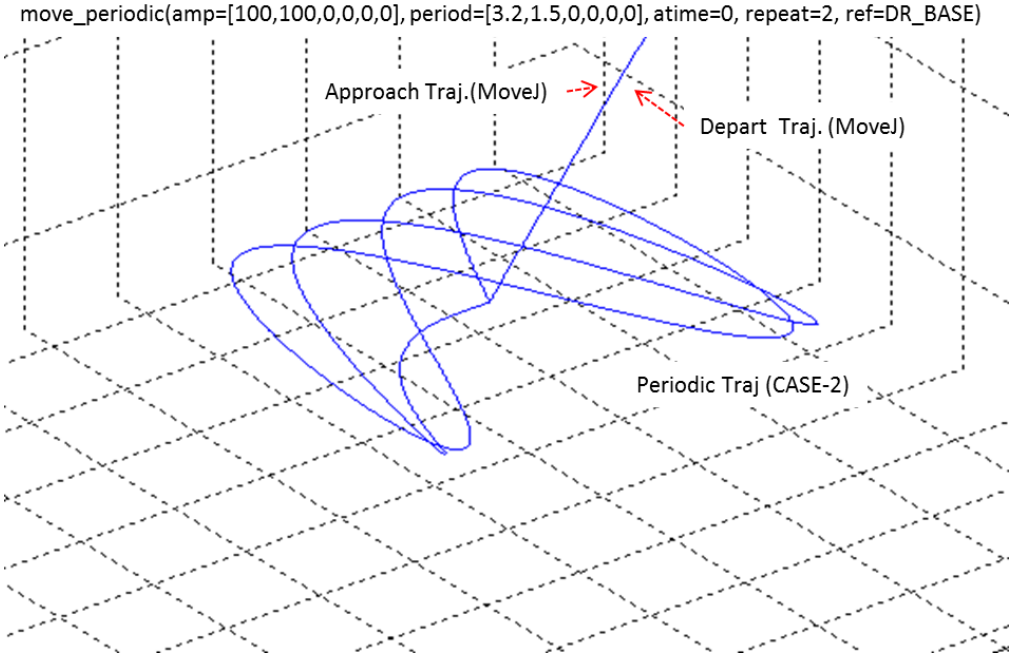
CASE-1) All-axis motions end at the same time

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)



CASE-2) Diff-axis motions end individually

move_periodic(amp=[100,100,0,0,0,0], period=[3.2,1.5,0,0,0,0], atime=0, 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	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()/amove_periodic()

8.21 amovej

▪ 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.

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

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.

▪ 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**

posj()/set_velj()/set_accj()/mwait()/movej()

8.22 amovel

▪ 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_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

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.

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()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/mwait()movel()

8.23 amovejx

▪ 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 list (float[6])	-	posx or position list
vel (v)	float list (float[6])	None None	velocity (same to all axes) or velocity (to an axis)
acc (a)	float list (float[6])	None None	acceleration (same to all axes) or acceleration (acceleration to an axis)
time (t)	float	None	Reach time [sec]
ref	int	None	reference coordinate ・ DR_BASE: base 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

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 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.

⚠ Caution

The blending into current active motion is disabled in the case of input with relative motion (mod=DR_MV_MOD_REL), and it is recommended to blend using 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)
amovejx (x1, vel=100, acc=200, sol=1)      # 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)                                   # Temporarily suspends the program execution until the motion is terminated.
```

- **Related commands**

posx()/set_velj()/set_accj()/get_current_posx()/mwait()/movejx()

8.24 amovec

▪ 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_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

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 inputted, the total rotated angle on the circular path is applied to the angle.
- 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.

▪ 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**

```
#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()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/mwait()/movec()

8.25 amovesj

▪ 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()/set_velj()/set_accj()/mwait()/amovesj()

8.26 amovesx

▪ 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 amovesj() 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_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/sec^2) and
        # 60(deg/sec^2). 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()set_velx()/set_accx()/set_tcp()/set_ref_coord()/mwait()/movesx()

8.27 amoveb

▪ 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_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

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.

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

Exception	Description
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()/set_velx()/set_accx()/set_tcp()/set_ref_coord()/mwait()/moveb()

8.28 amove_spiral

▪ Features

The asynchronous move_spiral motion operates in the same way as move_spiral() 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_spiral() motion results in an error for safety reasons. Therefore, the termination of the amove_spiral() motion must be confirmed using mwait() or check_motion() between amove_spiral() and the following motion command.

The radius increases in a radial direction and the robot moves in parallel with the rotating spiral motion in an axial direction. It moves the robot along the spiral trajectory on the surface that is perpendicular to the axis on the coordinate specified as ref and the linear trajectory in the axis direction.

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	10	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_TOOL : tool coordinate ・ user coordinate: user defined

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.
- This function does not support online blending of previous and subsequent motions.

⚠ 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 time value.

▪ Return

Value	Description
0	Success
Negative value	Error

▪ Exception

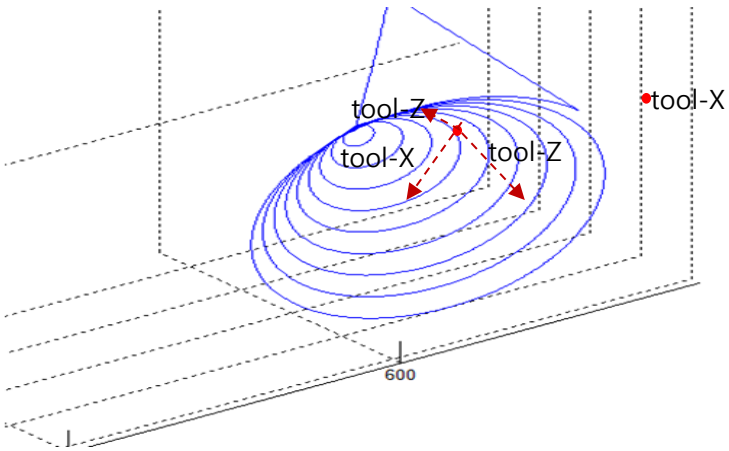
Exception	Description
DR_Error (DR_ERROR_TYPE)	Parameter data type error occurred
DR_Error (DR_ERROR_VALUE)	Parameter value is invalid
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred
DR_Error (DR_ERROR_STOP)	Program terminated forcefully

▪ Example

```
## hole search
# (A motion that completes 9.5 revolutions (rev) to the 30 mm radius (rmax) from 0 on the Tool-X/Y surface as the center of the rotation in the Tool-Z direction
# and the spiral trajectory that moves 50 mm (lmax) in the Tool-Z direction at the same time in 20 seconds
```

```
# from the initial position.
# D-Out (no. 1 channel) 3 seconds after the motion begins.)

J00 = posj(0,0,90,0,60,0)
movel(J00, vel=30, acc=30)          # Joint moves to the beginning pose
amove_spiral(rev=9.5,rmax=50.0,lmax=50.0,time=10.0,axis=DR_AXIS_Z,ref=DR_TOOL)
wait(3)
set_digital_output(1, 1)            # D-Out (no. 1 channel) ON
mwait(0)                            # Waits until the motion stops.
```



- **Related commands**

set_velx()/set_accx()/set_tcp()/set_ref_coord()/mwait()/move_spiral()

8.29 amove_periodic

▪ 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: The next command is executed after the robot starts from the current position and reaches (stops at) the end point of the periodic trajectory.
- amove_ periodic: 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 periodic trajectory.

▪ Parameters

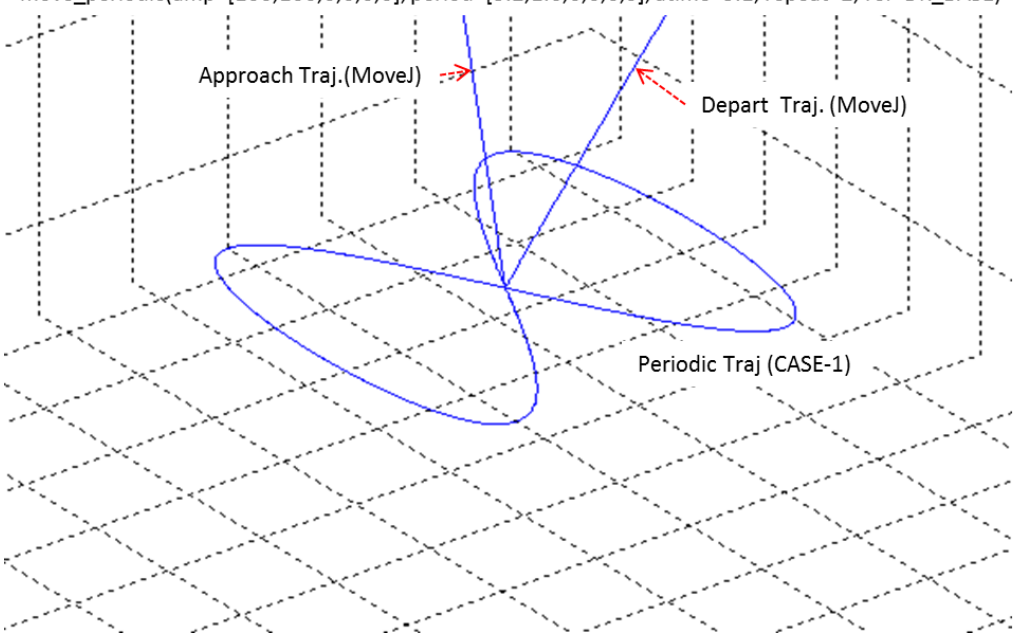
Parameter Name	Data Type	Default Value	Range	Description
amp	list (float[6])	-	0≤amp	Amplitude (motion between -amp and +amp) [mm] or [deg]
period	float or list (float[6])		0≤period	Period (time for 1 cycle) [sec]
atime	float	0.0	0≤atime	Acc-, dec- time [sec]
repeat	int	1	> 0	Repetition count
ref	int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

8.30 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()amovej()/amovel()/amovejx()/amovec()/amovesj()/amovesx()/amoveb()/ amove_spiral()/amove_periodic()

8.31 stop(st_mode)

▪ Features

This function stops the currently active motion. This function stops differently according to the st_mode received as an argument. All stop modes except Estop stop the motion in the currently active section.

▪ Parameters

Parameter Name	Data Type	Default Value	Description
st_mode	int	-	stop mode · DR_QSTOP_STO: Stop Category 1 · DR_QSTOP: Stop Category 2 · DR_SSTO: Stop Category 2 · DR_HOLD: emergency 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()/movel()/movejx()/movec()/movesj()/movesx()/moveb()/move_spiral()
/move_periodic()/amovej()/amovel()/amovejx()/amovec()/amovesj()/amovesx()/amoveb
()/amove_spiral()/amove_periodic()

8.32 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(1~100)

- **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()/movel()/movejx()/movec()/movesj()/movesx()/moveb()/move_spiral()/move_periodic()/amovej()/amovel()/amovejx()/amovec()/amovesj()/amovesx()/amoveb()/amove_spiral()/amove_periodic()

9. Auxiliary Control Functions

9.1 get_current_posj()

- **Features**

This function returns the current joint angle.

- **Parameters**

none

- **Return**

Value	Description
posj	Joint angle

- **Exception**

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

- **Example**

```
q1 = get_current_posj()
```

9.2 get_current_posx()

- **Features**

This e returns the pose and solution space of the current coordinate system. The pose is based on the base coordinate.

- **Parameters**

none

- **Return**

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

- **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
```

9.3 get_current_solution_space()

- **Features**

This function returns the current solution space value.

- **Parameters**

Not applicable

- **Return**

Value	Description
int	Solution space (0 ~ 7)

- **Exception**

Exception	Description
DR_Error (DR_ERROR_RUNTIME)	C extension module error occurred

9.4 get_joint_torque()

- **Features**

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

- **Parameters**

Not applicable

- **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()/get_tool_force()

9.5 get_external_torque()

- **Features**

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

- **Parameters**

Not applicable

- **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()/get_tool_force()

9.6 get_tool_force()

- **Features**

This function returns the external force applied to the current tool. The force is based on the base coordinate while the moment is based on the tool coordinate.

- **Parameters**

Not applicable

- **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()
```

- **Related commands**

get_joint_torque()/get_external_torque()

10. Other Settings and Safety-related Functions

10.1 set_tool(name)

- **Features**

This function retrieves the tool data registered in the Teach Pendant by name.

- **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 ("tool1")   # Calls the "tool1" data registered in the TP and sets it as the tool.
```

- **Related commands**

set_tcp()

11. System Functions

11.1 Robot Mode

11.1.1 set_robot_mode(robot_mode)

- **Features**

This function is to set the operation mode (manual / automatic) of the robot controller. Motion command is available in both modes, but manual handguiding is available in manual mode and operates in deceleration mode for safety.

In automatic mode, manual handguiding is not possible and operates in normal speed mode.

Manual mode: robot LED lights up blue

Auto mode: robot LED lights up in white

- **Parameters**

Parameter Name	Data Type	Default Value	Description
Robot_mode	int	-	• ROBOT_MODE_MANUAL : 0 • ROBOT_MODE_AUTONOMOUS : 1

- **Return**

Value	Description
0	Success
Negative value	Failed

- **Example**

```
#...
set_robot_mode(ROBOT_MODE_MANUAL)
#...
#...
#...
set_robot_mode(ROBOT_MODE_AUTONOMOUS)
...
```

11.1.2 get_robot_mode()

- **Features**

This function reads the current operating mode of the robot controller.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
none	-	-	-

- **Return**

Value	Description
ROBOT_MODE_MANUAL (0)	Robot mode is manual.
ROBOT_MODE_AUTONOMOUS (1)	Robot mode is auto.

- **Example**

```
#...
if get_robot_mode() != ROBOT_MODE_AUTONOMOUS:
    set_robot_mode(ROBOT_MODE_AUTONOMOUS)
#...
```

11.2 IO

11.2.1 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, 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)                    # 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.)
```

11.2.2 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
```

11.2.3 set_tool_digital_output(index, val=None)

▪ Features

This function sends 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 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, 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
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.)
```

11.2.4 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
```

11.2.5 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)
```

11.2.6 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)
```

11.2.7 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
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
```

11.2.8 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.
```

11.3 Modbus

11.3.1 add_modbus_signal (ip, port, name, reg_type, index, value=0)

- **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_MODBUS_DIG_OUTPUT or DR_MODBUS_REG_OUTPUT (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

Note

- The slaveid argument is only available in M2.40 and later versions.

- **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 192.168.127.254, 8 input points: “di1” - ”di8”, 8 output points: “do1” - "do8”
#Modbus IO 2: IP 192.168.127.253, 8 input points: “di9” - "di16”, 8 output points: “do9” - "do16”

# set <modbus 1> input : di1~di8
add_modbus_signal(ip="192.168.127.254",port=502, name="di1",
reg_type=DR_MODBUS_REG_INPUT, index=0)
add_modbus_signal(ip="192.168.127.254",port=502, name="di2",
reg_type=DR_MODBUS_REG_INPUT, index=1)
add_modbus_signal(ip="192.168.127.254",port=502, name="di3",
reg_type=DR_MODBUS_REG_INPUT, index=2)
add_modbus_signal(ip="192.168.127.254",port=502, name="di4",
reg_type=DR_MODBUS_REG_INPUT, index=3)
add_modbus_signal(ip="192.168.127.254",port=502, name="di5",
reg_type=DR_MODBUS_REG_INPUT, index=4)
add_modbus_signal(ip="192.168.127.254",port=502, name="di6",
reg_type=DR_MODBUS_REG_INPUT, index=5)
add_modbus_signal(ip="192.168.127.254",port=502, name="di7",
reg_type=DR_MODBUS_REG_INPUT, index=6)
add_modbus_signal(ip="192.168.127.254",port=502, name="di8",
reg_type=DR_MODBUS_REG_INPUT, index=7)

# set <modbus 1> output : do1~do8
add_modbus_signal(ip="192.168.127.254",port=502, name="do1",
reg_type=DR_MODBUS_REG_OUTPUT, index=0, value=0)
add_modbus_signal(ip="192.168.127.254",port=502, name="do2",
reg_type=DR_MODBUS_REG_OUTPUT, index=1, value=0)
add_modbus_signal(ip="192.168.127.254",port=502, name="do3",
reg_type=DR_MODBUS_REG_OUTPUT, index=2, value=0)
add_modbus_signal(ip="192.168.127.254",port=502, name="do4",
reg_type=DR_MODBUS_REG_OUTPUT, index=3, value=0)
add_modbus_signal(ip="192.168.127.254",port=502, name="do5",
reg_type=DR_MODBUS_REG_OUTPUT, index=4, value=0)
add_modbus_signal(ip="192.168.127.254",port=502, name="do6",
reg_type=DR_MODBUS_REG_OUTPUT, index=5, value=0)
add_modbus_signal(ip="192.168.127.254",port=502, name="do7",
reg_type=DR_MODBUS_REG_OUTPUT, index=6, value=0)
```

```
add_modbus_signal(ip="192.168.127.254",port=502, name="do8",
reg_type=DR_MODBUS_REG_OUTPUT, index=7, value=0)

#=============================================================
# set <modbus 2> input : di9~di16
add_modbus_signal(ip="192.168.127.253",port=502, name="di9",
reg_type=DR_MODBUS_REG_INPUT, index=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="di10",
reg_type=DR_MODBUS_REG_INPUT, index=1)
add_modbus_signal(ip="192.168.127.253",port=502, name="di11",
reg_type=DR_MODBUS_REG_INPUT, index=2)
add_modbus_signal(ip="192.168.127.253",port=502, name="di12",
reg_type=DR_MODBUS_REG_INPUT, index=3)
add_modbus_signal(ip="192.168.127.253",port=502, name="di13",
reg_type=DR_MODBUS_REG_INPUT, index=4)
add_modbus_signal(ip="192.168.127.253",port=502, name="di14",
reg_type=DR_MODBUS_REG_INPUT, index=5)
add_modbus_signal(ip="192.168.127.253",port=502, name="di15",
reg_type=DR_MODBUS_REG_INPUT, index=6)
add_modbus_signal(ip="192.168.127.253",port=502, name="di16",
reg_type=DR_MODBUS_REG_INPUT, index=7)

# set <modbus 2> output : do9~do16
add_modbus_signal(ip="192.168.127.253",port=502, name="do9",
reg_type=DR_MODBUS_REG_OUTPUT, index=0, value=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="do10",
reg_type=DR_MODBUS_REG_OUTPUT, index=1, value=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="do11",
reg_type=DR_MODBUS_REG_OUTPUT, index=2, value=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="do12",
reg_type=DR_MODBUS_REG_OUTPUT, index=3, value=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="do13",
reg_type=DR_MODBUS_REG_OUTPUT, index=4, value=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="do14",
reg_type=DR_MODBUS_REG_OUTPUT, index=5, value=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="do15",
reg_type=DR_MODBUS_REG_OUTPUT, index=6, value=0)
add_modbus_signal(ip="192.168.127.253",port=502, name="do16",
reg_type=DR_MODBUS_REG_OUTPUT, index=7, value=0)
```

11.3.2 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 “di1” and
"do1”
# and this signal registration is to be deleted. .
del_modbus_signal("di1")          # Deletes the registered “di1” contact
del_modbus_signal("do1")          # Deletes the registered “do1” contact
```

11.3.3 set_modbus_output(iobus, val)

- **Features**

This function sends the signal to an external Modbus system.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
iobus	string	-	Modbus name (set in the TP)
value	int	-	Modbus digital I/O • ON : 1 • OFF : 0
			Value for Modbus analog I/O

- **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 is connected, and the signals are registered as “do1” and “do2”.
set_modbus_output("do1", ON)
set_modbus_output("do2", OFF)

#Modbus analog I/O is connected, and the signals are registered as “reg1” and “reg2”.
set_modbus_output("reg1", 10)
set_modbus_output("reg2", 24)
```

11.3.4 get_modbus_input(iobus)

- **Features**

This function reads the signal from the Modbus system.

- **Parameters**

Parameter Name	Data Type	Default Value	Description
iobus	string	-	Modbus name (set in the TP)

- **Return**

Value	Description
0 or 1	ON or Off in the case of the Modbus digital I/O
value	The register value in the case of the Modbus analog 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**

```
#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")
```

Doosan Robotics

www.doosanrobotics.com